US009270774B2

(12) **United States Patent**
**Jalan et al.**

(10) **Patent No.: US 9,270,774 B2**
(45) **Date of Patent: Feb. 23, 2016**

(54) **COMBINING STATELESS AND STATEFUL SERVER LOAD BALANCING**

(71) Applicant: **A10 Networks, Inc.**, San Jose, CA (US)

(72) Inventors: **Rajkumar Jalan**, Saratoga, CA (US); **Feilong Xu**, San Jose, CA (US); **Lalgudi Narayanan Kannan**, Los Altos, CA (US); **Ronald Wai Lun Szeto**, San Francisco, CA (US)

(73) Assignee: **A10 Networks, Inc.**, San Jose, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/520,126**

(22) Filed: **Oct. 21, 2014**

(65) **Prior Publication Data**

US 2015/0039671 A1 Feb. 5, 2015

**Related U.S. Application Data**

(63) Continuation of application No. 13/280,336, filed on Oct. 24, 2011, now Pat. No. 8,897,154.

(51) **Int. Cl.**
***H04L 12/66*** (2006.01)
***H04L 29/08*** (2006.01)

(52) **U.S. Cl.**
CPC ..... ***H04L 67/28*** (2013.01); ***H04L 12/66*** (2013.01); ***H04L 67/10*** (2013.01); ***H04L 67/2819*** (2013.01); *H04L 67/1027* (2013.01); *H04L 67/142* (2013.01)

(58) **Field of Classification Search**
CPC ..... H04L 67/28; H04L 67/10; H04L 67/142; H04L 12/66; H04L 67/025; G06F 9/505; G06F 9/5055
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,218,602 | A | 6/1993 | Grant et al. |
| 5,774,660 | A | 6/1998 | Brendel et al. |
| 5,935,207 | A | 8/1999 | Logue et al. |
| 5,958,053 | A | 9/1999 | Denker |
| 6,003,069 | A | 12/1999 | Cavill |
| 6,047,268 | A | 4/2000 | Bartoli et al. |

(Continued)

FOREIGN PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| CN | 1372662 | A | 10/2002 |
| CN | 1449618 | | 10/2003 |

(Continued)

OTHER PUBLICATIONS

Cardellini et al., "Dynamic Load Balancing on Web-server Systems", IEEE Internet Computing, vol. 3, No. 3, pp. 28-39, May-Jun. 1999.

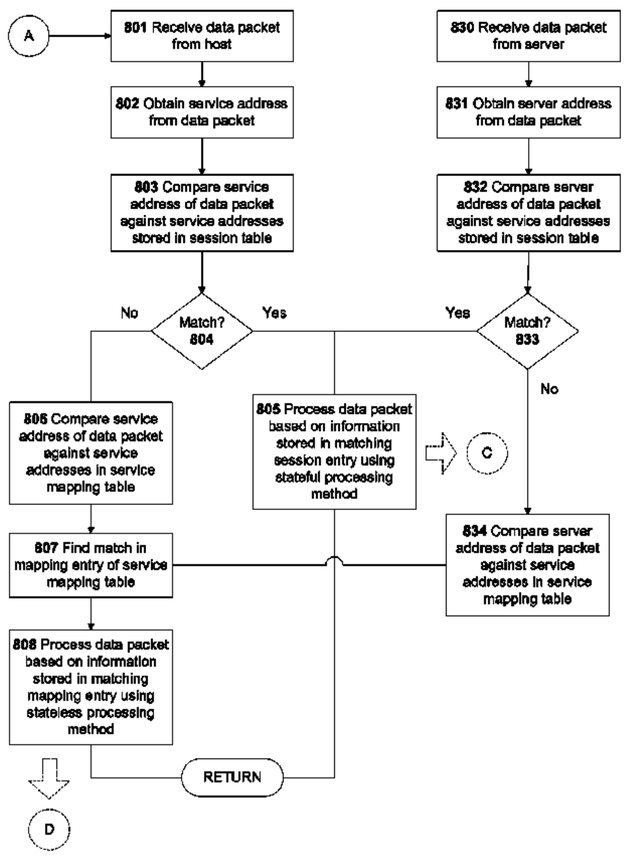
(Continued)

*Primary Examiner* — Brian O'Connor
(74) *Attorney, Agent, or Firm* — Carr & Ferrell LLP

(57) **ABSTRACT**

The processing of data packets sent over a communication session between a host and a server by a service gateway includes processing a data packet using a current hybrid-stateful or hybrid-stateless processing method. The processing then checks whether a hybrid-stateless or hybrid-stateful condition is satisfied. When one of the sets of conditions is satisfied, the process includes changing from a hybrid-stateful to a hybrid-stateless processing method, or vice versa, for a subsequently received data packet. If the conditions are not satisfied, the process continues as originally structured.

**15 Claims, 11 Drawing Sheets**



(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 6,131,163 | A | 10/2000 | Wiegel | |
| 6,219,706 | B1 | 4/2001 | Fan et al. | |
| 6,259,705 | B1 | 7/2001 | Takahashi et al. | |
| 6,321,338 | B1 | 11/2001 | Porras et al. | |
| 6,374,300 | B2 | 4/2002 | Masters | |
| 6,587,866 | B1 | 7/2003 | Modi et al. | |
| 6,748,414 | B1 | 6/2004 | Bournas | |
| 6,772,334 | B1 | 8/2004 | Glawitsch | |
| 6,779,033 | B1 | 8/2004 | Watson et al. | |
| 7,010,605 | B1 | 3/2006 | Dharmarajan | |
| 7,013,482 | B1 * | 3/2006 | Krumel | H04L 63/0227 709/229 |
| 7,058,718 | B2 | 6/2006 | Fontes et al. | |
| 7,069,438 | B2 | 6/2006 | Balabine et al. | |
| 7,076,555 | B1 | 7/2006 | Orman et al. | |
| 7,143,087 | B2 | 11/2006 | Fairweather | |
| 7,181,524 | B1 | 2/2007 | Lele | |
| 7,228,359 | B1 | 6/2007 | Monteiro | |
| 7,234,161 | B1 | 6/2007 | Maufer et al. | |
| 7,236,457 | B2 | 6/2007 | Joe | |
| 7,254,133 | B2 | 8/2007 | Govindarajan et al. | |
| 7,269,850 | B2 | 9/2007 | Govindarajan et al. | |
| 7,277,963 | B2 | 10/2007 | Dolson et al. | |
| 7,301,899 | B2 | 11/2007 | Goldstone | |
| 7,308,499 | B2 | 12/2007 | Chavez | |
| 7,310,686 | B2 | 12/2007 | Uysal | |
| 7,328,267 | B1 | 2/2008 | Bashyam et al. | |
| 7,334,232 | B2 * | 2/2008 | Jacobs et al. | 719/313 |
| 7,337,241 | B2 | 2/2008 | Boucher et al. | |
| 7,343,399 | B2 | 3/2008 | Hayball et al. | |
| 7,349,970 | B2 * | 3/2008 | Clement et al. | 709/228 |
| 7,370,353 | B2 | 5/2008 | Yang | |
| 7,391,725 | B2 | 6/2008 | Huitema et al. | |
| 7,398,317 | B2 | 7/2008 | Chen et al. | |
| 7,423,977 | B1 | 9/2008 | Joshi | |
| 7,430,755 | B1 | 9/2008 | Hughes et al. | |
| 7,467,202 | B2 | 12/2008 | Savchuk | |
| 7,472,190 | B2 * | 12/2008 | Robinson | 709/225 |
| 7,492,766 | B2 * | 2/2009 | Cabeca | H04L 12/4645 370/389 |
| 7,506,360 | B1 | 3/2009 | Wilkinson et al. | |
| 7,512,980 | B2 | 3/2009 | Copeland et al. | |
| 7,533,409 | B2 | 5/2009 | Keane et al. | |
| 7,552,323 | B2 | 6/2009 | Shay | |
| 7,584,262 | B1 | 9/2009 | Wang et al. | |
| 7,590,736 | B2 | 9/2009 | Hydrie et al. | |
| 7,613,193 | B2 | 11/2009 | Swami et al. | |
| 7,613,822 | B2 | 11/2009 | Joy et al. | |
| 7,673,072 | B2 | 3/2010 | Boucher et al. | |
| 7,675,854 | B2 | 3/2010 | Chen et al. | |
| 7,707,295 | B1 | 4/2010 | Szeto et al. | |
| 7,711,790 | B1 | 5/2010 | Barrett et al. | |
| 7,747,748 | B2 | 6/2010 | Allen | |
| 7,792,113 | B1 | 9/2010 | Foschiano et al. | |
| 7,808,994 | B1 * | 10/2010 | Vinokour | H04L 12/4645 370/392 |
| 7,826,487 | B1 | 11/2010 | Mukerji et al. | |
| 7,881,215 | B1 | 2/2011 | Daigle et al. | |
| 7,970,934 | B1 | 6/2011 | Patel | |
| 7,983,258 | B1 * | 7/2011 | Ruben | H04L 12/2859 370/389 |
| 7,990,847 | B1 | 8/2011 | Leroy et al. | |
| 7,991,859 | B1 | 8/2011 | Miller et al. | |
| 8,090,866 | B1 | 1/2012 | Bashyam et al. | |
| 8,122,116 | B2 | 2/2012 | Matsunaga et al. | |
| 8,185,651 | B2 | 5/2012 | Moran et al. | |
| 8,191,106 | B2 | 5/2012 | Choyi et al. | |
| 8,224,971 | B1 | 7/2012 | Miller et al. | |
| 8,296,434 | B1 | 10/2012 | Miller et al. | |
| 8,312,507 | B2 | 11/2012 | Chen et al. | |
| 8,379,515 | B1 | 2/2013 | Mukerji | |
| 8,499,093 | B2 * | 7/2013 | Grosser | H04L 61/103 370/392 |
| 8,554,929 | B1 | 10/2013 | Szeto et al. | |
| 8,560,693 | B1 | 10/2013 | Wang et al. | |
| 8,584,199 | B1 | 11/2013 | Chen et al. | |
| 8,595,791 | B1 | 11/2013 | Chen et al. | |
| RE44,701 | E | 1/2014 | Chen et al. | |
| 8,681,610 | B1 | 3/2014 | Mukerji | |
| 8,782,221 | B2 | 7/2014 | Han | |
| 8,813,180 | B1 | 8/2014 | Chen et al. | |
| 8,826,372 | B1 | 9/2014 | Chen et al. | |
| 8,879,427 | B2 * | 11/2014 | Krumel | H04L 29/06 370/255 |
| 8,885,463 | B1 | 11/2014 | Medved et al. | |
| 8,897,154 | B2 | 11/2014 | Jalan et al. | |
| 8,965,957 | B2 * | 2/2015 | Barros | 709/203 |
| 8,977,749 | B1 | 3/2015 | Han | |
| 8,990,262 | B2 * | 3/2015 | Chen et al. | 707/802 |
| 9,094,364 | B2 * | 7/2015 | Jalan et al. | |
| 9,106,561 | B2 | 8/2015 | Jalan et al. | |
| 9,154,584 | B1 | 10/2015 | Han | |
| 2001/0049741 | A1 | 12/2001 | Skene et al. | |
| 2002/0032777 | A1 | 3/2002 | Kawata et al. | |
| 2002/0078164 | A1 | 6/2002 | Reinschmidt | |
| 2002/0091844 | A1 | 7/2002 | Craft et al. | |
| 2002/0103916 | A1 | 8/2002 | Chen et al. | |
| 2002/0133491 | A1 | 9/2002 | Sm et al. | |
| 2002/0138618 | A1 | 9/2002 | Szabo | |
| 2002/0143991 | A1 | 10/2002 | Chow et al. | |
| 2002/0178259 | A1 | 11/2002 | Doyle et al. | |
| 2002/0194335 | A1 | 12/2002 | Maynard | |
| 2002/0194350 | A1 | 12/2002 | Lu et al. | |
| 2003/0009591 | A1 | 1/2003 | Hayball et al. | |
| 2003/0014544 | A1 | 1/2003 | Pettey | |
| 2003/0023711 | A1 | 1/2003 | Parmar et al. | |
| 2003/0023873 | A1 | 1/2003 | Ben-Itzhak | |
| 2003/0035409 | A1 | 2/2003 | Wang et al. | |
| 2003/0035420 | A1 | 2/2003 | Niu | |
| 2003/0131245 | A1 | 7/2003 | Linderman | |
| 2003/0135625 | A1 | 7/2003 | Fontes et al. | |
| 2003/0195962 | A1 | 10/2003 | Kikuchi et al. | |
| 2004/0062246 | A1 | 4/2004 | Boucher et al. | |
| 2004/0073703 | A1 | 4/2004 | Boucher et al. | |
| 2004/0078419 | A1 | 4/2004 | Ferrari et al. | |
| 2004/0078480 | A1 | 4/2004 | Boucher et al. | |
| 2004/0111516 | A1 | 6/2004 | Cain | |
| 2004/0187032 | A1 | 9/2004 | Gels et al. | |
| 2004/0199616 | A1 | 10/2004 | Karhu | |
| 2004/0199646 | A1 | 10/2004 | Susai et al. | |
| 2004/0202182 | A1 | 10/2004 | Lund et al. | |
| 2004/0210663 | A1 | 10/2004 | Phillips et al. | |
| 2004/0213158 | A1 | 10/2004 | Collett et al. | |
| 2005/0005207 | A1 | 1/2005 | Herneque | |
| 2005/0009520 | A1 | 1/2005 | Herrero et al. | |
| 2005/0021848 | A1 | 1/2005 | Jorgenson | |
| 2005/0027862 | A1 | 2/2005 | Nguyen et al. | |
| 2005/0036501 | A1 | 2/2005 | Chung et al. | |
| 2005/0036511 | A1 | 2/2005 | Baratakke et al. | |
| 2005/0044270 | A1 | 2/2005 | Grove et al. | |
| 2005/0074013 | A1 | 4/2005 | Hershey et al. | |
| 2005/0080890 | A1 | 4/2005 | Yang et al. | |
| 2005/0102400 | A1 | 5/2005 | Nakahara et al. | |
| 2005/0125276 | A1 | 6/2005 | Rusu | |
| 2005/0163073 | A1 | 7/2005 | Heller et al. | |
| 2005/0198335 | A1 | 9/2005 | Brown et al. | |
| 2005/0213586 | A1 | 9/2005 | Cyganski et al. | |
| 2005/0240989 | A1 | 10/2005 | Kim et al. | |
| 2005/0249225 | A1 | 11/2005 | Singhal | |
| 2006/0023721 | A1 | 2/2006 | Miyake et al. | |
| 2006/0036610 | A1 | 2/2006 | Wang | |
| 2006/0036733 | A1 | 2/2006 | Fujimoto et al. | |
| 2006/0069774 | A1 * | 3/2006 | Chen et al. | 709/225 |
| 2006/0069804 | A1 | 3/2006 | Miyake et al. | |
| 2006/0077926 | A1 | 4/2006 | Rune | |
| 2006/0092950 | A1 | 5/2006 | Arregoces et al. | |
| 2006/0098645 | A1 | 5/2006 | Walkin | |
| 2006/0168319 | A1 | 7/2006 | Trossen | |
| 2006/0187901 | A1 | 8/2006 | Cortes et al. | |
| 2006/0190997 | A1 | 8/2006 | Mahajani et al. | |
| 2006/0251057 | A1 | 11/2006 | Kwon et al. | |
| 2006/0277303 | A1 | 12/2006 | Hegde et al. | |
| 2006/0280121 | A1 | 12/2006 | Matoba | |
| 2007/0019543 | A1 | 1/2007 | Wei et al. | |

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 2007/0118881 | A1 | 5/2007 | Mitchell et al. | |
| 2007/0156919 | A1 | 7/2007 | Potti et al. | |
| 2007/0185998 | A1 | 8/2007 | Touitou et al. | |
| 2007/0195792 | A1 | 8/2007 | Chen et al. | |
| 2007/0230337 | A1 | 10/2007 | Igarashi et al. | |
| 2007/0245090 | A1 | 10/2007 | King et al. | |
| 2007/0259673 | A1 | 11/2007 | Willars et al. | |
| 2007/0283429 | A1 | 12/2007 | Chen et al. | |
| 2007/0286077 | A1 | 12/2007 | Wu | |
| 2007/0288247 | A1 | 12/2007 | MacKay | |
| 2007/0294209 | A1 | 12/2007 | Strub et al. | |
| 2008/0031263 | A1 | 2/2008 | Ervin et al. | |
| 2008/0101396 | A1 | 5/2008 | Miyata | |
| 2008/0109452 | A1 | 5/2008 | Patterson | |
| 2008/0109870 | A1 | 5/2008 | Sherlock et al. | |
| 2008/0134332 | A1 | 6/2008 | Keohane et al. | |
| 2008/0228781 | A1* | 9/2008 | Chen et al. | 707/10 |
| 2008/0250099 | A1 | 10/2008 | Shen et al. | |
| 2008/0291911 | A1 | 11/2008 | Lee et al. | |
| 2009/0049198 | A1 | 2/2009 | Blinn et al. | |
| 2009/0070470 | A1 | 3/2009 | Bauman et al. | |
| 2009/0077651 | A1 | 3/2009 | Poeluev | |
| 2009/0092124 | A1 | 4/2009 | Singhal et al. | |
| 2009/0106830 | A1 | 4/2009 | Maher | |
| 2009/0138606 | A1 | 5/2009 | Moran et al. | |
| 2009/0138945 | A1 | 5/2009 | Savchuk | |
| 2009/0141634 | A1 | 6/2009 | Rothstein et al. | |
| 2009/0164614 | A1 | 6/2009 | Christian et al. | |
| 2009/0172093 | A1 | 7/2009 | Matsubara | |
| 2009/0213858 | A1 | 8/2009 | Dolganow et al. | |
| 2009/0222583 | A1 | 9/2009 | Josefsberg et al. | |
| 2009/0228547 | A1 | 9/2009 | Miyaoka et al. | |
| 2010/0008229 | A1 | 1/2010 | Bi et al. | |
| 2010/0036952 | A1 | 2/2010 | Hazlewood et al. | |
| 2010/0054139 | A1 | 3/2010 | Chun et al. | |
| 2010/0061319 | A1 | 3/2010 | Aso et al. | |
| 2010/0064008 | A1 | 3/2010 | Yan et al. | |
| 2010/0083076 | A1 | 4/2010 | Ushiyama | |
| 2010/0094985 | A1 | 4/2010 | Abu-Samaha et al. | |
| 2010/0106833 | A1 | 4/2010 | Banerjee et al. | |
| 2010/0106854 | A1 | 4/2010 | Kim et al. | |
| 2010/0162378 | A1 | 6/2010 | Jayawardena et al. | |
| 2010/0210265 | A1 | 8/2010 | Borzsei et al. | |
| 2010/0217793 | A1 | 8/2010 | Preiss | |
| 2010/0223630 | A1* | 9/2010 | Degenkolb et al. | 719/320 |
| 2010/0228819 | A1 | 9/2010 | Wei | |
| 2010/0235507 | A1 | 9/2010 | Szeto et al. | |
| 2010/0235522 | A1 | 9/2010 | Chen et al. | |
| 2010/0235880 | A1 | 9/2010 | Chen et al. | |
| 2010/0238828 | A1 | 9/2010 | Russell | |
| 2010/0265824 | A1 | 10/2010 | Chao et al. | |
| 2010/0268814 | A1 | 10/2010 | Cross et al. | |
| 2010/0293296 | A1 | 11/2010 | Hsu et al. | |
| 2010/0312740 | A1 | 12/2010 | Clemm et al. | |
| 2010/0318631 | A1 | 12/2010 | Shukla | |
| 2010/0322252 | A1 | 12/2010 | Suganthi et al. | |
| 2010/0330971 | A1 | 12/2010 | Selitser et al. | |
| 2010/0333101 | A1 | 12/2010 | Pope et al. | |
| 2011/0007652 | A1 | 1/2011 | Bai | |
| 2011/0023071 | A1 | 1/2011 | Li et al. | |
| 2011/0029599 | A1 | 2/2011 | Pulleyn et al. | |
| 2011/0032941 | A1 | 2/2011 | Quach et al. | |
| 2011/0040826 | A1* | 2/2011 | Chadzelek et al. | 709/203 |
| 2011/0047294 | A1 | 2/2011 | Singh et al. | |
| 2011/0060831 | A1 | 3/2011 | Ishii et al. | |
| 2011/0093522 | A1 | 4/2011 | Chen et al. | |
| 2011/0110294 | A1 | 5/2011 | Valluri et al. | |
| 2011/0145324 | A1 | 6/2011 | Reinart et al. | |
| 2011/0153834 | A1 | 6/2011 | Bharrat | |
| 2011/0185073 | A1 | 7/2011 | Jagadeeswaran et al. | |
| 2011/0191773 | A1 | 8/2011 | Pavel et al. | |
| 2011/0196971 | A1 | 8/2011 | Reguraman et al. | |
| 2011/0276695 | A1 | 11/2011 | Maldaner | |
| 2011/0276982 | A1 | 11/2011 | Nakayama et al. | |
| 2011/0289496 | A1 | 11/2011 | Steer | |
| 2011/0292939 | A1 | 12/2011 | Subramaian et al. | |
| 2011/0302256 | A1 | 12/2011 | Sureshehandra et al. | |
| 2011/0307541 | A1 | 12/2011 | Walsh et al. | |
| 2012/0023231 | A1 | 1/2012 | Ueno | |
| 2012/0030341 | A1 | 2/2012 | Jensen et al. | |
| 2012/0066371 | A1 | 3/2012 | Patel et al. | |
| 2012/0084419 | A1 | 4/2012 | Kannan et al. | |
| 2012/0084460 | A1 | 4/2012 | McGinnity et al. | |
| 2012/0144014 | A1 | 6/2012 | Natham et al. | |
| 2012/0144015 | A1 | 6/2012 | Jalan et al. | |
| 2012/0170548 | A1 | 7/2012 | Rajagopalan et al. | |
| 2012/0173759 | A1 | 7/2012 | Agarwal et al. | |
| 2012/0191839 | A1 | 7/2012 | Maynard | |
| 2012/0240185 | A1 | 9/2012 | Kapoor et al. | |
| 2012/0290727 | A1 | 11/2012 | Tivig | |
| 2012/0297046 | A1 | 11/2012 | Raja et al. | |
| 2013/0046876 | A1 | 2/2013 | Narayana et al. | |
| 2013/0074177 | A1 | 3/2013 | Varadhan et al. | |
| 2013/0083725 | A1 | 4/2013 | Mallya et al. | |
| 2013/0100958 | A1 | 4/2013 | Jalan et al. | |
| 2013/0136139 | A1 | 5/2013 | Zheng et al. | |
| 2013/0148500 | A1 | 6/2013 | Sonoda et al. | |
| 2013/0166762 | A1 | 6/2013 | Jalan et al. | |
| 2013/0173795 | A1 | 7/2013 | McPherson | |
| 2013/0176854 | A1 | 7/2013 | Chisu et al. | |
| 2013/0191486 | A1 | 7/2013 | Someya et al. | |
| 2013/0198385 | A1 | 8/2013 | Han et al. | |
| 2013/0250765 | A1 | 9/2013 | Ehsan et al. | |
| 2013/0258846 | A1 | 10/2013 | Damola | |
| 2014/0012972 | A1 | 1/2014 | Han | |
| 2014/0089500 | A1 | 3/2014 | Sankar et al. | |
| 2014/0164617 | A1 | 6/2014 | Jalan et al. | |
| 2014/0169168 | A1 | 6/2014 | Jalan et al. | |
| 2014/0207845 | A1 | 7/2014 | Han et al. | |
| 2014/0258536 | A1 | 9/2014 | Chiong | |
| 2014/0269728 | A1 | 9/2014 | Jalan et al. | |
| 2014/0286313 | A1 | 9/2014 | Fu et al. | |
| 2014/0330982 | A1 | 11/2014 | Jalan et al. | |
| 2014/0334485 | A1 | 11/2014 | Jain et al. | |
| 2014/0359052 | A1 | 12/2014 | Joachimpillai et al. | |
| 2015/0156223 | A1 | 6/2015 | Xu et al. | |
| 2015/0237173 | A1 | 8/2015 | Virkki et al. | |
| 2015/0281087 | A1 | 10/2015 | Jalan et al. | |
| 2015/0281104 | A1 | 10/2015 | Golshan et al. | |
| 2015/0296058 | A1 | 10/2015 | Jalan et al. | |

FOREIGN PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| CN | 1529460 | | 9/2004 |
| CN | 1575582 | | 2/2005 |
| CN | 1714545 | A | 12/2005 |
| CN | 1725702 | | 1/2006 |
| CN | 101004740 | A | 7/2007 |
| CN | 101094225 | | 12/2007 |
| CN | 101169785 | A | 4/2008 |
| CN | 101189598 | | 5/2008 |
| CN | 101247349 | A | 8/2008 |
| CN | 101261644 | A | 9/2008 |
| CN | 102546590 | | 7/2012 |
| CN | 102571742 | | 7/2012 |
| CN | 102577252 | | 7/2012 |
| CN | 102918801 | | 2/2013 |
| CN | 103533018 | | 1/2014 |
| CN | 103944954 | | 7/2014 |
| CN | 104040990 | | 9/2014 |
| CN | 104067569 | | 9/2014 |
| CN | 104106241 | | 10/2014 |
| CN | 104137491 | | 11/2014 |
| CN | 104796396 | A | 7/2015 |
| EP | 1209876 | | 5/2002 |
| EP | 1770915 | | 4/2007 |
| EP | 1885096 | | 2/2008 |
| EP | 02296313 | | 3/2011 |
| EP | 2577910 | | 4/2013 |
| EP | 2622795 | | 8/2013 |
| EP | 2647174 | | 10/2013 |
| EP | 2760170 | | 7/2014 |
| EP | 2772026 | | 9/2014 |
| EP | 2901308 | A2 | 8/2015 |

(56) **References Cited**

FOREIGN PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| HK | 1182560 | | 11/2013 |
| HK | 1183569 | | 12/2013 |
| HK | 1183996 | | 1/2014 |
| HK | 1189438 | | 6/2014 |
| HK | 1198565 | A1 | 5/2015 |
| HK | 1198848 | A1 | 6/2015 |
| HK | 1199153 | A1 | 6/2015 |
| HK | 1199779 | A1 | 7/2015 |
| HK | 1200617 | A | 8/2015 |
| IN | Journal39/2015 | | 9/2015 |
| JP | H09-097233 | | 4/1997 |
| JP | 1999096128 | | 4/1999 |
| JP | H11-338836 | | 10/1999 |
| JP | 2000276432 | A | 10/2000 |
| JP | 2000307634 | A | 11/2000 |
| JP | 2001051859 | A | 2/2001 |
| JP | 2005141441 | A | 6/2005 |
| JP | 2006332825 | A | 12/2006 |
| JP | 2008040718 | A | 2/2008 |
| JP | 2013528330 | | 7/2013 |
| JP | 2014143686 | | 8/2014 |
| JP | 2015507380 | A | 3/2015 |
| KR | 10-0830413 | | 5/2008 |
| KR | 1020120117461 | | 8/2013 |
| WO | WO0113228 | | 2/2001 |
| WO | WO0114990 | | 3/2001 |
| WO | WO0145349 | | 6/2001 |
| WO | WO03103237 | | 12/2003 |
| WO | WO2004084085 | A1 | 9/2004 |
| WO | WO2008053954 | | 5/2008 |
| WO | WO2008078593 | A1 | 7/2008 |
| WO | WO2011049770 | | 4/2011 |
| WO | WO2011079381 | A1 | 7/2011 |
| WO | WO2011149796 | | 12/2011 |
| WO | WO2012050747 | | 4/2012 |
| WO | WO2012075237 | | 6/2012 |
| WO | WO2013070391 | | 5/2013 |
| WO | WO2013081952 | | 6/2013 |
| WO | WO2013096019 | | 6/2013 |
| WO | WO2013112492 | | 8/2013 |
| WO | WO2014052099 | | 4/2014 |
| WO | WO2014088741 | | 6/2014 |
| WO | WO2014093829 | | 6/2014 |
| WO | WO2014138483 | | 9/2014 |
| WO | WO2014144837 | | 9/2014 |
| WO | WO2014179753 | | 11/2014 |
| WO | WO2015153020 | A1 | 10/2015 |

OTHER PUBLICATIONS

Spatscheck et al., "Optimizing TCP Forwarder Performance", IEEE/ACM Transactions on Networking, vol. 8, No. 2, Apr. 2000.

Kjaer et al. "Resource allocation and disturbance rejection in web servers using SLAs and virtualized servers", IEEE Transactions on Network and Service Management, IEEE, US, vol. 6, No. 4, Dec. 1, 2009.

Sharifian et al. "An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads", The Journal of Supercomputing, Kluwer Academic Publishers, BO, vol. 53, No. 3, Jul. 3, 2009.

Hunt et al. NetDispatcher: A TCP Connection Router, IBM Research Report RC 20853 May 19, 1997.

Noguchi, "Realizing the Highest Level "Layer 7" Switch"= Totally Managing Network Resources, Applications, and Users =, Computer & Network LAN, Jan. 1, 2000, vol. 18, No. 1, p. 109-112.

Takahashi, "The Fundamentals of the Windows Network: Understanding the Mystery of the Windows Network from the Basics", Network Magazine, Jul. 1, 2006, vol. 11, No. 7, p. 32-35.

Ohnuma, "AppSwitch: 7th Layer Switch Provided with Full Setup and Report Tools", Interop Magazine, Jun. 1, 2000, vol. 10, No. 6, p. 148-150.

Koike et al., "Transport Middleware for Network-Based Control," IEICE Technical Report, Jun. 22, 2000, vol. 100, No. 53, pp. 13-18.

Yamamoto et al., "Performance Evaluation of Window Size in Proxy-based TCP for Multi-hop Wireless Networks," IPSJ SIG Technical Reports, May 15, 2008, vol. 2008, No. 44, pp. 109-114.

Abe et al., "Adaptive Split Connection Schemes in Advanced Relay Nodes," IEICE Technical Report, Feb. 22, 2010, vol. 109, No. 438, pp. 25-30.

* cited by examiner

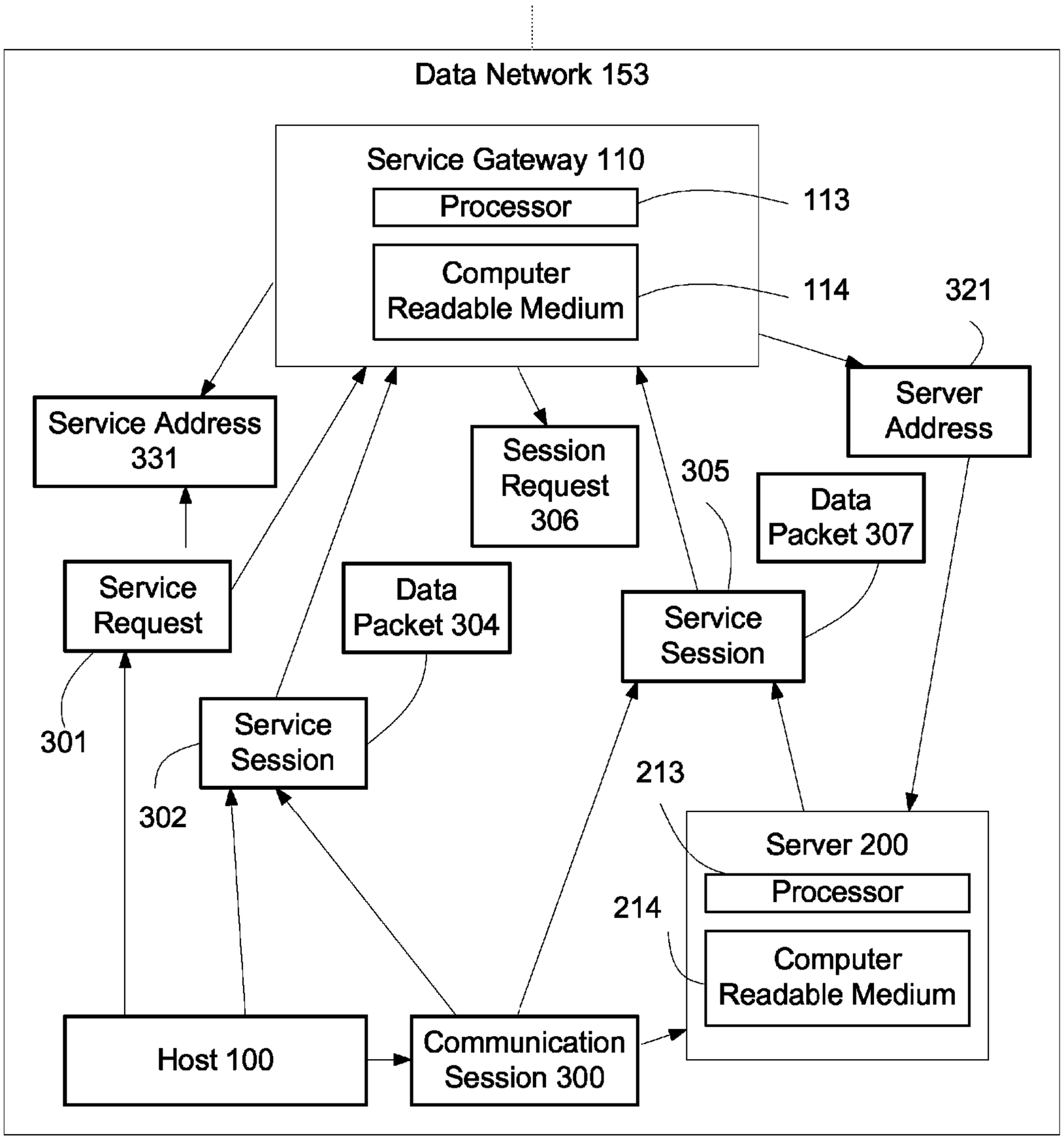



Figure 1 (Prior Art)

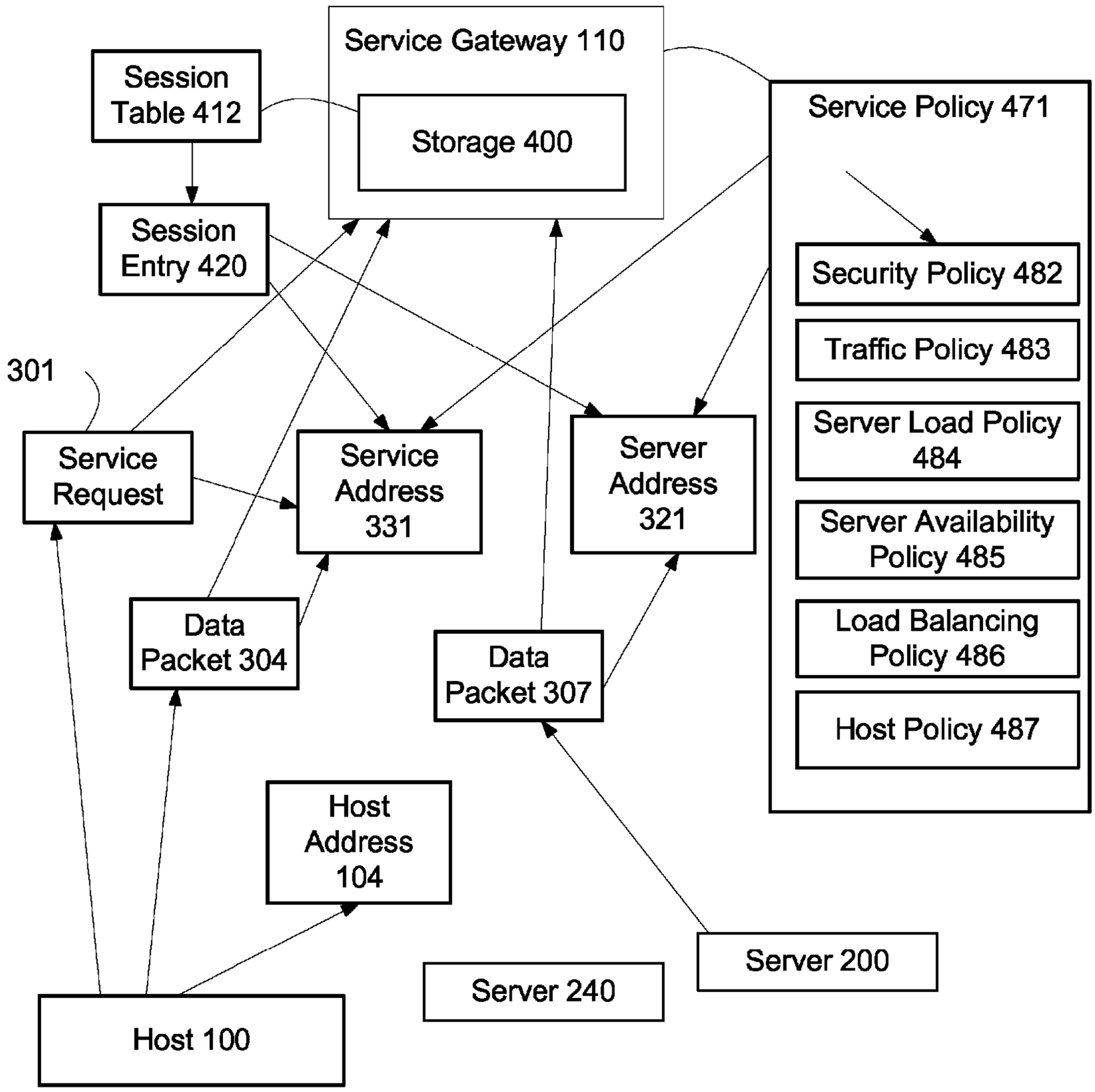



Figure 2 (Prior Art)

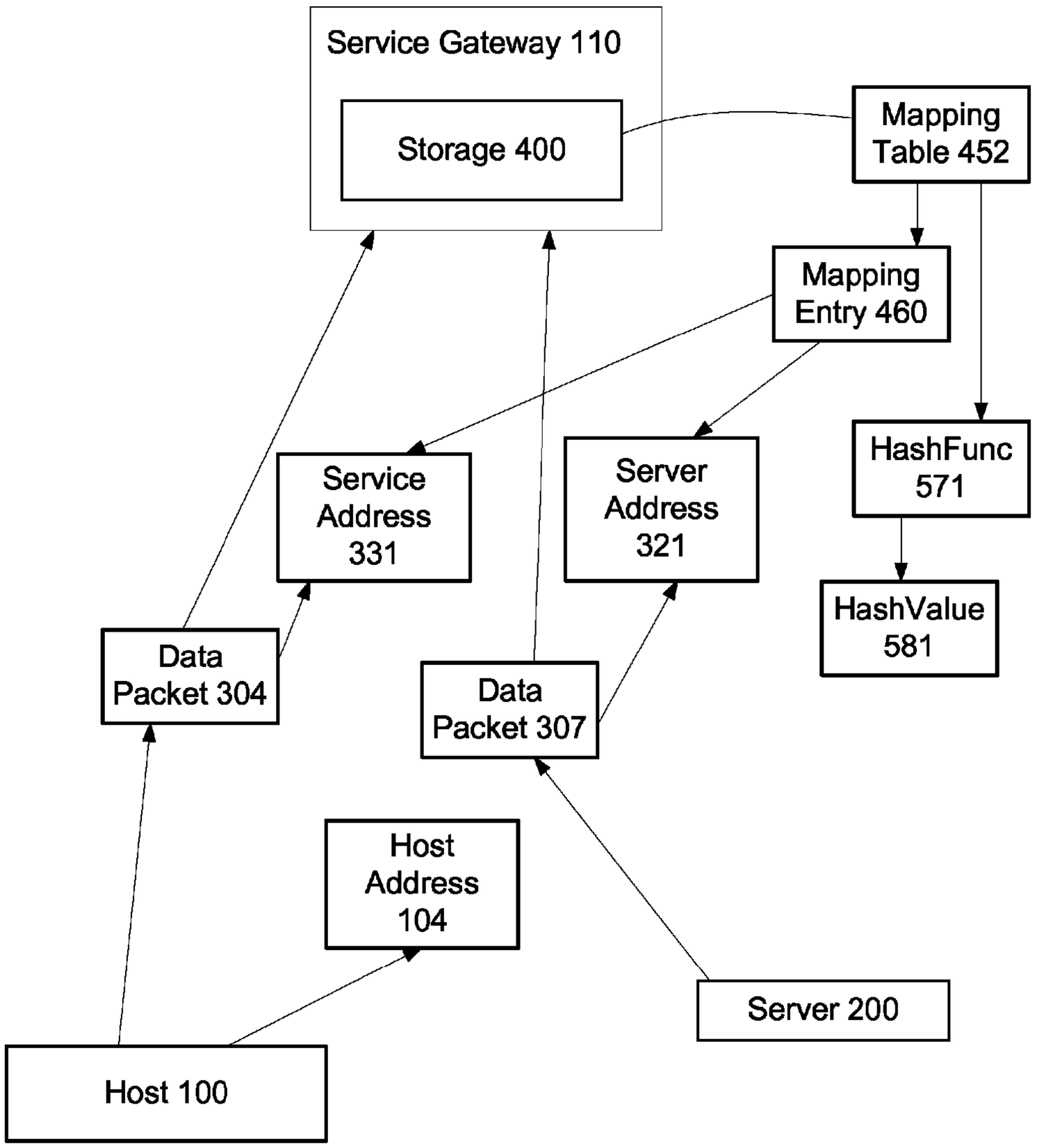



Figure 3 (Prior Art)

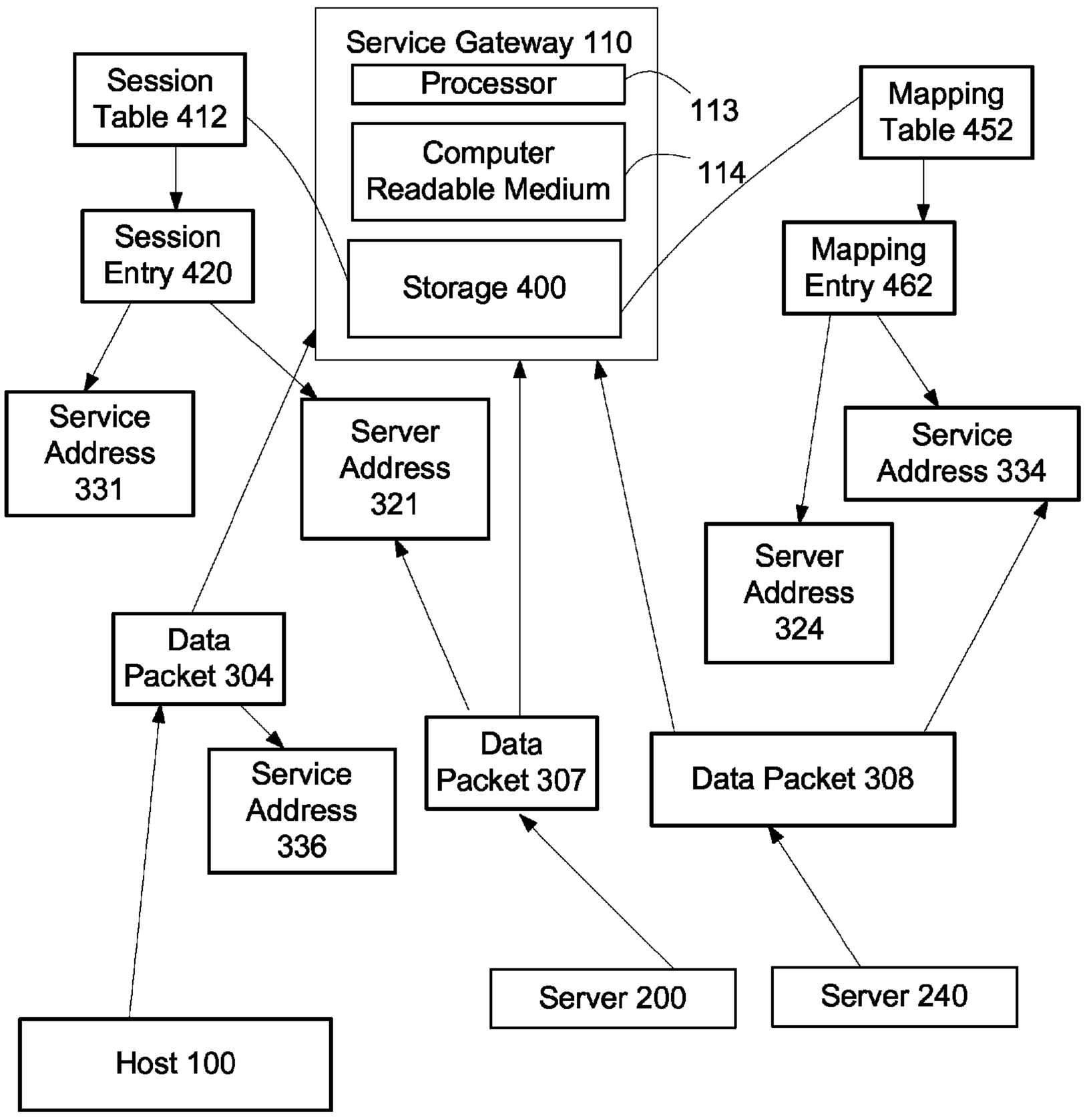

Service Gateway 110
Processor
Computer Readable Medium
Storage 400
113
114
Session Table 412
Session Entry 420
Service Address 331
Server Address 321
Mapping Table 452
Mapping Entry 462
Service Address 334
Server Address 324
Data Packet 304
Service Address 336
Data Packet 307
Data Packet 308
Host 100
Server 200
Server 240

Figure 4

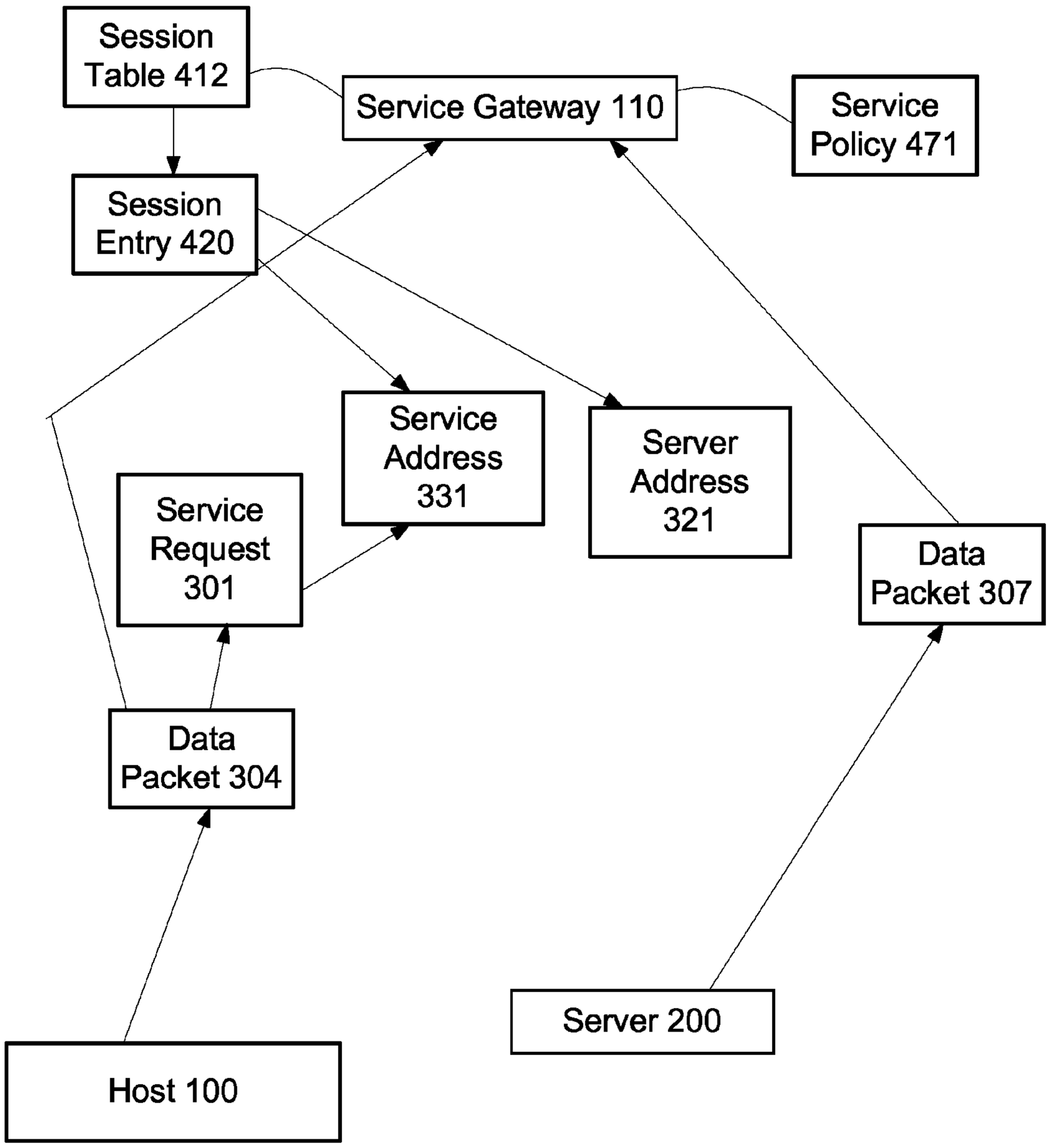

Session Table 412
Service Gateway 110
Service Policy 471
Session Entry 420
Service Address 331
Server Address 321
Service Request 301
Data Packet 307
Data Packet 304
Server 200
Host 100

Figure 5

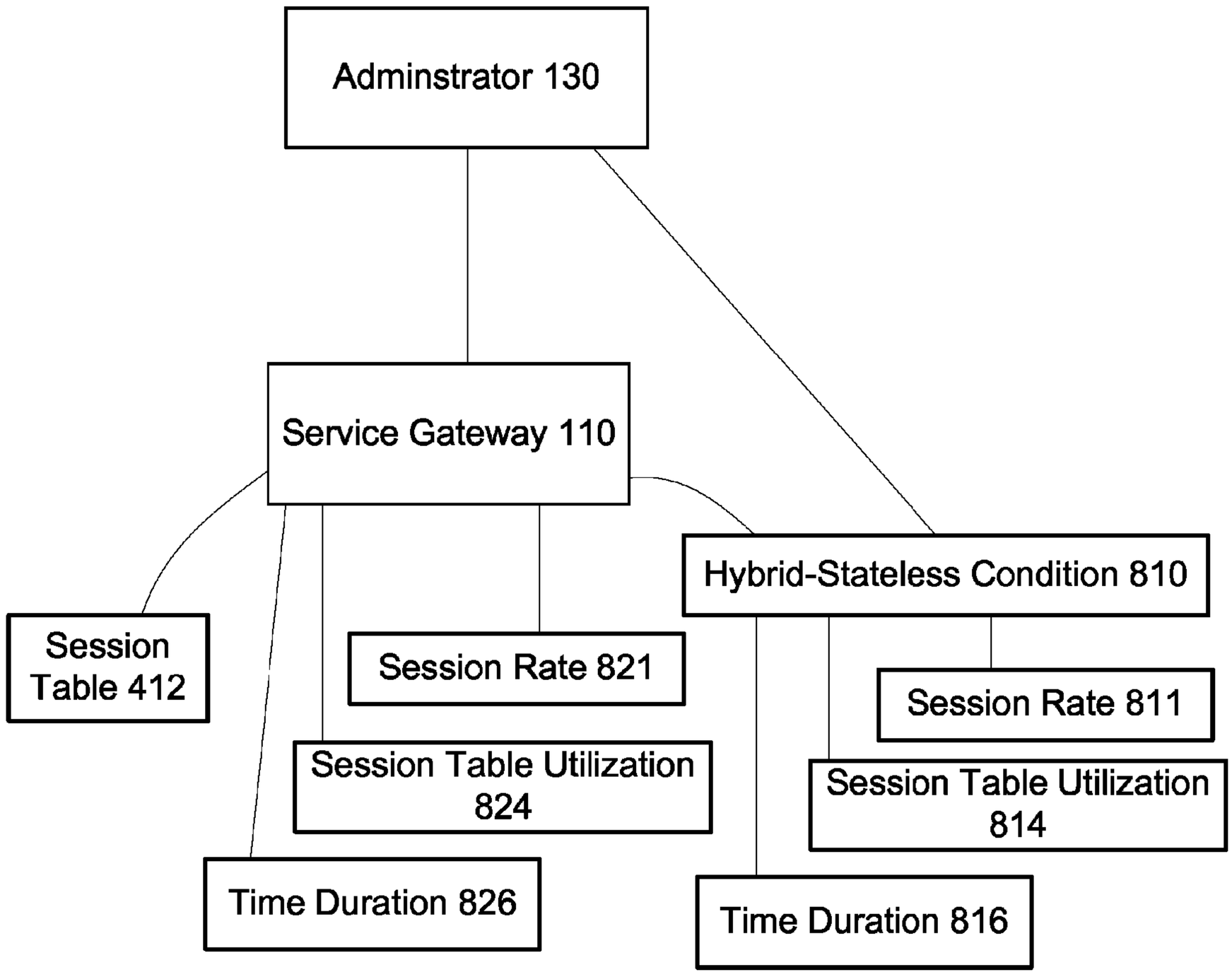

Adminstrator 130
Service Gateway 110
Session Table 412
Session Rate 821
Session Table Utilization 824
Time Duration 826
Hybrid-Stateless Condition 810
Session Rate 811
Session Table Utilization 814
Time Duration 816

Figure 6

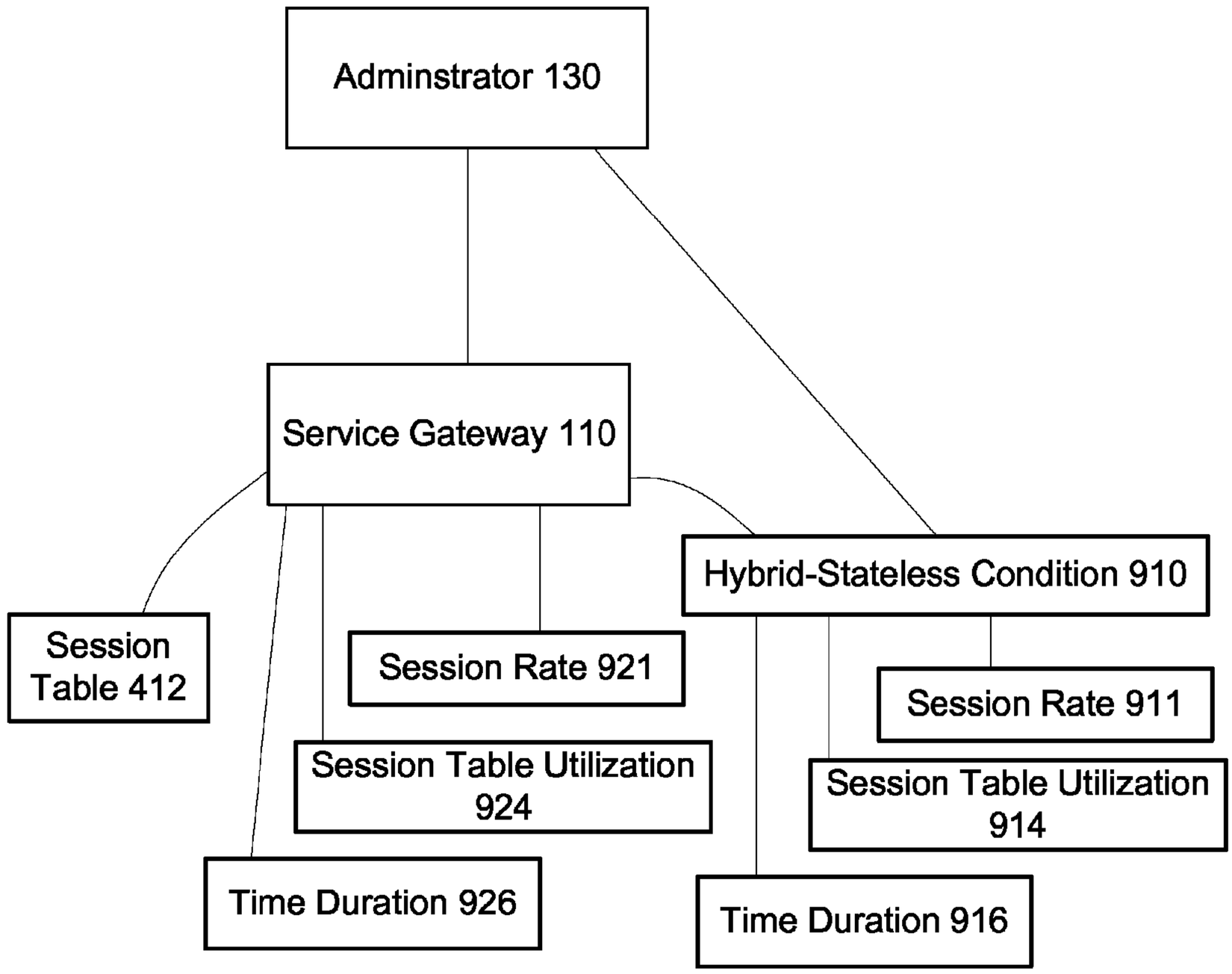

Adminstrator 130
Service Gateway 110
Hybrid-Stateless Condition 910
Session Table 412
Session Rate 921
Session Table Utilization 924
Time Duration 926
Session Rate 911
Session Table Utilization 914
Time Duration 916

Figure 7

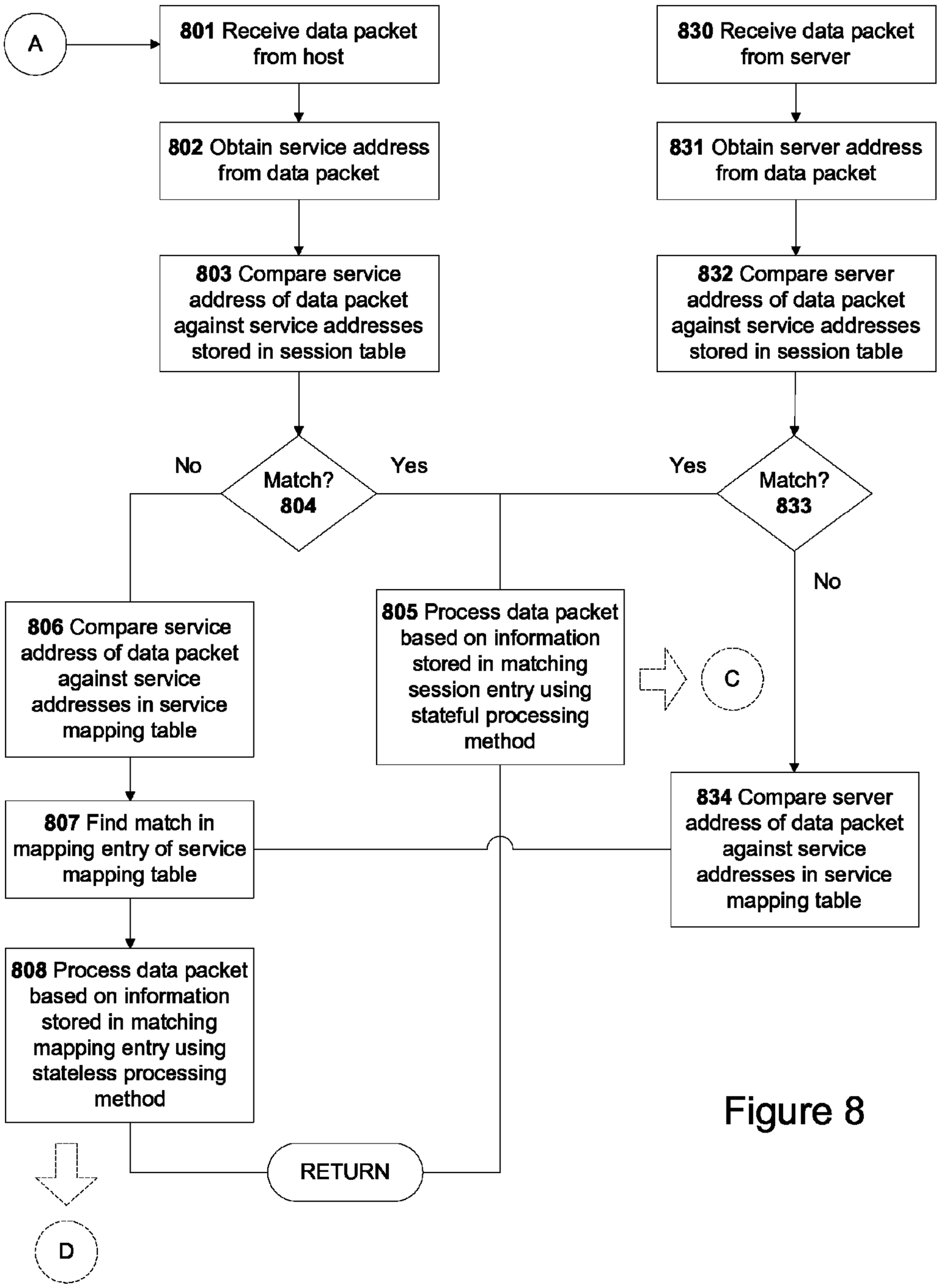

A
801 Receive data packet from host
802 Obtain service address from data packet
803 Compare service address of data packet against service addresses stored in session table
No
Match? 804
Yes
806 Compare service address of data packet against service addresses in service mapping table
807 Find match in mapping entry of service mapping table
808 Process data packet based on information stored in matching mapping entry using stateless processing method
D
805 Process data packet based on information stored in matching session entry using stateful processing method
C
RETURN
830 Receive data packet from server
831 Obtain server address from data packet
832 Compare server address of data packet against service addresses stored in session table
Yes
Match? 833
No
834 Compare server address of data packet against service addresses in service mapping table

Figure 8

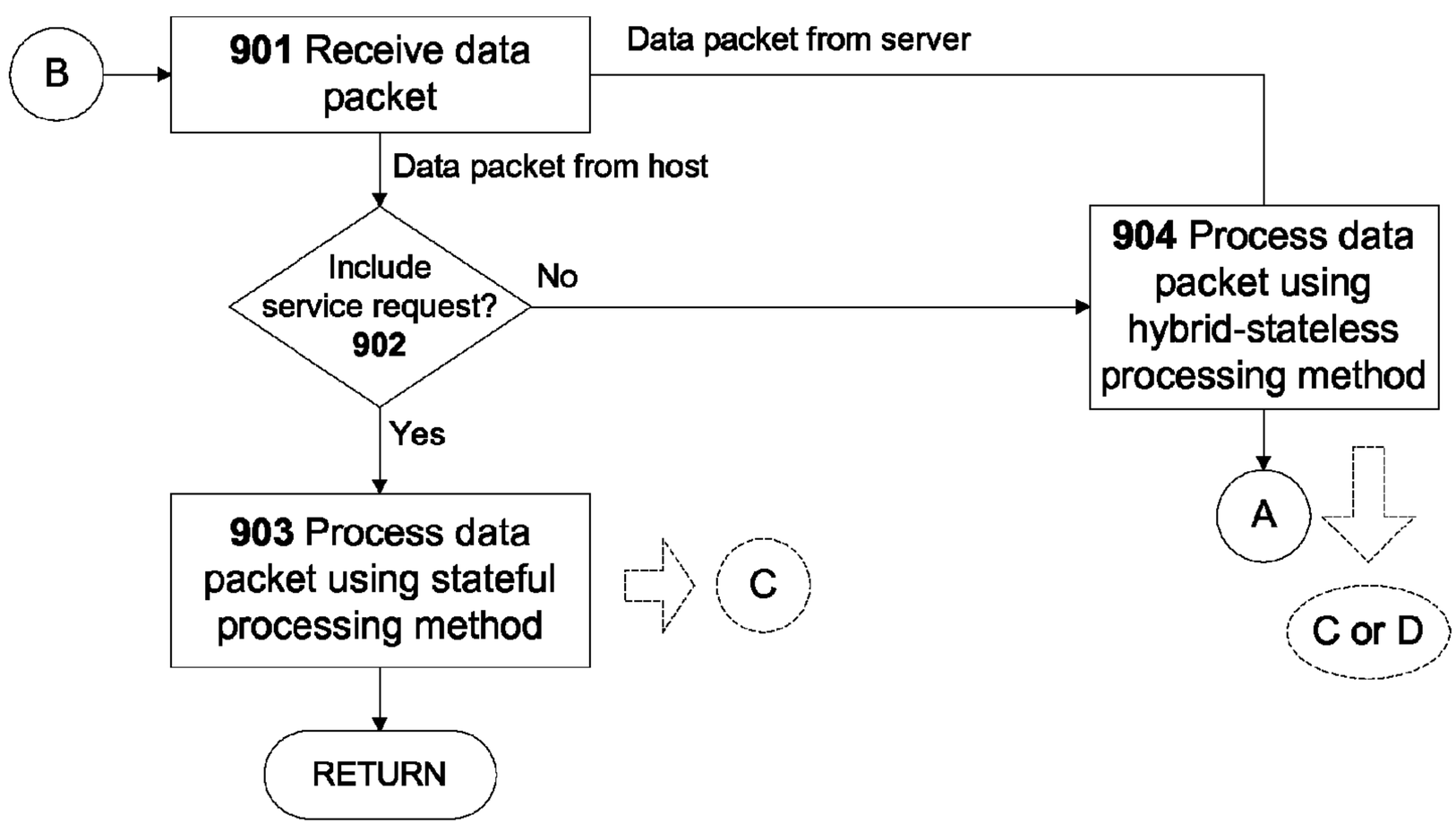

B
901 Receive data packet
Data packet from server
Data packet from host
Include service request? 902
No
Yes
904 Process data packet using hybrid-stateless processing method
903 Process data packet using stateful processing method
A
C
C or D
RETURN

Figure 9

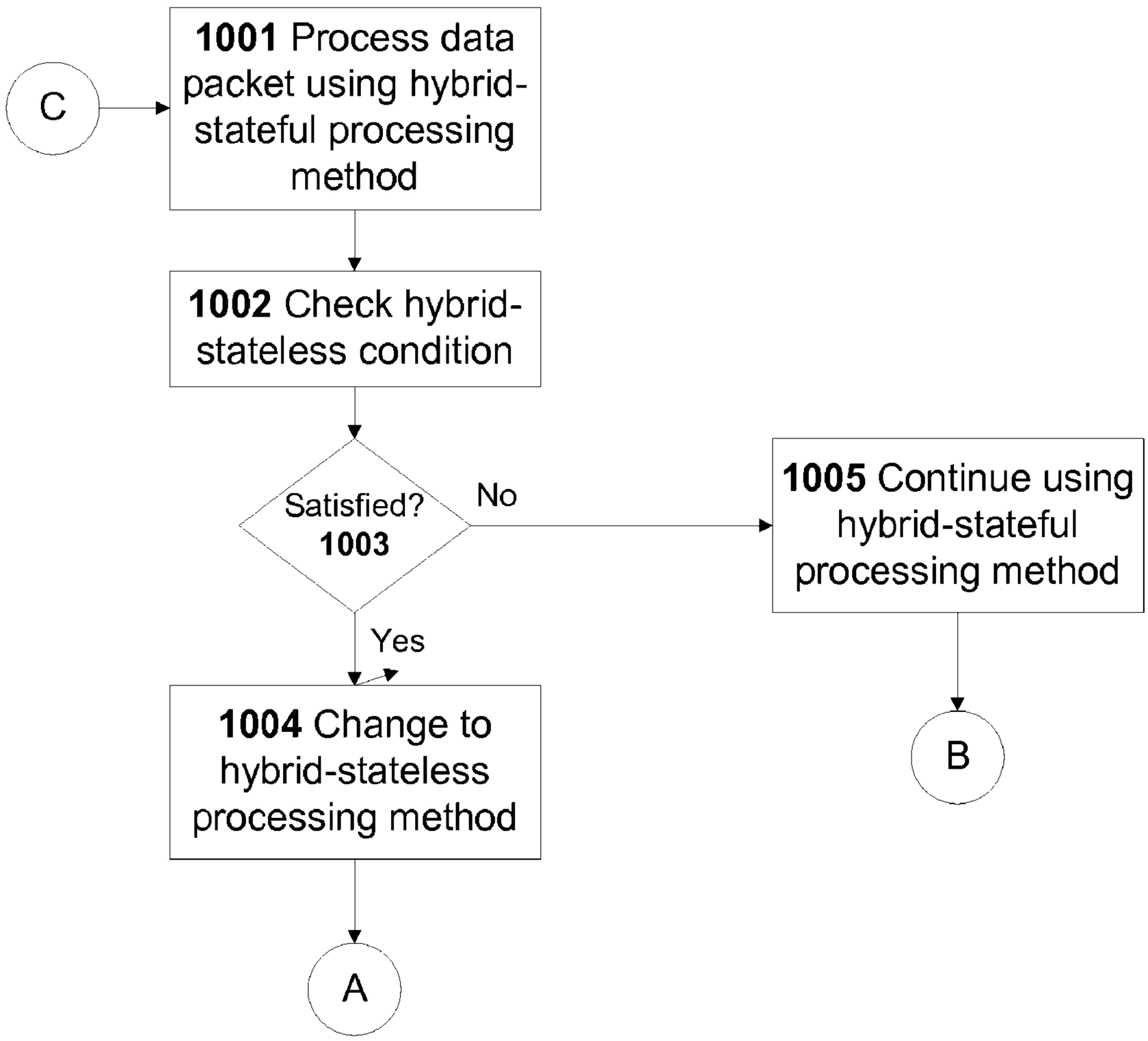

C
1001 Process data packet using hybrid-stateful processing method
1002 Check hybrid-stateless condition
Satisfied? 1003
No
1005 Continue using hybrid-stateful processing method
Yes
1004 Change to hybrid-stateless processing method
B
A

Figure 10

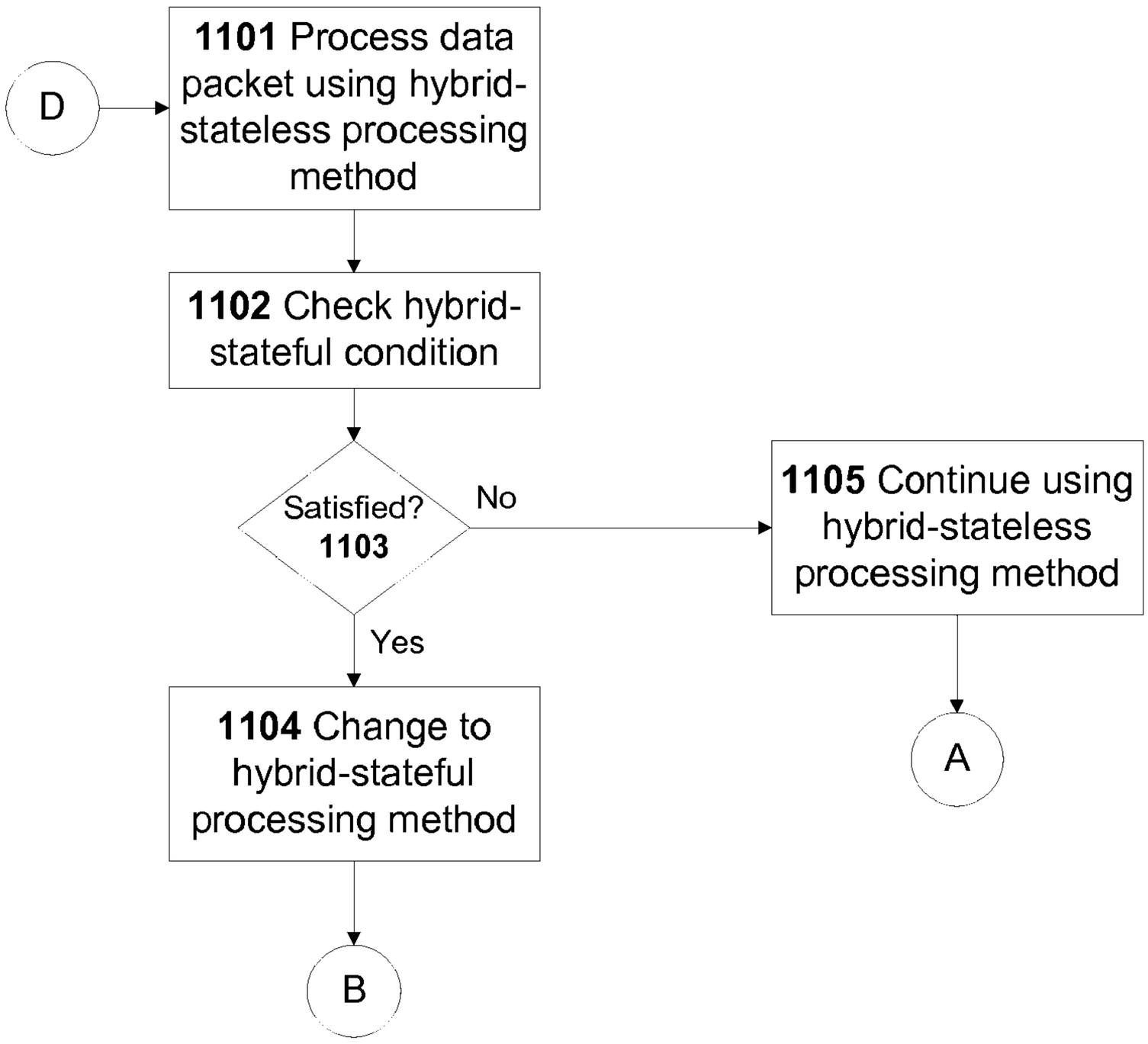

D
1101 Process data packet using hybrid-stateless processing method
1102 Check hybrid-stateful condition
Satisfied? 1103
No
1105 Continue using hybrid-stateless processing method
A
Yes
1104 Change to hybrid-stateful processing method
B

Figure 11

# COMBINING STATELESS AND STATEFUL SERVER LOAD BALANCING

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of and claims the priority benefit of U.S. patent application Ser. No. 13/280,336 filed on Oct. 24, 2011, and issued on Nov. 25, 2014 as U.S. Pat. No. 8,897,154, entitled "Combining Stateless and Stateful Server Load Balancing," the disclosure of which is incorporated herein by reference.

## BACKGROUND OF THE INVENTION

1. Field

This invention relates generally to data communications, and more specifically, to a service gateway.

2. Related Art

Demand for data communication services for consumer and corporate computing devices has been rapidly increasing. Service providers deploy service gateways such as server load balancers or traffic managers to bridge host computers or computing devices with servers providing the data services.

Service gateways provide services either using a stateful processing method or a stateless processing method. Generally, in a stateful processing method, packets are processed as a stream of packets, and each packet in the stream are processed in the same way. In a stateless processing method, packets are processed discretely, where each packet is assessed individually. The stateful processing method may be preferred over the stateless processing method due to the security and control features that may be implemented, however, the resource requirements of such features may make the services difficult to scale. The stateless processing method may be preferred over the stateful processing method due to its scalability, however, this is at the expense of security and control.

Traffic managed by service gateways is rarely uniform, as conditions on a network typically fluctuate, at times greatly. Currently, system administrators are required to choose either a stateful processing method or a stateless processing method for a particular service address, weighing the costs and benefits of each method. System administrators are not able to realize the advantages of both processing methods for such non-uniform traffic.

## BRIEF SUMMARY OF THE INVENTION

According to one embodiment of the present invention, a method for processing data packets sent over a communication session between a host and a server by a service gateway, comprises: processing a data packet using a hybrid-stateful processing method by the service gateway; checking by the service gateway whether a hybrid-stateless condition is satisfied; in response to determining that the hybrid-stateless condition is satisfied, changing to a hybrid-stateless processing method for a subsequently received data packet by the service gateway; and in response to determining that the hybrid-stateless condition is not satisfied, processing the subsequently received data packet using the hybrid-stateful processing method by the service gateway.

In another embodiment of the present invention, a method for processing data packets sent over a communication session between a host and a server by a service gateway, comprises: processing a data packet using a hybrid-stateless processing method by the service gateway, wherein the hybrid-stateless processing method processes the data packet using a stateless processing method unless a service address or a server address of the data packet matches a session entry in a session table; checking by the service gateway whether a hybrid-stateful condition is satisfied; in response to determining that the hybrid-stateful condition is satisfied, changing to a hybrid-stateful processing method for a subsequently received data packet by the service gateway, wherein the hybrid-stateful processing method processes the subsequently received data packet using a stateful processing method unless the subsequently received data packet either does not comprise a service request or the subsequently received data packet is received from the server; in response to determining that the hybrid-stateful condition is not satisfied, processing the subsequently received data packet using the hybrid-stateless processing method by the service gateway; wherein the hybrid-stateful processing method comprises: receiving the data packet by the service gateway; determining by the service gateway whether the data packet is received by the service gateway from the host or the server; in response to determining that the data packet is received from the host, determining by the service gateway whether the data packet comprises a service request; in response to determining that the data packet comprises the service request, processing the data packet using the stateful processing method by the service gateway; in response to determining that the data packet is received from the host and does not comprise the service request, processing the data packet using the hybrid-stateless processing method by the service gateway; and in response to determining that the data packet is received from the server, processing the data packet using the hybrid-stateless processing method by the service gateway.

In one aspect of the present invention, the hybrid-stateless processing method comprises: receiving the subsequently received data packet from the host by the service gateway; obtaining the service address from the subsequently received data packet by the service gateway; comparing the service address of the subsequently received data packet against service addresses stored in session entries in the session table by the service gateway; in response to determining that the session table comprises a session entry matching the service address of the subsequently received data packet, processing the subsequently received data packet based on information stored in the matching session entry using the stateful processing method by the service gateway. In response to determining that the session table does not comprise any session entry matching the service address of the subsequently received data packet: comparing the service address of the subsequently received data packet against service addresses stored in mapping entries in a mapping table by the service gateway, finding a mapping entry matching the service address of the subsequently received data packet by the service gateway, and processing the subsequently received data packet based on information stored in the matching mapping entry using the stateless processing method by the service gateway.

System and computer program products corresponding to the above-summarized methods are also described and claimed herein.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a service gateway for processing a communication session between a host and a plurality of servers.

FIG. **2** illustrates a stateful processing method.

FIG. **3** illustrates a stateless processing method.

FIG. **4** illustrates an embodiment of a service gateway performing a hybrid-stateless processing method combining a stateful processing method and a stateless processing method according to the present invention.

FIG. **5** illustrates an embodiment of a service gateway performing a hybrid-stateful processing method combining a stateful processing method and a stateless processing method according to the present invention.

FIG. **6** illustrates an embodiment of a service gateway changing from a hybrid-stateful processing method to a hybrid-stateless processing in response to a hybrid-stateless condition being satisfied according to the present invention.

FIG. **7** illustrates an embodiment of a service gateway changing from a hybrid-stateless processing method to a hybrid-stateful processing method in response to a hybrid-stateful condition being satisfied according to the present invention.

FIG. **8** is a flowchart illustrating an embodiment of a hybrid-stateless processing method according to the present invention.

FIG. **9** is a flowchart illustrating an embodiment of a hybrid-stateful processing method according to the present invention.

FIG. **10** is a flowchart illustrating an embodiment of a method for changing from a hybrid-stateful processing method to a hybrid-stateless processing in response to a hybrid-stateless condition being satisfied according to the present invention.

FIG. **11** is a flowchart illustrating an embodiment of a method for changing from a hybrid-stateless processing method to a hybrid-stateful processing method in response to a hybrid-stateful condition being satisfied according to the present invention.

## DETAILED DESCRIPTION OF THE INVENTION

The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention can take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment containing both hardware and software elements. In a preferred embodiment, the present invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

Furthermore, the present invention can take the form of a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. For the purposes of this description, a computer-usable or computer readable medium can be any apparatus that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

The medium can be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. Examples of a computer-readable medium include a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk. Current examples of optical disks include compact disk-read only memory (CD-ROM), compact disk-read/write (CD-R/W) and DVD.

A data processing system suitable for storing and/or executing program code will include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code in order to reduce the number of times code must be retrieved from bulk storage during execution.

Input/output or I/O devices (including but not limited to keyboards, displays, point devices, etc.) can be coupled to the system either directly or through intervening I/O controllers.

Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems, cable modem and Ethernet cards are just a few of the currently available types of network adapters.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified local function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the invention. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises" and/or "comprising," when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof. Embodiments of the present invention provide a security gateway with the capability of processing packets using either a hybrid stateless processing method or a hybrid stateful processing method, and with the capability for assessing conditions in determining whether to switch from using the hybrid stateful processing method to the hybrid stateless processing method or vice versa. Before describing the various embodiments of the present invention, the stateful only and stateless only methods are first described with reference to FIGS. **1** through **3**.

FIG. **1** illustrates a service gateway **110** for processing a communication session **300** between a host **100** and a server **200**. A plurality of data packets are sent between host **100** and server **200** over the communication session **300**. The service gateway **110** receives a service request **301** data packet from a host **100** to establish communication session **300**. Service

request **301** is delivered over a data network **153**. Service request **301** may be a Web service request such as a HTTP (Hypertext Transport Protocol) request, a secure HTTP request, a FTP (File Transfer Protocol) request, a file transfer request, a SIP (Session Initiation Protocol) session request, a request based on Web technology, a video or audio streaming request, a Web conferencing session request, or any request over the Internet, corporate network, data center network, or a network cloud. Service request **301** may be a request for a mobile application download, an advertisement delivery request, an e-book delivery request, a collaboration session request, or an on-line newspaper or magazine delivery request.

Host **100** is a computing device with network access capabilities. Host **100** may be a workstation, a desktop personal computer or a laptop personal computer. In some embodiments, host **100** is a Personal Data Assistant (PDA), a tablet, a smartphone, or a cellular phone. For other examples, host **100** may be a set-top box, an Internet media viewer, an Internet media player, a smart sensor, a smart medical device, a net-top box, a networked television set, a networked DVR, a networked Blu-ray player, or a media center.

Service gateway **110** is a computing device operationally coupled to a processor **113** and a computer readable medium **114** for storing computer readable program code to be executed by the processor **113**. Service gateway **110** may be implemented as a server load balancer, an application delivery controller, a service delivery platform, a traffic manager, a security gateway, a component of a firewall system, a component of a virtual private network (VPN), a load balancer for video servers, or a gateway to distribute load to one or more servers.

Server **200** is a computing device operationally coupled to a processor **213** and a computer readable medium **214** for storing computer readable program code to be executed by the processor **213**. The computer readable program code may implement server **200** as a Web server, a file server, a video server, a database server, an application server, a voice system, a conferencing server, a media gateway, a SIP server, a remote access server, a VPN server, a media center, an app server or a network server providing a network or application service to host **100**.

Data network **153** may include an Internet Protocol (IP) network. Data network **153** may include a corporate data network or a regional corporate data network, an Internet service provider network, a residential data network, a wired network such as Ethernet, a wireless network such as a WiFi network, or cellular network. Data network **153** may reside in a data center, or connects to a network or application network cloud.

Service request **301** from host **100** includes a service address **331**, such as an IP address. Service address **331** includes an application layer address or a transport layer port number, such as transmission control protocol (TCP) port number or user datagram protocol (UDP) port number. Service address **331** is associated with service gateway **110** so that service gateway **110** processes the service request **301**. Service address **331** may include a destination IP address of service request **301**, and optionally may include destination transport layer port number of service request **301**.

Service request **301** may include a TCP session request data packet, or a UDP data packet. Service address **331** is included in the data packet of service request **301**.

Service gateway **110** determines a server address **321** based on service address **331** obtained from service request **301**. Server address **321** is associated with server **200** and may include a network address or IP address of server **200**. Server address **321** may include an application layer address, such as a TCP port number or a UDP port number of server **200**.

Based on server address **321**, service gateway **110** sends a service session request **306** to server **200**. Subsequently service gateway **110** receives a response to session request **306** from server **200** and establishes a server-side service session **305** with server **200**. Based on session request **306** response, service gateway **110** sends a service request **301** response to host **100**, and establishes a host-side service session **302** with host **100** for service request **301**.

Communication session **300** includes host-side service session **302** and server-side service session **305**. Service session **302** includes one or more data packets from host **100** for communication session **300**. Service session **305** includes one or more data packets from server **200** for communication session **300**. Service session **302** may include service request **301**.

Upon establishment of service session **302** and service session **305**, service gateway **110** subsequently processes a data packet **304** of service session **302** received from host **100**. Data packet **304** includes service address **331**. Service gateway **110** modifies data packet **304** by replacing service address **331** with server address **321**. Service gateway **110** sends modified data packet **304** to server **200**.

When service gateway **110** receives a data packet **307** of service session **305** from server **200**, service gateway **110** processes data packet **307**. Data packet **307** of service session **305** may include server address **321**. Service gateway **110** modifies data packet **307** by replacing server address **321** with service address **331**. Service gateway **110** sends modified data packet **307** to host **100**.

There are two common methods in processing service session **302** and service session **305**: a stateful processing method and a stateless processing method. FIG. **2** illustrates a stateful processing method. In FIG. **2**, service gateway **110** maintains a service session table **412**. Session table **412** stores one or more service session entries. Service gateway **110** creates a session entry **420** for service session **302**. Session entry **420** stores service address **331** and server address **321** to associate service address **331** and server address **321**. Service gateway **110** may create session entry **420** after establishing host-side service session **302** and server-side service session **306**. Service gateway **110** may create session entry **420** after receiving service request **301**. Service gateway **110** stores service address **331** and server address **321** in session entry **420** after service gateway **110** determines the addresses. Service gateway **110** stores session entry **420** in session table **412**.

Service gateway **110** includes a storage **400** and stores session table **412** in storage **400**. Storage **400** is a memory module residing in service gateway **110**. Service gateway **110** includes a network processing module (not shown) comprising a field programmable gate array (FPGA), a network processor, an application specific integrated circuit (ASIC). Storage **400** is associated with the network processing module. Examples of storage **400** include a content addressable memory (CAM), a ternary content addressable memory (TCAM), a static random accessible memory (SRAM), or a dynamic random accessible memory (DRAM).

Service gateway **110** obtains service address **331** from service request **301**. Service gateway **110** maintains a service policy **471** and determines server address **321** based on service policy **471**. Service policy **471** may be based on a relationship between server **200** and service address **331**. Service policy **471** includes service address **331** and server address **321**. Service gateway **110** selects service policy **471** based on a match between service address **331** obtained from service

request **301** and the service address in the service policy **471**. Service gateway **110** applies service policy **471** to service request **301**. Service policy **471** may include a security policy **482** where a non-secure service request **301** can be sent to server **200**. Service policy **471** may include a traffic policy **483**, where service request **301** is served by server **200** when traffic load to server **200** is low. Service request **301** may be received from a predetermined network interface of service gateway **110** and traffic policy **483** indicates that service request **301** from the network interface should be sent to server **200**.

Server **240** also serves service request **301**. Service policy **471** may include a server load policy **484** indicating that service request **301** is to be sent to server **200** when server load of server **240** is high. In one example, service policy **471** includes a server availability policy **485** indicating that service request **301** is to be sent to server **200**, where server **200** is a back-up server to server **240**, and server **240** is not available. Service policy **471** may include a load balancing policy **486** between server **200** and server **240**. Service gateway **110** selects server **200** using the load balancing policy **486**, which may include a round robin or another load balancing scheme. Service policy **471** may include a host policy **487** indicating that service request **301** is to be sent to server **200** when host **100** satisfies host policy **487**.

After service gateway **110** applies service policy **471** to service request **301**, service gateway **110** retrieves server address **321** from service policy **471**. Service gateway **110** creates session entry **420** with service address **331** and server address **321**, associating service address **331** and server address **321**. Service gateway **110** stores session entry **420** in session table **412**.

Service gateway **110** uses session table **412** to process data packet **304** received from host **100**, and data packet **307** received from server **200**. When service gateway **110** receives data packet **304** from host **100**, service gateway **110** obtains service address **331** from data packet **304**. Service gateway **110** compares the obtained service address **331** against service addresses stored in session table **412**. When service gateway **110** determines there is a match between the obtained service address **331** and session entry **420** in session table **412**, service gateway **110** uses information stored in session entry **420** to process data packet **304**. Service gateway **110** modifies data packet **304** by replacing service address **331** with server address **321**, where server address **321** is obtained from the matched session entry **420**. Service gateway **110** sends modified data packet **304** to server **200**.

Service request **301** may include a host address **104** associated with host **100**. Service gateway **110** retrieves host address **104** from service request **301**. Service gateway **110** may use retrieved host address **104** when applying service policy **471**. Service gateway **110** stores host address **104** in service session entry **420**. Data packet **304** may include host address **104**. Service gateway **110** obtains host address **104** from data packet **304** and compares the obtained host address **104** against addresses stored in session table **412** and session entry **420**.

When service gateway **110** receives a data packet **307** of server-side service session **305** from server **200**, service gateway **110** retrieves server address **321** from data packet **307**. Service gateway **110** compares the obtained server address **321** against addresses stored in session table **412**, and determines there is a match with session entry **420**. In response to determining there is a match, service gateway **110** uses session entry **420** to process data packet **307**. Service gateway **110** modifies data packet **307** by replacing server address **321** with service address **331**, which is retrieved from the matched session entry **420**. Service gateway **110** sends modified data packet **307** to host **100**.

Data packet **307** may include host address **104**. Service gateway **110** obtains host address **104** from data packet **307** and uses the obtained host address **104** in comparing against addresses stored in session table **412** and session entry **420**.

Data packet **304** received from service session **302** may indicate a session termination request. For example, data packet **304** is a TCP FIN packet, a TCP RESET packet. Service gateway **110** inspects data packet **304** content and determines data packet **304** includes a session termination request. In response, service gateway **110** removes session entry **420** from session table **412**. Service gateway **110** may remove session entry **420** after processing data packet **304** or waits for a pre-determined period of time before removing session entry **420**.

The processing method illustrated in FIG. **2** is often referred as a stateful processing method. A stateful processing method allows service gateway **110** to apply one or more service policies to select server **200**. The service policies may include security policies and other policies to protect server **200**. Security policy **482** may cause service request **301** to be declined if a security concern is detected. Such security consideration is known to those skilled in the art and is not described in this application. Applying traffic policy **483** or server load policy **484** can also protect server **200** from overloading. Enforcing the service policies often improves service response time of server **200** to serve host **100**.

However, applying service policy **471** to service request **301** requires computation resource of service gateway **110**, such as CPU cycles. Such computation requirement may post a limitation on the ability of service gateway **110** to provide services when service gateway **110** receives and processes a large number of service requests over a short period of time.

For example, session table **412** has a certain capacity limit, such as 4 GB, 2000 entries, up to 10000 entries or 200 MB. The greater the number of service sessions serviced by service gateway **110** using a stateful processing method, the greater the number of session entries stored in session table **412**. The capacity of session table **412** may become a severe limitation to the servicing capabilities of service gateway **110**.

FIG. **3** illustrates a stateless processing method. In this method, service gateway **110** does not use session table **412**. Instead, service gateway **110** maintains and uses a service mapping table **452**. Service mapping table **452** is stored in storage **400**. Service mapping table **452** includes a service mapping entry **460**. Mapping entry **460** may include service address **331** and server address **321**, associating service address **331** and server address **321**. According to the service mapping entry **460**, server **200** with server address **321** serves host **100** for service address **331**.

When service gateway **110** receives a data packet **304** from host **100**, service gateway **110** obtains service address **331** from data packet **304**, and compares service address **331** with service addresses stored in service mapping table **452**. When service gateway **110** determines there is a match with mapping entry **460**, service gateway **110** retrieves server address **321** from mapping entry **460**. Service gateway **110** modifies data packet **304** by replacing service address **331** with server address **321**. Service gateway **110** sends modified data packet **304** to server **200**.

When service gateway **110** receives a data packet **307** from server **200**, service gateway **110** processes data packet **307** using service mapping table **452**. Service gateway **110** obtains server address **321** from data packet **307**. Service

gateway **110** compares server address **321** against server addresses stored in service mapping table **452**. When service gateway **110** determines there is a match with mapping entry **460**, service gateway **110** retrieves service address **331** from mapping entry **460**, and modifies data packet **307** by replacing server address **321** with service address **331**. Subsequently service gateway **110** sends modified data packet **307** to host **100**.

Service gateway **110** may match service address **331** or server address **321** against service mapping table **452** using a hash method. Service mapping table **452** includes a hash table using a hash function (HashFunc) **571**. Mapping entry **460** is associated with a hash value (HashValue **581**).

HashValue **581** includes the result of applying HashFunc **571** to service address **331**. HashValue **581** may include the result of applying HashFunc **571** to server address **321**.

HashValue **581** may include an index of mapping entry **460** in service mapping table **452**. Mapping entry **460** occupies an entry in service mapping table **452** indexed by HashValue **581**. For example, service mapping table **452** contains 1000 entries where the indices are 1-1000, and mapping entry **460** has an index of 894. In another example, service mapping table **452** contains 16 entries and mapping entry **460** has an index of 7.

Service gateway **110** applies HashFunc **571** to service address **331** of data packet **304** to obtain HashValue **581**. Assume that service gateway **110** searches service mapping table **452** for an entry with index HashValue **581** and finds mapping entry **460**. For data packet **307**, service gateway **110** applies HashFunc **571** to server address **321** of data packet **307** to obtain HashValue **581**. Service gateway **110** searches service mapping table **452** for an entry with index HashValue **581** and finds mapping entry **460**.

Mapping entry **460** may include HashValue **581**. After service gateway **110** applies hash function HashFunc **571** to obtain HashValue **581**, service gateway **110** searches service mapping table **452** and finds mapping entry **460** containing an index matching HashValue **581**.

Examples of hash functions HashFunc **571** include CRC checksum functions and other checksum functions; hash functions using a combination of bit-wise operators such as bit-wise AND operator, bit-wise OR operator, bit-wise NAND operator and bit-wise XOR operator; MD5 hash functions and other cryptography hash functions; Jenkins hash function and other non-cryptography hash functions; hardware based hash functions implemented in FPGA, ASIC or an integrated circuit board of service gateway **110**; and other types of hash functions or table lookup functions. Typically such hash functions are simple and can be calculated rapidly by service gateway **110**.

Data packet **304** includes host address **104** associated with host **100**. Service gateway **110** obtains host address **104** from data packet **304** and uses the obtained host address **104** in the processing of data packet **304**.

Data packet **307** includes host address **104**. Service gateway obtains host address **104** from data packet **307** and uses the obtained host address **104** in the processing of data packet **307**.

Typically, mapping entry **460** is configured by a service provider or an administrator of a service provider. Mapping entry **460** may be configured when server **200** becomes available, or when server address **321** or service address **331** becomes available. Server address **321** or service address **331** may be configured by the service provider to become available.

In this stateless processing method, service mapping table **452** is not related to the number of service sessions processed by service gateway **110**. The capacity of service mapping table **452** is related to the number of available service addresses and server addresses. Such capacity is usually small. Service mapping tables **452** may have a few tens of entries or a few thousand entries.

The advantages of a stateless processing method include small resource requirement for service mapping table **452**, a minimal or no computational requirement to handle service request **301**, or no requirements to apply service policy **471**. A stateless processing method is usually preferred over a stateful processing method when service gateway **110** receives a large number of service session requests in a short period of time, or under a heavy load of service requests. A stateless method is also preferred when the memory capacity of session table for new sessions is running low, say below 10% of the session table **412**. A stateless method protects service gateway **110** from resource overload and therefore maintains service quality towards host **100** under stressful situations.

However, a stateless processing method may be less desirable than a stateful processing method due to security concerns, since service gateway **110** does not apply security policy **482**. Similarly service gateway **110** does not apply any other policy in service policy **471**, affecting security of server **200**, security of data network **153**, traffic condition of data network **153**, and service quality rendered to host **100**. A stateful processing method is also preferred over the stateless processing method when service gateway **110** may select server address **321** from a plurality of server addresses. For example, a service provider may configure a plurality of servers to serve service address **331** in a load balancing manner. A service provider may configure a backup server for service address **331**.

In a typical deployment scenario, a service provider may use a stateful processing method for a first service address while using a stateless processing method for a different second service address. The service provider does not expect the first service to have significant traffic or usage. The service provider may not expect the second service to be a security concern. In reality, the first service may see a sudden surge of traffic due to an unforeseen situation, whereas the second service may suffer a security attack. Using a hybrid processing method according to the present invention, as described below, a service provider may combine a stateful processing method for the first service when the load is light and change to a stateless processing method when the load becomes heavy; and may deploy a hybrid processing method to combine a stateless processing method for the second service during normal circumstances and switch immediately to a stateful processing method when a security alert is detected for the second service.

The various embodiment of the present invention are now described with reference to FIGS. **4** through **11**.

FIG. **4** illustrates an embodiment of a service gateway **110** performing a hybrid-stateless processing method combining a stateful processing method and a stateless process method according to the present invention. FIG. **8** is a flowchart illustrating an embodiment of a hybrid-stateless processing method according to the present invention. In this embodiment, the computer readable medium **114** of the service gateway **110** stores computer readable program code, which when executed by processor **113**, implements the various embodiment of the present invention. Service gateway **110** maintains session table **412** and service mapping table **452** in storage **400**. In this embodiment of a hybrid-stateless processing method, service gateway **110** processes a received data packet **304** with a stateless method using service mapping table **452**

when the service address of the received data packet **304** does not match any service addresses stored in session table **412**.

Service gateway **110** connects to server **200** and server **240**. Server **200** is associated with server address **321**. Server **240** is associated with server address **324**. Service gateway **110** is associated with service address **331** and service address **334**.

In some embodiments, session table **412** includes a session entry **420** which stores service address **331** and server address **321**, associating service address **331** and server address **321**. Service mapping table **452** includes a mapping entry **462** which stores service address **334** and server address **324**, associating service addresses **334** and **324**.

In various embodiments, server **200** may be the same as server **240**. Server address **321** may be the same as server address **324**. Service address **331** may be the same as service address **334**.

Referring to both FIGS. **4** and **8**, service gateway **110** receives a data packet **304** from host **100** (**801**). Service gateway **110** obtains service address **336** from data packet **304** (**802**). Service gateway **110** compares service address **336** of data packet **304** against service addresses stored in session table **412** (**803**).

In some embodiments, service gateway **110** finds a match in session entry **420**, where service address **336** matches service address **331** of session entry **420** (**804**). In response to finding the match, service gateway **110** processes data packet **304** based on information stored in session entry **420** using a stateful processing method (**805**), such as the one described above with reference to FIG. **2**.

When service gateway **110** does not find a match in session table **412** (**804**), service gateway **110** compares service address **336** of data packet **304** against service addresses in service mapping table **452** (**806**). If service gateway **110** finds a match in mapping entry **462** of service mapping table **452**, wherein service address **336** matches service address **324** of mapping entry **462** (**807**), service gateway **110** processes data packet **304** based on information stored in mapping entry **462** using a stateless processing method (**808**), such as the one described above with reference to FIG. **3**.

In various embodiments, service gateway **110** receives a data packet **307** from server **200** (**830**). Service gateway **110** extracts server address **321** from data packet **307** (**831**) and compares server address **321** of data packet **307** against server addresses stored in session table **412** (**832**). When service gateway **110** finds a match in session entry **420**, with server address **321** of data packet **307** matching server address **321** of session entry **420** (**803**), service gateway **110** processes data packet **308** using the stateful processing method (**805**), as described above with reference to FIG. **2**.

In some embodiments, service gateway **110** receives a data packet **308** from server **240** (**830**). Service gateway **110** extracts server address **324** from data packet **308** (**832**) and compares server address **324** of data packet **308** against server addresses stored in session table **412** (**832**). When service gateway **110** does not find a match (**833**), service gateway **110** compares server address **324** of data packet **308** against server addresses stored in service mapping table **452** (**834**) and finds a match in mapping entry **462**, where server address **324** of data packet **308** matches server address **324** of mapping entry **462** (**807**). In response, service gateway **110** modifies data packet **308** based on information stored in mapping entry **462** using a stateless processing method (**808**). Service gateway **110** sends modified data packet **308**.

FIG. **5** illustrates an embodiment of a service gateway **110** performing a hybrid-stateful processing method combining a stateful processing method and a stateless processing method according to the present invention. FIG. **9** is a flowchart illustrating an embodiment of the hybrid-stateful processing method according to the present invention. Referring to both FIGS. **5** and **9**, service gateway **110** receives a data packet **304** from host **100** (**901**). In some embodiments, service gateway **110** determines that data packet **304** includes a service request **301** from host **100** (**902**). In response, service gateway **110** applies a stateful processing method to service request **301** (**903**). Service gateway **110** performs the stateful processing method, including applying service policy **471** to service request **301**, creating session entry **420** using service address **331** of service request **301** and server address **321** of service policy **471**, as described above with reference to FIG. **2**.

In various embodiments, service gateway **110** determines data packet **304** does not include a service request (**902**). In response, service gateway **110** processes data packet **304** using the hybrid-stateless processing method, as described above with reference to FIG. **4**.

In other embodiments, service gateway **110** receives a data packet **307** from server **200** (**901**). In this embodiment of a hybrid-stateful processing method, service gateway **110** applies a hybrid-stateless processing method to data packet **307** (**904**), as described above with reference to FIG. **4**.

FIGS. **6** and **10** illustrate an embodiment of a service gateway and a method, respectfully, for changing from a hybrid-stateful processing method to a hybrid-stateless processing method in response to a hybrid-stateless condition being satisfied according to the present invention. Referring to both FIGS. **6** and **10**, service gateway **110** is using a hybrid-stateful processing method (**1001**). Service gateway **110** maintains a hybrid-stateless condition **810**. Service gateway **110** checks if hybrid-stateless condition **810** is satisfied (**1002**). In response to determining that the hybrid-stateless condition **810** is satisfied (**1003**), service gateway **110** changes to a hybrid-stateless processing method (**1004**). The service gateway **110** processes the next data packet received using the hybrid-stateless processing method, as described above with reference to FIGS. **4** and **8**. In response to determining that the hybrid-stateless condition **810** is not satisfied (**1003**), the service gateway **110** continues using the hybrid-stateful processing method (**1005**), as described above with reference to FIGS. **5** and **9**.

In some embodiments, hybrid-stateless condition **810** includes a session rate **811**. For example, session rate **811** is 10 thousand sessions per second, 5 thousand active sessions per second, or one hundred sessions per 10 milliseconds.

In various embodiments, service gateway **110** calculates a session rate **821**. Session rate **821** can be calculated based on a count of active host-side service sessions over a period of time. When the service session is associated with a session entry in session table **412**, a service session is active. In various embodiments, session rate **821** calculates a difference between a count of received service requests and a count of received service termination requests over a period of time. In other embodiments, session rate **821** calculates a count of service requests received over a period of time.

In some embodiments, service gateway **110** calculates a session rate **821** in a predetermined period of time, such as every second, once every 250 milliseconds, once every 3 seconds or once every 10 seconds. In other embodiments, service gateway **110** calculates session rate **821** at variable times. For example, service gateway **110** calculates session rate **821** when a data packet from a host is received; when a service request is received; when a service termination request is received; or when a data packet is received from server **200**. Service gateway **110** compares session rate **821** with session rate **811** of hybrid-stateless condition **810**. If

session rate **821** exceeds or is equal to session rate **811**, service gateway **110** determines that hybrid-stateless condition **810** is met and satisfied.

In various embodiments, hybrid-stateless condition **810** includes a session table utilization **814**. A session table utilization is a parameter setting forth a percentage of the session table capacity that is storing session entries. Hybrid-stateless condition **810** is satisfied if a count of stored session entries of session table **412** exceeds session table utilization **814**. For example, session table utilization **814** is 90%, 85% or 95%. Service gateway **110** calculates a session table utilization **824** from time to time by calculating a count of stored session entries of session table **412**. In some embodiments, service gateway **110** calculates session table utilization **824** periodically, such as every second, once every 20 milliseconds, once every 500 milliseconds, or once every 2 seconds. In other embodiments, service gateway **110** calculates session table utilization **824** when service gateway **110** processes a service request, a service termination request, or a data packet.

Service gateway **110** compares session table utilization **824** with session table utilization **814** of hybrid-stateless condition **810**. When session table utilization **824** exceeds or is equal to session table utilization **814**, service gateway **110** determines that hybrid-stateless condition **810** is met and satisfied.

In some embodiments, hybrid-stateless condition **810** further includes a time duration **816**, where hybrid-stateless condition **810** must be considered met for at least a time duration **816** in order for the hybrid-stateless condition **810** to be satisfied. Examples of time duration **816** include 120 seconds, 30 seconds, and 5 seconds. Service gateway **110** checks from time to time whether the hybrid-stateless condition **810** is met, as described earlier. In various embodiments, service gateway **110** further includes a time duration **826** stored in memory. Initially, service gateway **110** assigns a value of 0 to the time duration **826**. From time to time, service gateway **110** checks if hybrid-stateless condition **810** is met. If hybrid-stateless condition **810** is met, service gateway **110** increases the time duration **826** by an amount of time elapsed since the last time the hybrid-stateless condition **810** was checked. After the time duration **826** is modified, service gateway **110** checks if the time duration **826** exceeds time duration **816**. If time duration **826** exceeds time duration **816**, service gateway **110** determines that hybrid-stateless condition **810** is satisfied. Service gateway **110** subsequently changes to employ a hybrid-stateless method with subsequently received data packets.

If service gateway **110** determines hybrid-stateless condition **810** is not met, service gateway **110** modifies the time duration **826** to a value of 0.

In some embodiments, service gateway **110** receives hybrid-stateless condition **810** from an operator or an administrator **130**. Administrator **130** can be a human operator provisioning hybrid-stateless condition **810** onto service gateway **110**. Administrator **130** can be a network management system sending hybrid-stateless condition **810** to service gateway **110**. Administrator **130** may include a storage medium storing hybrid-stateless condition **810**. Service gateway **110** retrieves hybrid-stateless condition **810** from the storage of administrator **130**.

FIGS. **7** and **11** illustrate an embodiment of a service gateway and a method, respectfully, for changing from a hybrid-stateless processing method to a hybrid-stateful processing method in response to a hybrid-stateful condition being satisfied according to the present invention. Referring to both FIGS. **7** and **11**, service gateway **110** employs a hybrid-stateless processing method (**1101**) Service gateway **110** maintains a hybrid-stateful condition **910**. Service gateway **110** checks if hybrid-stateful condition **910** is satisfied (**1102**). In response to determining that the hybrid-stateful condition **910** is satisfied (**1103**), service gateway **110** changes to a hybrid-stateful processing method (**1104**) and processes the next data packet using the hybrid-stateful processing method, described above with reference to FIGS. **5** and **9**. In response to determining that the hybrid-stateful condition **910** is not satisfied (**1103**), service gateway **110** continues using the hybrid-stateless processing method (**1105**) and processes the next data packet using the hybrid-stateless processing method, as described above with reference to FIGS. **4** and **8**.

In some embodiments, hybrid-stateful condition **910** includes a session rate **911**. For example, session rate **911** is 1 thousand sessions per second, 500 active sessions per second, or ten sessions per 10 milliseconds.

Service gateway **110** can calculate a session rate **921**. In some embodiments, session rate **921** calculates a difference between a count of received service requests and a count of received service termination requests over a period of time. Session rate **921** may also calculate a count of service requests received over a period of time. In various embodiments, service gateway **110** determines if a data packet received from a host includes a service request before applying a hybrid-stateless processing method to the received data packet. Service gateway **110** may also determine if a data packet received from a host or a server includes a service termination request before applying a hybrid-stateless processing method to the received data packet.

In some embodiments, service gateway **110** calculates session rate **921** in a predetermined period of time, such as every second, once every 100 milliseconds, once every 3 seconds, or once every 5 seconds. Service gateway **110** may also calculate session rate **921** at variable times. For example, service gateway **110** calculates session rate **921** when a data packet from a host is received; when a service request is received; when a service termination request is received; or when a data packet is received from a server. Service gateway **110** compares session rate **921** with session rate **911**. If session rate **921** is below or smaller than session rate **911**, service gateway **110** determines that hybrid-stateful condition **910** is met and satisfied.

In various embodiments, hybrid-stateful condition **910** includes a session table utilization **914**. Hybrid-stateful condition **910** is satisfied if a count of stored session entries of session table **412** does not exceed session table utilization **914**. For example, session table utilization **914** is 60%, 75% or 45%. Service gateway **110** calculates session table utilization **924** from time to time by calculating a count of stored session entries of session table **412**. In some embodiments, service gateway **110** calculates session table utilization **924** periodically, such as every second, once every 20 milliseconds, once every 500 milliseconds, or once every 2 seconds. Service gateway **110** may also calculate session table utilization **924** when service gateway **110** processes a service request, a service termination request, or a data packet.

Service gateway **110** compares session table utilization **924** with session table utilization **914** of hybrid-stateful condition **910**. If session table utilization **924** is smaller than session table utilization **914**, service gateway **110** determines that hybrid-stateful condition **910** is met and satisfied.

Hybrid-stateful condition **910** may further include a time duration **916**, where hybrid-stateful condition **910** must be considered met for at least a time duration **916** in order for the hybrid-stateful condition **910** is satisfied. Examples of time duration **916** include 100 seconds, 40 seconds, and 5 seconds.

Service gateway **110** checks from time to time if the hybrid-stateful condition **910** is met as described earlier. In some embodiments, service gateway **110** further includes a time duration **926** stored in memory. Initially, service gateway **110** assigns a value of 0 to the time duration **926**. From time to time, service gateway **110** determines if hybrid-stateful condition **910** is met. If hybrid-stateful condition **910** is met, service gateway **110** increases the time duration **926** by an amount of time elapsed since the last time the hybrid-stateful condition **910** was checked. In various embodiments, after the time duration **926** is modified, service gateway **110** checks if the time duration **926** exceeds time duration **916**. If time duration **926** exceeds time duration **916**, service gateway **110** determines hybrid-stateful condition **910** is satisfied. Service gateway **110** subsequently changes to employ a hybrid-stateful method with subsequently received data packets.

In some embodiments, service gateway **110** receives hybrid-stateful condition **910** from an operator or an administrator **130**. Administrator **130** can be a human operator provisioning hybrid-stateful condition **910** onto service gateway **110**. Administrator **130** can be a network management system sending hybrid-stateful condition **910** to service gateway **110**. Administrator **130** can include a storage medium storing hybrid-stateful condition **910**. Service gateway **110** retrieves hybrid-stateful condition **910** from the storage of administrator **130**.

Returning to FIG. **8**, FIG. **8** shows that when the service gateway **110** is processing data packets using the stateful processing method (**805**), the service gateway **110** would check whether the hybrid-stateless condition **801** is met (see FIG. **10**). FIG. **8** also shows that when the service gateway **110** is processing data packets using the stateless processing method (**808**), the service gateway **110** would check whether the hybrid-stateful condition **910** is met (see FIG. **11**). However, the references (C and D) to FIGS. **10** and **11** are not intended to convey any order of steps. The checking of the conditions **810** or **910** may occur concurrently with the processing of data packets, as described above with reference to FIGS. **4** and **8**.

Returning to FIG. **9**, FIG. **9** shows that when the service gateway **110** is processing data packets using the stateful processing method (**903**), the service gateway **110** would check whether the hybrid-stateless condition **810** is met (see FIG. **10**). FIG. **9** also shows that when the service gateway **110** is processing data packets using the hybrid-stateless processing method (**904**), the service gateway **110** would either check if the hybrid-stateless condition **810** or the hybrid-stateful condition **910** is met (see FIGS. **10** and **11**), depending on the processing during the hybrid states processing method per FIGS. **4** and **8**. However, the reference to FIGS. **10** (C) and **11** (D) are not intended to convey any order of steps. The checking of the conditions **810** or **910** may occur concurrently with the processing of data packets as illustrated in FIGS. **5** and **9**.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

**1**. A method for processing data packets sent over a communication session between a host and a server by a service gateway, comprising:

processing a data packet using a hybrid-stateful processing method by the service gateway, the hybrid-stateful processing method utilizing a service session table;

checking by the service gateway whether a hybrid-stateless condition is satisfied;

in response to determining that the hybrid-stateless condition is satisfied, changing to a hybrid-stateless processing method for a subsequently received data packet by the service gateway, the hybrid-stateless processing method utilizing a service mapping table; and

in response to determining that the hybrid-stateless condition is not satisfied, processing the subsequently received data packet using the hybrid-stateful processing method by the service gateway.

**2**. The method of claim **1**, wherein the checking by the service gateway whether the hybrid-stateless condition is satisfied comprises:

comparing a time duration stored in memory against a predetermined time duration by the service gateway;

determining whether the time duration stored in memory exceeds the predetermined time duration;

in response to determining that the time duration stored in memory exceeds the predetermined time duration, determining by the service gateway that the hybrid-stateless condition is satisfied; and

in response to determining that the time duration stored in memory does not exceed the predetermined time duration, determining by the service gateway that the hybrid-stateless condition is not satisfied.

**3**. The method of claim **1**, wherein the checking by the service gateway whether a hybrid-stateless condition is satisfied comprises receiving from an administrator the hybrid-stateless condition by the service gateway.

**4**. The method of claim **3**, wherein the administrator comprises:

a human operator;

a network management system; or

a storage medium storing the hybrid-stateless condition.

**5**. A system, comprising:

a service gateway comprising a processor and a computer readable storage medium having computer readable program code embodied therewith, the computer readable program code configured to:

process a data packet using a hybrid-stateful processing method, the hybrid-stateful processing method utilizing a service session table at the service gateway;

check whether a hybrid-stateless condition is satisfied;

in response to determining that the hybrid-stateless condition is satisfied, change to a hybrid-stateless processing method for a subsequently received data packet, the hybrid-stateless processing method utilizing a service mapping table at the service gateway; and

in response to determining that the hybrid-stateless condition is not satisfied, process the subsequently received data packet using the hybrid-stateful processing method.

**6**. The system of claim **5**, wherein the check whether a hybrid-stateless condition is satisfied comprises:

compare a time duration stored in memory against a predetermined time duration by the service gateway;

determine whether the time duration stored in memory exceeds the predetermined time duration;

in response to determining that the time duration stored in memory exceeds the predetermined time duration, determine by the service gateway that the hybrid-stateless condition is satisfied; and

in response to determining that the time duration stored in memory does not exceed the predetermined time duration, determine by the service gateway that the hybrid-stateless condition is not satisfied.

**7**. The system of claim **5**, wherein the check whether a hybrid-stateless condition is satisfied comprises receiving from an administrator the hybrid-stateless condition by the service gateway.

**8**. The system of claim **7**, wherein the administrator comprises:

a human operator;

a network management system; or

a storage medium storing the hybrid-stateless condition.

**9**. A method for processing data packets sent over a communication session between a host and a server by a service gateway, comprising:

processing a data packet using a hybrid-stateless processing method by the service gateway, the hybrid-stateless processing method utilizing a service mapping table;

checking by the service gateway whether a hybrid-stateful condition is satisfied;

in response to determining that the hybrid-stateful condition is satisfied, changing to a hybrid-stateful processing method for a subsequently received data packet by the service gateway, the hybrid-stateful processing method utilizing a service session table; and

in response to determining that the hybrid-stateful condition is not satisfied, processing the subsequently received data packet using the hybrid-stateless processing method by the service gateway.

**10**. The method of claim **9**, wherein the hybrid-stateful condition comprises a predetermined session rate, wherein the checking by the service gateway whether the hybrid-stateful condition is satisfied comprises:

calculating a session rate for a plurality of communication sessions received by the service gateway;

determining whether the calculated session rate is less than the predetermined session rate by the service gateway;

in response to determining that the calculated session rate is less than the predetermined session rate, determining by the service gateway that the hybrid-stateful condition is satisfied; and

in response to determining that the calculated session rate is greater than or equals the predetermined session rate, determining by the service gateway that the hybrid-stateful condition is not satisfied.

**11**. The method of claim **10**, wherein the calculated session rate comprises:

a difference between a count of received service requests and a count of received service termination requests over a predetermined period of time or

a count of service requests over the predetermined period of time.

**12**. The method of claim **9**, wherein the hybrid-stateful condition comprises a predetermined session table utilization, wherein the checking by the service gateway whether a hybrid-stateful condition is satisfied comprises:

counting a number of stored session entries in the session table by the service gateway;

determining whether the number of stored session entries does not exceed the predetermined session table utilization by the service gateway;

in response to determining that the number of stored session entries does not exceed the predetermined session table utilization, determining by the service gateway that the hybrid-stateful condition is satisfied; and

in response to determining that the number of stored session entries exceeds the predetermined session table utilization, determining by the service gateway that the hybrid-stateful condition is not satisfied.

**13**. The method of claim **9**, wherein the checking by the service gateway whether the hybrid-stateful condition is satisfied comprises:

comparing a time duration stored in memory against a predetermined time duration by the service gateway;

determining whether the time duration stored in memory exceeds the predetermined time duration;

in response to determining that the time duration stored in memory exceeds the predetermined time duration, determining by the service gateway that the hybrid-stateful condition is satisfied; and

in response to determining that the time duration stored in memory does not exceed the predetermined time duration, determining by the service gateway that the hybrid-stateful condition is not satisfied.

**14**. The method of claim **13**, wherein the checking by the service gateway whether a hybrid-stateful condition is satisfied comprises receiving from an administrator the hybrid-stateful condition by the service gateway.

**15**. The method of claim **14**, wherein the administrator comprises:

a human operator;

a network management system; or

a storage medium storing the hybrid-stateful condition.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

# CERTIFICATE OF CORRECTION



PATENT NO. : 9,270,774 B2
APPLICATION NO. : 14/520126
DATED : February 23, 2016
INVENTOR(S) : Rajkumar Jalan et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the claims,

Column 18, line 38 cancel the text beginning with "14. The method of claim 13" and ending with "by the service gateway." in column 18, line 41, and insert the following claim:

--14. The method of claim 9, wherein the checking by the service gateway whether a hybrid-stateful condition is satisfied comprises receiving from an administrator the hybrid-stateful condition by the service gateway.--

Signed and Sealed this
Seventh Day of June, 2016

Michelle K. Lee

Michelle K. Lee
*Director of the United States Patent and Trademark Office*