US 20080285468A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2008/0285468 A1

(43) Pub. Date: Nov. 20, 2008

Seo et al.

(54) METHOD AND COMPUTER-READABLE MEDIUM FOR DETECTING ABNORMAL PACKET IN VOIP

(75) Inventors: **Dong-Won Seo**, Anyang-city (KR); **Hee-Jo Lee**, Namyangju-city (KR)

Correspondence Address:
ALSTON & BIRD LLP
BANK OF AMERICA PLAZA, 101 SOUTH TRYON STREET, SUITE 4000
CHARLOTTE, NC 28280-4000 (US)

(73) Assignee: **KOREA UNIVERSITY INDUSTRY AND ACADEMY COLLABORATION FOUNDATION**

(21) Appl. No.: **12/075,016**

(22) Filed: **Mar. 7, 2008**

(30) **Foreign Application Priority Data**

May 15, 2007 (KR) 10-2007-0046950

Publication Classification

(51) **Int. Cl.**
H04L 12/26 (2006.01)

(52) **U.S. Cl.** **370/242**

(57) **ABSTRACT**

In a session initiation protocol (SIP) corresponding to one of protocols for the voice over Internet protocol (VoIP), an exception detection module acquires a packet for session establishment. If each of fields of the header of the packet is out of a defined character length and includes at least one prohibited character, the exception detection module discards the packet.

The exception detection module confirms a state of the session. If the packet is acquired more than a predetermined number of times in the state of the session, the exception detection module considers the packet to be abnormal and discards the packet.

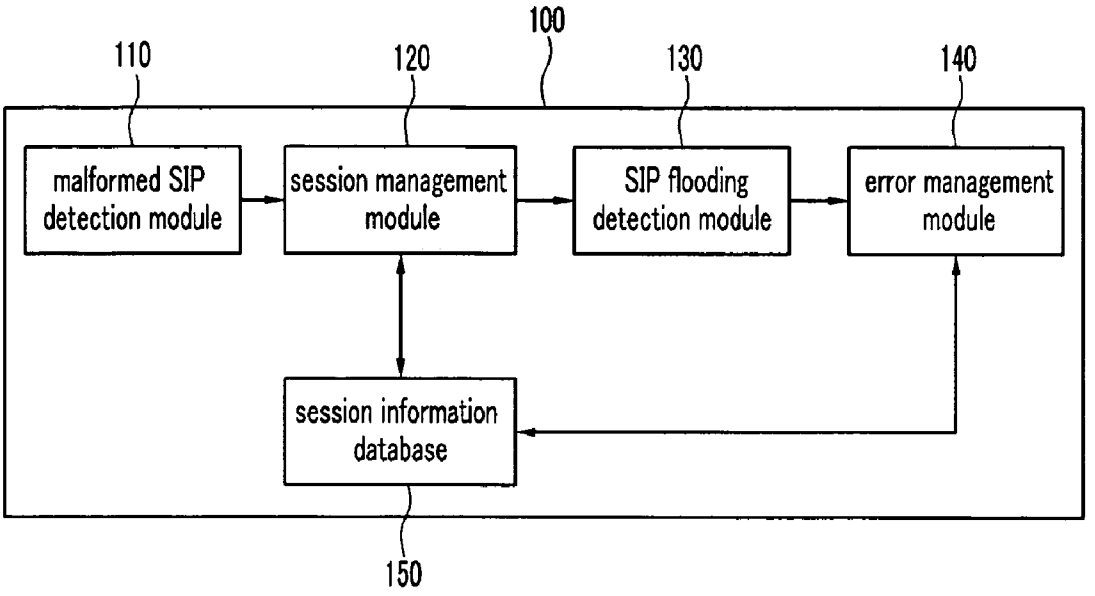


FIG. 1

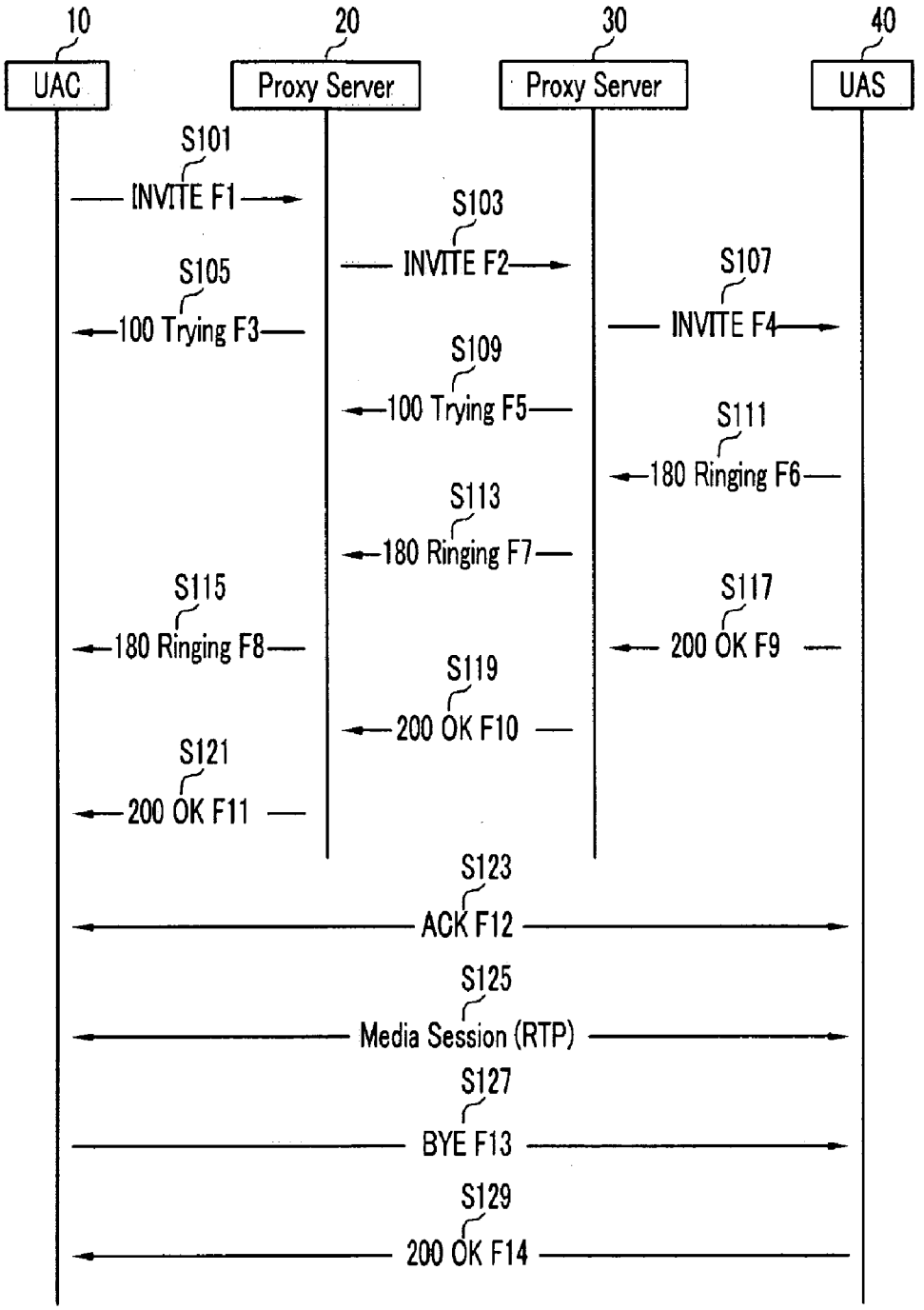

10
20
30
40
UAC
Proxy Server
Proxy Server
UAS
S101
INVITE F1
S103
INVITE F2
S105
100 Trying F3
S107
INVITE F4
S109
100 Trying F5
S111
180 Ringing F6
S113
180 Ringing F7
S115
180 Ringing F8
S117
200 OK F9
S119
200 OK F10
S121
200 OK F11
S123
ACK F12
S125
Media Session (RTP)
S127
BYE F13
S129
200 OK F14

FIG. 2

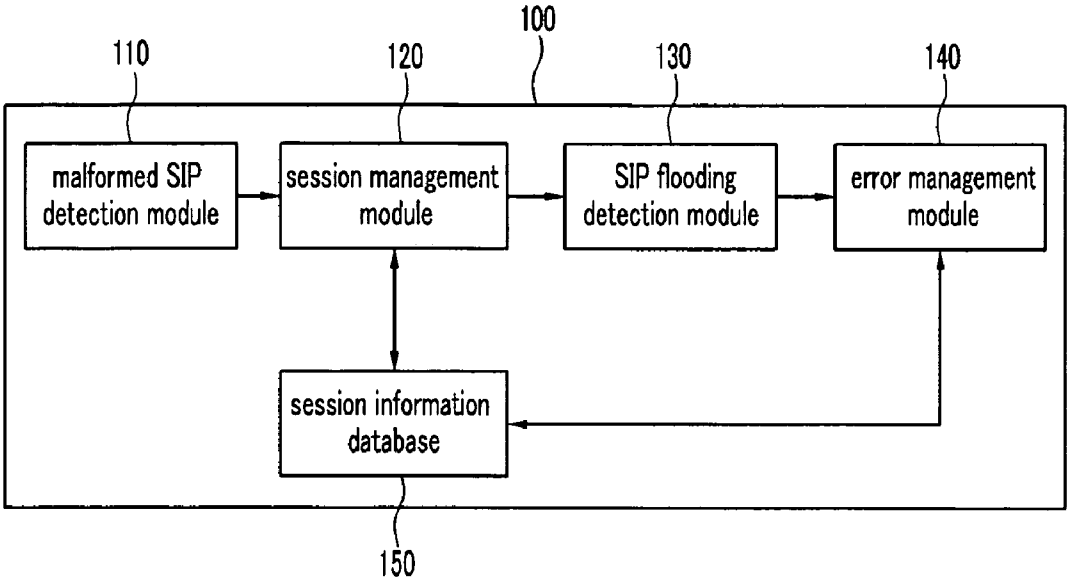
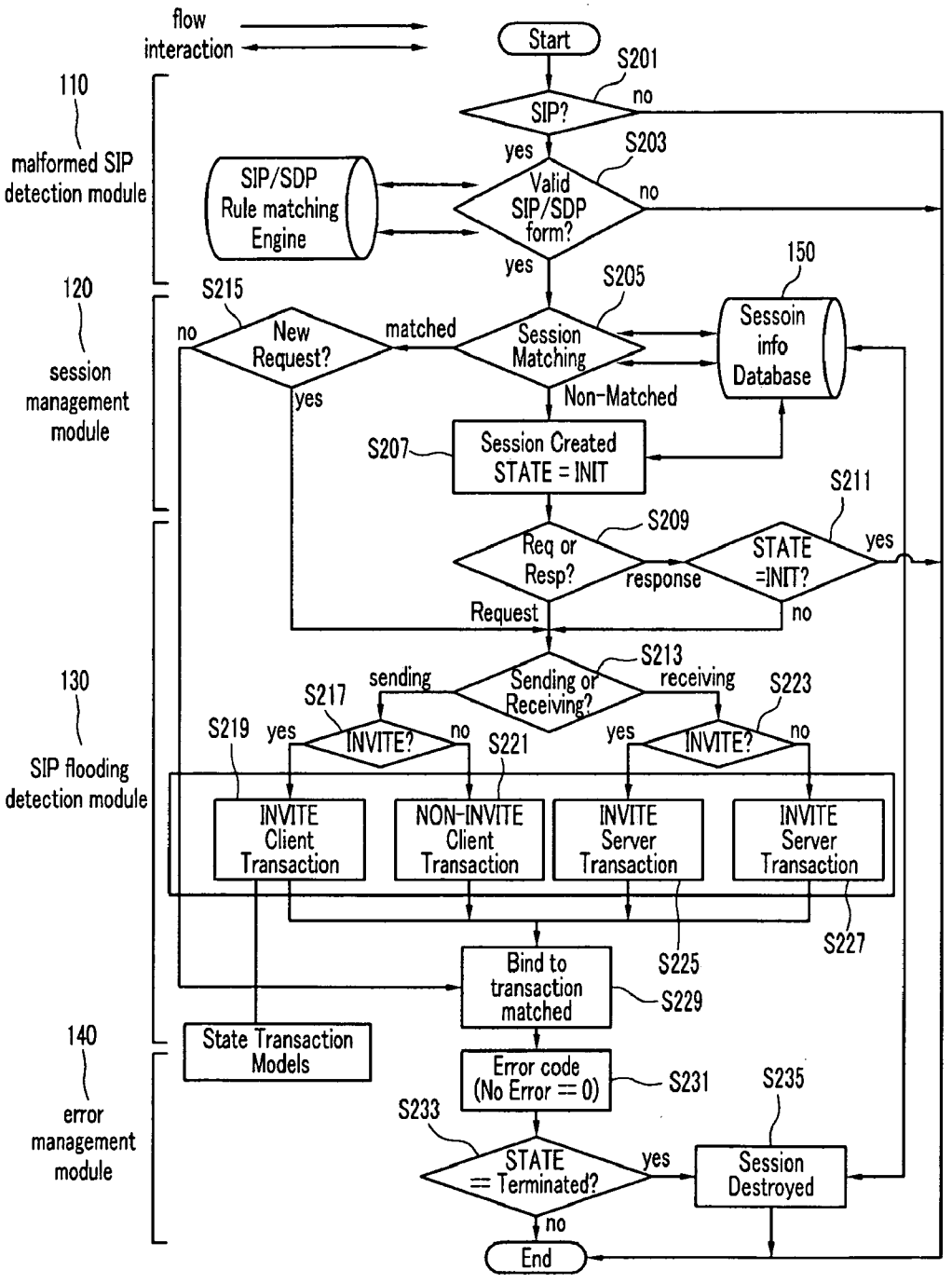

100
110
120
130
140
malformed SIP detection module
session management module
SIP flooding detection module
error management module
session information database
150

FIG. 3


flow
interaction
Start
S201
SIP?
no
yes
S203
Valid SIP/SDP form?
no
SIP/SDP Rule matching Engine
yes
110
malformed SIP detection module
120
session management module
S205
150
Sessoin info Database
S215
New Request?
no
matched
Session Matching
Non-Matched
yes
S207
Session Created STATE = INIT
S211
S209
Req or Resp?
response
STATE =INIT?
yes
no
Request
S213
Sending or Receiving?
sending
receiving
130
SIP flooding detection module
S217
S223
S219
yes
INVITE?
no
S221
yes
INVITE?
no
INVITE Client Transaction
NON-INVITE Client Transaction
INVITE Server Transaction
INVITE Server Transaction
S227
Bind to transaction matched
S225
S229
140
error management module
State Transaction Models
Error code (No Error = 0)
S231
S235
S233
STATE = Terminated?
yes
Session Destroyed
no
End

Original RFC rules	Improved rules
SIP_Version:(SIP\/\d\.\d)	SIP_Version:((SIP\/\d\.\d){7,9})
port:(\d+)	port:(\d{1,5})
hostname:((#domainlabel#\.)*#toplabel#(\.)?)	hostname : (((#domainlabel#{2,63}\.)*#toplabel#{2,6}(\.)?))
user:((#unreserved#|#escaped#|#user_unreserved#)+)	user:((#alphanum#|_|\-)+{1,12})
password:((#unreserved#|#escaped#|\&|\=|\+|\$|\,)*)	password:(((#unreserved#|#escaped#|\&|\=|\+|\$|\,)*){0,12})
SIP_Version:(SIP\/\d\.\d)	SIP_Version:((SIP\/\d\.\d){7,9})
extension_method:(#token#)	extension_method:(#ASCII_NAME#{1,20})
protocol_version:(#token#)	protocol_version:(\d{1,2}\.\d{1,2})
display_name:((#token##LWS#)*|#quoted_string#)	display_name:((\w|){1,32}|#quoted_string#)
callid : (#word#(\@#word#)?)	callid:(#ASCII#{1,50}(\@(\w\.*){1,32})?)
Max-Forwards:(Max-Forwards#HCOLON#\d+#CRLF#)	Max-Forwards:(Max-Forwards#HCOLON#\d{1,4}#CRLF#)

FIG. 4

FIG. 5

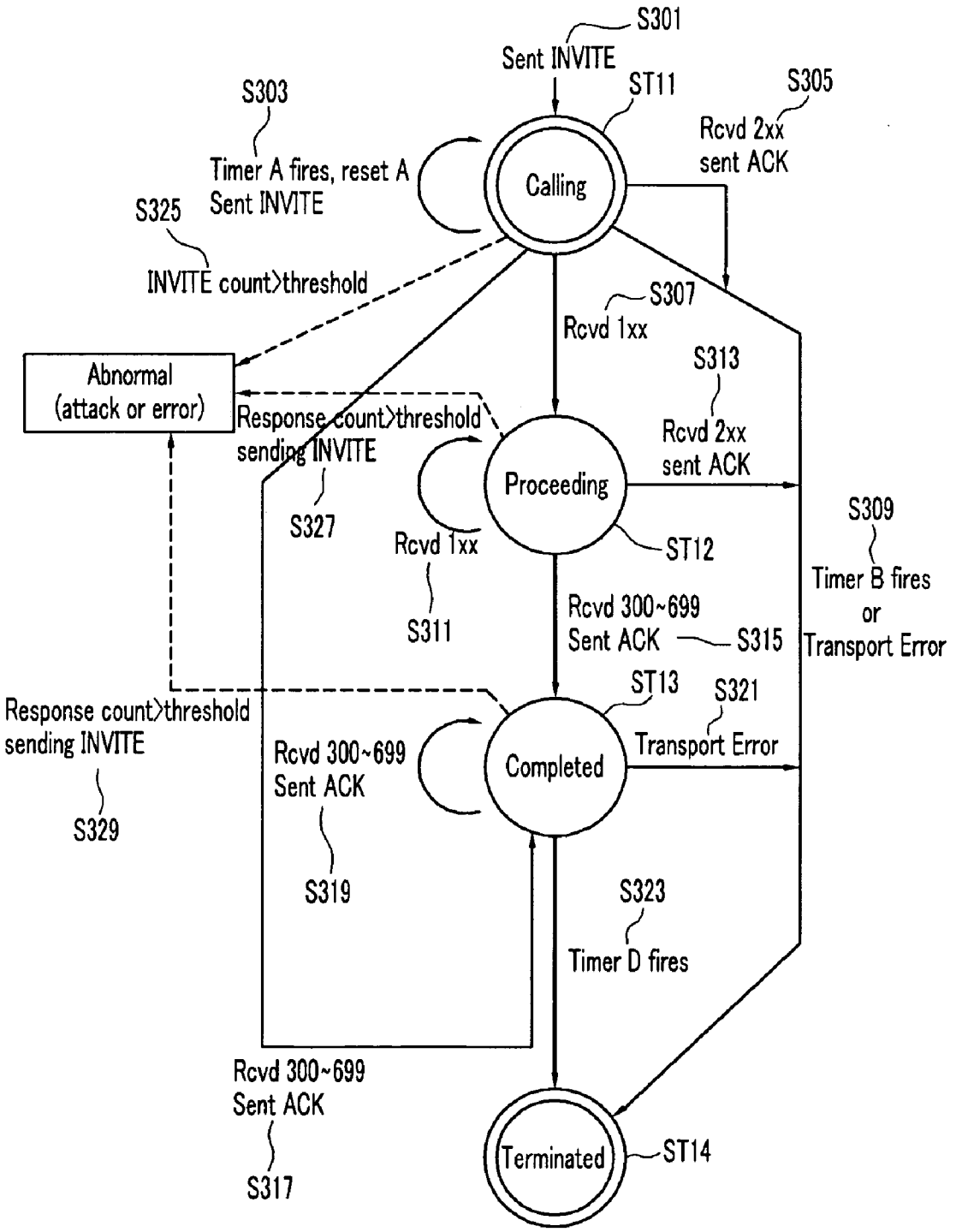

S301
Sent INVITE
ST11
S305
Rcvd 2xx
sent ACK
S303
Timer A fires, reset A
Sent INVITE
Calling
S325
INVITE count>threshold
S307
Rcvd 1xx
S313
Rcvd 2xx
sent ACK
Abnormal
(attack or error)
Response count>threshold
sending INVITE
S327
Proceeding
Rcvd 1xx
S311
ST12
S309
Timer B fires
or
Transport Error
Rcvd 300~699
Sent ACK
S315
ST13
S321
Transport Error
Response count>threshold
sending INVITE
S329
Rcvd 300~699
Sent ACK
S319
Completed
S323
Timer D fires
Rcvd 300~699
Sent ACK
S317
Terminated
ST14

FIG. 6

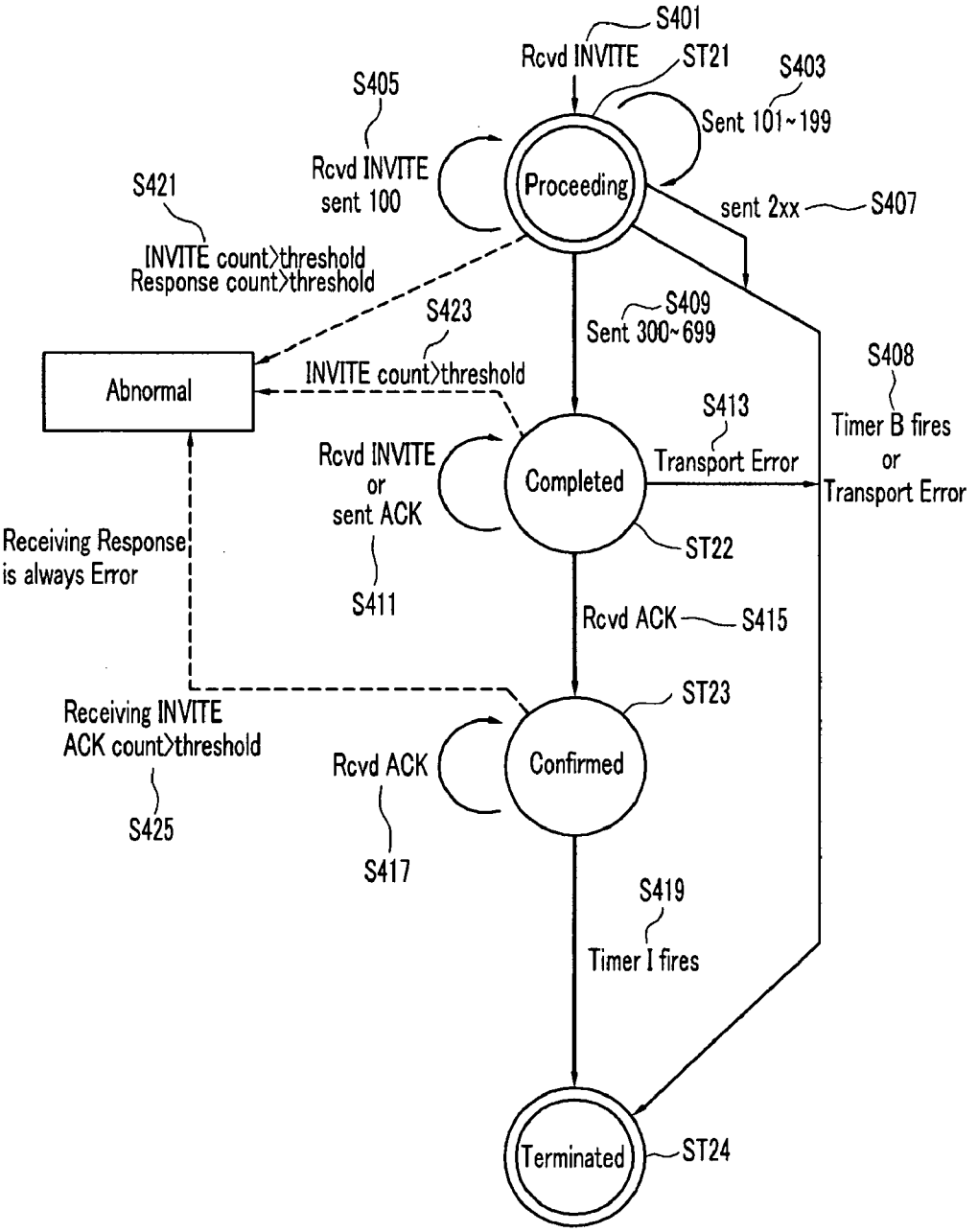

S401
Rcvd INVITE
ST21
S403
Sent 101~199
S405
Rcvd INVITE
sent 100
Proceeding
sent 2xx
S407
S421
INVITE count>threshold
Response count>threshold
S423
INVITE count>threshold
S409
Sent 300~699
S408
Timer B fires
or
Transport Error
Abnormal
S413
Transport Error
Rcvd INVITE
or
sent ACK
Completed
ST22
Receiving Response
is always Error
S411
Rcvd ACK
S415
ST23
Receiving INVITE
ACK count>threshold
Rcvd ACK
Confirmed
S425
S417
S419
Timer I fires
Terminated
ST24

FIG. 7

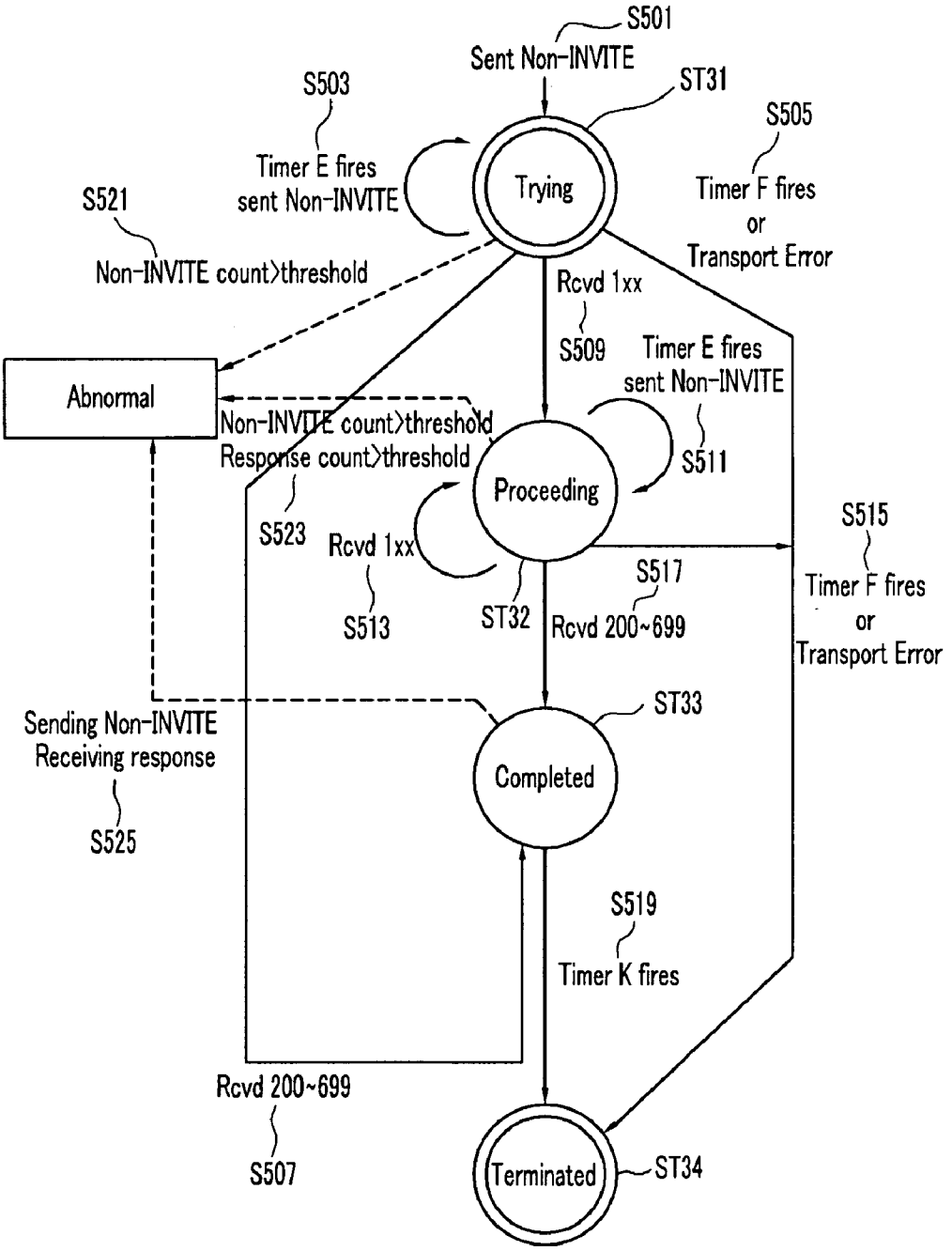

S501
Sent Non-INVITE
ST31
S503
Timer E fires
sent Non-INVITE
Trying
S505
Timer F fires
or
Transport Error
S521
Non-INVITE count>threshold
Rcvd 1xx
S509
Timer E fires
sent Non-INVITE
Abnormal
Non-INVITE count>threshold
Response count>threshold
S523
Proceeding
S511
S515
Timer F fires
or
Transport Error
Rcvd 1xx
S513
ST32
S517
Rcvd 200~699
ST33
Sending Non-INVITE
Receiving response
S525
Completed
S519
Timer K fires
Rcvd 200~699
S507
Terminated
ST34

FIG. 8

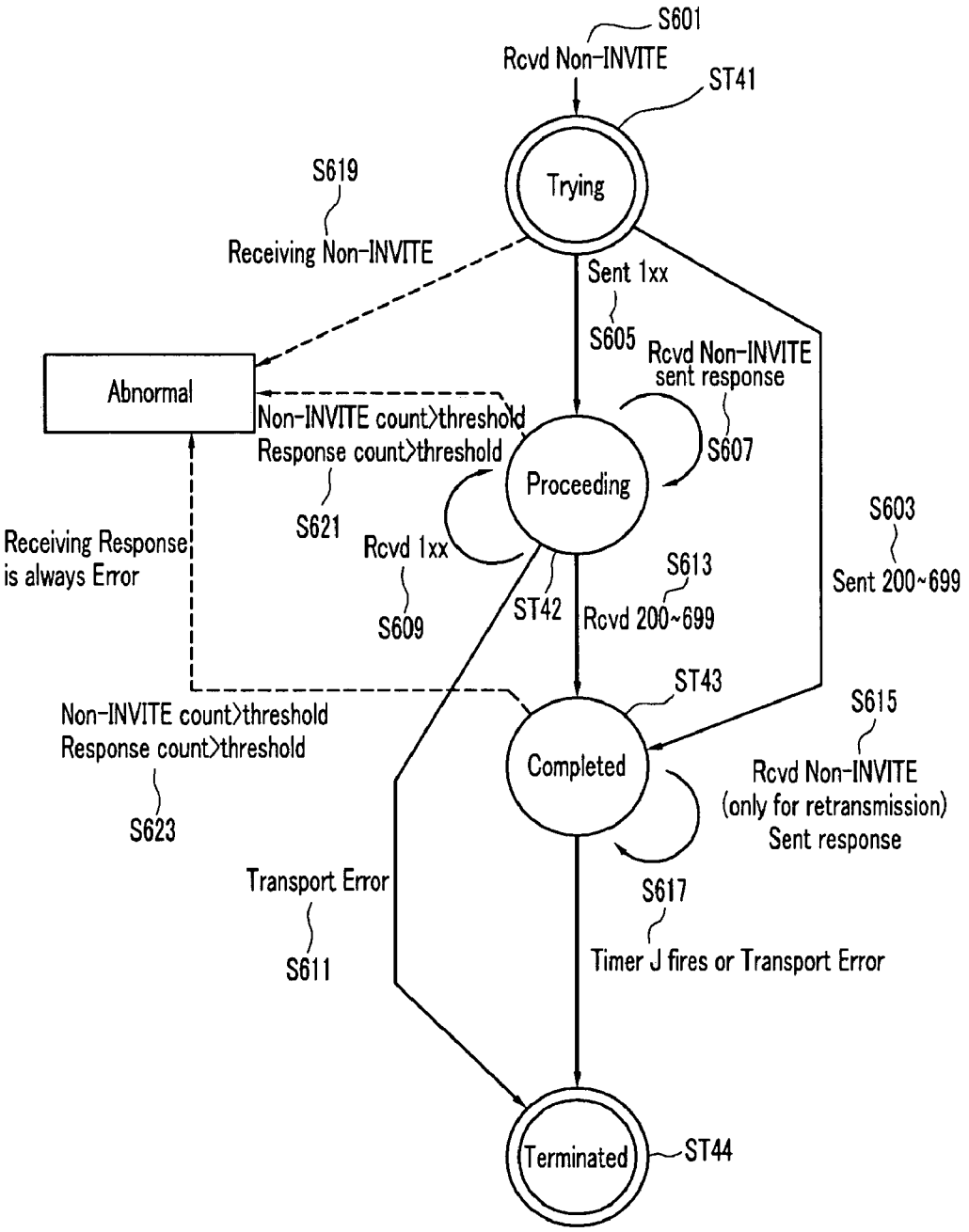
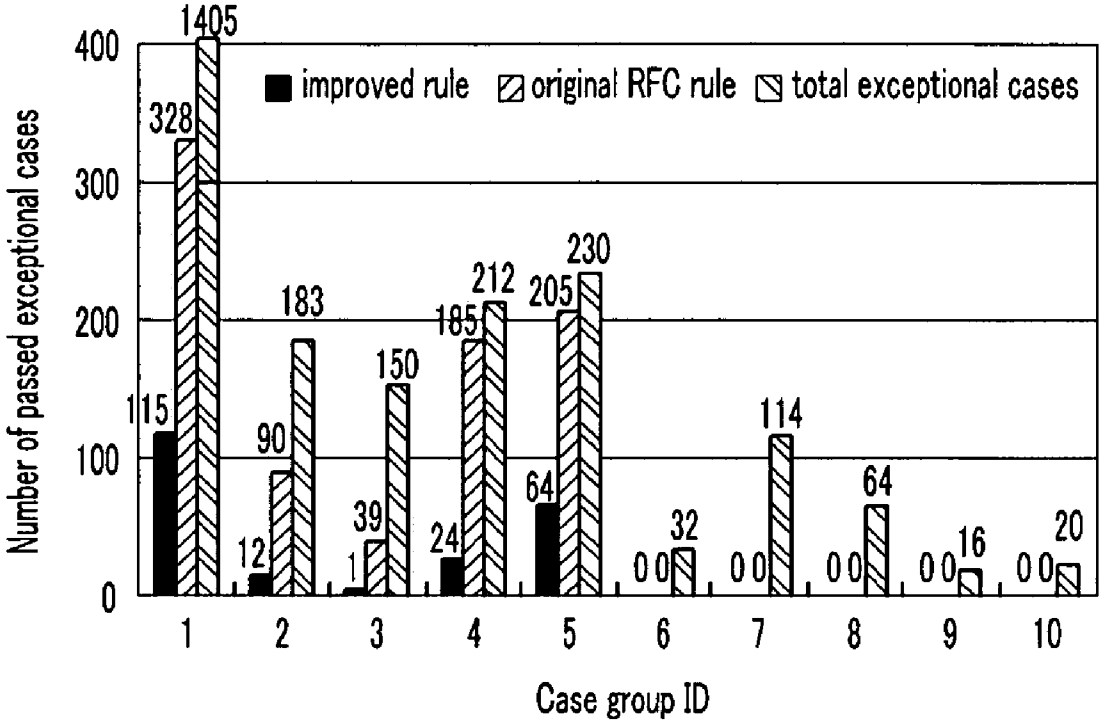

S601
Rcvd Non-INVITE
ST41
Trying
S619
Receiving Non-INVITE
Sent 1xx
S605
Rcvd Non-INVITE
sent response
S607
Abnormal
Non-INVITE count>threshold
Response count>threshold
S621
Proceeding
S603
Sent 200~699
Receiving Response
is always Error
Rcvd 1xx
S609
ST42
S613
Rcvd 200~699
ST43
S615
Non-INVITE count>threshold
Response count>threshold
S623
Completed
Rcvd Non-INVITE
(only for retransmission)
Sent response
Transport Error
S611
S617
Timer J fires or Transport Error
Terminated
ST44

FIG. 9


improved rule
original RFC rule
total exceptional cases
Number of passed exceptional cases
Case group ID
0
100
200
300
400
1
2
3
4
5
6
7
8
9
10
115
328
1405
12
90
183
1
39
150
24
185
212
64
205
230
0 0
32
0 0
114
0 0
64
0 0
16
0 0
20

METHOD AND COMPUTER-READABLE MEDIUM FOR DETECTING ABNORMAL PACKET IN VOIP

BACKGROUND OF THE INVENTION

[0001] (a) Field of the Invention

[0002] The present invention relates to a method for detecting abnormal packets in voice over Internet protocol (VoIP).

[0003] (b) Description of the Related Art

[0004] H.323 and a session initiation protocol (SIP) may be cited as examples of protocols for connecting a session in the voice over Internet protocol (VoIP). It is more complex to establish and disconnect connection in the H.323 than in the SIP, and it is more difficult to embody the H.323 than the SIP. Therefore, the SIP that has lower complexity and better extendibility than the H.323 is used in a service for the VoIP.

[0005] However, various weaknesses from offending VoIP users to paralyzing the VoIP service exist. An SIP attack and a real-time transportation protocol (RTP) attack may be cited as attacks of the VoIP. The RTP is a protocol for transmitting voice data after connecting a session.

[0006] An SIP flooding attack, a malformed SIP message attack, and a spoofing attack may be cited as SIP attacks. The SIP flooding attack is similar to a denial of service (DoS) attack, the malformed SIP message attack maliciously changes contents of the SIP header, and the spoofing attack hides or disguises information on an attacker. An RTP flooding attack, a media spamming attack and a man in the middle (MITM) attack may be cited as RTP attacks. The RTP flooding attack transmits a great quantity of voice information to a specified user, the media spamming attack transmits voice advertisements, and the MITM attack eavesdrops by intercepting packets transmitted between users.

[0007] Most VoIP attack tools existing in the Internet are packet generator s for enabling the SIP flooding attack or tools for randomly altering contents of an SIP header field. With regard to the spoofing attack and the MITM attack, their processes are complex and their success probability is low. However, with regard to an abnormal SIP message attack and the SIP message flooding attack, since public tools can vary attack methods and enable easy attacks, they are dangerous.

[0008] Research and papers regarding weakness of VoIP based on the SIP exist, but VoIP services embodying them do not exist or even if they do exist, the level is insufficient.

SUMMARY OF THE INVENTION

[0009] The present invention has been made in an effort to provide a method and a computer-readable medium storing a program for detecting weakness of voice services based on a packet such as in the VoIP.

[0010] According to an exemplary embodiment of the present invention, in a computer-readable medium that stores instructions executable by at least one processor to perform a method, the method comprises acquiring a packet; determining whether a header of the packet violates a first rule; if the header of the packet does not violate the first rule, confirming a session of the packet; confirming a state of the session; determining whether acquiring the packet in the state of the session violates a second rule; and if the second rule is violated, considering the packet to be abnormal.

[0011] In this case, the determining whether acquiring the packet in the state of the session violates the second rule may comprise determining whether the packet is acquired more than a predetermined number of times in the state of the session by acquiring the packet, and the considering the packet to be abnormal may comprise considering the packet to be abnormal if the packet is acquired more than the predetermined number of times in the state of the session.

[0012] Also, the determining whether the header of the packet violates the first rule may comprise determining whether each of fields of the header of the packet is out of a defined character length; and determining whether each of fields of the header of the packet includes at least one prohibited character.

[0013] A method for detecting an abnormal packet according to an exemplary embodiment of the present invention comprises acquiring a packet for session establishment; confirming a session to which the packet belongs; confirming a state of the session; determining whether the packet is received more than a predetermined number of times in the state of the session by acquiring the packet; and if the packet is received more than a predetermined number of times in the state of the session, considering the packet to be abnormal.

[0014] A method for detecting an abnormal packet according to an exemplary embodiment of the present invention comprises acquiring a packet for session establishment; determining whether each of fields of the header of the packet is out of a defined character length; determining whether the each of fields of the header of the packet includes at least one prohibited character; if each of the fields is not out of the defined character length and does not include at least one prohibited character, confirming a session to which the packet belongs; confirming a state of the session; determining whether acquiring the packet in the state of the session violates a predetermined rule; and if the predetermined rule is violated, considering the packet to be abnormal.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. **1** is a flow chart representing the calling procedure according to an exemplary embodiment of the present invention.

[0016] FIG. **2** is a block diagram representing the SIP exception detection module according to an exemplary embodiment of the present invention.

[0017] FIG. **3** is a flow chart representing the SIP exception detection method according to an exemplary embodiment of the present invention.

[0018] FIG. **4** is a drawing representing the SIP packet rules according to an exemplary embodiment of the present invention.

[0019] FIG. **5** is a state transition diagram representing the method for the INVITE client to manage the state of the session according to an exemplary embodiment of the present invention.

[0020] FIG. **6** is a state transition diagram representing the method for the INVITE server to manage the state of the session according to an exemplary embodiment of the present invention.

[0021] FIG. **7** is a state transition diagram representing the method for the non-INVITE client to manage the state of the session according to an exemplary embodiment of the present invention.

[0022] FIG. **8** is a state transition diagram representing the method for the non-INVITE server to manage the state of the session according to an exemplary embodiment of the present invention.

[0023] FIG. **9** is a graph showing the effectiveness of the exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0024] In the following detailed description, only certain exemplary embodiments of the present invention have been shown and described, simply by way of illustration. As those skilled in the art would realize, the described embodiments may be modified in various different ways, all without departing from the spirit or scope of the present invention. Accordingly, the drawings and description are to be regarded as illustrative in nature and not restrictive. Like reference numerals designate like elements throughout the specification.

[0025] Unless explicitly described to the contrary, the word "comprise" will be understood to imply the inclusion of stated elements but not the exclusion of any other elements. In addition, the word "unit" will be understood to be for processing a predetermined function or operation, which may be realized by hardware, software, or a combination thereof.

[0026] A calling procedure according to an exemplary embodiment of the present invention will be now described with reference to FIG. **1**.

[0027] FIG. **1** is a flow chart representing the calling procedure according to an exemplary embodiment of the present invention.

[0028] As shown in FIG. **1**, a system for SIP comprises a user agent client (UAC) **10** corresponding to a caller, a proxy server **20**, a proxy server **30** and a user agent server (UAS) **40** corresponding to a callee.

[0029] A request message and a response message may be cited as examples of an SIP message. For describing SIP messages, the UAC **10** will be called a client, and each of the proxy server **20**, the proxy server **30** and the UAS **40** will be called a server.

[0030] The request message is a message that the client transmits to a server. An INVITE request message, an ACK message, a BYE request message, a CANCEL request message, a REGISTER request message and an OPTIONS request message may be cited as examples of the request message.

[0031] The INVITE request message is a message that the UAC **10** transmits so as to invite the UAS **40**.

[0032] The ACK message is a message for confirming that the UAC **10** received an OK response message of the UAS **40**

[0033] The BYE request message is a message that the UAC **10** or the UAS **40** transmits to the opposite party so as to close a call.

[0034] The CANCEL request message is a message for withdrawing a request that is not already completed. For example, in a case in which the UAC **10** transmits the INVITE request message to the UAS **40** so as to request a call to the UAS **40**, the UAC **10** transmits the CANCEL request message so as to cancel the call.

[0035] The REGISTER request message is a message that the client transmits so as to register location information, an SIP address, or an Internet protocol (IP) address in the server.

[0036] The OPTIONS request message is a message that the client transmits so as to request information on the server.

[0037] In other words, the response message is a message that the server generally transmits to the client. A provisional response message, a success response message, a redirection response message, a client error response message, a server error response message and a global failure response message may be cited as examples of the response message.

[0038] The provisional response message is a message corresponding to a response code of 1XX (100~199). The provisional response message is a message that the server notes that it receives and processes the request message. A trying response message corresponding to the response code of **100** among the provisional response messages is a message for noting that the request message has not arrived at a final destination. The ringing response message corresponding to the response code of 180 among the provisional response messages is a message for noting that the request message has arrived at a final destination and is in a state of waiting for processing.

[0039] The success response message corresponding to the response code of 2XX (200~299) is a message for noting that the server has successfully received, understood, and accepted the request message. The OK response message corresponding to the response code of 200 among the success response messages is a message for noting that the UAS **40** has accepted the request.

[0040] The redirection response message corresponding to the response code of 3XX (300~399) is a message for noting that additional operations are needed for completing the request corresponding to the request message.

[0041] The client error response message corresponding to the response code of 4XX (400~499) is a message for noting that the request message includes errors or the server does not have the ability to process the request message.

[0042] The server error response message corresponding to the response code of 5XX (500~599) is a message for noting that the request message is valid but the server does not have the ability to process the request message.

[0043] The global failure response message corresponding to the response code of 6XX (600~699) is a message for noting that the request corresponding to the request message cannot be processed by any server.

[0044] As shown in FIG. **1**, the UAC **10** transmits the INVITE request message to the proxy server **20** in step S**101**.

[0045] Next, the proxy server **20** transmits the INVITE request message to the proxy server **30** in step S**103**, and transmits the trying response message to the UAC **10** in step S**105**.

[0046] The proxy server **30** transmits the INVITE request message to the UAS **40** in step S**107**, and transmits the trying response message to the proxy server **20** in step S**109**.

[0047] The UAS **40** that receives the INVITE request message transmits the ringing response message to the proxy server **30** in step S**111**. Next, the proxy server **30** that receives the ringing response message transmits the ringing response message to the proxy server **20** in step S**113**, and the proxy server **20** transmits the ringing response message to the UAC **10** in step S**115**.

[0048] If the UAS **40** accepts a call, the UAS **40** transmits the OK response message to the UAC **10** via the proxy server **30** and the proxy server **20** in steps S**117**, S**119** and S**121**.

[0049] After the UAC **10** that receives the OK response message transmits the ACK message to the UAS **40** in S**123**, the media session is generated between the UAC **10** and the UAS **40** in step S**125**.

[0050] Next, the UAC **10** trying to close the call transmits the BYE request message to the UAS **40** in step S**127**. After the UAS **40** transmits the OK response message to the UAC **10**, the media session is ended and the call is closed.

[0051] An SIP exception detection module according to an exemplary embodiment of the present invention will be now described with reference to FIG. **2**.

[0052] FIG. **2** is a block diagram representing the SIP exception detection module according to an exemplary embodiment of the present invention.

[0053] As shown in FIG. **2**, the SIP exception detection module **100** comprises a malformed SIP detection module **110**, a session management module **120**, an SIP flooding detection module **130**, an error management module **140** and a session information database **150**.

[0054] The SIP exception detection module **100** may be a inner module of the UAC **10**, the proxy server **20**, the proxy server **30** or the UAS **40**.

[0055] Each of the modules that the SIP exception detection module **100** comprises will be described with reference to FIG. **3**.

[0056] FIG. **3** is a flow chart representing the SIP exception detection method according to an exemplary embodiment of the present invention.

[0057] Firstly, the malformed SIP detection module **110** acquires a packet and confirms whether the packet is an SIP packet or not in step S**201**. If the packet is not an SIP packet, the malformed SIP detection module **110** discards the packet.

[0058] If the packet is an SIP packet, the malformed SIP detection module **110** confirms whether the packet satisfies SIP packet rules or not in step S**203**. If the packet does not satisfy the SIP packet rules, the malformed SIP detection module **110** discards the packet. If the packet satisfies the SIP packet rules, the malformed SIP detection module **110** transmits the packet to the session management module **120**.

[0059] The SIP packet rules will be now described with reference to FIG. **4**.

[0060] FIG. **4** is a drawing representing the SIP packet rules according to an exemplary embodiment of the present invention.

[0061] As shown in FIG. **4**, the SIP packet rules according to the exemplary embodiment of the present invention are made by modifying rules according to RFC 3261. Modified parts are gothically represented in FIG. **4**. The SIP packet rules according to the exemplary embodiment of the present invention comprise character length limit, special character length limit, etc.

[0062] The malformed SIP detection module **110** checks for violation of the character length limit, the special character limit, etc., of the header of the packet, so that it discards the packet if a violation exists. Concretely, the malformed SIP detection module **110** checks for an excess of the character length limit for each of fields of the header of the packet, so that if an excess exists it discards the packet. Also, the malformed SIP detection module **110** checks whether each of fields of the header of the packet includes a prohibited special character. If at least one of fields of the header includes a prohibited special character, the malformed SIP detection module **110** discards the packets.

[0063] Continually FIG. **3** will be now described.

[0064] The session management module **120** searches the session information database **150** to check generation of a session corresponding to the packet, in step S**205**.

[0065] If the session corresponding to the packet is not generated, in step S**205**, the session management module **120** generates the session corresponding to the packet, initiates a state of the session, and stores information on the session to the session information database **150** in step S**207**.

[0066] After establishing the session, the SIP flooding detection module **130** checks whether the packet is a request message or a response message in step S**209**.

[0067] If the packet is a response message, the SIP flooding detection module **130** checks whether the state of the session is an initial state in step S**211**. If the state of the session is the initial state, the SIP flooding detection module **130** ends the exception detection. If the state of the session is not the initial state, the SIP flooding detection module **130** checks whether the packet is a sending message or a receiving message in step S**213**.

[0068] On the other hand, if the session corresponding to the packet already exists in step S**205**, the session management module **120** checks whether the packet is a new request message of the session in step S**215**.

[0069] If the packet is a new request message of the session in step S**215**, the SIP flooding detection module **130** checks whether the packet is a sending message or a receiving message in step S**213**.

[0070] If the packet is the sending message in step S**213**, the SIP flooding detection module **130** checks whether the packet is the INVITE request message in step S**217**.

[0071] If the packet is an INVITE request message, the SIP flooding detection module **130** transacts the session corresponding to the packet as an INVITE client in step S**219**.

[0072] If the packet is not an INVITE request message, the SIP flooding detection module **130** transacts the session corresponding to the packet as a non-INVITE client in step S**221**.

[0073] If the packet is the receiving message in step S**213**, the SIP flooding detection module **130** checks whether the packet is an INVITE request message in step S**223**.

[0074] If the packet is an INVITE request message in step S**223**, the SIP flooding detection module **130** transacts the session corresponding to the packet as an INVITE server in step S**225**.

[0075] If the packet is not an INVITE request message in step S**223**, the SIP flooding detection module **130** transacts the session corresponding to the packet as a non-INVITE server in step S**227**.

[0076] Next, the SIP flooding detection module **130** manages the state of the session corresponding to the packet in step **229**. On the other hand, even if the packet is not a new request message of the session in step **215**, the SIP flooding detection module **130** manages the state of the session corresponding to the packet in step **229**.

[0077] A method for the SIP flooding detection module **130** to manage the state of the session according to an exemplary embodiment of the present invention will be now described with reference to FIG. **5** to FIG. **8**.

[0078] FIG. **5** is a state transition diagram representing the method for the INVITE client to manage the state of the session according to an exemplary embodiment of the present invention.

[0079] As shown in FIG. **5**, a calling state ST**11**, a proceeding state ST**12**, a completed state ST**13** and a terminated state ST**14** may be cited as states of the session of the INVITE client.

[0080] First, if the INVITE client sends the INVITE request message in the initial state, the INVITE client transits the state of the session to the calling state ST**11** in step S**301**.

[0081] If the timer A expires in the calling state ST**11**, the INVITE client resets the timer A and resends the INVITE request message in step S**303**. At this time, according to the RFC 3261, the timer A expires after T**1** (500 milliseconds).

[0082] If the INVITE client receives the success response message corresponding to the response code of 2XX in the calling state ST**11**, the INVITE client sends the ACK message and transits the state of the session to the terminated state ST**14** in step S**305**.

[0083] If the INVITE client receives the provisional response message corresponding to the response code of 1XX in the calling state ST**11**, the INVITE client transits the state of the session to the proceeding state ST**12** in step S**307**.

[0084] If a timer B expires in the calling state ST**11** or the INVITE client sends an error, the INVITE client transits the state of the session to the terminated state ST**14** in step S**309**. At this time, according to the RFC 3261, the timer B expires after T**1***64 milliseconds.

[0085] If the INVITE client receives the provisional response message corresponding to the response code of 1XX in the proceeding state ST**12**, the INVITE client maintains the state of the session as the proceeding state ST**12** in step S**311**.

[0086] If the INVITE client receives the success response message corresponding to the response code of 2XX in the proceeding state ST**12**, the INVITE client sends the ACK message and transits the state of the session to the terminated state ST**14** in step S**313**.

[0087] If the INVITE client receives the response message corresponding to the response code from 300 to 699 in the proceeding state ST**12**, the INVITE client sends the ACK message and transits the state of the session to the completed state ST**13** in step S**315**.

[0088] On the other hand, if the INVITE client receives the response message corresponding to the response code from 300 to 699 in the calling state ST**11**, the INVITE client sends the ACK message and transits the state of the session to the completed state ST**13** in step S**317**.

[0089] If the INVITE client receives the response message corresponding to the response code from 300 to 699 in the completed state ST**13**, the INVITE client sends the ACK message and maintains the state of the session as the completed state ST**13** in step S**319**.

[0090] If the INVITE client sends an error in the completed state ST**13**, the INVITE client transits the state of the session to the terminated state ST**14** in step S**321**.

[0091] If the timer D expires in the completed state ST**13**, the INVITE client transits the state of the session to the terminated state ST**14** in step S**323**. At this time, according to the RFC 3261, the timer D expires after 32,000 milliseconds.

[0092] If the INVITE client sends the INVITE request messages more than the predetermined number of times in the calling state ST**11**, the INVITE client determines the message as an abnormal message, an attack or an error in step S**325**.

[0093] If the INVITE client receives the response messages or sends the INVITE request messages more than the predetermined number of times in the proceeding state ST**12**, the INVITE client determines the message as an abnormal message, an attack or an error in step S**327**.

[0094] If the INVITE client receives the response messages or sends the INVITE request messages more than the predetermined number of times in the completed state ST**13**, the INVITE client determines the message as an abnormal message, an attack, or an error in step S**329**.

[0095] FIG. **6** is a state transition diagram representing the method for the INVITE server to manage the state of the session according to an exemplary embodiment of the present invention.

[0096] As shown in FIG. **6**, a proceeding state ST**21**, a completed state ST**22**, a confirmed state ST**23**, and a terminated state ST**24** may be cited as states of the session of the INVITE server.

[0097] If the INVITE server receives the INVITE request message in the initial state, the INVITE server transits the state of the session to the proceeding state ST**21** in step S**401**.

[0098] If the INVITE server receives the provisional response message corresponding to the response code of 1XX in the proceeding state ST**21**, the INVITE server maintains the state of the session as the proceeding state ST**21** in step S**403**.

[0099] If the INVITE server re-receives the INVITE request message in the proceeding state ST**21**, the INVITE server sends the trying response message corresponding to the response code of 100 and maintains the state of the session as the proceeding state ST**21** in step S**405**.

[0100] If the INVITE server sends the success response message corresponding to the response code of 2XX in the proceeding state ST**21**, the INVITE server transits the state of the session to the terminated state ST**24** in step S**407**.

[0101] If the timer B expires in the proceeding state ST**21** or the INVITE server sends an error, the INVITE server transits the state of the session to the terminated state ST**24** in step S**408**.

[0102] If the INVITE server sends the response message corresponding to the response code from 300 to 699 in the proceeding state ST**21**, the INVITE server transits the state of the session to the completed state ST**22** in step S**409**.

[0103] If the INVITE server receives the INVITE request message or sends the trying response message corresponding to the response code of 100 in the completed state ST**22**, the INVITE server transits the state of the session to the completed state ST**22** in step S**411**.

[0104] If the INVITE server sends an error in the completed state ST**22**, the INVITE server transits the state of the session to the terminated state ST**24** in step S**413**.

[0105] If the INVITE server receives the ACK message in the completed state ST**22**, the INVITE server transits the state of the session to the confirmed state ST**23** in step S**415**.

[0106] If the INVITE server re-receives the ACK message in the confirmed state ST**23**, the INVITE server maintains the state of the session to the confirmed state ST**23** in step S**417**.

[0107] If a timer I expires in the confirmed state ST**23**, the INVITE server transits the state of the session to the terminated state ST**24** in step S**419**. At this time, according to the RFC 3261, the timer I expires after T**4** (4000 milliseconds).

[0108] If the INVITE server receives the INVITE request messages or sends the response messages more than the predetermined number of times in the proceeding state ST**21**, the INVITE server determines the message as an abnormal message, an attack or an error in step S**421**.

[0109] If the INVITE server receives the INVITE request messages more than the predetermined number of times in the completed state ST**22**, the INVITE server determines the message as an abnormal message, an attack, or an error in step S**423**.

[0110] If the INVITE server receives the INVITE request messages or sends the response messages more than the predetermined number of times in the confirmed state ST**23**, the INVITE server determines the message as an abnormal message, an attack, or an error in step S**425**.

[0111] FIG. **7** is a state transition diagram representing the method for the non-INVITE client to manage the state of the session according to an exemplary embodiment of the present invention.

[0112] As shown in FIG. **7**, a trying state ST**31**, a proceeding state ST**32**, a completed state ST**33** and a terminated state ST**34** may be cited as states of the session of the non-INVITE client.

[0113] If the non-INVITE client sends the non-INVITE request message corresponding to the request message except the INVITE request message in the initial state, the non-INVITE client transits the state of the session to the trying state ST**31** in step S**501**.

[0114] If a timer E expires in the trying state ST**31**, the non-INVITE client re-sends the non-INVITE request message and maintains the state of the session to the trying state ST**31** in step S**503**. At this time, according to the RFC 3261, the timer E expires after T**1** (500 milliseconds).

[0115] If a timer F expires or the non-INVITE server sends an error in the trying state ST**31**, the non-INVITE server transits the state of the session to the terminated state ST**34** in step S**505**. At this time, according to the RFC 3261, the timer F expires after T**1***64 milliseconds.

[0116] If the non-INVITE client receives the response message corresponding to the response code from 200 to 699 in the trying state ST**31**, the non-INVITE client transits the state of the session to the completed state ST**33** in step S**507**.

[0117] If the non-INVITE client receives the provisional response message corresponding to the response code of 1XX in the trying state ST**31**, the non-INVITE client transits the state of the session to the proceeding state ST**32** in step S**509**.

[0118] If the timer E expires in the proceeding state ST**32**, the non-INVITE client sends the non-INVITE request message and maintains the state of the session to the proceeding state ST**32** in step S**511**.

[0119] If the non-INVITE client receives the provisional response message corresponding to the response code of 1XX in the proceeding state ST**32**, the non-INVITE client maintains the state of the session as the proceeding state ST**32** in step S**513**.

[0120] If the timer F expires or the non-INVITE client sends an error in the proceeding state ST**32**, the non-INVITE client transits the state of the session to the terminated state ST**34** in step S**515**.

[0121] If the non-INVITE client receives the response message corresponding to the response code from 200 to 699 in the proceeding state ST**32**, the non-INVITE client transits the state of the session to the completed state ST**33** in step S**517**.

[0122] If the timer K expires in the completed state ST**33**, the non-INVITE client transits the state of the session to the terminated state ST**34** in step S**519**. At this time, according to the RFC 3261, the timer K expires after T**4** (4,000 milliseconds).

[0123] If the non-INVITE client sends the non-INVITE request messages more than the predetermined number of times in the trying state ST**31**, the non-INVITE client determines the message as an abnormal message, an attack, or an error in step S**521**.

[0124] If the non-INVITE client sends the non-INVITE request messages or receives the response messages more than the predetermined number of times in the proceeding state ST**32**, the non-INVITE client determines the message as an abnormal message, an attack or an error in step S**523**.

[0125] If the non-INVITE client sends the non-INVITE request messages or receives the response messages in the completed state ST**33**, the non-INVITE client determines the message as an abnormal message, an attack, or an error in step S**525**.

[0126] FIG. **8** is a state transition diagram representing the method for the non-INVITE server to manage the state of the session according to an exemplary embodiment of the present invention.

[0127] As shown in FIG. **8**, a trying state ST**41**, a proceeding state ST**42**, a completed state ST**43**, and a terminated state ST**44** may be cited as states of the session of the non-INVITE server.

[0128] If the non-INVITE server receives the non-INVITE request message in the initial state, the non-INVITE server transits the state of the session to the trying state ST**41** in step S**601**.

[0129] If the non-INVITE server sends the response message corresponding to the response code from 200 to 699 in the trying state ST**41**, the non-INVITE server transits the state of the session to the completed state ST**43** in step S**603**.

[0130] If the non-INVITE server sends the provisional response message corresponding to the response code of 1XX in the trying state ST**41**, the non-INVITE server transits the state of the session to the proceeding state ST**42** in step S**605**.

[0131] If the non-INVITE server receives the non-INVITE request message in the proceeding state ST**42**, the non-INVITE server sends the response message and maintains the state of the session to the proceeding state ST**42** in step S**607**.

[0132] If the non-INVITE server sends the provisional response message corresponding to the response code of 1XX in the proceeding state ST**42**, the non-INVITE server maintains the state of the session as the proceeding state ST**42** in step S**609**.

[0133] If the non-INVITE server sends an error in the proceeding state ST**42**, the non-INVITE server transits the state of the session to the terminated state ST**44** in step S**611**.

[0134] If the non-INVITE server sends the response message corresponding to the response code from 200 to 699 in the proceeding state ST**42**, the non-INVITE server transits the state of the session to the completed state ST**43** in step S**613**.

[0135] If the non-INVITE server receives the non-INVITE request message in the completed state ST**43**, the non-INVITE server sends the response message and maintains the state of the session to the completed state ST**43** in step S**615**.

[0136] If a timer J expires or the non-INVITE server sends an error in the completed state ST**43**, the non-INVITE server transits the state of the session to the terminated state ST**44** in step S**617**. At this time, according to the RFC 3261, the timer J expires after T**1***64 milliseconds.

[0137] If the non-INVITE server receives the non-INVITE request messages in the trying state ST**41**, the non-INVITE server determines the message as an abnormal message, an attack, or an error in step S**619**.

[0138] If the non-INVITE server receives the non-INVITE request messages or sends the response messages more than the predetermined number of times in the proceeding state ST**42**, the non-INVITE server determines the message as an abnormal message, an attack or an error in step S**621**.

[0139] If the non-INVITE server receives the non-INVITE request messages or sends the response messages more than the predetermined number of times in the completed state

ST**43**, the non-INVITE server determines the message as an abnormal message, an attack or an error in step S**623**.

[0140] Continually FIG. **3** will be now described.

[0141] The error management module **140** receives information on the error from the SIP flooding detection module **130** in step S**231**. If the error management module **140** determines the received packet as an error, the error management module **140** discards the packet.

[0142] Next, in step S**233**, the error management module **140** determines whether the state of the session is the terminated state.

[0143] If the state of the session is the terminated state, the error management module **140** ends the session in step S**235**, and terminates proceeding with the packet.

[0144] The effectiveness of the method for detecting errors of the SIP packet according to an exemplary embodiment of the present invention will be now described with reference to FIG. **9**.

[0145] FIG. **9** is a graph showing the effectiveness of the exemplary embodiment of the present invention.

[0146] With reference to FIG. **9**, detecting exceptions according to the exemplary embodiment of the present invention is more effective than detecting exceptions according to the RFC 3261 defining the SIP.

[0147] In particular, to prove the effectiveness of the exemplary embodiment of the present invention, experiments using an SIP packet generator and an SIP attack detector were executed. 2,436 packets with altered headers were used for the abnormal SIP message attack. In the case of Simple RFC rules, 34.9% of attack packets were detected. However, in case of the exemplary embodiment of the present invention, only 10.67% of attack packets were not detected. Since normal packets exist among packets that were not detected, almost 100% of abnormal SIP messages may be detected.

[0148] For an experiment of an SIP flooding attack, five services that are free and are based on the SIP among VoIP services that are plentifully used in the whole world were selected. By analyzing transmission of the SIP packets in a normal situation with these services, a standard point of the SIP flooding attack may be established in the state transaction model. In normal cases the number of SIP packets that are transmitted or received in each of states is not over 8 per second per state. By regarding this value as the standard point, the possibility of detection of the SIP flooding attack was experimentally evaluated with 5 pps (packets per second) values of 1 pps, 3 pps, 5 pps, 10 pps and 34 pps. The case of 1 pps was not regarded as the SIP flooding attack. The cases of 3 pps, 5 pps, 10 pps and 34 pps were regarded as SIP flooding attacks after 3.1 seconds, 1.9 seconds, 0.8 seconds, and 0.2 seconds, respectively. Since the standard point is 8 pps per 1 state, the cases over 5 pps were regarded as SIP flooding attacks.

[0149] According to the exemplary embodiment of the present invention, abnormal packets may be easily detected in the voice service based on packets such as VoIP.

[0150] Also, according to the exemplary embodiment of the present invention, the abnormal SIP message attack and the SIP message flooding attack may be easily detected.

[0151] The above-described methods and apparatuses are not only realized by the exemplary embodiment of the present invention, but, on the contrary, are intended to be realized by a program for realizing functions corresponding to the configuration of the exemplary embodiment of the present invention or a recording medium for recording the program.

[0152] While this invention has been described in connection with what is presently considered to be practical exemplary embodiments, it is to be understood that the invention is not limited to the disclosed embodiments, but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

What is claimed is:

1. A computer-readable medium that stores instructions executable by at least one processor to perform a method comprising:

acquiring a packet;

determining whether a header of the packet violates a first rule;

if the header of the packet does not violate the first rule, confirming a session of the packet;

confirming a state of the session;

determining whether acquiring the packet in the state of the session violates a second rule; and

if the second rule is violated, considering the packet to be abnormal.

2. The computer-readable medium of claim **1**, wherein the determining whether acquiring the packet in the state of the session violates the second rule comprises:

determining whether the packet is acquired more than a predetermined number of times in the state of the session by acquiring the packet,

wherein the considering the packet to be abnormal comprises:

if the packet is acquired more than the predetermined number of times in the state of the session, considering the packet to be abnormal.

3. The computer-readable medium of claim **2**, wherein the determining whether the header of the packet violates the first rule comprises:

determining whether each of fields of the header of the packet is out of a defined character length; and

determining whether each of the fields of the header of the packet includes at least one prohibited character.

4. The computer-readable medium of claim **3**, wherein the method comprises:

if the state of the session is an initial state and a packet corresponding to the invite request message is sent, transiting the state of the session to a calling state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether the invite request message is sent more than the predetermined number of times in the calling state.

5. The computer-readable medium of claim **4**, wherein the method comprises:

if the state is the calling state and a packet corresponding to a provisional response message is received, transiting the state to a proceeding state; and

if the state is the proceeding state and a packet corresponding to a response message corresponding to an SIP response code from 300 to 699 is received, transiting the state to a completed state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether a packet corresponding to a response message is received more than the predetermined number of times in the proceeding state, and

determining whether a packet corresponding to a response message is received more than the predetermined number of times in the completed state.

6. The computer-readable medium of claim **3**, wherein the method comprises:

if the state is an initial state and a packet corresponding to an invite request message is received, transiting the state to a proceeding state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether the invite request message is received more than the predetermined number of times or a response message is sent more than the predetermined number of times in the proceeding state.

7. The computer-readable medium of claim **6**, wherein the method comprises:

if the state is the proceeding state and a packet corresponding to a response message corresponding to an SIP response code from 300 to 699 is received, transiting the state to a complete state; and

if a packet corresponding to an ACK message is received in the completed state, transiting the state to a confirmed state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether an invite request message is received more than the predetermined number of times in the completed state, and

determining whether the invite request message is received or the ACK message is received more than the predetermined number of times in the confirmed state.

8. The computer-readable medium of claim **3**, wherein the method comprises:

if the state is an initial state and a packet corresponding to an non-invite request message is sent, transiting the state to a trying state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether the non-invite request message is sent more than the predetermined number of times in the trying state.

9. The computer-readable medium of claim **8**, wherein the method comprises:

if a packet corresponding to a provisional response message is received in the trying state, transiting the state to a proceeding state; and

if a packet corresponding to a response message corresponding to an SIP response code from 200 to 699 is received in the proceeding state, transiting the state to a completed state, and

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether a non-invite request message is sent more than the predetermined number of times or a response message is received more than the predetermined number of times in the proceeding state, and

determining whether a non-invite request message is sent or a response message is received in the completed state.

10. The computer-readable medium of claim **3**, wherein the method comprises:

if the state is an initial state and a non-invite request message is received, transiting the state to a trying state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether the non-invite request message is received in the trying state.

11. The computer-readable medium of claim **10**, wherein the method comprises:

if a packet corresponding to a provisional response message is sent in the trying state, transiting the state to a proceeding state; and

if a packet corresponding to a response message corresponding to an SIP response code from 200 to 699 is sent in the proceeding state, transiting the state to a completed state,

wherein the determining whether the packet is acquired more than the predetermined number of times comprises:

determining whether a non-invite request message is received more than the predetermined number of times or a response message is sent more than the predetermined number of times in the proceeding state, and

determining whether a non-invite request message is received more than the predetermined number of times or a response message is sent more than the predetermined number of times in the completed state.

12. A method for detecting an abnormal packet, comprising:

acquiring a packet for session establishment;

confirming a session to which the packet belongs;

confirming a state of the session;

determining whether the packet is received more than a predetermined number of times in the state of the session by acquiring the packet; and

if the packet is received more than a predetermined number of times in the state of the session, considering the packet to be abnormal.

13. The method of claim **12**, further comprising:

if each of fields of the header of the packet is out of the defined character length, discarding the packet.

14. The method of claim **13**, further comprising:

if each of fields of the header of the packet includes at least one prohibited character, discarding the packet.

15. The method of claim **14**, further comprising:

if the packet is considered to be abnormal, discarding the packet.

16. A method for detecting an abnormal packet, comprising:

acquiring a packet for session establishment;

determining whether each of fields of the header of the packet is out of a defined character length;

determining whether each of the fields of the header of the packet includes at least one prohibited character;

if the each of fields is not out of the defined character length and does not include at least one prohibited character, confirming a session to which the packet belongs;

confirming a state of the session;

determining whether acquiring the packet in the state of the session violates a predetermined rule; and

if the predetermined rule is violated, considering the packet to be abnormal.

17. The method of claim **16**, wherein the determining whether acquiring the packet in the state of the session violates the predetermined rule comprises:

determining whether the packet is acquired more than a predetermined number of times in the state of the session by acquiring the packet,

wherein the considering the packet to be abnormal comprises:

if the packet is acquired more than the predetermined number of times in the state of the session, considering the packet to be abnormal.

* * * * *