object spaces are implemented. Every object within an agent has a global id. The contents of this identifier are sufficient to locate the object regardless of where it resides in the system.

[0108] A global id preferably contains the following data:

- An integer identifier;
- The subagent to which the instance belongs;
- "null" or the instance if it currently resides on the local base; and
- A reference count and any other information needed by the global garbage collector.

[0109] Forwarding pointers are needed if the object migrates from its original home. References which touch a forwarding pointer are updated to reflect the object's new location.

(h) <u>Communication Protocol</u>

[0110] One example of an implementation of a communication protocol suitable for the invention is discussed below. It should be noted, however, that other suitable communication protocols may be devised which are suitable for use in connection with the present invention, and that the present invention is not limited by the particular communication protocol set forth below.

[0111] This protocol was originally designed and implemented for the Kali language, as described in H. Cejtin et al., "Higher-Order Distributed Objects," *ACM Transactions on Programming Languages and Systems* Vol. 17, No. 5, pp. 704-739 (1995), and is described in U.S. Patent No. 5,745,703 entitled "Transmission Of Higher-Order Objects Across A Network Of Heterogeneous Machines," issued April 28, 1998. Both of these references are expressly incorporated herein. Much of the Kali implementation can be used to implement the communication protocol for the present invention.

(1) <u>Shared Data Structures</u>

[0112] Most instances exist on only a single subagent. On all other subagents the instance is represented by a remote reference that contains no fields or other data. There are several exceptions to this rule: classes, interned strings, and subagents.

[0113] Every subagent has a local copy of the static data of any class. The values of any non-constant static fields of a class are located on a single subagent.

[0114] All literal strings and string-valued constant expressions have global identity. Each subagent has its own copy of every interned string that it references. Strings contain no mutable data, so no confusion arises.

[0115] The local representation of another subagent must contain the information needed to communicate with that subagent. Unlike classes and interned strings, this data is local; it is not a copy of information found on other subagents. The structure of a subagent is described in the next section.

(2) <u>Subagent Data</u>

[0116] Every subagent instance has a global id. All subagent instances preferably contain the following fields:

- A globally-unique identifier;
- decode: a vector mapping ids to instance; and
- pending: a vector of mapping ids to partially transmitted instances.

[0117] Fields in subagent instances that a particular subagent is communicating with:

- base: the base on which the subagent resides;
- wait queue: a queue of threads waiting to for a connection to be established; and
- in-port, out-port: ports for talking with the subagent.

(3) <u>Communicating Instances</u>

[0118] An instance is preferably transmitted as three ids: that of the class of the instance, that of the subagent that created the instance, and that of the instance itself. If the instance was created by the local subagent and has not been transmitted before, a global id must be created for it and the instance added to the local subagent's decode vector.

[0119] Note that all ids of subagent instances are those of the local instance representing the subagent. A particular subagent may be assigned different ids by every other subagent on which it is known. Preferably, by convention the id of the local subagent itself is zero.

[0120] The receiving subagent uses the three ids as follows: the subagent id is looked up in the decode vector for the transmitting agent, and the instance id is looked up in that subagent's decode vector. The class id is used only if the second lookup fails.

[0121] For example, consider three subagents A, B, and C, and that A has assigned id "3" to B. Further, B has an instance I, to which it assigned id "2", that it has sent to A. When subagent A sends a reference to I to subagent C, it sends the ids "3" (for the subagent) and "2" (for the instance). Subagent C then uses its decode vector for subagent A to translate "2" into its subagent instance for B, and then uses the decode vector in that instance to translate the "3" into the local reference.

[0122] There are three problems that can arise with the receiver: it may not have a local entry for the subagent id; it may not have a local entry for the class id; and it may not have a local entry for the instance id. If it is missing the subagent id or the class id it can send a request back to the transmitter asking for the missing subagent's global identifier or the absolute name of the class. Once the global identifier is received, either the