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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/91412 A2

(51) International Patent Classification[7]: H04L 29/06

(21) International Application Number: PCT/IL01/00471

(22) International Filing Date: 23 May 2001 (23.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
136324 24 May 2000 (24.05.2000) IL

(71) Applicant *(for all designated States except US)*: **SOFTCOM COMPUTERS LTD.** [IL/IL]; 9 Hasivim Street, 49170 Petach-Tikva (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **AMIT, Noah** [IL/IL]; 20 D'Israeli Street, 34334 Haifa (IL). **AMIT, Yoni** [IL/IL]; 20 D'Israeli Street, 34334 Haifa (IL). **EADAN, Zvi** [IL/IL]; 20 Hagolan Street, 81504 Yavne (IL).

(74) Agent: **FREIMANN, Daniel**; P.O. Box 29814, 52 Nahalat Benjamin Street, 61297 Tel Aviv (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), 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, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/91412 A2

(54) Title: METHOD OF SURVEILLING INTERNET COMMUNICATION

(57) Abstract: A network probe terminal for tracking a network communication line and simulating a browser activity of a given terminal. The probe terminal monitors TCP/IP data packets routed through the communication line for filtering relevant requests and responses relating to a given IP address. These requests and responses are analyzed and sorted according to their type and content. based on the analysis, the probe terminal identifies all relevant data transactions relating to the navigation process of the given terminal. The probe terminal activates a virtual browser simulating the processing of identified data transactions to create navigation presentations similar to the real navigation as seen by the user of the given terminal.

METHOD OF SURVEILLING INTERNET COMMUNICATION

BACKGROUND OF THE INVENTION

The present invention is directed to a method and system for enabling surveillance and monitoring of networks communications by analysis of data traversing therethrough.

A huge amount of traffic is flowing through today's computer networks, not all of which is benign. Thus, an owner or supervisor of a given network may be most interested to be able to track or "listen in" in real time in order to effectively monitor and/or secure the network. Such monitoring or surveillance can be achieved by connecting a probe to the network in order to monitor data traveling between two of more nodes (e.g., user workstations) on the network.

In a system where communication between two nodes is in a form of discrete packets, the network probe can "read" a packet of data in order to gather information, such as regarding the sources and the destination addresses of the packet, or the protocol of the packet. In addition, statistical and related information can be computed such as the average or total amount of traffic of a certain protocol type during a given period of time, or the total number of packets being sent to or from a node. This information may be reported to a system administrator in real-time, or may be stored for later analysis.

Various attempts have already been made in this direction. For example, Clear View Network Window, a software program available from

Clear Communications Corporation, of Lincolnshire, Ill., U.S.A, allegedly offers predictive/proactive maintenance, intelligent root-cause analysis, and proof-of-quality reports. However, the output is designed for network fault management, which is not the same as "tapping" into a communication between nodes in the network. Thus, the Clearview system does not allow monitoring of data transferred between two nodes in the network with regard to contents or characteristics.

Livermore National Laboratory, Livermore, Cal., U.S.A, developed a group of computer programs to protect the computers of the U.S. Department of Energy by "sniffing" data packets that travel across a local area network. The United States Navy used one of these programs, known as the "iWatch" program, in order to wiretap on communications of a suspected computer hacker who had been breaking into computer systems at the U.S. Department of Defense and NASA. The iWatch program uses a network probe to read all packets that travel over a network and then "stores" this information in a common database. A simple computer program can then be written to read through the stored data, and to display only predefined "interesting" pieces of information.

Whenever an interesting piece of information is found, the stored data is rescanned and a specific number of characters located at both sides of the "interesting" piece are reported. These interesting characters are then

reviewed in order to determine the content of the message and used as a guide to future monitoring activity.

This system is restricted to history analyze of user activities and does not enable complete "tapping" of all user activities and full simulation of the users surfing activity.

Three major problems are encountered in the way of achieving continuous and reliable tracking:

(a) Individual browsers do not report all the activities performed to a web server. For example, when a browser loads web pages from its browser cache space or from a proxy server, it does not send requests to any "remote" web server through the cyberspace autostrade;

(b) Application programs designed to perform certain features by web browser of one manufacturer are usually not compatible with those manufactured by another vendor because browser interface mechanisms are different and proprietary to each one of them; and

(c) Individual browsers send their requests to web servers in a non-systematic order In other words, with regard to a given web server, a preceding request has no relation to a subsequent request. In processing of requests, a web site has no control over the sequences of the requests.

In an attempt to overcome these problems, US Patent No. 5,951,643 refers to a mechanism for dependably organizing and managing information for web synchronization and tracking among multiple consumer browsers.

However, this solution is limited to tracking activities of identified users, who agreed to be "tapped" and willingly cooperated and be connected to the host with designated application.

It is thus the prime object of the invention to provide a monitoring and surveillance method and system enabling network communication suppliers to tap any user connected to the network.

It is a further object of the invention to provide a tapping methodology enabling network communication suppliers to watch in real time all user activities while communicating a net work.

It is a still further object of the invention to enable web-site owner to monitor and tap users contacting their web site

SUMMARY OF THE INVENTION

Thus provided according to the present invention is a method of tracking a network communication line by network probe terminal ("terminal agent") simulating a browser ("original browser") activity of a given terminal comprising the steps of accessing the network communication line, tracing TCP/IP data packets routed through the communication line, selecting TCP/IP data packets relating to a given IP address; ("identified data packets"), selecting from the identified data packets current requests for new

connections ("original requests"), selecting from the identified data packets current web-page components indicating new addresses ("new navigation components"), dividing the new navigation components into two categories, embedded objects or frames ("false new components"), hyperlinks ("true new components"), dividing the original requests into original requests matching true the new components, or original requests failing to match any new connection components and belonging to HTTP or POST type as "primary requests", original requests matching the false components as "secondary requests", selecting from identified data packets, HTML data files relating to primary requests; ("respective primary responses"), generating "virtual" secondary requests according to the respective secondary responses, selecting from identified data packets responses relating to secondary virtual requests, ("respective secondary responses") and simulating web page presentation on the terminal agent according to the respective secondary responses.

BRIEF DESCRIPTION OF THE DRAWINGS

These and further features and advantages of the invention will become more clearly understood in the light of the ensuing description of a few preferred embodiments thereof, given by way of example only, with reference to the accompanying drawings, wherein-

Fig. 1 illustrates a typical network configuration, in which the present invention can be implemented;

Fig. 2 illustrates the terminal agent scheme of operation;

Fig. 3 illustrates the process of tracing and identifying TCP/IP data packets;

Fig. 4 is a flowchart of classifying TCP/IP requests;

Fig. 5 is a flowchart of simulating the creation of virtual secondary TCP/IP requests; and

Fig. 6 illustrates the process of simulating original browser activities.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to Fig. 1, let us assume that terminals 01, 02... are connected to the same communication line, where the communication line is used as internal network ("Intranet"), or external network such as the Internet. According to the present invention it is proposed to connect a designated network probe (hereinafter called "the Terminal Agent") to the data communication line. Alternatively, the terminals 01,02 etc., and the terminal agent may be connected to different data communication lines, or located at different local networks.

The general scheme of the terminal Agent operation is illustrated in Fig. 2.

The Terminal Agent is exposed to all data frames passing through the communication line. The data frames may contain information transferred

between the terminals or external data transmission to external sources such as Internet servers.

Let us further assume that the "Owner" of the data communication line, such as ISP or network of a private organization, is interested in monitoring in real time, the actual communication activities of a given terminal when surfing the internet.

The operation of the Terminal Agent is to first analyze the data frames for tracing TCP/IP data packets. As illustrated in Fig 3, the data analysis is processed according to the different protocol hierarchy (see RFC 0793 of the internet protocol), namely, first to analyze the local network protocol, filtering external data transmission ("gateway level "), then identifying internet protocol (IP) data frames, and finally detecting TCP("Transition Control Protocol") data packets of the "host level".

Upon analyzing the IP HEADER of the data packets, the IP addresses of the requesting terminal and of the message destination are identified. The owner of the communication line can easily relate the IP address to the users terminals. Therefore it is possible to filter out all other irrelevant data packets and restrict further processing to data transmission of one selected terminal (hereinafter called "the identified data packets").

The identified data packets are further analyzed according to the RFC 079 specification enabling full management and control of data communication ports.

According to known routines of managing TCP data communication ports, as processed by conventional browsers, e.g. the Internet Explorer, the terminal which operates the browser is the original source of all data transmission. For example let us assume that the terminal placed a request for YAHOO! home page, which request is delivered through the network to YAHOO! server. In response, the server sends an HTML data file containing all information of yahoo home web page components. Accordingly the browser sends new requests for receiving all components of the web-page by opening new communication "virtual" ports, where each port is used for transmitting different components of the same web-page. An "outsider" terminal, exposed to all data requests and respective responses is unable to differentiate between initial "primary" requests, e.g. requesting the complete YAHOO! home page and "secondary" requests for receiving the components thereof. For simulating the activity of the original browser by an "outsider" probe terminal it is essential to identify the primary requests as such.

Fig. 4 illustrates the process for differentiating the primary requests from the secondary requests. Primary requests are originated from different operations such as entering a new URL by the user, choosing a hyperlink, etc. Therefore, in order to detect same one must analyze the previous information transmitted to the same IP address. All new navigation components (addressing the browser to new location) of the web page received by the terminal are sorted according to their type, all embedded

objects, frames, etc., are marked as "false" components, while hyperlinks are marked as "true" components. All data is stored in the incoming buffer responses database for later use.

When identifying a request for a new connection according to TCP analysis, the request is examined according to the respective navigation components (RNC) in the incoming respond buffer. If the RNC is marked as "false" the request is ignored; if the RNC is marked as "true" the request is classified as primary; otherwise, if there is no RNC relating the said request, the connection type should be identified. If the connection is of an HTML type, or "post" type, it is classified as a primary request.

In order to view and monitor the activities of a terminal, all "original" browser activities must be reconstructed. For that purpose it is suggested to use a "virtual" browser. This virtual browser possesses all the capabilities of a "real" browser to download in real time web pages from the Internet. However its connection with the Internet is virtual in the sense that no actual data exchange with the Internet servers is preformed, but only simulating the activities of the original "real" browser.

The first function of the virtual browser is illustrated in Fig. 5. The browser is receiving all primary requests of the "real" browser. These primary requests and the respective primary responses from the Internet are analyzed and processed according to the conventional browser operation. However the outcome of secondary virtual requests (in conventional browser used to

complete the process of downloading web page components) are not transferred directly as usual through the Internet to the appropriate server but stored in a the virtual "secondary" requests buffer database

Although the virtual browser connection is not "real", all TCP protocol management of opening and controlling ports connection is processed by the terminal agent as if the connections are "real" ones.

The final process of simulating and presenting the web pages in the virtual browser is further illustrated in Fig. 6. All original secondary responses coming through the communication line are analyzed and recorded in the incoming responses buffer database. The virtual requests are compared to the respective secondary responses stored in the incoming responses buffer database, by the order of their arrival. If the respective secondary responses already exists in the buffer, these responses are transferred to the virtual browser, and processed (according to conventional browser operation) to present the visual picture of the respective web page components. As a result, the terminal agent is simulating in real time the exact process of downloading Internet web pages as it has been performed by the original terminal.

In case the respective responses do not appear in the incoming responses buffer database, activity of an original local cache is deduced. If the original local cache was not used with respect to said virtual request, it is suspend in the buffer database until the original secondary respective

responses arrive. Otherwise, if the real local cache was used relating to this respond, the local cache of the virtual browser is examined, and if respective secondary responses exist in the local cache, then the respective respond is transferred to the virtual browser and processed as described above. In case the respective responses do not exist in the virtual cache, either of the following alternatives may be applied. According to one, "passive" version of the terminal agents, no further action is taken to find the "missing" respond, and an "error" message will appear at the agent terminal instead of the web page component which appeared in the real terminal. According to this version, the simulation of the real terminal is not complete but the tapping activity is undetectable. According to another, "active" version, the terminal agent addresses the web page server to request the "missing" respond. Although this version enables the terminal agent to present more exact picture of the real terminal activities, it is traceable for more experienced terminal users, who are able to detect the tapping activity.

According to a further mode of implementation of the of the present invention, it is proposed to tap not only to related web page data packets, but to trace also related messages data packets e.g. e-mail or chats. To enables such tapping, the same method and principals as described above are applied at request for receiving and sending messages through the network other than requests for web pages. The process of analyzing such requests and the respective responses is more streamlined since there is no need to

check the cache memory activity, as by definition such information is always new.

Finally, it should be appreciated that the above-described embodiments are directed to Internet communication environment. However, the invention in its broad aspect is equally applicable to computerized network communication in general, such as satellite, cellular and others.

While the above description contains many specificities, these should not be construed as limitations on the scope of the invention, but rather as exemplification of the preferred embodiments. Those skilled in the art will envision other possible variations that are within its scope. Accordingly, the scope of the invention should be determined not by the embodiments illustrated, but by the appended claims and their legal equivalents.

WHAT IS CLAIMED IS:

1. A method of tracking a network communication line by network probe terminal ("terminal agent") simulating a browser("original browser") activity of a given terminal comprising the steps of:

I. Accessing the network communication line;

II. Tracing TCP/IP data packets routed through the communication line;

III. Selecting TCP/IP data packets relating to a given IP address ("identified data packets");

IV. Selecting from the identified data packets current requests for new connections ("original requests");

V. Selecting from the identified data packets current web page components indicating new addresses ("new navigation components");

VI. Dividing the new navigation components into two categories;

(f1) embedded objects or frames ("false new components");

(f2) hyperlinks ("true new components ");

VII. Dividing the original requests into "primary" or "secondary" requests according to the following criteria, respectively:

(g1) original requests matching true the new components, or original requests failing to match any new connection components and belonging to HTTP or POST type;

(g2) original requests matching the false components or original requests failing to match any new connection components and not belonging to HTTP or post type;

VIII. Selecting from the identified data packets, HTML data files relating to primary requests ("respective primary responses");

IX. Generating "virtual" secondary requests according to the respective secondary responses;

X. Selecting from the identified data packets responses relating to secondary virtual requests; ("respective secondary responses"); and

XI. Simulating web page presentation on the terminal agent according to the respective secondary responses.

2. The method of Claim 1 further comprising the steps of:

- Selecting virtual secondary requests currently not matching any respective original responses ("unanswered secondary requests ");

- Retrieving content form local cache of terminal agent ("virtual cache") if original browser local cache ("original cache") was used relating to unanswered secondary requests;

- Simulating web page presentation on agent terminal according to data relating to the unanswered secondary requests if the virtual cache contains such data; and

- Displaying error massages at the respective places on the agent terminal if the virtual cache does not contain data relating to the unanswered secondary requests;

3. The method of Claim 2 further comprising the step of:

- Addressing via the communication line to appropriate Internet server for receiving respective simulated responses relating to unanswered secondary requests if the virtual cache does not contain data, and displaying thereof.

4. The method of Claim 1 further comprising the steps of:

- Selecting from identified data packets data relating network messages e.g. e-mail ("messages data");

- Transforming the message data into text data file; and

- Displaying text data file on terminal agent.

5. The method of Claim 1 where in the net work is local-area network (LAN) and the terminal is connected to the extended communication line.

6. The method of Claim 1 wherein the communication line an is an external communication line e.g. telephone line, IZDN line, optical lines, etc.

7. The method of Claim 1 wherein the net work is a local-area network (LAN), and the terminal agent is situated in a location different from that of the given terminal.

8. The method of Claim 1 wherein the given IP address is identified by the communication line provider (ISP).

9. The method of Claim 1 wherein the given IP address is identified by the communication line owner.

10. The method of Claim 1 wherein the given IP address are addresses of web site visitors and identified by a web site owner.

11. A network probe terminal for tracking a network communication line and simulating a browser ("original browser") activity of a given terminal comprising of:

I. Connection means for accessing the network communication line;

II. Monitoring means for tracing TCP/IP data packets routed through the communication line;

III. First filtering module for selecting new connection requests ("original requests") and web page components indicating new

addresses ("new navigation components") out of TCP/IP data packets relating to a given IP address ("identified data packets");

IV. First Sorting means for dividing the new navigation components into two categories;

(f1) embedded objects or frames ("false new components");

(f2) hyperlinks ("true new components ");

V. Second Sorting means for dividing the original requests into "primary" or "secondary" requests according to the following criteria, respectively:

(g1) original requests matching the true new components, or original requests failing to match any new connection components and belonging to HTTP or POST type;

(g2) original requests matching the false components or original requests failing to match any new connection components and not belonging to HTTP or post type;

VI. Classifying module for selecting HTML data files relating to primary requests ("primary responses") from the identified data packets;

VII. Request generating module for creating "virtual" secondary requests according to the respective secondary responses;

VIII. Second filtering module for selecting responses relating to secondary virtual requests; ("secondary responses") from identified data packets; and

IX. Displaying means for simulating web page presentation on the terminal agent according to the secondary responses.

12. The network probe terminal of Claim 11 wherein the filtering module comprises means for selecting virtual secondary requests currently not matching any respective original responses ("unanswered secondary requests ");

13. The network probe terminal of Claim 11 further comprising cache module for activating terminal agent local cache in case the original browser local cache ("original cache") has been used in response to unanswered secondary requests;

14. The network probe terminal of Claim 13 further comprising a retrieval module for addressing via the communication line to appropriate Internet server and receiving respective simulated responses relating to unanswered secondary requests if the virtual local cache does not contain data.

15. The network probe terminal of Claim 11 further comprising an electronic message module for selecting identified data packets data relating to network messages e.g. e-mail ("messages data"), transforming

the message data into text data file, and displaying text data file on terminal.

16. The network probe terminal of Claim 11 wherein the given IP address is identified by the communication line provider (ISP).

17. The network probe terminal of Claim 11 wherein the given IP address is identified by the communication line owner.

18. The network probe terminal of Claim 11 wherein the given IP addresses are addresses of web site visitors and identified by the web site owner.

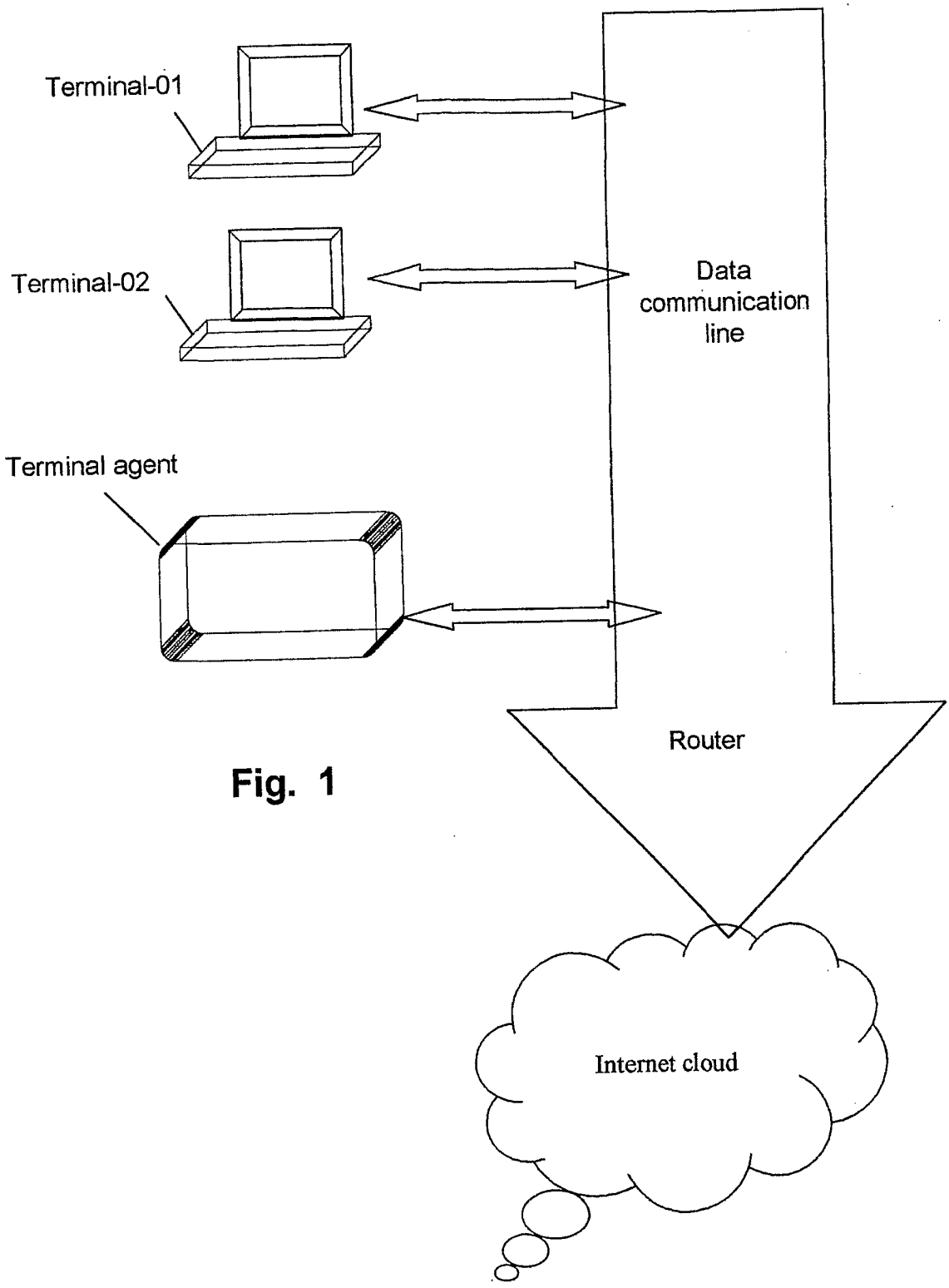

Terminal-01
Terminal-02
Terminal agent
Data communication line
Router
Internet cloud

Fig. 1

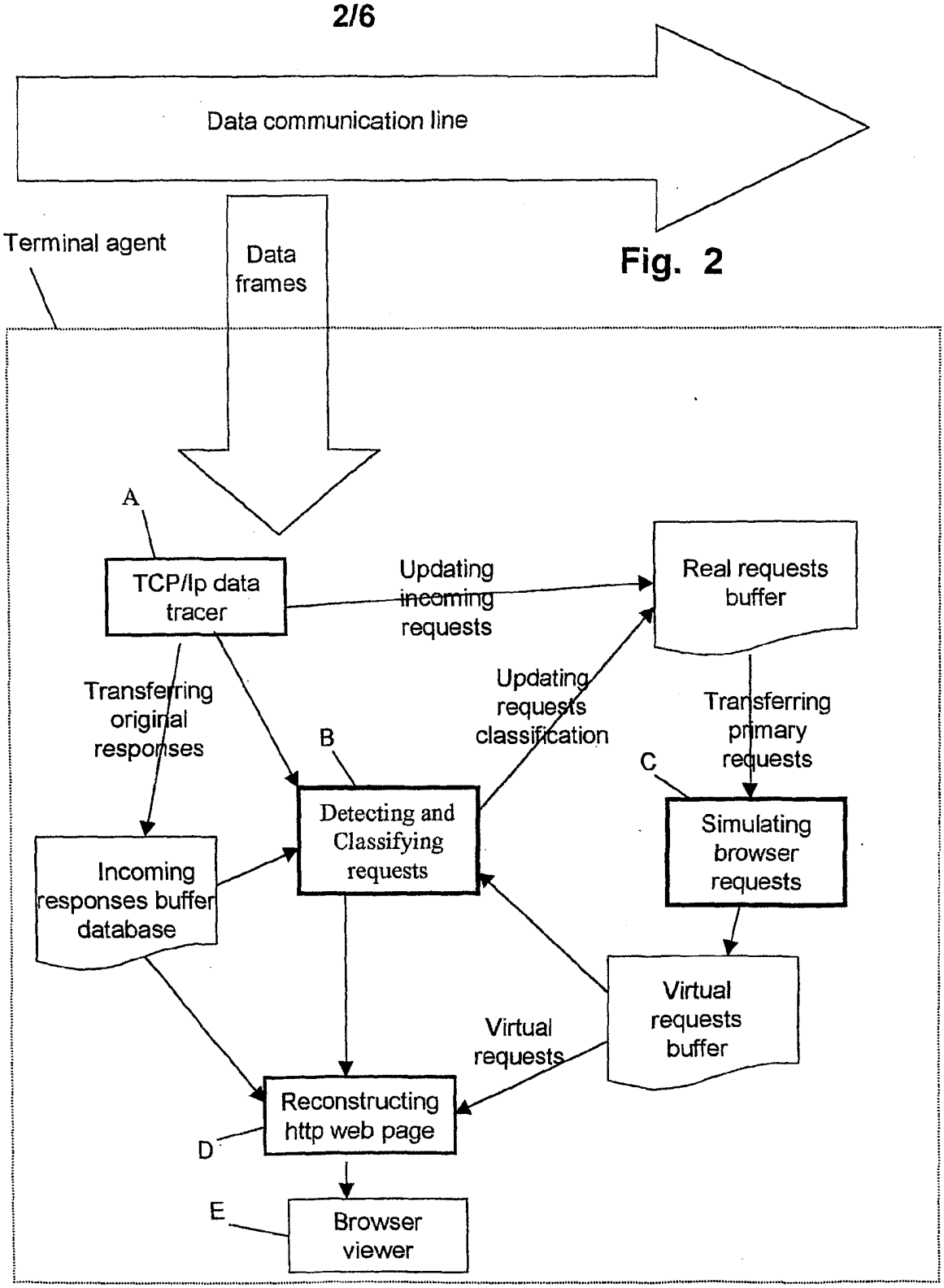

Data communication line
Terminal agent
Data frames
A
TCP/Ip data tracer
Updating incoming requests
Real requests buffer
Transferring original responses
Updating requests classification
Transferring primary requests
B
C
Detecting and Classifying requests
Simulating browser requests
Incoming responses buffer database
Virtual requests buffer
Virtual requests
Reconstructing http web page
D
E
Browser viewer

Fig. 2

A

Fig. 3

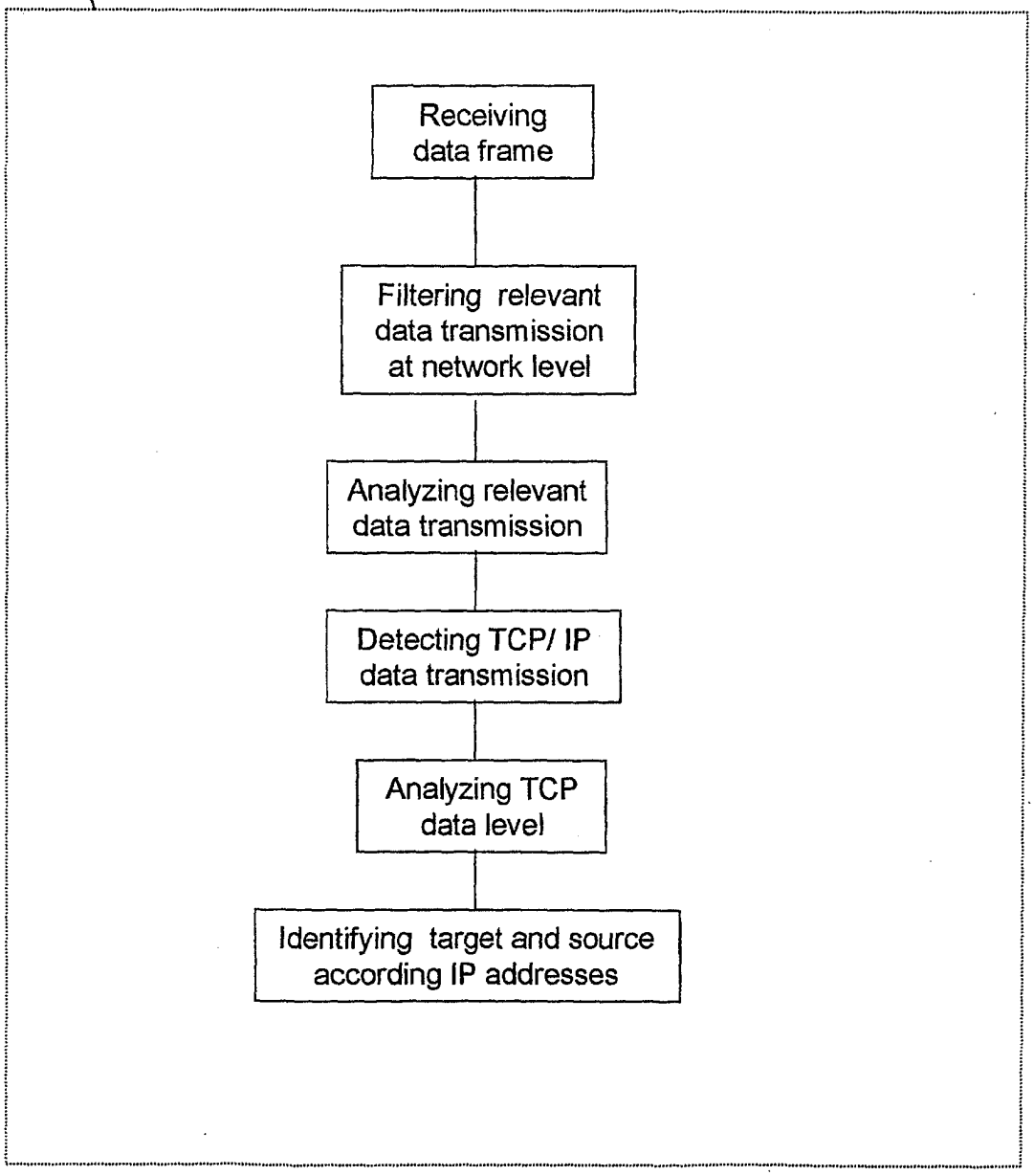

Receiving data frame
Filtering relevant data transmission at network level
Analyzing relevant data transmission
Detecting TCP/ IP data transmission
Analyzing TCP data level
Identifying target and source according IP addresses

Fig. 4

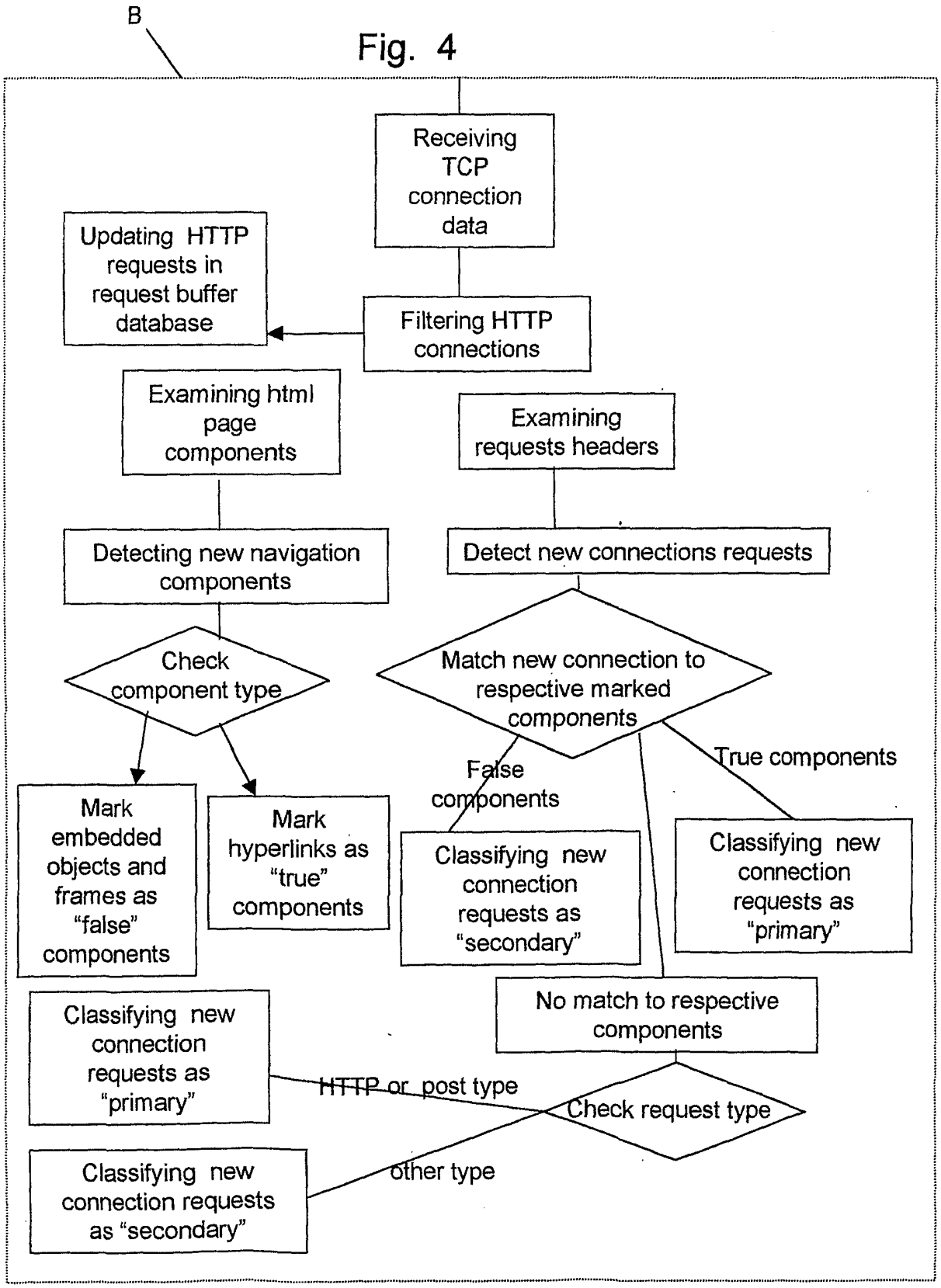

B
Receiving TCP connection data
Updating HTTP requests in request buffer database
Filtering HTTP connections
Examining html page components
Examining requests headers
Detecting new navigation components
Detect new connections requests
Check component type
Match new connection to respective marked components
Mark embedded objects and frames as "false" components
Mark hyperlinks as "true" components
False components
True components
Classifying new connection requests as "secondary"
Classifying new connection requests as "primary"
No match to respective components
Classifying new connection requests as "primary"
HTTP or post type
Check request type
other type
Classifying new connection requests as "secondary"

Fig. 5

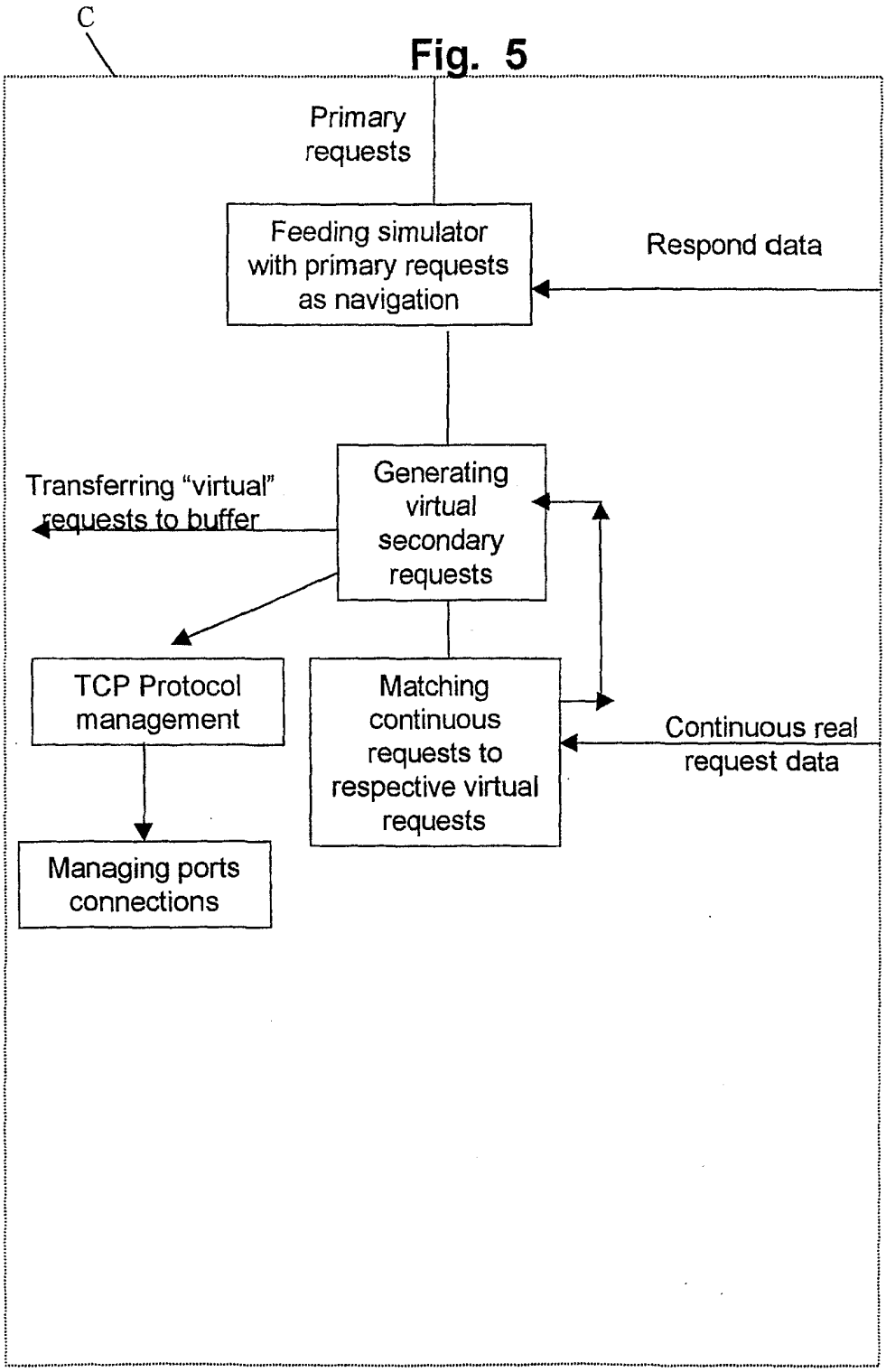
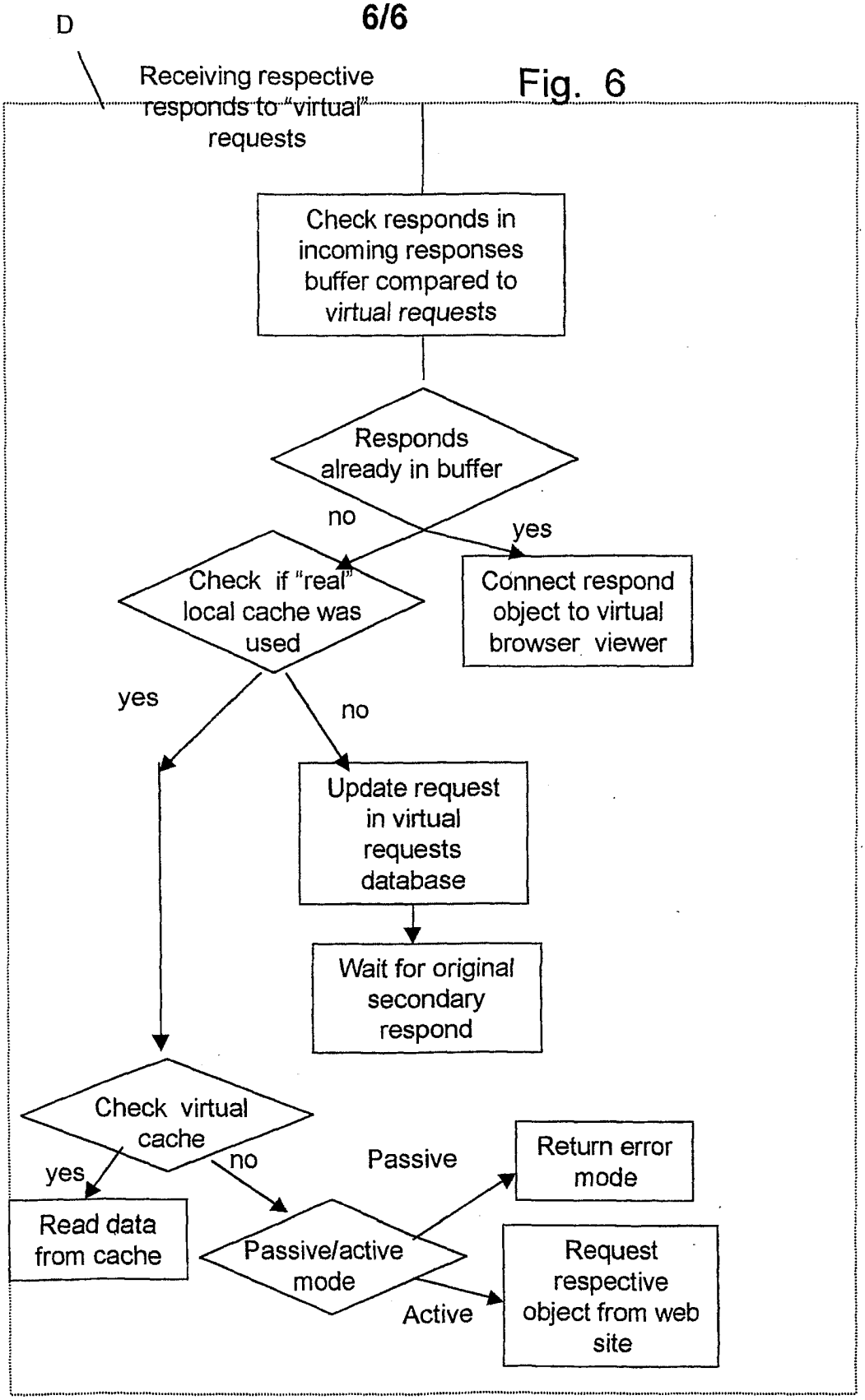

C
Primary requests
Feeding simulator with primary requests as navigation
Respond data
Generating virtual secondary requests
Transferring "virtual" requests to buffer
TCP Protocol management
Matching continuous requests to respective virtual requests
Continuous real request data
Managing ports connections

Fig. 6


D
Receiving respective responds to "virtual" requests
Check responds in incoming responses buffer compared to virtual requests
Responds already in buffer
no
yes
Check if "real" local cache was used
Connect respond object to virtual browser viewer
yes
no
Update request in virtual requests database
Wait for original secondary respond
Check virtual cache
yes
no
Passive
Return error mode
Read data from cache
Passive/active mode
Active
Request respective object from web site