Although the anchored or pinned objects remaining on a base from which an agent has migrated might not be accessed beyond a firewall as the agent migrates across the firewall, these anchored or pinned objects can reconnect to the agent if the agent returns to the original base.

[0054] FIG. 4C shows the results of agent 40j migrating under complete migration. Complete migration discards all anchored or pinned objects on the original base. Thus, as shown in FIG. 4C, agent 40j has moved from base 30j to base 30k, taking migratable objects 57 with it. Anchored object 58 is discarded from base 30j. A remote reference 59 to the discarded anchored object 58 would accordingly be invalid. Complete migration is useful when an agent resides on a service provider base, and the base does not permit the agent to leave any objects behind when it migrates. Complete migration does not keep network-transparent accesses to all objects, so some care must be taken when using complete migration.

[0055] While agent migration moves an entire agent to the destination base, base-specific agent migration just moves objects in the specified base of the agent. This is useful when an agent is distributed among several bases and a programmer desires that objects in a particular base to move to another base. If these objects move to a base already containing the agent, objects residing on the base and objects moving to the base are merged.

[0056] Object migration is the movement of an object to another base, which may necessitate moving related objects and threads as well. For example, threads associated with a migrating object implicitly move to the base to which the object migrates if they execute (or are currently executing) a method on that object. Threads or other objects may freely migrate among machines. If the object space associated with the agent within which the threads are currently executing spans the target base, migration is a simple matter of copying state into the subagent on the target base. However, if no subagent exists on the target base, a new one must be first created before migration commences.

[0057] Bases are preferably managed by a server. A server is a service provider that has a permanent address, and handles service requests intended for other bases that actually execute the services. For example, as shown in FIG. 5, a server 60 may handle a service request for an agent 40m to migrate to one of several bases 30 managed by server 60.

[0058] As a consequence of the shared memory abstraction provided by the agents of the present invention, tasks and data may freely migrate from one machine to another within an agent. Migration is semantics-preserving: moving threads or objects only has performance implications. In other words, all objects have global identity. The semantics and implementation of the present invention thus provide greater uniformity of expression and greater functionality than the prior art. As noted above, due to the global object space of each distributed agent, knowledge of the physical location or address of an object is not necessary (i.e. need not be manifested in the source program). Therefore, in order for an object to migrate, it is only necessary to appropriately update the object space for that object to reflect the object's destination after migration. This same mechanism makes partial migration (migration of less than an entire agent, such as a subset of the objects or tasks within the agent) feasible and highly effective in the present invention.

[0059] Network transparency implies object mobility. A mobile or movable object in the invention can be selectively moved among subagents (where a subagent is the portion of an agent resident on a particular base) by a programmer. A mobile object which migrates from one machine to another can still be accessed using its global identity managed by the agent's object space. In contrast to distributed shared memory systems, task and data movement among machines can be explicitly and selectively controlled by the programmer, and the transparent access of such objects within an agent is maintained regardless of the movement of the objects among the machines but within the agent. In this sense, the agent model presented in this invention is a significant refinement over a distributed shared memory model.

[0060] Moreover, in sharp contrast with the prior art (where only total migration of an agent was possible, and where no other agent knew the destination of a migrated agent and could no longer communicate with it without static references written into the source program), the present invention allows any agent within the system to access any other agent (or subagent) regardless of migration by merely consulting that agent's object space.

[0061] An exemplary method for implementing the network-centric migration of the present invention among a network comprising a plurality of computer machines is shown in FIG. 6. First, a plurality of object-oriented classes, including an object class, a base class, an agent class and a task class, are defined in a step 80. Next, an object migrate method is defined in the object class in a step 81. When called, the object migrate method migrates a selected object instance to a location specified with the base class (i.e. a base instance). In a step 82, a task migrate method is defined in the task class which, when called, migrates a selected task instance to a location specified with the base class. Similarly, in a step 83, an agent migrate method is defined in the agent class which, when called, migrates a selected agent process to a location specified with the base class.

[0062] After the migrate methods have been defined, an agent process is instantiated according to the agent class in a step 84. The agent process may include or encapsulate task instances instantiated according to the task class and object instances instantiated accord-