

(51) International Patent Classification:
G06F 12/12 (2006.01)

(21) International Application Number:

PCT/EP2011/003827

(22) International Filing Date:

29 July 2011 (29.07.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

10290430.7 30 July 2010 (30.07.2010) EP

(71) Applicant (for all designated States except US): **ALCATEL LUCENT** [FR/FR]; 3, avenue Octave Gréard, F-75007 Paris (FR).

(72) Inventors; and

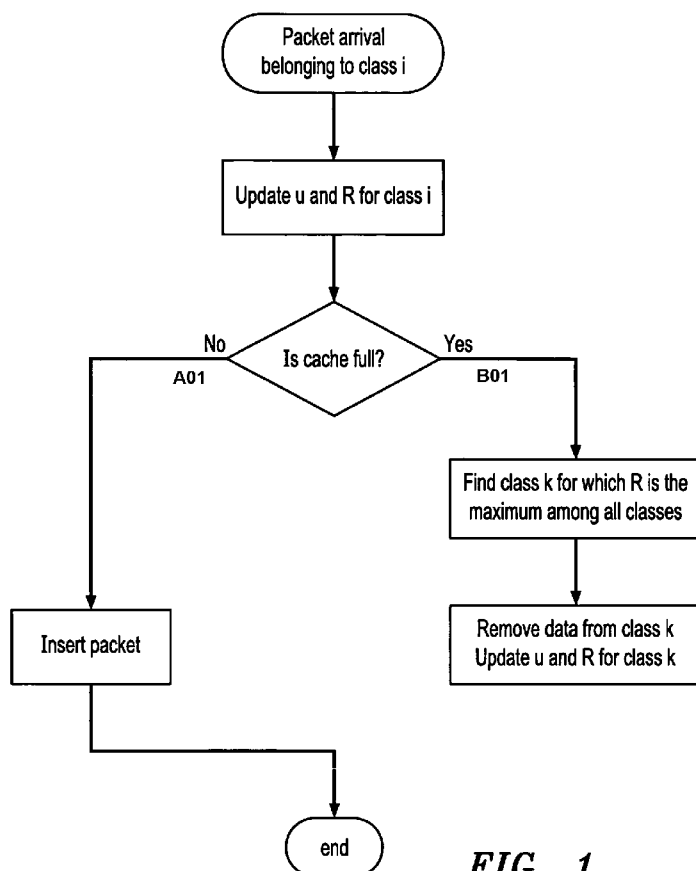
(75) Inventors/Applicants (for US only): **PERINO, Diego** [IT/FR]; 2 villa Haussmann, F-92130 Issy-les-Moulin-eaux (FR). **CAROFIGLIO, Giovanna** [IT/FR]; 49 avenue Sully Prudhomme, F-92290 Châtenay-Malabry (FR).(74) Agents: **COCKAYNE, Gillian** et al.; Alcatel-lucent Telecom Ltd., Unit 18, Core 3, Workzone, Innova Business Park, Electric Avenue, Enfield EN3 7XU (GB).

(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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, 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,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR ALLOCATING MEMORY IN A CACHE

**FIG. 1**

(57) Abstract: A method of using memory in a cache for a plurality different classes of content includes assigning respective weights to respective classes of content and guaranteeing a respective minimum amount of memory for each class of content to store valid data. The guaranteed minimum amount for each class is proportional to the given weight for that class. When the guaranteed minimum amount for a first class is not fully occupied by data in that class, one or more of the other classes temporarily re-use spare memory capacity of the guaranteed minimum amount for the first class. Content for classes that have exceeded their capacity is removed using class weights to determine the class of content to be removed.



GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

- 1 -

METHOD AND APPARATUS FOR ALLOCATING MEMORY IN A CACHE

FIELD OF THE INVENTION

The present invention relates to a method and apparatus for allocating memory in a cache, and more particularly for allocating memory for a plurality of different
5 classes of content

BACKGROUND OF THE INVENTION

A network cache is a component that transparently stores data to serve future content requests. Data stored within the cache is a copy of an original element that is
10 stored elsewhere. If a client or user generates a request for an element stored in the cache, that request is served by reading the cache without contacting the original data source. Thus caching may improve performance, for example, by enabling a faster response time, requiring less bandwidth and the like. If the requested data is not stored in the cache, the request is routed towards the original source or redirected to
15 other caches.

Each cache has a limited storage capacity which is managed by a cache replacement strategy. The strategy is used to select which data should be removed from the cache when new data must be stored. Also, data stored in a cache is valid only for a limited period of time after which it becomes obsolete. For instance, a copy
20 of given webpage is not valid once it has been updated at the original server. Such obsolete data is the first to be removed from a cache to make room for new elements. Cache coherence is concerned with ensuring that up to date copies of the original content are stored in the relevant cache or caches.

Several cache replacement and coherence mechanisms have been proposed to
25 manage caches, for example, to manage web page content, by adding and removing an entire web object.

BRIEF SUMMARY

According to a first aspect of the invention, a method of using memory in a

- 2 -

cache for a plurality different classes of content includes assigning respective weights to respective classes of content and guaranteeing a respective minimum amount of memory for each class of content to store valid data. The guaranteed minimum amount for each class is proportional to the given weight for that class. When the
5 guaranteed minimum amount for a first class is not fully occupied by data in that class, one or more of the other classes temporarily re-use spare memory capacity of the guaranteed minimum amount for the first class. Content for classes that have exceeded their capacity is removed using class weights to determine the class of content to be removed.

10 Weights are assigned to classes and the method guarantees that every class can exploit a minimum amount of memory, proportional to the weights. The eviction policy determines from which class to remove data and how 'extra' memory is shared among classes using different criteria.

 The invention may be applied to networks with caching capabilities such as
15 Content Centric Networks, Redundancy Elimination services, and the like. Performance of applications that generate content may be strongly impacted by the amount of memory space devoted at every node to their classes.

 In one embodiment, a diversity of web objects, for example, may be handled appropriately for each class. Web objects include, but are not limited to, web pages,
20 multimedia streams and images, for example.

 The method may be used to guarantee a minimum amount of memory to each class of content to store valid, that is, non-obsolete, data while providing a flexible and efficient use of the whole memory space. The method may be applied at object granularity and/or packet granularity levels.

25 In one embodiment, when the guaranteed minimum amount for one class is not fully occupied by data in that class, spare memory capacity of the guaranteed minimum amount is temporarily re-allocated to one or more of the other classes. Thus, if some classes do not completely exploit their devoted memory space because they do not have enough valid data, in this embodiment, other classes may temporarily
30 use the spare capacity.

- 3 -

In one embodiment, spare memory capacity is used according to class weights.
In another embodiment, spare capacity is uniformly allocated.

According to a second aspect of the invention, there is provided apparatus operative to implement a method in accordance with the first aspect.

5 According to a third aspect of the invention, there is provided a data storage medium storing a machine-executable program for performing a method in accordance with the first aspect.

BRIEF DESCRIPTION OF THE DRAWINGS

10 Some embodiments of the present invention will now be described by way of example only, and with reference to the accompanying drawings, in which:

Figure 1 is a schematic flowchart of a method in accordance with the invention;

15 Figure 2 schematically illustrates an apparatus in accordance with the invention.

DETAILED DESCRIPTION

20 In one embodiment of the invention, content to be stored in a cache may be denoted as being included in a particular class.

Consider a cache of size C and a set of content classes N . A weight w_i is associated with every class $i \in N$. The minimum amount of memory guaranteed to a class i is denoted as c_i^* and is given by:

25
$$c_i^* = \frac{w_i}{\sum_{n \in N} w_n} C \quad \text{-- equation (1)}$$

The amount of memory used by class i to store valid data is denoted as u_i , and D is the set of classes for which $c_i^* - u_i > 0$. This set D contains classes that do not have enough valid data to completely fill their guaranteed minimum amounts of

- 4 -

memory, valid data being non-obsolete data. The total amount of guaranteed minimum memory space that is unused by these classes in set D is denoted as F and is given by:

$$F = \sum_{n \in D} c_n^* - u_n \quad \text{-- equation (2)}$$

This additional memory space F is thus available for use by classes $N \setminus D$ that are not included in set D .

Different strategies may be used for sharing the additional memory space F among classes $N \setminus D$. In one mechanism, the additional memory space F is uniformly distributed among demanding classes, that is, $f_i = \frac{F}{|N \setminus D|}$. In another mechanism, additional memory space F is assigned according to priority criteria. In another mechanism, a FIFO (first in, first out) approach is used.

In one strategy, the additional space F is shared according to class weights. By sharing according to class weightings, proportional memory assignment among classes may be maintained. Using a weighting approach, a class $i \in N \setminus D$ receives a minimum amount of additional space as given by the following expression:

$$f_i = \frac{w_i}{\sum_{n \in N \setminus D} w_n} F \quad \text{-- equation (3)}$$

20

$R(u_i)$ is a function which is used to quantify the excess of each class i . Data is removed from memory allocated to a class k for which R_k is greatest compared to that for other classes. The flowchart of Figure 1 illustrates the mechanism applied when new content is received. If more than one class has the same maximum value for R , a choice is made at random as to which of these classes to select for data removal. Data is removed only from those classes that have exceeded their capacity c^* . For function R , $R(u_i) < R(u_j) \forall i \in D \forall j \in N \setminus D$.

25

- 5 -

The strategy used to share the additional memory space among classes is related to the particular mechanism used to remove data from the cache. For example, where there is a strategy of uniform allocation of additional memory space,

$$5 \quad F \begin{cases} R(u_i) = 0 & \text{if } u_i \leq c_i^* \\ R(u_i) = u_i & \text{otherwise} \end{cases} \text{ for all classes}$$

In contrast, and as now described in more detail, where the strategy is such that the additional space F is shared according to class weights and there is proportional allocation of additional memory space F ,

$$10 \quad R_k(u_k) = \frac{u_k}{w_k}$$

For this strategy, consider a cache storing three classes of content, $N = \{\text{Class 1, Class 2, Class 3}\}$, with weights $W_1=2$ $W_2=1$ $W_3=2$. It is further assumed that the cache size is equal to $C=10$ units and that there is proportional allocation of additional memory F .

15 According to equation (1): $c_1^*=4$ units, $c_2^*=2$ units, $c_3^*=4$ units.

Consider the following cache initial condition:

Class	# of unit
1	4
2	2
3	4

20 In such scenario $u_1=4$, $u_2=2$, $u_3=4$, $R_1=2$, $R_2=2$, $R_3=2$.

The following sets out an illustrative exemplary sequence of events following the starting conditions shown in the table above.

- 6 -

a) Arrival of one unit of Class 1.

1. $u_1=5, u_2=2, u_3=4, R_1=2.5, R_2=2, R_3=2$.
2. A unit of class 1 is removed according to Class 1 replacement policy.
3. $u_1=4, u_2=2, u_3=4, R_1=2, R_2=2, R_3=2$.

5

Class	# of unit
1	4
2	2
3	4
10	

b) Class 3 coherence mechanism declares 3 units of Class 3 data as obsolete.

1. Obsolete units of Class 3 are removed from the cache.
2. $u_1=4, u_2=2, u_3=1, R_1=2, R_2=2, R_3=0.5$.

15

Class	# of unit
1	4
2	2
3	1

c) Arrival of one unit of Class 2.

1. $u_1=4, u_2=3, u_3=1, R_1=2, R_2=3, R_3=0.5$.

20

Class	# of unit
1	4
2	3
3	1

d) Arrival of one unit of Class 2.

- 7 -

1. $u_1=4, u_2=4, u_3=1, R_1=2, R_2=4, R_3=0.5$.

Class	# of unit
1	4
2	4
3	1

e) Arrival of one unit of Class 2.

- 5 1. $u_1=4, u_2=5, u_3=1, R_1=2, R_2=5, R_3=0.5$.

Class	# of unit
1	4
2	5
3	1

f) Arrival of one unit of Class 1.

- 10 1. $u_1=5, u_2=5, u_3=1, R_1=2.5, R_2=5, R_3=0.5$.
 2. A unit of Class 2 is removed according to Class 2 replacement policy.
 3. $u_1=5, u_2=4, u_3=1, R_1=2.5, R_2=4, R_3=0.5$.

Class	# of unit
1	5
2	4
3	1

g) Arrival of one unit of Class 1.

- 15 1. $u_1=6, u_2=4, u_3=1, R_1=3, R_2=4, R_3=0.5$.
 2. A unit of Class 2 is removed according to Class 2 replacement policy.
 3. $u_1=6, u_2=3, u_3=1, R_1=3, R_2=3, R_3=0.5$.

- 8 -

Class	# of unit
1	6
2	3
3	1

h) Arrival of one unit of Class 1.

1. $u_1=7, u_2=3, u_3=1, R_1=3.5, R_2=3, R_3=0.5$.
2. A unit of Class 1 is removed according to Class 1 replacement policy.
- 5 3. $u_1=6, u_2=3, u_3=1, R_1=3, R_2=3, R_3=0.5$.

Class	# of unit
1	6
2	3
3	1

i) Arrival of one unit of Class 3.

- 10 1. $u_1=6, u_2=3, u_3=2, R_1=3, R_2=3, R_3=1$.
2. A unit of Class 1 is removed according to Class 1 replacement policy (ties are broken at random).
3. $u_1=5, u_2=3, u_3=2, R_1=2.5, R_2=3, R_3=1$.

Class	# of unit
1	5
2	3
3	2

15

A replacement and coherence mechanism may also be applied per every class.

With reference to Figure 2, an apparatus for allocating memory in a cache

- 9 -

includes a classifier 1 which receives a packet from 2 and assigns a class i depending on the characteristics of the received packet. The packet is sent to the cache memory 3 to be stored. If there is insufficient memory for the new packet, a packet from class k is removed.

5 The processor 4, indicated by the broken line, has a cache statistics store 5 which holds information including the cache occupancy, u for all classes and R for all classes. The classes, weights and R function to be applied by the processor 4 are obtained from controller 6 and used at 7 in conjunction with information from classifier 1 to update u and R for class i . The updated values are held in store 5 and
10 also applied to a sorter 8 to enable the classes to be sorted according to their R values. This sort information is then used by selector 9 to select the class k with the highest R value and the result sent to the cache memory 3. Values of u and R for class k are updated at 10.

 The present invention may be embodied in other specific forms without
15 departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes that come within the meaning and range of equivalency of the claims are to be embraced within their scope.

20

- 10 -

CLAIMS

1. A method of using memory in a cache for a plurality different classes of content including:

assigning respective weights to respective classes of content;

5 guaranteeing a respective minimum amount of memory for each class of content to store valid data, the guaranteed minimum amount for each class being proportional to the given weight for that class;

and, when the guaranteed minimum amount for a first class is not fully occupied by data in that class, one or more of the other classes temporarily re-using spare memory capacity of the guaranteed minimum amount for the first class;

10 and including removing content for classes that have exceeded their capacity using class weights to determine the class of content to be removed.

2. The method as claimed in claim 1 and including allocating spare memory capacity according to class weights.

3. The method as claimed in claim 2 and including allocating memory f_i to a class $i \in N \setminus D$ from spare memory capacity F according to the relation

$$f_i = \frac{w_i}{\sum_{n \in N \setminus D} w_n} F.$$

20

4. The method as claimed in claim 1, 2 or 3 wherein the content to be removed is determined using the relation $R_k(u_k) = \frac{u_k}{w_k}$ and selecting content to be removed from a class having a maximum value of R .

25 5. The method as claimed in any preceding claim and wherein the spare memory capacity is uniformly distributed across classes requiring additional capacity.

6. The method as claimed in claim 5 and including removing content for classes that

- 11 -

have exceeded their capacity using the relation

$$F \begin{cases} R(u_i) = 0 & \text{if } u_i \leq c_i^* \\ R(u_i) = u_i & \text{otherwise} \end{cases} \text{ for all classes}$$

and selecting content to be removed from a class having a maximum value of R.

5

7. The method as claimed in any preceding claim and wherein the spare memory capacity distributed across classes requiring additional capacity according to priority criteria.

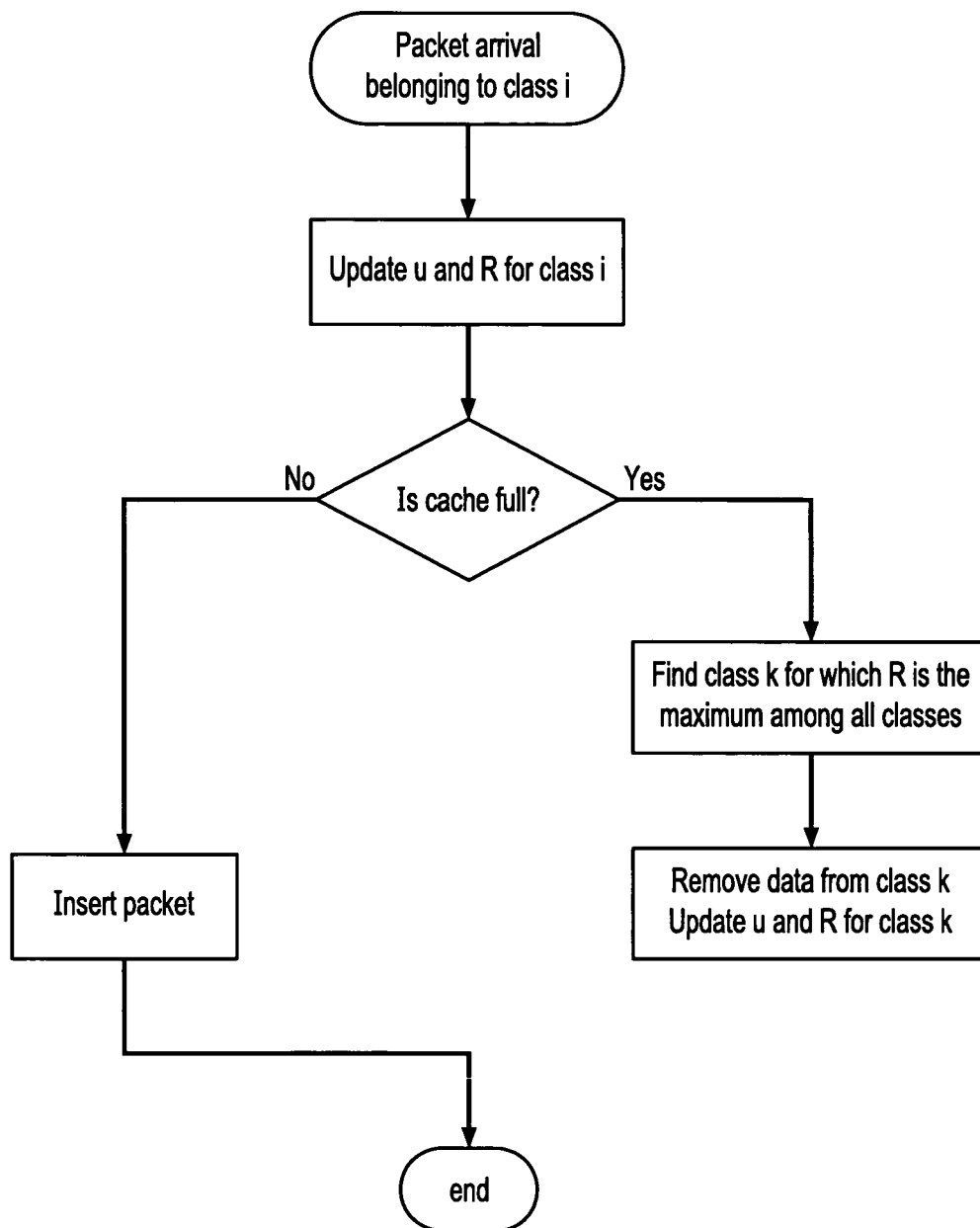
10 8. The method as claimed in any preceding claim and the spare memory capacity distributed across classes requiring additional capacity on a first in, first out basis.

9. Apparatus operative to carry out a method of using memory in a cache for a plurality different classes of content in which the method includes:

15 assigning respective weights to respective classes of content;
 guaranteeing a respective minimum amount of memory for each class of content to store valid data, the guaranteed minimum amount for each class being proportional to the given weight for that class;
 and, when the guaranteed minimum amount for a first class is not fully
 20 occupied by data in that class, one or more of the other classes temporarily re-using spare memory capacity of the guaranteed minimum amount for the first class;
 and including removing content for classes that have exceeded their capacity using class weights to determine the class of content to be removed.

25 10. A data storage medium storing a machine-executable program for performing a method for using memory in a cache for a plurality different classes of content including: guaranteeing a respective minimum amount of memory for each class of content to store valid data, the guaranteed minimum amount for each class being proportional to a given weight for that class.

1/2

**FIG. 1**

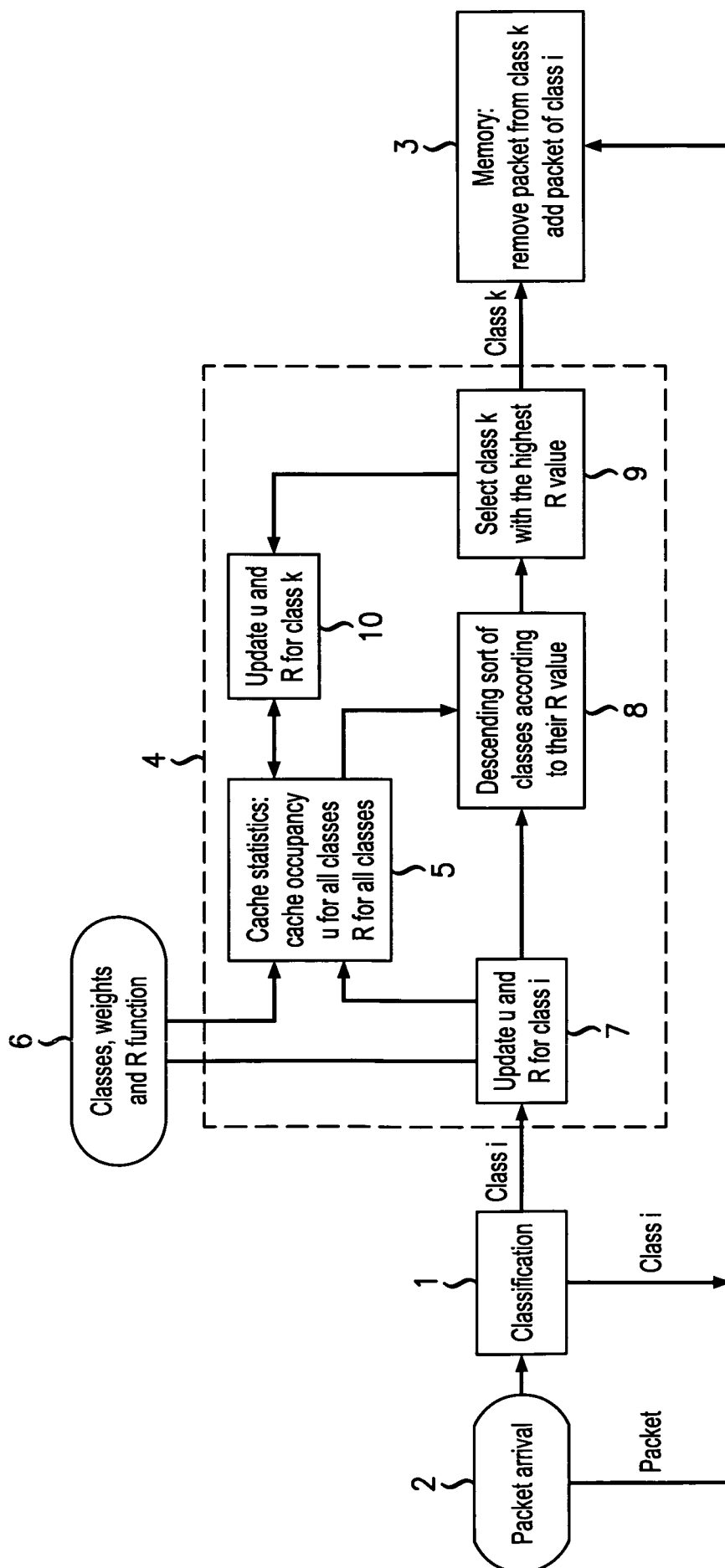


FIG. 2