

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 February 2006 (02.02.2006)

PCT

(10) International Publication Number
WO 2006/011987 A2

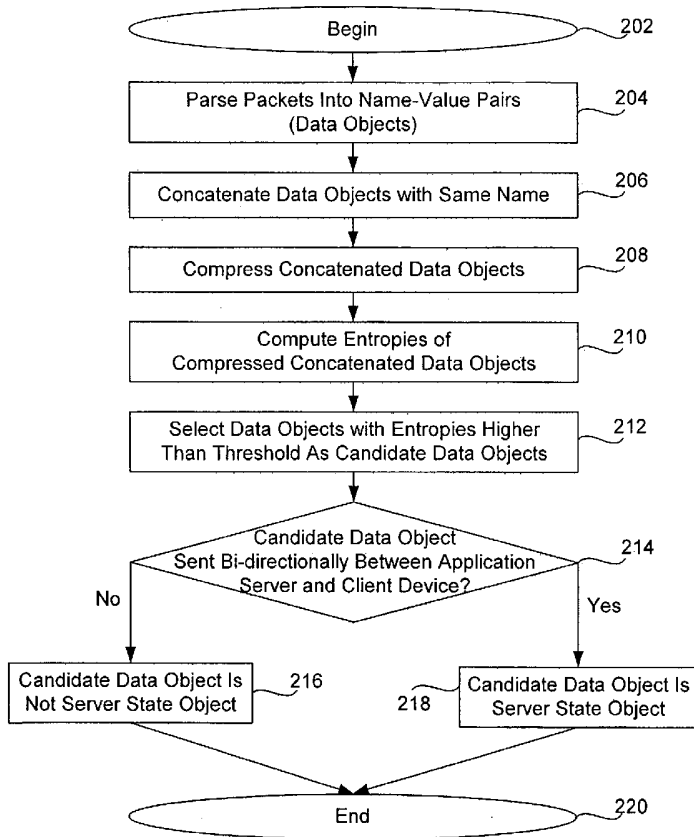
- (51) International Patent Classification:
G06F 15/16 (2006.01)
- (21) International Application Number:
PCT/US2005/020919
- (22) International Filing Date: 13 June 2005 (13.06.2005)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
10/877,362 25 June 2004 (25.06.2004) US
- (71) Applicant (for all designated States except US): **TEROS, INC.** [US/US]; 328 Gibraltar Drive, Sunnyvale, CA 94089 (US).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): **CHAUHAN, Abhishek** [IN/US]; 107 Rio Vis, Los Gatos, CA 95032 (US).
- (74) Agents: **SACHS Robert, R.** et al.; Fenwick & West LLP, Silicon Valley Center, 801 California Street, Mountain View, CA 94041 (US).

- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
- (84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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

[Continued on next page]

(54) Title: INTERFERRING SERVER STATE IN A STATELESS COMMUNICATION PROTOCOL



(57) Abstract: Server state objects are identified by an intermediate server among packets transmitted between an application server and a client device on a network based upon a stateless communication protocol, by monitoring and analyzing the packets transmitted between the application server and the client device. The packets are parsed into a plurality of namevalue pairs. The entropy of the name-value pairs having a same name field is computed, and candidate data objects that are likely to be server state objects are selected based upon the computed entropy. Candidate data objects that were transmitted bi-directionally between the application server and the client device are identified as server state objects.

WO 2006/011987 A2



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.

INFERRING SERVER STATE IN A STATELESS COMMUNICATION PROTOCOL

INVENTOR

5

Abhishek Chauhan

TECHNICAL FIELD

[0001] The present invention relates generally to a stateless communication protocol, and
10 more specifically, to identifying server state objects among packets transmitted between a server
and a client device under a stateless communication protocol such as XML (Extensive Markup
Language) or SOAP (Simple Object Access Protocol).

BACKGROUND OF THE INVENTION

15

[0002] Stateless communication protocols such as HTTP (Hypertext Transfer Protocol),
XML, or SOAP execute each command independently, without maintaining any state
information of previous commands. As such, in a strictly stateless protocol, a server will not
store the identity of a client device that previously communicated with the server, and thus will
20 not process a current transaction of the client in light of previous transactions.

[0003] Server state objects such as cookies or hidden fields are thus commonly used to
identify a client device communicating with a server using a stateless communication protocol.
For example, a cookie is a server state object that allows the server web site to store its own
information about the client device on the client device so that the server web site can later
25 determine the identity of the client device. Thus, a server can determine that a client device is
has previous communicated with the server by examining the server state objects, and allow the
client device to transact with the server based upon the established identity without additional
efforts to identify the client device.

[0004] Since server state objects identify a client device to a server and allows the client
30 device to communicate with the server based upon the determined identity, the server should

have a way to protect itself from fraudulent attempts to assume a certain identity of the client device. Thus, server state objects are typically immutable under stateless communication protocols in order to prevent the server from such fraudulent attempts. That is, when the server sends a server state object to the client device, the server expects the same server state object to come back without being changed or tampered on the client device.

[0005] Under a structured, well-defined communication protocol such as HTTP, an intermediate server, such as a firewall, gateway, or application proxy, may easily identify a server state object among the packets transmitted between an application server and the client device, because the packets containing the server state objects are configured to include certain fields that identify such server state objects. For example, a cookie includes a string such as <set-cookie:> and a hidden field includes a string such as <input type = hidden ...>, which can be readily identified by the intermediate server.

[0006] However, in less structured stateless communication protocols such as XML or SOAP, an intermediate server cannot readily identify server state objects, because these protocols are designed to allow application designers to define their own objects arbitrarily. Intermediate servers, such as firewalls, typically do not have access to the protocol definition given in the application server by the application designers, and thus server state objects arbitrarily defined by the application designers cannot be readily identified by the intermediate server. For example, in one application server, a server state could be held in an “ACCOUNT NUMBER” object while in another application the same server state could be held in a “CLIENT ID” object. The application server would have access to these protocol definitions and thus can identify these server state objects. However, the intermediate server (firewall or other filter) will not have the protocol/application definition of all applications behind the intermediate server.

[0007] Therefore, there is a need for identifying, by an intermediate server, server state objects transmitted between a server and a client device under a less structured stateless communication protocol such as XML or SOAP that does not have a predefined way of identifying the server state objects.

SUMMARY OF THE INVENTION

[0008] The present invention provides for identifying server state objects among packets transmitted between an application server and a client device on a network based upon a stateless communication protocol, where the are packets received by an intermediate server that does not have access to the protocol definition used by the client device and the application server. The packets transmitted between the application server and the client device are monitored by the intermediate server and parsed into a plurality of name-value pairs (data objects). The entropy of the data objects having a same name field is computed, and candidate data objects that are likely to be server state objects are selected based upon the computed entropy. In one embodiment, data objects with an entropy higher than a threshold are selected as the candidate data objects. Finally, the candidate data objects that are determined to have been transmitted bi-directionally between the application server and the client device are identified as server state objects.

[0009] In one embodiment, the entropy of the data objects is computed by concatenating the value fields of the parsed data objects that have a same name field, compressing the concatenated value fields using an entropy-based compression algorithm, and dividing the total size of the compressed concatenated value field by the total number of data objects having that same name field.

[0010] The embodiments of the present invention can identify server state objects that were transmitted between an application server and a client device even when a less structured stateless communication protocol such as XML or SOAP is used. Thus, an intermediate server can identify a client device that previously communicated with the application server by examining the identified server state objects, without having any access to the underlying application protocol used by the application server and the client. This enables the underlying application protocols to be modified, or new applications to be hosted on the network, without having to continually update information held by the intermediate server. This also facilitates the implementation of Internet transactions using such less structured stateless communication protocols.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a block diagram illustrating the architecture of a system for identifying server state objects, according to one embodiment of the present invention.

[0012] FIG. 2 is a flowchart illustrating a method for identifying server state objects,
5 according to one embodiment of the present invention.

[0013] FIG. 3 is a diagram illustrating the name-value pairs.

[0014] FIG. 4 is diagram illustrating the step of identifying name-value pair transmitted bi-directionally between the server and the client device.

DETAILED DESCRIPTION OF EMBODIMENTS

10 [0015] FIG. 1 is a block diagram illustrating the architecture of a system 100 for identifying server state objects according to one embodiment of the present invention. The system 100 is coupled between an application server 102 and a client device 104 to monitor packets 118 transmitted between the server 102 and the client device 104. The packets 118 may be transmitted bi-directionally between the application server 102 and the client device 104, or
15 only in a single direction from the application server 102 to the client device 104 or vice versa. The packets 118 are transmitted between the application server 102 and the client device using a stateless data communication protocol, such as XML, SOAP, or HTTP. However, the system 100 typically does not have the protocol definition given by the application server 106 by which to analyze the packets to identify server state objects. Although one client device 104 is shown
20 in FIG. 1, it should be noted that in practice there will be numerous client devices 104 communicating with the application server 102. Also, while only a single application server 102 is shown, any number of application servers 102 may be deployed behind the system 100 to service the client devices 104.

[0016] In one embodiment, the system 100 includes an application firewall 106, a learning
25 engine 108, and a data storage module 110. The application firewall 106 monitors the packets 118 transmitted from the server 102 to the client device 104 and from the client device 104 to the server 102. The application firewall 106 parses the monitored packets 118 and filters malicious or undesirable packets 118 according to the filtering rules 116 provided by the learning engine 108. The filtering rules 116 provide the application firewall 106 with the

criteria to pass through or block certain packets 118 transmitted between the server 102 and the client device 104.

[0017] The application firewall 106 also parses the monitored packets 118 and provides the learning engine 108 with data 114 collected according to the data collection parameters 112 provided by the learning engine 108. The data collection parameters 112 provide the application firewall 106 with criteria on the type of data that the application firewall 106 should collect from the monitored packets 118. These data collection parameters 112 include, for example, how to parse the packets 118 to collect name-value (name-attribute) pairs corresponding to the packets 118 according to the communication protocol. However, in the typical embodiment, the application firewall 106 will not have access to the underlying protocol definition (e.g., a document type definition (DTD)) as used by the application server 104

[0018] In one embodiment, the application firewall 106 itself may be a computer system running software dedicated for the application firewall functions. In another embodiment, the application firewall 106 may be a software module that is part of a large software system running on a computer system 100.

[0019] The learning engine 108 provides the filtering rules 116 and the data collection parameters 112 to the application firewall 106, and receives the data 114 collected by the application firewall 106 according to the data collection parameters 112. The packets 118 are those that are transmitted between the server 102 and the client device 104 according to the employed communication protocol. For example, the packets 118 may also be XML packets or SOAP packets. The application firewall 106 parses the packets 118 into name-value (name-attribute) pairs 114, and the learning engine 108 receives these name-value pairs 114 for further processing to identify a server state object among the packets 118. The name-value (name-attribute) pairs are also referred to herein as "data objects."

[0020] The learning engine 108 identifies server state objects among the packets 118 transmitted between the server 102 and the client device 104 by analyzing the data objects 114 or name-value pairs 114 received from the application firewall 106. In this regard, the learning engine 108 identifies the name-value pairs 114, and for each identified name field, concatenates the values for that name. After accumulation of a minimum number of values for each name, the learning engine 108 then compresses the concatenated values for that name using a compression algorithm. The compression algorithm may be an entropy-based compression

algorithm such as a Lempel-Ziv type compression algorithm. Then, the learning engine 108 computes the entropy of each of the compressed values by dividing the total size of the compressed values by the total number of values having that same name.

[0021] The learning engine 108 selects candidate data objects that are likely to be server state objects from the data objects according to their computed entropies. Since server state objects typically contain relatively distinct strings in their value fields in order to maintain the state of each client, the entropies of server state objects are likely to be high. Thus, in one embodiment, the learning engine 108 selects data objects having an entropy value higher than a threshold value as the candidate data objects. The threshold value for the entropy differs depending upon the underlying communication protocol used for transmission of the packets 118. A typical server state object has an entropy value corresponding to 32 bits or more, since it is often an object reference into the server implementation or a key in a database table. Thus, in one embodiment, the threshold value is set at an entropy level corresponding to 24 bits.

[0022] The learning engine 108 identifies one or more of the candidate data objects as server state objects if the candidate data object was transmitted both from the application server 102 to the client device 104 and from the client device 104 to the application server 102. This is because a server state object would have been sent by the application server 102 to the client device 104 and stored in the client device 104 so that the application server 102 can later identify the client device 104, and also sent from the client device 104 to the application server 102 later on so that the client device 104 can identify itself to the application server 102.

[0023] In one embodiment, the learning engine 108 identifies candidate data objects that were bi-directionally transmitted between the server 102 and the client device 104 by dividing related packets under the communication protocol into a first group of packets that were transmitted from the server to the client device and a second group of packets that were transmitted from the client device to the server. Related packets are packets that correspond to a "related message" under the communication protocol. For example, packets that form a part of a request/response communication of an HTTP session are related packets under the HTTP protocol. In SOAP, requests and responses may be carried out in separate connections. Therefore, the notion of related packets under SOAP is externally provided. In one embodiment, messages between the same source and destination points (based upon source IP address/port and destination IP address/port) may be considered related packets under SOAP.

In other embodiments, messages between two same services (identified by the URL (Uniform Resource Locator)) are considered related packets under SOAP. The learning engine 108 identifies the candidate data objects included in both the first group of packets and the second group of packets as server state objects.

5 [0024] In one embodiment, the data storage module 110 may temporarily store the collected data 114 and other interim data, while the learning engine 108 analyzes the collected data 114 in real time as the data objects 114 are received from the application firewall 106 to identify the server state objects among the packets 118. In another embodiment, the storage module 110 stores the received data objects in the collected data 114 for a predetermined period of time or
10 until the amount of the stored data objects reach a certain threshold, and the learning engine 108 processes the stored data objects in batches to determine whether the data objects are server state objects.

[0025] FIG. 2 is a flowchart illustrating a method for identifying server state objects among packets 118 transmitted between a server 102 and a client device 104 on a network based upon a
15 stateless communication protocol, according to one embodiment of the present invention. The method described in FIG. 2 may be performed in the system 100 of FIG. 1.

[0026] As the method begins 202, the application firewall 106 parses 204 the packets 118 transmitted from the server 102 to the client device 104 or from the client device 104 to the server 102 into a plurality of name-value pairs (data objects). FIG. 3 describes examples of
20 name-value pairs 303. Referring to FIG. 3, the packets 118 are parsed and divided into names 302 and values (attributes) 304 corresponding to the names 302. For example, the name "Order ID" 305 may be a name corresponding to Internet sales transactions performed between the server web site 102 and the client device 104. These name fields "Order ID" 305 have corresponding value fields "ABCDEFABCDEF," "111BBCCCDDD," and
25 "DDDCCC1112233." The name fields "Author" 306, "Account No." 307, and "Title" 308 have corresponding value fields "John," "123456789," and "Internet Sales," respectively. While FIG. 3 only shows three name-value pairs 303, in practice there can be hundreds, or even thousands of name-value pairs 303 maintained by the application firewall 106 at any given time.

[0027] Referring back to FIG. 2, the learning engine 108 concatenates 206 the value fields
30 of the parsed data objects having the same name field. For example, some of the name-value pairs in FIG. 3 have same names "Order ID" 305. Thus, the value fields 304 corresponding to

the name "Order ID" are concatenated to obtain a concatenated value "ABCDEFABCDEF111BBCCCDDDDDDCCC1112233" corresponding to the name field "Order ID" 305.

5 [0028] After a minimum number of name-value pairs for any given name have been identified, the learning engine 108 compresses 208 the concatenated values for that name using a compression algorithm. In one embodiment, the compression algorithm may be an entropy-based compression algorithm such as a Lempel-Ziv type compression algorithm. Then, the learning engine 108 computes 210 the entropy of the compressed concatenated values by dividing the total size of the compressed concatenated value list by the total number of name-value pairs for that name. Therefore, in the example shown in FIG. 3, the concatenated value list "ABCDEFABCDEF111BBCCCDDDDDDCCC1112233" is compressed 208 using a compression algorithm, and the total size of the compressed concatenated values is divided by the number (three in the example of FIG. 3) of name-value pairs having the same name field ("Order ID").

15 [0029] As is apparent, there will be numerous data objects that are potential candidates for being server state objects, and for each of these candidates there will be a computed entropy measure. From this set of potential candidates, the learning engine 108 selects 212 candidate name-value pairs that are likely to be server state objects according to the computed entropies. As stated previously, in one embodiment, the learning engine 108 selects the name-value pairs having an entropy measure that is higher than a predetermined threshold as the candidate name-value pairs. The threshold value for the entropy differs depending upon the underlying communication protocol used for transmission of the packets 118. A typical server state object has an entropy value corresponding to 32 bits or more, since it is often an object reference into the server implementation or a key in a database table. Thus, in one embodiment, the threshold value is set at an entropy level corresponding to 24 bits. Thus, in the foregoing, "Order ID" and "Account No." may be selected as having sufficiently high entropy, while "Author" and "Title" may not.

20 [0030] Then, the learning engine 108 determines 214 which of the candidate name-value pairs have been sent bi-directionally between the application server 102 and the client device 104. If the candidate name-value pair was sent bi-directionally, the learning engine 108 labels 218 that the candidate name-value pair as a server state object. If the name-value pair was not

sent bi-directionally, the learning engine 108 determines 216 that the candidate name-value pair is not a server state object.

[0031] FIG. 4 illustrates the process of identifying a name-value pair transmitted bi-directionally between the server and the client device as described in FIG. 2. As stated previously, the learning engine 108 identifies one or more of the candidate name-value pairs as server state objects if the candidate name-value pair was transmitted bi-directionally between the application server 102 and the client device 104. This is because a server state object would have been sent by the application server 102 to the client device 104 and stored in the client device 104 so that the application server 102 can later identify the client device 104, and also sent from the client device 104 to the application server 102 later on so that the client device 104 can identify itself to the application server 102.

[0032] Referring to FIG. 4, in one embodiment, the learning engine 108 divides related packets under the communication protocol into a first group 402 of packets that were transmitted from the application server 102 to the client device 104 and a second group 404 of packets that were transmitted from the client device 104 to the application server 102 in sequential order of their associated time-stamp indicating the receipt of the corresponding packets 118 by the application firewall 106. Related packets are packets that correspond to a related message under the communication protocol. For example, packets that form parts of a request/response of an HTTP session are related packets under the HTTP protocol. In SOAP, requests and responses may be carried out in separate connections. Therefore, the notion of related packets under SOAP is externally provided. In one embodiment, messages between the same source and destination points (based upon source IP address/port and destination IP address/port) may be considered related packets under SOAP. In other embodiments, messages between two same services (identified by the URL (Uniform Resource Locator)) are considered related packets under SOAP. In the example shown in FIG. 4, the name-value pairs 406, 408, 410, and 412 are candidate name-value pairs that were transmitted bi-directionally between the application server 102 and the client device 104. It should be noted that there may also be additional candidate name-value pairs other than the candidate name-value pairs 406, 408, 410, 412, although they are not shown in FIG. 4.

[0033] The learning engine 108 identifies the name-value pairs included in both the first group 402 of packets and the second group 404 of packets as server state objects. For example,

the candidate name-value pairs 406, 408, 410, 412, 420, 422 are included in both the first group 402 of packets and the second group 404 of packets and determined to be server state objects by the learning engine 108. However, the candidate name-value pairs 416 and 418 are included in only the second group 404 of packets and thus are not server state objects.

5 [0034] The present invention has the advantage that server state objects transmitted between an application server and a client device may be identified by an intermediate server, such as a filter or a gateway, even when a less structured stateless data communication protocol such as XML or SOAP is used, where the intermediate server does not have access to the underlying application protocol. This facilitates the implementation of Internet transactions using such less
10 structured stateless data communication protocols, because a server can identify a client device that previously communicated with the server by examining the server state objects.

[0035] Although the present invention has been described above with respect to several embodiments, various modifications can be made within the scope of the present invention. For example, the present invention is not limited to any communication protocol and may be used
15 with any stateless communication protocol. The architecture of the system 100 of FIG. 1 is merely exemplary and may be modified in any way to the extent that the system 100 may monitor the packets transmitted bi-directionally between the application server 102 and the client device 104. Compression of the value fields of the data objects may be done using any type of compression algorithm, and the method of calculating the entropy of the concatenated
20 compressed data objects is not limited to what is described herein. The method of identifying bi-directional packets is not limited to what is described herein in FIG. 4, and different methods may be used as long as they are suitable for identifying packets bi-directionally transmitted between the application server 102 and the client device. The present invention may also be used to identify server state objects when a plurality of client devices 104 communicate with the
25 application server 102 or when the client device 104 communicates with a plurality of application servers 102.

[0036] Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

WHAT IS CLAIMED IS:

1. A method for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the method comprising:

5 computing entropy measures for a plurality of data objects in the packets;
selecting candidate data objects from the plurality of data objects according to
their entropy measures; and
identifying at least one of the candidate data objects as a server state object in
response to the at least one candidate data object being transmitted bi-
10 directionally between the server and the client device.

2. The method of claim 1, wherein computing entropy measures comprises:
compressing the data objects; and
computing the entropy measures of the compressed data objects.

15

3. The method of claim 2, wherein compressing the data objects comprises:
concatenating the data objects having a same name field; and
compressing the concatenated data objects using a compression algorithm.

20

4. The method of claim 3, wherein the compression algorithm is an entropy-based
compression algorithm.

25

5. The method of claim 3, wherein computing the entropy measures of the
compressed data objects comprises dividing a total size of value fields of the compressed
concatenated data objects by a total number of data objects having the same name field.

30

6. The method of claim 1, wherein the at least one candidate data object is
determined to have been transmitted bi-directionally between the server and the client device
by:

dividing related packets under the communication protocol into a first group of
packets that were transmitted from the server to the client device and a

second group of packets that were transmitted from the client device to the server; and

determining whether the at least one candidate data object is included in both the first group of packets and the second group of packets.

5

7. The method of claim 6, wherein the related packets are packets that correspond to a related message under the communication protocol.

8. A method for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the method comprising:

10

parsing the packets into a plurality of name-value pairs;

computing the entropies of the name-value pairs having a same name;

selecting the name-value pairs having entropies higher than a threshold as

15

candidates for the server state objects;

determining whether the candidates were sent bi-directionally between the server and the client device; and

determining that at least one of the candidates is a server state object in response to determining that the at least one candidate was sent bi-directionally

20

between the server and the client device.

9. The method of claim 8, wherein computing the entropies of the name-value pairs having a same name comprises:

compressing the name-value pairs having the same name; and

25

computing the entropies of the compressed name-value pairs.

10. The method of claim 9, wherein compressing the name-value pairs having the same name comprises:

concatenating the value fields of the name-value pairs having the same name;

30

and

compressing the concatenated value fields of the name-value pairs using a compression algorithm.

11. The method of claim 10, wherein the compression algorithm is an entropy-based compression algorithm.

5 12. The method of claim 10, wherein computing the entropies of the compressed name-value pairs comprises dividing a total size of the compressed concatenated value fields of the name-value pairs by a total number of the name-value pairs having the same name.

10 13. The method of claim 8, wherein determining whether the candidates were sent bi-directionally between the server and the client device comprises:

dividing related packets under the communication protocol into a first group of packets that were transmitted from the server to the client device and a second group of packets that were transmitted from the client device to the server; and

15 determining whether the candidates are included in both the first group of packets and the second group of packets.

14. The method of claim 13, wherein the related packets are packets that correspond to a related message under the communication protocol.

20 15. A system for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the system comprising:

a parsing module coupled between the server and the client device for parsing the packets transmitted between into a plurality of data objects; and

25 an analysis module coupled to the parsing module for receiving the data objects and determining whether the name-value pairs are server state objects by: computing entropy measures of the data objects;

30 selecting candidate data objects from the data objects according to their entropy measures; and

identifying at least one of the candidate data objects as a server state object in response to the at least one candidate data object being

transmitted bi-directionally between the server and the client device.

16. The system of claim 15, wherein the parsing module operates as a firewall
5 filtering the packets transmitted between the server and the client device in accordance with
rules provided by the analysis module.

17. The system of claim 15, wherein the parsing module collects data regarding the
packets transmitted between the server and the client device in accordance with data collection
10 parameters provided by the analysis module.

18. The system of claim 15, wherein the analysis module processes the data objects
to determine whether the data objects are server state objects in real time as the data objects are
received from the parsing module.

19. The system of claim 15, further comprising a storage module coupled to the
analysis module, the analysis module storing the received data objects in the storage module and
processing the stored data objects in batches to determine whether the data objects are server
state objects.

20. The system of claim 15, wherein the analysis module computes the entropy
measures of the data objects by:

compressing the data objects; and
computing the entropy measures of the compressed data objects.

21. The system of claim 20, wherein the analysis module compresses the data objects
by:
concatenating the data objects having a same name field; and
compressing the concatenated data objects using a compression algorithm.

22. The system of claim 21, wherein the compression algorithm is an entropy-based
compression algorithm.

23. The system of claim 21, wherein the analysis module computes the entropy measures of the compressed data objects by dividing a total size of value fields of the compressed concatenated data objects by a total number of data objects having the same name field.

24. The system of claim 15, wherein the analysis module determines that the at least one candidate data object was transmitted bi-directionally between the server and the client device by:

dividing related packets under the communication protocol into a first group of packets that were transmitted from the server to the client device and a second group of packets that were transmitted from the client device to the server; and

determining whether the at least one candidate data object is included in both the first group of packets and the second group of packets.

25. The system of claim 24, wherein the related packets are packets that correspond to a related message under the communication protocol.

26. A system for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the system comprising:

a parsing module coupled between the server and the client device for parsing the packets transmitted between into a plurality of name-value pairs; and

an analysis module coupled to the parsing module for receiving the name-value pairs and determining whether the name-value pairs are server state objects by:

computing the entropies of the name-value pairs having a same name;

selecting the name-value pairs having entropies higher than a threshold as candidates for the server state objects;

determining whether the candidates were sent bi-directionally between the server and the client device; and

determining that at least one of the candidates is a server state object in response to determining that the at least one candidate was sent bi-directionally between the server and the client device.

5 27. The system of claim 26, wherein the parsing module operates as a firewall filtering the packets transmitted between the server and the client device in accordance with rules provided by the analysis module.

10 28. The system of claim 26, wherein the parsing module collects data regarding the packets transmitted between the server and the client device in accordance with data collection parameters provided by the analysis module.

15 29. The system of claim 26, wherein the analysis module processes the name-value pairs in real time as the name-value pairs are received from the parsing module to determine whether the name-value pairs are server state objects.

20 30. The system of claim 26, further comprising a storage module coupled to the analysis module, the analysis module storing the received name-value pairs in the storage module and processing the stored name-value pairs in batches to determine whether the name-value pairs are server state objects.

25 31. The system of claim 26, wherein the analysis module computes the entropies of the name-value pairs having a same name by:
 compressing the name-value pairs having the same name; and
 computing the entropies of the compressed name-value pairs.

30 32. The system of claim 31, wherein the analysis module compresses the name-value pairs having the same name by:
 concatenating value fields of the name-value pairs having the same name; and
 compressing the concatenated value fields of the name-value pairs using a
 compression algorithm.

33. The system of claim 32, wherein the compression algorithm is an entropy-based compression algorithm.

5 34. The system of claim 32, wherein the analysis module computes the entropies of the compressed name-value pairs by dividing a total size of the compressed concatenated value fields of the name-value pairs by a total number of name-value pairs having the same name.

35. The system of claim 26, wherein the analysis module determines whether the
10 candidates were sent bi-directionally between the server and the client device by:

dividing related packets under the communication protocol into a first group of packets that were transmitted from the server to the client device and a second group of packets that were transmitted from the client device to the server; and

15 determining whether the candidates are included in both the first group of packets and the second group of packets.

36. The system of claim 35, wherein the related packets are packets that correspond to a related message under the communication protocol.

20 37. A computer program product stored on a computer readable medium and adapted to perform a method for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the method comprising:

25 computing entropy measures for a plurality of data objects in the packets;
selecting candidate data objects from the plurality of data objects according to their entropy measures; and

30 identifying at least one of the candidate data objects as a server state object in response to the at least one candidate data object being transmitted bi-directionally between the server and the client device.

38. The computer program product of claim 37, wherein computing entropy measures comprises:

5 compressing the data objects; and
computing the entropy measures of the compressed data objects.

39. The computer program product of claim 38, wherein compressing the data objects comprises:

10 concatenating the data objects having a same name field; and
compressing the concatenated data objects using a compression algorithm.

40. The computer program product of claim 39, wherein computing the entropy measures of the compressed data objects comprises dividing a total size of value fields of the compressed concatenated data objects by a total number of data objects having the same name
15 field.

41. The computer program product of claim 37, wherein the at least one candidate data object is determined to have been transmitted bi-directionally between the server and the client device by:

20 dividing related packets under the communication protocol into a first group of packets that were transmitted from the server to the client device and a second group of packets that were transmitted from the client device to the server; and

25 determining whether the at least one candidate data object is included in both the first group of packets and the second group of packets.

42. A computer program product stored on a computer readable medium and adapted to perform a method for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the method
30 comprising:

parsing the packets into a plurality of name-value pairs;
computing the entropies of the name-value pairs having a same name;

selecting the name-value pairs having entropies higher than a threshold as candidates for the server state objects;
determining whether the candidates were sent bi-directionally between the server and the client device; and
5 determining that at least one of the candidates is the server state object in response to determining that the at least one candidate was sent bi-directionally between the server and the client device.

43. The computer program product of claim 42, wherein computing the entropies of
10 the name-value pairs having a same name comprises:
compressing the name-value pairs having the same name; and
computing the entropies of the compressed name-value pairs.

44. The computer program product of claim 43, wherein compressing the name-
15 value pairs having the same name comprises:
concatenating the value fields of the name-value pairs having the same name;
and
compressing the concatenated value fields of the name-value pairs using a
compression algorithm.

20 45. The computer program product of claim 44, wherein computing the entropies of the compressed name-value pairs comprises dividing a total size of the compressed concatenated value fields of the name-value pairs by a total number of name-value pairs having the same name.

25 46. The computer program product of claim 42, wherein determining whether the candidates were sent bi-directionally between the server and the client device comprises:
dividing related packets under the communication protocol into a first group of
packets that were transmitted from the server to the client device and a
30 second group of packets that were transmitted from the client device to the server; and
determining whether the candidates are included in both the first group of packets and the second group of packets.

47. A system for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the system comprising:

5 parsing means coupled between the server and the client device for parsing the packets transmitted between into a plurality of data objects; and analysis means coupled to the parsing means for receiving the data objects and determining whether the name-value pairs are server state objects by: computing entropy measures of the data objects;

10 selecting candidate data objects from the data objects according to their entropy measures; and identifying at least one of the candidate data objects as a server state object in response to the at least one candidate data object being transmitted bi-directionally between the server and the client

15 device.

48. A system for identifying server state objects among packets transmitted between a server and a client device on a network based upon a communication protocol, the system comprising:

20 parsing means coupled between the server and the client device for parsing the packets transmitted between into a plurality of name-value pairs; and analysis means coupled to the parsing means for receiving the name-value pairs and determining whether the name-value pairs are server state objects by: computing the entropies of the name-value pairs having a same name;

25 selecting the name-value pairs having entropies higher than a predetermined threshold as candidates for the server state objects; determining whether the candidates were sent bi-directionally between the server and the client device; and

30 determining that at least one of the candidates is a server state object in response to determining that the at least one candidate was sent bi-directionally between the server and the client device.

1/4

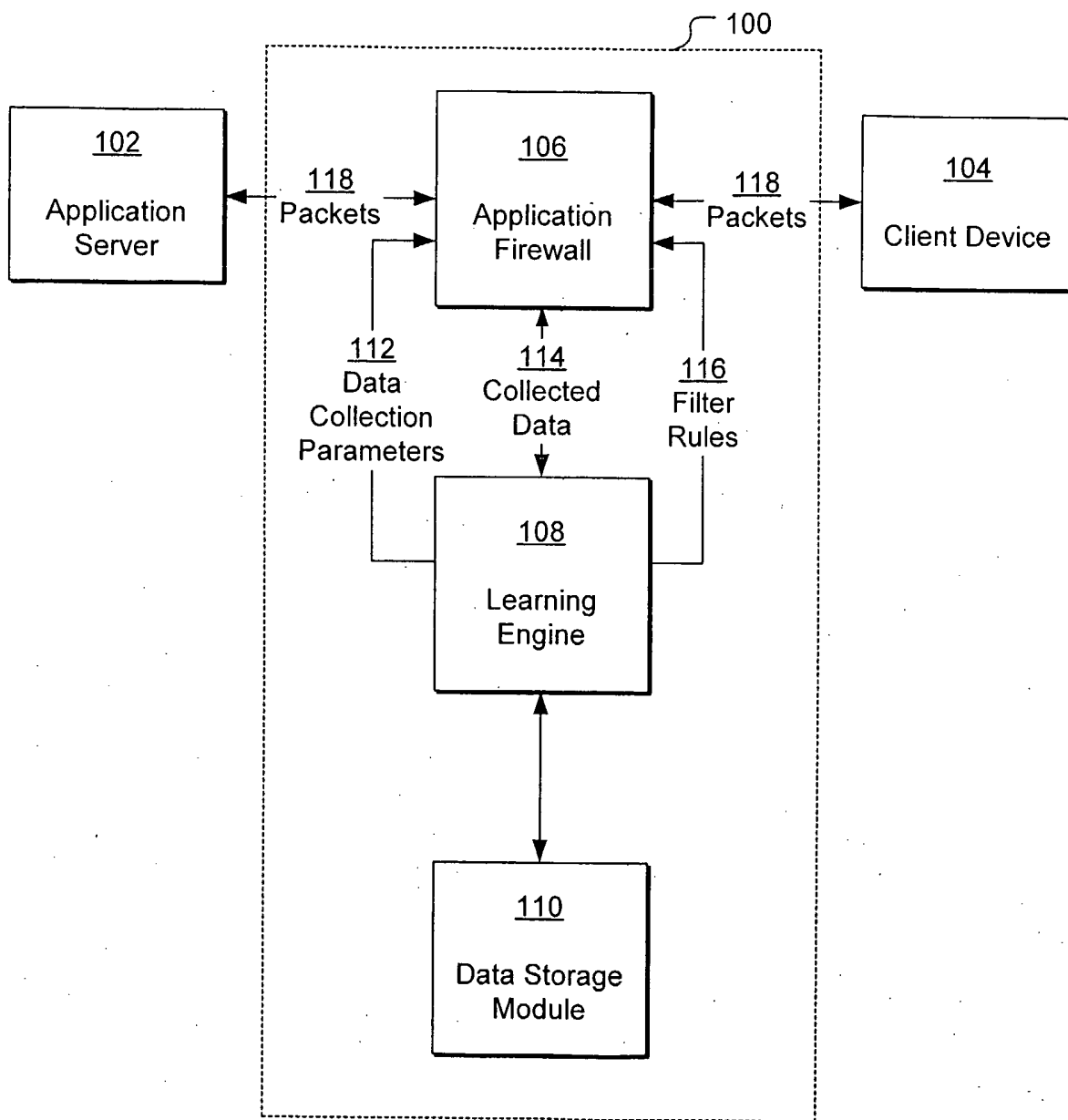


FIG. 1

2/4

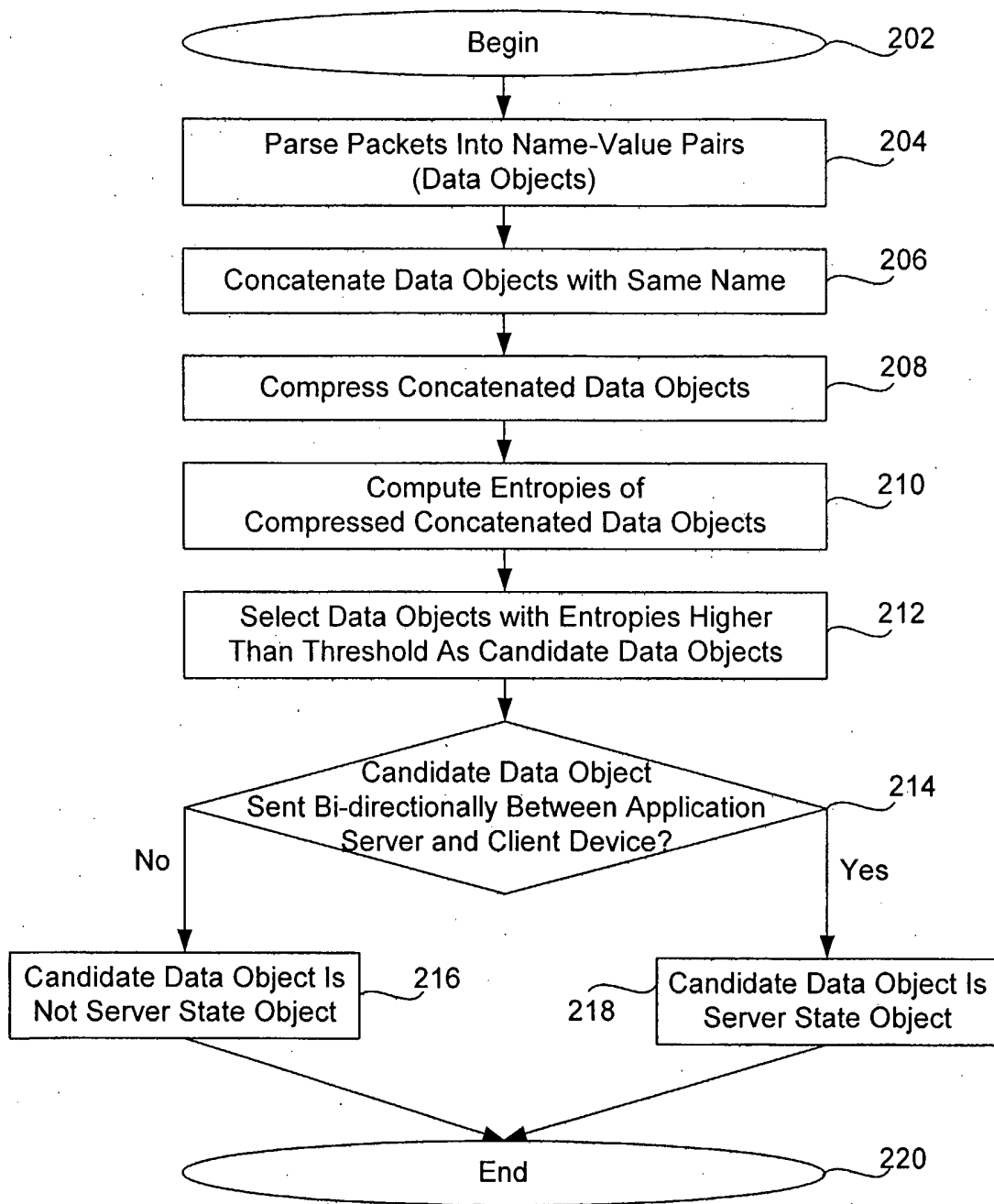


FIG. 2

3/4

303

Name 302	Value (Attribute) 304
Order ID 305	ABCDEFABCDEF
.	.
:	:
Author 306	John
Order ID 305	111BBCCDDDD
Account No. 307	123456789
.	.
:	:
Title 308	Internet Sales
Order ID 305	DDDCCC1112233

FIG. 3

4/4

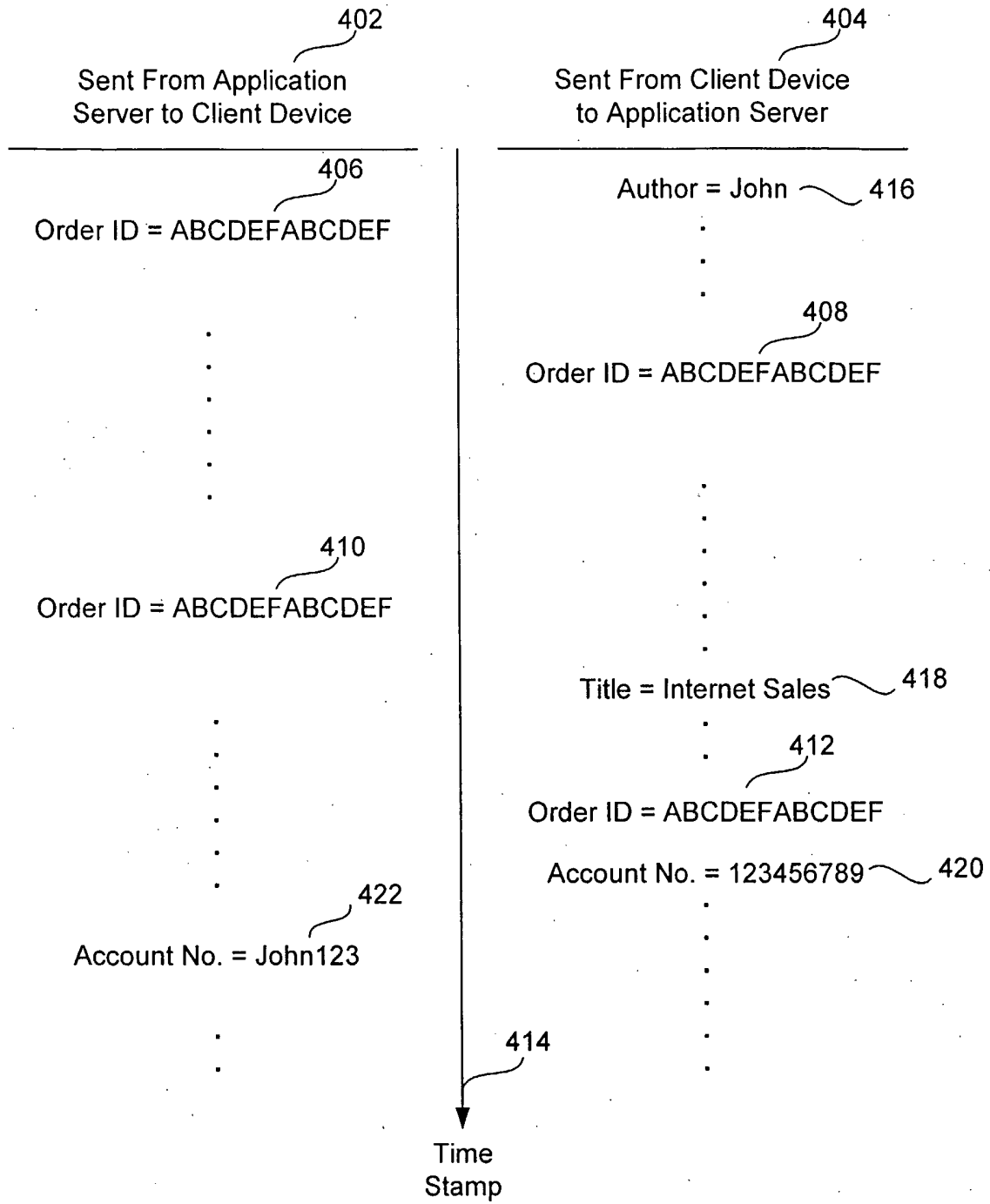


FIG. 4