(12) UK Patent (19) GB (11) 2513344 (13) B

(45) Date of B Publication 15.03.2017

(54) Title of the Invention: Communication system utilizing HTTP

(51) INT CL: ***H04L 29/06*** (2006.01) ***H04L 29/08*** (2006.01)

(21) Application No: **1307340.8**

(22) Date of Filing: **23.04.2013**

(60) Parent of Application No(s)
1412182.6 under section 15(9) of the Patents Act 1977

(43) Date of A Publication **29.10.2014**

(56) Documents Cited:

WO 2003/019390 A1	**FR 002869490 A1**
US 6892240 B1	**US 20100238831 A1**
US 20100042677 A1	**US 20080298376 A1**

(58) Field of Search:
As for published application 2513344 A viz:
INT CL **H04L**
Other: **EPODOC, INSPEC, WPI**
updated as appropriate

(72) Inventor(s):
Tuomas Mikael Kärkkäinen
Valtteri Hakkarainen
Ossi Kalevo

(73) Proprietor(s):
Gurulogic Microsystems Oy
Linnankatu 34, Turku 20100, Finland

(74) Agent and/or Address for Service:
Basck Ltd
16 Saxon Road, CAMBRIDGE, Cambridgeshire,
CB5 8HS, United Kingdom

GB 2513344 B

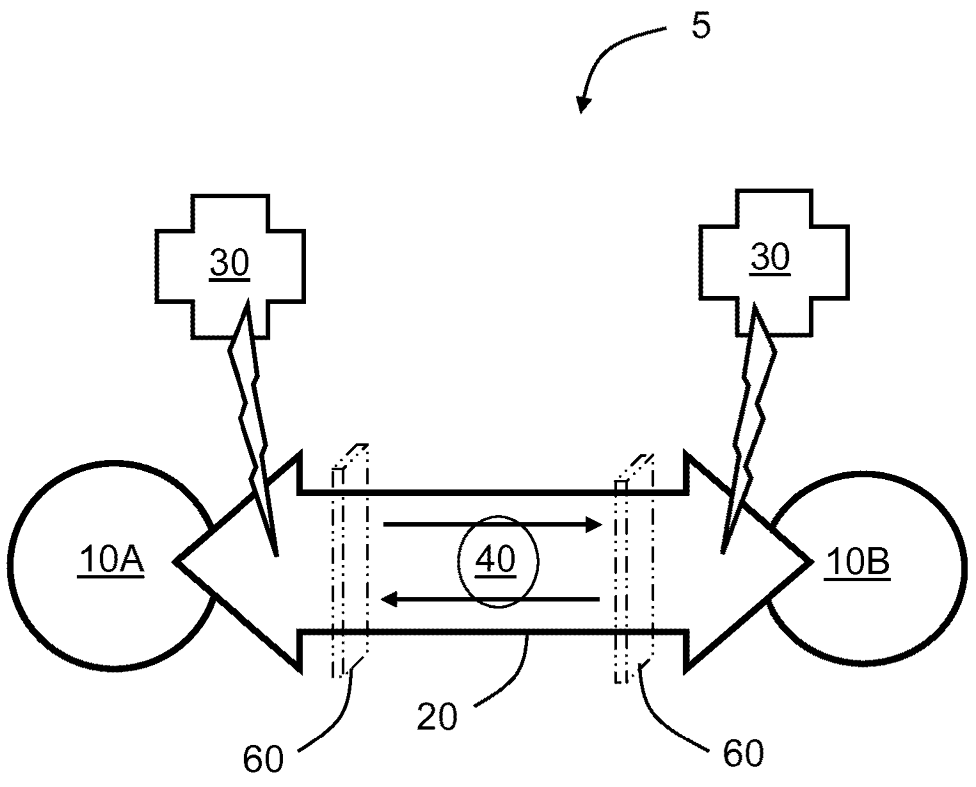

FIG. 1

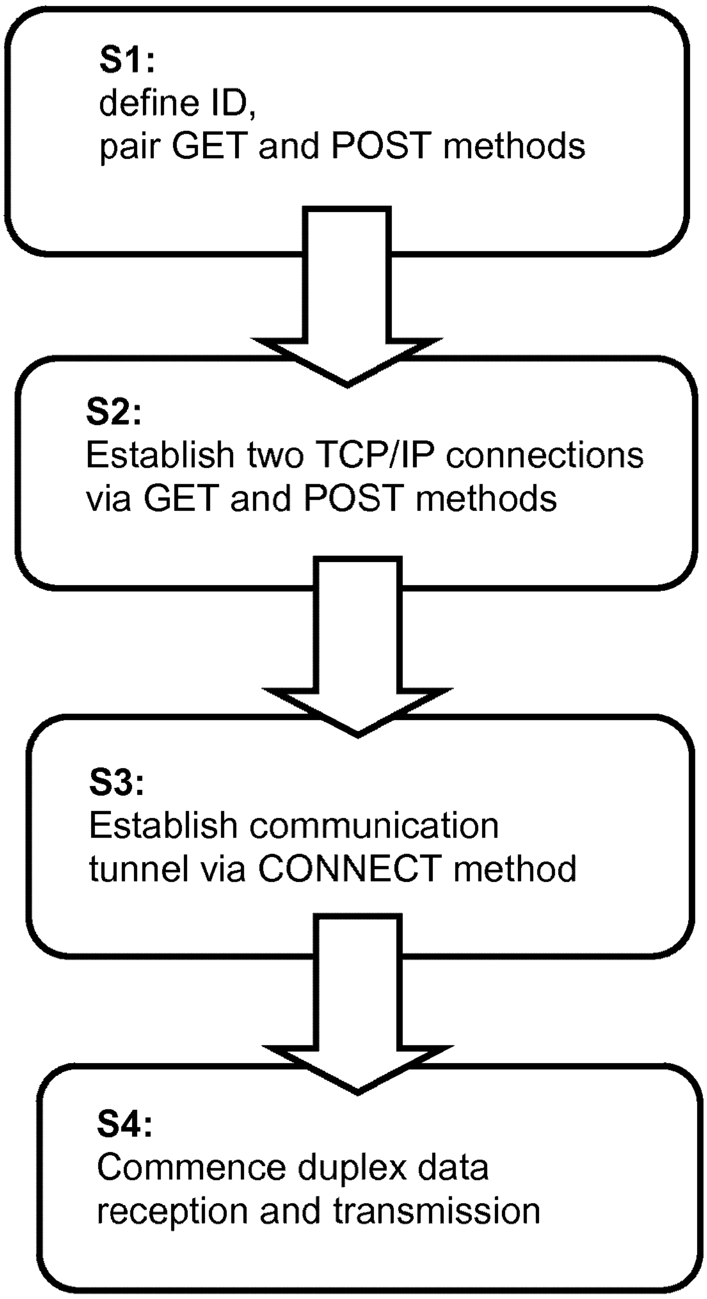
S1:
define ID,
pair GET and POST methods
S2:
Establish two TCP/IP connections via GET and POST methods
S3:
Establish communication tunnel via CONNECT method
S4:
Commence duplex data reception and transmission

FIG. 2

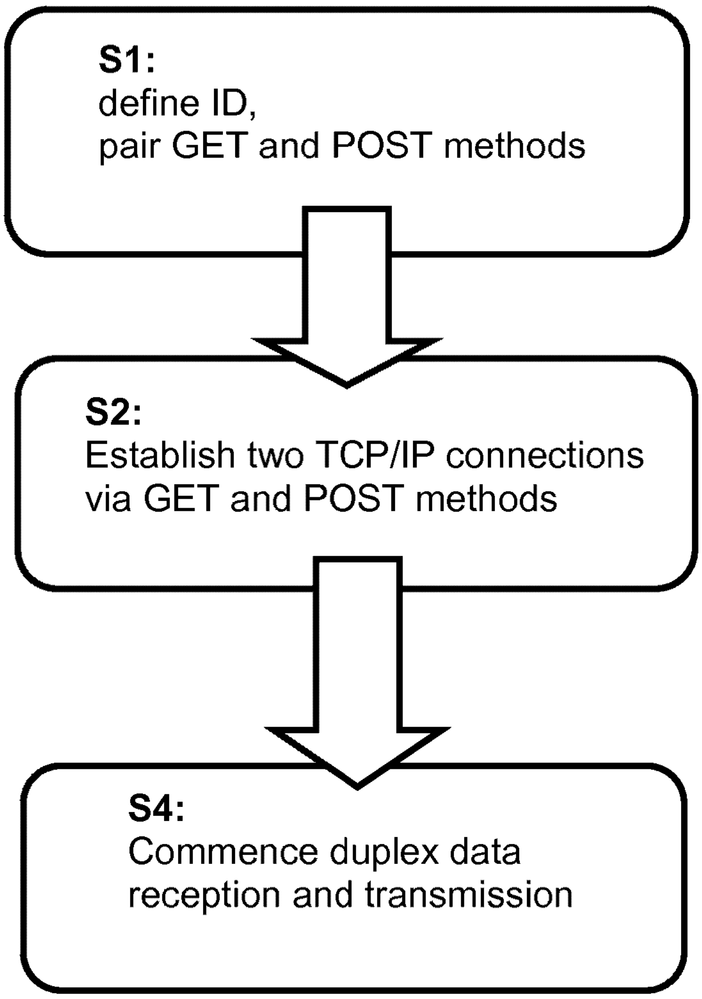
S1:
define ID,
pair GET and POST methods
S2:
Establish two TCP/IP connections
via GET and POST methods
S4:
Commence duplex data
reception and transmission

FIG. 3

COMMUNICATION SYSTEM UTILIZING HTTP

Technical Field

The present disclosure relates to communication systems, for example to communication systems which utilize Real-Time Hypertext Transfer Protocol (HTTP) for communicating various types of digital data, for example graphics data, image data, video data, audio data and similar. Moreover, the present disclosure is also concerned with methods of operating aforesaid communication systems for communicating various types of data. Furthermore, the present disclosure is also concerned with software products recorded on machine-readable data storage media, wherein the software products are executable upon computing hardware for implementing aforesaid methods.

Background

In overview, Hypertext Transfer Protocol (HTTP) is widely used for implementing the contemporary Internet. The Protocol is an application protocol for distributed, collaborative hypermedia information systems. In implementation, HTTP is a multi-linear set of objects which are operable to build a network using logical links to define the network; the links are often referred to as being “hyperlinks” which define a network relationship between nodes.

HTTP is operable to function as a request-response protocol, for example in a client-serving model as implemented for the Internet. In the model, a web browser is optionally used to implement a client, and a software application executing upon a server may host a web site. In operation, a given client submits a HTTP request message to the server, which responds by providing resources such as HTML files and other content, or performs data processing functions on behalf of the client, or even returns a response message to the client. The aforesaid web browser is susceptible to being implemented in various ways, for example as a user agent, as a web crawler or any other software executable upon computing hardware that accesses, consumes or displays Internet-derived data content.

HTTP is designed to permit immediate network elements to enable communications between clients and servers. High-traffic web-sites of the Internet often employ web cache servers that are operable to deliver content on behalf of upstream servers to improve response times for data and/or service delivery. Moreover, HTTP proxy servers at private network boundaries are beneficially used to facilitate communication for clients without a globally routable Internet address, namely by relaying messages via external servers.

HTTP resources are identified and located on a given network by using Uniform Resource Identifiers (URI's), also referred to as Uniform Resource Locators (URL's). Moreover, URI's and hyperlinks are expressed in Hypertext Markup Language (HTML) that is capable of forming webs of mutually interlinked hypertext documents.

An HTTP session is implemented by way of a sequence of network request-response transactions. For example, an HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server. An HTTP server listens for the client's request message and responds by sending back a status line, for example "HTTP/1.1 200 OK" together with an associated message. A body of this associated message is often the requested resource, although an error message may alternatively be returned.

HTTP defines methods, conveniently referred to as "verbs", for indicating a desired action to be performed in respect of an identified resource. The resource is, for example, a data file or an output from an executable object residing on one or more servers. Examples of HTTP methods, also known as HTTP "verbs", are provided in Table 1.

Table 1: HTTP methods (HTTP "verbs")

"Verb"	Details
GET	Requests a representation of a specified resource, wherein requests using "GET" should only retrieve data
HEAD	Requests a response which is identical to that obtainable from GET, but devoid of any response body; "HEAD" is often employed for retrieving meta-data in an efficient manner
POST	Requests that a given server accepts an entity enclosed in the request as a new sub-ordinate of a given web resource identified by a URL
PUT	Requests that an enclosed entity be stored in respect of a supplied

	URI (URL). If the URI refers to an already existing resource, that resource is modified.
DELETE	Requests deletion of a specified resource
TRACE	Results in a received request to be echoed back to the given client
OPTIONS	Returns HTTP methods supported by a server associated with a given URL
CONNECT	Converts a requested connection to a transparent TCP/IP tunnel, for example for facilitating TLS and SSL-encrypted communication (HTTPs) through an unencrypted HTTP proxy as aforementioned; by default, an HTTP connection is unencrypted, whereas an HTTPS connection is encrypted.
PATCH	Requests application of partial modifications to a given resource

Thus, a principal transfer protocol employed by contemporary web browsers is aforesaid HTTP; several associated "ecosystems", and software that they utilize, in particular browser software applications, are not able to function without using HTTP. As aforementioned, HTTP is based upon requests, see Table 1, that are transmitted and, on response to these requests, HTML pages or binary data such as images or audio streams/files are commonly served in response to receiving the requests.

On account of the complexity of the Internet, Internet communication delays, namely "latency", can arise in operation. Such delays can cause problems in demanding data exchange situations, for example when two-way (full-duplex) communication is desired, where real-time response is desired, for example transfer and reception of video images and/or audio with very little delay. Bi-directional communication via the Internet is known from Voice-over-Internet-Protocol (VoIP) and also from Internet-based video conferencing, for example as contemporarily provided using Skype software and similar; "Skype" is a registered trademark.

It is known to employ protocols known as "WebSockets", as described at a web-site http://tools.ietf.org/html/rfc6455, for addressing specific types of communication needs. Following communication properties are thereby capable of being achieved:

(i) a WebSocket is employed inside an HTTP/HTTPS tunnel; in such a case, firewalls have already been opened for ports 80/443, because they are contemporarily commonly utilized on web browsers; and

(ii) a WebSocket is utilized in a full-duplex connection mode, wherein only one TCP connection is able to communicate both ways in real-time, namely it is able to transmit and receive data with one connection by changing the direction of data delivery.

However, such WebSockets can be port-dependent which represents an undesirable limitation.

Summary

The present disclosure seeks to provide a communication system which is capable of providing two-way data communication via an HTTP communication network in an improved manner.

Moreover, the present disclosure seeks to provide an improved method of operating a communication system for providing two-way data communication via an HTTP communication network.

According to a first aspect of the present invention, there is provided a communication system as claimed in appended claim 1: there is provided a communication system which is operable to support HTTP-based communication, wherein the communication system is operable to establish a two-way real-time communication link between two nodes of the system by employing a combination of GET and POST methods associated with HTTP, and

wherein data exchange via the communication link is implemented in a chunked manner and/or as a series of multipart data blocks; and

a maximum segment size (MSS) for data chunks and/or multipart data blocks communicated through the communication link is optimized as a function of a communication network capability supporting the communication link; and

the communication link includes a reception connection and a transmission connection for providing the two-way communication, and wherein the connections are maintained open until an empty chunk and/or an empty multipart data block is received.

The communication system is of advantage in that it is capable of providing real-time two-way communication with reduced latency.

Optionally, in the communication system, the two-way communication link is TCP/IP and/or UDP tunnelled by employing a CONNECT method associated with HTTP.

Optionally, the CONNECT method is capable of being used in three different types of scenario:

(i) a connection is tunneled into a target; this is beneficially a default scenario;

(ii) a connection is tunneled via a local host to a target, resulting in data being transferred from a transmission process in the local service to a forwarding proxy process, from within the data is transmitted to the target; such an approach is beneficial because it is capable of preventing anti-virus software from analyzing the data and inadvertently blocking or otherwise interfering with the data;

(iii) a connection is tunneled into a forwarding proxy server which then redirects the data to its target; such an approach is beneficial to employ in load-balancing systems, namely in systems wherein a network load caused by clients is distributed optimally to the target. For example, it is faster to transmit data in a backbone network than via direct connection.

Optionally, in the communication system, the communication link is operable to employ encryption of data communicated therethrough.

Optionally, in the communication system, the communication link is operable to provide communication of at least one of: graphics data, image data, video data, audio data, unstructured data.

According to a second aspect of the disclosure, there is provided a method of establishing a communication link via a communication system which is operable to support HTTP-based communication, wherein the method includes:

(a) using the communication system to establish a two-way real-time communication link between two nodes of the system by employing a combination of GET and POST methods associated with HTTP;

(b) exchanging data via the communication link in a chunked manner and/or as a series of multipart data blocks; and

(c) optimizing a maximum segment size (MSS) for data chunks and/or multipart data blocks communicated through the communication link as a function of a communication network capability supporting the communication link;

wherein the communication link includes a reception connection and a transmission connection for providing the two-way communication, and wherein the connections are maintained open until an empty chunk and/or an empty multipart data block is received.

Optionally, the method includes TCP/IP and/or UDP tunnelling the two way communication link by employing a CONNECT method associated with HTTP.

19 07 16

Optionally, in the method, the communication link is operable to employ encryption of data communicated therethrough.

Optionally, in the method, the communication link is operable to provide communication of at least one of: graphics data, image data, video data, audio data, unstructured data.

According to a third aspect of the disclosure, there is provided a software product recorded on machine-readable data storage media, wherein the software product is executable upon computing hardware for implementing the method pursuant to the second aspect of the disclosure.

Optionally, the software product is expressed in HTTP and is executable upon a server of a communication network operating according to HTTP.

The present invention is of advantage in that the communication system is capable of providing two-way, full-duplex communication, either unencrypted or encrypted, by utilizing known HTTP transfer protocol in such a way that extra configurations are not necessary in software or hardware firewalls and/or in anti-virus software applications executing in the communication system.

Moreover, the present invention is of advantage in that it improves the functionality and reliability of communication applications, and thus simplifies technical maintenance issues associated with the system, for example data security settings.

It will be appreciated that features of the invention are susceptible to being combined in various combinations without departing from the scope of the invention as defined by the appended claims.

Description of the diagrams

Embodiments of the present disclosure will now be described, by way of example only, with reference to the following diagrams wherein:

FIG. 1	is an illustration of a communication network operable to employ HTTP;
FIG. 2	is an illustration of a set of steps of a method of the disclosure; and
FIG. 3	is an illustration of an alternative set of steps of a method of the disclosure.

In the accompanying diagrams, an underlined number is employed to represent an item over which the underlined number is positioned or an item to which the underlined number is adjacent. A non-underlined number relates to an item identified by a line linking the non-underlined number to the item. When a number is non-underlined and accompanied by an associated arrow, the non-underlined number is used to identify a general item at which the arrow is pointing.

Description of embodiments of the disclosure

In overview, with reference to FIG. 1, there is hereinafter described a system, a portion of which is indicated generally by **5**, and associated method, which is capable of deducing delays, namely "latency", in respect of HTTP for two-way real-time communication in a manner that description of HTTP employed conforms to standards such as RFC2621, RFC2068 and RFC1945. Normally, HTTP is not designed to enable real-time two-way communication between first and second nodes **10A**, **10B**, wherein a given client is able simultaneously to transmit real-time data and to receive real-time in such a manner that:

(i) a communication connection **20** employed between the two nodes **10A**, **10B** is operable to support the two-way communication in an encrypted format;

(ii) virus protection software **30** does not interfere with contents **40** being transmitted and received via the communication connection **20**;

(iii) firewalls **60** are not able to prevent network traffic unless a general blockage of Internet traffic, namely “WWW traffic”, is blocked, for example in a situation of a banking connection employed for secure financial transactions; and

(iv) network devices, for example bridges and routers, are not able to analyze and interfere with data to be communicated via the communication connection **20**.

Embodiments of the present disclosure are capable of addressing functionalities (i) to (iv) by employing following features:

(a) two mutually different types of GET and POST methods are used, see Table 1 above, wherein the GET method constructs a reception connection via the communication connection **20**, and the POST method constructs a transmission connection via the communication connection **20**;

(b) both connections are tunnelled using the CONNECT method as employed in contemporary HTTP; and

(c) a form of “chunked” or multi-part transfer encoding is employed, as will be elucidated in more detail below.

Conventionally, HTTP is used for Internet sessions, wherein the GET and POST methods are employed in a mutually independent manner. For example, the GET method is used for requesting HTML content from a web-server which is operable to function as a host for a web-browser client, wherein connections for the GET method remain open until all response data is delivered from the host to the client. Moreover, a connection procedure is employed which is the same as the POST method, see Table 1, except that data is delivered from the client to the host.

In embodiments described hereinafter, communication is executed in such a manner that a given socket is used in a half-duplex manner, which distinguishes the embodiments from known approaches, for example aforesaid WebSockets. In the embodiments, transmission and/or reception of data is more efficient than in a full-duplex connection, because network interface cards do not need to switch their input/output (I/O) states between reception and transmission. Such switching employed in known technical art consumes system resources and correspondingly decreases potential communication speed.

In the embodiments described hereinafter, a socket is utilized after an initialisation of HTTP GET and POST methods only, either in a reception mode or in a transmission mode. In consequence, a network adapter used only needs to operate in a half-duplex state only, thereby saving network infrastructure and device resources, because the connection operates solely in either a transmitting mode or a reception mode after negotiated HTTP GET and/or POST method headers until a finish of the connection occurs. Moreover, other benefits also arise, for example firewalls and routers, namely hubs and switches, receive less switching load and thus will not break as fast as known contemporary full-duplex communication approaches that use only one full-duplex connection. Thus, embodiments described hereinafter are much more resource-efficient than aforesaid WebSockets, for example.

Aforementioned known WebSockets can be easily analysed by firewalls as belonging to an unidentified connection type and thus disconnected, thereby preventing or restricting their usage, irrespective of whether or not an associated connection is tunnelled or not. In embodiments described hereinafter, a GET or POST connection functions according to HTTP protocol, and thus firewalls cannot restrict or prevent communication utilizing these methods.

In the embodiments as described hereinafter, UDP protocol which is estimated to be substantially three times faster than TCP, is beneficially employed,. Optionally, the embodiments can use peer-to-peer (P2P) connections, which allow communication to be achieved at application level.

Embodiments described herewith are differentiated from known HTTP implementations, in that known HTTP implementations are devoid of any link between GET and POST methods; in contradistinction, embodiments described herein employ GET and POST methods merged together in a novel manner for providing a real-time full-duplex data communication. The mentioned full-duplex data communication is implemented by using one reception connection and one transmission connection. One reception connection or one transmission connection can use one half-duplex connection mode or one full-duplex connection mode.

Although embodiments will be described below based upon Transport Control Protocol (TCP), it will be appreciated that User Datagram Protocol (UDP) can be employed as an alternative. Although both the UDP and TCP rely on an underlying Internet Protocol (IP), and both a UDP datagram and a TCP segment are transmitted in an IP packet, the UDP is distinguished in that it is a connectionless protocol that makes it possible to achieve peer-to-peer communications between applications, not only inside a local area network (LAN), but also in the outer Internet, by using a network address translation (NAT) traversal technique. By employing such an approach, a need to transfer data via servers in the system **5** can be avoided, resulting in considerable communication network capacity being saved. An additional benefit resulting from using UDP in the system 5 is that it is substantially three times more efficient in its use of network communication capacity than TCP, because UDP is not a controlled protocol. Moreover, the MSS capacity measured in bytes in both IPv4 and IPv6 communication networks, for example used for implementing the system **5**, is larger, because UDP headers are smaller than corresponding TCP headers.

Although use of TCP for both GET and POST connections will be described in the following, it will be appreciated that, optionally, only one of these connections uses TCP and the other of these connections uses UDP. Moreover, it will also be appreciated that both the GET and POST connections can utilize UDP.

It will be appreciated that the data in the transmitting or receiving end can also change from the circuit switched to IP-based data and correspondingly from IP-based to circuit switched data, without departing from the scope of the invention.

In a first example embodiment, a series of steps are performed as follows with reference to FIG. 2:

<u>STEP 1 (S1):</u> a client to a data connection generates a unique stream identification (ID), wherein the ID is employed to pair GET and POST methods together, so that a

server employed to implement the data connection is aware that the pair of GET and POST methods belong to the same client. The ID employed will be elucidated in greater detail later. However, it will be appreciated that GET and POST methods do not limit the present invention when the unique stream identification (ID) is used to combine transmission and reception connections. Even though the principal purpose of the Stream ID is to bind the transmission and reception connections of the client together at the server, it can simultaneously be used also for authenticating and identifying the client. This means that the server can then discard harmful, erroneous and/or unidentified connections before their processing continues. Such functionality makes it possible to protect the server and to reduce/prune the server load caused by unidentified connection requests and unnecessary computing. In other words, this enables the system to conserve resources, which provides a benefit of saving energy and decreasing the number of servers that are needed in the server facilities, especially in load balancing systems.

19 07 16

STEP 2 (S2): the client then establishes two TCP/IP connections to the server, for example at its default port "80", after which the client transmits a header associated with a CONNECT method. In operation, the CONNECT method converts the requested data connection into a transparent TCP/IP tunnel, for example usually to facilitate TLS and SSL-encrypted communication (HTTP) through an unencrypted proxy as aforementioned.

When implementing the STEPS 1 and 2, various forms of encryption are optionally employed, for example SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2 or similar types of encryption. However, the aforesaid tunnel is beneficially transparent for ensuring secure communication between different "ecosystems". Moreover, it is also beneficial to employ hardware which is protected against malicious attacks or interference. Such a transparent tunnel connection as employed for implementing embodiments of the disclosure is capable of preventing hacker, hostile software, anti-virus software, firewall software or other devices and/or software that are operable to monitor and analyze data traffic from interfering with data that is communicated via the tunnel connection.

<u>STEP 3 (S3):</u> depending upon the receiving or transmitting connection employed for the communication tunnel, the header of the GET method or the POST method continues to be transmitted and received. The header contains necessary information for a given communication session provided by the communication tunnel. Moreover, the header beneficially employs a convention form of data structure, although the header includes following parameters:

(i) the stream ID kind of information for bonded/linked connections; and

(ii) the transfer encoding as chunked or multi-part format.

Information included in the header ensures that transfer and reception of data occurs as individual data blocks. Beneficially, a Maximum Segment Size (MSS) of the data is optimized to a capability of a network supporting the communication tunnel, taking into consideration an amount of bytes used for the chunked or multi-part header, so that bytes are not lost when transferring and receiving data; a reliable and secure data exchange is thereby provided.

19 07 16

<u>STEP 4 (S4)</u>: once the HTTP request header has been transmitted, and a corresponding successful response has been received from the server, duplex data reception and transmission are then commenced. There has thereby been successfully made two connections with the server, namely a reception connection and a transmission connection; these connections are maintained in an open state until an empty data chunk or an empty multi-part data block is received.

Two example embodiments will next be elucidated by way of HTTP code.

<u>Example 1</u>: there is provided HTTP code which is operable when executed to create a simple tunnelled reception connection between the client and the server, wherein a peer with an IP address *192.168.0.101* connects to a host with an IP address *192.168.0.100*. Use of both “GET” and “CONNECT” methods in the HTTP code is to be found, together with chunked transfer-coding being specified:

```
<connect>
<send> CONNECT 192.168.0.100:80 HTTP/1.0 \r\n
<send> Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n
<send> \r\n
<send> GET /readstream? streamid=12345&param1=value1&param2=value2 HTTP/1.1 \r\n
<send> Host: 192.168.0.100 \r\n
<send> Transfer-Coding: chunked \r\n
<send> User-Agent : Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n
<send> \r\n
<recv> HTTP/1.1 200 OK \r\n
<recv> 5AD\r\n
<recv> 1453 bytes of data...  \r\n
<recv> 5AD\r\n

<recv> 1453 bytes of data...  \r\n
...
<recv> 5AD\r\n
<recv> 1453 bytes of data...  \r\n
<recv> 0 \r\n
<disconnect from 192.168.0.100>
```

Example 2: there is provided HTTP code which is operable when executed to create a simple tunnelled transmission connection between the client and the server, wherein a peer with an IP address *192.168.0.101* is connected with the host that has a corresponding IP address *192.168.0.100*. Use of both "POST" and "CONNECT" methods in the HTTP code is to be found, together with chunked transfer-coding being specified:

```
<connect to 192.168.0.100>
<send> CONNECT 192.168.0.100:80 HTTP/1.0 \r\n
<send> Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n
<send> \r\n
<send> POST /writestream?streamid=12345&param1=value1&param2=value2 HTTP/1.1 \r\n
<send> Host: 192.168.0.100 \r\n
<send> Transfer-Coding: chunked \r\n
<send> User-Agent : Mozilla/5.0 (Windows NT 5.0) Gurulogic \r\n
<send> \r\n
<send> 5AD\r\n
<send> 1453 bytes of data...  \r\n
<send> 5AD\r\n
<send> 1453 bytes of data...  \r\n
...
<send> 5AD\r\n
<send> 1453 bytes of data...  \r\n
<send> 0 \r\n
<recv> HTTP/1.1 200 OK \r\n
```

In these two Examples 1 and 2, it is assumed that the MSS is 1460 bytes, so actually the data size for an optimized chunk is 1453 bytes. An optimized chunk size is calculated in the system by using a formula as given in Equation 1 (Eq. 1):

MSS = (beginning of chuck header) – (end of chunk header) Eq. 1

The beginning of the chunk header consists of the length of the actual chunk data, for example in hexadecimal notation, and of the end of one or more line characters, which are usually both Carriage Return (CR) and Line Feed (Lf). The end of the chunk is similar to the end of line characters, which complete the chunk.

Referring next to FIG. 2, it will be appreciated that the STEP 3 (S3), namely establishing a connection tunnel by utilizing the CONNECT method, is optionally omitted as provided in FIG. 3. The connection tunnel is omitted when there is not a requirement for the tunnel. Thus, when a communication is not utilized, only STEPS 1, 2 and 4 are employed. Moreover, in respect of FIG. 2, it is also to be appreciated that the connection tunnel can be constructed only for the GET connection or the POST connection, namely an asymmetrical tunnel communication arrangement between a plurality of nodes; optionally, the communication tunnel is used only for GET or POST connections.

Modifications to embodiments described in the foregoing are possible without departing from the scope of the invention as defined by the accompanying claims. Expressions such as “including”, “comprising”, “incorporating”, “consisting of”, “have”, “is” used to describe and claim the present invention are intended to be construed in a non-exclusive manner, namely allowing for items, components or elements not explicitly described also to be present. Reference to the singular is also to be construed to relate to the plural. Numerals included within parentheses in the accompanying claims are intended to assist understanding of the claims and should not be construed in any way to limit subject matter claimed by these claims.

CLAIMS

We claim:

1. A communication system which is operable to support HTTP-based communication, wherein the communication system is operable to establish a two-way real-time communication link between two nodes of the system by employing a combination of GET and POST methods associated with HTTP, and

wherein data exchange via the communication link is implemented in a chunked manner and/or as a series of multipart data blocks; and

a maximum segment size (MSS) for data chunks and/or multipart data blocks communicated through the communication link is optimized as a function of a communication network capability supporting the communication link; and

the communication link includes a reception connection and a transmission connection for providing the two-way communication, and wherein the connections are maintained open until an empty chunk and/or an empty multipart data block is received.

2. The communication system as claimed in claim 1, wherein the two-way communication link is TCP/IP and/or UDP tunnelled by employing a CONNECT method associated with HTTP.

3. The communication system as claimed in claim 1, wherein the communication link is operable to employ encryption of data communicated therethrough.

4. The communication system as claimed in claim 1, wherein the communication link is operable to provide communication of at least one of: graphics data, image data, video data, audio data, text data, unstructured data.

5. A method of establishing a communication link via a communication system which is operable to support HTTP-based communication, wherein the method includes:

(a) using the communication system to establish a two-way real-time communication link between two nodes of the system by employing a combination of GET and POST methods associated with HTTP;

(b) exchanging data via the communication link in a chunked manner and/or as a series of multipart data blocks; and

(c) optimizing a maximum segment size (MSS) for data chunks and/or multipart data blocks communicated through the communication link as a function of a communication network capability supporting the communication link;

wherein the communication link includes a reception connection and a transmission connection for providing the two-way communication, and wherein the connections are maintained open until an empty chunk and/or an empty multipart data block is received.

6. The method as claimed in claim 5, wherein the method includes TCP/IP and/or UDP tunnelling the two way communication link by employing a CONNECT method associated with HTTP.

7. The method as claimed in claim 5, wherein the communication link is operable to employ encryption of data communicated therethrough.

8. The method as claimed in claim 5, wherein the communication link is operable to provide communication of at least one of: graphics data, image data, video data, audio data, text data, unstructured data.

9. A software product recorded on machine-readable data storage media, wherein the software product is executable upon computing hardware for implementing the method as claimed in claim 5.

10. The software product as claimed in claim 9, wherein the software product is expressed in HTTP and is executable upon a server of a communication network operating according to HTTP.