Distributed Intelligent Agent

1. Requirements

Build a distributed web server. This web server contains multiple host servers. Each server can holds several agents. Each agent can handle the client requests at specific ports. These agents can be assigned to different groups. The agents within the same group can share their data and state. They may also be able to communicate with each other. Moreover, there is an independent name server, which maintains the entire status of this web server, including the host server, agents, and their states.

[system diagram]

1. Technical Design
   1. Data Model
      1. Server(Entity class for host server)

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Properties** | **Type** | **Description** |
| 1 | IP | String | Host Server IP |
| 2 | Port | Int | Host Server Port |
| 3 | Agents | List<Agent> | Agents held by this server |

* + 1. Agent(Entity class for agent)

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Properties** | **Type** | **Description** |
| 1 | ID | Int | Agent id |
| 2 | Name | String | Agent name |
| 3 | IP | String | Agent IP |
| 4 | Port | Int | Agent Port |
| 5 | Server | Server | Where the agent locates |
| 6 | Group | Group | Belongs to which group |

* + 1. Group(Entity class for group)

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Properties** | **Type** | **Description** |
| 1 | ID | Int | Group id |
| 2 | Name | String | Group name |
| 3 | Color | String | Color |

* 1. Name Server
     1. Start up

Name server shall be started up before other host servers and agents being setup. There should be only one name server for DIA system. The instance of the name server contains three list for host server, agent and group. Group list is generated along with the creation of name server. It contains three items, each has a specific color, which makes easy to distinguish the agents later.

* + 1. Monitor

After the Name server is setup, it’s ready to handle requests. There are two channels that the name server is monitoring at: one is for handling the request from client browser which asking for the latest state of the DIA server; another is for processing the communication from other host servers or agents. For the first one, name server will setup a socket which listens at **port 48060.** When a request comes, it will get the latest information of host servers and agent, then generate html page and send back via TCP. For the second one, name server will start a CommunicationListener, which runs in a separate thread, to handle the request.

* 1. CommunicationListener
     1. Start up

A communication listener is started up when the name server is generated. It has the reference to the host serve list, agent list and group list, which are owned by name server. The communication listener will manipulate these lists when receives UDP messages.

* + 1. Monitor

The communication listener monitors at **port 48050**. It will receive the UDP messages from host servers or agents, handle it, sometime send result back to the sender. Each message contains a header to indicate what request it is, so the listener can handle it accordingly. The requests are listed as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Message Header** | **Description** | **Feedback** |
| 1 | [NewHostServer] | Register new host server |  |
| 2 | [NewAgent] | Register new agent | Send back agent name |
| 3 | [RequireHostServer] | Look for available host server for migration | Send back host server’s ip and port |
| 4 | [AgentMigration] | Update agent’s new ip and port after migration |  |
| 5 | [AgentKilled] | Remove agent from list after being killed |  |

* 1. Host Server
     1. Start up

Host server shall be started up after name server has been setup. Unlike the name server, there can be multiple host servers within one DIA system. Host server can be started up with specified port number, or with **default port 45050**. Each instance of the host server contains an agent list, which means one host server can create multiple agents. Once the host server is created, it should contact name server to register itself.

* + 1. Monitor

Host server monitors at the specified port or 45050 by default. Whenever a request comes, it will create a new agent for it.

* 1. Agent
     1. Start up

The agent is the instance of class AgentListener, which is created by host server. Each agent will find a new available port number when it is created. And it will contact name server for registration.

* + 1. Monitor

Agent monitors at its port to handle requests. It dispatches the requests to AgentWorker.

* 1. AgentWorker
     1. Start up

A new instance of AgentWorker will be created each time when a new request is received by agent.

* + 1. Handle Requests

The requests that a worker handles are listed as follows:

|  |  |  |
| --- | --- | --- |
| **No.** | **Request** | **Description** |
| 1 | Refresh | Update the state (each time increment 1) |
| 2 | Migration | Migrate agent to another host server and port |
| 3 | Kill | Kill agent |

* 1. Log

One requirement is that the execution logs can be saved to log file. This LogHelper class is for this purpose. It contains three static methods, see below.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | Write() | Add log to the list |
| 2 | Save() | Save the list to file, one item one line |
| 3 | Clear() | Clear log list |

* The logs for name server are stored in file **Log\_NameServer.txt**.
* The logs for host server and its agents are stored in file **Log\_HostServer\_#port#.txt**.
  1. UdpHelper

UDP communication is used in name server, host server and agent. It is necessary to create a common helper class to provide the basic udp operation services. Class UdpHelper is invented for this purpose, it contains two following methods.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | sendAndReceiveUDP() | Two way operation, sending and receiving. |
| 2 | sendUDP() | One way operation, only sending. |