(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 June 2016 (30.06.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/102004 A1

(51) International Patent Classification:
*H04L 29/06* (2006.01)

(21) International Application Number:
PCT/EP2014/079173

(22) International Filing Date:
23 December 2014 (23.12.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventors; and
(71) Applicants *(for US only)*: **TOUITOU, Dan** [IL/DE]; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE). **GAMPEL, Eran** [IL/DE]; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE).

(74) Agent: **KREUZ, Georg**; Huawei Technologies Duesseldorf GmbH, Messerschmittstr. 4, 80992 Munich (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published**:
— *with international search report (Art. 21(3))*

(54) **Title**: NETWORK EXTENDED TCP SPLICING

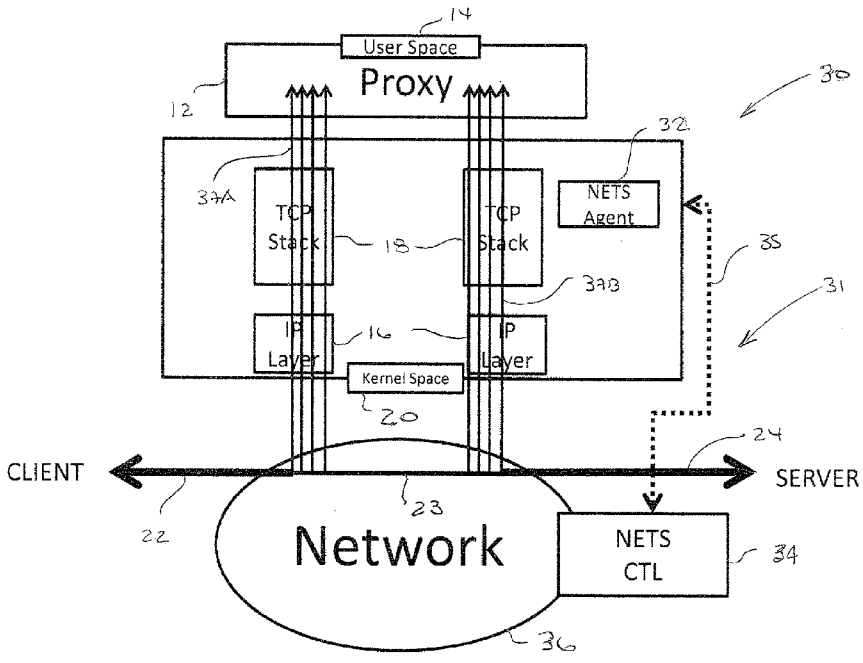


FIG. 3

(57) **Abstract**: A system for offloading information transfer between a client and a server in a communications network is disclosed. The system includes a network agent to issue a data transfer offload command responsive to receiving a TCP (transmission control protocol) splice command from an application proxy. The system additionally includes a network controller to offload data transfer from a kernel space to a network responsive to the data transfer offload command.

WO 2016/102004 A1

# NETWORK EXTENDED TCP SPLICING

## FIELD OF THE INVENTION

The present invention relates to network communication generally and to a system and a method for extending the functionality of TCP (Transmission Control Protocol) splicing in network communications in particular.

## BACKGROUND OF THE INVENTION

Many of today's communication networks rely on proxies to deliver network functions. Generally, a proxy may act as an intermediary between a client and a server in order to provide additional functionality such as better user experience.

Many proxies are implemented as software programs which run in user space at the application level of the TCP/IP stack or OSI stack. In the application level, the proxy may sit on a user's computing device as an application proxy where it may intercept connections between the computing device (client) and a server and operate on these connections. Example of these application proxies may include HTTP proxies, SOCKS proxies, and WEB proxies, among others. An application proxy may be used as a load balancer that intercepts a request coming from a client, analyzes it and based on the result of the analysis may select a server to whom it may forward the request for more efficient delivery of the client's request. Application proxies may also be used for content filtering, for caching, for accessing specific network sites, among numerous other applications.

An example of an application proxy implemented in user space is shown in the representative client – server connection shown in Figure 1. In the figure are illustrated a client-server connection 10 including application proxy 12 in user space 14, and IP layer 16 and TCP stack 18 in kernel space 20. Information transfer between the client and the server may include sending packets, and

is shown by the network connection 22 between the client and proxy 12 and the network connection 24 between the proxy and the server.

In client-server connection 10, packets sent from the client to the server may be sent through network connection 22 to IP layer 16 and TCP stack 18 where the information contained by the packets may be copied into buffers in the kernel space 20. From kernel space 20 the packets may be sent into user space 14 where again the information contained by the packets may be copied into buffers by proxy 12. In addition to copying the information, proxy 12 may additionally operate on the information, including making changes in the information, before sending the packets to the server through network connection 24. In a reverse direction, packets from the server to the client may be sent through network connect 24 to IP layer 16 and TCP stack 18 where again the information contained therein may be copied into the buffers in kernel space 20, and from there into user space 14 where again the information may be copied into the buffers by proxy 12. Again, proxy 12 may perform additional operations on the information received from the server before sending it onwards to the client.

Proxy 12 may intercept the client-server connections at the socket layer and may identify the client and server each by information which may include an address and a port number contained in the packets (client socket and server socket). This may eliminate proxy 12 having to copy the TCP and IP headers in the packets into the buffers in user space 14, which may somewhat improve the performance of client – server connection 10. Nevertheless, the rest of the information being transferred between the client and the server may still be copied into the buffers in user space 14. As a result, numerous calls may still be required to access the data from the buffers. Furthermore, the packets are still required to go through all levels of TCP stack 18 at least twice. Both these conditions may contribute to lowering system performance.

In an effort to improve the performance of the representative client – server connection previously described, a technique known as TCP splicing may be used. Such a technique is described in U.S.

Patent No. 5,941,988 to Bhagwat et al. wherein is disclosed "A method of merging two separate TCP connections terminating at a common host and 'gluing' them into a single connection between two end systems, where the single connection preserves TCP end-to-end semantics. The technique retains the session setup functions of the transport layer proxy, but provides a method to push the data copying into kernel space to improve the relay operation. More specifically, a byte stream arriving on one end of the split connection is mapped directly into the sequence number space of the other split connection. This process of mapping, or TCP gluing, involves updating a subset of TCP and IP header fields; that is, source and destination addresses, port numbers, sequence numbers and checksum. The changes to the TCP/IP packet headers are performed on-the-fly as packets are relayed over the glued connection between the original separate TCP connections".

An example of the application of the above technique is now described with reference to Figure 2 which schematically illustrates a client – server connection 11 including the use of TCP splicing. Client-server connection 11 includes proxy 12 in user space 14, and IP layer 16 and TCP stack 18 in kernel space 20. Client – server connection is initially established by proxy 12 in user space 14, for example through connection 27A through which packets initially received from the client through network connection 22 may be sent from kernel space 20 to user space 14, and through connection 27B for packets initially received from the server through network connection 24. Once proxy 12 determines from the initial packets the client socket and the server socket, a splice connection 26 may be established by the proxy in kernel space 20 within IP layer 16. So long as splice connection 26 is maintained, the packets are transferred directly through IP layer 16 without having to go through proxy 12, substantially increasing system performance compared to representative client – server connection 10 shown in Figure 1.

## SUMMARY OF THE PRESENT INVENTION

The object of the present invention is to improve a network communication between a client and a server. This object is solved by the subject matter of the independent claims. The dependent claims protect further embodiments thereof.

There is provided, in accordance with an embodiment of the present invention, a system for offloading information transfer between a client and a server in a communications network including a network agent to issue a data transfer offload command responsive to receiving a TCP (transmission control protocol) splice command from an application proxy. The system additionally includes a network controller to offload data transfer from a kernel space to a network responsive to the data transfer offload command.

In accordance with an embodiment of the present invention, the network agent resides in the kernel space of a TCP/IP stack.

In accordance with an embodiment of the present invention, the network agent resides in the TCP stack in the kernel space of a TCP/IP stack.

In accordance with an embodiment of the present invention, the network controller resides on the network.

In accordance with an embodiment of the present invention, the network controller includes a load balancer.

In accordance with an embodiment of the present invention, the network agent receives the TCP splice command in a socket layer of a TCP/IP stack.

In accordance with an embodiment of the present invention, the system additionally includes the network.

In accordance with an embodiment of the present invention, the network is a software-defined network (SDN).

In accordance with an embodiment of the present invention, the network controller includes a protocol oblivious forwarding controller.

In accordance with an embodiment of the present invention, the network controller directly connects the client with the server through the network.

There is provided, in accordance with an embodiment of the present invention, a method for offloading information transfer between a client and a server from a TCP/IP stack in a communications network, the method including transferring proxy functionality from an application proxy to a network controller responsive to the application proxy issuing a TCP splice command.

In accordance with an embodiment of the present invention, the method additionally includes receiving the TCP splice command by a network agent.

In accordance with an embodiment of the present invention, the receiving includes intercepting the TCP command.

In accordance with an embodiment of the present invention, the method additionally includes the network agent sending an offload command to the network controller responsive to the receiving.

In accordance with an embodiment of the present invention, the receiving is through a socket layer of the TCP/IP stack.

In accordance with an embodiment of the present invention, the method additionally includes the network controller modifying a TCP and/or an IP header in a packet.

In accordance with an embodiment of the present invention, the method additionally includes the network controller maintaining proxy functionality according to any one of a duration of the lifetime of sockets; an amount of time required to transfer information from the client to the server and/or from the server to the client; an amount of information transferred from the client to the server and/or from the server to the client; a time-out value to set up a network splice; and a request or override received from the application proxy.

In accordance with an embodiment of the present invention, the method additionally includes the network controller returning proxy functionality to the application proxy.

In accordance with an embodiment of the present invention, the method additionally includes the network controller offloading the information transfer to a network.

## BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanying drawings in which:

Figure 1 schematically illustrates a representative client – server network connection;

Figure 2 schematically illustrates a representative client – server network connection including use of TCP splicing;

Figure 3 schematically illustrates a client – server network connection including a Network Extended TCP Splicing system, according to an embodiment of the present invention; and

Figure 4 is a flow chart of a method of offloading information to a network using the Network Extended TCP Splicing system, according to an embodiment of the present invention.

It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference numerals may be repeated among the figures to indicate corresponding or analogous elements.

## DETAILED DESCRIPTION OF THE PRESENT INVENTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, and components have not been described in detail so as not to obscure the present invention.

Applicants have realized that, despite the advantages provided by the use of known TCP splicing techniques, the functionality of the TCP splicing may be limited by the capacity of the hardware on which the application proxy sits. For example, a CPU may have insufficient processing power to drive the TCP/IP software, or may experience increased utilization due to latency in memory devices and I/O devices. Whichever may be the case, hardware capacity limitations may degrade overall system performance.

Applicants have further realized that the above problem associated with limited hardware capacity may be overcome by offloading information transfer to the network and recalling the information "on request". Applicants have additionally realized that techniques used for TCP splicing which enable the application proxy to offload information transfer between two sockets to the OS kernel (kernel space) may be applied to enable the application proxy to offload information transfer between the two sockets to the network. Offloading to the network, Applicants have realized, may leverage network forwarding performance through increased proxy capacity.

Description of Network Extended TCP Splicing System

Reference is now made to Figure 3 which schematically illustrates a client – server network connection 30 including a Network Extended TCP Splicing (NETS) system 31, according to an embodiment of the present invention. Client-server connection 30 may additionally include application proxy 12 in user space 14, and IP layer 16 and TCP stack 18 in kernel space 20. NETS system 31 may include a NETS agent 32 and a NETS controller 34. NETS system 31 may

additionally an altered configuration of TCP stack 18 which may include NETS agent 32 as described further on below. NETS system 31 may additionally include a network 36 which may be implemented through SDN (software-defined networking) and may include use of protocol oblivious forwarding (POF).

NETS system 31 may use application proxy 12 interceptions of client-server connections at the socket layer to offload information transfer between two sockets (client and server) to network 36. At network 36, NETS system 31 may create a NETS TCP splice 23 to allow a direct connection of the client with the server over the network instead of through kernel space 20 as is commonly done in the art. Furthermore, NETS system 31 may transfer proxy functionality from application proxy 12 to network 36 to allow load balancing to be performed at the network.

As part of the offloading process, NETS system 31 may maintain proxy functionality at network 36 based on one or more predetermined criteria. These may include (a) a duration of the lifetime of the sockets; (b) an amount of time required to transfer the information from the client to the server and/or from the server to the client; (c) an amount of information to be transferred from the client to the server and/or from the server to the client; (d) a time-out value to set up the splice; (f) an override received from the application proxy; (g) or other events which may be associated with the client – server connection and information transfer; (h) or any combination thereof. NET system 31 may additionally transfer proxy functionality from network 36 to application proxy 12 to return control of the sockets "on request", which may be upon termination of any one or any combination of the previously mentioned events or upon determination that there is no need to offload information.

As may be appreciated from Figure 3, as a result of creating NETS TCP splice 23 interconnecting network connections 22 and 24, packets may be directly transferred between the client and the server through network 36 without passing through kernel space 20 (and also without passing through user space 14). In operation, the connection between the client and the server may be

initially established by proxy 12, for example, through connections 37A and 37B in a similar fashion to connections 27A and 27B in Figure 2. Once the connection is established NETS system 31 may then create NETS TCP splice 23 offloading information transfer to network 36 with proxy functionality transferred to the NETS system. Upon termination of the offloading, proxy functionality may be returned back to application proxy 12 as shown by connection 37A for packets sent by the client and by connection 37B for packets sent by the server.

NETS agent 32 may receive TCP splicing commands from proxy 12 and, responsively, send offload commands to NETS controller 34, as shown by double headed arrow 35. The splicing commands may be intended for the socket layer and may be intercepted by NETS agent 32 acting as a proxy. Alternatively, the splicing commands may be specifically intended for NETS agent 32. On some occasion, NETS agent 32 may receive the splicing commands from proxy 12 and may not act on the commands (i.e. not send offload commands to NETS controller 34), for example, when network 36 capacity is not suitable for offloading.

NETS agent 32 may include a module which may be integrated into TCP stack 18 (altered TCP stack) in kernel space 20. The module may be integrated within the socket layer, or in addition to the socket layer in TCP stack 18. Alternatively, NETS agent 32 may replace the socket layer. NETS agent 32 may include hardware and/or software, and may additionally include an API (application program interface) for interfacing with application proxy 12 and/or NETS controller 34.

NETS controller 34 may be integrated to network 36 and may receive the offload commands from NETS agent 32. NETS controller 34 may additionally translate the commands into network device configurations to create NETS TCP splice 23 and allow load balancing to be performed in network 36. NETS controller 34 may additionally send information to NETS agent 32 associated with network 36 capacity. The network information may be sent to NETS agent 32 through a northbound interface, also indicated by double headed arrow 35.

NETS controller 34 may use fast path offloading (FPO) to offload traffic processing to network 36. Use of FPO may allow NETS controller 34 to forward and redirect one or more packets in network 31 to another destination by modifying the TCP and IP fields, for example, by changing TCP and IP headers in the packets. The modifications made to the TCP and IP headers may include information associated with source and destination IP, source and destination port and event TCP.seq and TCP.ack number.

Implementation Example of Socket Extension in NETS System

Following are exemplary function calls which may be used by the NETS System to perform the various functions described below:

(a) int NETS_attach(int fd1, int fd2)

/* attaches two sockets for future offload.

returns fd to newly created control socket */

(b) int write(int ctl_sckt, char *ctl_buf, int)

/* ctl_buf contains request for offloading + parameters such as timeout, byte number etc… if request successful, the application is expected not to read and/or write from the sockets attached to the control socket. */

(c) struct pollfd

/* new event : NETS_resume This is the event we expect to receive while polling on a control socket. It means that NETS has returned control of the attached sockets to the application/ Upon reception of the NETS_resume the application can read from the control socket data relevant to the resume such as number of bytes transferred. Also, after NETS_resume that

```
        application can resume reading and writing from/to the attached
        sockets. */
int read(int ctl_sckt, char *CTL_buf, int)
                /* read from control socket a control buf. Called after
        NETS_resume received on control socket during poll. Contains
        information such as – number of bytes sent since last offload,
        time elapsed since last offload & flags such a FIN/RST sent
        etc…*/
```

Exemplary Method for Offloading Using NETS

Following is described an exemplary method 400 of offloading information transfer to a network including a NETS TCP splice, according to an embodiment of the present invention. For clarity, exemplary method 400 described herein will be explained with reference to NETS system 31 and client – server connection 30. Furthermore, the ordinary person skilled in the art may realize that method 400 may be practiced with more or less steps and/or with a different sequence of steps.

At step 402, the server socket and the client socket may be determined by application proxy 12. The sockets may be determined from one or more initial packets received through network connection 22 and/or server connection 24 and which pass through kernel space 20 and are intercepted in the socket layer by application proxy 12.

At step 404, application proxy 12 may issue a TCP splice command to create a TCP splice in kernel space 20 based on the sockets' information. The TCP splice command may be intercepted by NETS agent 32.

At step 406, an offload command may be issued by NETS agent 32 to NETS controller 34 responsive to the TCP splice command from application proxy 12.

At step 408, NETS controller 34 may receive the offload command from NETS agent 32 and responsive to the command, takes proxy control of network 36. NETS controller may adjust network device configurations in the packets, for example, by modifying the TCP and IP headers in the packets to perform load balancing. NETS controller 34 may retain control of the proxy functionality according to predetermined criteria which may include any one or combination of (a) a duration of the lifetime of the sockets; (b) an amount of time required to transfer the information from the client to the server and/or from the server to the client; (c) an amount of information to be transferred from the client to the server and/or from the server to the client; (d) a time-out value to set up the splice; (f) a request or override received from the application proxy; (g) or other events which may be associated with the client – server connection and information transfer. While proxy functionality is maintained by NETS controller 34, information transfer between the client and the server is direct through network 36 (through network splice 23).

At step 410, NETS controller 34 returns control of the proxy functionality to application proxy 12 upon termination of the offloading. Termination may be based on the predetermined criteria for returning control and may include signaling to NETS agent 32 that the control is to be returned. Alternatively, signaling that the control of proxy functionality is to be returned to proxy 12 may originate from NETS agent 32 according to the predetermined criteria. The signaling may include signaling to application proxy 12. Network splice 23 is broken and information transfer returns to proxy 12 through kernel 20 into user space 14.

Unless specifically stated otherwise, as apparent from the preceding discussions, it is appreciated that, throughout the specification, discussions utilizing terms such as "processing," "computing," "calculating," "determining," or the like, refer to the action and/or processes of a computer, computing system, or similar electronic computing device that manipulates and/or transforms data represented as physical, such as electronic, quantities within the computing system's registers

and/or memories into other data similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices.

Embodiments of the present invention may include apparatus for performing the operations herein. This apparatus may be specially constructed for the desired purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk, including floppy disks, optical disks, magnetic-optical disks, read-only memories (ROMs), compact disc read-only memories (CD-ROMs), random access memories (RAMs), electrically programmable read-only memories (EPROMs), electrically erasable and programmable read only memories (EEPROMs), magnetic or optical cards, Flash memory, or any other type of media suitable for storing electronic instructions and capable of being coupled to a computer system bus.

The processes and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the desired method. The desired structure for a variety of these systems appears from the description above. In addition, embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and equivalents will now occur to those of ordinary skill in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the true spirit of the invention.

## CLAIMS

What is claimed is:

1. A system for offloading information transfer between a client and a server in a communications network comprising:

a network agent to issue a data transfer offload command responsive to receiving a TCP (transmission control protocol) splice command from an application proxy; and

a network controller to offload data transfer from a kernel space to a network responsive to the data transfer offload command.

2. A system according to claim 1 wherein said network agent resides in the kernel space of a TCP/IP stack.

3. A system according to claim 1 wherein said network agent resides in the TCP stack in the kernel space of a TCP/IP stack.

4. A system according to claim 1 wherein said network controller resides on the network.

5. A system according to claim 1 wherein said network controller comprises a load balancer.

6. A system according to claim 1 wherein said network agent receives the TCP splice command in a socket layer of a TCP/IP stack.

7. A system according to claim 1 further comprising said network.

8. A system according to claim 7 wherein said network is a software-defined network (SDN).

9. A system according to claim 8 wherein said network controller comprises a protocol oblivious forwarding controller.

10. A system according to claim 1 wherein said network controller directly connects the client with the server through said network.

11. A method for offloading information transfer between a client and a server from a TCP/IP stack in a communications network, the method comprising:

transferring proxy functionality from an application proxy to a network controller responsive to the application proxy issuing a TCP splice command.

12. A method according to claim 11 further comprising receiving the TCP splice command by a network agent.

13. A method according to claim 12 wherein said receiving comprises intercepting the TCP command.

14. A method according to claim 12 further comprising the network agent sending an offload command to the network controller responsive to said receiving.

15. A method according to claim 12 wherein said receiving is through a socket layer of the TCP/IP stack.

16. A method according to claim 12 further comprising said network controller modifying a TCP and/or an IP header in a packet.

17. A method according to claim 11 further comprising the network controller maintaining proxy functionality according to any one of a duration of the lifetime of sockets; an amount of time required to transfer information from the client to the server and/or from the server to the client; an amount of information transferred from the client to the server and/or from the server to the client; a time-out value to set up a network splice; and a request or override received from the application proxy.

18. A method according to claim 11 further comprising the network controller returning proxy functionality to the application proxy.

19. A method according to claim 11 further comprising the network controller offloading the information transfer to a network.

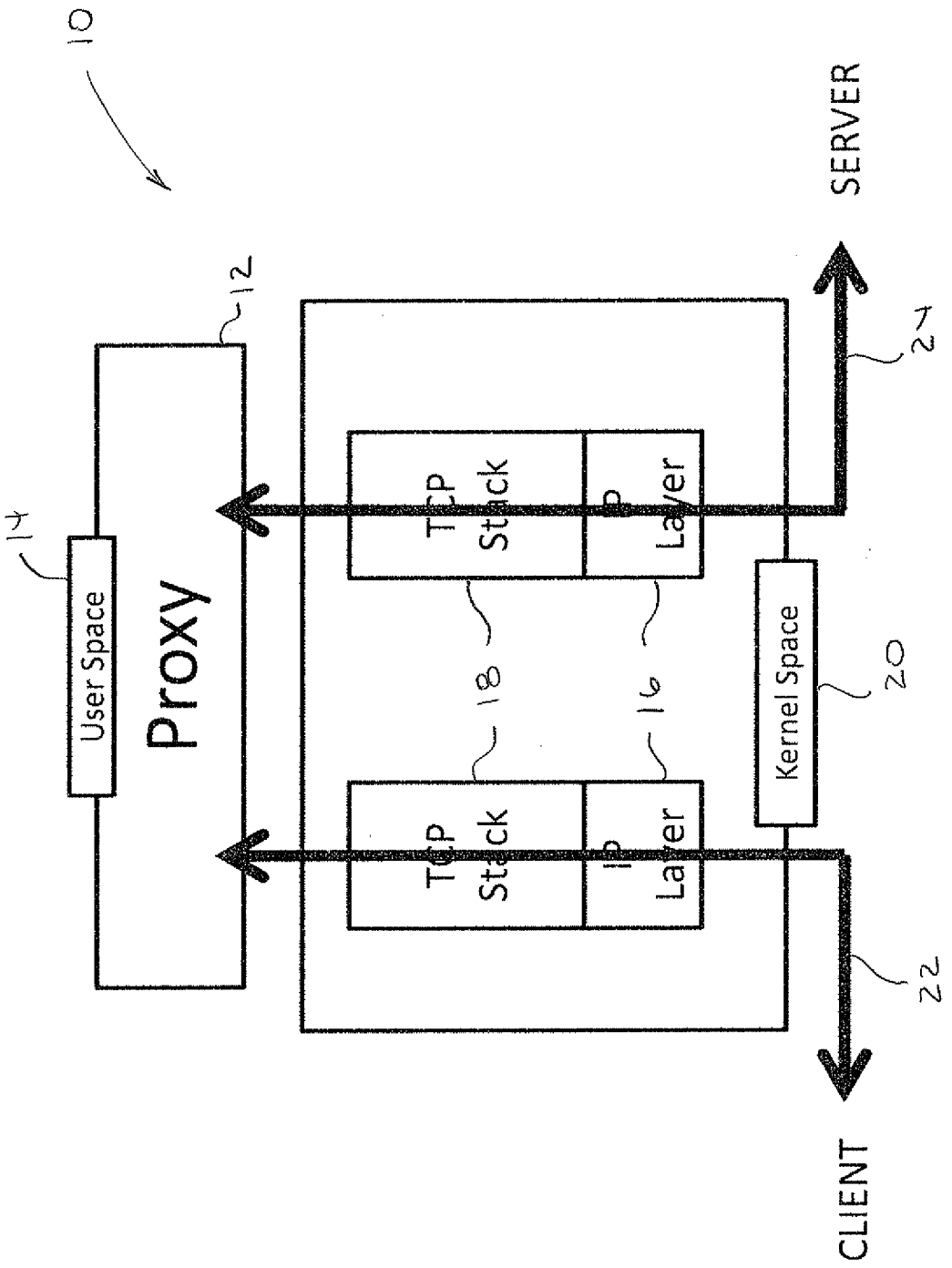
10
SERVER
12
27
User Space
Proxy
14
TCP Stack
IP Layer
18
16
Kernel Space
20
22
CLIENT

FIG. 1

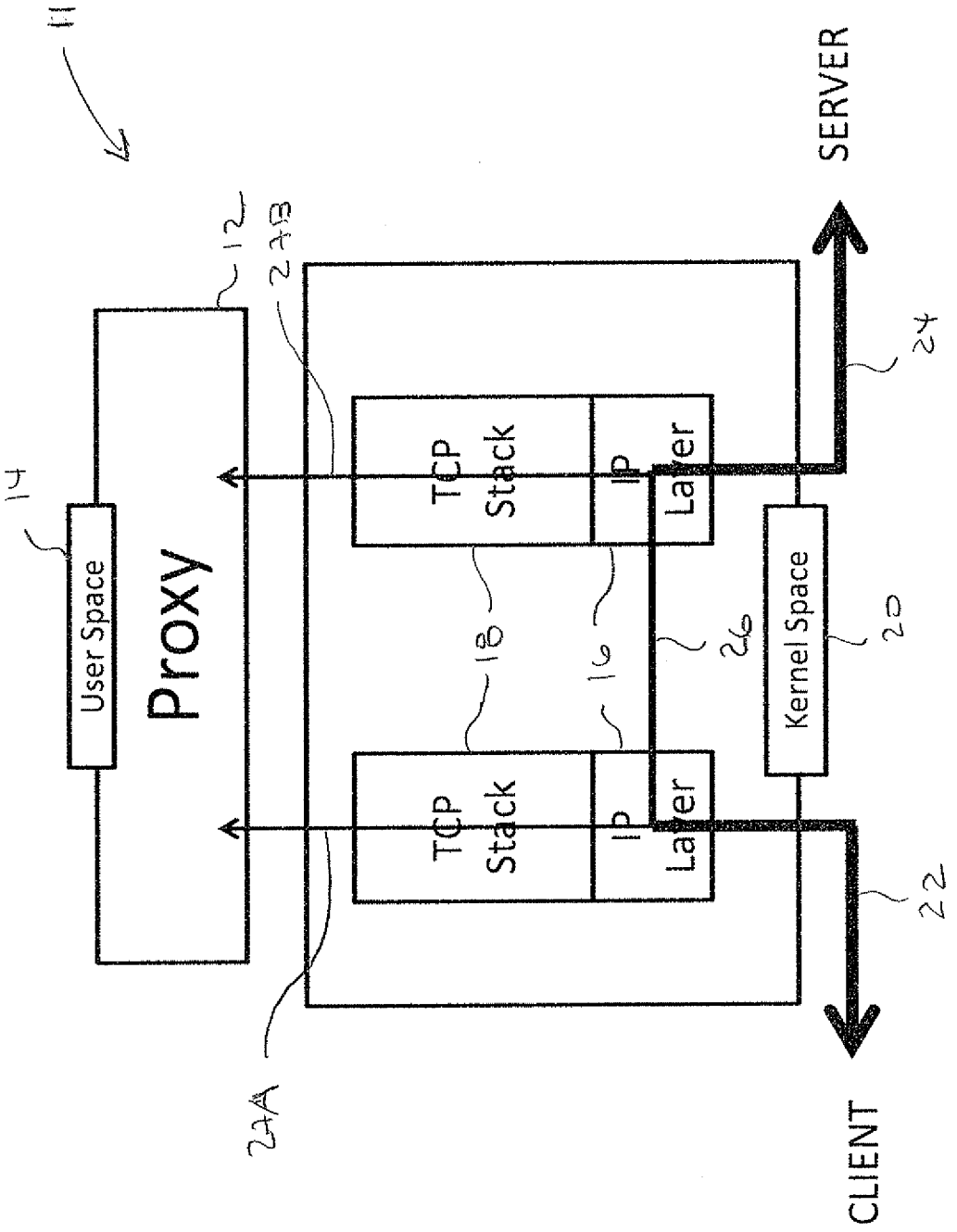
11
12
23
14
User Space
Proxy
TCP Stack
IP Layer
18
16
26
Kernel Space
20
24
SERVER
22
CLIENT
24

FIG. 2

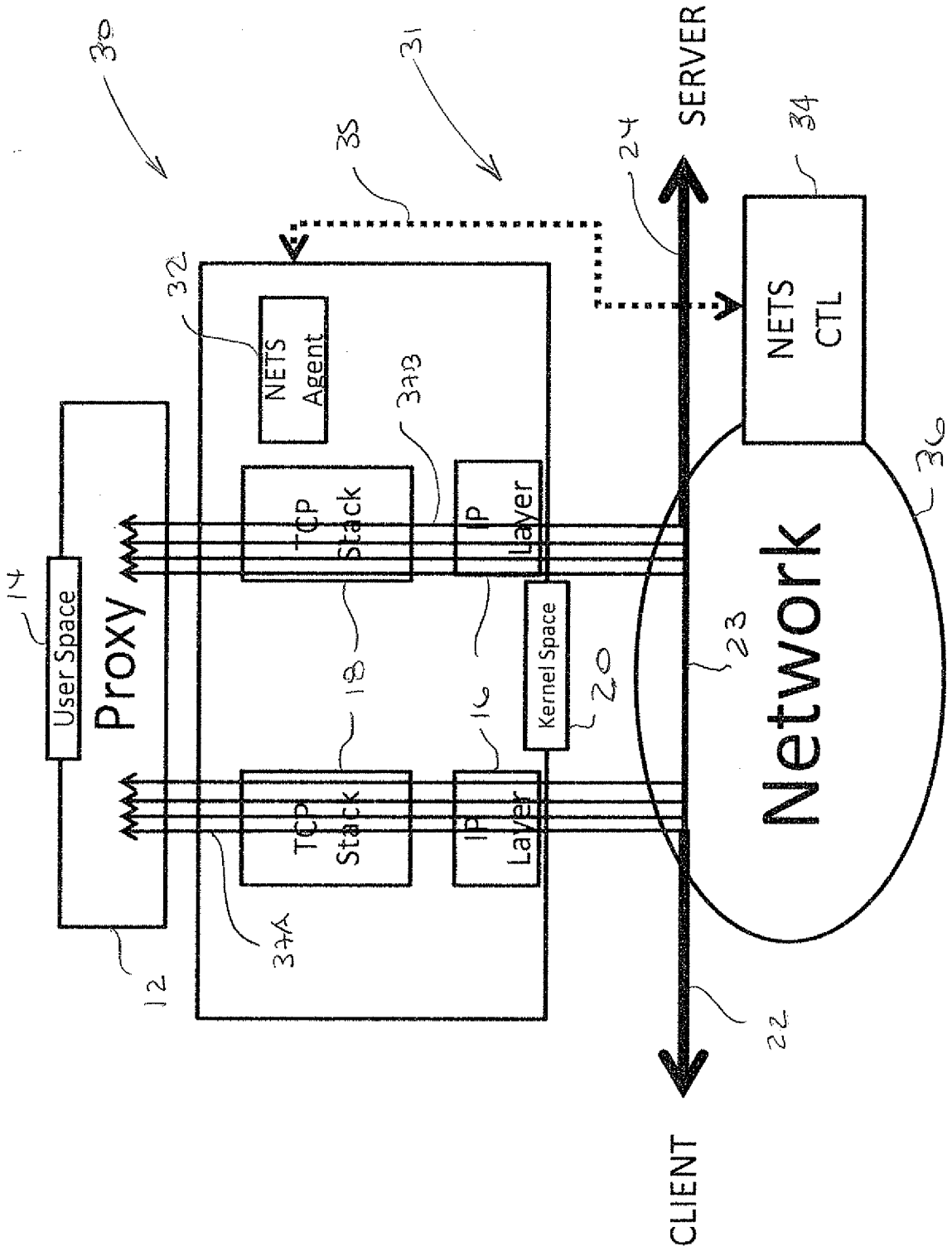
30
31
35
24
SERVER
34
NETS CTL
32
NETS Agent
33B
TCP Stack
IP Layer
User Space
Proxy
14
18
16
Kernel Space
20
23
Network
36
34A
12
22
CLIENT

FIG. 3

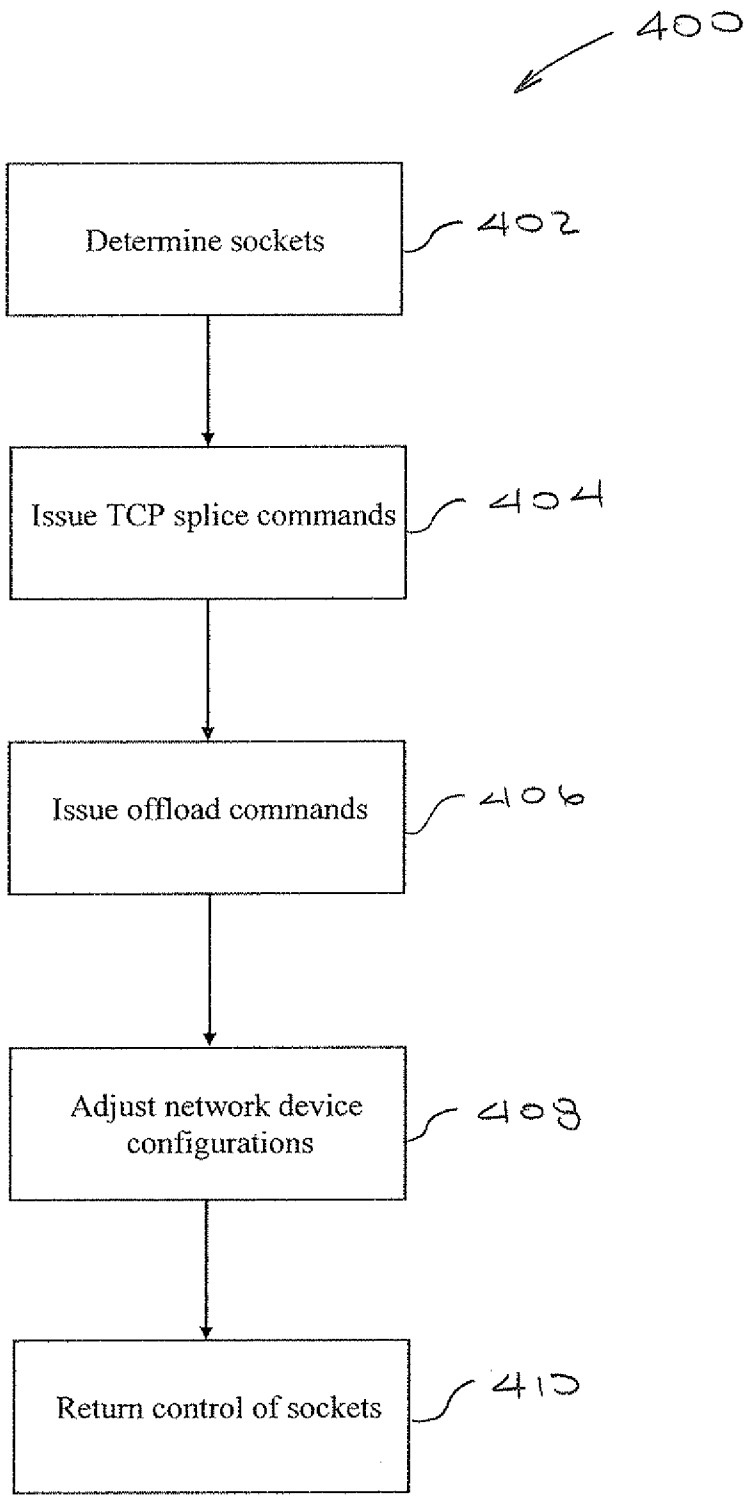
400
Determine sockets
402
Issue TCP splice commands
404
Issue offload commands
406
Adjust network device configurations
408
Return control of sockets
410

FIG. 4

# INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/079173

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | Li Zhao, Yan Luo, Laxmi Bhuyan: "SpliceNP: A TCP Splicer using A Network Processor", , 1 October 2005 (2005-10-01), XP002741884, Retrieved from the Internet: URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4675273 [retrieved on 2015-07-07] page 135 page 135, column 2, paragraph 2 page 136, column 1, paragraph 1 - paragraph 2 page 137, column 1, last paragraph - column 3, paragraph 2 page 138; figures 4b,4c | 1-19 |

-----

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 7 July 2015 | 20/07/2015 |

| Name and mailing address of the ISA/ | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Manea, Anda |

1

Form PCT/ISA/210 (second sheet) (April 2005)

# INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/079173

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | US 2006/190609 A1 (CHETUPARAMBIL MADHU K [US] ET AL) 24 August 2006 (2006-08-24) the whole document | 12-19 |

1

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2014/079173

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 2006190609 A1 | 24-08-2006 | CN 1829228 A | 06-09-2006 |
| | | US 2006190609 A1 | 24-08-2006 |