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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 January 2017 (26.01.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/014757 A1

(51) International Patent Classification:
H04L 12/933 (2013.01) *H04L 12/935* (2013.01)

(21) International Application Number:
PCT/US2015/041352

(22) International Filing Date:
21 July 2015 (21.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **VALVERDE GARRO, Diego**; UltraPark, La Aurora, Heredia, Building 8, San Jose (CR). **VIQUEZ CALDERON, Claudio Enrique**; UltraPark, La Aurora, Heredia, Building 8, San Jose (CR).

(74) Agents: **HEWLETT PACKARD ENTERPRISE** et al.; 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: USING A SINGLE CACHE TABLE

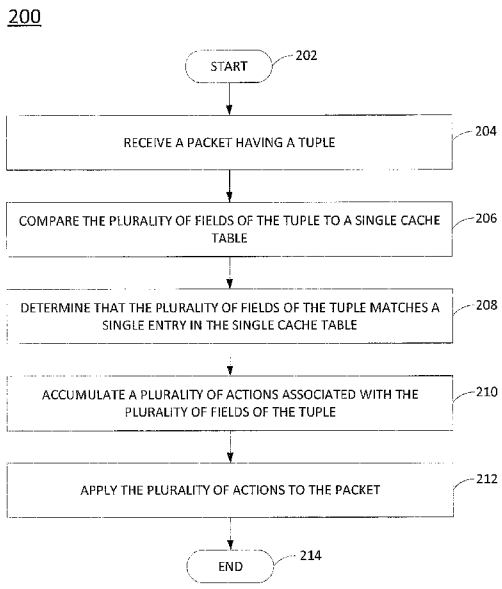

FIG. 2

(57) Abstract: In an example implementation, a method is provided. The method includes receiving at a switch a packet having a tuple, wherein the tuple comprises a plurality of fields, wherein the switch operates using a protocol that compares each one of the plurality of fields to a table of a plurality of tables in series. The switch compares the plurality of fields of the tuple to a single cache table instead of the plurality of tables in series. When a match is found, a plurality of actions associated with the plurality of fields of the tuple is accumulated. The plurality of actions is applied to the packet.

WO 2017/014757 A1

USING A SINGLE CACHE TABLE

BACKGROUND

[0001] OpenFlow is a standard protocol for communication between the control and forwarding layers of a software defined network architecture. When a packet enters a switch in an OpenFlow implementation, multiple fields in the tuple are matched against a series of corresponding tables for each field to determine an action or routing for the packet.

[0002] A typical flow tuple can be very large. For example, a tuple can include a 128 bit wide Internet Protocol version 6 (IPV6) fields, port number, and the like. Thus, searching large fields of a tuple in series in accordance with the OpenFlow protocol can be inefficient.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] FIG. 1 is an example switch of the present disclosure;

[0004] FIG. 2 is a flowchart of an example method for using a single cache table in a switch; and

[0005] FIG. 3 is a flowchart of an example method for merging entries in the single cache table.

DETAILED DESCRIPTION

[0006] The present disclosure discloses a method, system and apparatus for

using a single cache table in a switch. As discussed above a single packet may be matched against multiple tables for each different field in an OpenFlow implementation. A packet flow identifier can be defined as a tuple consisting of packet header fields. When a packet enters a switch in an Open Flow implementation, multiple fields in the tuple are matched against a series of corresponding tables for each field to determine an action or routing for the packet.

[0007] Examples of the present disclosure provide a method for using a single cache table in the switch rather than separate tables in series for each field of the tuple. In one example, the present disclosure may transform the fields of the tuple into a unique key by applying a mathematical function to the tuple. The single cache table may comprise a plurality of entries. Each entry may include a key and associated actions. As a result, when a packet enters a switch, the key calculated by applying the mathematical function to the tuple of fields of the packet may be compared to a single cache table rather than multiple tables in series to accumulate the associated actions for the packet.

[0008] In addition, as new entries are added to the single cache table, a compression algorithm may be applied to the single cache table as a background process. The compression algorithm may merge multiple entries having at least one common field into a single entry. As a result, the number of entries that are compared to the key are further reduced, thereby, further improving the efficiency of using the single cache table in a switch using the OpenFlow protocol.

[0009] FIG. 1 illustrates an example switch 100 of the present disclosure. In one example, the switch 100 includes a processor 102, a computer readable memory 104, an ingress (in) port 114 and an egress (out) port 116. In one example, the processor 102 may be a central processing unit (CPU) or an application specific integrated circuit (ASIC) processor.

[0010] In one example, the switch 100 may receive a packet 118 via the ingress port 114 and transmit the packet 118 via the egress port 116. Although a single packet 118 is illustrated in FIG. 1, it should be noted that the switch 100 may process many packets 118 continuously. In one example, the packet 118

may include a tuple 120 of fields used by the switch 100 to identify and apply actions to the packet 118 before the packet 118 leaves the switch 100.

[0011] In one example, the fields of the tuple 120 may be any type of fields used by the switch 100 to apply an action to the packet 118 using the OpenFlow protocol.

[0012] In one example, the switch 100 may operate using an OpenFlow protocol to identify and apply the actions to the packet 118. However, the present disclosure may be used for any switch communication protocol that uses a series of look up tables to be used to accumulate actions for each field of a tuple of fields of a packet. The OpenFlow protocol is one example of a switch communication protocol that uses a series of look up tables.

[0013] In one example, the computer readable medium may include a single cache table 106 and a series of tables 108-1 to 108-N (herein referred to individually as table 108 or collectively as tables 108). In one example, the tables 108 are the tables that are used for the OpenFlow protocol. Examples of tables used in the OpenFlow protocol include flow tables, group tables, and the like. For example, each table 108 may include a column of field values 110 and a column of actions 112.

[0014] In one example, the fields may include an identification number, a priority value, a label, a traffic class, a time to live (TTL) value, a port number, and the like. In one example, the actions may include: copy, pop, push, decrement, set, quality of service actions, group, output, and the like.

[0015] When a value of a first field matches an entry in the table 108-1, a particular action may correspond for that match. Then, a value of a second field is matched against an entry in the table 108-2 to find a corresponding action for that match. This is repeated for the Nth field and the Nth table 108-N. The actions are accumulated and applied to the packet 118 before the packet leaves the switch 100.

[0016] In one example, many packets 118 may have common fields that require the same actions found in the tables 108. As a result, the same searches on the same fields, resulting in the same group of actions may be performed. However, this can lead to inefficiencies in the switch 100 due to the

serial and repetitive nature of the searching performed in the tables 108 for the OpenFlow protocol.

[0017] In one example, the switch 100 may use the single cache table 106 instead of the series of tables 108 to find all of the actions associated with the fields of the tuple 120 of the packet 118. In one example, the single cache table 106 may include a column of keys 122 and a column of actions 124. In one example, the column of keys 122 may simply include a plurality of columns for each field in the tuple 120.

[0018] In one example, a mathematical function may be applied to the tuple 120 to calculate a unique key that is stored as an entry in the column of keys 122. In a first instance of seeing a particular tuple 120, the tables 108 may be used to identify the corresponding actions. The group of actions may be associated with the tuple 120 and stored in the actions column 124 for the associated key that corresponds to the tuple 120.

[0019] In one example, the mathematical function may be a hash function. In one example, any type of hash function may be used (e.g., a Bernstein hash function, a Fowler-Noll-Vo hash function, a Jenkins hash function, and the like). In one example, the mathematical function may be programmed into the processor 102 when the processor 102 is an ASIC chip.

[0020] Illustrating an example of how the mathematical function is applied, the tuple 120 may include three fields and be 128 bits long. However, a hash function may be applied to reduce the tuple to a single unique key of 19 bits. Thus, additional efficiency is gained by requiring a search on a smaller amount of information due to applying the hash function to tuple 120.

[0021] The next time the tuple 120 is received, the processor 102 may apply the mathematical function to the tuple, calculate the key and search the single cache table 106 for the key. When a match is found, the processor 102 may identify the actions associated with the key and apply the actions to the packet 118 before transmitting the packet 118 out the egress port 116 onto the next destination of the packet 118.

[0022] In one example, applying the actions may include executing the actions that are accumulated based upon the match as described above.

However, in one example, one of the actions in the OpenFlow protocol may be a "goto" action. Thus, applying the action may actually require sending the packet to another table within the OpenFlow protocol to accumulate additional actions. Thus, the term "applying" may include executing actions or moving the packet to another table when a "goto" action is applied.

[0023] In another example, applying the action may include dropping the packet and performing no action. For example, the OpenFlow protocol may drop packets for a "table-miss" (e.g., no match is found for the field in a table). However, in one example, the present disclosure creates a new entry in the single cache table 106 when no match is found. Subsequently, when the fields are matched against the series of tables 108 and a "table-miss" occurs, a "drop" action may be associated with the new entry in the single cache table 106.

[0024] To illustrate by example, a tuple (x, y, z) 120 for a packet 118 may be received at the switch 100. A mathematical function f(x, y, z) may be applied to the tuple 120 to calculate a unique key of 19 bits. The single cache table 106 may be searched to see if the unique key matches an entry in the single cache table 106.

[0025] In one example, a match is found and the actions push, pop and decrement may be associated with the tuple (x, y, z) 120. Thus, the actions push, pop and decrement are accumulated and applied to the packet 118.

[0026] Notably, the switch is programmed to use the OpenFlow protocol and the tables 108 for the OpenFlow protocol are stored in the computer readable memory 104. However, the processor 102 first applies the mathematical function to the tuple 120 and compares the calculated key to entries in the single cache table 106. In other words, the single cache table 106 is used initially, or first, instead of using the tables 108 to execute the OpenFlow protocol.

[0027] In one example, if no match is found, the tuple (x, y, z) 120 may be a new tuple that the switch 100 has not received. If the tuple (x, y, z) 120 is a new tuple 120, then a new entry can be created in the single cache table 106. For example, the processor 102 may use the tables 108 to perform a traditional OpenFlow protocol search for each field x, y and z of the tuple 120 and

accumulate the associated actions. The key calculated by applying the hash function on the tuple (x, y, z) 120 may be stored in the column of keys 122 as a new entry and the actions obtained from the tables 108 may be stored in the column of actions 124 for the new entry.

[0028] In one example, the single cache table 106 may be a ternary content-addressable memory (TCAM) table. A TCAM table allows for a third matching state that includes a wildcard (e.g., an X) or a "don't care" for one or more bits of the stored data word. For example, a search for 1X0 would return matches of 110 and 100.

[0029] In one example, the single cache table 106 may be further compressed as a background operation in the switch 100. For example, two or more entries may have at least one common field. For example, two different keys in the column of keys 122 may have the same actions. As a result, the two different keys may be merged into a single entry (e.g., using a new key that corresponds to both keys or including both keys in the same entry) and be associated with the same actions. In another example, when the single cache table 106 uses a column for each field instead of a mathematical function, some common fields with the same action may be merged into a single entry. As a result, by continuously compressing the single cache table 106 additional efficiencies are gained. For example, compressing the single cache table 106 reduces a number of entries that need to be scanned to match a key of a tuple 120 or the fields of the tuple 120.

[0030] FIG. 2 illustrates an example flowchart of a method 200 for using a single cache table in a switch. In one example, the method 200 may be performed by the processor 102 of the switch 100.

[0031] At block 202 the method 200 begins. At block 204, the method 200 receives a packet having a tuple. For example, the switch may receive the packet via an ingress port. The packet may have a tuple of fields that are usually searched in a plurality of tables in series to identify an action for each field match. One example of a switch protocol that uses a series tables is the OpenFlow protocol.

[0032] At block 206, the method 200 compares the plurality of fields of the

tuple to a single cache table. In one example, instead of using the plurality of tables in series, the switch may compare the plurality of fields of the tuple to entries in the single cache table.

[0033] In one example, the single cache table may include a plurality of entries associated with tuples of fields that the switch has previously received and the associated actions for those tuples. As a result, instead of searching multiple tables in series, the present disclosure may require a single look up to accumulate all of the actions that would otherwise have required multiple look ups in each of the plurality of tables.

[0034] In one example, a mathematical function may be applied to the fields of the tuple to create a key. In one example, the mathematical function may be a hash function that reduces the number of bits of the tuple (e.g., 128 bits) to 19 bits.

[0035] In one example, the single cache table may be a TCAM table. As a result, the hash function may include wildcards or don't care values (e.g., an "X", an "*", and the like).

[0036] At block 208, the method 200 determines that the plurality of fields of the tuple matches a single entry in the single cache table. In one example, the fields of the tuple or the key calculated by applying the mathematical function to the fields of the tuple may be compared against each entry in the single cache table. In one example, a match may be found.

[0037] In one example, if no match is found, a new entry can be created in the single cache table. For example, the new entry may include the key that was calculated by applying the mathematical function. In addition, the plurality of tables in series may be used to look up each action for each field of the tuple. The accumulated actions may then be stored in the single cache table with the new entry. When the same tuple is received by the switch at a later time, a match may be found and looking up the plurality of tables in series may be avoided.

[0038] At block 210, the method 200 accumulates a plurality of actions associated with the plurality of fields of the tuple. For example, when a match is found, the actions associated with the matching entry in the single cache table

are accumulated.

[0039] At block 212, the method 200 applies the plurality of actions to the packet. In one example, applying the actions may include executing the actions that are accumulated based upon the match as described above. However, in one example, one of the actions in the OpenFlow protocol may be a "goto" action. Thus, applying the action may actually require sending the packet to another table within the OpenFlow protocol to accumulate additional actions. Thus, the term "applying" may include executing actions or moving the packet to another table when a "goto" action is applied.

[0040] In another example, applying the action may include dropping the packet and performing no action. For example, the OpenFlow protocol may drop packets for a "table-miss" (e.g., no match is found for the field in a table). However, in one example, the present disclosure creates a new entry in the single cache table when no match is found. Subsequently, when the fields are matched against the series of tables and a "table-miss" occurs, a "drop" action may be associated with the new entry in the single cache table. At block 214, the method 200 ends.

[0041] FIG. 3 illustrates an example flowchart of another method 300 for merging entries in the single cache table that is used in the switch. In one example, the method 300 may be performed by the processor 102 of the switch 100.

[0042] At block 302 the method 300 begins. At block 304, the method 300 creates a single cache table. In one example, the single cache table comprises a plurality of entries. The single cache table can be used to identify actions for a packet in the switch using an OpenFlow protocol by comparing a plurality of fields of a tuple of the packet entering the switch to the plurality of entries in the single cache table to find a match.

[0043] In other words, unlike traditional Open Flow switches that use a series of flow tables to accumulate actions for each field of the tuple, the present disclosure uses the single cache table. As a result, efficiencies are gained by using a single search on a single table as opposed to multiple searches on multiple tables.

[0044] In addition, the single cache table may include entries that are keys of shorter bit length than the bit length of the tuple. The keys can be calculated by applying a mathematical function to the fields of the tuple (e.g., a hash function to reduce a 128 bit tuple to a 19 bit key). As a result, additional efficiencies are gained by matching shorting data words (e.g., there are less numbers of bits to compare for each comparison).

[0045] In one example, the single cache table may be a TCAM table. As a result, the hash function may include wildcards or don't care values (e.g., an "X", an "*", and the like).

[0046] At block 306, the method 300 scan the single cache table to identify two or more entries that have at least one common field. In one example, some entries within the single cache table may have some entries that have at least one common field. For example, two different key entries may have the same actions (e.g., the at least one common field).

[0047] Thus, further efficiencies can be gained in processing the packet through the switch if the single cache table can be compressed. In other words, the lower number of entries in the single cache table, the faster the comparison between the fields of the tuple and the single cache table.

[0048] At block 308, the method 300 merges the two or more entries into a new single entry in the single cache table. As noted above, further efficiencies can be gained in processing the packet through the switch if the single cache table can be compressed. As a resulting, merging the two or more entries that have at least one common field into a single entry reduces the number of entries and the overall size of the single cache table.

[0049] At block 310, the method 300 determines if all the entries of the single cache table have been scanned. If there are additional entries that need to be scanned (e.g., the answer is no), then the method 300 may return to block 306 and repeat block 306 and block 308 until all of the entries have been scanned. In other words, the scanning and the merging may be repeated until all of the plurality of entries have been scanned.

[0050] If the there are no additional entries that need to be scanned (e.g., the answer is yes), then the method 300 may proceed to block 312. At block 312,

the method 300 ends.

[0051] It should be noted that although not explicitly specified, any of the blocks, functions, or operations of the example methods 300 and 400 described above may include a storing, displaying, and/or outputting block. In other words, any data, records, fields, and/or intermediate results discussed in the methods can be stored, displayed, and/or outputted to another device. Furthermore, blocks, functions, or operations in FIGs. 2 and 3 that recite a determining operation, or involve a decision, do not necessarily require that both branches of the determining operation be practiced.

[0052] It will be appreciated that variants of the above-disclosed and other features and functions, or alternatives thereof, may be combined into many other different systems or applications. Various presently unforeseen or unanticipated alternatives, modifications, or variations, therein may be subsequently made which are also intended to be encompassed by the following claims.

CLAIMS

1. A method, comprising:

receiving, by a processor of a switch, a packet having a tuple, wherein the tuple comprises a plurality of fields, wherein the switch operates using a protocol that compares each one of the plurality of fields to a table of a plurality of tables in series;

comparing, by the processor, the plurality of fields of the tuple to a single cache table instead of the plurality of tables in series;

determining, by the processor, that the plurality of fields of the tuple matches a single entry in the single cache table;

accumulating, by the processor, a plurality of actions associated with the plurality of fields of the tuple; and

applying, by the processor, the plurality of actions to the packet.

2. The method of claim 1, further comprising:

determining, by the processor, that the plurality of fields of the tuple does not match any entry in the single cache table; and

creating, by the processor, a new entry in the single cache table for the plurality of fields of the tuple.

3. The method of claim 1, wherein the protocol comprises an OpenFlow protocol.

4. The method of claim 1, wherein the single cache table comprises a ternary content-addressable memory (TCAM) table.

5. The method of claim 1, wherein the comparing, further comprises:

applying, by the processor, a function to the plurality of fields of the tuple to calculate a unique key; and

performing, by the processor, a look up in the single cache table on the unique key.

6. The method of claim 5, wherein the function comprises a hash function.

7. A switch, comprising:

an ingress port;

an egress port;

a processor; and

a non-transitory computer readable memory storing a plurality of tables for an OpenFlow protocol, a single cache table and a plurality of instructions, which when executed by the processor cause the processor to:

compare a plurality of fields of a tuple of a packet received via the ingress port to the single cache table;

determine that the plurality of fields of the tuple matches a single entry in the single cache table;

accumulate a plurality of actions associated with the plurality of fields of the tuple;

apply the plurality of actions to the packet; and

transmit the packet via the egress port.

8. The switch of claim 7, wherein the processor comprises an application specific integrated circuit (ASIC) chip.

9. The switch of claim 7, wherein the processor compares each one of the plurality of fields to a corresponding one of the plurality of tables in series to accumulate actions for the plurality of fields when the plurality of fields do not match any entry in the single cache table.

10. The switch of claim 9, wherein a new entry is created in the single cache table with the actions that are accumulated for the plurality of fields of the tuple.

11. A method, comprising:

creating, by a processor of a switch, a single cache table, wherein the

single cache table comprises a plurality of entries, wherein the single cache table is used to identify actions for a packet in the switch using an OpenFlow protocol by comparing a plurality of fields of a tuple of the packet entering the switch to the plurality of entries in the single cache table to find a match;

scanning, by the processor, the single cache table to identify two or more entries that have at least one common field;

merging, by the processor, the two or more entries into a new single entry in the single cache table; and

repeating, by the processor, the scanning and the merging until all of the plurality of entries have been scanned.

12. The method of claim 11, wherein the scanning, the merging and the repeating are performed as a background process in the switch.

13. The method of claim 11, wherein each of the plurality of entries comprises a unique key calculated by applying a function to the plurality of fields of the tuple.

14. The method of claim 11, wherein the function comprises a hash function.

15. The method of claim 11, wherein the two or more entries comprise different unique keys and the at least one common field comprises the actions.

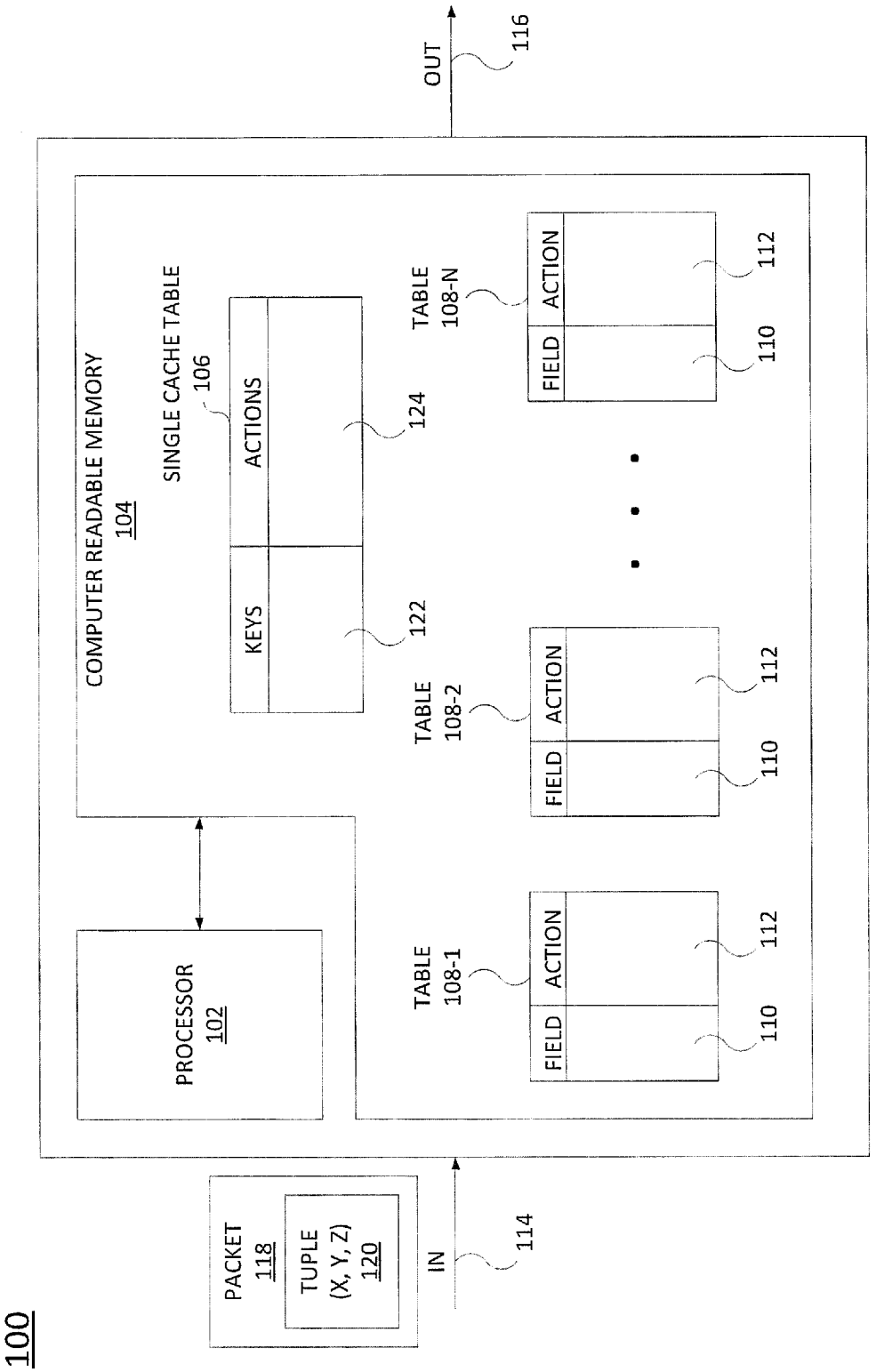
100
PACKET
118
TUPLE
(X, Y, Z)
120
IN
114
OUT
116
COMPUTER READABLE MEMORY
104
PROCESSOR
102
SINGLE CACHE TABLE
106
KEYS
ACTIONS
122
124
TABLE
108-1
TABLE
108-2
TABLE
108-N
FIELD
ACTION
110
112

FIG. 1

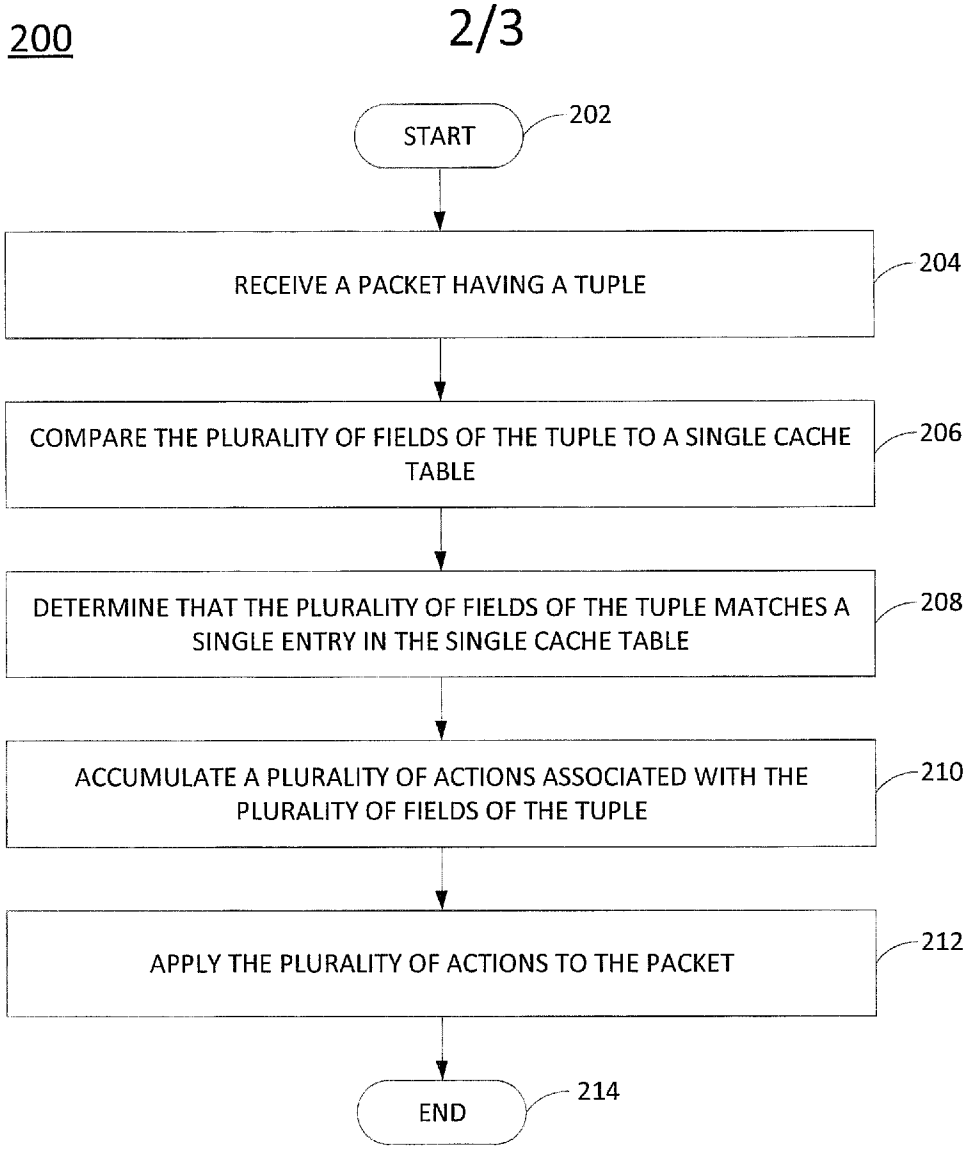
200
START
202
RECEIVE A PACKET HAVING A TUPLE
204
COMPARE THE PLURALITY OF FIELDS OF THE TUPLE TO A SINGLE CACHE TABLE
206
DETERMINE THAT THE PLURALITY OF FIELDS OF THE TUPLE MATCHES A SINGLE ENTRY IN THE SINGLE CACHE TABLE
208
ACCUMULATE A PLURALITY OF ACTIONS ASSOCIATED WITH THE PLURALITY OF FIELDS OF THE TUPLE
210
APPLY THE PLURALITY OF ACTIONS TO THE PACKET
212
END
214

FIG. 2

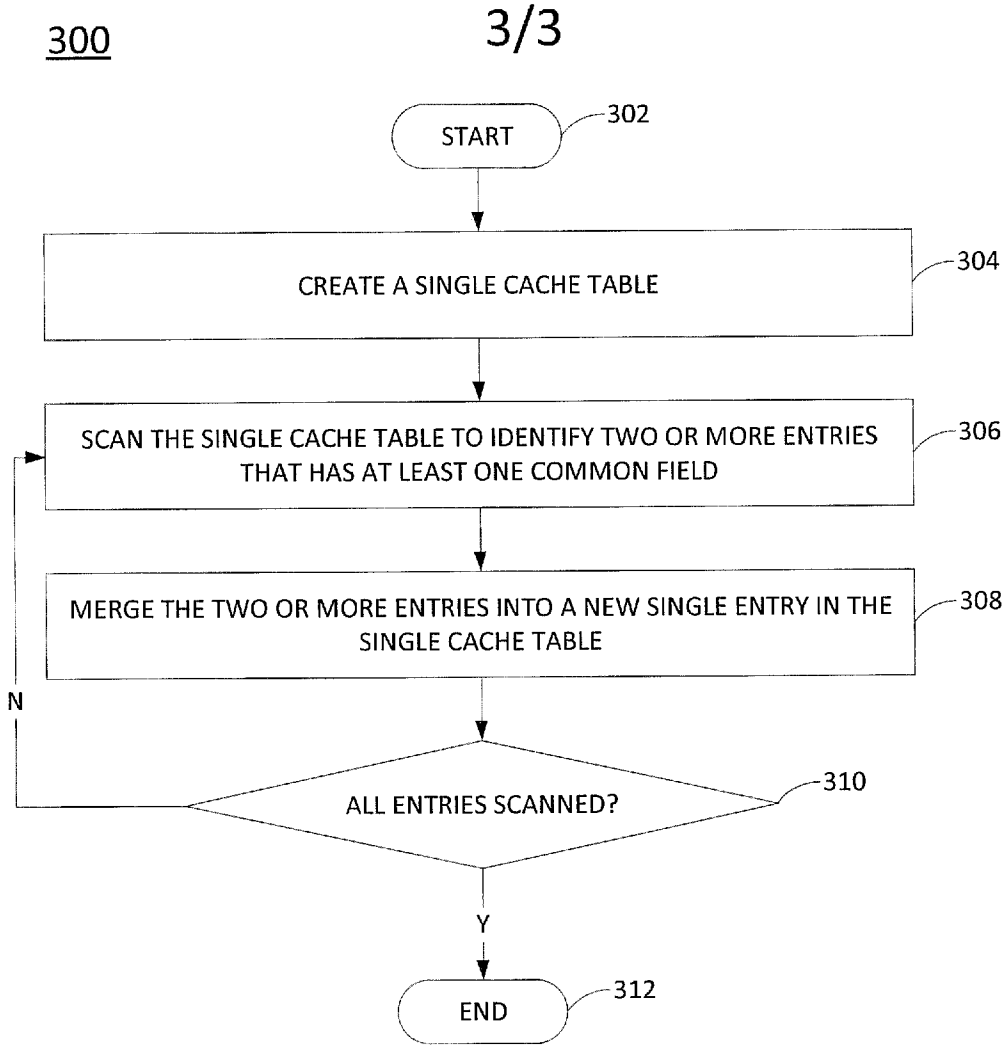
300
START
302
CREATE A SINGLE CACHE TABLE
304
SCAN THE SINGLE CACHE TABLE TO IDENTIFY TWO OR MORE ENTRIES THAT HAS AT LEAST ONE COMMON FIELD
306
MERGE THE TWO OR MORE ENTRIES INTO A NEW SINGLE ENTRY IN THE SINGLE CACHE TABLE
308
ALL ENTRIES SCANNED?
310
N
Y
END
312

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/041352

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/933(2013.01)i, H04L 12/935(2013.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L 12/933; H04L 12/743; G06F 12/00; H04L 29/06; H04L 12/28; H04L 12/935

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: tuple, single cache table, match, action, single entry

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2012-0246400 A1 (SANDEEP BHADRA et al.) 27 September 2012 See paragraphs [0004], [0007], [0015], [0020], [0037]-[0038], [0043], [0047]-[0048]; claims 10, 12, 14, 16; and figures 6, 8-9.	1-15
A	US 2013-0290622 A1 (SUDDHA SEKHAR DEY et al.) 31 October 2013 See paragraphs [0033]-[0038]; and figures 3-4.	1-15
A	US 2014-0241361 A1 (TEXAS INSTRUMENTS INCORPORATED) 28 August 2014 See paragraphs [0027], [0031], [0060], [0105]-[0106]; and figure 7.	1-15
A	WO 2013-093858 A1 (TELEFONAKTIEBOLAGET L M ERICSSON (PUBL)) 27 June 2013 See page 23, lines 21-27; page 25, lines 18-30; claim 1; and figures 1-2.	1-15
A	US 2002-0089937 A1 (SRINIVASAN VENKATACHARY et al.) 11 July 2002 See paragraphs [0006]-[0007]; and figure 1.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 April 2016 (08.04.2016)	**30 May 2016 (30.05.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
KIM, Seong Woo
Telephone No. +82-42-481-3348

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/041352

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012-0246400 A1	27/09/2012	JP 2014-512121 A	19/05/2014
		US 8874876 B2	28/10/2014
		WO 2012-129432 A2	27/09/2012
		WO 2012-129432 A3	06/12/2012
US 2013-0290622 A1	31/10/2013	US 8886879 B2	11/11/2014
US 2014-0241361 A1	28/08/2014	None	
WO 2013-093858 A1	27/06/2013	CN 103999430 A	20/08/2014
		EP 2795873 A1	29/10/2014
		US 2013-0163426 A1	27/06/2013
		US 2014-204948 A1	24/07/2014
		US 8718064 B2	06/05/2014
		US 9077668 B2	07/07/2015
US 2002-0089937 A1	11/07/2002	US 7978709 B1	12/07/2011