on multiple processors if the underlying operating system allows processes to execute on multiple processors, as is possible on shared-memory multiprocessors. Additionally, multiple bases may run on the same machine. For example, in FIG. 1, bases 30b and 30c are shown running on the same machine 10b.

[0038] Each agent 40 is a mobile software component that serves as both a global object space and a protection domain. An agent 40 manages a set of objects that all reside in the same consistent and unique object space. Each agent 40 encapsulates a collection of objects, including simple objects (such as data objects) as well as a collection of threads or concurrently executing tasks. Each agent 40 runs on one or more bases 30, and several agents 40 or several parts of agents may simultaneously reside on the same base 30. Thus, agents of the present invention may reside on one or more bases and also on one or more distinct machines. Accordingly, an agent's state may itself be distributed over a collection of distinct machines. The details of components necessary to implement such agents are described below in the section entitled "Runtime Data Structure."

[0039] The agent metaphor allows programmers to write mobile code systems that perform their tasks in autonomous ways. Unlike prior art agents, the agents of the present invention encapsulate both concurrent, distributed tasks and data. In other words, an agent's state may be truly distributed within a network. References to data within an agent do not require knowledge of the physical location where the data resides, so objects within agents may be accessed in a network-transparent manner. As a result, network-centric software is rendered easy to write and maintain. Because references among objects within an agent are location- or network-transparent, the agents of the present invention can be thought of as providing a "shared memory" abstraction in a distributed network environment. Moreover, the agents of the present invention also offer enhanced modularity and protection facilities by providing encapsulation of tasks and data which preferably prohibits transparent access of tasks and data in one agent from other agents as will be discussed in greater detail below.

[0040] Moreover, multiple agents may be created within a network system, and the distribution of agents and even portions of agents (called "subagents") among the machines of the network may be altered dynamically through migration as will be discussed in more detail below. Further, since each agent can contain multiple tasks, a single mobile software component (the agent) can execute tasks simultaneously on different, potentially heterogeneous, machines, thereby enabling greater concurrency within a single mobile software component.

Communication Between Objects

[0041] With regard to communication between objects within the same agent ("intra-agent communication"), the encapsulation provided by each agent and the potentially distributed nature of each agent of the present invention provides important benefits. In particular, objects within a particular agent's object space may be transparently accessed by other objects in the same agent regardless of which base (and therefore regardless of which machine) they reside on. In other words, an object can access any other object in the same agent whether that object is on the same or some other machine in the network and without manifest knowledge of the physical location where that object resides.

[0042] Communication between objects residing in different agents ("interagent communication") is preferably governed as follows. All objects are assigned global identifiers. However, only those objects which implement a special "Remote" interface can be accessed from outside the agent to which they belong. When a objects implementing a Remote interface are passed as arguments to remote procedures, remote references to the objects are supplied. When other objects (i.e. objects not implementing a Remote interface) appear as arguments, copies of these objects are passed. The semantics of such a Remote interface may be similar to the Java/RMI specification, for example.

[0043] The above-described intra-agent and interagent communication arrangements are generally illustrated in FIG. 2. In the figure, two bases 30b and 30c are located on machine 10b, and agent 40a resides on multiple bases 30a and 30b running on distinct machines 10a and 10b, respectively. Each agent 40 includes one or more objects 50. Objects implementing a Remote interface are represented by a rectangle containing cross-hatching and are designated by reference numeral 52, while objects not implementing a Remote interface are represented by a solid black rectangle and are designated by reference numeral 54. Remote references to objects are represented by an unshaded rectangle and are designed by the reference numeral 56. Arrows represent dependencies between references and the object the reference.

[0044] Communication across bases but within an agent is performed by remote reference, as shown by arrows A and B in FIG. 2, in which any kind of object may be accessed and modified consistently. A remote reference can be thus viewed as merely a stub to the actual object it references. This implementation guarantees that access of a remote reference will entail communication with the machine on which the object actually resides, and such communication may serve to access relevant data or initiate a new computation. As noted above, the portions of a single agent found on separate bases are known as subagents. Within a subagent (that is, within an agent on the same base),