

**Intel Corporation, Cavium, Inc., Wistron
Corporation, and Dell Inc.**

v.

Alacritech, Inc.

Case Nos.: IPR2017-01309, IPR2017-01718, IPR2018-00327, and
IPR2018-00371

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

Case Nos.: IPR2017-01406, IPR2017-01707, IPR2018-00329, and
IPR2018-00375


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

Patent Owner's Demonstratives

Hearing: September 13, 2018

-01391/-01406 PO Demonstrative, 1

Overview of the '036 Patent



US007237036B2

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

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

(75) Inventors: **Laurence B. Boucher**, San Jose, CA (US); **Stephen E. J. Blight**, San Jose, CA (US); **Peter K. C. Francisco**, CA (US); **David Saratoga**, CA (US); **Clive 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. 10/092,277, 1998, provisional application No. filed on Oct. 14, 1997.

(60) Provisional application No. 60/098,296, filed on Oct. 14, 1997.

(51) **Int. Cl.**
G06F 15/38 (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.

22 Claims, 89 Drawing Sheets

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

Ex. 1001.001 ('036 Patent)

tative control instructions for a given message that allow data from the message to be processed via a fast-path which accesses message data directly at its source or delivers it directly to its intended destination. This fast-path bypasses

Ex. 1001.096 ('036 Patent)

'036 Patent (IPR2017-01391 Ex. 1001)

-01391/-01406 PO Demonstrative, 2

Challenged Claims of the '036 Patent (Claims 1-7)

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. 1001.142-43 ('036 Patent)

Challenged Claims of the '072 Patent (Claims 1-21)

1. A method comprising:
establishing, at a host computer, a transport layer connection, including creating a context that includes protocol header information for the connection;
transferring the protocol header information to an interface device;
transferring data from the network host to the interface device, after transferring the protocol header information to the interface device;
dividing, by the interface device, the data into segments;
creating headers for the segments, by the interface device,
from a template header containing the protocol header information; and
prepending the headers to the segments to form transmit packets.

Ex. 1001.142 ('072 Patent)

Challenged Claims of the '072 Patent (Claims 1-21)

9. A method comprising:
creating, at a computer, a context including protocol information and status information for a network connection, the protocol information providing a template header for the network connection;
transferring the protocol information and status information to an interface device;
transferring data from the computer to the interface device, after transferring the protocol information and status information to the interface device;
dividing, by the interface device, the data into segments;
creating headers for the segments, by the interface device,
from the template header;
prepending the headers to the segments to form packets;
and
transmitting the packets on a network.

Ex. 1001.142 ('072 Patent)

Challenged Claims of the '072 Patent (Claims 1-21)

15. A method comprising:
establishing, at a computer, a Transmission Control Protocol (TCP) connection corresponding to a context that includes status information and Internet Protocol (IP) addresses and TCP ports for the connection;
transferring the context to an interface device;
transferring data from the network host to the interface device;
dividing, by the interface device, the data into segments;
creating headers for the segments, by the interface device,
from a template header that includes the IP addresses and TCP ports; and
prepending the headers to the segments to form transmit packets.

Ex. 1001.142 ('072 Patent)

Overview of Instituted Grounds

'036 Patent

Challenged Claims	103(a) References
1-7	Erickson and Tanenbaum

'072 Patent

Challenged Claims	103(a) References
1-21	Erickson and Tanenbaum

Overview of Erickson

United States Patent Erickson et al.

[19]

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

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

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 B. 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.⁵ **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

4,389,063	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	Reed et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Nocross et al.	395/280
5,642,481	6/1997	Polizetti	395/185.01
5,671,442	9/1997	Fooney 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 Kernel 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 PIEL—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).

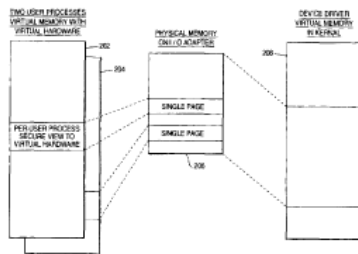
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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

[57] **ABSTRACT**

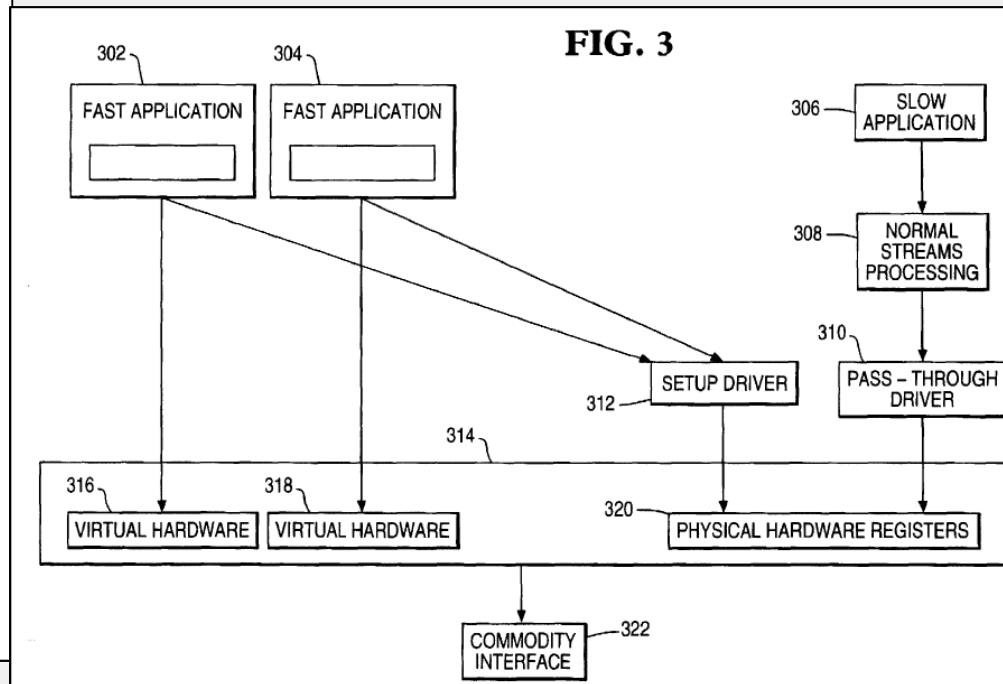
A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device.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



Erickson (Ex. 1005)

[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**



Ex. 1005.004

Overview of Erickson

US00576861

United States Patent [19] (11) Patent Number:
Erickson et al. [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. Handley**, 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.⁵ **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

4,389,063	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	Reed et al.	395/830
5,548,778	8/1996	Hirayama et al.	395/823
5,553,244	9/1996	Nocross et al.	395/280
5,642,481	6/1997	Polizetti et al.	395/185.01
5,671,442	9/1997	Fossey 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).

ABSTRACT

A method of controlling an input device connected to a computer to facilitate address space for the I/O device memory of the computer, wherein physical virtual registers that are used by the I/O device are mapped into the virtual address space of the computer. In essence, control registers in the virtual address space of the computer are mapped into the address space of the I/O device. The sequence of actions can be triggered by programming specified values into the mapped virtual address space.


19 Claims, 7 Drawings

Within the `udpscript` procedure described above, the `nextid()` function provides a monotonically increasing 16-bit counter required by the IP protocol. The `ipchecksum()` function performs a checksum for the IP 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 likelihood, the adapter would have a very limited knowledge of the user process' virtual address space, probably only knowing how to map virtual-to-physical for a very limited range, maybe as small as a single page. Pages in the user process' virtual address space for such buffers would need to be fixed. The `udpscript` procedure would need to be enhanced if the user data were allowed to span page boundaries. The `udpchecksum()` procedure generates a checksum value for both the UDP Header 708 plus the user data (not shown).

Ex. 1005 at 8:20-24

Erickson (Ex. 1005)

Overview of Erickson



US005768618A

United States Patent [19] [11] Patent Number: 5,768,618
 Erickson et al. [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. Handley, 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.⁵ 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	Nooncross et al.	395/280
5,640,481	6/1997	Pedersen	395/185.01
5,671,442	9/1997	Fossey et al.	395/834

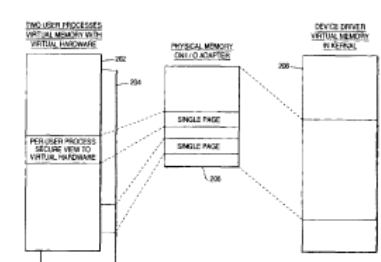
FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off.

OTHER PUBLICATIONS

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

19 Claims, 7 Drawing Sheets



Erickson (Ex. 1005)

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

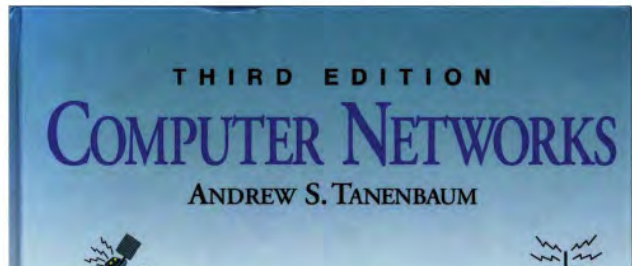
An example of programming that triggers the I/O device adapter is provided below:

```

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

Ex. 1005.012

Overview of Tanenbaum



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).



INTEL Ex.1006.001

Ex. 1006.583

Tanenbaum (Ex. 1006)

Overview of Tanenbaum



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

Ex. 1006.498



Tanenbaum (Ex. 1006)

'036 Patent: the Combination Fails to Suggest "the TCP State Information Is Updated by Said Second Processor"

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. 1001.142-43 ('036 Patent)

'036 Patent: the Combination Fails to Suggest "the TCP State Information Is Updated by Said Second Processor"



Robert Horst
Petitioner's Expert

14 Q. But when the total length field, the datagram
15 ID field, the IP checksum field, the UDP checksum field
16 are modified or initialized, that modification is done
17 on the I/O device, right?
18 A. Right. This script is run on the I/O device.
19 So those changes to the header are done on the I/O
20 device.

Ex. 2028 (Horst Dep. Tr.) at 125:14-20

'036 Patent: the Combination Fails to Suggest "the TCP State Information Is Updated by Said Second Processor"

112. Moreover, the claimed updating is performed when the data is transferred to the first apparatus memory, *i.e.*, on the *receiving side* of the system. At best, therefore, the combination of *Erickson* and *Tanenbaum* would disclose a system that updates a header template on transmit side by the network interface device and updated a connection record on the receive side by the host CPU. The combination does not show or suggest the claim approach, whereby the processor of the network interface device updates TCP state information created or maintained by the host CPU.

Ex. 2026 at 53

UNITED STATES PATENT AND TRADEMARK

BEFORE THE PATENT TRIAL AND APPEAL

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01406¹
U.S. Patent 7,673,072

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01707, has been joined as a petitioner in this proceeding.

Almeroth Decl. (Ex. 2026)

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

dividing, by the interface device, the data into segments; creating headers for the segments, by the interface device, from a template header containing the protocol header information; and prepending the headers to the segments to form transmit packets.

Ex. 1001.142 ('072 Patent), Claim 1

dividing, by the interface device, the data into segments; creating headers for the segments, by the interface device, from the template header; prepending the headers to the segments to form packets; and transmitting the packets on a network.

Ex. 1001.142 ('072 Patent), Claim 9

dividing, by the interface device, the data into segments; creating headers for the segments, by the interface device, from a template header that includes the IP addresses and TCP ports; and prepending the headers to the segments to form transmit packets.

Ex. 1001.142 ('072 Patent), Claim 15

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

US005768618A

United States Patent [19] Patent Number: 5,768,612
 Erickson et al. [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. Handley, 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.⁵ 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	Nozcross et al.	395/280
5,640,481	6/1997	Pedersen	395/185.01
5,671,442	9/1997	Fossey et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off.

OTHER PUBLICATIONS

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

19 Claims, 7 Drawing Sheets

Erickson (Ex. 1005)

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

An example of programming that triggers the I/O device adapter is provided below:

```

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

Ex. 1005.012

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORPORATION
Petitioner

v.

ALACRITECH, INC.
Patent Owner

Case IPR No. Unassigned
U.S. Patent No. 7,673,072
Title: FAST-PATH APPARATUS FOR RECEIVING DATA
A TCP CONNECTION

Declaration of Robert Horst, Ph.D. in Support of
Petition for *Inter Partes* Review
of U.S. Patent No. 7,673,072

INTEL Ex.1003.001

Erickson in view of Tanenbaum96

[1.4] dividing, by the interface device, the data into segments;

The "*dividing, by the interface device, the data into segments*" limitation is met by the foregoing obvious TCP script for Erickson. Erickson's senduserdatagram (Ex.1005, Erickson at 7:23-32) user process triggers the I/O adapter to send each segment, by "spanking" the GO register. The adapter calculates the TCP checksum, transmits the packet, updates the shared state in Hardware Register 504, and waits for GO to be set again for the next segment. Repeated invocations of the TCP script (by repeatedly spanking the GO register) with pointers to data for consecutive segments result in the interface device dividing the user data stream into TCP segments.

Ex. 1003 at A-12

Horst Decl. (Ex. 1003)

-01391/-01406 PO Demonstrative, 19

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION
Petitioner

v.

ALACRITECH, INC.
Patent Owner

Case IPR No. Unassigned
U.S. Patent No. 7,673,072
Title: FAST-PATH APPARATUS FOR RECEIVING DATA OVER
A TCP CONNECTION

Declaration of Robert Horst, Ph.D. in Support of
Petition for *Inter Partes* Review
of U.S. Patent No. 7,673,072

Erickson in view of Tanenbaum96

[1.4] dividing, by the interface device, the data into segments;

met by a second obvious TCP script for Erickson. This alternative script builds on the single segment script, but requires only one "spank" of the GO register for a multi-segment send. As Erickson assumes, this script would sequentially transfer an amount of data that does not exceed the limited range the adapter is able to map, possibly a single page:

"In all likelihood the adapter would have a very limited knowledge of the user process' virtual address space, probably only knowing how to map virtual-to-physical for a very limited range. maybe as small as a single page." (Ex.1005, Erickson at 8:22-24).

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

The "*dividing the data into multiple segments*" limitation is also met by a third obvious TCP script for Erickson. Like the second alternative script, this third alternative script builds on the single segment script, but requires only one "spank" of the GO register for a multi-segment send. Instead of transferring one MSS-sized segment of user data at a time, the adapter would DMA all of the user data identified by the user data address pointer and length in one large transfer. The adapter would then repeatedly extract one segment of data at a time from the transferred block, encapsulate it in a packet, and transmit.

INTEL Ex.1003.001

Ex. 1003 at A-13

Horst Decl. (Ex. 1003)

-01391/-01406 PO Demonstrative, 20

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)



Robert Horst
Petitioner's Expert

16 Q. You also state at the bottom of A-78, you say
17 that "It would also be obvious to create the following
18 TCP script for Ericsson," and you go on to page A-77 --
19 sorry. That's A-39 of A-78, now on A-40 of A-78. You
20 referred to this as an alternative script. Do you see
21 that?

22 A. Yes.

23 Q. Okay. The alternative script is not disclosed
24 in Ericsson itself, though, right?

2 A. The TCP scripts are not disclosed in Ericsson.

3 Q. (By Mr. Mack) The TCP script is not disclosed;
4 but this alternative that you're describing at the top
5 of page A-40 is also not disclosed, right?

6 MR. CONSTANT: Objection to form.

7 A. The script itself was not disclosed, but the
8 combination of Ericsson and Tanenbaum would make a
9 script like this obvious.

10 Q. (By Mr. Mack) And then beginning in the next
11 paragraph you refer to a third TCP script. Do you see
12 that?

13 A. Yes.

14 Q. That's, yet, another script that's not
15 disclosed in Ericsson, right?

16 MR. CONSTANT: Objection, form.

17 A. Yes.

Ex. 2028 (Horst Dep. Tr.) at 98:16-99:17

-01391/-01406 PO Demonstrative, 21

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)



Robert Horst
Petitioner's Expert

14 Q. So after the GO register is spanked,
15 if there are multiple segments to be transmitted
16 the adapter would access the I/O bus for multiple
17 times to do the DMA; is that correct?

18 MR. CONSTANT: Objection. Form.

19 A. The -- each individual segment would
20 require a DMA transfer to get that data to the
21 adapter, **yes.**

22 Q. And in order to do a DMA transfer,
23 the adapter need to access the I/O bus; is that
24 correct?

25 A. **Yes,** that's the path through which
2 the data gets from the host to the adapter.

Ex. 2029 (Horst Dep. Tr.) at 99:19-100:2

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)



Robert Horst
Petitioner's Expert

21 Q. And if adding up all this access is
22 to I/O bus, can you tell if the third
23 transcript -- if the third script is faster than
24 the slow path?
25 A. Again, without knowing the details,
2 there's no way to know which one is faster.

Ex. 2029 (Horst Dep. Tr.) at 97:21-98:2

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

US000768618A

United States Patent (19) **Patent Number:** 5,768
Erickson et al. **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 ENDOPTED SUB PORTIONS OF ADDRESS SPACE**

[75] **Inventors:** Greg R. Erickson; Douglas E. Handley, both of Poway, P. Keith Muller, Curtis B. 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.:** G06F 15/02
[52] **U.S. Cl.:** 395/829
[58] **Field of Search:** 395/829, 832, 846, 882, 284, 309, 503, 473

References Cited
U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,289	10/1988	Bosmer et al.	395/823
5,016,161	5/1991	Van Lee et al.	395/878
5,016,166	5/1991	Van Lee et al.	395/874
5,127,086	6/1992	Bosenthal et al.	311/200
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,587	5/1995	Rand et al.	395/830
5,548,778	4/1996	Hirosewa	395/825
5,551,244	5/1996	Necoros et al.	395/280
5,642,481	6/1997	Polunoni	395/181,011
5,671,442	9/1997	Phoney et al.	395/834

FOREIGN PATENT DOCUMENTS
153148 7/1995 European Pat. Off.

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

"The DASH Local Kernel Structure" by David P. Anderson and Shiu-Yuan Tsou, Report No. UCB/CSD 88/463, 1988, Computer Science Division (EECS), Univer. California, Berkeley 94720.

"A Users' Guide to PIC1—A Portable Instrumentation Communication Library" By G.A. Grant et al., Oak National Laboratory, Mathematical Sciences Sect. Box 2009, Bldg. 9207-A, Oak Ridge, TN 3783 (Aug. 1990).

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

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

Primary Examiner—Moustafa M. Meiky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Walter & Schaub

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

19 Claims, 7 Drawing Sheets

INTEL Ex. 1005.001

Within the udpscript procedure described above, the nextid() function provides a monotonically increasing 16-bit counter required by the IP protocol. The ipchecksum() function performs a checksum for the IP 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 likelihood, the adapter would have a very limited knowledge of the user process' virtual address space, probably only knowing how to map virtual-to-physical for a very limited range, maybe as small as a single page. Pages in the user process' virtual address space for such buffers would need to be fixed. The udpscript procedure would need to be enhanced if the user data were allowed to span page boundaries. The udpchecksum() procedure generates a checksum value for both the UDP Header 708 plus the user data (not shown).

Ex. 1005 at 8:20-24

Erickson (Ex. 1005)

'072 Patent "Dividing, By The Interface Device, The Data Into Segments" and "Creating Headers for the Segments" (All Challenged Claims)

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01406¹
U.S. Patent 7,673,072

CORRECTED PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMEROOTH, PH.D.

99. Because Erickson's network adapter has "very limited knowledge" of the user process's virtual address space (and operates on fixed, single pages, as described above), the modification proposed by Petitioner and its expert would be **technically infeasible** and **extremely difficult to implement without significantly changing Erickson to become a different system**—one that supported a different memory access scheme and supported different protocols. For these reasons, the combination of **Erickson and Tanenbaum would not provide a POSA with a reasonable expectation of success** to make or use an implementation that supported dividing data segments.

Ex. 2026 at 48-49

¹ Cavium, who filed a Petition in Case IPR2017-01707, has been joined as a petitioner in this proceeding.

No Motivation to Combine – Dr. Horst’s Admission

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

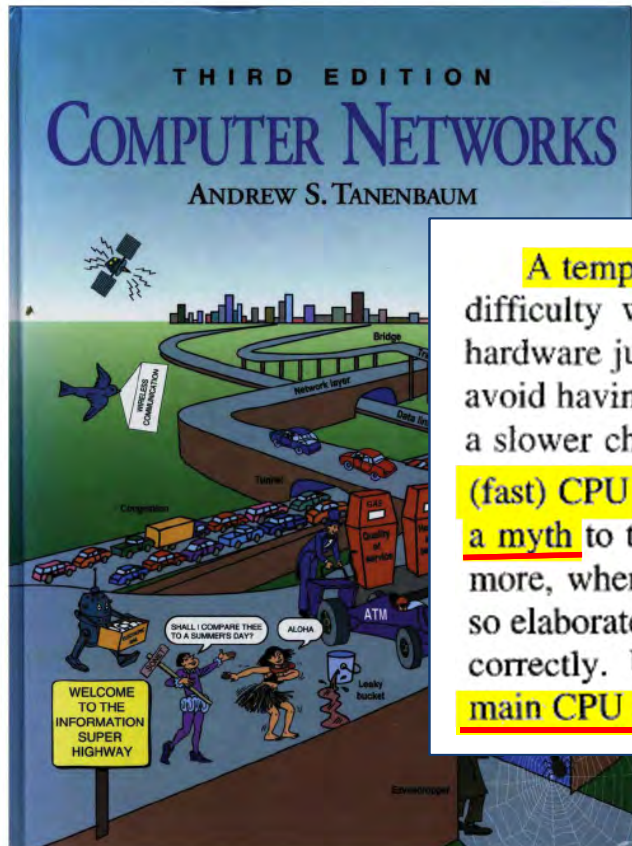
Networks customized to storage networking, such as Fibre Channel, were developed largely due to the perception that standard networking protocols are too heavyweight for attaching storage. Conventional wisdom says that IP storage is impractical without special purpose NICs to accelerate the TCP/IP protocol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

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

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

Ex. 2300.001

No Motivation to Combine – Tanenbaum Teaching Away



INTEL Ex.1006.001

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.

Ex. 1006.588-89

Tanenbaum (Ex. 1006)

No Motivation to Combine – Tanenbaum Teaches Away

87. (*Id.* at .588-89.) Accordingly, based on the above passage, Tanenbaum identifies myriad difficulties with implementing TCP header bypass in a chip separate from the host CPU and advises against attempting such an implementation. These difficulties are precisely the issues that were solved by the invention of the '036 patent. At the very least, Tanenbaum teaches that integration of transport entity functions in the NIC is complicated and non-trivial, but does not provide any insight or suggestion of how the complications may be overcome. Instead, Tanenbaum suggests a “Fast TPDU Processing” method that is implemented and performed on the host—precisely the opposite of the '036 invention.

Ex. 2026 at 39

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01391¹
U.S. Patent 7,237,036

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Almeroth Decl. (Ex. 2026)

-01391/-01406 PO Demonstrative, 29

No Motivation to Combine – Speed

Furthermore, the premise of Patent Owner’s argument—that the combination must be predictably “faster” than the prior art—is also wrong. The principal reason for offloading TCP/IP is not to make the network “faster.” Ex. 1223, ¶ 44. Offloading TCP/IP protocol processing is intended to lower the processing load on the host computer’s main CPU, which readily occurs with offload to a second processor, a well-known approach. *Id.*

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION

ALACRITECH, INC.

Patent Owner.

Case IPR2017-01391¹
 U.S. Patent No. 7,237,036
 Title: FAST-PATH APPARATUS FOR RECEIVING DATA
 CORRESPONDING A TCP CONNECTION

PETITIONER’S REPLY TO PATENT OWNER’S RESPONSE TO
 PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 7,237,036

Mail Stop “PATENT BOARD”
 Patent Trial and Appeal Board
 U.S. Patent and Trademark Office
 P.O. Box 1450
 Alexandria, VA 22313-1450

¹ Cavium, Inc., which filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding. Wistron Corporation, which filed a Petition in Case IPR2018-00327, has been joined as a petitioner in this proceeding.

Petitioner Reply, Paper 41

Petitioner Reply, Paper 41 at 13

-01391/-01406 PO Demonstrative, 30

No Motivation to Combine – Complexity

POSITA would put where in the system.” In addition, because the TCP transmission mechanism is much more complex than UDP and must support congestion control in the form of a sliding transmission window, retransmissions, and ordered, reliable delivery, it would not have been at all apparent to a POSITA that a TCP implementation of *Erickson’s* simplistic approach of using a local script to update header fields would be successful or operable to maintain and exchange the much more complex state information required by TCP.

UNITED STATES PATENT

BEFORE THE PATENT

INTE
CAV

Pe

ALACR

Pat

Case IP
U.S. Pa

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

Ex. 2026.43

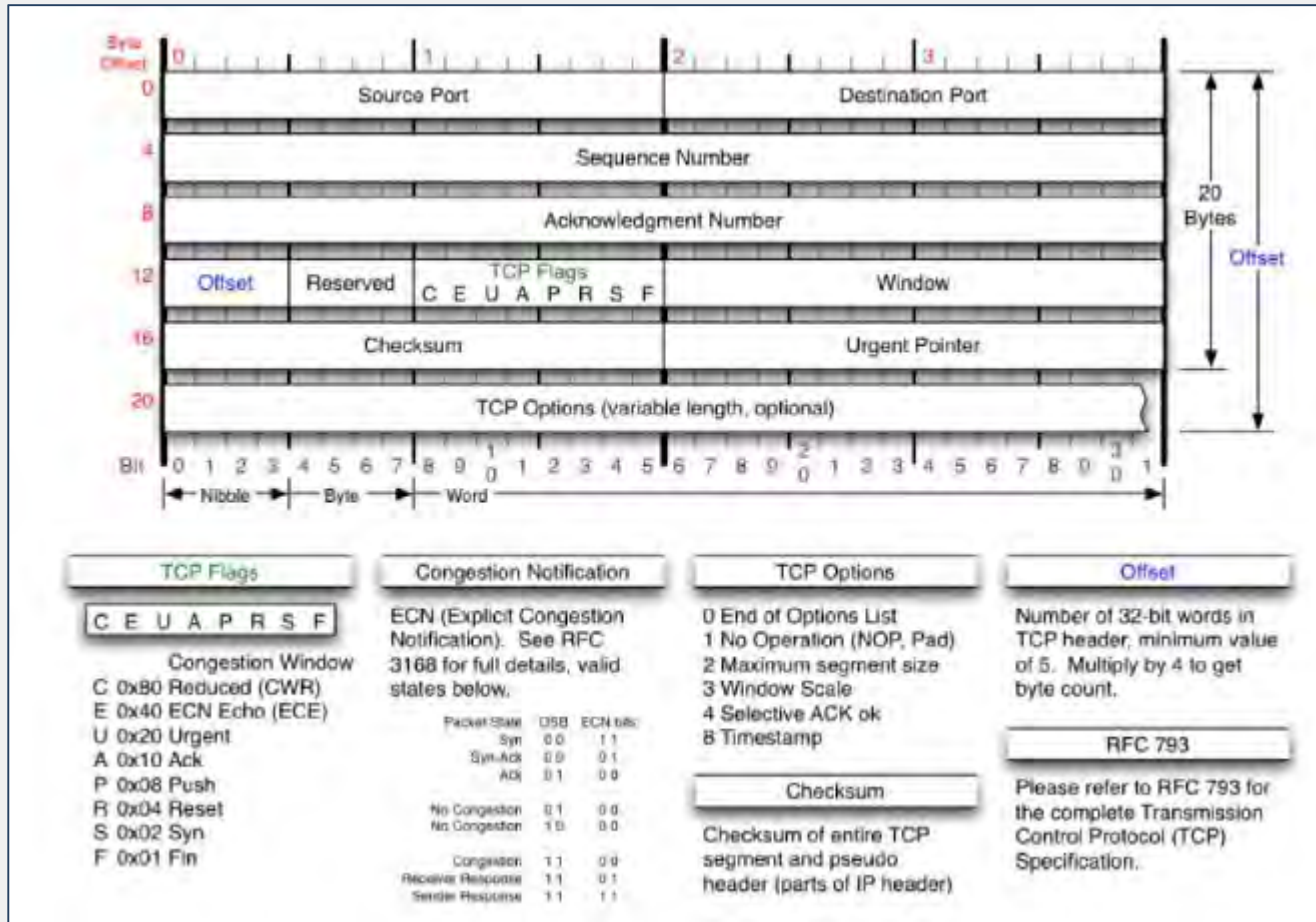
¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Almeroth Decl. (Ex. 2026)

-01391/-01406 PO Demonstrative, 31

No Motivation to Combine – Complexity



Ex. 2042

No Motivation to Combine – Complexity



Robert Horst
Petitioner's Expert

16 Q. (By Mr. Mack) Right. The pseudocode that we
17 looked at earlier, that was a UDP script, right?
18 A. Yes.
19 Q. And TCP is a different protocol than UDP,
20 right?
21 A. Yes.
22 Q. TCP is connection oriented, right?
23 A. Yes.
24 Q. There's a three-way TCP handshake in order to
25 establish a TCP connection?
2 A. Yes.
3 Q. And how are UDP connections established?
4 A. UDP is connectionless.
5 Q. So would you say the UDP connection process is
6 simpler than the TCP connection process?
7 A. Yes.
8 Q. And UDP doesn't support retransmission, does
9 it?
10 A. The UDP protocol itself doesn't do
11 retransmission like TCP does.
12 Q. And UDP also doesn't support flow control, does
13 it?
14 A. It doesn't support the type of flow control
15 that TCP does, no.
16 Q. And TCP and UDP have different header sizes,
17 right?
18 A. I believe so, yes.

Ex. 2028 (Horst Dep. Tr.) at 96:16-97:18

-01391/-01406 PO Demonstrative, 33

No Motivation to Combine – Complexity



Robert Horst
Petitioner's Expert

4 Q. And UDP doesn't include sequence numbers in the
5 header, right?
6 A. Right.
7 Q. And TCP does include sequence numbers, right?
8 A. Yes.
9 Q. Are there any other differences between TCP and
10 UDP that you can think of?
11 A. TCP has a number of other features that are not
12 in UDP.
13 Q. Would you say that TCP is a more complicated
14 protocol than UDP?
15 A. Yes.

Ex. 2028 (Horst Dep. Tr.) at 98:4-15

No Motivation to Combine – Complexity



Robert Horst
Petitioner's Expert

21 Q. So what does it take to determine
22 this Window field for a TCP header?
23 MR. CONSTANT: Objection. Form.
24 A. I don't really understand the
25 question. The determination of how the Window
2 field works is a fairly complex process,
3 described in my report, described in Tanenbaum.
4 There's not a short answer to that question.
5 Q. Does the sender need to determine the
6 Window field before it send out the TCP packet?
7 MR. CONSTANT: Objection. Form.
8 Asked and answered.
9 A. The Window field is a part of the TCP
10 header, so the Window field does have to be
11 determined before the packet is sent out.

Ex. 2600 (Horst Dep. Tr.) at 13:21-14:11

No Motivation to Combine – Complexity



Robert Horst
Petitioner's Expert

16 Q. You just mentioned trade-off. What
17 do you mean by trade-off, what trade-off?

18 MR. CONSTANT: Objection. Form.

19 A. Trade-offs means at every step of a
20 design you have to evaluate alternatives and
21 decide which alternative is the best one to use.

22 Q. So in the -- in the design of network
23 protocol offload, what trade-off is involved?

24 MR. CONSTANT: Objection. Form.

25 A. There are trade-offs on what kind of
2 a processor to use and in what kind of -- which
3 portions of the protocol to offload, all the
4 things that I talked about in the background
5 section of my report.

18 Q. You also mentioned about what should
19 be offloaded is also a trade-off to be
20 considered. Why that is a trade-off to be
21 considered?

22 MR. CONSTANT: Objection. Form.

23 A. When more of the protocol is
24 offloaded, there's more complexity in the
25 software that's run on the network interface
2 card, so it's -- it may be more difficult, but in
3 some cases it could be higher performance. So
4 there's trade-offs on the performance versus the
5 complexity.

Ex. 2029 (Horst Dep. Tr.) at 16:19-17:7, 17:18-18:5

Secondary Considerations: Long-Felt Need

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality

Ex. 2031 at 1 (1993 IBM Research Division ACM SIGCOMM article)

1 Introduction

Many researchers have observed that the performance of remote applications have not kept pace with modern communication network speeds. Part of this imbalance is attributed to the protocols used by the remote applications, namely, UDP/IP and TCP/IP. It is now widely believed that the problems with these protocols are not inherent in the protocols themselves but in their particular implementations [Dalt93, Whet95]. The following factors are cited as contributors to the poor performance of UDP/IP and TCP/IP:

Ex. 2032 at 1 (1995 Univ. of Berkeley paper)

Secondary Considerations: Long-Felt Need

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to “bottlenecks” in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication pro-

Ex. 2033 at 1 (1990 IEEE article from AT&T Bell Labs)

Abstract

Server network performance is increasingly dominated by poorly scaling operations such as I/O bus crossings, cache misses and interrupts. Their overhead prevents performance from scaling even with increased CPU, link or I/O bus bandwidths. These operations can be reduced by redesigning the host/adaptor interface to exploit additional processing on the adapter. Offloading processing to the adapter is beneficial not only because it allows more cycles to be applied but also of the changes it enables in the host/adaptor interface. As opposed to other approaches such as RDMA, TCP offload provides benefits without requiring changes to either the transport protocol or API.

Ex. 2034 at 1 (2005 USENIX Conference paper from IBM)

Secondary Considerations: Long-Felt Need

117. The nexus between the long-felt need and the claimed invention is clear and direct. **The aforementioned bottlenecks are all solved by the accelerated network processing technologies recited in the challenged claims.** In addition, the technologies recited in the challenged claims alleviated processing demand on the host CPU in such a way as to permit faster network throughput and transmission/reception speeds—the precise long-felt but unresolved need in the industry outlined above.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIEBUNAL

INTEL CORPORATION
CAVIUM

Petitioner

ALACRITECH, LLC

Patent Owner

Case IPR2017-01391¹
U.S. Patent 7,237,036

CORRECTED PATENT OWNER'S EXHIBIT 2026

DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

01391 Ex. 2026 at 56; see also 01406 Ex. 2026 at 63

-01391/-01406 PO Demonstrative, 39

Secondary Considerations: Industry Praise

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

Ex. 2039 at 4 (2001 HP Labs research paper)

Secondary Considerations: Industry Praise

"The ANX 1500 is an evolutionary advancement of Alacritech's long standing leadership in protocol acceleration, which is now applied not just to protocols, but to the optimization of all storage system interactions," said Jeff Boles, a technology analyst with Taneja Group. "The ANX 1500 substantially augments the performance of existing NAS investments from the best place possible – from right in the customer's Ethernet network – without reconfiguration or changes in management for the existing infrastructure. We think Alacritech is setting the stage for a next generation of solutions that will accelerate storage from outside the storage array, and more cost effectively share an investment in performance across many storage systems and clients. I've talked to early-stage customers using the product, and they believe it's game-changing."

Ex. 2040 at 3 (Shoreline Ventures Press Release)

-01391/-01406 PO Demonstrative, 41

Secondary Considerations: Industry Praise

121. The industry universally praised **commercial embodiments of the features described in the challenged claims**. HP found that the “Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed” and “achiev[e] lower processor utilization than native NT’s TCP/IP protocol stack for transmission of large enough messages.” (Ex. 2039 at ¶ 4.) HP found that the test performance of the Alacritech iNIC “raises challenging questions for [HP’s] choice of implementation technology because “the Alacritech approach is definitely better than our offload.” *Id.*

01391 Ex. 2026 at 58; see also 01406 Ex. 2026 at 65

UNITED STATES PATENT AND TRADE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01391¹
U.S. Patent 7,237,036

CORRECTED PATENT OWNER'S EXHIBIT 2026

DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Secondary Considerations: Failure of Others

To this day, TCP offload has never firmly caught on in the commercial world (except sometimes as a stopgap to add TCP support to immature systems [16]), and has been scorned by the academic community and Internet purists. This paper starts by analyzing why TCP offload has repeatedly failed.

Ex. 2041 at 2 (2003 HP Labs paper presented at HotOS IX conference)

TCP offload has been unsuccessful in the past for two kinds of reasons: fundamental performance issues, and difficulties resulting from the complexities of deploying TCP offload in practice.

Id. at 2

Secondary Considerations: Failure of Others

offload were mismatched to the applications in question.” *Id.* The TCP offload described above is a form of network processing offload that is described by the challenged claims, and this failure of others therefore has a direct nexus to the claimed inventions.

01391 Ex. 2026 at 59;
01406 Ex. 2026 at 66

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,
Petitioners,

v.,

ALACRITECH INC.,
Patent Owner.

Case IPR2017-01391¹
U.S. Patent 7,237,036

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

-01391/-01406 PO Demonstrative, 44

Secondary Considerations: Skepticism

IP Storage and the CPU Consumption Myth

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

Abstract

This paper addresses a key issue that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2.

The
accel
CPU
been
comm
unme
Ex
date
many
I/O p
Howe
of arc
rapid
A
I/O)
proce
serve
from
starte
task,
as the
some
Some
or w
and s
usual
and e
envir
costly
for if
the p
kills
gener
the c
Si
accel
of en
every

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

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

Ex. 2300 at 194 (2001 paper by Petitioner's expert, Dr. Horst)

Secondary Considerations: Skepticism

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

Networks customized to storage networking, such as Fibre 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 paper 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.

2. The Historical A

There are many historical examples of hardware accelerators to offload processor work from the main CPU. Some examples, such as network communications processors, have been successful, but others have fallen far short of unmet expectations.

Examples of front-end accelerators date from the early days of computing. In many systems, the primary I/O processor to offload the main CPU. However, it has become in vogue to use a different architecture to deliver a rapid pace of technology change.

A specific recent example is the use of an I/O processor, such as an Intel Itanium processor, to serve as an I/O processor for its attached I/O devices. It started, the i960 embedded processor, but its performance did not live up to expectations as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that hardware accelerators should have been used more extensively. However, every protocol worthy of a name is difficult to stay afloat in a market that is far from stable, and even if formally approved, there are many bug fixes. It seems extremely

as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Secondary Considerations: Skepticism

(Ex. 2300 at 194.) (emphasis added). This level of skepticism and teaching away is directly related to the heart of the claimed invention – offloading network processing, and therefore has a direct nexus to the challenged claims.

01391 Ex. 2026 at 61; see also 01406 Ex. 2026 at 68

UNITED STATES PATENT AND

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01391¹
U.S. Patent 7,237,036

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

'036 Patent – Contingent Motion to Amend (Proposed Claim 23 – Substitute for Claim 1)

Proposed Claim 23

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:

a communication processing mechanism connected to the first processor,

said communication processing mechanism containing a second processor

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

if the incoming message packet should be processed by the second processor, to process **ing 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,

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.**

Support for Proposed Claim 23 (Substitute for Claim 1)

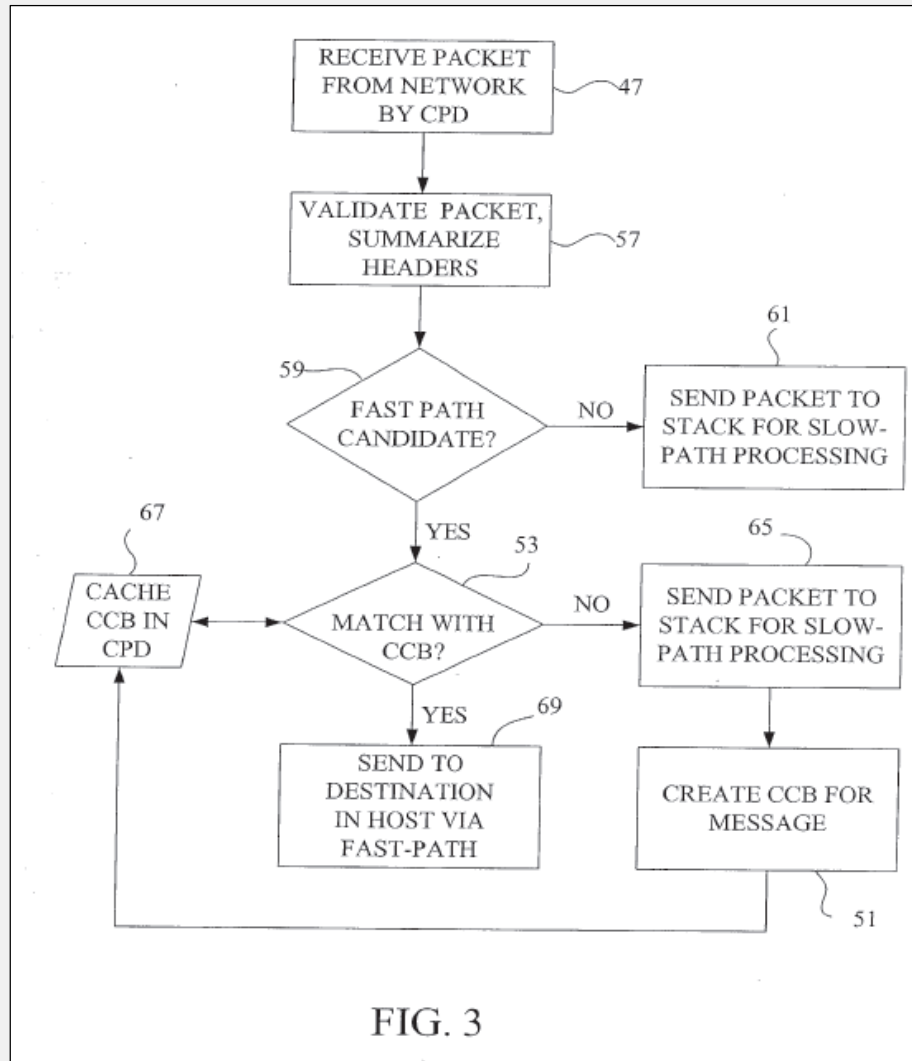


FIG. 3

Ex. 2020.152

Proposed Claim 23 (Substitute for Claim 1) Is Valid Over the Prior Art

Accordingly, adapting Erickson to include the “header prediction” described in Tanenbaum96 discloses *running instructions on the second processor (running scripts on the processor on the network interface card), wherein the second processor determining whether an incoming message packet should be processed by the second processor (Erickson’s I/O adapter with the header prediction described in Tanenbaum96 to determine if the incoming message packet should be processed by the second processor) as a fast path packet.*

Petitioner Opposition to Motion to Amend, Paper 36 at 15-16

Filed: Ap

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and CAVIUM, INC.,
Petitioner,

v.

ALACRITECH, INC.,
Patent Owner.

Case IPR2017-01391
U.S. Patent No. 7,237,036¹
Title: FAST-PATH APPARATUS FOR RECEIVING DATA
CORRESPONDING A TCP CONNECTION

PETITIONER’S RESPONSE IN OPPOSITION TO PATENT OWNERS’
CONTINGENT MOTION TO AMEND UNDER 37 C.F.R. § 42.121

Mail Stop “PATENT BOARD”
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

¹ Cavium, Inc., which filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Proposed Claim 23 (Substitute for Claim 1) Is Valid Over the Prior Art

63. I understand that Petitioner alleges Erickson discloses a slow path for both incoming and outgoing message packets. Opp., 20. However, the amended element is not only about using a slow path, but using a slow path *after the determination*. Erickson only discloses a prior art slow path.

64. I understand that Petitioner further alleges that based upon a second processor's determination using header prediction on Tanenbaum96 (which, as noted above, is not done on a second processor) to classify packets according to the slow or fast path, it is obvious to use a first processor to process incoming packets on a slow path. Opp., 21. In my opinion, this argument is misplaced. The critical inquiry of the amendment is when to use the slow path, which according to the amendment, is after the determination limitation. In my opinion, neither reference discloses this limitation. *Id.*

Ex. 2305.22

Case No. IPR2017-0
U.S. Patent No. 7,237,036

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-01391¹
U.S. Patent 7,237,036

DECLARATION OF KEVIN ALMEROOTH, PH.D. IN SUPPORT OF
PATENT OWNER'S REPLY IN SUPPORT OF CONTINGENT MOTION
TO AMEND UNDER TO 37 C.F.R. § 42.121

¹ Cavium, who filed a Petition in Case IPR2017-01718, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2305

Almeroth Decl. (Ex. 2305)

'072 Patent – Contingent Motion to Amend (Proposed Claim 22 – Substitute for Claim 1)

Proposed Claim 22

22. A method comprising:

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

transferring the **context** ~~protocol header information~~ to an interface device;

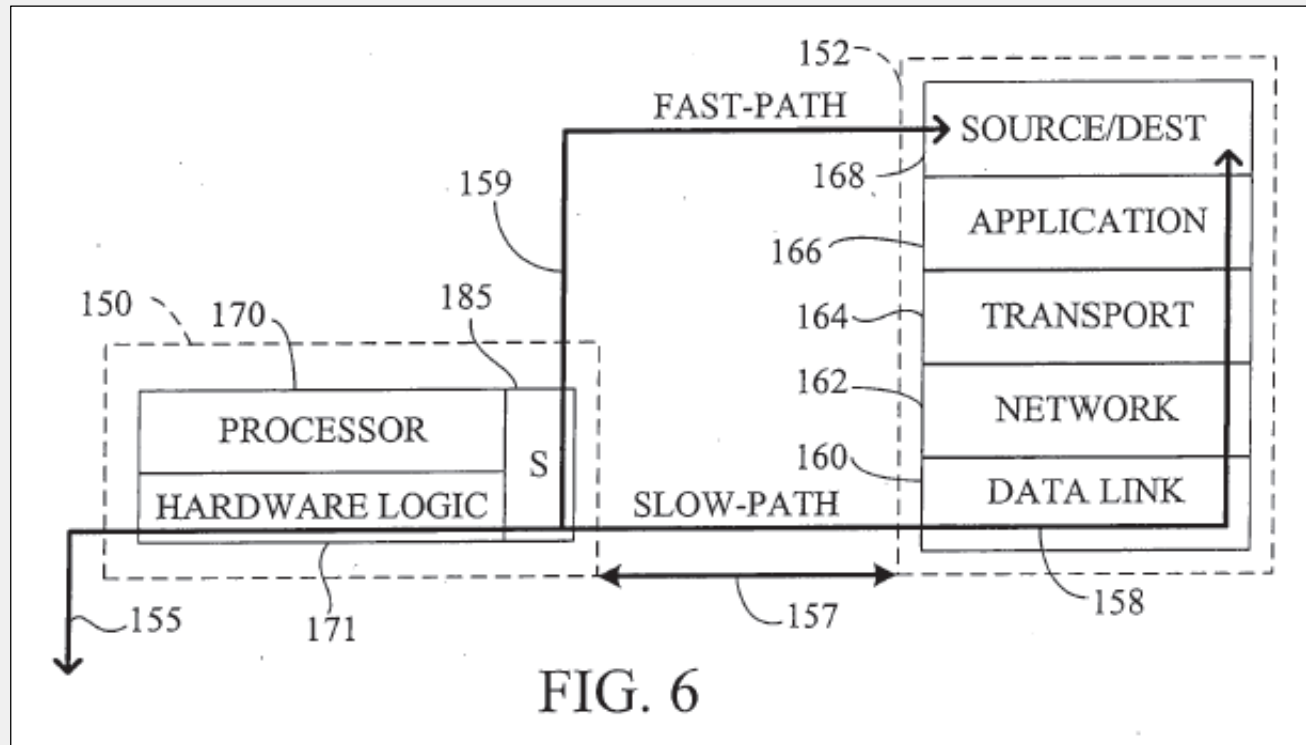
transferring data from the network host to the interface device, after transferring the **context** ~~protocol header information~~ to the interface device;

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

creating headers for the segments, by the interface device, from a template header containing ~~the~~ protocol header information **including IP address and TCP state information**; and

prepending the headers to the segments to form transmit packets.

Support for Proposed Claim 22 (Substitute for Claim 1)



Ex. 2024.154 Fig. 6

Support for Proposed Claim 22 (Substitute for Claim 1)

2.1.4 The TCB cache

Consider a situation in which a TCP connection is being handled by the card and a fragmented TCP segment for that connection arrives. In this situation, it will be necessary for the card to turn control of this connection over to the host.

This introduces the notion of a **Transmit Control Block (TCB) cache**. A TCB is a structure that contains the entire context associated with a connection. **This includes the source and destination IP addresses and source and destination TCP ports that define the connection.** It also contains information about the connection itself such as the current send and receive sequence numbers, and **the first-hop MAC address, etc.** The complete set of TCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the TCB cache. The INIC can own up to 256 TCBs at any given time.

Ex. 2019 at 6

Proposed Claim 22 (Substitute for Claim 1) Is Valid Over the Prior Art

Case No. IP
U.S. Patent

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP., CAVIUM, INC., and
WISTRON CORPORATION,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-01406¹
U.S. Patent 7,673,072

DECLARATION OF KEVIN ALMEROOTH, PH.D. IN SUPPORT OF
PATENT OWNER'S REPLY IN SUPPORT OF CONTINGENT MOTION
TO AMEND UNDER TO 37 C.F.R. § 42.121

¹ Wistron Corporation, which filed a Petition in Case IPR2018-00329, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2305

67. I understand that Petitioner then argues that Erickson, in view of Tanenbaum96, discloses this limitation because Erickson discloses transferring the elements to the network interface device. Opp., 13. Petitioner is again incorrect. As noted above (and in the Corrected Response), Erickson is only concerns UDP; there is no teaching regarding TCP. Also, neither reference discloses transferring the context information to an interface device or second processor. See Corrected Response, 24-33, 33-34, 45; see also *id.*, 16-19.

Ex. 2305.67

Almeroth Decl. (Ex. 2305)

-01391/-01406 PO Demonstrative, 56

**Intel Corporation, Cavium, Inc., Wistron
Corporation, and Dell Inc.**

v.

Alacritech, Inc.

Case Nos.: IPR2017-01392, IPR2017-01728,
IPR2018-00328, IPR2018-00372
U.S. Patent No. 7,337,241

Patent Owner's Demonstratives

Hearing: September 13, 2018

-01392 PO Demonstrative, 1

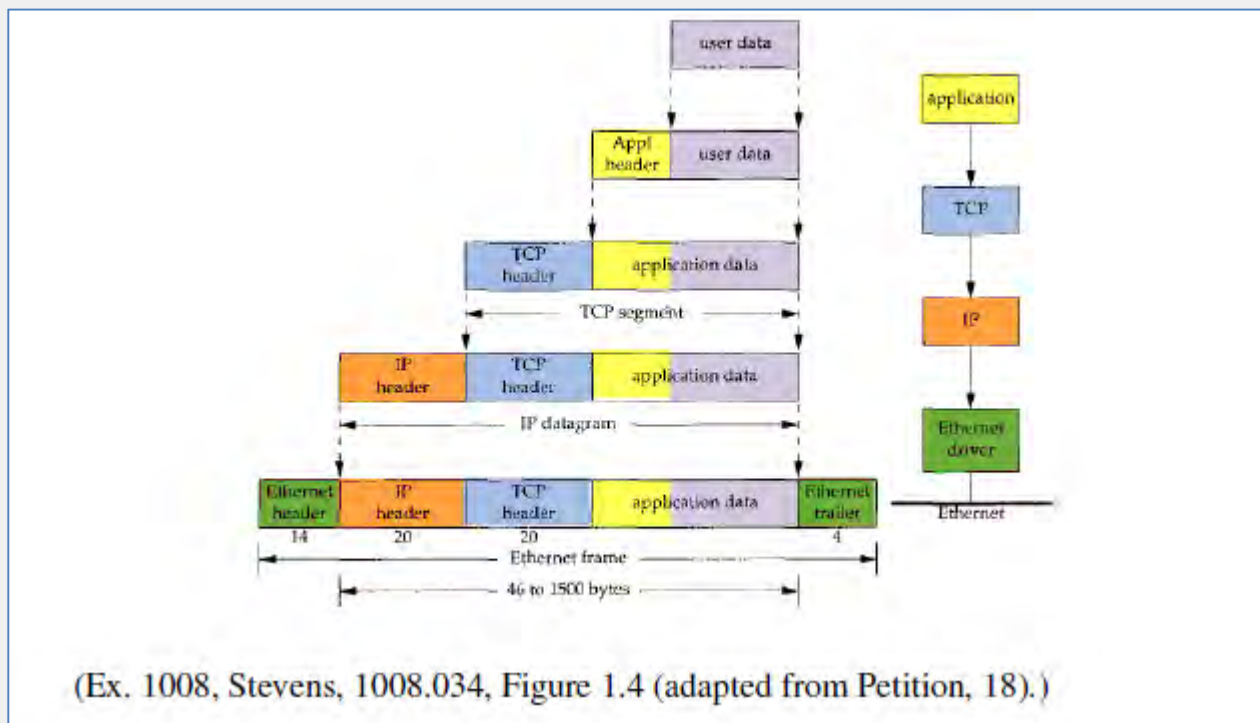
Overview of the '241 Patent

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

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

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

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



Corrected POR at 10.

-01392 PO Demonstrative, 2

Overview of Interrupt Processing

CHAPTER 10

Interrupt Handling



Although some devices can be controlled using nothing but their I/O real devices are a bit more complicated than that. Devices have to interact with the external world, which often includes things such as spinning disks.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

00073-00001/9802806

Alacritech Exhibit 2026

70. Because the processing of each layer typically involves a copy and data manipulation operation (for example a checksum computation operation), the host CPU must be “interrupted” at least one time per layer in order to process the data and construct (transmit side) or deconstruct (receive side) the packet. An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. One popular textbook I am familiar with, *Linux Device Drivers* by A. Rubini and J. Corbert, was published in June 2001 and explains “[a]n *interrupt* is simply a signal that the hardware can send when it wants the processor’s attention.” (Ex. 2004 at 258 (emphasis in original)). The purpose of the interrupt is to “let the processor know when something has happened.” (*Id.*) While computers attempt to run multiple processes simultaneously, a single processor can operate on only one task at a time. However, not all tasks need work done by the processor all of the time—some tasks are stalled waiting, for example, on input from a user or for a packet to arrive. Interrupts are used to inform the processor as to which tasks are ready for work to be done by the processor. An

Ex. 2026 at 31 (citing Ex. 2004).

Overview of Interrupt Processing

71. The interrupt process used in connection with traditional network interface cards results in “repeated copying and interrupts to the CPU” of the host computer. (Ex. 1001 at 5:24-28). When the host CPU is interrupted, it generally must stop all other tasks it is currently working on, including tasks completely unrelated to the network processing. “For the most part, a driver need only register

73. Frequent interrupts to the host CPU can be very disruptive to the host system generally and cause system instability and degraded system performance. (Id.).

Ex. 2026 at 32-33

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,
Petitioners,

v.

ALACRITECH INC.,
Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

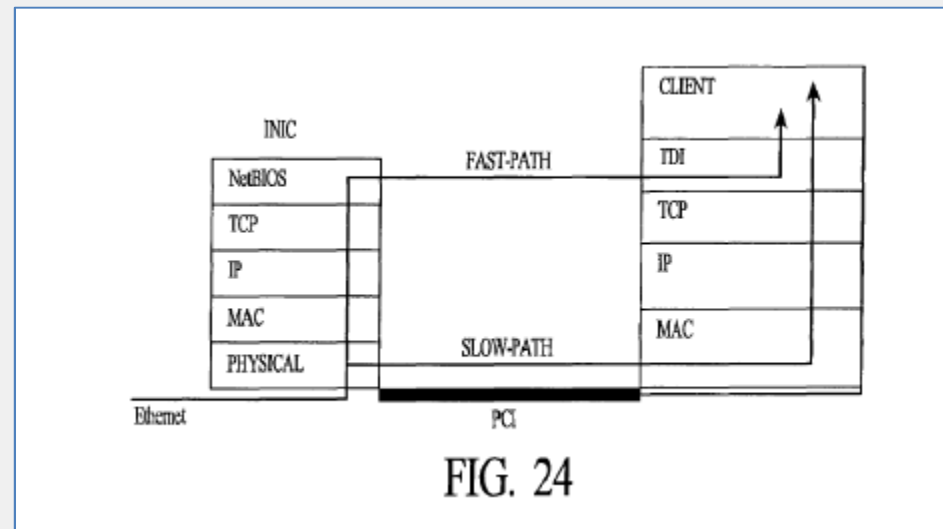
CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Overview of the '241 Patent

directly to its intended destination. This fast-path bypasses conventional protocol processing of headers that accompany the data. The fast-path employs a specialized microprocessor designed for processing network communication, avoiding the delays and pitfalls of conventional software layer processing, such as repeated copying and interrupts to the CPU. In effect, the fast-path replaces the states that are

'241 Patent, Ex. 1001, 5:22-28.



'241 Patent, Ex. 1001, Fig. 24.

Overview of the '241 Patent

Interrupt reduction in the present invention is obtained by:

- Transmit and receive fast-paths functioning **at an upper layer** (session or higher)
- All interactions between the NIC and host use **full transfer sizes**

can be sent to the network by the fast-path. Both the input and output fast-paths attain a huge reduction in interrupts by functioning at an upper layer level, i.e., session level or higher, and interactions between the network microprocessor and the host occur using the full transfer sizes which that upper layer wishes to make. For fast-path communications, an interrupt only occurs (at the most) at the beginning and end of an entire upper-layer message transaction, and there are no interrupts for the sending or receiving of each lower layer portion or packet of that transaction.

'241 Patent, Ex. 1001, 11:37-46.

- This arrangement permits **(at most) one interrupt** at the beginning and end of an **entire upper-layer message transaction** (regardless of size)

Challenged Claims of the '241 Patent (Claims 1-24)

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

- receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;
- processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;
- sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;
- sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination.

Challenged Claims of the '241 Patent (Claims 1-24)

9. A method for communicating information over a network, the method comprising:
obtaining data from a source in memory allocated by a first processor;
dividing the data into multiple segments;
prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and
transmitting the packets to the network.

Challenged Claims of the '241 Patent (Claims 1-24)

17. A method for communicating information over a network, the method comprising:
providing, by a first mechanism, a block of data and a Transmission Control Protocol (TCP) connection;
dividing, by a second mechanism, the block of data into multiple segments;
prepending, by the second mechanism, an outbound packet header to each of the segments, thereby forming an outbound packet corresponding to each segment, the outbound packet header containing an outbound media access control layer header, an outbound Internet Protocol (IP) header and an outbound TCP header, wherein the prepending of each outbound packet header occurs without an interrupt dividing the prepending of the outbound media access control layer header, the outbound (IP) header and the outbound TCP header; and transmitting the outbound packets to the network.

Overview of the Prior Art

Challenged Claims	103(a) References
1-8	Erickson, Tanenbaum, and Alteon
9-17, 19-21, and 24	Erickson and Tanenbaum
18, 22, and 23	Erickson, Tanenbaum, and Alteon

Alteon Does Not Qualify As A "Printed Publication"



“A given reference is ‘publicly accessible’ upon a satisfactory showing that such document has been disseminated or otherwise made available *to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence, can locate it.*”

Bruckelmyer v. Ground Heaters, Inc., 445 F.3d 1374, 1378 (Fed. Cir. 2006) (emphasis added)



“[T]he [putative publications] were not accessible to the public because they had *not been either cataloged or indexed in a meaningful way.*”

In re Cronyn, 890 F.2d 1158, 1161 (Fed. Cir. 1989)

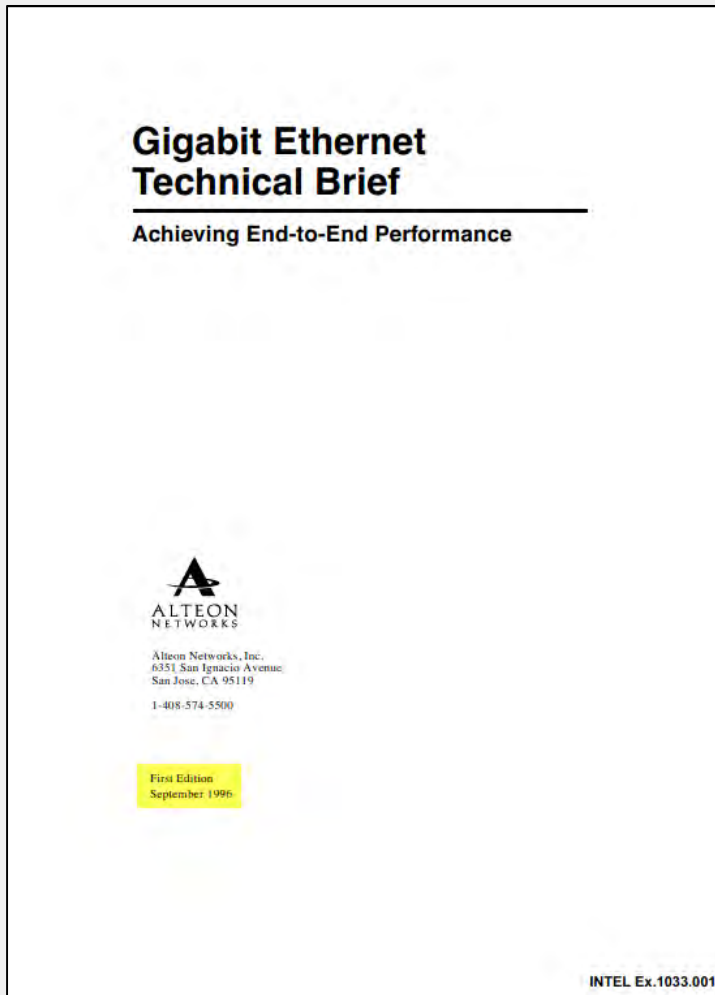
Alteon Does Not Qualify As A "Printed Publication"

9.3. "Gigabit Ethernet Technical Brief: Achieving End-to-End Performance" by Alteon Networks (Ex.1033, "Alteon")

Alteon, "Gigabit Ethernet Technical Brief: Achieving End-to-End Performance" is a technical brief describing multiple generations of Ethernet adapters, including Alteon's third generation intelligent network adapter with an on board processor. Alteon was published on or before January 26, 1997 and is therefore at least § 102(a) prior art to the earliest possible priority date for the 241 Patent claims. Ex.1087, Librarian Declaration of Christopher Butler.

Corrected Pet. at 40.

Alteon Does Not Qualify As A “Printed Publication”



- Appears to be the “first edition” of an Alteon Networks internal technical brief dated September 1996
- No indication on the document itself that it was every published or accessible in 1996
- Even Petitioner alleges it was only “published on or before January 26, 1997”

Ex. 1033

Alteon Does Not Qualify As A "Printed Publication"

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019852806.6

Alacritech Exhibit 2026

105. In my opinion, Petitioner has not explained why a person interested and ordinarily skilled in the subject matter or art exercising reasonable diligence would have been able to locate *Alteon* prior to the priority date of the '241 Patent. In fact, I have reviewed the main *Alteon* webpage (www.alteon.com) as of the date the "crawler" allegedly accessed the document, and it *does not include a link to the URL* from which crawler allegedly accessed the technical brief.

106. It is unclear, therefore, how a person of ordinary skill in the art would be able to access the URL from which the technical brief was allegedly accessed by the crawler without knowledge of the full URL path (which appears to be <http://www.alteon.com/techbrief.ps>). An automated software "crawler" obviously has much greater faculties to locate webpages than a POSITA. In fact, a crawler, like the one that apparently located the *Alteon* reference, only functions to discover arbitrary webpages and might spend its entire life doing so. The fact that a crawler located the *Alteon* reference therefore says nothing about whether a POSITA exercising reasonable diligence would have been able to locate it.

Ex. 2026 at 51-52

-01392 PO Demonstrative, 14

Alteon Does Not Qualify As A "Printed Publication"

At this preliminary stage of the proceeding, Petitioner has shown sufficiently that Alteon qualifies as a prior art printed publication. However, we note Petitioner does not assert that Exhibit 1033 (the Alteon reference relied upon by Petitioner) is identical to "Exhibit A" of the Butler Declaration (a version of the Alteon reference authenticated by Mr. Butler. For purposes of this Decision, we presume Exhibit 1033 is identical to the document Mr. Butler attests to as "Exhibit A" in his Declaration. Furthermore, Patent Owner correctly observes that Petitioner does not explain how an ordinarily skilled artisan would locate the URL (www.alteon.com/techbrief.ps) without locating the underlying home page (www.alteon.com). Prelim. Resp. 42. However, Patent Owner provides, in Exhibit 2006, a copy of an archived web page corresponding to the URL www.alteon.com. Although Mr. Butler's Declaration does not address Patent Owner's Exhibit 2006, it appears that both Exhibit 2006 (an archive printout of the www.alteon.com home page) and "Exhibit A" of the Butler Declaration (an archive printout of www.alteon.com/techbrief.ps) were archived on the same day (January 13, 1997). Although Petitioner does not identify a link from the archived home page to the techbrief.ps web page, on the record before us we find it is sufficiently shown for purposes of this Decision that the two pages, archived on the same date, are linked in such a manner that the interested ordinarily skilled artisan would have been able to locate and access the techbrief.ps document.

Trials@uspto.gov
571-272-7822

Paper 11
Entered: November 30, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner,

v.

ALACRITECH, INC.,
Patent Owner.

Case IPR2017-01392
Patent 7,337,241 B2

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

FISHMAN, *Administrative Patent Judge*.

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

I. INTRODUCTION

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

Institution Decision, Paper 11 at 8-9.

-01392 PO Demonstrative, 15

Alteon Does Not Qualify As A “Printed Publication”

Patent Owner’s Response notably does not argue that Alteon is not prior art, only that Petitioner has failed to put forth sufficient evidence. This is likely because Patent Owner submitted a version of Alteon to the Patent office during prosecution and admitted that the submitted version is prior art. Ex. 1002 at .309-10 (Rows Q and N). The sum total of evidence leads to the undeniable conclusion that Alteon was available prior art and would have been readily found by a POSA.

Reply at 2.

- This new argument made in reply should be ignored

The Combination Fails to Suggest Validation of Network and Transport Layer Headers “Without an Interrupt Dividing the Processing” of the Headers (Claims 1-8)

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

- receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;
- processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;
- sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;
- sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination.

The Combination Fails to Suggest Validation of Network and Transport Layer Headers “Without an Interrupt Dividing the Processing” of the Headers (Claims 1-8)

III. THE COMBINATION OF ERICKSON, TANENBAUM96 AND ALTEON RENDERS CLAIMS 1-8 OBVIOUS

A. Patent Owner Has Failed to Rebut Petitioner’s Showing That The Prior Art Discloses Validation of Network and Transport Layer Headers “Without an Interrupt Dividing the Processing” of the Layer Headers (Claim 1)

The Board recognized that Petitioner relies on Erickson, not Alteon, to teach

that the validation of headers is done by the interface and not the host computer.

Paper 11 at 14. Alteon discloses the absence of interrupts for each incoming

Reply at 4.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP., CAVIUM, INC., and
WISTRON CORPORATION,
Petitioner,

v.

ALACRITECH, INC.,
Patent Owner.

Case IPR2017-01392¹
U.S. Patent No. 7,337,241

Title: FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING
TO A TCP CONNECTION

**PETITIONER'S REPLY TO PATENT OWNER'S RESPONSE TO
PETITION FOR INTER PARTES REVIEW OF U.S. PATENT NO. 7,337,241**

Mail Stop “PATENT BOARD”
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

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

Erickson Only Describes Header Creation (Transmit-side), Not Header Validation (Receive-side) On the NIC



Robert Horst
Petitioner's Expert

8 Q. (By Mr. Mack) And this pseudocode corresponds
9 to the steps needed to -- at least part of the steps
10 needed to transmit the completed UDP data program 702
11 over the media, right?

12 A. Yes.

13 Q. And that's what it means when it says in
14 column 7, lines 45, "The I/O device adapter stores the
15 user data provided by the user process and the I/O
16 device's adapter memory and then transmits the completed
17 UDP datagram over the media," right?

18 MR. CONSTANT: Objection, form.

19 A. Yes.

20 Q. (By Mr. Mack) Why does the IP checksum have to
21 be computed over the template in order to transmit the
22 completed UDP datagram over the media?

23 A. The IP checksum has to be a valid value. So
24 they show an example where that's computed in the
25 adapter.

Ex. 2028 (Horst Dep. Tr.) at 72:8-25

Erickson Only Describes Header Creation (Transmit-side), Not Header Validation (Receive-side) On the NIC

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019850280.6

Alacritech Exhibi

(Institution Decision at 14.) In my view, this argument misses the mark. The portions of *Erickson* relied upon by Petitioner (4:18-23, 7:50-64, and 8:10-25) all relate to **checksum generation** used in the transmit process, not **checksum validation** used in the received process, as recited by claim 1. Petitioner's expert, Dr. Horst, confirmed this at his deposition and indicated that the pseudocode and related description of checksums in *Erickson* relates to the **transmit path** (where the checksums were written to the template header) and **not validation on the receive path** (where checksums would be read from the header and validated). (Ex. 2028 at 72:8-74:16.) Although checksum generation and validation may be related, they are not synonymous, and any one of the host, the network interface, or some other network device could validate layer headers that were created or generated by a different component or device. In addition, because checksums are traditionally generated (on the transmit side) and validated (on the receive side) independently, the disclosure of a particular process or methodology on the transmit side (checksum generation) does not reveal any information about the receive side (checksum validation). The two processes are independent of each other, and there is no requirement in the field that processing on the send side has any relationship to processing on the receive side.⁵

Ex. 2026 at 55-56

-01392 PO Demonstrative, 20

Alteon's Reduced Interrupts Are For The Copy Operation, Not Network and Transport Layer Validation

Repeating Data Steps 1, 2, and 3, the NIC puts as many packets as it can into the buffer until one of the following occurs:

- It runs low on buffer space
- The interrupt timer expires

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

The NIC card is said to be at idle during any period of time with no work to be performed that extends past the interrupt period. The next time an event occurs that requires the attention of the host, the NIC does not wait for the interrupt timer to expire. Instead, it begins to act on the data immediately. This is another example of adaptive interrupts.

In a busy environment, issuing a smaller number of interrupts over a regularly spaced time interval greatly improves network performance. In a light environment, it is preferable to issue interrupts more frequently to reduce latency. Alteon adapters allow you to configure the optimum number of interrupts for heavy loads on your network, while it dynamically evaluates network activity to determine which method is most appropriate for normal traffic loads.

Control Step A: The NIC issues a single interrupt, on a timed basis, telling the protocol stack how many packets it has copied to the buffer.

Ex. 1033.023

**Gigabit Ethernet
Technical Brief**
Achieving End-to-End



Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119
1-408-574-5500

First Edition
September 1996

INTEL Ex.1033.001

-01392 PO Demonstrative, 21

Interrupts For the Copy Operation Are Distinct From the Interrupts For Network and Transport Layer Validation

113. The “interrupt timer” operation is clarified in Control Step A and Data Step 3 of *Alteon* where it states “[t]he NIC issues a single interrupt, on a timed basis, telling the protocol stack how many packets it had copied to the buffer” and then “the data is copied to the application memory by the protocol stack.” (Ex. 1033.023.) While *Alteon* may support a single interrupt for copying multiple packets to the host, *Alteon* does not disclose any reduction in interrupts in connection with validating network or transport layer headers, as recited by claim 1. This is because the copy operation is distinct from the header validation operation, and interrupts can be triggered during one or both operations independent of the other operation. While *Alteon*’s interrupt timer can be used to copy more than one data packet into the operating system buffer space, it does not show or reasonably suggest the validation of any layer headers without the generation of an interrupt dividing the processing of the headers. (*Id.*) *Alteon*’s adaptive interrupts merely permits the flexibility to control the frequency of interrupts based on the amount of data being transmitted or received, but it does not tell us anything about whether or how many interrupts are generated during the network and transport layer header validation process.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,*Petitioners,*

v.

ALACRITECH INC.,

*Patent Owner.*Case IPR2017-01392¹
U.S. Patent 7,337,241CORRECTED PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined petitioner in this proceeding.

06073-000019850280.6

Alacritech Exhibit 2026

Ex. 2026 at 54-55

-01392 PO Demonstrative, 22

Petitioner Does Not Rely on Tanenbaum for Validation of Network or Transport Layer Headers Without an Interrupt Dividing the Processing

Accordingly, Erickson in combination with Alteon discloses processing the packets by a first mechanism (I/O device), so that for each packet the network layer header and the transport layer header are validated (checksum) without an interrupt dividing the processing of the network layer header and the transport layer header (Alteon's processing method).

Pet. at 54-55.

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION
Petitioner

v.

ALACRITECH, INC.
Patent Owner

Case IPR No. 2017-01392
U.S. Patent No. 7,337,241
Title: FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION

Corrected Petition For *Inter Partes* Review of U.S. Patent No. 7,337,241 Under 35 U.S.C. §§ 311-319 and 37 C.F.R. §§ 42.1-.80, 42.100-.123

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Patent Owner's Priority Application Refers to ACK Interrupts, Not Interrupts From Network and Transport Layer Validation

Importantly, Patent Owner's alleged priority application admits that fewer than one interrupt per received packet was normal in the prior art. When discussing the alleged problems in the prior art, the 1997 Provisional notes that "[a] 64k SMB request . . . is typically made up of 44 TCP segments" and that if "we assume 4 incoming frames per input, and an acknowledgement for every 2 segments, . . . we are still left with 33 interrupts . . ." Ex. 1031 at .006. Thus, even Patent Owner's alleged priority application admits that the prior art employed fewer than one interrupt for each received segment (33 interrupts for 44 segments). Ex. 1223, ¶¶ 53-56. A significant fraction (11 packets) were received without any

interrupts to the host. There are simpl

Reply at 6-7.

1.3 Too Many Interrupts

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

Ex. 1031.006

The Combination Fails to Suggest Sending, By the NIC, Data From Each of the Plurality of Packets to Application Memory Without Sending the Headers

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

- receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;
- processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;
- sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;
- sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination.

The Combination Fails to Suggest Sending, By the NIC, Data From Each of the Plurality of Packets to Application Memory Without Sending the Headers

Table 4. Steps in Data Reception using Second Generation NIC

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

Ex. 1033.021

Gigabit Ethernet Technical Brief

Achieving End-to-End Performance



Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119
1-408-574-5500

First Edition
September 1996

INTEL Ex.1033.001

Petitioner Only Relies on Erickson For Writing Data Into Memory Using DAI and Doesn't Cite to Tanenbaum At All

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION
Petitioner

v.

ALACRITECH, INC.
Patent Owner

Case IPR No. 2017-01392
U.S. Patent No. 7,337,241
Title: FAST-PATH APPARATUS FOR RECEIVING DATA CONNECTIONS
TO A TCP CONNECTION

Corrected Petition For *Inter Partes* Review of U.S. Patent No. 7,337,241
35 U.S.C. §§ 311-319 and 37 C.F.R. §§ 42.1-80, 42.101-106

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Erickson in combination with Tanenbaum96 and Alteon discloses this limitation. In the previous limitation, packets are sorted by type with the first type containing data. To finish processing the data packets, data is written into memory.

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

Ex.1005, Erickson at 4:53-5:14, Fig. 4, *see also id.* at Fig. 3 (illustrating that I/O device adapter 314 sends data to applications 302 and 304 that reside within the memory of the host computer), *see also id.* at 6:1-41, 8:17-37; and Ex.1003, Horst Decl. at Section V.H.1 (explaining using pointer to write data directly from I/O device to host memory).

Pet. at 56-57 (see also Pet. at 56-58 (no reference to Tanenbaum)).

No Motivation to Combine Erickson, Tanenbaum, and Alteon

126. *First*, as described above, *Erickson* is primarily concerned with checksum and packet generation in the transmit path. *Alteon*'s copy processing using a single interrupt occurs on the receive path. While complete systems include both receive and transmit path processing and that processing must handle the protocols headers it receives, the two sides are independent, have different challenges, and require different functionality. A POSITA looking to improve receive side interrupt processing would not automatically look to transmit side functionality.

127. Petitioner's alleged motivation of increased performance is generic to the point of being inconsequential. *Id.* "Increased performance" would be true for just about every advance in networking technology and would not motivate a POSITA to make the particular modification claimed in the '241 Patent. It is only through hindsight that a person would select the references identified by the Petitioner and combine them as Petitioner describes. *Id.*

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Ex. 2026 at 63-64

Erickson and Alteon Are Fundamentally Incompatible

United States Patent (19)
Erickson et al.

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

[75] Inventors: Greg R. Erickson; Douglas E. Handley, both of Poway; P. Kelly Muller; Curtis B. Stanley, both of Diego, all of Calif.

[73] Assignee: NCR Corporation, Dayton, Ohio

[21] Appl. No.: 877,678

[22] Filed: Dec. 21, 1998

[51] Int. Cl. G06F 9/00

[52] U.S. Cl. 406

[58] Field of Search 359/821, 359/829, 832, 846, 882, 284, 309

[56] References Cited

U.S. PATENT DOCUMENTS

4,589,863	6/1986	Dahl et al.	79
4,777,289	10/1988	Boomer et al.	36
5,016,161	5/1991	Van Lee et al.	29
5,016,166	5/1991	Van Lee et al.	29
5,127,098	8/1992	Rosenthal et al.	21
5,208,587	1/1994	Shimozono et al.	26
5,420,687	8/1995	Koel et al.	38
5,548,378	8/1996	Hirayama	29
5,553,244	8/1996	Reynolds et al.	39
5,662,481	8/1997	Probst	39(5)
5,671,642	9/1997	Finney et al.	38

FOREIGN PATENT DOCUMENTS

53148 7/1993 European Pat Off

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Remote Virtual Memory Resizing", by Shin-Yuan Tsai, David P. Anderson, in *Software-Practice & Experience* 24(3), 251-267 (Mar., 1994).

INTEL Ex.1005.001

Within the udpscript procedure described above, the nextid() function provides a monotonically increasing 16-bit counter required by the IP protocol. The ipchecksum() function performs a checksum for the IP 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 likelihood, the adapter would have a very limited knowledge of the user process' virtual address space, probably only knowing how to map virtual-to-physical for a very limited range, maybe as small as a single page. Pages in the user process' virtual address space for such buffers would need to be fixed. The udpscript procedure would need to be enhanced if the user data were allowed to span page boundaries. The udpchecksum() procedure generates a checksum value for both the UDP Header 708 plus the user data (not shown).

Erickson (Ex. 1005)

Ex. 1005 at 8:20-24

Alteon (Ex. 1033)

Gigabit Ethernet
Technical Brief

Achieving End-to-End Performance

Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119
1-408-574-5500

Repeating Data Steps 1, 2, and 3, the NIC puts as many packets as it can into the buffer until one of the following occurs:

- It runs low on buffer space
- The interrupt timer expires

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

Ex. 1033.023

-01392 PO Demonstrative, 29

Erickson and Alteon Are Fundamentally Incompatible

130. Even assuming the differences between *Erickson's* send-side and *Alteon's* receive-side can be handled, *Alteon's* interrupt timer “allows a single interrupt to be issued for multiple data packets that are sent into the operating system buffer space.” (Ex. 1033.023.) In other words, *Alteon* explains that its system “puts as many packets as it can into the buffer” until the interrupt timer expires. Since the buffer is “provided by the host operating system,” *Alteon's* network interface device must have extended knowledge of the user process’ virtual address space. Due to express limitations in *Erickson's* network adapter, however, this is not possible in the system described by *Erickson*, which operates on a datagram-by-datagram (and page-by-page) basis. *Erickson's* architecture is therefore fundamentally incompatible with *Alteon's* multi-page (and multi-packet) implementation and would have to be significantly reworked to support *Alteon's* interrupt timers and single interrupt copying process for multiple packets.

Ex. 2026 at 66

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Erickson and Alteon Are Fundamentally Incompatible



Robert Horst
Petitioner's Expert

10 Q. Why is it that the adapter probably only knows
11 how to map virtual-to-physical addresses for as small as
12 a single page?

13 A. The adapter is able to read and write that
14 space. And if you allowed it to read and write anything
15 in the processor, it could corrupt important data
16 structures or cause a security hole.

Ex. 2028 (Horst Dep. Tr.) at 76:10-16

7 Q. And Ericsson doesn't explain how to enhance the
8 UDP script procedure in order to span page boundaries,
9 correct?

10 MR. CONSTANT: Objection, form.

11 A. Ericsson doesn't describe that, but prior to
12 Ericsson there have been plenty of I/O controllers that
13 could transfer to multiple physical pages.

14 Q. (By Mr. Mack) But the UDP script procedure,
15 the pseudocode that's shown at the bottom of column 7,
16 that procedure only supports single pages, correct?

17 A. That pseudocode only shows how you would
18 support single pages.

Ex. 2028 (Horst Dep. Tr.) at 78:7-18

The Combination Fails to Suggest Dividing the Data into Multiple Segments and Prepending a Packet Header to Each Segment (Claims 9 and 17)

9. A method for communicating information over a network, the method comprising:
obtaining data from a source in memory allocated by a first processor;
dividing the data into multiple segments;
prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and
transmitting the packets to the network.

17. A method for communicating information over a network, the method comprising:
providing, by a first mechanism, a block of data and a Transmission Control Protocol (TCP) connection;
dividing, by a second mechanism, the block of data into multiple segments;
prepending, by the second mechanism, an outbound packet header to each of the segments, thereby forming an outbound packet corresponding to each segment, the outbound packet header containing an outbound media access control layer header, an outbound Internet Protocol (IP) header and an outbound TCP header, wherein the prepending of each outbound packet header occurs without an interrupt dividing the prepending of the outbound media access control layer header, the outbound (IP) header and the outbound TCP header; and
transmitting the outbound packets to the network.

The Combination Fails to Suggest Dividing the Data into Multiple Segments and Prepending a Packet Header to Each Segment (Claims 9 and 17)

An example of user process programming that triggers the I/O device adapter is set forth below:

```

senduserdatagram (void *USERDATA_ADDRESS,
                 int      USERDATA_LENGTH)
/* wait till adapter available */
{
    while (vhl_p->STATUS != IDLE) {};
    vhr_p->STARTINGADDRESS = USERDATA_ADDRESS;
    vhr_p->LENGTH          = USERDATA_LENGTH
/* trigger adapter */
    vhr_p->GO              = 1;
/* wait till adapter completes */
    while (vhl_p->STATUS == BUSY) {};
}

```

Ex. 1005 at 7:21-33

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

An example of programming that triggers the I/O device adapter is provided below:

```

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

```

Ex. 1005 at 7:50:64

141. The combination of *Erickson* and *Tanenbaum* does not show or suggest this claimed feature. In *Erickson*, the pseudocode procedures that prepare a UDP datagram for transmission require the memory address and length of the user data to be provided as two parameters. (Ex. 1005 at 7:21-8:2.) As such, it is clear that the both the `senduserdatagram()` and `udpscript()` procedures in *Erickson* operate on a single UDP datagram only. There is no built-in functionality within *Erickson's* I/O device adaptor for dividing a large block of user data into multiple segments and prepending a packet header to each of the segments.

Ex. 2026 at 71

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

The Combination Fails to Suggest Dividing the Data into Multiple Segments and Prepending a Packet Header to Each Segment (Claims 9 and 17)

142. Petitioner alleges that the combination of *Tanenbaum* and *Erickson* discloses this limitation because *Tanenbaum* describes a TCP entity that can accept user data streams and break them up into pieces, sending each piece as a separate IP datagram. (Petition at 73.) But this traditional TCP functionality occurs on the host and not on the network interface device (*i.e.*, the claimed “second [processor/mechanism]”). The combination would therefore not show or suggest dividing user data into segments and prepending a packet header to each of the segments by the network interface device. At best, the combination would show traditional segmentation by the host protocol stack and then the prepending of a single header to a single UDP datagram. In other words, there is no mechanism for prepending packet headers to each of a number of divided data segments in *Erickson*’s I/O device.

UNITED STATES PATENT AND TRADE

BEFORE THE PATENT TRIAL AND A

INTEL CORP. and
CAVIUM, INC.,*Petitioners,*

v.

ALACRITECH INC.,

*Patent Owner.*Case IPR2017-01392¹
U.S. Patent 7,337,241CORRECTED PATENT OWNER’S P
DECLARATION OF KEVIN ALME

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Ex. 2026 at 71-72

The Combination Fails to Suggest Dividing the Data into Multiple Segments and Prepending a Packet Header to Each Segment (Claims 9 and 17)

143. Because *Erickson*'s network adapter has "very limited knowledge" of the user process' virtual address space (and operates on fixed, single pages, as described above), the modification proposed by Petitioner and its expert would be technically infeasible and extremely difficult to implement without significantly changing *Erickson* to become a different system—one that supported a different memory access scheme and supported different protocols. For these reasons, the combination of *Erickson* and *Tanenbaum* would not enable a POSITA to make or use an implementation that supported prepending packet headers by the network interface device to multiple divided data segments without undue experimentation.

Ex. 2026 at 71-72

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,
Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Petitioner's Expert Manufactures Three Scripts Not Disclosed or Suggested by Erickson (Claims 9 and 17)

Erickson in view of Tanenbaum96 (Ground 2)

[9.2] dividing the data into multiple segments;

This alternative script builds on the single segment script, but spans the GO bit only once for a multi-segment send. As Erickson assumes, the user data length is within in single page. Ex.1005, Erickson at 8:22-24. The script would DMA one segment of data at a time from the block identified by the user data address pointer and length passed to the script. Alternatively, the adapter could DMA all of the data in one large transfer and transmit one segment at a time. The segmentation code is within the skills of a person of ordinary skill in the art in light of the disclosures by Tanenbaum.

It would have also been obvious to create this third TCP script. Like the second alternative script, this third alternative script builds on the single segment script, but spans the GO bit only once for a multi-segment send. Instead of transferring one segment of user data at a time, the adapter would DMA all of the user data in one large transfer identified by the address pointer and length. The adapter would then repeatedly extract one segment of data at a time, place it into a packet, and transmit. This segmentation code

Ex. 1003 at A-40

UNITED STATES PATENT AND TRADEMARK
OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION
Petitioner

v.

ALACRITECH, INC.
Patent Owner

Case IPR No. Unassigned
U.S. Patent No. 7,337,241
Title: FAST-PATH APPARATUS FOR RECEIVING DATA
TO A TCP CONNECTION

Declaration of Robert Horst, Ph.D. in Support
Petition for *Inter Partes* Review
of U.S. Patent No. 7,337,241

INTEL Ex.1003.001

Petitioner's Expert Manufactures Three Scripts Not Disclosed or Suggested by Erickson (Claims 9 and 17)



Robert Horst
Petitioner's Expert

16 Q. You also state at the bottom of A-78, you say
17 that "It would also be obvious to create the following
18 TCP script for Ericsson," and you go on to page A-77 --
19 sorry. That's A-39 of A-78, now on A-40 of A-78. You
20 referred to this as an alternative script. Do you see
21 that?

22 A. Yes.

23 Q. Okay. The alternative script is not disclosed
24 in Ericsson itself, though, right?

2 A. The TCP scripts are not disclosed in Ericsson.

3 Q. (By Mr. Mack) The TCP script is not disclosed;
4 but this alternative that you're describing at the top
5 of page A-40 is also not disclosed, right?

6 MR. CONSTANT: Objection to form.

7 A. The script itself was not disclosed, but the
8 combination of Ericsson and Tanenbaum would make a
9 script like this obvious.

10 Q. (By Mr. Mack) And then beginning in the next
11 paragraph you refer to a third TCP script. Do you see
12 that?

13 A. Yes.

14 Q. That's, yet, another script that's not
15 disclosed in Ericsson, right?

16 MR. CONSTANT: Objection, form.

17 A. Yes.

Ex. 2028 (Horst Dep. Tr.) at 98:16-99:17

-01392 PO Demonstrative, 38

No Expectation of Success Adapting a UDP Script to TCP



Robert Horst
Petitioner's Expert

16 Q. (By Mr. Mack) Right. The pseudocode that we
17 looked at earlier, that was a UDP script, right?
18 A. Yes.
19 Q. And TCP is a different protocol than UDP,
20 right?
21 A. Yes.
22 Q. TCP is connection oriented, right?
23 A. Yes.
24 Q. There's a three-way TCP handshake in order to
25 establish a TCP connection?
2 A. Yes.
3 Q. And how are UDP connections established?
4 A. UDP is connectionless.
5 Q. So would you say the UDP connection process is
6 simpler than the TCP connection process?
7 A. Yes.
8 Q. And UDP doesn't support retransmission, does
9 it?
10 A. The UDP protocol itself doesn't do
11 retransmission like TCP does.
12 Q. And UDP also doesn't support flow control, does
13 it?
14 A. It doesn't support the type of flow control
15 that TCP does, no.
16 Q. And TCP and UDP have different header sizes,
17 right?
18 A. I believe so, yes.

Ex. 2028 (Horst Dep. Tr.) at 96:16-97:18

-01392 PO Demonstrative, 39

No Expectation of Success Adapting a UDP Script to TCP



Robert Horst
Petitioner's Expert

4 Q. And UDP doesn't include sequence numbers in the
5 header, right?
6 A. Right.
7 Q. And TCP does include sequence numbers, right?
8 A. Yes.
9 Q. Are there any other differences between TCP and
10 UDP that you can think of?
11 A. TCP has a number of other features that are not
12 in UDP.
13 Q. Would you say that TCP is a more complicated
14 protocol than UDP?
15 A. Yes.

Ex. 2028 (Horst Dep. Tr.) at 98:4-15

No Expectation of Success Adapting a UDP Script to Work With TCP

Erickson only provides a working example of a *UDP script*. All these differences between UDP and TCP would require a POSITA to fundamentally redesign *Erickson* to include functionality not discussed in either reference. As part of the redesign, the memory access system would need to change to support the more frequent passing, and possibly updating of TCP state information. Without the claims as a guide, it is unclear what functionality a POSITA would put where in the system. In addition, because the TCP transmission mechanism is much more complex than UDP and must support congestion control in the form of a sliding transmission window, retransmissions, and ordered, reliable delivery, it would not have been at all apparent to a POSITA that a TCP implementation of *Erickson's* simplistic approach of using a local script to update header fields would be successful or operable to maintain and exchange the much more complex state information required by TCP.

Ex. 2026 at 73

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Secondary Considerations: Long-Felt Need

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality

Ex. 2031 at 1 (1993 IBM Research Division ACM SIGCOMM article)

1 Introduction

Many researchers have observed that the performance of remote applications have not kept pace with modern communication network speeds. Part of this imbalance is attributed to the protocols used by the remote applications, namely, UDP/IP and TCP/IP. It is now widely believed that the problems with these protocols are not inherent in the protocols themselves but in their particular implementations [Dalt93, Whet95]. The following factors are cited as contributors to the poor performance of UDP/IP and TCP/IP:

Ex. 2032 at 1 (1995 Univ. of Berkeley paper)

Secondary Considerations: Long-Felt Need

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to “bottlenecks” in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication pro-

Ex. 2033 at 1 (1990 IEEE article from AT&T Bell Labs)

Abstract

Server network performance is increasingly dominated by poorly scaling operations such as I/O bus crossings, cache misses and interrupts. Their overhead prevents performance from scaling even with increased CPU, link or I/O bus bandwidths. These operations can be reduced by redesigning the host/adaptor interface to exploit additional processing on the adapter. Offloading processing to the adapter is beneficial not only because it allows more cycles to be applied but also of the changes it enables in the host/adaptor interface. As opposed to other approaches such as RDMA, TCP offload provides benefits without requiring changes to either the transport protocol or API.

Ex. 2034 at 1 (2005 USENIX Conference paper from IBM)

Secondary Considerations: Long-Felt Need

148. The nexus between the long-felt need and the claimed invention is clear and direct. The aforementioned bottlenecks are all solved by the accelerated network processing technologies recited in the challenged claims. In addition, the technologies recited in the challenged claims alleviated processing demand on the host CPU in such a way as to permit faster network throughput and transmission/reception speeds—the precise long-felt but unresolved need in the industry outlined above.

Ex. 2026 at 76

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIEBUNAL

IN

ALACRITECH, INC. v. CAVIUM, INC.

Case

U.S. Patent and Trademark Office

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019852806

Alacritech Exhibit 2026

Secondary Considerations: Industry Praise

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

Ex. 2039 at 4 (2001 HP Labs research paper)

Secondary Considerations: Industry Praise

“The ANX 1500 is an evolutionary advancement of Alacritech’s long standing leadership in protocol acceleration, which is now applied not just to protocols, but to the optimization of all storage system interactions,” said Jeff Boles, a technology analyst with Taneja Group. “The ANX 1500 substantially augments the performance of existing NAS investments from the best place possible – from right in the customer’s Ethernet network – without reconfiguration or changes in management for the existing infrastructure. We think Alacritech is setting the stage for a next generation of solutions that will accelerate storage from outside the storage array, and more cost effectively share an investment in performance across many storage systems and clients. I’ve talked to early-stage customers using the product, and they believe it’s game-changing.”

Ex. 2040 at 3 (Shoreline Ventures Press Release)

-01392 PO Demonstrative, 46

Secondary Considerations: Industry Praise

152. The industry universally praised commercial embodiments of the features described in the challenged claims. HP found that the “Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed” and “achiev[e] lower processor utilization than native NT’s TCP/IP protocol stack for transmission of large enough messages.” (Ex. 2039 at ¶ 4.) HP found that the test performance of the Alacritech iNIC “raises challenging questions for [HP’s] choice of implementation technology because “the Alacritech approach is definitely better than our offload.” *Id.*

Ex. 2026 at 78

UNITED STATES PATENT AND TRADE
BEFORE THE PATENT TRIAL AND A

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019850280.6

Alacritech Exhibit 2026

Secondary Considerations: Failure of Others

To this day, TCP offload has never firmly caught on in the commercial world (except sometimes as a stopgap to add TCP support to immature systems [16]), and has been scorned by the academic community and Internet purists. This paper starts by analyzing why TCP offload has repeatedly failed.

Ex. 2041 at 2 (2003 HP Labs paper presented at HotOS IX conference)

TCP offload has been unsuccessful in the past for two kinds of reasons: fundamental performance issues, and difficulties resulting from the complexities of deploying TCP offload in practice.

Id. at 2

Secondary Considerations: Failure of Others

offload were mismatched to the applications in question.” *Id.* The TCP offload described above is a form of network processing offload that is described by the challenged claims, and this failure of others therefore has a direct nexus to the claimed inventions.

Ex. 2026 at 79

UNITED STATES PATENT AND TRADE
 BEFORE THE PATENT TRIAL AND AP

INTEL CORP. and
 CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

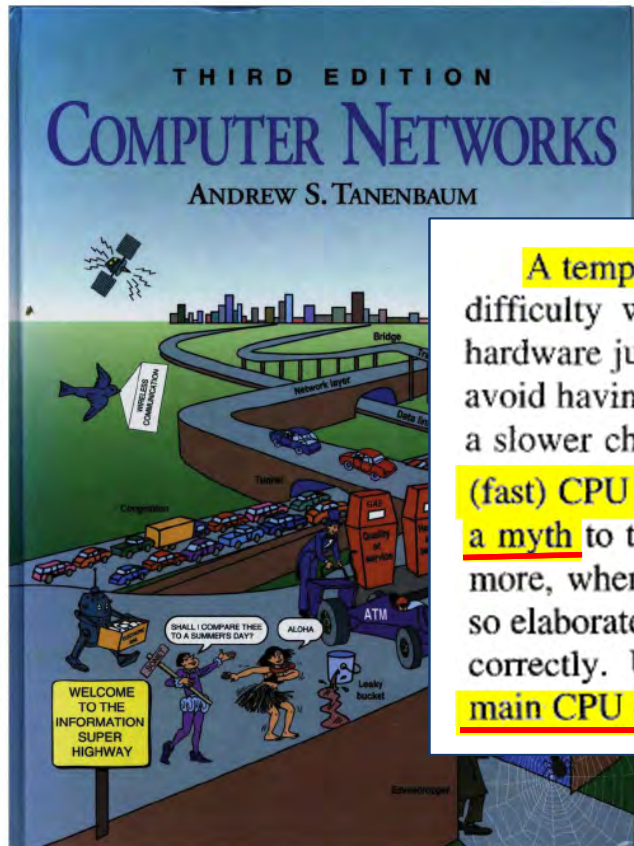
Patent Owner.

Case IPR2017-01392¹
 U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Secondary Considerations: Skepticism



INTEL Ex.1006.001

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.

Ex. 1006 at 588-589

Secondary Considerations: Skepticism

IP Storage and the CPU Consumption Myth

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

Abstract

This paper addresses a key issue that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2.

The accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

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

Ex. 2300 at 194 (2001 paper by Petitioner's expert, Dr. Horst)

Secondary Considerations: Skepticism

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

Networks customized to storage networking, such as Fibre 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 paper 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.

2. The Historical A

There are many historical examples of hardware accelerators to offload processor work from the main CPU. Some examples, such as network communications processors, have been successful, but others have fallen far short of unmet expectations.

Examples of front-end accelerators date from the early days of computing. In many systems, the primary I/O processor to offload the main CPU. However, it has become in vogue to use a different architecture to deliver a faster rate of technology change.

A specific recent example is the use of an I/O processor, such as an Intel 80199, to offload the main CPU from its attached I/O device. The idea is to serve as an I/O processor from its attached I/O device. It started, the i960 embedded processor, but its performance did not live up to expectations as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that hardware accelerators should have been used more extensively. However, the cost of embedded programming and testing every protocol worthy of a name is high, and it is difficult to stay afloat when the hardware is far from stable, and even when formally approved, there are often bug fixes. It seems extremely

as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Secondary Considerations: Skepticism

(Ex. 2300 at 194.) (emphasis added). This level of skepticism and teaching away is directly related to the heart of the claimed invention – offloading network processing, and therefore has a direct nexus to the challenged claims.

Ex. 2026 at 81

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROETH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019852380.6

Alacritech Exhibit 2026

'241 Patent – Contingent Motion to Amend (Proposed Claim 25 – Substitute for Claim 1)

Proposed Claim 25

25. (proposed substitute for claim 1) A method for network communication, the method comprising:

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

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

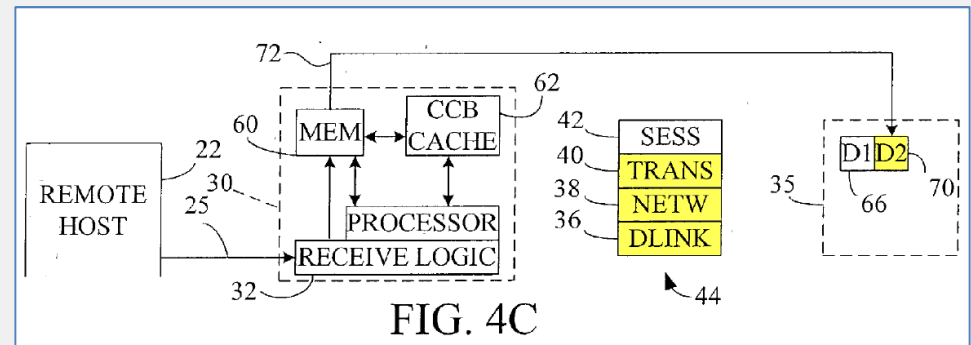
* * *

sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application running on a host computer without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination or to a host protocol stack running on the host computer.

Support for Proposed Claim 25 (Substitute for Claim 1)

[0057] All received message frames which have been determined by the CPD hardware assist to be fast-path candidates are examined 53 by the network microprocessor or INIC comparator circuits to determine whether they match a CCB held by the CPD. Upon confirming such a match, the CPD removes lower layer headers and sends 69 the remaining application data from the frame directly into its final destination in the host using direct memory access (DMA) units of the CPD. This operation may occur immediately upon receipt of a message packet, for example when a TCP connection already exists and destination buffers have been negotiated, or it may first be necessary to process an initial header to acquire a new set of final destination addresses for this transfer. In this latter case, the CPD will queue subsequent message packets while waiting for the destination address, and then DMA the queued application data to that destination.

Ex. 2021 at ¶ [0057]



Ex. 2021 at Fig. 4C

packet. The processor 55 checks for a match between the hash and each CCB that is stored in the cache 62 and, finding a match, sends the data (D2) 70 via a fast-path directly to the destination in storage 35, as shown by arrow 72, bypassing the session layer 42, transport layer 40, network layer 38 and data link layer 36. The remaining data packets from the message can also be sent by DMA directly to storage, avoiding the relatively slow protocol layer processing and repeated copying by the CPU stack 44.

Ex. 2021 at ¶ [0061]

Proposed Claim 25 (Substitute for Claim 1) Is Valid Over the Prior Art

Table 4. Steps in Data Reception using Second Generation NIC

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

Ex. 1033.021

Gigabit Ethernet Technical Brief

Achieving End-to-End Performance



Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119
1-408-574-5500

First Edition
September 1996

INTEL Ex.1033.001

Proposed Claim 25 (Substitute for Claim 1) Is Valid Over the Prior Art

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01392¹
U.S. Patent No. 7,337,241

PATENT OWNER'S EXHIBIT 2305

DECLARATION OF KEVIN ALMEROOTH, PH.D.

IN SUPPORT OF PATENT OWNER'S REPLY TO PATENT OWNER'S
CONTINGENT MOTION TO AMEND UNDER 37 C.F.R. § 42.121

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as petitioner in this proceeding.

Alacritech Exhibit 2305

34. In fact, *Erickson* teaches the opposite of the claimed invention. For example, Figure 4 of *Erickson* and the corresponding description of Figure 4 teach that “[t]he information coming from interconnect 410 is routed *directly* to a process 402 or 404.” Ex. 1005 at 5:8-11. The word “directly” indicates that the information from interconnect 410 is not processed before it is routed to process 402 or 404, and is instead transferred in its entirety with the header information.

Similarly, Figure 5 and its corresponding written description teaches that in *Erickson* “endpoint table 514 points to various protocol data 518 in the memory 512 in order to accommodate multiple communication protocols . . . which indicate how data or information is to be transferred from the memory 512 of the I/O device adapter to the portion of main memory 502 associated with a user process.” Ex. 1005 at 5:59-67. In other words, the “data or information” in memory 512, including the “protocol data,” is transferred to the main memory associated with the user process. One skilled in the art would understand “protocol data” includes header information. *Erickson* thus teaches sending header information to the destination, which is the opposite of what the substitute claims require.

Ex. 2305 at 16

-01392 PO Demonstrative, 57

'241 Patent – Contingent Motion to Amend (Proposed Claim 33 – Substitute for Claim 9)

Proposed Claim 33

33. (proposed substitute for claim 9) A method for communicating information over a network, the method comprising:

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

dividing the data into multiple segments;

prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment ... ; and

transmitting the packets to the network, **wherein the dividing, prepending, and transmitting occur without the second processor generating an interrupt to the first processor.**

** substantially similar amendments to remaining independent claim 17
(proposed substitute claim 41)*

Support for Proposed Claim 33 (Substitute for Claim 9)

35 by writing to a response buffer. Thus, fast-path transmission of data communications also relieves the host CPU of per-frame processing. A vast majority of data transmissions can be sent to the network by the fast-path. Both the input and output fast-paths attain a huge reduction in interrupts by functioning at an upper layer level, i.e., session level or higher, and interactions between the network microprocessor and the host occur using the full transfer sizes which that upper layer wishes to make. For fast-path communications, an interrupt only occurs (at the most) at the beginning and end of an entire upper-layer message transaction, and there are no interrupts for the sending or receiving of each lower layer portion or packet of that transaction.

Ex. 2021 at ¶ [0064]

Proposed Claim 33 (Substitute for Claim 9) Is Valid Over the Prior Art

An example of user process programming that triggers the I/O device adapter is set forth below:

```

senduserdatagram (void *USERDATA_ADDRESS,
                  int    USERDATA_LENGTH)
/* wait till adapter available */
{
    while (vhl_p->STATUS != IDLE) {};
    vhr_p->STARTINGADDRESS = USERDATA_ADDRESS;
    vhr_p->LENGTH           = USERDATA_LENGTH
/* trigger adapter */
    vhr_p->GO                = 1;
/* wait till adapter completes */
    while (vhl_p->STATUS == BUSY) {};
}
    
```


Ex. 1005 at 7:21-33

```

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

Ex. 1005 at 7:50:64

-01392 PO Demonstrative, 60



US005768618A

United States Patent (19) Patent Number: **5,768,618**
 Erickson et al. 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: **Greg R. Erickson; Douglas E. Hambley, both of Poway; P. Keith Muller; Curtis B. 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.⁵ **G06F 15/02**

(52) U.S. Cl. **395/829**

(58) **Field of Search** **395/821, 823, 395/829, 832, 846, 852, 284, 309, 500, 473**

References Cited

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boemer et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/878
5,016,166	5/1991	Van Loo et al.	395/874
5,127,008	6/1992	Rosenfeld et al.	718/002
5,280,587	1/1994	Shimodaira et al.	395/880
5,401,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Noronca et al.	395/280
5,647,481	6/1997	Pedersen	950/850
5,671,442	9/1997	Fenney 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 Tsou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

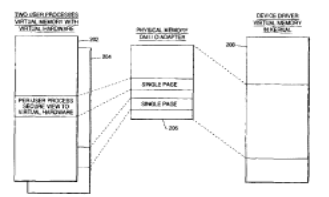
"The DASH Local Kernel Structure" by David P. Anderson and Shin-Yuan Tsou, Report No. UCB/CSD 88/463, Nov. 7, 1988, Computer Science Division (ERCS), University of California, Berkeley 94720.

"A Users' Guide to PCL—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. Shunkel, 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 10596.

19 Claims, 7 Drawing Sheets



INTEL Ex.1005.001

Proposed Claim 33 (Substitute for Claim 9) Is Valid Over the Prior Art

UNITED STATES PATENT AND TRADEMARK OFFICE
 BEFORE THE PATENT TRIAL AND APPEAL BOARD

CAVIUM, INC.,
 Petitioners,

v.

ALACRITECH INC.,
 Patent Owner.

Case IPR2017-01392¹
 U.S. Patent No. 7,337,241

PATENT OWNER'S EXHIBIT 2305
DECLARATION OF KEVIN ALMEROOTH, PH.D.
 IN SUPPORT OF PATENT OWNER'S REPLY TO PATENT OWNER'S
 CONTINGENT MOTION TO AMEND UNDER 37 C.F.R. § 42

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined petitioner in this proceeding.

Alacritech Exhibit 2305

39. I also disagree with Petitioner that “all processing to generate headers for packets to be sent from the network interface device of Erickson is performed by the processing capability of Erickson’s network interface device with no reason to interrupt the processing of the host computer requesting transmission.” Opp. at 16. *Erickson* teaches and suggests reasons why the host would need to be interrupted to perform the dividing, prepending, or transmitting steps. For example, *Erickson* teaches that within its “udpscript procedure” there is a “nextid() function” which “provides a monotonically increasing 16-bit counter.” Ex. 1005 at 8:10-13. One skilled in the art would understand that the host processor and the network interface processor would want to maintain consistency between the count, and that to do so the network interface processor would issue interrupts to the host processor during these steps.

40. Numerous interrupts would also be generated by *Erickson’s* system at other times. See *supra* ¶ 29. For example, each time a packet is transmitted, the “senduserdatagram” process must be called. Before calling this process, *Erickson* explains that the “user process” holds the user data of the datagram and certain pointers, memory addresses, and “virtual registers” must be set by the user process. Ex. 1005 at 7:5-18. This would require triggering multiple interrupts to the host after each packet is transmitted in order to configure the “senduserdatagram” process for the next packet. *Erickson*, even when combined with *Yanovich*

Ex. 2305 at 18-19

-01392 PO Demonstrative, 61

INTEL CORPORATION

v.

ALACRITECH, INC.

Case Nos.: IPR2017-01405, IPR2017-01735,
IPR2018-00336

U.S. Patent No. 7,124,205 B2

Patent Owner's Demonstratives

Hearing: September 13, 2018

Overview of the '205 Patent

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

(10) **Patent No.:** US 7,124,205 B2
 (45) **Date of Patent:** Oct. 17, 2006

(54) **NETWORK INTERFACE DEVICE THAT**
 5,485,579 A 1/1996 Hiz et al. 395/200.12

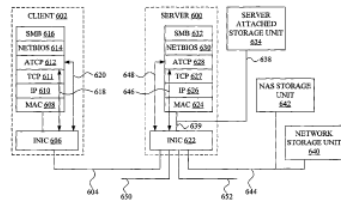
NETWORK INTERFACE DEVICE THAT FAST-PATH PROCESSES SOLICITED SESSION LAYER READ COMMANDS

Inventors: Peter K. Craft, San Francisco, CA (US); Clive M. Philbrick, San Jose, CA (US); Laurence B. Boucher, Saratoga, CA (US)

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

4,366,538 A	12/1982	Johnson et al.	364,200	processor stack of the host performs no network layer or transport layer processing. Other data transfers are, however, handled in a slow-path by the host protocol stack. In one embodiment, the host protocol stack has an ISCSI layer, but a response to a solicited ISCSI read request command is nevertheless processed by the network interface device in fast-path. In another embodiment, an initial portion of a response to a solicited command is handled using the dedicated fast-path and then after an error condition occurs a subsequent portion of the response is handled using the slow-path. The interface device uses a command status message to communicate status to the host.
4,589,063 A	5/1986	Shah et al.	710/8	
4,991,133 A	2/1991	Davis et al.	364/900	
5,056,058 A	10/1991	Hirata et al.	364/900	
5,058,110 A	10/1991	Beach et al.	370/85.6	
5,097,442 A	3/1992	Ward et al.	365/78	
5,165,131 A	11/1992	Row et al.	395/200	
5,212,778 A	5/1993	Daly et al.	395/400	
5,280,477 A	1/1994	Trapp	370/85.1	
5,289,580 A	2/1994	Latif et al.	395/275	
5,303,344 A	4/1994	Yokoyama et al.	395/200	
5,412,782 A	5/1995	Hausman et al.	395/250	
5,418,912 A	5/1995	Christmann	709/234	
5,448,566 A	9/1995	Richter et al.	370/94.1	

36 Claims, 25 Drawing Sheets



Ex. 1001 (“‘205 Patent”)

(57)

ABSTRACT

A network interface device connected to a host provides hardware and processing mechanisms for accelerating data transfers between the host and a network. Some data transfers are processed using a dedicated fast-path whereby the protocol stack of the host performs no network layer or transport layer processing. Other data transfers are, however, handled in a slow-path by the host protocol stack. In one embodiment, the host protocol stack has an ISCSI layer, but a response to a solicited ISCSI read request command is nevertheless processed by the network interface device in fast-path. In another embodiment, an initial portion of a response to a solicited command is handled using the dedicated fast-path and then after an error condition occurs a subsequent portion of the response is handled using the slow-path. The interface device uses a command status message to communicate status to the host.

‘205 Patent, Abstract

OSI Model

context of the Internet and other telecommunication or computing systems. The Open Systems Interconnection model (or “OSI model”) is one well known example, describing a seven layer stack where a particular layer serves the layer above it and is served by the layers below it. The seven layers of the OSI model are:

- Layer 7: Application Layer
- Layer 6: Presentation Layer
- Layer 5: Session Layer
- Layer 4: Transport Layer
- Layer 3: Network Layer
- Layer 2: Data Link Layer
- Layer 1: Physical Layer

UNITED STATES PATENT AND TRADEMARK OFFICE
 BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
 CAVIUM, INC.,
Petitioners,

-v.-

ALACRITECH INC.,
Patent Owner.

Case IPR2017-01405¹
 U.S. Patent 7,124,205

**CORRECTED PATENT OWNER’S EXHIBIT 2026
 DECLARATION OF KEVIN ALMEROOTH, PH.D.**

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined petitioner in this proceeding.

Alacritech Exhibit 2026

Ex. 2026 at 21-22

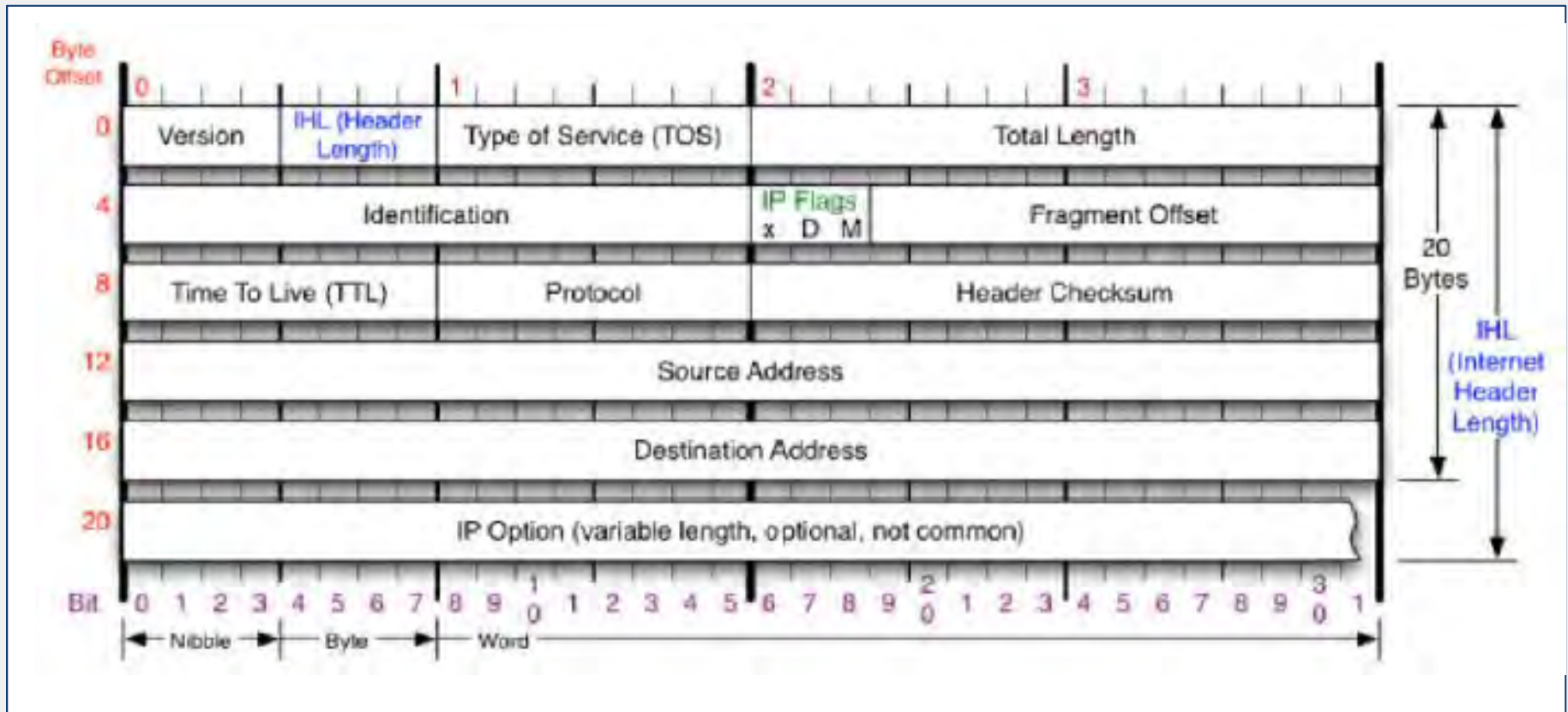
Layer 3: Network Layer

61. The Internet Protocol (or “IP”) is an example of a well-known network (layer 3) protocol. IPv4 was published as RFC 760 in January 1980 while its successor IPv6 was published as RFC 2460 in December 1998. The IP protocol describes a set of rules for dividing a message into multiple parts (called “IP packets”) and then transmitting those packets from an IP sender to an IP destination across multiple routers or other links in a computer network. Each packet of information includes an IP address for its destination, analogous to sending a letter through the mail by placing the letter inside an envelope that has the recipient’s postal address printed on it. The format of an IP header is depicted below:

Ex. 2026 at 22-23

-01405 PO Demonstrative, 4

IP Header



Ex. 2042

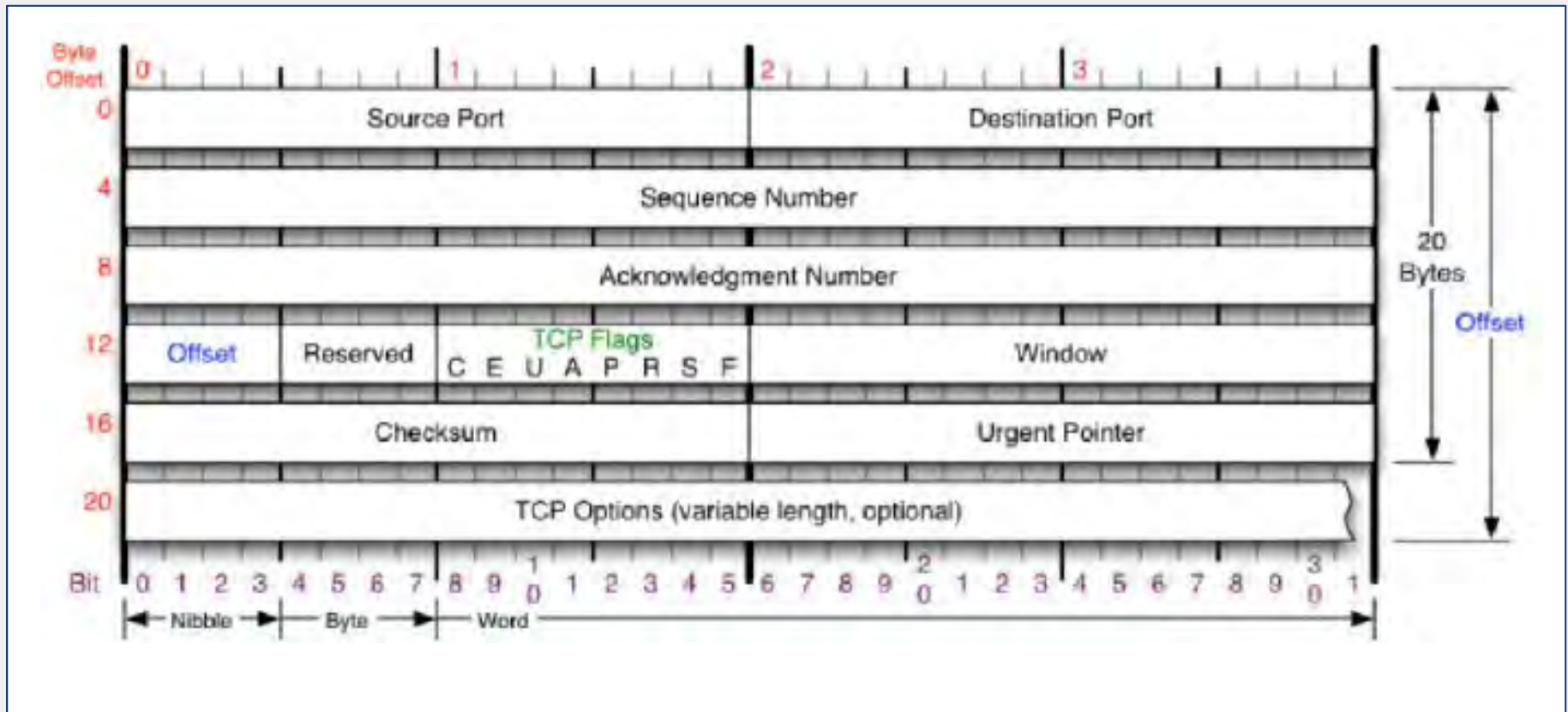
Layer 4: Transport Layer

earlier rule set published in December 1974 as RFC 675. TCP is an example of a transport (layer 4) protocol in the OSI model. TCP is responsible for adding reliability and ordering to the stream of network information—for example, the packets of information sent using IP as the network-layer protocol may not arrive at the destination in the same order intended by the sender of the message. TCP sets rules for breaking up and transmitting the message so that the recipient is able to reliably receive and reassemble the message. Another common analogy from the physical world is the example of sending a multi-page letter through the mail by separately numbering each page and mailing it in its own envelope. IP, like the

Ex. 2026 at 24-25

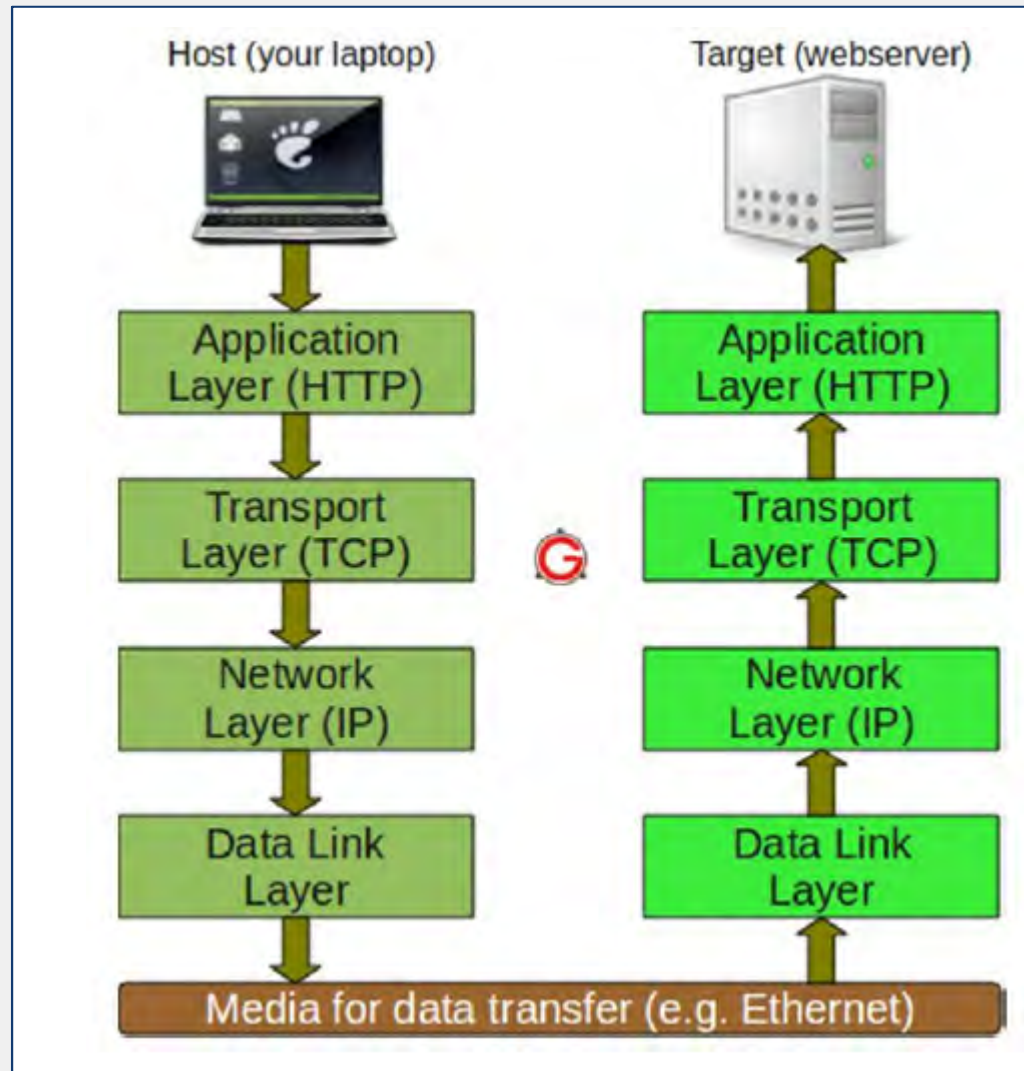
-01405 PO Demonstrative, 6

TCP Header



Ex. 2042

Message Processing Under TCP/IP



Ex. 2026 at 26-27; Ex. 2043

Message Processing Under TCP/IP

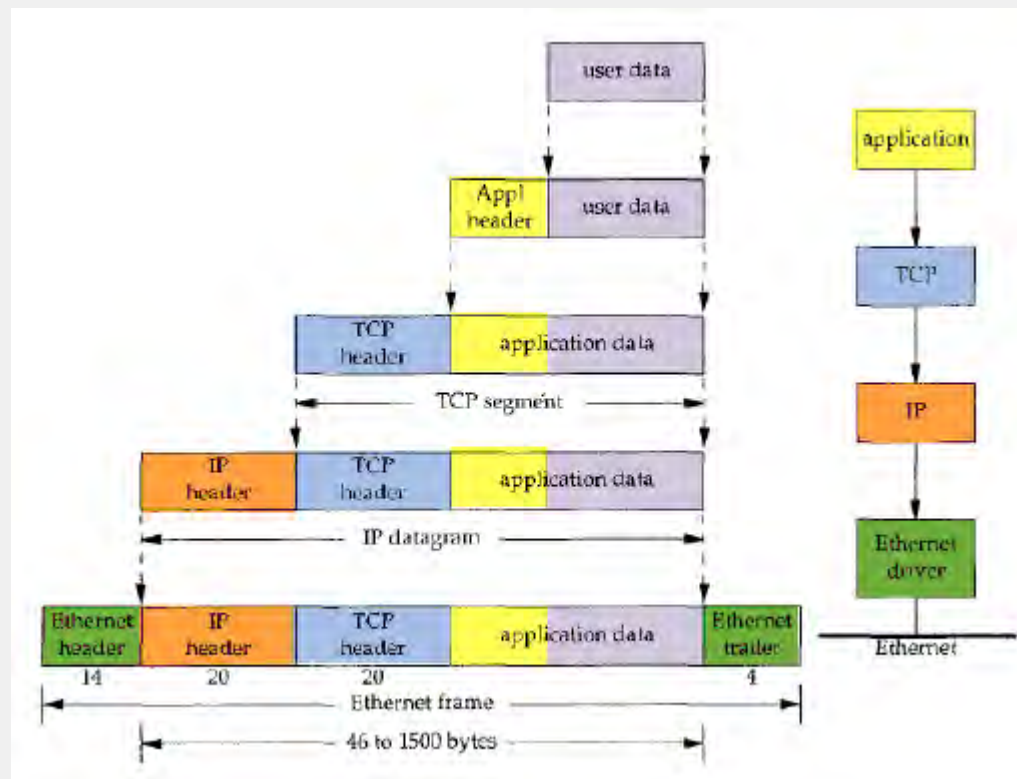
65. Much of this processing is typically handled by the CPU. Thus, sending and receiving data over a network can negatively impact the CPU's ability to perform other functions, particularly as the volume of data sent or received increases. For the purposes of this case, the manner by which the CPU handles the required protocol processing—i.e., the specific software steps it takes to perform the needed TCP and IP processing—is immaterial. At most, it is sufficient that the CPU does perform or is capable of performing that processing, whether through software included as part of the operating system or through some other means.

Ex. 2026 at 27-28

Overview of the '205 Patent

accelerated. Conventionally such data is divided into packets for transportation over the network, with each packet encapsulated in layers of control information that are processed one layer at a time by the CPU of the receiving computer. Although the speed of CPUs has constantly

Ex. 1001, '205 Patent, 1:49-52



Ex. 1008, Figure 1.4

Overview of the '205 Patent

68. On the receiving side, the receiving host generally performs the reverse of the sending process, beginning with receiving the bits from the network. Headers are removed, one at a time, and the received data is processed, in order, from the lowest (physical) layer to the highest (application) layer before transmission to a destination within the receiving host (*e.g.*, to the operating system space where the received data may be used by an application running on the receiving host). (Ex. 1001 at 1:49-52.) Each layer of the receiving host recognizes and manipulates only the headers associated with that layer, since to that layer the higher layer header data is included with and indistinguishable from the payload data.

Ex. 2026 at 29-30

-01405 PO Demonstrative, 11

Overview of the '205 Patent

69. Because the processing of each layer typically involves a copy and data manipulation operation (for example a checksum generation or validation operation), the host CPU must be “interrupted” at least one time per layer in order to process the data and construct (transmit side) or deconstruct (receive side) the packet. An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. When the host CPU is interrupted, it generally must stop all other tasks it is currently working on, including tasks completely unrelated to the network processing. Frequent interrupts to the host CPU can be very disruptive to the host system generally and cause system instability and degraded system performance.

Ex. 2026 at 30

-01405 PO Demonstrative, 12

Overview of the '205 Patent

70. The invention of the '205 Patent includes a “fast-path” where the host CPU is relieved of certain TCP/IP processing, which is instead performed by the INIC.

Ex. 2026 at 30

72. The claimed arrangement allows for enhanced network and system performance, a stark reduction or elimination of interrupts seen by the host CPU, faster data throughput, increased system stability, and an overall better user experience.

Ex. 2026 at 31

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

1. An apparatus comprising:

a host computer having a protocol stack and a destination memory, the protocol stack including a session layer portion, the session layer portion being for processing a session layer protocol; and

a network interface device coupled to the host computer, the network interface device receiving from outside the apparatus a response to a solicited read command, the solicited read command being of the session layer protocol, performing fast-path processing on the response such that a data portion of the response is placed into the destination memory without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.

3. The apparatus of claim 1, wherein the session layer protocol is **ISCSI**.

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

8. A method, comprising:

issuing a read request to a network storage device, the read request passing through a network to the network storage device;

receiving on a network interface device a packet from the network storage device in response to the read request, the packet including data, the network interface device being coupled to a host computer by a bus, the host computer having a protocol stack for carrying out network layer and transport layer processing;

performing fast-path processing on the packet such that the data is placed into a destination memory without the protocol stack of the host computer doing any network layer processing on the packet and without the protocol stack of the host computer doing any transport layer processing on the packet;

receiving on the network interface device a subsequent packet from the network storage device in response to the read request, the subsequent packet including subsequent data; and

performing slow-path processing on the subsequent packet such that the protocol stack of the host computer does network layer processing and transport layer processing on the subsequent packet.

9. The method of claim 8, wherein the read request is in the form of a **SCSI command**, wherein the SCSI command is attached to a header in accordance with an iSCSI protocol.

10. The method of claim 8, wherein the read request is an **iSCSI read request**.

16. The method of claim 8, wherein the bus is a PCI bus, and wherein the read request is a **iSCSI read request**.

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

22. An apparatus comprising:
a host computer having a protocol stack and a destination memory; and
a network interface device coupled to the host computer, the network interface device receiving a first portion of a response to an ISCSI read request command, the first portion being processed such that a data portion of the first portion is placed into the destination memory on the host computer with the protocol stack of the host computer doing substantially no network layer or transport layer processing, the network interface device receiving a second portion of the response to the ISCSI read request command, the protocol stack of the host computer doing network layer and transport layer processing on the second portion.

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

24. The apparatus of claim 22, wherein the ISCSI read request command is passed from the host computer to the network interface device, the ISCSI read request command being accompanied by an indication of where the destination memory is located on the host computer.

25. The apparatus of claim 24, wherein the indication includes a scatter-gather list.

26. The apparatus of claim 24, wherein an indication of where the destination memory is located on the host computer is passed from the host computer to the network interface device, the indication being passed to the network interface device before the first portion of the response is received onto the network interface device.

27. The apparatus of claim 22, wherein the response to the ISCSI read request command is received onto the computer via a single cable, the computer also receiving other network

communications over the single cable, the other network communications not being ISCSI communications.

28. The apparatus of claim 22, wherein the host computer does exception handling as a consequence of the computer having received the second portion of the response.

29. The apparatus of claim 22, wherein the host computer does error handling as a consequence of the computer having received the second portion of the response.

30. The apparatus of claim 22, wherein an enclosure contains both the host computer and the network interface device.

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

31. An apparatus comprising:
a host computer having a protocol stack and a destination memory; and
means, coupled to the host computer, for receiving from outside the apparatus a response to **an ISCSI read request command** and for fast-path processing a portion of the response to the ISCSI read request command, the portion including data, **the portion being fast-path processed such that the data is placed into the destination memory on the host computer without the protocol stack of the host computer doing significant network layer or significant transport layer processing**, the means also being for receiving a subsequent portion of the response to the ISCSI read request command and for **slow-path processing the subsequent portion such that the protocol stack of the host computer does network layer and transport layer processing on the subsequent portion.**

32. The apparatus of claim **31**, wherein the network layer and transport layer processing done on the subsequent portion by the means includes error condition handling.

33. The apparatus of claim **31**, wherein the network layer and transport layer processing done on the subsequent portion by the means includes exception condition handling.

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

35. A host bus adapter that is adapted for sending an **ISCSI** solicited read request and for receiving a response in return, the host bus adapter also being adapted for coupling to a host computer that has a protocol stack, the protocol stack having an ISCSI layer, the host bus adapter being adapted for processing the response such that **a data portion of the response is placed into a memory on the host computer without the host computer doing any network layer or transport layer processing on the response.**

Challenged Claims: 3, 9-10, 16, 22, 24-33, 35, 36

36. A method, comprising:
sending from a host bus adapter an **ISCSI** solicited read request;
receiving onto the host bus adapter a response to the ISCSI solicited read request; and
the host bus adapter processing the response such that a **data portion of the response is placed into a destination memory on a host computer** that is coupled to the host bus adapter **without a protocol stack of the host computer doing any network layer processing on the response and without the host computer doing any transport layer processing on the response,** the protocol stack of the host computer having an ISCSI layer.

Overview of the Prior Art

Challenged Claims	103(a) References
3, 9-10, 16, 22, 27-33, 35-36	Thia + Satran I / Satran II
24-26	Thia + Satran I / Satran II + Carmichael

Overview of Thia

14

A Reduced Operation Protocol multiple-layer bypass architecture

Y.H. Thia (*)¹ and C.M. WoodsideNewbridge Networks, Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Abstract — The Reduced Operation Protocol Engine (ROPE) is a hardware implementation of the critical functions of a multiple-layer protocol stack, based on the “bypass concept” of a fast path for data transfer. The motivation for identifying this separate processing path is that it involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inter-layer operations such as queue and buffer management, context switching and movement of data across layers, all of which are a significant overhead. ROPE is intended to support high-speed bulk data transfer. The paper describes the design of a ROPE chip for the OSI Session and Transport layer protocols, using VHDL. The design is practical in terms of chip complexity and area, using current gate array technology, and simulation shows that it can support a data rate approaching 1 gigabit per second.

Keyword co
Keywords: 1

1 Introdu

The adv
rates, has sh
munications
quality-of-se
combination

of operating system overhead, protocol complexity, and per-socket processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementation of existing protocols [5, 35], parallel processing techniques [14, 21, 38], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offboard processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the gains.
- The complexity of the adaptor may be offset by the complexity of the protocol.

¹ This research

This paper presents a feasibility study for a new approach to hardware assistance. It combines the relatively simple operations needed for data transfer across multiple layers and provides a hardware “fast path” for them, which will be efficient for bulk data transfer. It is

Ex. 1015 (Thia) at 1

Ex. 1015 (Thia) at 2

to obtain the simulation throughput results presented in section 5. This was sufficient for our present purposes, to estimate the space complexity and timing of the chip, and the final step of generating a chip layout for fabrication and fault analysis was not performed. As

Ex. 1015 (Thia) at 8

This Is Inoperable and Not Enabled

81. Accordingly, This discloses (at best) an *inoperative* device, and is a non-enabling reference. (*See id.*) This fails to teach much of the subject matter of the Challenged Claims. This's feasibility study describes—at a high level—the idea of offloading presentation, session and transport layer processing to a Reduced Operation Protocol Engine (“ROPE”) chip, but does not disclose many of the implementation details required to enable the implementation, let alone enable it for use with TCP/IP.

UNITED STATES
BEFORE THE

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

Ex. 2026 at 38

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

This Is Inoperable and Not Enabled

82. This describes a bypass stack on the ROPE chip that provides a hardware “fast path” for bulk data transfer. (Ex. 1015.002-003.) This’s disclosure of the ROPE bypass architecture consists of bare flowcharts shown below, which lack implementation details necessary to implement the device:

UNITED STATES
BEFORE THE

CAVIUM, INC.,
 Petitioners,
 v.
 ALACRITECH INC.,
 Patent Owner.
 Case IPR2017-01405¹
 U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

00073-0000109854448.5

Alacritech Exhibit 2026

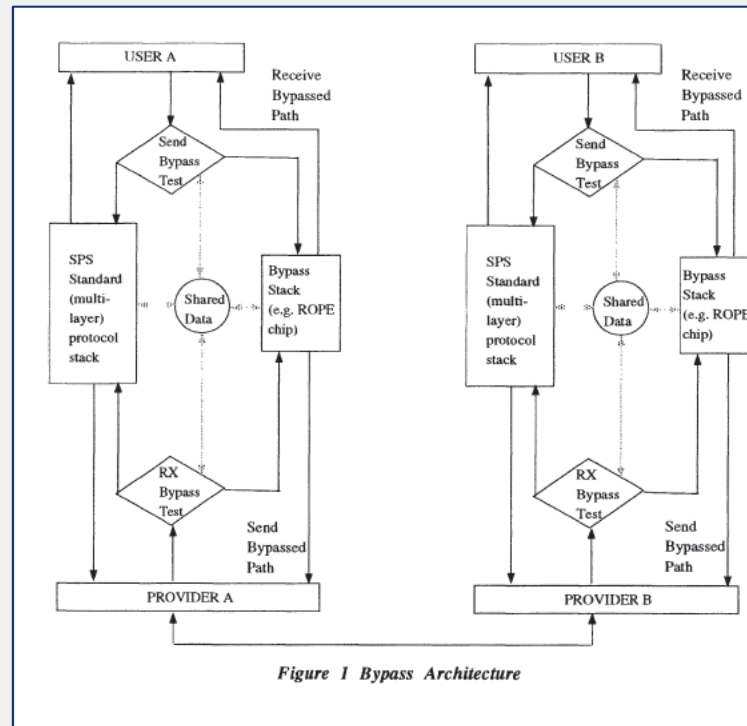


Figure 1 Bypass Architecture

Ex. 2026 at 38

Ex. 1015 (Thia) at 3

-01405 PO Demonstrative, 24

This Is Inoperable and Not Enabled

14

A Reduce multiple-l

Y.H. Thia

Newbridge N
Dept. of Syst

Abstract
critical functi
path for data
involves only
hardware. M
and buffer ma
are a signific
paper describ
using VHDL.
array technol
per second, is

Keyword cod
Keywords: N

The send bypass test identifies outgoing packets that are data packets in the data transfer phase. The receive bypass test matches the incoming PDU headers with a template that identifies the predicted bypassable headers. The bypass stack performs all the relevant protocol processing in the data transfer phase. The shared data are used to maintain state consistency between the SPS and the bypass stack, including window flow control parameters and connection identifiers. Whenever there is a change in the processing path between the SPS and the bypass stack, checks are performed to ensure that there are no outstanding packets in the current path, i.e. "no in-transit PDUs", before the change is made. A more detailed discussion on this is presented in another paper [33].

1 Introduction

The advent of Fibre Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-points of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-octet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementation of existing protocols [5, 25], parallel processing techniques [14, 21, 38], special part of the protocol f

The key problem
 Partitioning the fi
lead to a comple
offset the potenti
may be offloaded
protocol logic.

¹ This research was done w

Ex. 1015 (Thia) at 3

(Ex. 1015.003.) However, Thia does not address or disclose how the bypass test is implemented, what values of the incoming headers must match the template in order to identify predicted bypassable headers, or what the template even is.

Ex. 2026 at 41

Thia Does Not Disclose Network Layer Bypass

A Reduced Operation multiple-layer bypass

Y.H. Thia (*)¹ and C.M.

Newbridge Networks, Inc.,
Dept. of Systems and Comp

Abstract — The Reduce critical functions of a multipath for data transfer. The r involves only a small subset hardware. Multiple-layer by and buffer management, con are a significant overhead. F paper describes the design of using VHDL. The design is p array technology, and simula per second, in a connection

Keyword codes: C.2.2, B.4
Keywords: Network Proto

1 Introduction

The advent of Fibre Op rates, has shifted the perform munications processing in the quality-of-service guarantees combination of operating sys the data stream. To alleviate improved software implement [14, 21, 38], special protocol part of the protocol functions

The key problems associ

- Partitioning the functiona lead to a complex additi offset the potential gain t may be offloaded, but th protocol logic.

¹ This research was done while Dr. Th

Layer	Procedure	Bypass Chip	Host	Per-Octet (A)	Per-Packet (B)	Per-Group-Of-Packets Aggregated to Per-Packet for bulk data transfer (B)	Remarks
Presentation	Encoding	X		X			
	Encryption	X		X			
	Compression	X		X			
	Context Alteration		X			X	
Session	Synchronization Management		X			X	
	Token management		X			X	
Transport (Class 4)	Checksum (Optional)	X		X			
	Timer Management	X			X	X	Depends on Implementation
	Generation of ACK packets (Flow Control)	X				X	
	Resequencing	X			X		
All 3 layers	Header Construction	X			X		
	Header Decode	X			X		
	Buffer Management	X			X		Minimized (Simple scheme)
	Context Switching	X			X	X	Moved away from host OS
	Data Copying	With multiple-layer bypass, data Copying within layers is eliminated.				X	Use of dual-ported memory and DMA..

Table 1 Bypassable versus Non-bypassable functions

Ex. 1015 (Thia) at 6

This Does Not Perform Segmentation or Reassembly Within the Bypass Path

14

A Reduced Operation Protocol Engine (ROPE) for a

The scope of functions included in a bypass may be narrowly defined, or more extended. A bypass does not include fast connection setup but also does not interfere with it. There is no segmentation/reassembly within the bypass path, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. The Segmentation and Reassembly sublayer of the ATM adaptation layer is a good place for such functions [25].

Keyword codes: C.2.2, D.4.1
Keywords: Network Protocols, Data Communications Devices

transfer to/from the host system memory”), .009.) Notably, Thia’s ROPE chip does not perform any reassembly of the PDUs into larger data blocks.

(Ex.1015.014 (“There is no segmentation/reassembly within the bypass path”).) Thia’s bypass chip performs some of the protocol processing functions (such as validating checksums, decoding headers, etc.) but then provides the PDU to the host for reassembly.

Ex. 1015 (Thia) at 14

Ex. 2026 at 42

This Does Not Disclose Transport Layer Bypass

88. Accordingly, This does not contain any teaching or explanation of a dedicated NIA that implements or even tests TCP header bypass, let alone using flow keys and operation codes, and no explanation of how one would actually implement the theorized, generalized header prediction bypass stack in an actual NIC using the TCP protocol. Nor does This provide any teaching relating to the NIC performing packet data reassembly, such that certain header information is not passed to the host. These details are hardly trivial—they are exactly the prior art hurdles the '205 patent addressed and overcame.

Ex. 2026 at 43

UNITED STATES PATENT

BEFORE THE PATENT TRI

INTEL C
CAVIUM

Petition

ALACRITECH

Patent

Case IPR
U.S. Patent

CORRECTED PATENT

DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

00973-00001/9454448.5

Alacritech Exhibit 2026

Satran I

Internet-Draft
 <draft-satran-iscsi-00.txt>
 Expires 14 August 2000

J. Satran
 D.
 K.
 C. Sapun
 Cisco S
 M. To
 P.
 C.
 E. Zeidner
 SanGate
 February 2000

SCSI/TCP (SCSI over TCP)

Ex. 1056 ("Satran I") at 1

SCSI/TCP (SCSI over TCP)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Satran, Smith, Sapuntzakis, Meth

[Page 1]

Satran II

Internet-Draft
<draft-satran-iscsi-01.txt>
Expires January 10, 2001

J. S.
D.
K.
C. Sapuntzakis
Cisco Systems
Randy Heintz
Hewlett-Packard Co.
Efri Zeidner
SANGate
Paul Von Stamwitz
Adaptec
Luciano Dalle Ore
Quantum
July 10, 2000

iSCSI (Internet SCSI)

Ex. 1057 ("Satran II") at 1

iSCSI (Internet SCSI)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt> The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Acknowledgements

A large group of people contributed through their review, comments and valuable insights to the creation of this document - too many to mention them all. Nevertheless, we are grateful to all of them. We are especially grateful to those that found the time and

Carmichael

United States Patent [19] [11] **Patent Number:** 5,894,560
 Carmichael et al. [45] **Date of Patent:** Apr. 1

Patent Number: 5,894,560

[54] **METHOD AND APPARATUS FOR CONTROLLING I/O CHANNELS RESPONSIVE TO AN AVAILABILITY OF A PLURALITY OF I/O DEVICES TO TRANSFER DATA**

5,355,476 10/1994 Fukumura
 5,367,639 11/1994 Sodos
 5,386,532 1/1995 Sodos
 (List continued on next page.)

FOREIGN PATENT DOCUMENTS

0317481 5/1989 European Pat. Off. G06F 15/16
 0530543 3/1993 European Pat. Off. G06F 13/32
 0537401 4/1993 European Pat. Off. G06F 15/16
 0549924 7/1993 European Pat. Off. G06F 13/28
 9306553 4/1993 WIPO G06F 13/28

OTHER PUBLICATIONS

IBM Technical Disclosure Bulletin, Jan., 1994; DMA Controller Channel Interlocking; vol. 37, No. 1; pp. 337-342.
 IBM Technical Disclosure Bulletin, Feb., 1994; Scheme for Arithmetic Logic Unit and Data Path Isochronous Hardware; vol. 38, No. 2.
 Programming Interface for Bus Master IDE Channel 0.9; Jun. 14, 1994; Brad Hosler; Intel Corporation.

Primary Examiner—Christopher B. Shin
Attorney, Agent, or Firm—David K. Lucente

ABSTRACT

An apparatus and method for improving the performance of a computer system under the multi-tasking, multi-threaded operating particular, the invention provides an apparatus to chain contiguous DMA scatter gather sub blocks of a PRD table for channel 0 with contiguous DMA scatter gather sub blocks of a PRD table for channel 1, using a manager, while maintaining maximum media DMA block transfers are scheduled based on of data from the I/O device's buffer memory, using both media or network idle time as well I/O bus idle time. Near maximum aggregate multiple I/O buses and their associated device. The apparatus and method thus provides significant advantages over prior techniques by channel systems implemented with a single

7 Claims, 15 Drawing Sheets

[75] **Inventors:** Richard D. Carmichael, Longmont; Joel M. Ward; Michael A. Winchell, both of Fort Collins, all of Colo.

[73] **Assignee:** LSI Logic Corporation, Milpitas, Calif.

[21] **Appl. No.:** 08/702,998

[22] **Filed:** Aug. 26, 1996

Related U.S. Application Data

[63] Continuation of application No. 08/407,439, Mar. 17, 1995, abandoned.

[51] **Int. Cl.⁶** G06F 13/00

[52] **U.S. Cl.** 395/845; 395/827; 395/840

[58] **Field of Search** 395/825, 826, 395/827, 840, 841, 844, 845, 800, 600

References Cited

U.S. PATENT DOCUMENTS

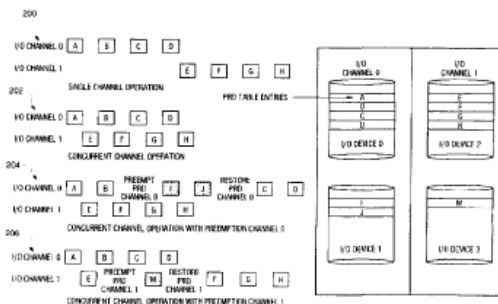
4,371,932	2/1983	Dinwiddie, Jr. et al.	364/200
4,782,439	11/1988	Borkar et al.	395/800
4,805,137	2/1989	Grant et al.	364/900
4,807,121	2/1989	Halford	364/200
4,821,170	4/1989	Bernick et al.	395/856
4,831,523	5/1989	Lewis et al.	364/200
5,015,160	5/1991	Lambeth et al.	395/844
5,031,097	7/1991	Katakami et al.	395/848
5,131,081	7/1992	MacKenna et al.	395/275
5,179,709	1/1993	Bailey et al.	395/725
5,185,876	2/1993	Nguyen et al.	395/425
5,206,933	4/1993	Farrell et al.	395/200
5,212,795	5/1993	Hendry	395/725
5,251,303	10/1993	Fogg, Jr. et al.	395/275
5,251,312	10/1993	Sodos	395/425
5,301,279	4/1994	Riley et al.	395/275
5,305,319	4/1994	Sowell	370/85.13

Ex. 1053 ("Carmichael") at 1

METHOD AND APPARATUS FOR CONTROLLING I/O CHANNELS RESPONSIVE TO AN AVAILABILITY OF A PLURALITY OF I/O DEVICES TO TRANSFER DATA

Inventors: Richard D. Carmichael, Longmont; Joel M. Ward; Michael A. Winchell, both of Fort Collins, all of Colo.

Carmichael at 1



All Independent Claims Involve Network-Layer Bypass

response such that a data portion of the response is placed into the destination memory without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.

'205 Patent, Claim 3 (through Claim 1)

processed such that the data is placed into the destination memory on the host computer without the protocol stack of the host computer doing significant network layer or significant transport layer processing, the means also being for receiving a subsequent portion of

'205 Patent, Claim 31 (whence Claims 32-33 depend)

performing fast-path processing on the packet such that the data is placed into a destination memory without the protocol stack of the host computer doing any network layer processing on the packet and without the protocol stack of the host computer doing any transport layer processing on the packet;

'205 Patent, Claims 9-10, 16 (through Claim 8)

an iSCSI layer, the host bus adapter being adapted for processing the response such that a data portion of the response is placed into a memory on the host computer without the host computer doing any network layer or transport layer processing on the response.

'205 Patent, Claim 35

first portion is placed into the destination memory on the host computer with the protocol stack of the host computer doing substantially no network layer or transport layer processing, the network interface device receiving a second portion of the response to the iSCSI

'205 Patent, Claim 22 (whence Claims 24-30 depend)

memory on a host computer that is coupled to the host bus adapter without a protocol stack of the host computer doing any network layer processing on the response and without the host computer doing any transport layer processing on the response, the protocol

'205 Patent, Claim 36

Petitioner Only Relies on Thia for Network-Layer Bypass

Specifically, as shown in Figure 1, Thia discloses a bypass architecture in which the NIA makes a determination whether to fast-path or slow-path the data.

Pet. at 46; see also Pet. at 57, 66-67, 82, 88, 90

No contention *Satran I, Satran II, or Carmichael* disclose network-layer bypass.

Petitioner's Argument as to Network-Layer Bypass

transfer phase.”). The bypass stack then performs all of the relevant protocol processing, including for the network and transport layers. *Id.*, .001 (“The paper describes the design of a ROPE chip for the OSI Session and Transport layer protocols, using VHDL.”), at .013 (“It can be concluded from this study that it is feasible to implement the bypass stack (at least for the transport and session layers) in VLSI and that the performance would be at least an order of magnitude higher than software protocol processing.”), at .004 (“**The advantage is increased further in cases where some layers, like the network and application layers, have been further subdivided into sublayers.**”).

Pet. at 47; see also Pet. at 57, 67, 83-84, 88, 90

Thia Does Not Disclose Network-Layer Bypass

2 The Bypass Concept

.....

2.3 Multiple-layer bypass

A bypass for multiple layers instead of just one gives additional gains by avoiding:

- Overhead of encoding and decoding the interface control information passed between layers;
- Executing the full general protocol logic for the layers to decide how to manipulate the data;
- Queueing of data at layer boundaries.

The advantage is increased further in cases where some layers, like the network and application layers, have been further subdivided into sublayers.

Ex. 1015 (Thia) at 2, 4

Thia Does Not Disclose Network-Layer Bypass

The advantage is increased further in cases where some layers, like the network and application layers, have been further subdivided into sublayers.

Ex. 1015 (Thia) at 2

100. As recited above, Section 2.3 describes additional theoretical advantages of bypassing more than one layer, beyond the theoretical advantages of bypassing a single layer disclosed earlier in Section 2. More specifically, Section 2.3 notes that bypassing multiple layers would theoretically allow the processor to also bypass inter-layer processing, such as “[o]verhead of encoding and decoding the interface control information passed between layers” and “[q]ueuing of data at layer boundaries.” (Ex. 1015.004.) Section 2.3 claims that the theoretical advantages of bypassing multiple layers are “increased further” for layers that are themselves “further subdivided into sublayers.” Thia then gives two examples of layers that have sublayers: “the network and application layers.”

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S
DECLARATION OF KEVIN ALM

¹ Cavium, who filed a Petition in Case IPR2017-01405, is the petitioner in this proceeding.

Ex. 2026 at 48-49

-01405 PO Demonstrative, 36

Thia Does Not Disclose Network-Layer Bypass

101. Read in its proper context, it is clear that Thia does not include network layer bypass. Thia merely recites additional theoretical advantages of bypassing more than one layer, then notes that those advantages would further apply to layers that are themselves composed of sublayers, and thus can benefit from offloading intra-(sub)layer processing. Importantly, Thia never mentions the network layer again, nor provide any disclosure as to how to bypass the network layer. Notably, Thia makes no conclusions as to the network layer, limiting its proof-of-concept to the transport and session layers. (Ex. 1015.013: “It can be concluded from this study that it is feasible to implement the bypass stack (at least for the transport and session layers) in VLSI and that the performance would be at least an order of magnitude higher than software protocol processing.”) Thia does not indicate how to bypass the network layer or even which portions of the network layer should be bypassed.

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMERTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a
petitioner in this proceeding.

Thia Repeatedly Excludes Network-Layer Bypass

Abstract — The Reduced Operation Protocol Engine (ROPE) presented here offloads critical functions of a multiple-layer protocol stack, based on the “bypass concept” of a fast path for data transfer. The motivation for identifying this separate processing path is that it involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inter-layer operations such as queue and buffer management, context switching and movement of data across layers, all of which are a significant overhead. ROPE is intended to support high-speed bulk data transfer. The paper describes the design of a ROPE chip for the OSI Session and Transport layer protocols, using VHDL. The design is practical in terms of chip complexity and area, using current gate array technology, and simulation shows that it can support a data rate approaching 1 gigabit per second, in a connection attached to an end-system.

Ex. 1015 (Thia) at 1

7 Summary

It can be concluded from this study that it is feasible to implement the bypass stack (at least for the transport and session layers) in VLSI and that the performance would be at least an order of magnitude higher than software protocol processing. The bypass system offloads the critical protocol functions and the associated non-protocol-specific functions onto a “Reduced Operation Protocol Engine” (ROPE). The gate count for the bypass chip can

Ex. 1015 (Thia) at 13

-01405 PO Demonstrative, 38

Thia Repeatedly Excludes Network-Layer Bypass

Layer	Procedure	Bypass Chip	Host	Per-Octet (A)	Per-Packet (B)	Per-Group-Of-Packets Aggregated to Per-Packet for bulk data transfer (B)	Remarks
Presentation	Encoding	X		X			
	Encryption	X		X			
	Compression	X		X			
	Context Alteration		X			X	
Session	Synchronization Management		X			X	
	Token management		X			X	
Transport (Class 4)	Checksum (Optional)	X		X			
	Timer Management	X			X	X	Depends on Implementation
	Generation of ACK packets (Flow Control)	X				X	
	Resequencing	X			X		
All 3 layers	Header Construction	X			X		
	Header Decode	X			X		
	Buffer Management	X			X		Minimized (Simple scheme)
	Context Switching	X			X	X	Moved away from host OS
	Data Copying	With multiple-layer bypass, data Copying within layers is eliminated.				X	Use of dual-ported memory and DMA..

Table 1 Bypassable versus Non-bypassable functions

Ex. 1015 (Thia) at 6
-01405 PO Demonstrative, 39

Thia Repeatedly Excludes Network-Layer Bypass

6 Considerations for the Session and Presentation layers (not implemented)

The next steps would logically be to enhance the Session [18, 19] processing to the BSS (Basic Synchronized Subset) or BAS (Basic Activity Subset) level, or to add Presentation processing

In the BSS, synchronization points occur but the session services do not actually save session SDUs and do not themselves perform the recovery operations. These activities are the responsibility of the application layer or the application program. The session layer merely decrements the serial number back to the synchronization point and the user must apply it to determine where to begin recovery procedures. Hence these activities are best handled on the host processor and are not very suitable for bypassing. The benefit they would confer (if bypassed) would be to reduce the frequency of switching paths.

Presentation processing can definitely be bypassed. During the data transfer phase, it consists only of the Presentation data encoding/decoding functions. Substantial performance gains could result if the presentation conversions are simple and are used consistently, although the inflexibility of a hardware version is an evident weakness. One possible application of ROPE with hardwired presentation conversion is in video servers with the proposed encoding standards such as MPEG [13].

Ex. 1015 (Thia) at 13

-01405 PO Demonstrative, 40

All Independent Claims Involve Network-Layer Bypass

response such that a data portion of the response is placed into the destination memory without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.

'205 Patent, Claim 3 (through Claim 1)

processed such that the data is placed into the destination memory on the host computer without the protocol stack of the host computer doing significant network layer or significant transport layer processing, the means also being for receiving a subsequent portion of

'205 Patent, Claim 31 (whence Claims 32-33 depend)

performing fast-path processing on the packet such that the data is placed into a destination memory without the protocol stack of the host computer doing any network layer processing on the packet and without the protocol stack of the host computer doing any transport layer processing on the packet;

'205 Patent, Claims 9-10, 16 (through Claim 8)

an iSCSI layer, the host bus adapter being adapted for processing the response such that a data portion of the response is placed into a memory on the host computer without the host computer doing any network layer or transport layer processing on the response.

'205 Patent, Claim 35

first portion is placed into the destination memory on the host computer with the protocol stack of the host computer doing substantially no network layer or transport layer processing, the network interface device receiving a second portion of the response to the iSCSI

'205 Patent, Claim 22 (whence Claims 24-30 depend)

memory on a host computer that is coupled to the host bus adapter without a protocol stack of the host computer doing any network layer processing on the response and without the host computer doing any transport layer processing on the response, the protocol

'205 Patent, Claim 36

All Independent Claims Bypass Most or All Transport Layer Processing on the Response or Packet

response such that a data portion of the response is placed into the destination memory **without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.**

'205 Patent, Claim 3 (through Claim 1)

processed such that the data is placed into the destination memory on the host computer **without the protocol stack of the host computer doing significant network layer or significant transport layer processing,** the means also being for receiving a subsequent portion of

'205 Patent, Claim 31 (whence Claims 32-33 depend)

performing fast-path processing on the packet such that the data is placed into a destination memory without the protocol stack of the host computer doing any network layer processing on the packet and **without the protocol stack of the host computer doing any transport layer processing on the packet;**

'205 Patent, Claims 9-10, 16 (through Claim 8)

an ISCSI layer, the host bus adapter being adapted for processing the response such that a data portion of the response is placed into a memory on the host computer **without the host computer doing any network layer or transport layer processing on the response.**

'205 Patent, Claim 35

first portion is placed into the destination memory on the host computer **with the protocol stack of the host computer doing substantially no network layer or transport layer processing,** the network interface device receiving a second portion of the response to the ISCSI

'205 Patent, Claim 22 (whence Claims 24-30 depend)

bus adapter without a protocol stack of the host computer doing any network layer processing on the response and **without the host computer doing any transport layer processing on the response,** the protocol stack of the host computer having an ISCSI layer.

'205 Patent, Claim 36

Among Other Things, the Transport Layer Reassembles Packets



Techopedia explains *Transport Layer*

Transport layers work transparently within the layers above to deliver and receive data without errors. The send side breaks application messages into segments and passes them on to the network layer. The receiving side then reassembles segments into messages and passes them to the application layer.

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Ex. 2044

INTEL
CAVIUM
Petitioner

ALACRIT
Patent

Case IPR
U.S. Patent

CORRECTED PATENT
DECLARATION OF K...

105. This only discloses bypassing “some common TP4 functionality,” specifically “checksum, retransmission on timeout and resequencing.” (Ex. 1015.010; *see also* Ex. 1015.006, Table 1 (showing transport procedures implemented on the bypass chip).) But This does not disclose bypassing reassembling the incoming packets, which is a primary responsibility of the transport layer.⁵ In fact, This explicitly rejects bypassing reassembly:

⁵ Cavium, who filed a Petition in Case petitioner in this proceeding.

06073-00001985444.5

Alacritech Exhibit 2026

Ex. 2026 at 52

-01405 PO Demonstrative, 43

This Only Discloses Bypassing "Some" Transport Layer Functionality (and Not Reassembly)

Layer	Procedure	Bypass Chip	Host	Per-Octet (A)	Per-Packet (B)	Per-Group-Of-Packets Aggregated to Per-Packet	Remarks
Presentation	Encoding						
	Encryption						
	Compression						
	Context Alterat						
Session	Synchronizatio Management						
	Token management		X			X	
Transport (Class 4)	Checksum (Optional)	X		X			
	Timer Management	X			X	X	Depends on Implementation
	Generation of ACK packets (Flow Control)	X				X	
	Resequencing	X			X		
All 3 layers	Header Construction	X			X		
	Header Decode	X			X		
	Buffer Management	X			X		Minimized (Simple scheme)
	Context Switching	X			X	X	Moved away from host OS
	Data Copying	With multiple-layer bypass, data Copying within layers is eliminated.				X	Use of dual-ported memory and DMA..

4.5 Second Design, including major procedures for Transport Class 4 (Implemented)

This section describes extensions to the first design, which only supports Session BCS and TP2 functionality, to include **some common TP4 functionality**. Procedures for checksum, retransmission on timeout and resequencing were implemented. Extensions to the Session layer functionality and procedures for presentation layer conversion were not implemented, but are also discussed in section 6.

Ex. 1015 (Thia) at 10

Table 1 Bypassable versus Non-bypassable functions

Ex. 1015 (Thia) at 6

-01405 PO Demonstrative, 44

Thia Expressly Discloses It Does Not Bypass Reassembly and Further Teaches Away From Doing So

The scope of functions included in a bypass may be narrowly defined, or more extended. A bypass does not include fast connection setup but also does not interfere with it. There is no segmentation/reassembly within the bypass path, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. The Segmentation and Reassembly sublayer of the ATM adaptation layer is a good place for such functions [25].

Ex. 1015 (Thia) at 14

All Independent Claims Require Bypass of All or All Significant Transport Layer Processing on the Response or Packet

response such that a data portion of the response is placed into the destination memory **without the protocol stack of the host computer performing any network layer processing or any transport layer processing on the response.**

'205 Patent, Claim 3 (through Claim 1)

processed such that the data is placed into the destination memory on the host computer **without the protocol stack of the host computer doing significant network layer or significant transport layer processing,** the means also being for receiving a subsequent portion of

'205 Patent, Claim 31 (whence Claims 32-33 depend)

performing fast-path processing on the packet such that the data is placed into a destination memory without the protocol stack of the host computer doing any network layer processing on the packet and **without the protocol stack of the host computer doing any transport layer processing on the packet;**

'205 Patent, Claims 9-10, 16 (through Claim 8)

an ISCSI layer, the host bus adapter being adapted for processing the response such that a data portion of the response is placed into a memory on the host computer **without the host computer doing any network layer or transport layer processing on the response.**

'205 Patent, Claim 35

first portion is placed into the destination memory on the host computer **with the protocol stack of the host computer doing substantially no network layer or transport layer processing,** the network interface device receiving a second portion of the response to the ISCSI

'205 Patent, Claim 22 (whence Claims 24-30 depend)

bus adapter without a protocol stack of the host computer doing any network layer processing on the response and **without the host computer doing any transport layer processing on the response,** the protocol stack of the host computer having an ISCSI layer.

'205 Patent, Claim 36

Thia Discloses a Theoretical System

4 VLSI implementation of a Reduced Operation Protocol Engine (ROPE) chip using VHDL

The VHSIC Hardware Description Language (VHDL) [6, 27] was used to model and synthesize the chip. VHDL is an industry standard language which can be used to represent all levels of abstraction, from logic gates to the system level. By utilizing VHDL as a specification tool, it is possible to begin simulation and debugging of complex systems before details regarding the implementation are fully specified. It also offers the potential of an automatic path from the protocol specification to VLSI implementation, in which any modifications to the specification can be easily propagated to the gate level design.

Ex. 1015 (Thia) at 6-7

Thia Discloses a Theoretical System

4.2 Architectural Description

Figure 2 shows the block diagram of the system. The host processor and NIA components provide logical interfaces for simulation of behaviour, but they insert no timing delays. For modeling purposes they were described as being infinitely fast, either as a source or as a sink. This places the maximum stress on the ROPE chip. The architectural considerations involved in the chip design can be summarized as follows:

- Movement of data across the host bus interface are minimized by using an on-chip DMA for fast block data transfer to/from the host system memory.
- On-chip dual-ported memory is used, rather than the host memory, to avoid critical constraints on bus access latency and throughput.
- The control registers of the bypass chip are I/O mapped to the host processor. This enables the host processor to configure the bypass chip directly.

The presentation module shown in the Figure was allowed for in the data structures but was not fully designed.

Ex. 1015 (Thia) at 7

-01405 PO Demonstrative, 48

No Motivation to Repeat a Feasibility Study with iSCSI

116. First, a person of skill would not “have been motivated to combine Thia and Satran to provide Thia with a communication protocol that takes advantage of the platform – that is, to provide Thia with a real-world communications protocol,” because Thia does not function in the “real world” and thus has no need for “a real-world communications protocol.” Instead, Thia uses a “vestigial high-speed network interface adapter which acts simply as an infinite source/sink for data packets” and operates on a “test sequence” of packets. (Ex. 1015.008.) Having confirmed the feasibility of bypassing portions of the session and transport layers (Ex. 1015.013), a person of skill would have no reason to repeat Thia’s simulated experiment with a different sequence of packets, let alone iSCSI packets.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMEROTI

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been removed as a
petitioner in this proceeding.

06073-000019854048.5

Alacritech Exhibit 2026

Ex. 2026 at 59

No Motivation to Repeat a Feasibility Study with iSCSI

117. Second, it is not the case that “the addition of SCSI improves the operation Thia,” because Thia is a feasibility study, not an operating product.

Because there is no designed product to manufacture and sell, there is no need to “broaden[] Thia’s appeal to, and acceptance by, the market.” It is thus only through impermissible hindsight that Petitioners arrive at repeating Thia’s feasibility study with iSCSI packets.

Ex. 2026 at 59

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIEBUNAL

INTEL CORPORATION
CAVIUM

Petition

v.

ALACRITECH

Patent Owner

Case IPR2017-01735
U.S. Patent 7,124,205

CORRECTED PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

06073-000019854048.5

Alacritech Exhibit 2026

No Motivation to Repeat a Feasibility Study with iSCSI

118. Third, there would be no need to “improve Thia by adding the functionality provided by Satran,” because Thia only discloses that bypassing certain portions of certain layers would be computationally advantageous. There is no need to “improve” a general finding that bypassing may be beneficial with another general finding using a different series of packets, let alone iSCSI packets. Notably, Thia is not “designed to work in a network environment”; rather, Thia functions in a simulation of a network environment and thus has no need for the “network communications” provided by iSCSI.

UNITED STATES PATENT AND
 TRADEMARK OFFICE
 BEFORE THE PATENT TRIEBUNAL

INTEL CORPORATION
 CAVIUM INC.
 Petitioners

ALACRITECH CORPORATION
 Patent

Case IPR2017-01735
 U.S. Patent and Trademark Office

CORRECTED PATENT OWNER'S EXHIBIT 2026
 DECLARATION OF KEVIN ALMEROOTH, PH.D.

Ex. 2026 at 59-60

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

No Motivation to Repeat a Feasibility Study with iSCSI

119. Fourth, iSCSI's improvements to the performance of a network system are irrelevant to Thia, which as discussed above is a feasibility study that functions in a simulated network environment. There are no "network-attached devices" or "storage devices" on this simulated network—which is modeled as "an infinite source/sink for data packets" (Ex. 1015.008)—whose performance needs to be improved. Repeating Thia's feasibility study with iSCSI packets thus does not offer any relevant improvement. It merely uses a different set of packets to achieve the same result. Petitioners have not provided any motivation to repeat Thia's feasibility study at all, let alone to repeat it with iSCSI packets.

Ex. 2026 at 60

UNITED STATES PATENT AND
 TRADEMARK OFFICE
 BEFORE THE PATENT TRIEBUNAL

INTEL CORPORATION
 CAVIUM INCORPORATED
 Petitioners

v.

ALACRITECH CORPORATION
 Patent Owner

Case IPR2017-01735
 U.S. Patent and Trademark Office

CORRECTED PATENT OWNER
 DECLARATION OF KEY

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

No Motivation to Repeat a Feasibility Study with iSCSI

120. Finally, there is no “predictable result” of “Thia providing fast-path capability to Satran communications,” because Thia is a feasibility study, not a real-world system. Nor would a person of skill in the art believe that the findings of Thia’s feasibility study could be somehow improved by using different simulated packets.

Ex. 2026 at 60

UNITED STATES PATENT
OFFICE
BEFORE THE PATENT TRIEBUNAL

INTEL
CAV
Petitioner

ALACR

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

No Motivation to Repeat a Feasibility Study with iSCSI

121. At bottom, iSCSI is one of hundreds of real and artificial protocols which Thia could use to simulate incoming or outgoing packets to the theoretical ROPE system. A person of skill in the art would have no reason to repeat the Thia feasibility study using iSCSI packets as opposed to any other form of packet, nor would a person of skill in the art see any benefit in doing so. It is only through hindsight that Petitioners arrive at that combination. In fact, it is unlikely a person of skill in the art would see *any* reason to modify Thia in any way. Thia concludes that “it is feasible to implement the bypass stack (at least for the transport and session layers.” A person of skill in the art would see little reason to expend the time/resources to vary the parameters of the Thia study to as to arrive at the same conclusion.

UNITED STATES PATENT AND TRADEMARK
OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMEROTH

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

Secondary Considerations: Long-Felt Need

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality

Ex. 2031 at 1 (1993 IBM Research Division ACM SIGCOMM article)

1 Introduction

Many researchers have observed that the performance of remote applications have not kept pace with modern communication network speeds. Part of this imbalance is attributed to the protocols used by the remote applications, namely, UDP/IP and TCP/IP. It is now widely believed that the problems with these protocols are not inherent in the protocols themselves but in their particular implementations [Dalt93, Whet95]. The following factors are cited as contributors to the poor performance of UDP/IP and TCP/IP:

Ex. 2032 at 1 (1995 Univ. of Berkeley paper)

Secondary Considerations: Long-Felt Need

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to “bottlenecks” in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication pro-

Ex. 2033 at 1 (1990 IEEE article from AT&T Bell Labs)

Abstract

Server network performance is increasingly dominated by poorly scaling operations such as I/O bus crossings, cache misses and interrupts. Their overhead prevents performance from scaling even with increased CPU, link or I/O bus bandwidths. These operations can be reduced by redesigning the host/adaptor interface to exploit additional processing on the adapter. Offloading processing to the adapter is beneficial not only because it allows more cycles to be applied but also of the changes it enables in the host/adaptor interface. As opposed to other approaches such as RDMA, TCP offload provides benefits without requiring changes to either the transport protocol or API.

Ex. 2034 at 1 (2005 USENIX Conference paper from IBM)

Secondary Considerations: Long-Felt Need

126. The nexus between the long-felt need and the claimed invention is clear and direct. The aforementioned bottlenecks are all solved by the accelerated network processing technologies recited in the challenged claims. In addition, the technologies recited in the challenged claims alleviated processing demand on the host CPU in such a way as to permit faster network throughput and transmission/reception speeds—the precise long-felt but unresolved need in the industry outlined above.

Ex. 2026 at 63-64

UNITED STATES PATENT

BEFORE THE PATENT

INTERNATIONAL

ALACRIT

Petitioner

Case IPR2017-01735
U.S. Patent

CORRECTED PATENT OWNER'S EXHIBIT 2026
 DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

06073-000019854048.5

Alacritech Exhibit 2026

Secondary Considerations: Industry Praise

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

Ex. 2039 at 4 (2001 HP Labs research paper)

Secondary Considerations: Industry Praise

“The ANX 1500 is an evolutionary advancement of Alacritech’s long standing leadership in protocol acceleration, which is now applied not just to protocols, but to the optimization of all storage system interactions,” said Jeff Boles, a technology analyst with Taneja Group. “The ANX 1500 substantially augments the performance of existing NAS investments from the best place possible – from right in the customer’s Ethernet network – without reconfiguration or changes in management for the existing infrastructure. We think Alacritech is setting the stage for a next generation of solutions that will accelerate storage from outside the storage array, and more cost effectively share an investment in performance across many storage systems and clients. I’ve talked to early-stage customers using the product, and they believe it’s game-changing.”

Ex. 2040 at 3 (Shoreline Ventures Press Release)

-01405 PO Demonstrative, 59

Secondary Considerations: Industry Praise

130. The industry universally praised **commercial embodiments of the features described in the challenged claims.** HP found that the “Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed” and “achiev[e] lower processor utilization than native NT’s TCP/IP protocol stack for transmission of large enough messages.” (Ex. 2039 at ¶ 4.) HP found that the test performance of the Alacritech

Ex. 2026 at 65

UNITED STATES PATENT AND
BEFORE THE PATENT TRIAL

INTEL CORP.
CAVIUM, INC.
Petitioners

v.

ALACRITECH
Patent Owner

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

Secondary Considerations: Failure of Others

To this day, TCP offload has never firmly caught on in the commercial world (except sometimes as a stopgap to add TCP support to immature systems [16]), and has been scorned by the academic community and Internet purists. This paper starts by analyzing why TCP offload has repeatedly failed.

Ex. 2041 at 2 (2003 HP Labs paper presented at HotOS IX conference)

TCP offload has been unsuccessful in the past for two kinds of reasons: fundamental performance issues, and difficulties resulting from the complexities of deploying TCP offload in practice.

Id. at 2

Secondary Considerations: Failure of Others

offload were mismatched to the applications in question.” *Id.* The TCP offload described above is a form of network processing offload that is described by the challenged claims, and this failure of others therefore has a direct nexus to the claimed inventions.

Ex. 2026 at 66

UNITED STATES PATENT AND TRADE
 BEFORE THE PATENT TRIAL AND AP

INTEL CORP. and
 CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

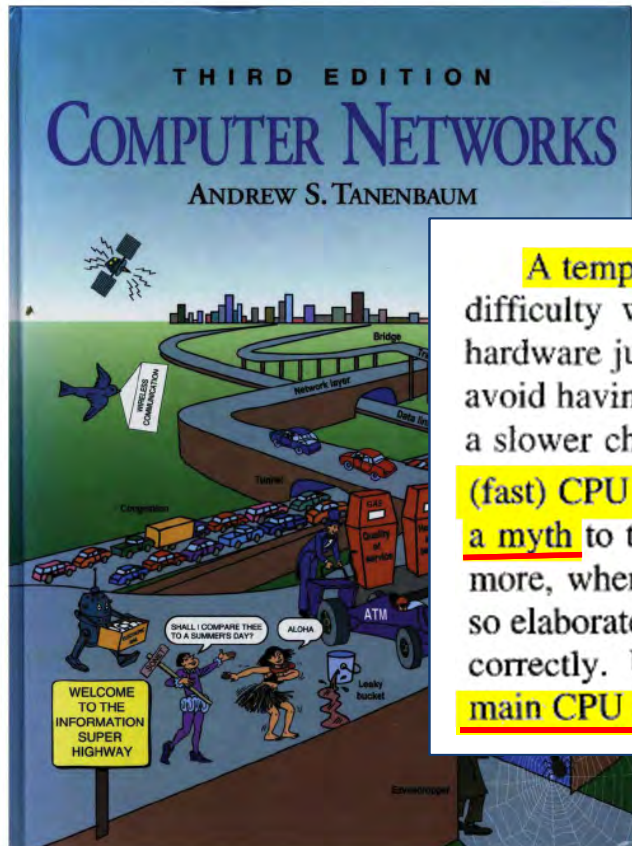
Patent Owner.

Case IPR2017-01392¹
 U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Secondary Considerations: Skepticism



INTEL Ex.1006.001

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.

Ex. 1006 at 588-589

Secondary Considerations: Skepticism

IP Storage and the CPU Consumption Myth

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

Abstract

This paper addresses a key issue that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2.

The
accel
CPU
been
comm
unme
Ex
date
many
I/O p
Howe
of arc
rapid
A
I/O)
proce
serve
from
starte
task,
as the
some
Some
or w
and s
usual
and e
envir
costly
for if
the p
kills
gener
the c
Si
accel
of en
ever
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

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

Ex. 2300 at 194 (2001 paper by Petitioner's expert, Dr. Horst)

Secondary Considerations: Skepticism

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2. The Historical Argument

There are many historical examples of accelerators to offload processing tasks from CPU. Some examples, such as graphics have been successful, but the history of communications processors is filled with unmet expectations.

Examples of front-end communication date from the early days of mainframe computing systems, the primary CPU was accompanied by an I/O processor to offload low-level protocol processing. However, it has become increasingly difficult of architecture to deliver real performance gain at a rapid pace of technology evolution.

A specific recent example is the Intel I²O (Intelligent I/O) initiative. The idea was to have a communications processor, such as an Intel i960, on the motherboard to serve as an I/O processor to offload and isolate the CPU from its attached I/O devices. At the time the i960 embedded processor was adequate to the task, but its performance did not increase at the same rate as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, and a generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that the problem with accelerators should have been optimized hardware of embedded programmable processors. Unfortunately, every protocol worthy of acceleration continues to change, and it is difficult to stay ahead of the moving target. New protocols proposed for IP storage, iSCSI, are far from stable, and even after the standard is formally approved, there will likely be a number of enhancements and bug fixes. It seems

A specific recent example is the Intel I²O (Intelligent I/O) initiative. The idea was to have a communications processor, such as an Intel i960, on the motherboard to serve as an I/O processor to offload and isolate the CPU from its attached I/O devices. At the time the initiative started, the i960 embedded processor was adequate to the task, but its performance did not increase at the same rate as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Secondary Considerations: Skepticism

(Ex. 2300 at 194.) (emphasis added). This level of skepticism and teaching away is directly related to the heart of the claimed invention – offloading network processing, and therefore has a direct nexus to the challenged claims.

Ex. 2026 at 68

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01405¹
U.S. Patent 7,124,205

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01735, has been joined as a petitioner in this proceeding.

'205 Patent – Contingent Motion to Amend

Proposed Claim 37

37. (proposed substitute for claim 3) the apparatus of claim 1, wherein the session layer protocol is ISCSI, and wherein the fast-path processing reassembles the data portion of the response with a second data portion of a second response.

** substantially similar amendments to dependent claims 9-10 and 16
(proposed substitute claims 38-40)*

'205 Patent – Contingent Motion to Amend

Proposed Claim 41

41. (proposed substitute for claim 22) An apparatus comprising:
a host computer having a protocol stack and a destination memory; and
a network interface device coupled to the host computer, the network interface device receiving a first and second portions of a response to an ISCSI read request command, the first and second portions being processed such that a first data portion of the first portion and a second data portion of the second portion are reassembled and [[is]] placed into the destination memory on the host computer with the protocol stack of the host computer doing substantially no network layer or transport layer processing, the network interface device receiving a [[second]] third portion of the response to the ISCSI read request command, the protocol stack of the host computer doing network layer and transport layer processing on the [[second]] third portion.

** Substantially similar amendments to independent claims 35, 36, and 31 (proposed substitute claims 49-51)*

*** Proposed substitutes for dependent claims 24-30 and 32-33 (substitute claims 42-48 and 52-53) update the claim dependency*

-01405 PO Demonstrative, 68

Support for “reassembles the data portion of the response”

[0092] To perform this read, the file system instructs INIC 622 to fetch the 64 KB of data from network storage unit 640 into the INIC 622 file cache. The INIC 622 then sends a request for the data over network 644 to network storage unit 640. The request may take the form of one or more SCSI commands to the storage unit 640 to read the blocks, with the commands attached to TCP/IP headers, according to iSCSI or similar protocols. A controller on the storage unit 640 responds to the commands by reading the requested blocks from its disk drive or drives, adding iSCSI or similar protocol headers to the blocks or frame-sized portions of the blocks, and sending the resulting frames over network 644 to INIC 622. The frames are received by the INIC 622, processed by the INIC 622 sequencers, matched with the storage CCB, and reassembled as a 64 KB file stream in the INIC file cache that forms part of the requested 100 KB file. Once the file stream is stored on INIC 622 file cache, SMB constructs a read reply and sends a scatter-gather list denoting that file stream to INIC 622, and passes the reply to the INIC 622 to send the data over the network according to the server CCB. The INIC 622 employs the scatter-gather list to read data packets from its file cache, which are prepended with IP/TCP/NetBios/SMB headers created by the INIC based on the server CCB, and sends the resulting frames onto network 604. The remaining 36 KB of the file is sent by similar means. In this manner a file on a network storage unit may be transferred under control of the server without any of the data from the file encountering the I/O bus or server protocol stack.

'124 Application, EX2022

-01405 PO Demonstrative, 69

Support for “reassembles the data portion of the response”

10. Paragraphs [0090] through [0097] likewise discuss a more detailed embodiment involving two receiving devices depicted in Fig. 14: a client (item 602) that originates the data request and a server (item 600) that retrieves the requested data from a storage unit (items 634, 640, and 642), stores it on a cache, then transmits that data to the client. While a person of skill in the art would not need things spelled out this explicitly, here the specification explicitly observes that “[t]he frames are [] *reassembled* as a 64 KB file stream in the INIC file cache.” ([0092].) While that file cache is on the “INIC” and not the host memory, the specification also discloses at, *e.g.*, Paragraph [0056] that file caches may be on host memory as well.

EX2305 at 4

Support for "reassembles the data portion of the response"

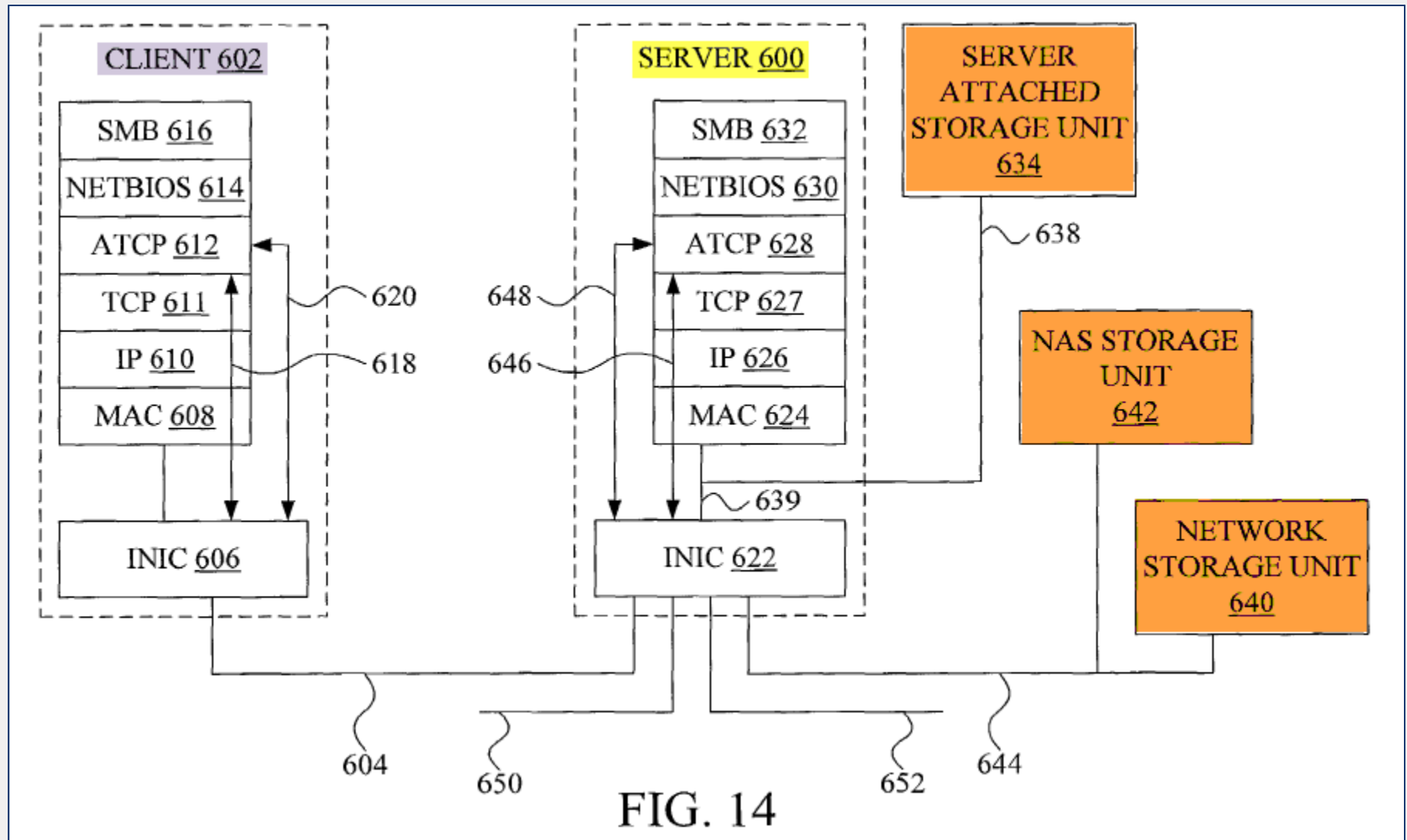


FIG. 14

EX2022 at 12, Fig. 14

-01405 PO Demonstrative, 71

Support for “reassembles the data portion of the response”

11. As to this latter disclosure, Petitioner asserts that because the reassembly is “done by the device transmitting the data portions,” a person of skill in the art would not understand it to disclose reassembly by the receiver. (Op. at 5.) That is incorrect. As I discuss above, the server transmitting the data is *also* a receiver that receives the data from one of the storage units listed in items 634, 640, and 642. In the process of receiving the data, the server reassembles the data in the file cache.

EX2305 at 4

Support for “reassembles the data portion of the response”

[0057] Once the CCB indicates the destination, fast-path

• • •

Upon matching the packet summary with the CCB, assuming no exception conditions exist, the data of the packet, without network or transport layer headers, is sent by direct memory access (DMA) unit 68 to the destination in file cache 80 or file cache 24 denoted by the CCB.

'124 Application, EX2022

8. In addition, paragraphs [0056] through [0058] describe an embodiment where “the data of the packet, without network or transport layer headers, is sent by direct memory access (DMA) unit 68 to the destination in file cache 80 or file cache 24 denoted by the CCB.” In placing the data of each incoming packet in “file cache 24”—which is in host memory (Fig. 1)—the DMA unit on the network interface device thus reassembles the data.

EX2305 at 3

-01405 PO Demonstrative, 73

Thia Does Not Disclose Reassembly



The scope of functions included in a bypass may be narrowly defined, or more extended. A bypass does not include fast connection setup but also does not interfere with it. There is no segmentation/reassembly within the bypass path, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. The Segmentation and Reassembly sublayer of the ATM adaptation layer is a good place for such functions [25].

Ex. 1015 (Thia) at 14

Thia Does Not Disclose Reassembly

22. Whether a person of skill in the art would consider a protocol layer to be a “lower” layer is a function of context. For instance, in relation to Layer 7 (Application Layer), every other layer would be deemed a “lower” layer. I have reviewed the context of the Thia disclosures, and it is my opinion that a person of skill in the art would consider the “lower layers” described in Ex. 1015.014 to include the transport layer. *See, e.g.,* <http://www.certiology.com/computing/computer-networking/osi-layer-model.html> (“The ‘physical layer’ up to the ‘transport layer’ is known as the **lower layers**, whereas the ‘session layer’ up to the ‘application layer’ makes up the **upper layers** of the OSI model.”) (emphasis in original); <https://www.lifewire.com/layers-of-the-osi-model-illustrated-818017> (depicting the Application, Presentation, and Session layers as “Upper Layers” and the **Transport, Network, Data Link, and Physical layers as “Lower Layers”**). I do not see anything in Thia that suggests a departure from this understanding.

Ex. 2305 at 10

-01405 PO Demonstrative, 75

Thia Does Not Disclose Reassembly

18. One mechanism by which the transport layer on the receiver may deal with out-of-sequence packets is to simply discard them, as recited in Thia itself: “out-of-sequence PDUs [] will be discarded.” (Ex. 1015.010.) In that case, the receiver will not acknowledge that the out-of-sequence packet was received, “timer T1 expires, [and] the transport entity can retransmit either the first TPDU, or all TPDU(s) (Go-back-N) waiting for acknowledgment.” (Ex. 1015.011.) Alternately, the transport layer can “buffer data packets for resequencing.” (*Id.*) That is, the transport layer can store the out-of-sequence packet on “the on-chip buffer” or “slower external memory” until the missing packets earlier in the sequence arrive and are processed, after which the transport layer can process the out-of-sequence packet normally. (*Id.*)

Ex. 2305 at 8

-01405 PO Demonstrative, 76

Thia Does Not Disclose Reassembly

19. While resequencing thus ensures that packets are processed in the correct order, resequencing does not itself process the packets. That resequencing is an alternative solution to retransmission is evidence of that fact, as a retransmission by the sender does not process the packets by the receiver. For example, resequencing does not perform checksum validation, which Thia itself discloses as a separately implemented bypass functionality. (Ex. 1015.006 at Table 1; *see also* 1015.010.) By the same token, Thia's disclosure of bypassing resequencing does not mean that Thia also bypasses reassembling the packets. In fact, Thia explicitly states otherwise, reciting that "[t]here is no segmentation/reassembly within the bypass path." (Ex. 1015.014.)

Ex. 2305 at 8-9

-01405 PO Demonstrative, 77

INTEL CORPORATION, CAVIUM, INC., DELL INC.

v.

ALACRITECH INC.

Case Nos.: IPR2017-1409, 1410, 1736, 1737;

IPR2018-338, 339


U.S. Patent No. 8,131,880

Patent Owner's Demonstratives

Hearing: September 13, 2018

-1409, -1410 PO Demonstrative, 1

Overview of the '880 Patent



US008131880B2

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

(10) Patent No.: **US 8,131,880 B2**
(45) Date of Patent: ***Mar. 6, 2012**

INTELLIGENT NETWORK INTERFACE DEVICE AND SYSTEM FOR ACCELERATED COMMUNICATION

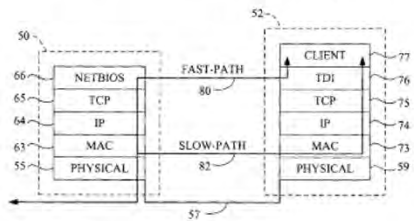
Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

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

Continuation-in-part of application No. 09/098,296, filed on Apr. 27, 1998, now Pat. No. 6,226,680, and a continuation-in-part of application No. 09/141,713, filed on Aug. 28, 1998, now Pat. No. 6,389,479.
(60) Provisional application No. 60/098,296, filed on Aug. 27, 1998, provisional application No. 60/061,809, filed on Oct. 14, 1997.

processing by the host. The device 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 devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

60 Claims, 40 Drawing Sheets



INTEL Ex.1001.001

(57) **ABSTRACT**

An intelligent network interface card (INIC) or communication processing device (CPD) works with a host computer for data communication. The device provides a fast-path that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU. The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path processing. A context for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device 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 devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

EX1001 at Abstract

'880 Patent (EX1001)

Problems Addressed by the '880 Patent

Too many data moves:

driver interface. This can add up to a whopping 8 trips across the system memory bus (the 4 trips described above, plus the move to replenish the cache, plus 3 more to copy from the miniport to the protocol driver). That's enough to bring even today's advanced memory busses to their knees.

EX1001 at 1:57-2:14

Inefficient use of the PCI bus:

We noted earlier that when the CPU has to access system memory, it may be stalled for several hundred nanoseconds. When it has to read from PCI, it may be stalled for many microseconds. This happens every time the CPU takes an interrupt from a standard NIC. The first thing the CPU must do when it receives one of these interrupts is to read the NIC Interrupt Status Register (ISR) from PCI to determine the cause of the interrupt. The most troubling thing about this is that since interrupt lines are shared on PC-based systems, we may have to perform this expensive PCI read even when the interrupt is not meant for us.

EX1001 at 3:17-41

Too much processing by the CPU:

This is particularly expensive because while the CPU is moving this data it can do nothing else. While moving the data the CPU is typically stalled waiting for the relatively slow memory to satisfy its read and write requests. A CPU, which can execute an instruction every 5 nanoseconds, must now wait as long as several hundred nanoseconds for the memory controller to respond before it can begin its next instruction.

* * *

But the data movement is not the only drain on the CPU. There is also a fair amount of processing that must be done by the protocol stack software. The most obvious expense is

EX1001 at 2:15-54

Too many interrupts:

A 64 k server message block (SMB) request (write or read-reply) is typically made up of 44 TCP segments when running over Ethernet, which has a 1500 byte maximum transmission unit (MTU). Each of these segments may result in an interrupt to the CPU. Furthermore, since TCP must

* * *

Interrupts tend to be very costly to the system. Often when

EX1001 at 2:55-3:16

'880 Patent – Independent Claim 1

1. A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:

- parsing a header portion of a first packet received at a network interface for the host computer system to determine if said first packet conforms to a TCP protocol;
- generating a flow key to identify a first communication flow that includes said first packet, wherein said flow key includes a TCP connection for the communication flow;
- associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol; and
- processing, by the network interface, said packet according to the TCP connection, including updating a control block representing the TCP connection on the network interface.

EX1001

'880 Patent – Independent Claim 1

1. A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:

- parsing a header portion of a first packet received at a network interface for the host computer system to determine if said first packet conforms to a TCP protocol;
- generating a flow key to identify a first communication flow that includes said first packet, wherein said flow key includes a TCP connection for the communication flow;
- associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol; and
- processing, by the network interface, said packet according to the TCP connection, including updating a control block representing the TCP connection on the network interface.

EX1001

'880 Patent – Independent Claim 32

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

- receiving a packet from a network at a network interface of a host computer system;
- parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;
- generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet, wherein said flow key includes a TCP connection for the communication flow and a first hop medium access control (MAC) layer address;
- determining whether a header in said header portion conforms to a pre-selected protocol;
- storing said flow key in a database;
- associating an operation code with said packet, wherein said operation code identifies a status of said packet;
- storing said packet in a packet memory;
- if said header conforms to the TCP protocol:
- storing a data portion of said packet in a re-assembly buffer;
- storing said header portion in a header buffer; and
- processing, by the network interface, said packet according to the TCP connection.

EX1001

-1409, -1410 PO Demonstrative, 6

'880 Patent – Independent Claim 41

41. An apparatus for transferring a packet to a host computer system, comprising:

- a traffic classifier, disposed in a network interface for the host computer system, configured to classify a first packet received from a network by a communication flow that includes said first packet;
- a packet memory, disposed in the network interface, configured to store said first packet;
- a packet batching module, disposed in the network interface, configured to determine whether another packet in said packet memory belongs to said communication flow;
- a flow re-assembler, disposed in the network interface, configured to re-assemble a data portion of said first packet with a data portion of a second packet in said communication flow; and
- a processor, disposed in the network interface, that maintains a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

EX1001

'880 Patent – Independent Claim 43

43. A computer system for receiving a packet from a network, comprising:


- a memory configured to store packets received from a network; and
- a network interface for the computer system, the network interface configured to receive a first packet from said network, the network interface comprising:
 - a parser configured to extract information from a header portion of a first packet;
 - a flow manager configured to examine said information;
 - a flow database configured to store an identifier of a first communication flow comprising multiple packets, including said first packet; and
 - a re-assembler for storing data portions of said multiple packets without header portions in a first portion of said memory; and
- a processor for processing said first packet and for maintaining a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

EX1001

'880 Patent – Instituted Grounds

IPR	Challenged Claims	103(a) References
IPR2017-01409	<u>1</u> , 5–10, 12, 14, 16, 17, 20–23, 27, 28, 45, and 55	Thia and Tanenbaum
IPR2017-01410	<u>32</u> , 34, 35, 39, and <u>41–43</u>	Thia and Tanenbaum
IPR2017-01410	37 and 38	Thia, Tanenbaum, and Nahum

Thia and Tanenbaum Were Considered During Original Prosecution



US008131880B2

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

(10) Patent No.: US 8,131,880 B2
(45) Date of Patent: *Mar. 6, 2012

(54) INTELLE
DEVICE
COMMU

(75) Inventors

(73) Assignee

(*) Notice

(21) Appl. No.

(22) Filed

(65)
US 2004/

Re

(63) Continuat
Nov. 7,
continuat
filed on F
continuat
filed on D
continuat
Aug. 27,
continuat
filed on A
continuat
filed on A

(60) Provision
27, 1998,
filed on O

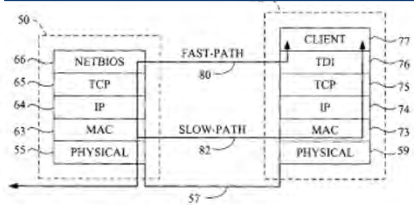
(56) **References Cited**

* * *

Andrew S. **Tanenbaum**, Computer Networks, Third Edition, 1996, ISBN 0-13-349945-6.

* * *

Thia, Y.H. Publication entitled "A Reduced Operational Protocol Engine (ROPE) for a multiple-layer bypass architecture", *Protocols for High Speed Networks*, pp. 224-239, 1995.



INTEL Ex.1001.001

Examiner	/Jerry Dennison/	Date Considered	03/22/2010
----------	------------------	-----------------	------------

'880 Patent (EX1001)

'880 File History (EX1002.477, 491)

A Reduced Operation Protocol Engine (ROPE) for a Multiple-layer Bypass Architecture ("Thia")

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*)¹ and C.M. Woodside (**)Newbridge Networks, Inc., Ottawa, Canada (*)
Dept. of Systems and Computer Engineering, U

This paper presents a feasibility study for a new approach to hardware assistance.

Abstract — The Reduced Operation Protocol Engine (ROPE) presented here offloads critical functions of a multiple-layer protocol stack, based on the "bypass concept" of a fast path for data transfer. The motivation for identifying this separate processing path is that it involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inter-layer operations such as queue and buffer management, context switching and movement of data across layers, all of which are a significant overhead. ROPE paper describes the design of a using VHDL. The design is pre array technology, and simulating per second, in a connection a

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols

1 Introduction

The advent of Fibre Optic rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-points of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-octet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementation of existing protocols [5, 35], parallel processing techniques [14, 21, 38], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offboard processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

¹ This research was done while Dr. Thia was at Carleton University

(*) Newbridge is a (eds), *Proceedings for High Speed Networks '97*
(**) Springer Science+Business Media, Dordrecht, 1997

INTEL Ex.1015.001

EX1015.002

to obtain the simulation throughput results presented in section 5. This was sufficient for our present purposes, to estimate the space complexity and timing of the chip, and the final step of generating a chip layout for fabrication and fault analysis was not performed. As

EX1015.008

Thia (EX1015)

Thia Is A Non-Enabling Reference

Case No. IPR2017-1409¹
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Decl. of Dr. Kevin Almeroth (EX2026)

115. In my opinion, Thia therefore discloses an inoperative device, in that one of ordinary skill would not have been able to construct a working device based on Thia's disclosures. In particular, Thia's high level disclosure of the basic architecture and idea of offloading certain processing associated with the OSI presentation, session, and transport layers to a "Reduced Operation Protocol Engine" or "ROPE" chip fails to provide necessary implementation details that would be required to actually build and implement a working device.

116. For example, Thia describes a bypass stack on the ROPE chip that provides a hardware "fast path" for bulk data transfer (Ex. 1015.002-003), but the bypass architecture is shown as nothing more than the bare flowcharts shown in Fig. 1 (below), which lacks many of the implementation details necessary to implement an actual working device.

120. In keeping with Thia's failure to enable implementation of an operative device, however, Thia does not address or disclose how the bypass test is implemented, what values of the incoming headers must match the template in order to identify predicted bypassable headers, or what the template even is.

-1409, -1410 PO Demonstrative, 12

This Is A Non-Enabling Reference

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*)¹ and C.M. Woodside (*)Newbridge Networks, Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering,

Abstract — The Reduced Operation Protocol Engine (ROPE) is a hardware implementation of the critical functions of a multiple-layer protocol stack for data transfer. The motivation for this involves only a small subset of the complete hardware. Multiple-layer bypass also eliminates buffer management, context switching and are a significant overhead. ROPE is intended paper describes the design of a ROPE chip for using VHDL. The design is practical in terms of array technology, and simulation shows that it is 100 per second, in a connection attached to an er

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Commun

1 Introduction

The advent of Fibre Optic technology, with its high data rates, has shifted the performance bottleneck in communications processing in the end-points of the network. Quality-of-service guarantees will reinforce this combination of operating system overhead, protocol processing and the data stream. To alleviate the end-system bottleneck, improved software implementation of existing protocols [14, 21, 38], special protocol structures [15, 30] and the partitioning of protocol functions to an adaptor.

The key problems associated with offloading protocol processing are:

- Partitioning the functionality between the SPS and the bypass stack lead to a complex additional protocol between the SPS and the bypass stack that may offset the potential gain from offloading. This may be offloaded, but this leaves the protocol logic.

¹ This research was done while Dr. Thia was at Carleton University.

(*) Newbridge Networks, Inc., Ottawa, Canada
(*) Springer Science+Business Media Dordrecht

2.1 Bypass Architecture

Figure 1 illustrates the architecture of a bypass implementation for any standard protocol. The standard protocol stack (SPS) is the processing path taken by all PDUs during a connection without the bypass. The SPS may refer to a single layer or to multiple adjoining layers of a layered protocol stack. The bypass has 4 key components:

- *Send Bypass Test;*
- *Receive Bypass Test;*
- *Bypass Stack;*
- *Shared data* for access by the two tests, the Bypass stack and the SPS.

The send bypass test identifies outgoing packets that are data packets in the data transfer phase. The receive bypass test matches the incoming PDU headers with a template that identifies the predicted bypassable headers. The bypass stack performs all the relevant protocol processing in the data transfer phase. The shared data are used to maintain state consistency between the SPS and the bypass stack, including window flow control parameters and connection identifiers. Whenever there is a change in the processing path between the SPS and the bypass stack, checks are performed to ensure that there are no outstanding packets in the current path, i.e. "no in-transit PDUs", before the change is made. A more detailed discussion on this is presented in another paper [33].

INTEL Ex.1015.001

EX1015.003

Thia (EX1015)

Thia Does Not Disclose Reassembly By The ROPE Chip

14

A Reduced Operation Protocol Engine multiple-layer bypass architecture

Y.H. Thia (*) and C.M. Woodside (**)

Newbridge Networks, Inc., Ottawa, Canada (*) and Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada (**)

Abstract — The Reduced Operation Protocol critical functions of a multiple-layer protocol stack, path for data transfer. The motivation for identifying involves only a small subset of the complete protocol hardware. Multiple-layer bypass also eliminates and buffer management, control switching and more are a significant overhead. ROPE is intended to be paper describes the design of a ROPE chip for the Q using VHDL. The design is practical in terms of chip array technology, and simulation shows that it can per second, in a connection attached to an end-user.

Keyword codes: C.2.2, B.4.1

Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-points of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-packet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementations of existing protocols [3, 35], parallel processing techniques [14, 23, 36], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

† The research was done while Dr. Thia was at Carleton University.

** *Published in *IEEE/ACM Transactions on High-Speed Networks*, Vol. 4, No. 1, pp. 10-20, 1996. © Springer-Verlag New York, Inc. 1996.

INTEL Ex.1015.001

A bypass does not include fast connection setup but also does not interfere with it. **There is no segmentation/reassembly within the bypass path**, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. *The Segmentation and*

EX1015.014

122. Once the relevant bypassable operations have been performed by the ROPE chip, the PDU is copied to the host for further processing, such as reassembly into a larger data block. (Ex. 1015.007 (“Movement of data across the host bus interface are minimized by using an on-chip DMA for fast block data transfer to/from the host system memory”), .009.)

123. Based on my review and understanding of Thia, there is no disclosure of reassembly of the PDUs being performed on the ROPE chip. In fact, Thia explicitly discloses that reassembly is not performed in the bypass path. (Ex.1015.014 (“There is no segmentation/reassembly within the bypass path”).) In other words, Thia’s bypass chip performs some of the protocol processing functions (such as validating checksums, decoding headers, etc.) but then provides the PDUs to the host for reassembly into a larger data block.

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

INTELLECTUAL PROPERTY APPEALS BOARD

INTER PARTIAL REVIEW AND APPEAL BOARD

CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
STATEMENT OF KEVIN ALMEROOTH, PH.D.

¹ Petition in Case IPR2017-01736, has been joined as a
ing.

Alacritech Exhibit 2026

EX2026 at ¶¶ 122-23

-1409, -1410 PO Demonstrative, 14

Thia Discloses OSI, Not IP or TCP/IP

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*) and C.M. Woodside (**)

Newbridge Networks, Inc., Ottawa, Canada (*) and
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada (**)

Abstract — The Reduced Operation Protocol Engine (ROPE) presented here offloads critical functions of a multiple-layer protocol stack, based on the “bypass concept” of a fast path for data transfer. The motivation for identifying this separate processing path is that it involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inner-layer operations such as queue and buffer management, control switching and movement of data across layers, all of which are a significant overhead. ROPE is intended to support high-speed bulk data transfer. The paper describes the design of a ROPE chip for the OSI Session and Transport layer protocols, using VHDL. The design is practical in terms of chip complexity and area, using current gate array technology, and simulation shows that it can support a data rate approaching 1 gigabit per second, in a connection attached to an end-system.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-point of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-packet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementations of existing protocols [5, 35], parallel processing techniques [14, 23, 36], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

† The research was done while Dr. Thia was at Carleton University.

**Published in *IEEE/ACM Transactions on High-Speed Networks*, Vol. 4, Springer-Verlag, New York, New York, 1996.

INTEL Ex.1015.001

are a significant overhead. ROPE is intended to support high-speed bulk data transfer. The paper describes the design of a ROPE chip for the OSI Session and Transport layer protocols, using VHDL. The design is practical in terms of chip complexity and area, using current gate

EX1015.001

125. Nor is there any suggestion that Thia can be used with TCP/IP. For example, Table 1 of Thia references OSI’s Presentation and Session layers, which do not exist in TCP/IP. Additionally, Thia nowhere identifies any layers that are specific to the TCP/IP protocol, such as the Internet layer for example. Also, Thia’s Figure 2 suggests that its NIA is an “FDDI or ATM” device, which refers to Fiber Distributed Data Interface and Asynchronous Transfer Mode, respectively, neither of which utilize TCP/IP protocol.

126. Thia contemplates use with “standard protocols,” which I and anyone of skill in the art would understand refers to protocols that conform to the OSI model, but nowhere discloses that its bypass architecture is compatible with TCP/IP, nor does it test the “feasibility” of its bypass chip with TCP/IP.

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

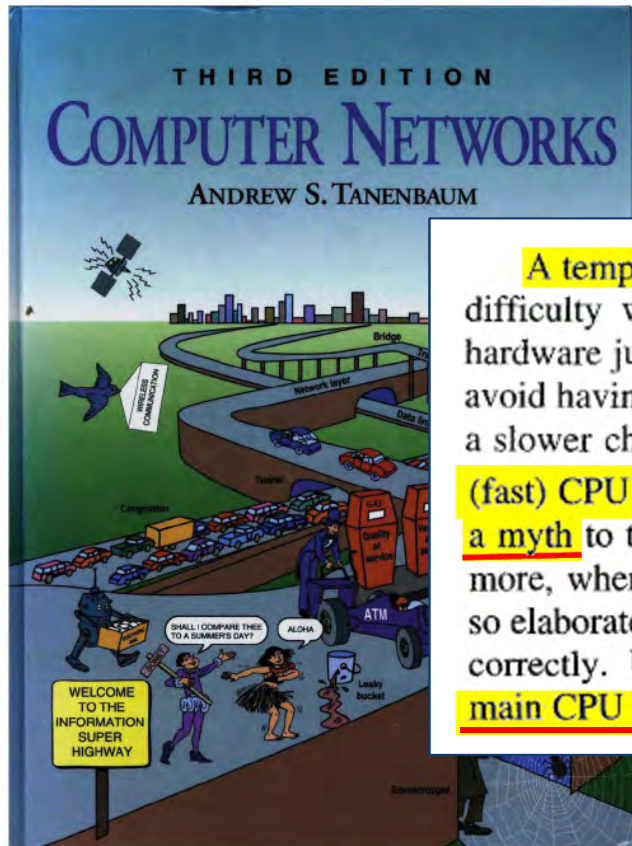
ADAMARK OFFICE
D APPEAL BOARD
AVIUM, INC.

IBIT 2026
MEROETH, PH.D.

17-01736, has been joined as a
Alacritex Exhibit 2026

EX2026 at ¶¶ 125-126
-1409, -1410 PO Demonstrative, 15

Tanenbaum Teaches Away From Offloading TCP/IP Protocol Processing From The Host CPU



INTEL Ex.1006.001

Tanenbaum (EX1006)

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.

EX1006.588-89

Nahum

Appears in "Proceedings of the First Symposium on Operating Systems Design and Implementation," Usenix Association, November 1994.

Performance Issues in Parallelized Network Protocols

Erich M. Nahum, David J. Yates, James F. Kurose, and Don Towsley*

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

Parallel processing has been proposed as a means of improving network protocol throughput. Several different strategies have been taken towards parallelizing protocols. A relatively popular approach is *packet-level parallelism*, where packets are distributed across processors.

This paper provides an experimental performance study of packet-level parallelism on a contemporary shared-memory multiprocessor. We examine several unexplored areas in packet-level parallelism and investigate how various protocol structuring and implementation techniques can affect performance. We study TCP/IP and UDP/IP protocol stacks, implemented with a parallel version of the α -kernel running in user space on Silicon Graphics multiprocessors.

Our results show that only limited packet-level parallelism can be achieved within a single connection under TCP but that using multiple connections can improve available parallelism. We also demonstrate that packet ordering plays a key role in determining single-connection TCP performance, that careful use of locks is a necessity, and that selective exploitation of caching can improve throughput. We also describe experiments that compare parallel protocol performance on two generations of a parallel machine and show how computer architectural trends can influence performance.

1 Introduction

Parallel processing has been proposed as a means of improving network protocol throughput. Two trends motivate the use of parallelism in network processing. First, network bandwidths are increasing by orders of magnitude, with the advent of technologies such as ATM. Second, shared-memory multiprocessors are becoming more common, as

* This research supported in part by NSF under grant NCR-9206908 and ARPA under contract number F19628-92-C-0089. Erich Nahum was supported by an ARPA Research Assistantship in Parallel Processing. David Yates is the recipient of a Motorola Codex University Partnership in Research Grant. The authors can be reached at {nahum, yates, kurose, towsley}@cs.umass.edu.

shown by recent vendor introductions [1, 8, 9]. There is thus an opportunity to exploit the potential of packet-level network protocol processing, and this has become an area of research.

The approach we study here is that of *packet-level parallelism*, sometimes referred to as *thread-per-processor-per-message parallelism*. Originally, Hutchinson and Peterson in the α -kernel [14], it distributes packets across processors, achieving both with multiple connections and within a connection. Packets can be processed on any processor, maximizing flexibility and utilization. Other system approaches include [5, 11].

Several other approaches to parallelism have been proposed and are briefly described here; more surveys can be found in [5, 11]. In *layered* protocols, are assigned to specific processors, messages passed between layers through interprocessor communication. Parallelism gains can be achieved through pipelining effects. An example is *connection-level parallelism* associates connections with single processor or thread, achieving speedup through multiple connections. Multiprocessor STREAMS matches this model [26, 27]. *Functional* decomposes functions within a single process into them to processing elements. Example [19, 23, 25]. The relative merits of one approach or the others depends on many factors, including architecture, the number of connections, whether implementation is in hardware or software, the thread policies employed, and the cost of primitives such as locking and context switching.

Schmidt and Studt [28] show that packet-level parallelism and connection-level parallelism generally perform better than layer parallelism on a shared-memory multiprocessor, due to the context-switching overhead when crossing layers using layer parallelism.

This paper provides an experimental performance study of packet-level parallelism using TCP/IP and UDP/IP protocol stacks. We have conducted this study in the context of a multiprocessor implementation of the α -kernel, which

This paper provides an experimental performance study of packet-level parallelism on a contemporary shared-memory multiprocessor. We examine several unexplored areas in packet-level parallelism and investigate how various protocol structuring and implementation techniques can affect performance. We study TCP/IP and UDP/IP protocol stacks, implemented with a parallel version of the α -kernel running in user space on Silicon Graphics multiprocessors.

EX1079.001

INTEL Ex.1079.001

Nahum (EX1079)

-1409, -1410 PO Demonstrative, 17

Claims 1 & 32 (“associating an operation code with said [first] packet . . .”)

1. A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a network interface for the host computer system to determine if said first packet conforms to a TCP protocol;
 generating a flow key to identify a first communication flow that includes said first packet, wherein said flow key includes a TCP connection for the communication flow;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol; and

processing, by the network interface, said packet according to the TCP connection, including updating a control block representing the TCP connection on the network interface.

EX1001

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface of a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet, wherein said flow key includes a TCP connection for the communication flow and a first hop medium access control (MAC) layer address;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to the TCP protocol:

storing a data portion of said packet in a re-assembly buffer;

storing said header portion in a header buffer; and

processing, by the network interface, said packet according to the TCP connection.

Petitioners Have Not Identified Any "Operation Code"; They Argue It "Would Have Been Obvious"

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

IN THE MATTER OF

A

Case
U.S.Title: INTELLIGENT NETWORK
ACCELERATIONPetition For *Inter Partes* Review
35 U.S.C. §§ 311-319

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

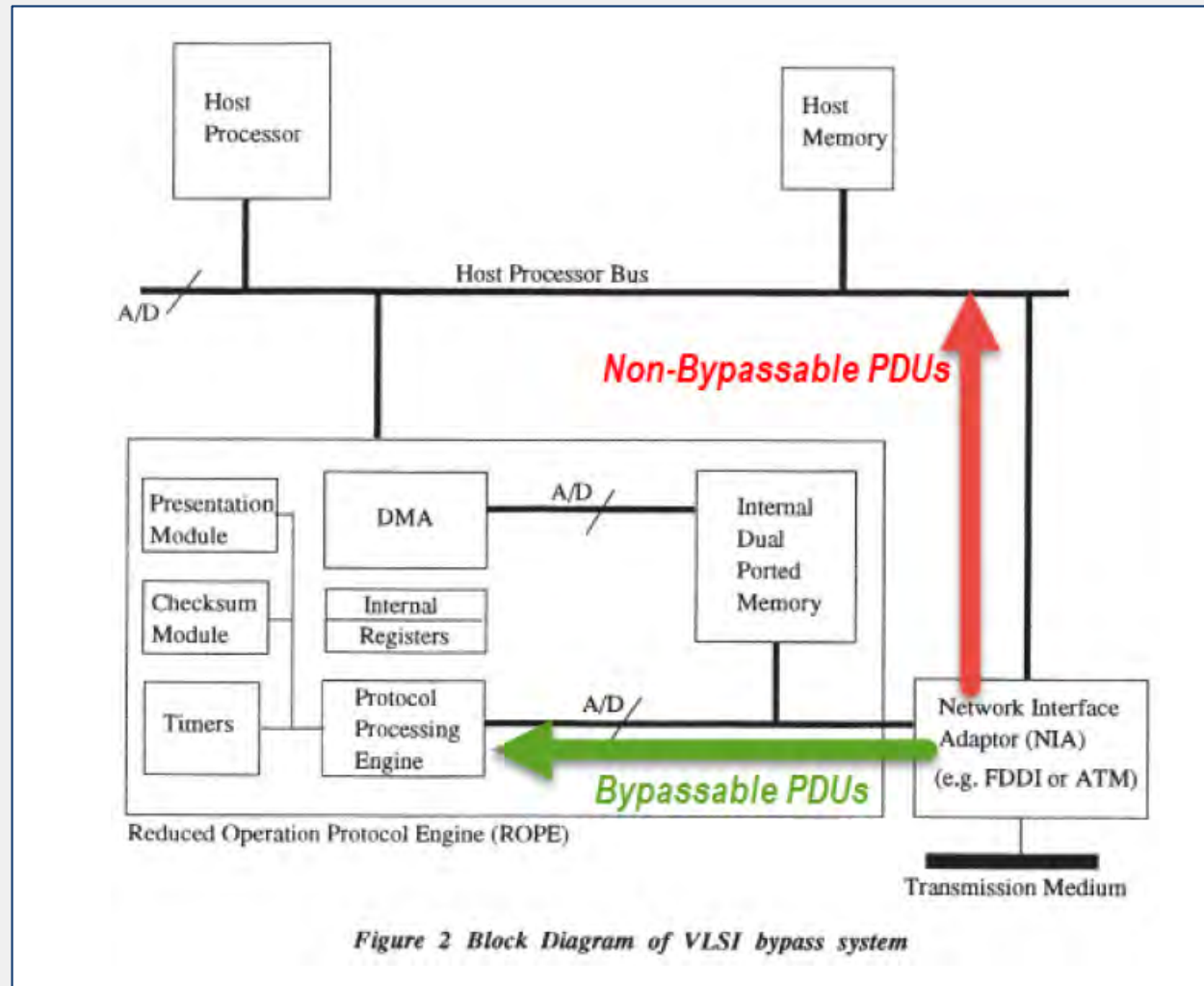
It would have been obvious to a POSA to route the packet to the ROPE chip

using an operation code within the NLA that indicates that the packet is a candidate for fast-path processing (*i.e.*, processing in accordance with the TCP protocol that avoids processing by the host computer). See Ex.1003 at A-19 to A-21. It would

have been obvious to indicate the result of the "receive bypass test" using an operation code (e.g., a bit flag, which Patent Owner has argued provides the claimed operation code¹¹) instructing whether to process the packet on the ROPE chip or on the host. And indeed, Thia teaches that a flag is set signifying that a

1409 Petition at 49-50; 1410 Petition at 53-54

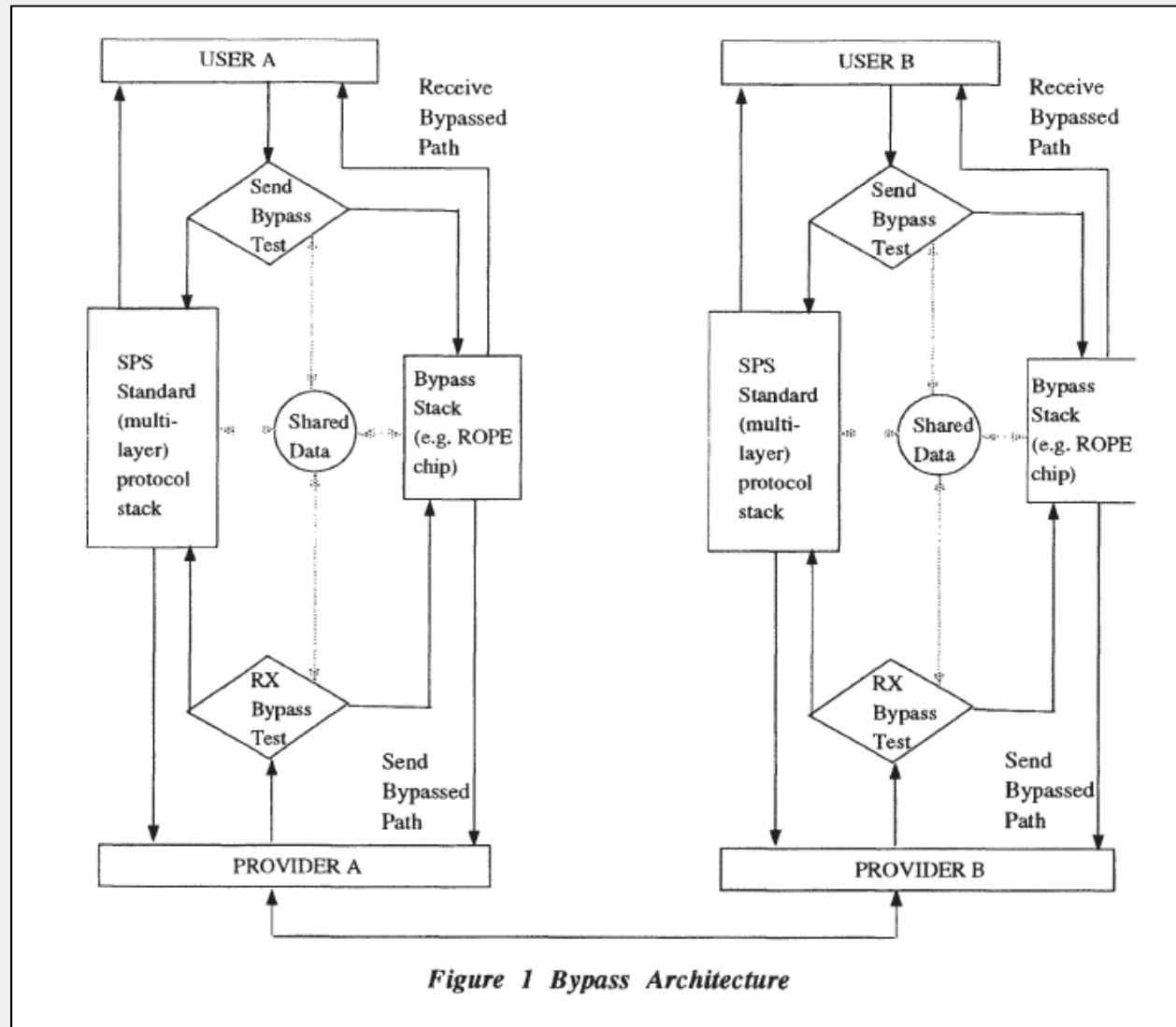
This Performs Bypass *Without* An Operation Code



EX1015.007 (annotations added)

-1409, -1410 PO Demonstrative, 20

This Performs Bypass *Without* An Operation Code



EX1015.003

-1409, -1410 PO Demonstrative, 21

It Would Not Have Been Obvious to Use Thia With an Operation Code—There Is No Reason To Do So

Case No. _____
U.S. Patent No. _____

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has filed a similar petition in this proceeding.

Alacritech

bypass system shown in Fig. 2 (below), Thia does not use an “operation code” for the routing of PDUs. Rather, once the NIA performs the receive bypass test and determines that a PDU is bypassable, all it needs to do is pass the PDU to the Protocol Processing Engine on the ROPE chip for processing by the bypass stack (path annotated with green arrows). Thia does not need to also associate the PDU with an operation code indicating that the PDU is bypassable because only those PDUs that are bypassable are ever passed to the Protocol Processing Engine.¹ PDUs that are not bypassable, meanwhile, are passed directly to the host for processing by the host SPS (path annotated with red arrows). Again, Thia does not need to also associate non-bypassable PDUs with an operation code because only those PDUs that are not bypassable are passed directly to the host. Thus there is no place within Thia’s system for an operation code.

EX2026 at ¶ 139

Decl. of Dr. Kevin Almeroth (EX2026)

It Would Not Have Been Obvious to Use Thia With an Operation Code—There Is No Reason To Do So

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMERO

140. Accordingly, it would not have been obvious to a person of ordinary skill to add an additional, unnecessary, and superfluous operation code and step of associating packets with said operation code to Thia when such a code was not required and would have had no place within the operation of Thia's theoretical system.

EX2026 at ¶ 140

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Decl. of Dr. Kevin Almeroth (EX2026)

Thia's "no in-transit PDU" Flag Is Not An Operation Code

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia^(*) and C.M. Woodside^(**)Newbridge N
Dept. of SystAbstract
critical functi
path for data
involves only
hardware. M
and buffer m
are a signific
paper describ
using VHDL.

array technology, and simulation shows that it can support a data rate approaching 1 gigabit per second, in a connection attached to an end-system.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fibre Optic tec
rates, has shifted the performance
communications processing in the end-
quality-of-service guarantees will r
combination of operating system o
the data stream. To alleviate the en
improved software implementation
[14, 21, 38], special protocol struct
part of the protocol functions to an

The key problems associated v
□ Partitioning the functionality be
lead to a complex additional p
offset the potential gain from d
may be offloaded, but this leav
protocol logic.

^(*) This research was done while Dr. Thia was at Carleton University

^(**) Y. Nishitani et al. (Eds.), *Proceedings for High Speed Networks IV*
© Springer Science+Business Media Dordrecht 1997

INTEL Ex.1015.001

and connection identifiers. Whenever there is a change in the processing path between the SPS and the bypass stack, checks are performed to ensure that there are no outstanding packets in the current path, i.e. "no in-transit PDUs", before the change is made. A more detailed discussion on this is presented in another paper [33].

EX1015.003

Whenever the host processor encounters a switch in the processing path, i.e. from the bypass stack to the SPS, it will issue a BYPASS_SYNC procedure. This procedure will flush the bypass chip of any "in-transit PDU" for that particular connection and return any updated information, for example window control parameters, from the bypass chip to the host in order to maintain state consistency between the two paths. This may occur when

EX1015.010

Thia (EX1015)

Thia's "no in-transit PDU" Is Not An Operation Code

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADE

BEFORE THE PATENT TRIAL AND AP

INTEL CORPORATION, and CAVIUM

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMERO

¹ Cavium, who filed a Petition in Case IPR2017-01
petitioner in this proceeding.

Alacritech Exhibit 2026

143. Thus the "no in-transit PDUs" inquiry relates to synchronization, not a decision to bypass (or not bypass). The "no in-transit PDUs" determination is a corollary to a fully separate (but unexplained in Thia) determination to switch between SPS (host) and bypass (ROPE) processing. (Ex. 1015.003, .004.)

144. In short, the "no in-transit PDUs" determination is not about a packet's eligibility for bypass or fast-past processing. It is about determining whether, given a separate determination to switch a path between SPS (host) and bypass processing, there are outstanding packets on the current path. (*Id.* at .003.)

EX2026 at ¶¶ 143-144

Decl. of Dr. Kevin Almeroth (EX2026)

Thia's "no in-transit PDU" Is Not An Operation Code

Case No. IPR2017-1
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,
Petitioners,

v.

ALACRITECH INC.,
Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined petitioner in this proceeding.

Alacritech Exhibit 2

Decl. of Dr. Kevin Almeroth (EX2026)

consistency between the two paths.” (*Id.* at .010.) In other words, when Thia’s system transitions from processing by the ROPE chip to processing by the host, it needs to move all existing “in-transit PDUs” (i.e., any PDU currently stored on the ROPE chip that is on its way, or “in-transit,” to the host) from the ROPE chip to the host—otherwise there could be faults due to processing conflicts if the bypass chip attempts to process certain PDUs in a connection flow while the host attempts to process others. For example, a single consistent instance of the connection flow likely would not be able to be maintained if there is a state associated with the flow and the flow state changes from PDU to PDU.

142. In instances where the same state is being updated by different processes, “race conditions” can occur. Such conditions have a variable and random outcome depending on which PDU is processed first—an undesirable condition in protocol processing. Such conditions can lead to missed PDUs, incorrect state, resets, and the like. Accordingly, one of ordinary skill would have understood that Thia’s “no-in-transit PDUs” flag relates to synchronization to avoid the aforementioned undesirable behavior.

Thia's "no in-transit PDU" Is Not An Operation Code

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer b

Y.H. Thia (*)¹ and
Newbridge Networks,
Dept. of Systems and

Abstract — The critical functions of a path for data transfer, involves only a small hardware. Multiple-l and buffer managem are a significant over paper describes the de

using VHDL. The design is practical in terms of chip complexity and area, using current gate array technology, and simulation shows that it can support a data rate approaching 1 gigabit per second, in a connection attached to an end-system.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fibre Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-points of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-octet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementation of existing protocols [5, 35], parallel processing techniques [14, 21, 38], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offboard processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

¹ This research was done while Dr. Thia was at Carleton University

^{*} Y. H. Thia et al. (Eds.), *Proceedings for High Speed Networks '97*
© Springer Science+Business Media, Dordrecht, 1997

INTEL Ex.1015.001

The "no-in-transit PDU" test can often be avoided. At the beginning of data transfer on a new connection, it is automatically satisfied. It holds as long as no packet fails a bypass test, and it is sufficient to maintain a flag to indicate this. Once a packet fails, and goes to the SPS, then a full "no-in-transit PDU" test must be performed for each packet until the test succeeds, after which control can go back to the flag. Token management and synchronization points of

EX1015.004

Thia (EX1015)

Thia's "no in-transit PDU" Is Not An Operation Code

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMERO

¹ Cavium, who filed a Petition in Case IPR2017-00000, is the sole petitioner in this proceeding.

146. I disagree with Petitioners' allegations that this flag is somehow related to the claimed "operation code." (*See, e.g.*, Petition at 49-50.) Thia's flag cannot satisfy the claimed "operation code" because it is not "associat[ed] . . . with [a] first packet" at all—it's associated with a system status (i.e., whether the system is in the synchronization mode in which it is looking for any in-transit PDUs on the ROPE chip that need to be flushed to the host). For the same reasons, Thia's flag also does not "indicat[e] a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol," as required by the claims.

Alacritech Exhibit 2026

EX2026 at ¶ 146

Decl. of Dr. Kevin Almeroth (EX2026)

Thia's "BYPASS_START" Is Not An Operation Code

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*)¹ andNewbridge Networks, Inc.
Dept. of Systems and C

Abstract — The ROPE is a critical functions of a network path for data transfer. It involves only a small amount of hardware. Multiple-layer bypass architectures and buffer management are a significant overhead. This paper describes the design of the ROPE using VHDL. The design is implemented using array technology, and simulates at 100 million operations per second, in a connection.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fibre Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-points of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-octet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementation of existing protocols [5, 35], parallel processing techniques [14, 21, 38], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offboard processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

¹ This research was done while Dr. Thia was at Carleton University.

^{*} Y. H. Thia et al. (Eds.), *Proceedings of High Speed Networks '97*
© Springer Science+Business Media, Dordrecht, 1997

INTEL Ex.1015.001

Four high level procedures are made available to the host processor to control the bypass chip, namely: **BYPASS_START**, **BYPASS_DMA**, **BYPASS_SYNC** and **BYPASS_RESTART**. On receiving the first bypassable PDU, the **BYPASS_START** procedure is called. This procedure sets up a bypassable connection by sending information like its initial window flow control parameters and the **DST_REF** field to the bypass chip. These

EX1015.009

Thia (EX1015)

Thia's "BYPASS_START" Is Not An Operation Code

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE

by the host (I/O mapped)." Ex.1015.009. It would have been obvious to call the
BYPASS_START procedure using an operation code to establish the bypass
connection. Ex.1003 at A-21.

Case IPR. No. **Unassigned**
U.S. Patent No. 8,131,880

Title: INTELLIGENT NETWORK INTERFACE DEVICE AND SYSTEM FOR
ACCELERATED COMMUNICATION

Petition For Inter Partes Review of U.S. Patent No. 8,131,880 Under
35 U.S.C. §§ 311-319 and 37 C.F.R. §§ 42.1-.80, 42.100-.123

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

1409 Petition at 51

1409 Petition

-1409, -1410 PO Demonstrative, 30

It Would Not Have Been Obvious To Use An Operation Code With "BYPASS_START"

Case No. IPR2017-1409¹
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined petitioner in this proceeding.

Alacritech Exhibit 2

Decl. of Dr. Kevin Almeroth (EX2026)

149. In particular, Thia discloses that the NIA performs the “receive bypass” test, after which (1) bypassable PDUs are processed via the bypass stack on the ROPE chip before being passed to the host, and (2) non-bypassable packets are passed directly to the host for processing by the host SPS, as shown in Fig. 1 (bypass architecture). Thia further discloses that the BYPASS_START procedure is called “[o]n receiving the first bypassable PDU.” (Ex. 1015.009.) Accordingly, once the first bypassable PDU is received by the ROPE bypass stack (via the Protocol Processing Engine on the ROPE chip, as I previously illustrated in ¶ 139, above), the BYPASS_START procedure is called. There is no suggestion that the BYPASS_START procedure is initiated by anything other than a standard procedure call within Thia’s programming. In other words, once a bypassable packet is received (*i.e.*, the receive bypass test on the NIA receives a bypassable packet and passes it to the ROPE bypass stack for processing), the BYPASS_START procedure is called using a function call. There is no reason one of ordinary skill would have initiated the procedure using an operation code, nor have Petitioners or Dr. Lin even explained how BYPASS_START could have even been called using the claimed “operation code.”

EX2026 at ¶ 149

-1409, -1410 PO Demonstrative, 31

Thia's Incoming PDU Header Does Not Contain An Operation Code

Trials@uspto.gov
571-272-7822

Paper No. 8
Entered: November 21, 2017

UNITED STATES PATENT

BEFORE THE PATENT

INTEL CORP.

ALACRIT

Case I
Patent

Before STEPHEN C. SIU, DANIEL
WILLIAM M. FINK, *Administrative Patent Judges*
SIU, *Administrative Patent Judge*.

processing by the host computer, as recited in claim 1). At this preliminary stage of the proceeding, we do not discern a meaningful difference between the claimed "operation code" that indicates whether a data packet is a candidate for transfer that avoids processing by the host computer, as recited in claim 1, and the component in an "incoming PDU header" that is matched with a template to identify the packet is "bypassable" in the "data transfer phase," as disclosed by Thia. In both cases, a component is used to indicate whether a data packet may bypass processing by a host computer.

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

I. INTRODUCTION

Intel Corporation ("Petitioner") requests *inter partes* review of claims 1, 5-10, 12, 14, 16, 17, 20-23, 27, 28, 45, and 55 of U.S. Patent No.

1409 *Inst. Dec.* at 10; 1410 *Inst. Dec.* at 11

Argument not raised by Petitioners

Thia's Incoming PDU Header Does Not Contain An Operation Code

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*)¹ and C.M. Woodside (*)

Newbridge Networks, Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering

Abstract — The Reduced Operation Protocol Engine (ROPE) is a critical function of a multiple-layer protocol path for data transfer. The motivation for its invention is to reduce the processing overhead of a multiple-layer protocol path. Multiple-layer bypass also obtains a significant overhead. ROPE is intended to reduce the overhead of a multiple-layer protocol path by using VHDL. The design is practical in terms of any technology, and simulation shows that it performs in a corrective manner in an environment.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-point of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-byte processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementations of existing protocols [3, 31], parallel processing techniques [14, 23, 30], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

¹ The research was done while Y.H. Thia was at Queen University.

© 2003 IEEE. Published in *IEEE Transactions on High Speed Networks*, Vol. 11, No. 3, June 2003. This paper is a U.S. Government work and, as such, is in the public domain in the United States of America.

INTEL Ex.1015.001

phase. The receive bypass test matches the incoming PDU headers with a template that identifies the predicted bypassable headers. The bypass stack performs all the relevant

Thia, EX1015.003

152. I disagree. There is no disclosure in Thia of a “component” in the incoming PDU header that is matched to a template or that is otherwise indicative of whether the PDU is “bypassable.” Rather, Thia discloses that a PDU is determined to be “bypassable” based on the results of a “receive bypass test [that] matches the incoming PDU headers with a template that identifies the predicted bypassable headers.” (Ex. 1015.003.) Thia does not disclose or even suggest identifying (or matching) *a component* within the incoming PDU header to determine whether it is bypassable. Rather, Thia discloses matching the *entire* header of the incoming PDU with a template header that is able to predict whether the header is bypassable or not.

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

PTO TRADEMARK OFFICE

LAND APPEAL BOARD

and CAVIUM, INC.,
vs.
H INC.,
per
-1409/
31,880

EXHIBIT 2026
ALMEROOTH, PH.D.

PR2017-01736, has been joined as a

Alacritech Exhibit 2026

Almeroth Decl., EX2026 at ¶¶ 152-153

Thia Does Not Disclose “associating an operation code with said first packet”

1. A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:

* * *

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, including whether said packet is a candidate for transfer to the host computer system that avoids processing said header portion by the host computer system in accordance with said TCP protocol; and

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

154. Additionally, the claims require “*associating* an operation code with said first packet.” Thus even if there was a component within Thia’s incoming PDU header that is indicative of the status of the PDU (including whether the PDU is bypassable), there is no disclosure in Thia of the claimed step of associating such a component with the PDU. Indeed, there is no such “associating” step because all Thia teaches is matching the incoming PDU header to a template to determine whether the PDU is bypassable and, if it is, taking the bypass processing path as opposed to the SPS path on the host (see Fig. 1 of Thia).

UNITED STATES PATENT AND TRADEMARK OFFICE
 BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORPORATION, and CAVIUM, INC.,
 Petitioners,

v.

ALACRITECH INC.,
 Patent Owner

Case IPR2017-1409
 U.S. Patent 8,131,888

PATENT OWNER'S EXHIBIT
 DECLARATION OF KEVIN ALMEROOTH

¹ Cavium, who filed a Petition in Case IPR2017-1409, is a petitioner in this proceeding.

Alacritech Exhibit 2026

Claim 32 (“said flow key includes . . . a first hop medium access control (MAC) layer address”)

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet, wherein said flow key includes a TCP connection for the communication flow and a first hop medium access control (MAC) layer address;

EX1001

Further, it would have been obvious to have a flow key additionally include a next-hop MAC layer address. A flow key can comprise any characters or numbers, and it would have been obvious to have the flow key comprise information relevant to the connection in order to verify the flow key against the information relevant to the connection during lookup. See Ex.1003 ¶¶30-31, A-14; see also Ex.1006.585 (“... some simple function of the two IP addresses and two ports is the key.... Both addresses and both ports must be compared to verify that the correct record has been found.”). A next-hop MAC layer address indicates the physical location a packet is first routed on its way to its final destination for the connection and is therefore information relevant to the connection. See Ex.1003, ¶¶30-31, A-14.

Claim 32 (“said flow key includes . . . a first hop medium access control (MAC) layer address”)

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet, wherein said flow key includes a TCP connection for the communication flow and a first hop medium access control (MAC) layer address;

EX1001

125. Just because the next-hop MAC address may be information that is relevant to the connection does not mean it would have been obvious to store it as part of the flow key. Indeed, there are any number of pieces of information that may be relevant to a TCP/IP connection—not all of which would be stored in the flow key. In my opinion, Petitioners’ and Dr. Lin’s arguments are based on hindsight. Based on the wide array of information available in the different packet headers, it is only through hindsight that the Petitioners and Dr. Lin arrived at exactly the particular piece of information required by the claims—i.e., the first hop MAC address. They do not justify the inclusion of this piece of information as compared to any other piece of information. In my opinion, it was not known to include the first hop MAC address in the flow key.

126. The first hop MAC address is the MAC address of the immediate gateway through which the device receiving the packet or PDU is communicating to the source of the PDU. It is not the MAC address of the source or destination of the PDU. There is no apparent reason why, of all the possible information relevant to a connection, this *first hop MAC address* would have been important enough or necessary to store in the flow key. In my opinion, it would not have been obvious to include it in the flow key.

No Suggestion of the Element in Prior Art



“To imbue one of ordinary skill in the art with knowledge of the invention in suit, when no prior art reference or references of record convey or suggest that knowledge, is to fall victim to the insidious effect of a hindsight syndrome wherein that which only the inventor taught is used against its teacher.”

In re Ethicon, Inc., 844 F.3d 1344, 1355-56 (Fed. Cir. 2017), citing and quoting *W.L. Gore & Assocs., Inc. v. Garlock, Inc.*, 721 F.2d 1540, 1553 (Fed. Cir. 1983)

Claim 45 (1409 Pet.); Claims 32, 41 and 43 (1410 Pet.)

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

storing said packet in a packet memory;
if said header conforms to the TCP protocol:
storing a data portion of said packet in a re-assembly buffer;
storing said header portion in a header buffer; and

43. A computer system for receiving a packet from a network, comprising:

* * *

a processor for processing said first packet and for maintaining a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

41. An apparatus for transferring a packet to a host computer system, comprising:

* * *

a processor, disposed in the network interface, that maintains a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

45. The method of claim 1, wherein said operation code indicates whether the packet corresponds to Transport Control Protocol (TCP).

EX1001

While This does not specify whether the header is parsed to determine if the header conforms to TCP, This explicitly teaches that its bypass architecture is intended for “any standard protocol.” TCP/IP is a standard protocol and

1409 Petition at 43; 1410 Petition at 28

-1409, -1410 PO Demonstrative, 38

Claim 45 (1409 Pet.); Claims 32, 41 and 43 (1410 Pet.)

32. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

storing said packet in a packet memory;
if said header conforms to the TCP protocol:
storing a data portion of said packet in a re-assembly buffer;
storing said header portion in a header buffer; and

43. A computer system for receiving a packet from a network, comprising:

* * *

a processor for processing said first packet and for maintaining a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

41. An apparatus for transferring a packet to a host computer system, comprising:

* * *

a processor, disposed in the network interface, that maintains a TCP connection for the communication flow, the TCP connection stored as a control block on the network interface.

45. The method of claim 1, wherein said operation code indicates whether the packet corresponds to Transport Control Protocol (TCP).

EX1001

While This does not specify whether the header is parsed to determine if the header conforms to TCP, This explicitly teaches that its bypass architecture is intended for “any standard protocol.” TCP/IP is a standard protocol and

1409 Petition at 43; 1410 Petition at 28

-1409, -1410 PO Demonstrative, 39

Claim 41 (“a flow re-assembler, disposed in the network interface, configured to re-assemble . . .”); also Claim 43

14

A Reduced Operation Protocol Engine multiple-layer bypass architecture

Y.H. Thia (*) and C.M. Woodside (**)

Newbridge Networks, Inc., Ottawa, Canada (*) and Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada (**)

Abstract — The Reduced Operation Protocol critical functions of a multiple-layer protocol stack, path for data transfer. The motivation for identifying involves only a small subset of the complete protocol hardware. Multiple-layer bypass also eliminates all and buffer management, control switching and more are a significant overhead. ROPE is intended to be paper describes the design of a ROPE chip for the Q using VHDL. The design is practical in terms of chip array technology, and simulation shows that it can per second, in a connector attached to an end-user.

Keyword codes: C.2.2, B.4.1

Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-point of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-packet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementations of existing protocols [3, 35], parallel processing techniques [14, 23, 36], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

*The research was done while Dr. Thia was at Carleton University.

**Published in *IEEE/ACM Transactions on High-Speed Networks*, Vol. 1, No. 1, Springer-Verlag, New York, 1993.

INTEL Ex.1015.001

A bypass does not include fast connection setup but also does not interfere with it. **There is no segmentation/reassembly within the bypass path**, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. The Segmentation and

EX1015.014

154. Second, Thia also does not disclose a flow re-assembler on the NIC, as it explicitly states that reassembly of the PDUs is *not* performed in the bypass path (i.e., Thia’s ROPE chip does not perform any reassembly). (Ex.1015.014 (“There is no segmentation/reassembly within the bypass path . . .”).) Consistent with my explanation of Thia, above, **Thia’s bypass chip performs some of the PDU protocol processing functions (such as validating checksums, decoding headers, etc.) for OSI protocols, but then provides the entire PDU to the host for reassembly.**

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

TRADEMARK OFFICE
AND APPEAL BOARD

CAVIUM, INC.

NC.

409
1,880

EXHIBIT 2026
ALMEROOTH, PH.D.

2017-01736, has been joined as a

Alacritech Exhibit 2026

Dr. Almeroth Decl., EX2026 at ¶ 154

Claim 41 (“a flow re-assembler, disposed in the network interface, configured to re-assemble . . .”); also Claim 43

interface (Ex.1006.498). This provides an express disclosure. This discloses copying the data portions of packets from the network interface to the host, and thus discloses a flow re-assembler on the network interface to re-assemble data portions of packets within a communication flow. See Ex.1015.005 (“The data portion of a PDU may be physically moved for the following reasons: • Copying between the adaptor buffer and the host system memory.”). Like Tanenbaum96,

1410 Pet. At 77

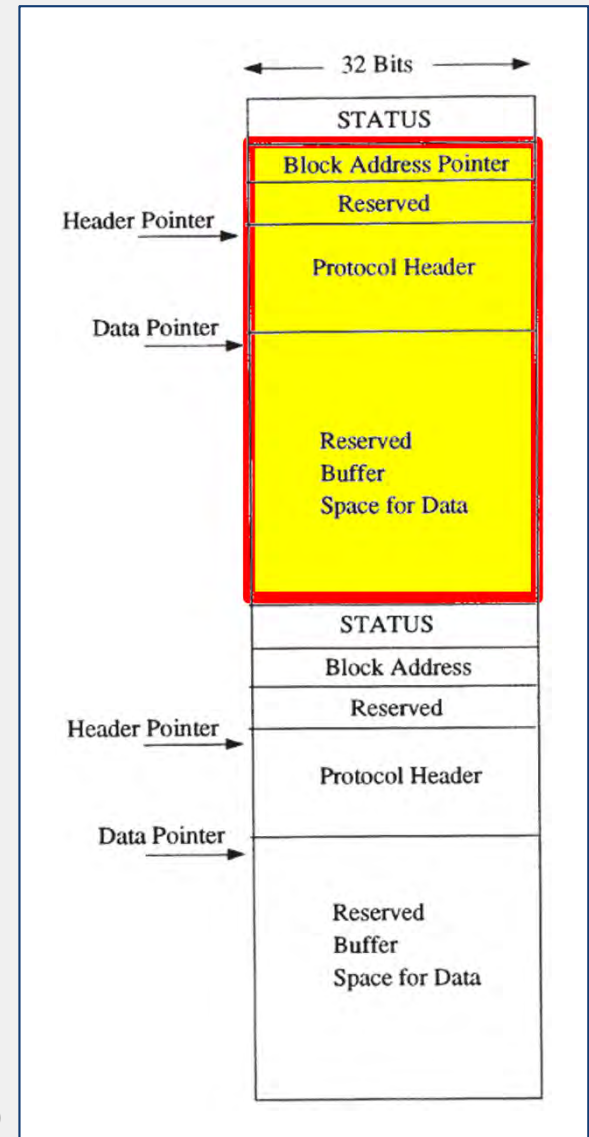
14
A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture
 Y.H. This (*) and C.M. Woodside (**)
 Newbridge Networks, Inc., Ottawa, Canada (*) and
 Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada (**)
 Abstract — The Reduced Operation Protocol Engine (ROPE) presented here offloads critical functions of a multiple-layer protocol stack based on the “bypass concept” of a fast path for data transfer. The motivation for identifying this separate processing path is that it involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some lower-layer operations such as queue and buffer management, context switching and movement of data across layers, all of which

For subsequent bypassable packets, the host processor initiates the BYPASS_DMA procedure which checks for free buffer space in the bypass chip and programs the DMA by sending the starting address pointer where the PDU is located, and its total length.

combination of operating system overhead, protocol complexity, and per-packet processing on the data stream. To alleviate the end-system bottleneck, one may consider new protocols [10], improved software implementation of existing protocols [15, 35], parallel processing techniques [14, 21, 38], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.
 The key problems associated with offload processing include:
 □ Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

This, EX1015.009

This, EX1015.011 (Fig. 4, excerpted and annotated)



-1409, -1410 PO Demonstrative, 41

Claim 55 (1409 Pet.); Claim 43 (1410 Pet.)

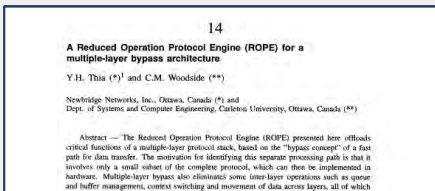
43. A computer system for receiving a packet from a network, comprising:

* * *

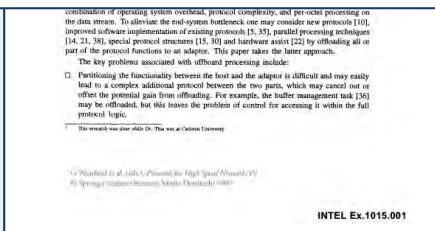
a re-assembler for storing data portions of said multiple packets without header portions in a first portion of said memory; and

55. The method of claim 1, further comprising: receiving, by said communication device, a second packet that corresponds to said communication flow, said second packet including at least a transport layer header and data; and transferring said data from said communication device to said host computer system, without transferring said transport layer header from said communication device to said host computer system.

EX1001

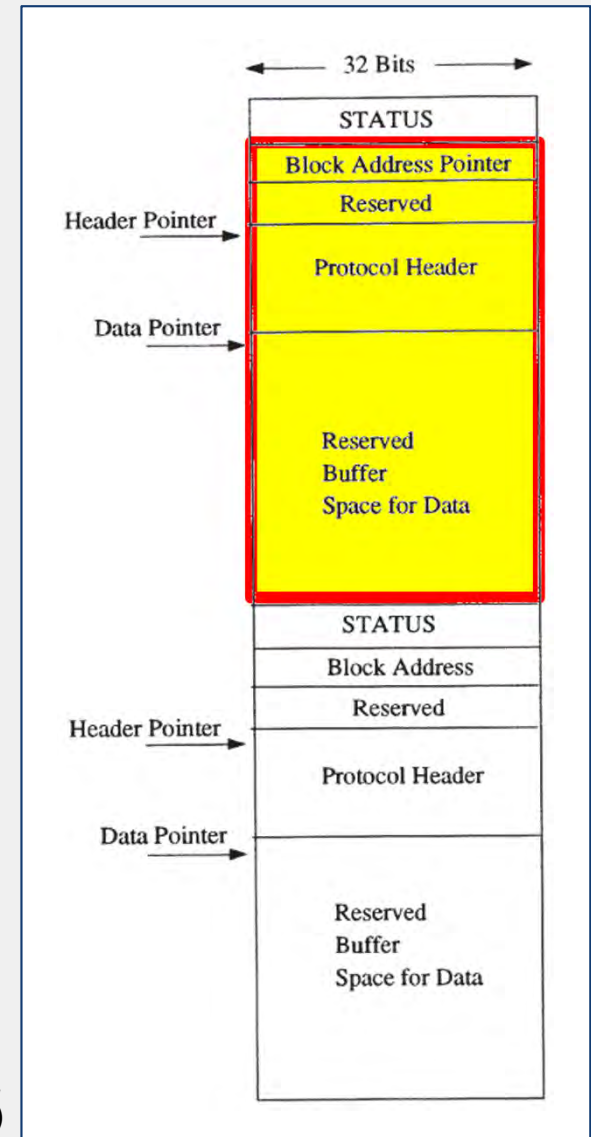


For subsequent bypassable packets, the host processor initiates the BYPASS_DMA procedure which checks for free buffer space in the bypass chip and programs the DMA by sending the starting address pointer where the PDU is located, and its total length.



Thia, EX1015.009

Thia, EX1015.011 (Fig. 4, excerpted and annotated)



-1409, -1410 PO Demonstrative, 42

Claim 55 (1409 Pet.); Claim 43 (1410 Pet.)

55. The method of claim 1, further comprising:
 receiving, by said communication device, a second packet that corresponds to said communication flow, said second packet including at least a transport layer header and data; and
 transferring said data from said communication device to said host computer system, without transferring said transport layer header from said communication device to said host computer system.

EX1001

This discloses transferring data to host system memory without transferring the transport header: "The data portion of a PDU may be physically moved for the following reasons: • Copying between the adaptor buffer and the host system memory...." *Id.* at .005. *See also id.* at Fig. 2 (annotated):

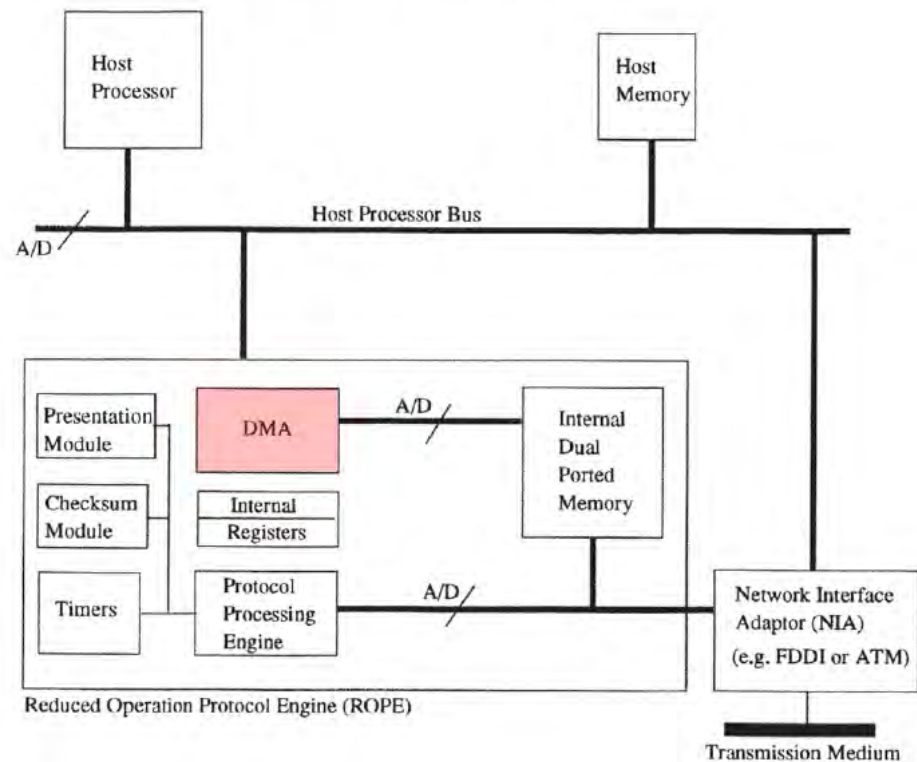


Figure 2 Block Diagram of VLSI bypass system

1409 Petition at 86-87

-1409, -1410 PO Demonstrative, 43

No Motivation to Combine Thia and Tanenbaum

- All Challenged Claims are limited to TCP protocol bypass
- Petitioners admit Thia does not disclose TCP

While Thia does not specify whether the header is parsed to determine if the header conforms to TCP. Thia explicitly teaches that its bypass architecture is intended for “any standard protocol.” TCP/IP is a standard protocol and

1409 Petition at 43; see also 1410 Petition at 73

- Petitioners rely on Tanenbaum for TCP

In view of these similarities, as well as the strong Internet-driven trend towards TCP/IP, a POSA would have been motivated to combine Thia with the TCP/IP protocol teachings of Tanenbaum⁹⁶. See Ex.1003, at ¶112. As described

1409 Petition at 32; see also 1410 Petition at 78

No Motivation to Combine Thia and Tanenbaum

- Tanenbaum teaches speeding up the host software

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;

Tanenbaum, EX1006.583

- Tanenbaum teaches away from offloading TCP protocol processing to, e.g., Thia's ROPE chip

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.

Tanenbaum, EX1006.588-89, -1410 PO Demonstrative, 45

No Motivation to Combine Thia and Tanenbaum

Furthermore, the networking hardware and software have completely changed since the second edition appeared. In 1988, nearly all networks were based on copper wire. Now, many are based on fiber optics or wireless communication. Proprietary networks, such as SNA, have become far less important than public networks, especially the Internet. The OSI protocols have quietly vanished, and the TCP/IP protocol suite has become dominant. In fact, so much has changed, the book has almost been rewritten from scratch.

Tanenbaum, EX1006.016

Given the enormous complexity of the model and the protocols, it will come as no surprise that the initial implementations were huge, unwieldy, and slow. Everyone who tried them got burned. It did not take long for people to associate “OSI” with “poor quality.” While the products got better in the course of time, the image stuck.

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.

Tanenbaum, EX1006.060-.061

No Motivation to Combine Thia and Tanenbaum

Case No. IPR
U.S. Patent No.

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been a petitioner in this proceeding.

Alacritech Ex

167. *Second*, it is precisely the “strong Internet-driven trend towards TCP/IP” in the 1990s and the fact that “OSI protocols have quietly vanished and the TCP/IP protocol suite has become dominant” described in Tanenbaum (Ex. 1006.016) that would have militated against one of ordinary skill using Thia with TCP/IP protocol.

168. In particular, those of ordinary skill would not have sought to use TCP/IP with Thia’s “feasibility study” of a bypass architecture for outdated and unpopular *OSI* protocols, given the undisputed lack of interest in *OSI* in the relevant timeframe. Indeed, Tanenbaum explains that the lack of interest in *OSI* was due, *inter alia*, to “the enormous complexity of the [*OSI*] model and the protocols” which resulted in “initial implementations [that] were huge, unwieldy, and slow.” (Ex. 1006 at .060-061.) Accordingly, there would have been no reason to expect that Thia’s feasibility results for *OSI* protocols would have carried to a TCP/IP implementation.

Decl. of Dr. Kevin Almeroth (EX2026)

No Motivation to Combine Thia and Tanenbaum

- Thia is limited to standard **OSI** protocols

are a significant overhead. ROPE is intended to support high-speed bulk data transfer. The paper describes the design of a ROPE chip for the **OSI Session and Transport layer protocols**, using VHDL. The design is practical in terms of chip complexity and area, using current gate

Thia, EX1015.001

Layer	Procedure	Bypass Chip	Host	Per-Octet (A)	Per-Packet (B)	Per-Group-Of-Packets Aggregated to Per-Packet for bulk data transfer (B)	Remarks
Presentation	Encoding	X		X			
	Encryption	X		X			
	Compression	X		X			
	Context Alteration		X			X	
Session	Synchronization Management		X			X	
	Token management		X			X	
Transport (Class 4)	Checksum (Optional)	X		X			
	Timer Management	X			X	X	Depends on Implementation
	Generation of ACK packets (Flow Control)	X				X	
	Resequencing	X			X		

* * *

Table 1 Bypassable versus Non-bypassable functions

Thia, EX1015.006

-1409, -1410 PO Demonstrative, 48

No Motivation to Combine Thia and Tanenbaum

Case No. IPR
U.S. Patent No.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been a petitioner in this proceeding.

Alacritech Exhibit 2026

171. Additionally, Table 1 identifies “Bypassable versus Non-bypassable functions” for the Presentation, Session, and Transport (Class 4) layers of the OSI model. (Ex. 1015.006.) The Presentation and Session layers only exist as part of the OSI model, not TCP/IP, and Transport (Class 4) is an OSI-specific transport layer defined by ISO/IEC 8073/ITU-T Recommendation X.224, “Information Technology - Open Systems Interconnection – Protocol for providing the connection-mode transport service,” which again does not relate to the TCP/IP protocol. Those of skill in the art would have known that Transport (Class 4) or “TP4” used short reference numbers to refer to connections between source and destination, as opposed to the so-called four-tuples of source and destination addresses and sockets used in TCP/IP as identifiers.

Decl. of Dr. Kevin Almeroth (EX2026)

No Motivation to Combine Thia and Tanenbaum

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*) and C.M. Woodside (**)

Newbridge Networks, Inc., Ottawa, Canada (*) and Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada (**)

Abstract — The Reduced Operation Protocol Engine (ROPE) presented here offloads critical functions of a multiple-layer protocol stack, based on the “bypass concept” of a 4 path for data transfer. The motivation for identifying this separate processing path in this involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inner-layer operations such as queuing and buffer management, control switching and movement of data across layers, all of which are a significant overhead. ROPE is intended to support high-speed bulk data transfer. This paper describes the design of a ROPE chip for the OSI Session and Transport layer processing using VHDL. The design is practical in terms of chip complexity and area, using current gate array technology, and simulation shows that it can support a data rate approaching 1 giga-bit per second, in a connection attached to an end-system.

Keyword codes: C.2.2, B.4.1

Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-point of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-byte processing of the data stream. To alleviate the end-system bottleneck one may consider new protocols [1], improved software implementations of existing protocols [2, 3], parallel processing techniques [14, 23, 36], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may even lead to a complex additional protocol between the two parts, which may cancel out the potential gain from offloading. For example, the buffer management task [1] may be offloaded, but this leaves the problem of control for accessing it within the host protocol logic.

*The research was done while Dr. Thia was at Carleton University.

**As published in: *IEEE/ACM Transactions on High Speed Networks*, Vol. 4, Springer-Verlag, New York, New York, 1996.

INTEL Ex.1015.001

Figure 2 Block Diagram of VLSI bypass system

Thia, EX1015.007

172. Additionally, Thia’s Figure 2 shows a “Block Diagram of VLSI bypass system” suggesting that Thia’s NIA is an “FDDI or ATM” device. (Ex. 1015.007.) Neither FDDI nor ATM utilize the TCP/IP protocol.

173. Thia’s authors even explained that Thia is “based on the ‘protocol bypass concept’” presented in a separate paper that was *also* directed to OSI protocols: [37] “The protocol bypass Concept for High Speed OSI Data transfer.”

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,
Petitioners,
v.
ALACRITECH INC.,
Patent Owner

Case IPR2017-1409
U.S. Patent 8,131,880

PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

who filed a Petition in Case IPR2017-01736, has been joined as a
s proceeding.

Alacritech Exhibit 2026

No Motivation to Combine Thia and Tanenbaum

14

A Reduced Operation Protocol Engine (ROPE) for a multiple-layer bypass architecture

Y.H. Thia (*)¹ and C.M. Woodside (**)

Newbridge Networks, Inc., Ottawa, Canada (*) and
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada

Abstract — The Reduced Operation Protocol Engine (ROPE) presented here critical functions of a multiple-layer protocol stack, based on the “bypass concept” path for data transfer. The motivation for identifying this separate processing path involves only a small subset of the complete protocol, which can then be implemented in hardware. Multiple-layer bypass also eliminates some inner-layer operations such as and buffer management, control switching and movement of data across layers, all are a significant overhead. ROPE is intended to support high-speed bulk data transfer paper describes the design of a ROPE chip for the OSI Session and Transport layer processing using VHDL. The design is practical in terms of chip complexity and area, using current array technology, and simulation shows that it can support a data rate approaching per second, in a connection attached to an end-system.

Keyword codes: C.2.2, B.4.1
Keywords: Network Protocols, Data Communications Devices

1 Introduction

The advent of Fiber Optic technology, which offers high bandwidth and low bit error rates, has shifted the performance bottleneck from the communications channel to the communications processing in the end-point of the system [26]. Other trends such as improved quality-of-service guarantees will reinforce this effect. The heavy processing load is due to a combination of operating system overhead, protocol complexity, and per-packet processing on the data stream. To alleviate the end-system bottleneck one may consider new protocols [10], improved software implementations of existing protocols [3, 35], parallel processing techniques [14, 23, 36], special protocol structures [15, 30] and hardware assist [22] by offloading all or part of the protocol functions to an adaptor. This paper takes the latter approach.

The key problems associated with offload processing include:

- Partitioning the functionality between the host and the adaptor is difficult and may easily lead to a complex additional protocol between the two parts, which may cancel out or offset the potential gain from offloading. For example, the buffer management task [36] may be offloaded, but this leaves the problem of control for accessing it within the full protocol logic.

¹ The research was done while Dr. Thia was at Carleton University.

^{**} Y.H. Thia is at 1409 U.S. Patent No. 8,131,880
© Springer Science+Business Media Dordrecht 1987

INTEL Ex.1015.001

host processor is also relieved of acknowledgment processing. An existing implementation of the OSI stack can be adapted for bypassing with only a small modification of the original software, thus providing an easy migration path for current systems.

Thia, EX1015.014

for current systems.” (Id. at .014.) Thus, those of ordinary skill would not have been motivated to implement Thia’s system by modifying TCP/IP stack software when Thia recommended that “an easy migration” path would be to modify existing OSI stack software.

176. Indeed, Thia’s recommendation teaches away, or at the very least dissuades, from a TCP/IP implementation, as it would not have made sense for one of ordinary skill to attempt using Thia with a TCP/IP protocol stack when Thia itself recommends an easy implementation would be to modify an existing OSI stack, thereby implying that using any other stack (such as the TCP/IP protocol stack) would not have been easy.

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

TRADEMARK OFFICE
—
AND APPEAL BOARD
—
CAVIUM, INC.,
—
NC,
—
409,
1,880
—

EXHIBIT 2026
ALMEROOTH, PH.D.

2017-01736, has been joined as a

Alacritech Exhibit 2026

No Motivation to Combine Thia and Tanenbaum

- Thia was aware of TCP/IP, but never suggested extending its system to TCP protocol

data link layer has been disappointing so far. In [8], dedicated VLSI chips are used to support TCP checksums. Also, some newer lightweight transport protocols are specially designed for VLSI implementation [1, 3].

Thia, EX1015.002

throughput of 2.362 Gbps. This is consistent with results presented by Clark [5] which claims that TCP processing can achieve up to 800 Mbps (without checksum) on current RISC-based processors. With OSI checksum, the throughput performance drops to 313.3 Mbps. This

Thia, EX1015.013

Secondary Considerations: Long-Felt Need

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality

Ex. 2031 at 1 (1993 IBM Research Division ACM SIGCOMM article)

1 Introduction

Many researchers have observed that the performance of remote applications have not kept pace with modern communication network speeds. Part of this imbalance is attributed to the protocols used by the remote applications, namely, UDP/IP and TCP/IP. It is now widely believed that the problems with these protocols are not inherent in the protocols themselves but in their particular implementations [Dalt93, Whet95]. The following factors are cited as contributors to the poor performance of UDP/IP and TCP/IP:

Ex. 2032 at 1 (1995 Univ. of Berkeley paper)

Secondary Considerations: Long-Felt Need

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to “bottlenecks” in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication pro-

Ex. 2033 at 1 (1990 IEEE article from AT&T Bell Labs)

Abstract

Server network performance is increasingly dominated by poorly scaling operations such as I/O bus crossings, cache misses and interrupts. Their overhead prevents performance from scaling even with increased CPU, link or I/O bus bandwidths. These operations can be reduced by redesigning the host/adaptor interface to exploit additional processing on the adapter. Offloading processing to the adapter is beneficial not only because it allows more cycles to be applied but also of the changes it enables in the host/adaptor interface. As opposed to other approaches such as RDMA, TCP offload provides benefits without requiring changes to either the transport protocol or API.

Ex. 2034 at 1 (2005 USENIX Conference paper from IBM)

Secondary Considerations: Long-Felt Need

179. The nexus between the long-felt need and the claimed invention is clear and direct. **The aforementioned bottlenecks are all solved by the accelerated network processing technologies recited in the challenged claims.** In addition, the technologies recited in the challenged claims alleviated processing demand on the host CPU in such a way as to permit faster network throughput and transmission/reception speeds—the precise long-felt but unresolved need in the industry outlined above.

1409 Ex. 2026 at 70; see also 1410 Ex. 2026 at 78

UNITED STATES PATENT AND TRADEMARK OFFICE
 BEFORE THE PATENT TRIEBUNAL
 INTEL CORPORATION
 Petitioner
 v.
 ALACRITECH, INC.
 Patent Owner
 Case IPR2017-01736
 U.S. Patent 8,151,880

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.

Secondary Considerations: Industry Praise

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

Ex. 2039 at 4 (2001 HP Labs research paper)

Secondary Considerations: Industry Praise

“The ANX 1500 is an evolutionary advancement of Alacritech’s long standing leadership in protocol acceleration, which is now applied not just to protocols, but to the optimization of all storage system interactions,” said Jeff Boles, a technology analyst with Taneja Group. “The ANX 1500 substantially augments the performance of existing NAS investments from the best place possible – from right in the customer’s Ethernet network – without reconfiguration or changes in management for the existing infrastructure. We think Alacritech is setting the stage for a next generation of solutions that will accelerate storage from outside the storage array, and more cost effectively share an investment in performance across many storage systems and clients. I’ve talked to early-stage customers using the product, and they believe it’s game-changing.”

Ex. 2040 at 3 (Shoreline Ventures Press Release)

-1409, -1410 PO Demonstrative, 57

Secondary Considerations: Industry Praise

183. The industry universally praised commercial embodiments of the features described in the challenged claims. HP found that the “Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed” and “achiev[e] lower processor utilization than native NT’s TCP/IP protocol stack for transmission of large enough messages.” (Ex. 2039 at ¶ 4.) HP found that the test performance of the Alacritech iNIC “raises challenging questions for [HP’s] choice of implementation technology because “the Alacritech approach is definitely better than our offload.” *Id.*

UNITED STATES PATENT
 OFFICE
 BEFORE THE PATENT
 TRIAL AND APPEALS
 BOARD
 IN THE
 CASE OF
 ALACRITECH, INC.
 Petitioner,
 v.
 HP INC.,
 Respondent.
 Case IPR2017-01728
 U.S. PATENT AND
 TRADEMARK OFFICE

CORRECTED PATENT
 DECLARATION OF KEVIN ALMEROOTH, PH.D.

1409 Ex. 2026 at 71-72; see also 1410 Ex. 2026 at 80

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-00001985280.6

Alacritech Exhibit 2026

Secondary Considerations: Failure of Others

To this day, TCP offload has never firmly caught on in the commercial world (except sometimes as a stopgap to add TCP support to immature systems [16]), and has been scorned by the academic community and Internet purists. This paper starts by analyzing why TCP offload has repeatedly failed.

Ex. 2041 at 2 (2003 HP Labs paper presented at HotOS IX conference)

TCP offload has been unsuccessful in the past for two kinds of reasons: fundamental performance issues, and difficulties resulting from the complexities of deploying TCP offload in practice.

Id. at 2

Secondary Considerations: Failure of Others

offload were mismatched to the applications in question.” *Id.* The TCP offload described above is a form of network processing offload that is described by the challenged claims, and this failure of others therefore has a direct nexus to the claimed inventions.

1409 Ex. 2026 at 72-73; see also 1410 Ex. 2026 at 81

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

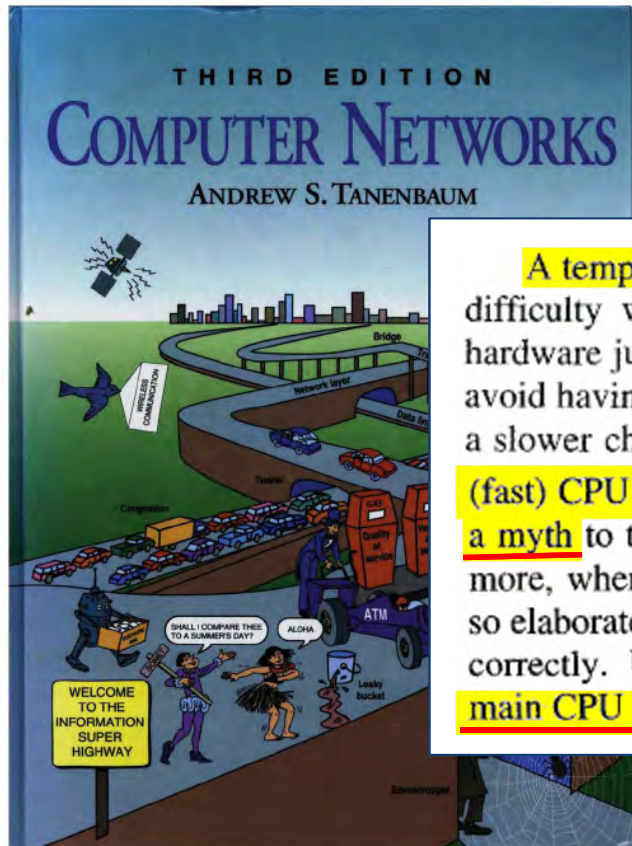
Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROETH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

Secondary Considerations: Skepticism



INTEL Ex.1006.001

Tanenbaum (Ex. 1006)

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.

Ex. 1006.588-89

Secondary Considerations: Skepticism

IP Storage and the CPU Consumption Myth

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

Abstract

This paper addresses a key issue that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2.

The
acceler
CPU
been
comm
unme
Ex
date
many
I/O p
Howe
of arc
rapid
A
I/O)
proce
serve
from
starte
task,
as the
some
Some
or w
and s
usual
and e
envir
costly
for if
the p
kills
gener
the c
Si
accel
of en
ever

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

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

Ex. 2300 at 194 (2001 paper by Petitioner's expert, Dr. Horst)

Secondary Considerations: Skepticism

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

Networks customized to storage networking, such as Fibre 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 paper 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.

2. The Historical A

There are many historical examples of hardware accelerators to offload processor work from the main CPU. Some examples, such as network communications processors, have been successful, but others have fallen far short of unmet expectations.

Examples of front-end accelerators date from the early days of computing. In many systems, the primary I/O processor to offload the main CPU. However, it has become in vogue to use a different architecture to deliver a faster rate of technology change.

A specific recent example is the use of an I/O processor, such as an Intel 8086, to offload the main CPU from its attached I/O devices. The idea started in the 1960s with embedded processors, but its performance dropped significantly as the main CPU. If the performance of the accelerator drops below that of the main CPU, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that hardware accelerators should have been used more extensively. However, the cost of embedded programming and testing every protocol worthy of a name is high, and it is difficult to stay afloat with new protocols proposed frequently. Most are far from stable, and even when formally approved, there are often significant enhancements and bug fixes. It seems extremely

as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Secondary Considerations: Skepticism

(Ex. 2300 at 194.) (emphasis added). This level of skepticism and teaching away is directly related to the heart of the claimed invention – offloading network processing, and therefore has a direct nexus to the challenged claims.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,

Petitioners,

- v. -

ALACRITECH INC.,

Patent Owner

Case IPR2017-1409¹
U.S. Patent 8,131,880

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

1409 Ex. 2026 at 81; see also 1410 Ex. 2026 at 83

-1409, -1410 PO Demonstrative, 64

'880 Patent – Contingent Motions to Amend

Proposed Claim 61

61. (proposed substitute for claim 1) A method of transferring a packet to a host computer system, wherein the packet is received at a communication device from a network, comprising:

storing a first packet received at a network interface for the host computer system in a packet memory;

* * *

if said first packet conforms to the TCP protocol:

storing a data portion of said first packet in a re-assembly buffer;

storing said header portion in a header buffer, wherein the header buffer is separate from the packet memory;

re-assembling said data portion of said first packet with a data portion of a second packet in the communication flow; and

processing, by the network interface, said first and second packets according to the TCP connection, including updating a control block representing the TCP connection on the network interface.

'880 Patent – Contingent Motions to Amend

Proposed Claim 79

79. (proposed substitute for claim 32) A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

* * *

storing said header portion in a header buffer, wherein the header buffer is separate from said packet memory;

re-assembling the data portion of said packet with a data portion of another packet in the communication flow; and

processing, by the network interface, said packet and said other packet according to the TCP connection.

** substantially similar amendments to remaining independent claims 41 (proposed substitute claim 85) and 43 (proposed substitute claim 87)*

Support for “storing a first packet . . . in a packet memory”

[0546] The INIC also has a Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, moves the frame into a DRAM buffer and queues the frame address and status for processing by the Receive CPU into one of the hardware queues mentioned above.

* * *

[0593] As frames are received by the INIC from a network, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte +1 of this buffer

* * *

[0763] External SDRAM provides frame buffering, which is configurable as 4 MB, 8 MB, 16 MB or 32 MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224 MB/s. The buffer provides temporary storage of both incoming and outgoing frames.

'237 Application, EX2025

automatic movement of input frames into DRAM buffers from the MAC Fifos.

* * *

Receive input queue: it is expected that hardware will automatically DMA arriving frames into frame buffers and queue an event into a RCV-event queue.

* * *

External SDRAM provides frame buffering, which is configurable as 4MB, 8MB, 16MB or 32MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224MB/s. The buffer provides temporary storage of both incoming and outgoing frames. The protocol processor accesses the frames

'809 Provisional, EX2019 at 53, 56, 76

-1409, -1410 PO Demonstrative, 67

Support for Storing the Header In a Header Buffer Separate From Packet Memory

[0593] As frames are received by the INIC from a network, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte +1 of this buffer is queued into the Q_RECV queue. The pointer contains a bit (bit 29) that informs the microcode if this frame is definitely not a fast-path candidate (e.g., not TCPIP, or has an error of some sort). Receive frame processing involves extracting this pointer from the Receive hardware queue, and setting up a DMA into an SRAM header buffer of the first X bytes from the DRAM frame buffer. The size of the

'237 Application, EX2025

2. The method of claim 1, wherein said parsing comprises: copying a header portion of said first packet into a header memory; and
8. The method of claim 1, further comprising storing said first packet in a packet memory prior to said transferring.
18. The method of claim 1, wherein said first packet is determined to conform to said pre-selected protocol, said transferring comprising:
 - storing a data portion of said first packet in a re-assembly storage area, wherein said re-assembly storage area is configured to only store data portions of packets in said first communication flow; and
 - storing one or more headers from said header portion in a header storage area.

'237 Application, EX2025

automatic movement of input frames into DRAM buffers from the MAC Fifos.

* * *

There is considerable hardware assist here. The first step in receive processing is to dma the frame header into an SRAM header buffer. It is useful for header validation to be

'809 Provisional, EX2019 at 53, 57

-1409, -1410 PO Demonstrative, 68

"Separate From" Is Not Indefinite

24. A POSA would have understood that to have a header buffer be "separate from" a packet memory could mean that the header buffer and packet memory are located on the same memory device, but the physical location on the memory device where the header is stored is different from the physical location on the memory device where the packets are stored. Alternatively, "separate from" could refer to the memory device itself, such that the header buffer is on a different memory device than the packet memory. Or, "separate from" may not be talking about separateness in a physical sense, but rather in a virtual sense. Specifically, a POSA would have understood that it could mean that the virtual address for the header is separate from the virtual address for the packet.

Decl. of Dr. Lin, EX1210

19. Consistent with Dr. Lin's own examples, a POSA would understand the header buffer "separate from" the packet memory means the header buffer cannot encompass the same memory location(s) as the packet memory. The packet memory stores the entire packet and the header buffer is a separate memory device or a separate location in memory that stores just the header portion of the packet.

Decl. of Dr. Almeroth, EX2305, -1409, -1410 PO Demonstrative, 69

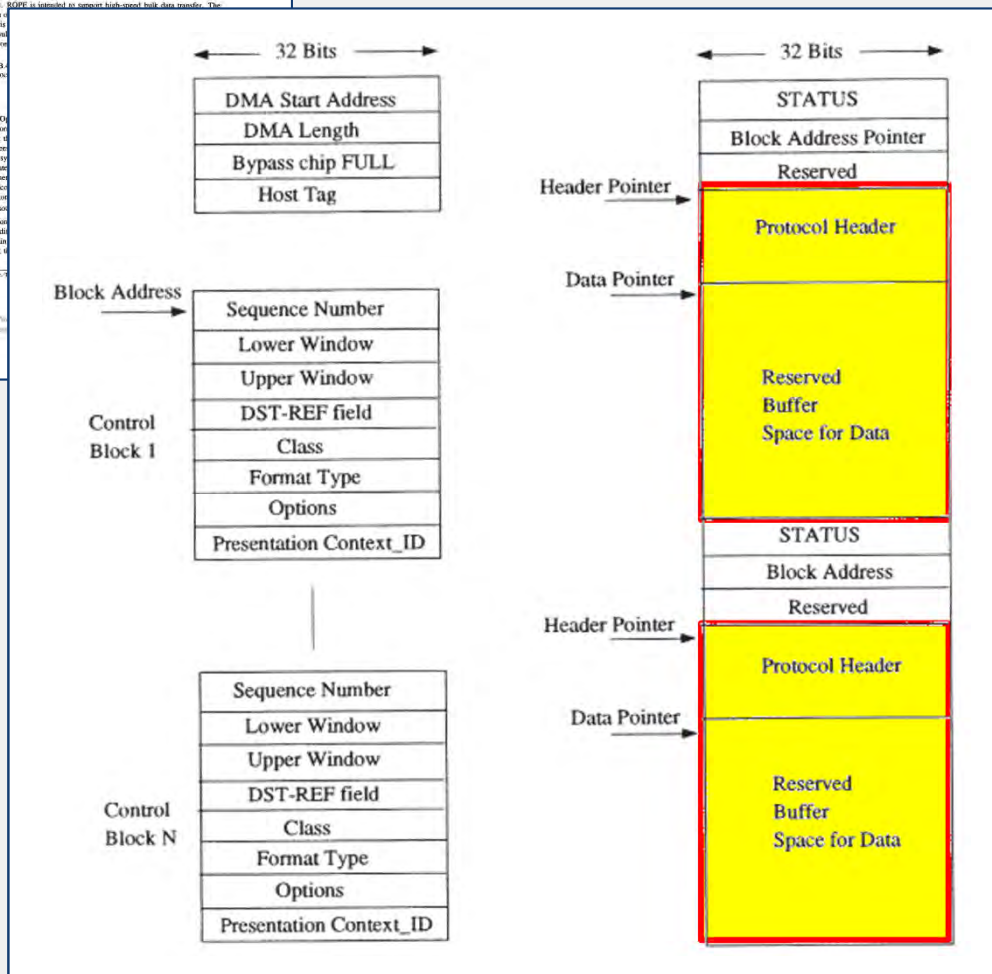
The Cited Art Does Not Disclose Storing the Header In a Header Buffer Separate From Packet Memory

A Reduced Operation multiple-layer bypass
 Y.H. This (*) and C.S.
 Newbridge Networks, Inc.,
 Dept. of Systems and Com.
 Abstract — The basic
 critical functions of a multi-
 path for data transfer. The
 involves only a small set of
 hardware. Multiple-layer bypass also eliminates some inter-layer operations such as queue
 and buffer management, control switching and movement of data across layers, all of which
 are a significant overhead. This paper describes the design of
 using VHDL. The design is
 array technology, and simulates
 per second, in a connective
 Keyword codes: C.S.2, B.
 Keyword: Network Protoc.

1 Introduction
 The advent of Fiber Op-
 tics, has defined the perfor-
 mance processing in a
 quality-of-service guarantee
 combination of operating in
 the data stream. To alleviate
 improved software implemen-
 tation. The key problems asso-
 ciated with the design of a
 protocol function.
 Partitioning the function
 lead to a complex addi-
 tion of the potential gain
 may be offsetted by a
 protocol logic.
 The research was done while the

© 1995 IEEE. All rights reserved.
 This paper is published in IEEE/ACM
 Transactions on Networking, Vol. 3, No. 4,
 August 1995.

retransmission strategy was used. For a large window, the on-chip buffer may not be sufficient to hold the unacknowledged data packets for retransmission or to buffer data packets for resequencing, and slower external memory would be needed.



EX1015.011

The Cited Art Does Not Disclose Storing the Header In a Header Buffer Separate From Packet Memory

Filed: April 4, 2018

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

This also discloses storing said packet received at the network interface in a packet memory. For example, Figure 4 illustrates a packet buffer on the ROPE chip memory for storing multiple packets.

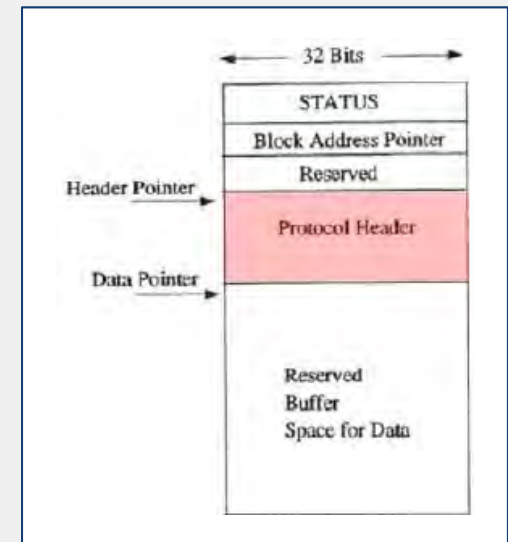
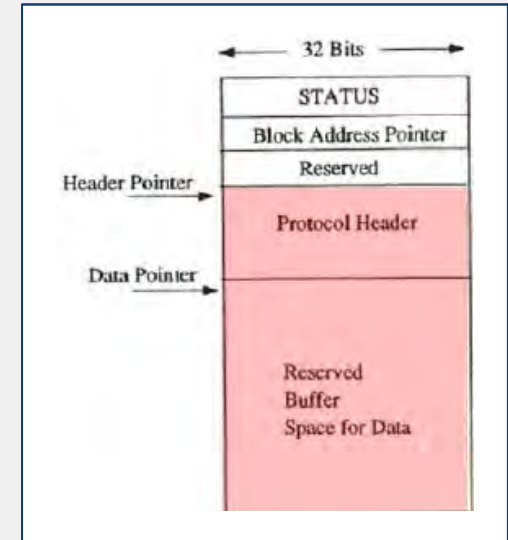
Title: INTELLIGENT NETWORK INTERFACE DEVICE AND SYSTEM FOR

PETITION
CONT

explained for claim limitation 61.a. The header portion is stored in the header buffer of a packet memory. For example, Figure 4 of This illustrates the header portion being stored in the header buffer of the ROPE chip's packet memory:

Mail Stop "A"
Patent Trial & Appeal Board
U.S. Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

¹ Cavium, Inc., which filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.



1409 MTA Opp. at 12-13, 20-21

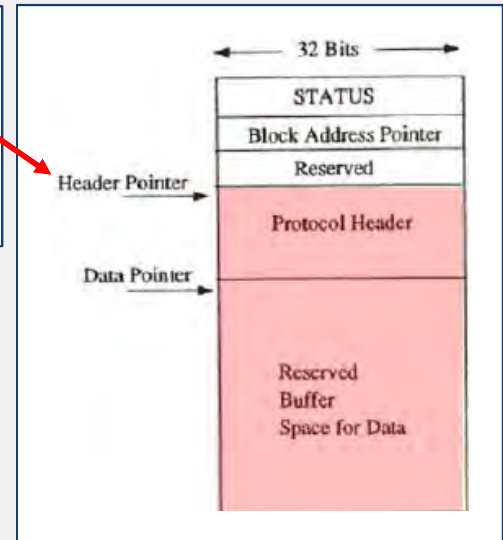
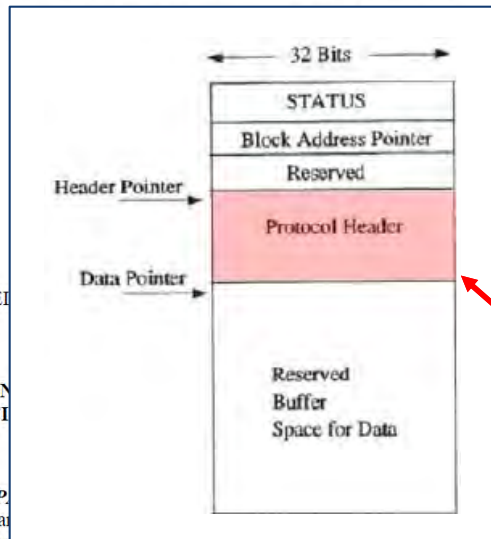
-1409, -1410 PO Demonstrative, 71

The Cited Art Does Not Disclose Storing the Header In a Header Buffer Separate From Packet Memory

This also discloses storing said packet received at the network interface in a packet memory. For example, Figure 4 illustrates a packet buffer on the ROPE chip memory for storing multiple packets.

UNI

BEFORE THE PATENT TRIAL AND APPEAL BOARD



Title: INTELI

PETITION CONTI

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

¹ Cavium, Inc., which filed a Petition in Case IPR2017-01736, has been joined as a petitioner in this proceeding.

explained for claim limitation 61.a. The header portion is stored in the header buffer of a packet memory. For example, Figure 4 of This illustrates the header portion being stored in the header buffer of the ROPE chip's packet memory:

A POSA would have found it obvious that, after buffering a packet in external packet memory for resequencing, the same packet would be transferred to the ROPE's internal packet memory, as shown in Figure 4 above, and have its header portion stored in a header buffer. Specifically, it would have been obvious

1409 MTA Opp. at 12-13, 20-21

1409 MTA Opp. at 22-23

-1409, -1410 PO Demonstrative, 72

Support for Packet Data Re-Assembly

[0763] External SDRAM provides frame buffering, which is configurable as 4 MB, 8 MB, 16 MB or 32 MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224 MB/s. The buffer provides temporary storage of both incoming and outgoing frames. The protocol processor accesses the frames within the buffer in order to implement TCP/IP and NETBIOS. Incoming frames are processed, assembled then transferred to host memory under the control of the protocol processor. For transmit, data is moved from host memory to buffers where various headers are created before being transmitted out via the Mac.

'237 Application, EX2025

host. However for SMB, the INIC is performing reassembly of data when the frame consists of headers and data. So there may not yet be a complete SMB to pass to the host.

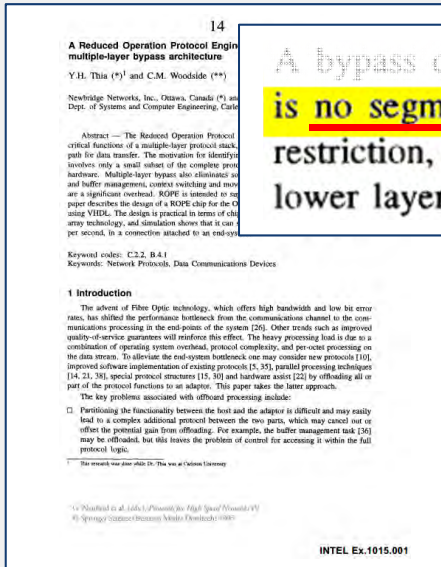
* * *

within the buffer in order to implement TCP/IP and NETBIOS. Incoming frames are processed, assembled then transferred to host memory under the control of the protocol processor. For transmit, data is moved from

'809 Provisional, EX2019 at 57, 76

-1409, -1410 PO Demonstrative, 73

The Cited Art Does Not Disclose Packet Data Re-Assembly



A bypass does not include fast connection setup but also does not interfere with it. **There is no segmentation/reassembly within the bypass path**, but we do not see this as a major restriction, as research suggests that fragmentation of PDUs should be restricted only to the lower layers and should occur only once in the protocol stack [23]. The Segmentation and

EX1015.014

28. It is also my opinion that Thia and Tanenbaum also fail to disclose re-assembly of packet data. In particular, Thia's ROPE chip does not perform any reassembly of PDUs into large data blocks. Indeed, Thia explains at page 14 that "[t]here is no segmentation/reassembly within the bypass path" In other words, Thia's ROPE chip performs some of the protocol processing functions (such as validating checksums, decoding headers, and the like) but then provides the entire PDU to the host for reassembly into a larger data block.

29. Additionally, Tanenbaum does not disclose the NIC re-assembling data portions of two packets in the same communication flow.

Case No. IPR2017-1409
U.S. Patent No. 8,131,880

INTEL AND TRADEMARK OFFICE

INTEL TRIAL AND APPEAL BOARD

INTEL CORPORATION, and CAVIUM, INC.,
Petitioners,
v.
ALACRITECH INC.,
Patent Owner

Case IPR2017-1409,
U.S. Patent 8,131,880

EXHIBIT 2026
OF KEVIN ALMEROOTH, PH.D.

In Case IPR2017-01736, has been joined as a

Alacritex Exhibit 2026

Dr. Almeroth Decl., EX2305 at ¶¶ 28-29

-1409, -1410 PO Demonstrative, 74

INTEL CORPORATION

v.

ALACRITECH, INC.

Case Nos.: IPR2017-01393 and IPR2017-01714

U.S. Patent No. 9,055,104 B2

Patent Owner's Demonstratives

Hearing: September 13, 2018

Overview of the '104 Patent

(12) **United States Patent**
 Philbrick et al. (10) Patent No.: **US 9,055,104 B2**
 (45) Date of Patent: ***Jun. 9, 2015**

(54) **FREEDING TRANSMIT MEMORY ON A NETWORK INTERFACE DEVICE PRIOR TO** (56) **References Cited**

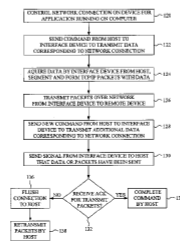
FREEDING TRANSMIT MEMORY ON A NETWORK INTERFACE DEVICE PRIOR TO RECEIVING AN ACKNOWLEDGMENT THAT TRANSMIT DATA HAS BEEN RECEIVED BY A REMOTE DEVICE

Inventors: Clive M. Philbrick, San Jose, CA (US);
 Peter K. Craft, San Francisco, CA (US)

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

(60) Provisional application No. 60/374,788, filed on Apr. 22, 2002.
 (51) Int. Cl. G06F 15/16 (2006.01) H04L 29/06 (2006.01)
 (52) U.S. Cl. CPC H04L 69/16 (2013.01); H04L 69/161 (2013.01); H04L 69/163 (2013.01); H04L 69/10 (2013.01)
 (58) **Field of Classification Search**
 CPC H04L 69/10; H04L 69/16
 USPC 709/230
 See application file for complete search history.

24 Claims, 4 Drawing Sheets



Ex. 1001 (“‘104 Patent”)

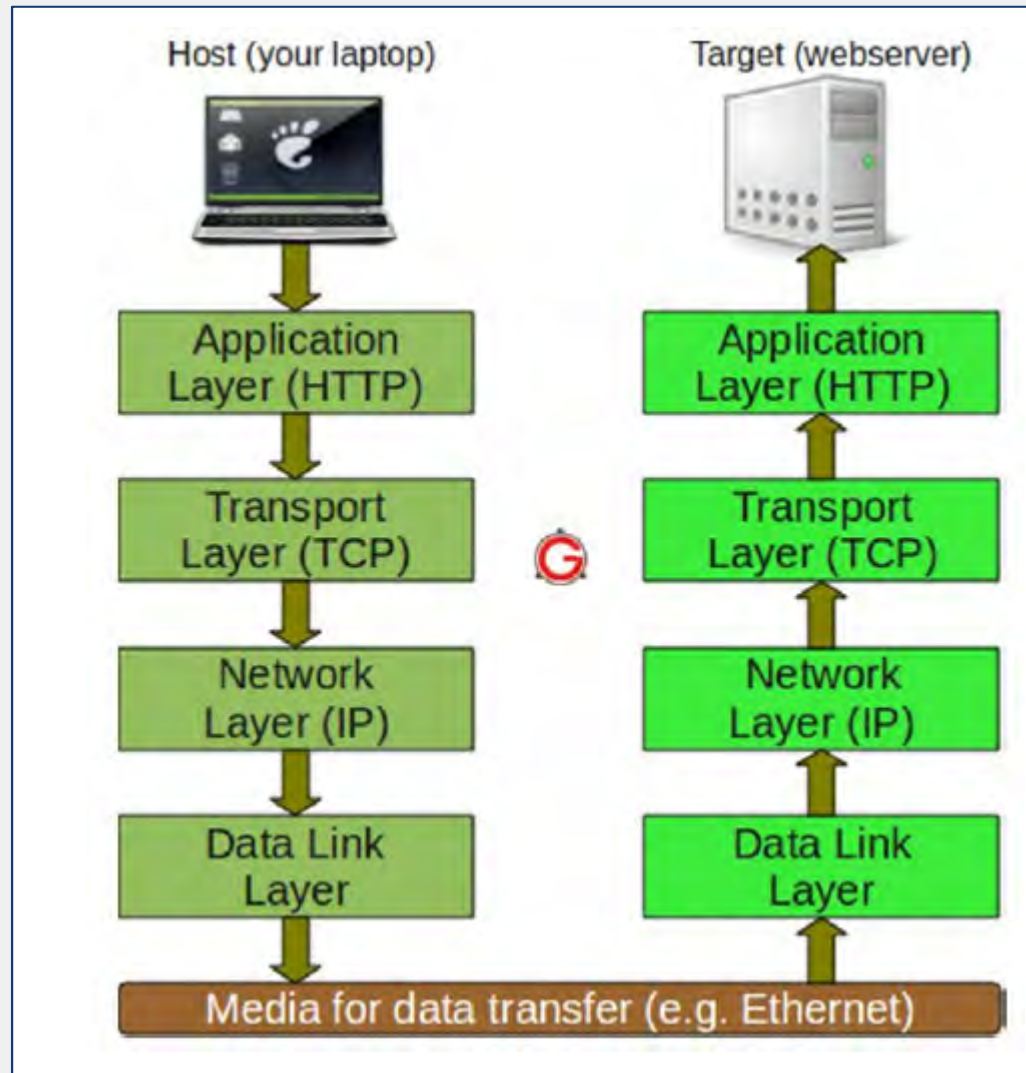
(57)

ABSTRACT

A transmit offload engine (TOE) such as an intelligent network interface device (INIC), video controller or host bus adapter (HBA) that can communicate data over transport protocols such as Transport Control Protocol (TCP) for a host. Such a device can send and receive data for the host to and from a remote host, over a TCP connection maintained by the device. For sending data, the device can indicate to the host that data has been transmitted from the device to a network, prior to receiving, by the device from the network, an acknowledgement (ACK) for all the data, accelerating data transmission. The greatest sequence number for which all

'104 Patent, Abstract

Message Processing Under TCP/IP



Ex. 2043

Message Processing Under TCP/IP

One way that TCP guarantees delivery of data is through the use of acknowledgments (ACKs) and the sequenced delivery of the data. That is, after data has been sent in sequen-

'104 patent 2:10-12

been received. As shown in the prior art diagram of FIG. 1, to transmit data corresponding to a TCP connection from a local host having an attached INIC to a remote host over a network, the local host first sends 20 to the INIC a command to transmit the data. The INIC then 22 acquires the data, divides it into segments and adds TCP and IP headers to each data segment to create a TCP/IP packet corresponding to each segment. Next, the INIC transmits 24 the resulting packets onto the network. After the remote host has received and validated the

'104 patent 2:15-23

Message Processing Under TCP/IP

network. After the remote host has received and validated the packets, the remote host sends ACKs back to the local host indicating how much of the data has been successfully received.

Upon receiving an ACK 26 for all the transmitted data, the INIC sends a command complete 28 to the local host indicating that the transmit command has been completed by the transport function of the INIC, and an upper layer such as a session layer of the host is informed that its request to transmit data has been completed. For the case in which an ACK is not

'104 patent 2:23-32

Message Processing Using The Claimed Invention

UNITED STATES I

BEFORE THE PA

67. In contrast to the conventional approach, the '104 patent describes systems and methods by which a network interface device, after sending all the packets to the remote host over the network, sends a command response to the host computer indicating the data has been sent prior to receiving an ACK indicating all the sent data has been received by the remote host. (*Id.* at 3:42-51.) The command

Ex. 2026 at 28

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Overview of the '104 Patent

Upon receiving the signal that the data was sent, the host can send a yet another command to the interface device to transmit additional data. Relieving the interface device from the duty of maintaining the command until all the data for the command has been ACKed frees memory space on the interface device for storing another command, allowing the interface device to transmit more data. This is particularly useful

'104 patent 4:1-7

Overview of the '104 Patent

In one embodiment, the interface device caches thirty-two of the most active TCP connections in SRAM, while about four thousand TCP connections are maintained in DRAM. SRAM memory may be relatively expensive especially in terms of on-chip real estate, and therefore SRAM memory space may be relatively scarce. For each of the thirty-two active TCP connections in this embodiment, pointers to (also known as indications of) up to three transmit commands are stored: commands that have been sent, commands that are being sent, and commands that are to be sent. Once these three pointers or indications have been stored, that connection can not transmit any more data in this embodiment. Particularly for the situation in which a number of transmit commands are desired to be sent in a rapid sequence for a connection, waiting for an ACK to be returned corresponding to one of the commands can stall the transmission of data. This embodiment avoids that delay by freeing the SRAM that stores the command pointers or indications once the data has been sent and typically prior to receiving an ACK for all that data, while sending a signal to the host that the data has been sent.

'104 patent 4:24-43

Challenged Claims: 1, 6, 9, 12, 15, and 22

1. A method for communication involving a computer, a network, and a network interface device of the computer, the network interface device being coupled to the network, the method comprising:

receiving, by the network interface device from the computer, a command to transmit application data from the computer to the network;

sending, by the network interface device to the network, data corresponding to the command, including prepending a transport layer header to at least some of the data;

sending, by the network interface device to the computer, a response to the command indicating that the data has been sent from the network interface device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data corresponding to the command has been received; and

maintaining, by the network interface device, a Transport Control Protocol (TCP) connection that the command, the data and the ACK correspond to.

'104 Patent, Claim 1

-01393 PO Demonstrative, 9

Challenged Claims: 1, 6, 9, 12, 15, and 22

12. A method for communication involving a computer, a network, and a network interface device of the computer, the network interface device being coupled to the network, the method comprising:

receiving, by the network interface device from the computer, a pointer to a command to transmit data from the computer to the network;

sending, by the network interface device to the network, data corresponding to the command;

sending, by the network interface device to the computer, a response to the command indicating that the data has been sent from the device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data has been received; and

maintaining, by the network interface device, a Transport Control Protocol (TCP) connection that the command, the data and the ACK correspond to.

'104 Patent, Claim 12

-01393 PO Demonstrative, 10

Challenged Claims: 1, 6, 9, 12, 15, and 22

22. A system for communication involving a computer, a network, and a network interface device of the computer, the network interface device being coupled to the network, the system comprising:

means for receiving, by the network interface device from the computer, a command to transmit data from the computer to the network;

means for sending, by the network interface device to the network, data corresponding to the command, including means for prepending a transport layer header to at least some of the data; and

means for sending, by the network interface device to the computer, an indication that the data has been sent from the network interface device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that the data has been received.

'104 Patent, Claim 22

-01393 PO Demonstrative, 11

Overview of the Prior Art

Challenged Claims	103(a) References
1, 6, 9, 12, 15, and 22	Connery in view of knowledge of a POSA
1, 6, 9, 12, 15	Connery in view of Boucher

Connery Was Already Considered by the PTO

U.S. Department of Commerce, Patent and Trademark Office						
INFORMATION DISCLOSURE STATEMENT BY APPLICANT						
FREEING TRANSMIT MEMORY ON A NETWORK INTERFACE PRIOR TO RECEIVING AN ACKNOWLEDGMENT THAT TRANSMISSION HAS BEEN RECEIVED BY A REMOTE DEVICE						
/K.M./	73	5,892,903	April 6, 1999			
	74	5,898,713	April 27, 1999			
	75	5,913,028	June 15, 1999	Wang, et al.	395	200.33
	76	5,920,566	July 6, 1999	Ariel Hendel et al	370	401
	77	5,930,830	July 27, 1999	Mendelson et al.	711	171
	78	5,931,918	August 3, 1999	Row et al.	709	300
	79	5,935,205	August 10, 1999	Murayama et al.	709	216
	80	5,937,169	August 10, 1999	Connery et al.	395	200.8
	81					
	82					
	83					
	84					
	85					
	86					
	87					
	88					
	89					
	90					
	91	6,016,513	January 18, 2000	Glen H. Lowe	709	250
	92	6,021,446	February 1, 2000	Gentry et al.	709	303
	93	6,021,507	February 1, 2000	Shawfu Chen	714	2
	94	6,026,452	February 15, 2000	William Michael Pitts	710	56
	95	6,034,963	March 7, 2000	Minami et al.	370	401
	96	6,038,562	March 14, 2000			
	97	6,041,058	March 21, 2000			
Examiner	/Kevin Mai/		Date Considered			
*EXAMINER: Initial if reference considered, whether or not citation in conformance with 37 CFR 1.902(b)(2) is required.						
Examiner			/Kevin Mai/		Date Considered	
					02/07/2011	

Ex. 1002 ("Prosecution File History") at .167

Prosecution File History at .167

Prosecution File History at .167

Overview of Connery

United States Patent [19] [11] **Patent Number**
Connery et al. [45] **Date of Patent**

[54] **OFFLOAD OF TCP SEGMENTATION TO A SMART ADAPTER** "Internet Protocol: DARPA Specification", rfc 791, prepared Sep. 1981, printed from hio-state.edu/hbin/rfc", 42 p

[75] Inventors: **Glenn William Connery; W. Paul Sherer**, both of Sunnyvale; **Gary Jaszewski**, Los Gatos; **James S. Binder**, San Jose, all of Calif. Gilbert, H., "Introduction to printed from web site "http://tcpip.htm", 5 pages.

[73] Assignee: **3Com Corporation**, Santa Clara, Calif.

[21] Appl. No.: **08/960,238** *Primary Examiner*—Robert B. Attorney, Agent, or Firm—Sonsini, Goodrich & Rosati

[22] Filed: **Oct. 29, 1997**

[51] **Int. Cl.** **G06F 13/38** [57] **ABSTRACT**

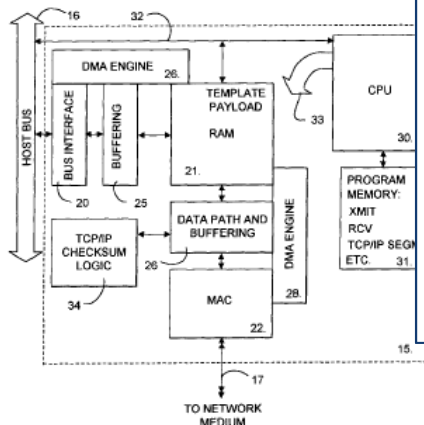
[52] **U.S. Cl.** **395/200.8**

[58] **Field of Search** 364/DIG. 1, DIG. 2; 395/200.3, 200.36, 200.37, 200.48, 200.5, 200.53, 200.55, 200.6, 200.66, 200.8

[56] **References Cited**
U.S. PATENT DOCUMENTS
 5,321,819 6/1994 Szczepanek 395/200.8
 5,727,149 3/1998 Hirata et al. 395/200.8

OTHER PUBLICATIONS
 Postel, J., "The TCP Maximum Segment Size and Related Topics", rfc879, dated Nov. 1983, printed from web site "http:// www.cis.ohio-state.edu/hbin/rfc", 10 pages.
 Clark, D., "Window and Acknowledgement Strategy in TCP", rfc813, dated Jul. 1982, printed from web site "http:// www.cis.ohio-state.edu/hbin/rfc", 18 pages.
 "Transmission Control Protocol: DARPA Internet Program Protocol Specification", rfc793, prepared by Univ. of Southern Calif., dated Sep. 1981, printed from web site "http:// www.cis.ohio-state.edu/hbin/rfc", 77 pages.

83 Claims, 6 D



[57]

ABSTRACT

A method is provided for sending data from a data source executing a network protocol such as the TCP/IP protocol stack, which includes a process for generating headers for packets according to the network protocol. The method includes sending such data on a network through a smart network interface. The network protocol defines a datagram in the data source, including generating a header template and supplying a data payload. **The datagram is supplied to the network interface. At the network interface, a plurality of packets of data are generated from the datagram. The plurality of packets include respective headers, such as TCP/IP headers, based on the header template, and include respective segments of the data payload. The network interface supports packets having a pre-specified length, and the data payload is greater than the pre-specified length, such as two to forty times larger or more. Thus, the higher layer processing specifies a very large datagram, which is automatically segmented at the network interface layer, instead of at the TCP layer.**

Connery, Abstract

-01393 PO Demonstrative, 15

Overview of Connery

United States Patent [19] [11] **Patent Number:** 5,937,169
Connery et al. [45] **Date of Patent:** Aug. 10, 1999

[54] **OFFLOAD OF TCP SEGMENTATION TO A SMART ADAPTER** "Internet Protocol: DARPA Internet Program Protocol Specification", rfc 791, prepared by Univ. of Southern Calif., dated Sep. 1981, printed from web site "http://www.cis.ohio-state.edu/hbin/rfc", 42 pages.
 [75] **Inventors:** Glenn William Connery; W. Paul Sherer, both of Sunnyvale; Gary Jaszewski, Los Gatos; James S. Binder, San Jose, all of Calif.
 Gilbert, H., "Introduction to TCP/IP", dated Feb. 2, 1995, printed from web site "http://pelt.cis.yale.edu/pelt/comm/tcpip.htm", 5 pages.
 [73] **Assignee:** 3Com Corporation, Santa Clara, Calif.

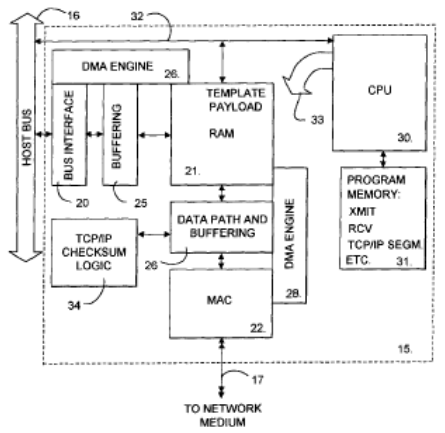
[21] Appl. No.: 08/960,238
 [22] Filed: Oct. 29, 1997
 [51] Int. Cl.⁵
 [52] U.S. Cl.
 [58] Field of Search 364
 395/200.3, 200.36, 200.3
 200.53, 200.55, 200

[56] **References Cited**
 U.S. PATENT DOCUMENTS
 5,321,819 6/1994 Szczepanek
 5,727,149 3/1998 Hirata et al.

OTHER PUBLICATIONS
 Postel, J., "The TCP Maximum Segment Topics", rfc879, dated Nov. 1983, printed from "http://www.cis.ohio-state.edu/hbin/rfc",
 Clark, D., "Window and Acknowledgment TCP", rfc813, dated Jul. 1982, printed from "http://www.cis.ohio-state.edu/hbin/rfc", 18 pages.
 "Transmission Control Protocol: DARPA Internet Program Protocol Specification", rfc793, prepared by Univ. of Southern Calif., dated Sep. 1981, printed from web site "http://www.cis.ohio-state.edu/hbin/rfc", 77 pages.

plurality packets. The packets are then sent to the destination. Finally, an acknowledgment is received from the destination that the plurality of packets was successfully sent according to the network protocol.

83 Claims, 6 Drawing Sheets



Connery at 3:59-62

Overview of Boucher

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY

(51) International Patent Classification ⁶ : G06F 13/00	A1	(11) International Publication Number: (11) International Publication Number: WO 00/13091
(21) International Application Number: PCT/US98/24943	(22) International Filing Date: 20 November 1998 (20.11.98)	(43) International Publication Date:
(30) Priority Data: 09/141,713 28 August 1998 (28.08.98) US	(81) Designated States: AU, CA, IL, JP, KR, MX, SG, Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).	Published With international search report.
(71) Applicant: ALACRITECH CORPORATION [US/US]; Suite 302, 888 N. First Street, San Jose, CA 95112 (US).	(71)(72) Applicants and Inventors: BOUCHER, Laurence, B. [US/US]; 20605 Montolvo Drive, Saratoga, CA 95070 (US). BLIGHTMAN, Stephen, E. J. [GB/US]; 3733 Arlen Court, San Jose, CA 95132 (US). CRAFT, Peter, K. [US/US]; 156 Henry Street, San Francisco, CA 94114 (US). HIGGEN, David, A. [GB/US]; 17880 Los Alamos Drive, Saratoga, CA 95070 (US). PHILBRICK, Clive, M. [AU/US]; 1170 Roycott Way, San Jose, CA 95125 (US). STARR, Daryl [US/US]; 446 Folsom Court, Milpitas, CA 95035 (US).	
(74) Agents: LAUER, Mark, A. et al.; Suite 280, 7041 Koll Center Parkway, Pleasanton, CA 94566 (US).		
(54) Title: INTELLIGENT NETWORK INTERFACE DEVICE AND SYSTEM FOR ACCELERATING COMMUNICATION		

(57) Abstract

An intelligent network interface card or communication processing device provides a fast-path (159) that avoids protocol processing and offloading time-intensive processing tasks from the host CPU (28). The device provides assistance such as validation of packets that do not fit fast-path criteria, with the device providing assistance such as validation of packets selected for either fast-path or slow-path (158) processing. A context (50) for a connection is determined and stored in the device to move data free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device 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 (482, 484, 486) devoted to receive, transmit and utility processing, providing full duplex communication for four Fast Ethernet nodes.

(11) International Publication Number:

WO 00/13091

Ex. 1049 ("Boucher") at 1

(71)(72) Applicants and Inventors: BOUCHER, Laurence, B. [US/US]; 20605 Montolvo Drive, Saratoga, CA 95070 (US). BLIGHTMAN, Stephen, E. J. [GB/US]; 3733 Arlen Court, San Jose, CA 95132 (US). CRAFT, Peter, K. [US/US]; 156 Henry Street, San Francisco, CA 94114 (US). HIGGEN, David, A. [GB/US]; 17880 Los Alamos Drive, Saratoga, CA 95070 (US). PHILBRICK, Clive, M. [AU/US]; 1170 Roycott Way, San Jose, CA 95125 (US). STARR, Daryl [US/US]; 446 Folsom Court, Milpitas, CA 95035 (US).

Boucher at 1

(54) Title: INTELLIGENT NETWORK INTERFACE DEVICE AND SYSTEM FOR ACCELERATING COMMUNICATION

Boucher at 1

Boucher Was Already Considered by the PTO

U.S. Department of Commerce, Patent and Trademark Office				Application No.: Unknown			
INFORMATION DISCLOSURE STATEMENT BY APPLICANT							
FREEING TRANSMIT MEMORY ON A NETWORK INTER PRIOR TO RECEIVING AN ACKNOWLEDGMENT THAT T HAS BEEN RECEIVED BY A REMOTE DEVIC							
/K.M./	195	2001/0004354	January 10, 2001				
	196	2001/0013059	August 9, 2001				
	197	2001/0014892	August 16, 2001				
	198	2001/0014954	August 16, 2001	Purcell et al.	714	4	
	199	2001/0025315	January 10, 2001	Jolitz	709	231	
	200	2001/0048681	December 6, 2001	Bilic et al.	370	389	
	201	2001/0053148	December 20, 2001	Bilic et al.	370	389	
	202	2002/0073223	June 13, 2002	B. Scott Darnell et al	709	232	
	203	2002/0112175	August 15, 2002	Makofka et al	713	200	
	204	200					
	205	200					
	206	200					
	207	200					
	208	200					
	209	200					
	210	200					
	211	200					
		215	December 16, 1999	WO 99/65219	PCT/US/99/13184		
		216	March 9, 2000	WO 00/13091	PCT/US98/24943		
Foreign Patent Documents							
Examiner Initial	Document Number	Date	Country	Class	Subclass	Tr a	
/K.M./	212 WO 98/19412	May 7, 1998	PCT/US97/17257				
	213 WO 98/50852	November 12, 1998	PCT/US98/08719				
	214 WO 99/04343	January 28, 1999	PCT/US98/14729				
	215 WO 99/65219	December 16, 1999					
	216 WO 00/13091	March 9, 2000					
Examiner	/Kevin Mai/	Date Considered					
*EXAMINER: Initial if reference considered, whether or not citation in conformance with 37 CFR 1.97(b)(2). ALL REFERENCES CONSIDERED			Examiner	/Kevin Mai/	Date Considered	02/07/2011	

Prosecution File History at .172

Prosecution File History at .172

Prosecution File History at .172

Overview of Boucher

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : G06F 13/00</p> <p>(21) International Application Number: PCT/US98/24943</p> <p>(22) International Filing Date: 20 November 1998 (20.11.98)</p> <p>(30) Priority Data: 09/141,713 28 August 1998 (28.08.98) US</p> <p>(71) Applicant: ALACRITECH CORPORATION [US/US]; Suite 302, 888 N. First Street, San Jose, CA 95112 (US).</p> <p>(71)(72) Applicants and Inventors: BOUCHER, Laurence, B. [US/US]; 20605 Montolvo Drive, Saratoga, CA 95070 (US). BLIGHTMAN, Stephen, E. J. [GB/US]; 3733 Arlen Court, San Jose, CA 95132 (US). CRAFT, Peter, K. [US/US]; 156 Henry Street, San Francisco, CA 94114 (US). HIGGEN, David, A. [GB/US]; 17880 Los Alamos Drive, Saratoga, CA 95070 (US). PHILBRICK, Clive, M. [AU/US]; 1170 Roycott Way, San Jose, CA 95125 (US). STARR, Daryl [US/US]; 446 Folsom Court, Milpitas, CA 95035 (US).</p> <p>(74) Agents: LAUER, Mark, A. et al.; Suite 280, 7041 Koll Center Parkway, Pleasanton, CA 94566 (US).</p>	<p>(11) International Publication Number: WO 00/13091</p> <p>(43) International Publication Date: 9 March 2000 (09.03.00)</p> <p>(81) Designated States: AU, CA, IL, JP, KR, MX, SG, Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</p> <p>Published <i>With international search report</i></p>
---	--

(54) Title: INTELLIGENT NETWORK INTERFACE DEVICE AND

(57) Abstract

An intelligent network interface card or communication processing device (30) works with a host computer (20) for data communication. The device provides a fast-path (159) that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU (28). The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path (158) processing. A context (50) for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device 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 (482, 484, 486) devoted to receive, transmit and utility processing, providing full duplex communication for four Fast Ethernet nodes.

“The device provides a fast-path (159) that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-sensitive processing tasks from the host CPU (28). The host retains a fallback processing capability for messages that do not fit fast-path criteria”

Boucher, Abstract

All Independent Claims Involve Sending a Response Prior to Receiving an ACK that All Data Has Been Transmitted

sending, by the network interface device to the computer, a response to the command indicating that the data has been sent from the network interface device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data corresponding to the command has been received; and

'104 Patent, Claim 1

sending, by the network interface device to the computer, a response to the command indicating that the data has been sent from the device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data has been received; and

'104 Patent, Claim 12

means for sending, by the network interface device to the computer, an indication that the data has been sent from the network interface device to the network, prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that the data has been received.

'104 Patent, Claim 22

-01393 PO Demonstrative, 20

Petition Relies on Unasserted Prior Art, Including Conclusory, Unproven Knowledge of a POSA

Given Connery's teaching of one interrupt per large packet, Dr. Horst explains how a POSA designing the system would naturally design the system with an interrupt to signal the end of the large packet transmission, because (as Connery suggests) interrupts were commonly used in the prior art to signal the end of a packet transmission, and Connery teaches that multiple such interrupts would have been reduced to one interrupt when segmentation is handled by the network interface (instead of the host handling the segmentation). *Id.*

Given Connery's teachings and goals, even if there are other potential interpretations of the single interrupt disclosed in Connery, Dr. Horst explains how a POSA would have chosen the transmit completion interrupt as one of a small number of choices that would have been obvious to try, and that its design would have been well within the capabilities of a POSA. *Id.* (citing Ex.1044, U.S. Patent No. 5,434,872 ("Petersen") at Abstract, 3:65-4:17, 7:55-56, 9:33-36, Fig. 4, 7:60-63). That is because, at least in part, the single completion interrupt per large packet helps achieve Connery's goal of reducing host CPU utilization. Ex.1003, Horst Decl., A-17-18. Further, interrupts for transmit completion (which are

Pet. at 59

-01393 PO Demonstrative, 21

Petition Relies on Unasserted Prior Art, Including Conclusory, Unproven Knowledge of a POSA

⁵ More specifically, Dr. Horst explains that, in the timeframe of the Connery prior art reference, and as corroborated by the teachings of the Petersen patent, a POSA would have been well aware of the practice of a network interface generating an interrupt when data transmission is complete in order to notify the host of such

Pet. at 60

Second, Dr. Horst explains how it would have been known (or at least obvious to a POSA in view of Connery) that the “response” sent from the network interface to the host (*i.e.*, the response indicating that data was sent to the network, *see* [1.4] *supra*) would be sent “prior to receiving, by the network interface device from the network, an acknowledgement (ACK) that all the data corresponding to the command has been received,” for the following reasons. Ex.1003, Horst Decl.,

Pet. at 62

-01393 PO Demonstrative, 22

Connery Fails to Disclose Sending a Response Prior to Receiving an ACK

UNITED STATES PATENT AND TRADEMARK
BEFORE THE PATENT TRIAL AND APPEAL

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

at 7:60-63.) A POSITA would have recognized that the network interface device would interrupt the host CPU after receiving an ACK, which is the exact opposite of the claimed limitation of sending a response to the computer "prior to receiving" an ACK. Further, the function of an ACK is to acknowledge that data has actually been received at the destination to guarantee reliability of transmission. As such, a POSITA would infer that the network interface device in Connery would rely on the ACK to ensure that the data has been delivered so interrupt the host CPU after receiving the ACK. At a minimum, Connery is completely silent regarding interrupting the host CPU prior to receiving an ACK.

Ex. 2026 at 39

Longer Latency in ACK Path Does Not Render the Claimed Invention Obvious

UNITED STATES PATENT AND TRADEMA

BEFORE THE PATENT TRIAL AND APPE

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

96. The fact that the ACK path has a longer latency than the interrupt path does not render obvious to interrupt the host before receiving the ACK, especially given that the function of the ACK is precisely to guarantee reliability of receipt at the destination, which the interrupt to the host is indicating. Indeed, interrupting the host before receiving an ACK is counterintuitive given the function of the ACK is to acknowledge successful receipt of the transmitted data. The '104 patent

Ex. 2026 at 40

Petitioner's Arguments Are Based on Hindsight

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

97. Indeed, in my opinion, the Petition's argument that "[b]y generating the interrupt quickly, the next packet can be sent without waiting for the previous ACK, in order to keep the network link busy and reduce the transfer time for large blocks of data" is based on improper hindsight, which I understand to be improper in determining obviousness. It is based on hindsight because it is based on knowledge that sending a response "prior to receiving . . . an acknowledgement (ACK)" could be implemented as shown by Patent Owner's invention. (Petition at 63.) That this could be achieved without undesirable effects to reliability and without breaking the established networking protocol (such as TCP/IP) would not have been known to a POSITA at the time of the invention.

Ex. 2026 at 41

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Petersen Is Not Part of an Instituted Ground and Fails to Support Dr. Horst's Opinion

If enabled, and when the transmit complete condition is satisfied, a **transmit complete interrupt** is generated for handling by the interrupt controller **60** in the transmit DMA module **67**.

Ex. 1044 ("Peterson") at 9:33-36

⁵ More specifically, Dr. Horst explains that, in the timeframe of the Connery prior art reference, and as **corroborated by the teachings of the Petersen patent**, a POSA would have been well aware of the practice of a network interface generating an interrupt when data transmission is complete in order to notify the host of such transmission, and further that a POSA would have been readily able to apply such a practice to Connery's system, as explained *supra*. Ex.1003, Horst Decl., A-16-18.

Pet. at 60

Petersen Is Not Part of an Instituted Ground and Fails to Support Dr. Horst's Opinion



Robert Horst
Petitioner's Expert

20 Q. Okay. So the question is -- the
21 question is: What would trigger -- in Petersen,
22 what would trigger the generation of the transmit
23 complete interrupt?

* * *

6 A. This reference in column 9, it says
7 that, "If enabled, and when the transmit complete
8 condition is satisfied, a transmit complete
9 interrupt is generated for handling by the
10 interrupt controller in the transmit DMA module."

11 So that indicates to me that the
12 interrupt signals that the full segment has been
13 transferred to the controller and at that point
14 the interrupt is generated.

15 Q. What do you mean by the controller?

16 A. The network interface that's the
17 subject of this application.

Ex. 2029 (Horst Dep. Tr.) at 112-13

-01393 PO Demonstrative, 27

Petersen Is Not Part of an Instituted Ground and Fails to Support Dr. Horst's Opinion

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

100. However, the “transmit complete interrupt” disclosed in Petersen does not refer to an interrupt generated when a network interface device has completed transmitting data. Instead, the “transmit complete interrupt” refers to when the data has been transmitted from the host CPU to the network interface device.

101. In Petersen, the transmit DMA module refers to the DMA module for transmitting data between the host computer and the network interface device. In this context of the transmit complete interrupt being generated “for handling by the interrupt controller in the transmit DMA module,” the “transmit complete interrupt” refers to an interrupt that all the data has been transmitted from the host computer and received by the network interface card—in sharp contrast to data being transmitted by network interface card.

Ex. 2026 at 42

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Sending a Response Prior to Receiving an ACK Would Not Have Been Obvious

UNITED STATES PATENT AND TRADEMARK

BEFORE THE PATENT TRIAL AND APPEALS

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT
DECLARATION OF KEVIN ALMEROOTH

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been identified as the petitioner in this proceeding.

Alacritech Exhibit 2026

ACKs in the field. As discussed above, Petitioner argues that Connery in view of the knowledge of a POSA would render it obvious for a network interface device to send a response “prior to receiving” an ACK. However, **this is incorrect because this would go against the conventional role of ACKs in the field.** (Petition at 56-60.) **In my experience, ACKs play an important role in guaranteeing the reliability of transmission.** Conventionally, a network interface device would be designed to send a transmit complete interrupt to the host computer once it receives data from the host computer. Thereafter, the network interface device would transmit the data in one or more packets to the destination, and **wait for ACKs corresponding to the packets coming from the destination.** Once the ACKs are received, the network interface device would notify the host computer that all the data has been transmitted from the network interface device and received at the destination. Indeed, a network interface device that notifies the host computer that

Ex. 2026 at 43

-01393 PO Demonstrative, 29

Sending a Response Prior to Receiving an ACK Would Not Have Been Obvious

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

destination. Indeed, a network interface device that notifies the host computer that data has been transmitted prior to receiving ACKs for the data would have been counterintuitive and not obvious to a POSITA. A POSITA would not have been motivated to modify this mechanism given that modifications can lead to significant problems in reliability, speed, and efficiency of the network interface device.

Ex. 2026 at 43

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

No Motivation to Modify Connery

packets (algorithmic).” (*Id.* at 7:60-63.) In my opinion, **Connery does not disclose any motivation to modify the interrupts** on the host CPU to occur before the network interface receives an ACK that all the data has been received at the destination.

Ex. 2026 at 45

UNITED STATES PATENT

BEFORE THE PATENT TRI

INTEL C
CAVIUM

Petition

ALACRIT

Patent

Case IPR2017-01393¹
U.S. Patent 9,055,104

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Alacritech Exhibit 2026

Secondary Considerations: Long-Felt Need

1. Introduction

As data transmission speeds have increased dramatically in recent years, the processing of protocols has become one of the major bottlenecks in data communications. Current experimental networks provide a bandwidth in the Gb/s range. New multimedia applications require that networks guarantee the quality

Ex. 2031 at 1 (1993 IBM Research Division ACM SIGCOMM article)

1 Introduction

Many researchers have observed that the performance of remote applications have not kept pace with modern communication network speeds. Part of this imbalance is attributed to the protocols used by the remote applications, namely, UDP/IP and TCP/IP. It is now widely believed that the problems with these protocols are not inherent in the protocols themselves but in their particular implementations [Dalt93, Whet95]. The following factors are cited as contributors to the poor performance of UDP/IP and TCP/IP:

Ex. 2032 at 1 (1995 Univ. of Berkeley paper)

Secondary Considerations: Long-Felt Need

Abstract

The communication speed in wide-, metropolitan-, and local-area networking has increased over the past decade from Kbps lines to Gbps lines; i.e., six orders of magnitude, while the processing speed of commercial CPUs that can be employed as communications processor has changed only two to three orders of magnitude. This discrepancy in speed translates to “bottlenecks” in the communications process, because the software that supports some of the high-level functionality of the communication process is now several order of magnitude slower than the transmission media. Moreover, the overhead introduced by the operating system (OS) on the communication pro-

Ex. 2033 at 1 (1990 IEEE article from AT&T Bell Labs)

Abstract

Server network performance is increasingly dominated by poorly scaling operations such as I/O bus crossings, cache misses and interrupts. Their overhead prevents performance from scaling even with increased CPU, link or I/O bus bandwidths. These operations can be reduced by redesigning the host/adaptor interface to exploit additional processing on the adapter. Offloading processing to the adapter is beneficial not only because it allows more cycles to be applied but also of the changes it enables in the host/adaptor interface. As opposed to other approaches such as RDMA, TCP offload provides benefits without requiring changes to either the transport protocol or API.

Ex. 2034 at 1 (2005 USENIX Conference paper from IBM)

Secondary Considerations: Long-Felt Need

111. The nexus between the long-felt need and the claimed invention is clear and direct. **The aforementioned bottlenecks are all solved by the accelerated network processing technologies recited in the challenged claims.** In addition, the technologies recited in the challenged claims alleviated the host CPU from receiving notice only after a network interface device receives an ACK in such a way as to permit faster network throughput and transmission/reception speeds—the precise long-felt but unresolved need in the industry outlined above.

Ex. 2026 at 47-48

UNITED STATES PATENT

BEFORE THE PATENT TRI

INTEL C
CAVIUM

Petition

ALACRITECH

Patent Owner

Case IPR2017-01393¹
U.S. Patent 9,055,104

PATENT OWNER'S EXHIBIT 2026

DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

Secondary Considerations: Industry Praise

Our measurement shows that an Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed. Its accumulated host processor utilization, while lower than native NT's, is higher than that with our offload implementation. Its performance degrades when messages are smaller than 2k bytes because it has no means of aggregating out-going messages (i.e. no Nagel Algorithm).

Ex. 2039 at 4 (2001 HP Labs research paper)

Secondary Considerations: Industry Praise

“The ANX 1500 is an evolutionary advancement of Alacritech’s long standing leadership in protocol acceleration, which is now applied not just to protocols, but to the optimization of all storage system interactions,” said Jeff Boles, a technology analyst with Taneja Group. “The ANX 1500 substantially augments the performance of existing NAS investments from the best place possible – from right in the customer’s Ethernet network – without reconfiguration or changes in management for the existing infrastructure. We think Alacritech is setting the stage for a next generation of solutions that will accelerate storage from outside the storage array, and more cost effectively share an investment in performance across many storage systems and clients. I’ve talked to early-stage customers using the product, and they believe it’s game-changing.”

Ex. 2040 at 3 (Shoreline Ventures Press Release)

-01393 PO Demonstrative, 36

Secondary Considerations: Industry Praise

115. The industry universally praised **commercial embodiments of the features described in the challenged claims**. HP found that the “Alacritech NIC is able to sustain network bandwidth comparable to that of Native NT for large messages, which is close to wire-speed” and “achiev[e] lower processor utilization than native NT’s TCP/IP protocol stack for transmission of large enough messages.” (Ex. 2039 at ¶ 4.) HP found that the test performance of the Alacritech

Ex. 2026 at 49

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORPORATION
CAVIUM,

Petitioner

v.

ALACRITECH CORPORATION,
Respondent

Patent Owner.

Case IPR2017-01392¹
U.S. Patent 7,337,241

**CORRECTED PATENT OWNER’S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.**

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-00001985280.6

Alacritech Exhibit 2026

Secondary Considerations: Failure of Others

To this day, TCP offload has never firmly caught on in the commercial world (except sometimes as a stopgap to add TCP support to immature systems [16]), and has been scorned by the academic community and Internet purists. This paper starts by analyzing why TCP offload has repeatedly failed.

Ex. 2041 at 2 (2003 HP Labs paper presented at HotOS IX conference)

TCP offload has been unsuccessful in the past for two kinds of reasons: fundamental performance issues, and difficulties resulting from the complexities of deploying TCP offload in practice.

Id. at 2

Secondary Considerations: Failure of Others

offload were mismatched to the applications in question.” *Id.* The TCP offload described above is a form of network processing offload that is described by the challenged claims, and this failure of others therefore has a direct nexus to the claimed inventions.

Ex. 2026 at 50

UNITED STATES PATENT AND TRADEMARK OFFICE
 BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORP. and
 CAVIUM, INC.,
Petitioners,

v.

ALACRITECH INC.,
Patent Owner.

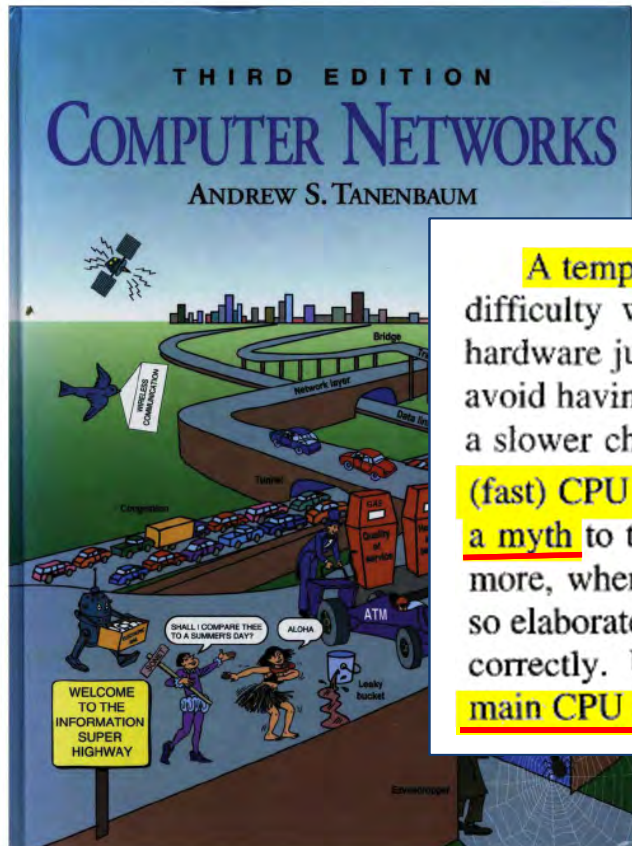
Case IPR2017-01392¹
 U.S. Patent 7,337,241

CORRECTED PATENT OWNER'S EXHIBIT 2026
DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding.

06073-000019850280.6 Alacritech Exhibit 2026

Secondary Considerations: Skepticism



INTEL Ex.1006.001

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.

Ex. 1006 at 588-589

Secondary Considerations: Skepticism

IP Storage and the CPU Consumption Myth

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

Abstract

This paper addresses a key issue that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

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 paper 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.

2.

The
accel
CPU
been
comm
unme
Ex
date
many
I/O p
Howe
of arc
rapid
A
I/O)
proce
serve
from
starte
task,
as the
some
Some
or w
and s
usual
and e
envir
costly
for if
the p
kills
gener
the c
Si
accel
of en
ever

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

conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

Ex. 2300 at 194 (2001 paper by Petitioner's expert, Dr. Horst)

Secondary Considerations: Skepticism

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 that arises when attaching storage devices directly to IP networks: the perceived need for hardware acceleration of the TCP/IP networking stack. While many implicitly assume that acceleration is required, the evidence shows that this conclusion is not well founded. In the past, network accelerators have had mixed success, and the current economic justification for hardware acceleration is poor given the low cost of host CPU cycles. The I/O load for many applications is dominated by disk latency, not transfer rate, and hardware protocol accelerators have little effect on the I/O performance in these environments. Application benchmarks were run on an IP storage subsystem to measure performance and CPU utilization on Email, database, file serving, and backup applications. The results show that good performance can be obtained without protocol acceleration.

1. Introduction

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].

Networks customized to storage networking, such as Fibre 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 paper 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.

2. The Historical A

There are many historical examples of hardware accelerators to offload processor work from the main CPU. Some examples, such as network communications processors, have been successful, but many others have fallen far short of unmet expectations.

Examples of front-end accelerators date from the early days of computing. In many systems, the primary I/O processor to offload the main CPU. However, it has become in vogue to use a different architecture to deliver a faster rate of technology change.

A specific recent example is the use of an I/O processor, such as an Intel 80190, to offload the main CPU from its attached I/O device. The idea is to serve as an I/O processor for its attached I/O device. It started, the i960 embedded processor, but its performance did not live up to expectations as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that hardware accelerators should have been used more extensively. However, the cost of embedded programming and testing every protocol worthy of a name is high, and it is difficult to stay afloat when the hardware is far from stable, and even when formally approved, there are often bug fixes. It seems extremely

as the main CPU. If the performance does not keep up, at some point an accelerator becomes a decelerator. Somewhere in between, performance is about equal with or without the attached processor, but the development and support costs become a burden. The accelerator is usually a different CPU architecture than the main CPU, and it usually has a different software development environment. Maintaining two such environments is costly, and even if they were identical, there is overhead for inventing and testing the software interface between the processors. The software development cost eventually kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Secondary Considerations: Skepticism

This level of skepticism and teaching away is **directly related to the heart of the claimed invention – offloading network processing and handling of ACKs, and therefore has a direct nexus to the challenged claims.**

Ex. 2026 at 52

UNITED STATES PATENT AND

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP. and
CAVIUM, INC.,

Petitioners,

v.

ALACRITECH INC.,

Patent Owner.

Case IPR2017-01393¹
U.S. Patent 9,055,104

PATENT OWNER'S EXHIBIT 2026

DECLARATION OF KEVIN ALMEROOTH, PH.D.

¹ Cavium, who filed a Petition in Case IPR2017-01714, has been joined as a petitioner in this proceeding.

**Intel Corporation, Cavium, Inc., Wistron
Corporation, and Dell Inc.**

v.

Alacritech, Inc.

Case Nos.: IPR2017-01391, IPR2017-01392, IPR2017-01406,
IPR2017-01409, IPR2017-01410

**Patent Owner's Demonstratives
Motions to Exclude**

Hearing: September 13, 2018

Motions to Exclude, PO Demonstrative, 1

PO's Motion to Exclude: Dates on Tanenbaum

Case No. IPR2017-01391
U.S. Patent No. 7,237,036

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

INTEL CORPORATION
Petitioner,

v.

ALACRITECH, INC.,
Patent Owner

Case IPR2017-01391
U.S. Patent No. 7,237,036

PATENT OWNER'S OBJECTIONS TO EVIDENCE
UNDER 37 C.F.R. § 42.601

.

1

Exhibit 1006
(Tanenbaum96)

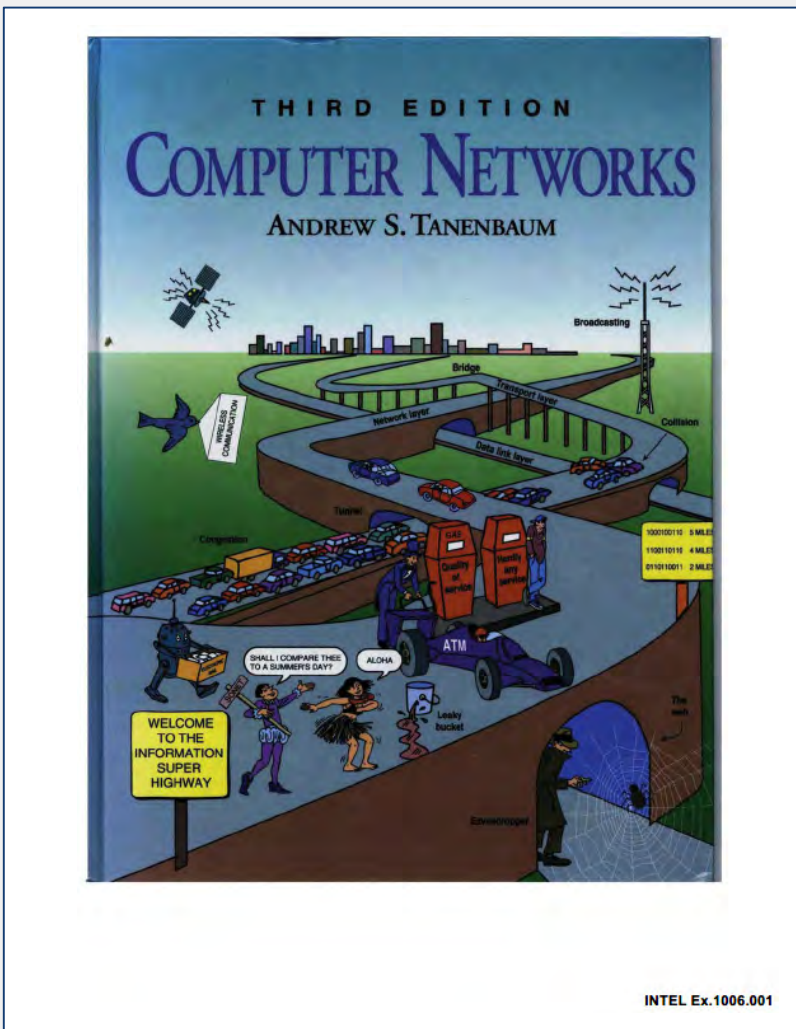
FRE 901: Patent Owner objects to this exhibit because Petitioner has failed to establish that this exhibit is what Petitioner claims it is, and has failed to authenticate this exhibit.

FRE 801: Patent Owner also objects to this exhibit because it is hearsay under FRE 801 and does not fall within the hearsay exceptions under FRE 803. **To the extent that Petitioner attempts to reply on any date that appears on this exhibit to establish public accessibility, the date is hearsay under FRE 801 and does not fall within the hearsay exceptions under FRE 803.**


Patent Owner also objects to this exhibit because Petitioner fails to establish that this exhibit is publicly available before the priority date of the patent at issue.

PO's Objection to Evidence, Paper 10 at 3.

PO's Motion to Exclude: Dates on Tanenbaum



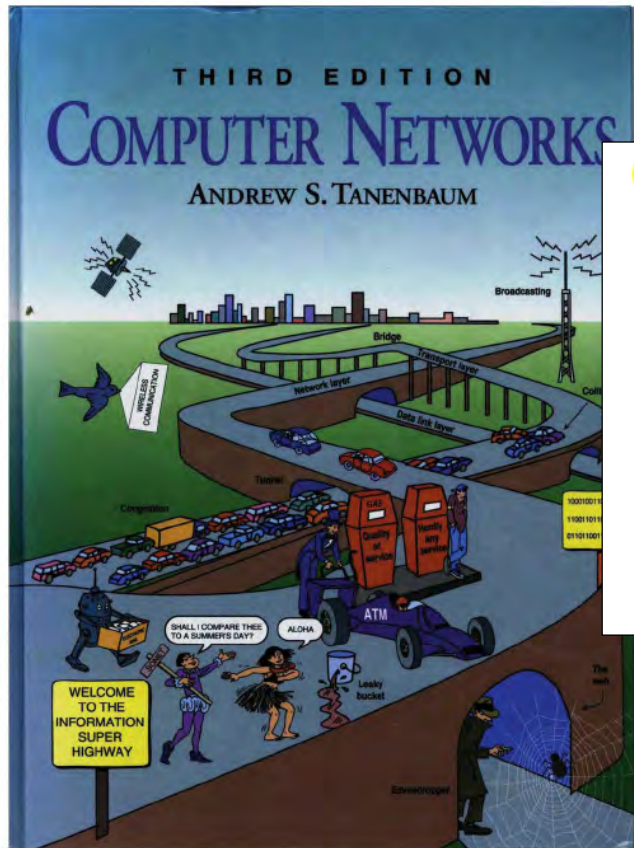
Tanenbaum (Ex. 1006)



© 1996 by Prentice Hall PTR
 Prentice-Hall, Inc.
 A Simon & Schuster Company
 Upper Saddle River, New Jersey 07458

Ex. 1006.005

PO's Motion to Exclude: Dates on Tanenbaum



INTEL Ex.1006.001

Tanenbaum (Ex. 1006)

Library of Congress Cataloging in Publication Data

Tanenbaum, Andrew S. 1944-

Computer networks / Andrew S. Tanenbaum. -- 3rd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-13-349945-6

I. Computer networks. I. Title.

TK5105.5.T36 1996

96-4121

004.6--dc20

CIP

Ex. 1006.005

PO's Motion to Exclude: Dates on Tanenbaum



“Accordingly, we determine that to the extent that the dates presented in [the] Exhibit [] are relied upon as proof of dates relevant to the creation or publication date of [the] Exhibit [] itself, **those dates are inadmissible hearsay.**”

Standard Innovation Corp. v. Lelo, Inc., IPR2014-000148, Paper 42 at 15-16 (April 23, 2015)

PO's Motion to Exclude: Dates on Tanenbaum

[The Library of Congress](#) > Cataloging in Publication Program

Cataloging in Publication Program

Print Subscribe Share/Save Give Feedback

A Cataloging in Publication record (aka CIP data) is a bibliographic record prepared by the Library of Congress for a book that has not yet been published. When the book is published, the publisher includes the CIP data on the copyright page thereby facilitating book processing for libraries and book dealers. [Learn more about the program](#)

Library of Congress Website Ex. 2500.001

How can I get cataloging for a book which is already published?

CIP data is available only for works that are not yet published. Published works are not eligible for CIP data.

Library of Congress Website Ex. 2501.002

PO's Motion to Exclude: Major Declaration

Case No. IPR2017-01391
U.S. Patent No. 7,237,036

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORP
Petitioner

v.

ALACRITECH
Patent Owner

Case IPR2017-01391
U.S. Patent No. 7,237,036

PATENT OWNER'S OBJECTION
UNDER 37 C.F.R. § 42.101

Exhibit 1011
(Librarian
Declaration of
Rice Majors)

FRE 602: Patent Owner objects to this exhibit because Petitioner does not introduce evidence of declarant's personal knowledge of the subject matter of the testimony contained therein.

FRE 701 and FRE 702: Patent Owner objects to this exhibit because it includes opinion testimony of lay witness without meeting the requirement of FRE 701 and Petitioner fails to establish the witness as an expert under FRE 702.

FRE 801: Patent Owner also objects to this exhibit because it is hearsay under FRE 801.

PO's Objection to Evidence, Paper 10 at 3.

PO's Motion to Exclude: Major Declaration

Declaration from Santa Clara University

I, Rice Majors, declare:

1. I am an Associate University Librarian ("SCU Library"). I am familiar with the SCU Library's online catalog and its procedures regarding the receipt, indexing, and availability of library materials.

2. According to SCU Library policies and procedures, library materials have been indexed in the online catalog (from 1992 to the present) and are made freely available to employees and students of Santa Clara University, and may be used in the library building by members of the general public. The SCU Library catalog is searchable by subject, author, and title (since 1992) and by keyword (since the mid-1990s).

3. The SCU Library holds a copy of a publication by Tanenbaum, Andrew S., Computer Networks. Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1983) ("Tanenbaum"). The SCU Library catalog is searchable by subject, author, and title (since 1992) and by keyword (since the mid-1990s).

4. When a monograph is received and cataloged by the SCU Library, the date of cataloging is set and retained in the catalog record. The catalog date ("Catalog Date") for Tanenbaum is November 1, 1996 (see Exhibit A). The volume would have been available for use within a few weeks of that date.

I declare under the penalty of perjury that the foregoing is true and correct.

Statements and the like are punishable by fine or imprisonment, or both under Section 1001 of Title 18 of the United States Code and may jeopardize the validity of the application or

2. According to SCU Library's policies and procedures, library materials have been indexed in the online catalog (from 1992 to the present) and are made freely available to employees and students of Santa Clara University, and may be used in the library building by members of the general public. The SCU Library catalog is searchable by subject, author, and title (since 1992) and by keyword (since the mid-1990s).

4. When a monograph is received and cataloged by the SCU Library, the date of cataloging is set and retained in the catalog record. The catalog date ("Catalog Date") for Tanenbaum is November 1, 1996 (see Exhibit A). The volume would have been available for use within a few weeks of that date.

Majors Declaration, Ex. 1011.001

PO's Motion to Exclude: Major Declaration

b15720184 Fri Jan 20 09:43:40 PST 2017
 Last Updated: 02-24-2015 Created: 08-09-1996 Revisions: 26

LANG engEnglish CAT DATE 11-01-1996 BCODE3 -
 SKIP 0 BIB LVL mMONOGRAPH/BOOK COUNTRY njuNew Jersey
 LOCATION umm University Library MAT TYP aBooks
 Main Stacks

MARC Leader #####cam a22##### a 4500

o 001 34079009
 y 003 OCoLC
 y 005 20141116012020.0
 y 008 960102t19961996njuab b 001 0 eng
 l 010 96004121
 i 020 0133499456
 y 040 nlcabnlcau-pbw

Fri Jan 20 09:43:40 PST 2017

b15720184 Last Updated: 02-24-2015 Created: 08-09-1996 Revisions: 26

r 337 unmediated |bn|2rdamedia
 r 338 volume|bnc|2rdacarrier
 n 504 Includes bibliographical references (p. 767-794) and index.
 d 650 0 Computer networks.
 y 918 .bckstg|b2014-10-14

Ex. 1011.003.



(16) Statements in Ancient Documents. A statement in a document that was prepared before January 1, 1998, and whose authenticity is established.

Fed. R. Evid. 803(16)

Ex. 1011.003.

PO's Motion to Exclude: Major Declaration



(6) Records of a Regularly Conducted Activity. A record of an act, event, condition, opinion, or diagnosis if:

(A) **the record was made at or near the time by — or from information transmitted by — someone with knowledge;**

(B) the record was **kept in the course of a regularly conducted activity** of a business, organization, occupation, or calling, whether or not for profit;

(C) **making the record was a regular practice of that activity;**

(D) all these conditions are shown by the **testimony of the custodian or another qualified witness**, or by a certification that complies with Rule 902(11) or (12) or with a statute permitting certification; and

(E) the opponent does not show that the source of information or the method or circumstances of preparation indicate a lack of trustworthiness.

Fed. R. Evid. 803(6)

Motions to Exclude, PO Demonstrative, 10

PO's Motion to Exclude: Major Declaration



Residual Exception

(a) In General. Under the following circumstances, a hearsay statement is not excluded by the rule against hearsay even if the statement is not specifically covered by a hearsay exception in Rule 803 or 804:

- (1) **the statement has equivalent circumstantial guarantees of trustworthiness;**
- (2) it is offered as evidence of a material fact;
- (3) **it is more probative on the point for which it is offered than any other evidence that the proponent can obtain through reasonable efforts;** and
- (4) **admitting it will best serve the purposes of these rules and the interests of justice.**

Fed. R. Evid. 807(a)