Class3 object created on Base2 is established on Base1 (arrow D). Hence, future references to Class3 initiated on Base1 will refer to static fields and methods found in the Class3 class object resident on Base2.

Runtime System

[0080] The runtime system manages a data structure of a base and provides special functions described in this invention by the inventors. As FIG. 9 shows, each base is attached to a corresponding runtime system that provides certain management functions and communication support. A single communication system may be used to serve all of the runtime systems on a particular machine. An agent may comprise a plurality of subagents, each of which resides on separate bases. In this case, subagents in the separate bases are connected with the communication system supported by the runtime system or systems found on their respective bases.

[0081] FIG. 10 depicts subcomponents in a base and its runtime system in detail. A base includes a plurality of data blocks, including class files, object memory, task memory and subagent control storage. The object memory stores all objects in a subagent, including reference objects that refer to remote objects outside the subagent. The object memory is managed by an object manager in a runtime system and pointed to by an object table in the subagent control storage. The task memory stores thread frames, used by the task executer to manage task execution. Class files hold programming code that is accessed by the task executer. The subagent control storage stores management information for a subagent. An agent ID in the subagent control storage identifies the specific agent to which the subagent belongs (that is, the agent of which the subagent is a part). An object table in the subagent control storage points to an object memory in the subagent. A task stack in the subagent control storage points to a task memory to maintain the subagent's execution states.

[0082] An agent manager manages subagents in a base, using subagent control storage and communicating with a task executer that instantiates agents, executes programs in the class files, instantiates objects in the object memory, and manages execution task stacks in the task memory. Since both tasks and objects can migrate freely within an agent and among subagents residing on different bases, some mechanism must be available to transmit object and task state among machines of potentially different types (i.e. heterogeneous machines). Serialization is a process wherein a complex data structure (such as a tree or graph) with internal pointers is transformed into a flat object (such as an array). Pointers in the original are replaced with indices in the flattened representation and reinstantiated as pointers on the receiving agent. An implementation of a serializer is straightforward, requiring only special care to ensure that cycles in the input structure are properly recognized.

[0083] The task executer also communicates with a task serializer, to which the executer makes requests to serialize task objects, and a remote access controller, to which the executer makes requests to call remote methods. Details of one implementation of the remote access controller are described below (in the section entitled "Runtime Data Structure"). An object manager implements the object space discussed above by managing objects in the object memory, and in particular by instantiating objects, reclaiming garbage objects, and making requests for serializing objects to an object serializer. A communication system mediates interaction among bases in machines connected to the network.

[0084] While agent and object migration issues have been generally discussed above (see section entitled "Agent and Object Migration"), the participation of the runtime system in such migration is highlighted in the following additional discussion.

[0085] FIGS. 11A-11E show an example sequence of agent migration. As shown in FIG. 11A, an agent comprising a single Subagent A may reside on a Base A on a Machine A on the left side of the diagram. A Base A task executer is instructed to execute an agent migrate method on the agent comprising Subagent A to migrate Subagent A to Base B on Machine B. The Base A task executer requests a Base A agent manager to obtain agent control data for Subagent A and send it to a Machine A communication system. The agent control data comprises header information about the migrating agent, along with its tasks and objects. Next, the Base A task executer requests a Base A task serializer to serialize task objects within Subagent A in task memory, and the Base A task serializer sends the serialized tasks to the Machine A communication system. Similarly, objects are also serialized and sent to the Machine A communication system by the Base A object manager and object serializer. As shown in FIG. 11B. the Machine A communication system then sends the serialized objects, serialized tasks and agent control data for Subagent A over the network to the communication system for Machine B.

[0086] After the Machine B communication system receives the agent migration data for Subagent A (including the agent control data, serialized tasks and serialized objects), a Base B agent manager allocates a memory block for Subagent A on Base B (denoted Subagent A'), and creates a subagent control storage on Base B for Subagent A'. Machine B task executer and object manager also create task objects and data objects in Base B task memory and object memory, respectively. After Subagent A' is thus instantiated on Base B, a class request is sent from Base B to Base A over the network as shown in FIG. 11C. As shown in FIG. 11D, Base A responds to the class request by sending over the network to Base B class files for the agent which are necessary for resuming the agent on