GMIS Framework Design and Resource Management Algorithms

The main objective of GMIS framework is to provide resource management services like discovery, monitoring and brokerage of resources in a grid network. This chapter discusses about the GMIS framework design with data models, data flow diagrams and algorithms. Data flow diagrams provide the communications and interfaces between the GMIS components. It also discusses about the design of resource’s agent software.

4.1 Introduction

GMIS provides information regarding the operating status of different resources to the grid users. The main services of GMIS framework are, keep track of hardware and software resources in a grid, poll resources and receive the dynamic state of the grid resources attributes such as load, memory and disk information of all the managed resources, process job requests from grid users and provide resource latest information to users such as current jobs running, current load of resources and status of resources. GMIS also does some degree of fault recovery, such as resubmission of the jobs in case of failures. As discussed in Chapter 3, the components contribute to the working of GMIS are Discovery Server, Data Collection Manager, Topology Manager, Data Handler, Monitor Service, Resource Selector, Viewer and Communicator Server. Interactions among these GMIS components are shown in the figure 4.1.

In this figure 4.1, resources are physical resources in the grid network. GMIS Communicator Server communicates with the resources on behalf of GMIS components. Each physical resource on grid network is represented by one or more objects in GMIS topology. Each physical resource contains a communicator agent which will respond to the GMIS communicator server requests. These agents also send events on behalf of resources based on resource status change or job status change.
Viewer provides user interface to access GMIS services. Agents of corresponding resources respond to the GMIS requests. The following table 4.1 shows the GMIS components interaction with other GMIS components.

<table>
<thead>
<tr>
<th>GMIS Component</th>
<th>Interacts with</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery Server</td>
<td>Topology Manager, Communicator Server</td>
</tr>
<tr>
<td>Data Collection</td>
<td>Topology Manager, Communicator Server</td>
</tr>
<tr>
<td>Manager</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1: GMIS Components Interaction
<table>
<thead>
<tr>
<th>Database</th>
<th>Data Handler (DH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology Manager</td>
<td>Data Collection Manager, Communicator Server, Data Handler</td>
</tr>
<tr>
<td>Monitor Service</td>
<td>Data Handler</td>
</tr>
<tr>
<td>Resource Selector</td>
<td>Topology Manager, Data Handler, Communicator Server</td>
</tr>
<tr>
<td>Communicator Server</td>
<td>Discovery Server, Data Collection Manager, Resource Selector, Communicator Server</td>
</tr>
<tr>
<td>Agent Software</td>
<td>Communicator Agent (CA), Resource Discovery Agent (RDA), Critical Information Agent (CIA)</td>
</tr>
<tr>
<td>Viewer</td>
<td>Discovery Server, Monitoring Service, Resource Selection</td>
</tr>
</tbody>
</table>

### Table 4.1: GMIS Component Interaction with other GMIS Components

The important design considerations discussed in Chapter 1 are used in designing GMIS framework.

#### 4.2 Discovery Server

Resource discovery on grid network is a vital function in grid resource management system. Discovery server provides a simple and convenient way to add managed resources to the GMIS topology.

##### 4.2.1 Dynamic Grid Network Discovery

In GMIS dynamic grid network discovery takes place in three levels. The first level is node level, which uses an Internet protocol address or Fully Qualified Domain Name (FQDN) is used to discover resource information. Second and third levels are cluster level and grid level where a range of Internet protocol addresses are used to discover resources. A user interface is provided on viewer to provide Internet protocol addresses or FQDN. Discovery Server (DS) processes requests from the viewer to discover resources with the given IP addresses or FQDN. DS sends a message to the agent software running at resources using these IP addresses, through the communicator server. Upon receiving the request, agent running at the resource...
collects the resource information including application information and sends information to the DS through communicator server (CS). DS sends a message to the Topology Manager (TM) to update the topology with the new resource information. Topology Manager updates resource topology and database with the resource information. Data flow diagram of this process is shown in figure 4.2.

**Figure 4.2: Data Flow Diagram of Dynamic Network Discovery**

1. To discover resources on grid, user initiates Grid/Cluster/Node level discovery using user interface in viewer.
2. Discovery server sends a message to the communicator server (CS) to discover resources on grid network.
3. CS broadcasts message to all the IP addresses mentioned in user interface.
4. Agent software that runs on the resource(s) responds to the broadcast message sent by the CS with resource information.
5. CS forwards resource information to DS.
6. DS forwards resource information to Topology Manager (TM). Topology Manager updates hierarchical resource management information tree and stores in cache storage.
7. Topology Manager updates the database with resource information.
4.2.2 User based Resource Discovery

In user based resource discovery, resources information is added manually to the GMIS using user interface provided on the viewer. Resource information provided by the user will be sent to Discovery Server (DS). Upon receiving the user information, Discovery Server validates and sends resource information to Topology Manager (TM). TM adds resource information in GMIS topology and database. Data Collection Manager (DCM) polls the resource through communicator server, to verify the resource information provided by the user and to get the latest status of resource. Resource agent software running on the resource responds to the DCM poll request. DCM forwards the latest resource information to TM. TM updates the topology and database with the latest resource information. This process is shown as a data flow diagram in the following figure 4.3.

**Figure 4.3: Data Flow Diagram of User based Resource Discovery**

1. User provides resource information details on user interface provided on the viewer. This information is passed to the Discovery Server (DS).
2. DS validates the data and sends message to Topology Manager (TM). TM adds resource information to the GMIS topology.
3. TM adds the resource information in database.
4. TM sends a request message to Data Collection Manager (DCM) to verify the resource details.
5. DCM sends a request to the Communicator Server (CS) to verify and to get the latest status of the resource.
6. CS sends a request to the resource agent software of that resource.
7. Resource agent software responds to the CS request.
8. CS forwards the resource information to the DCM.
9. DCM updates topology manager with the latest resource information.
10. TM updates the latest resource information in topology as well as Database.

4.2.3 Discovery Server Algorithm

*Algorithm Discovery Server:*

1. Initialize the IPC library and create two communication channels. One channel is for receiving messages from the viewer and communicator server. Other channel for sending messages to communicator server and topology manager.
2. For each received message
3. BEGIN
4. Parse the message.
5. If (message is from viewer)
6. BEGIN
7. If (message type is DYNAMIC_RESOURCE_DISCOVERY) then
8. BEGIN
9. Prepare a message with Message type as RESOURCE_DISCOVERY with resource IP address
10. Send the message to communicator server.
11. END
12. Else
13. END
14. Else
15. if (message type is USER_RESOURCE_DISCOVERY) then
16. BEGIN
17. Validate the user inputs
18. Prepare a message with Message type as ADDRESOURCE
19. Send the message to topology manager
20. END
21. END
22. Else
23. BEGIN
24. If (resource information is valid information)
25. Forward the message to topology manager
26. END
27. Else
28. Ignore the message.
29. END
4.3 Topology Manager

The main objective of topology manager is maintenance of managed resource object data of various resources and maintenance of object hierarchy.

The functionalities of Topology Manager are:

1. Caching of Resource Information at startup for quicker access to other services.
2. Interfaces with DS to get the newly discovered resource information.
3. Updates resource availability (addition/deletion) information based on requests from Data Collection Manager.
4. Correlates event information and poll response from resources through DCM.
5. Updates the resource info in GMIS topology and database.
6. Maintains submitted job information in cache memory.

At the time of initialization GMIS retrieves the resource information from the database, and resource information is populated into the topology by topology manager. It verifies that resource is still available for that GMIS or not through data collection manager. If resource is still available then the resource information will be retained. Otherwise that resource will be deleted from the topology as well as from database. Topology manager updates the resource information in topology and database based on the information retrieved by the Discovery Server or Data Collection Manager.

4.3.1 Topology Manager Algorithm

*Algorithm: Topology Manager*

1. Initialize the IPC library and create two communication channels, one for sending and other for receiving messages from other GMIS components.
2. Initialize MIB in the shared memory.
3. Retrieve grids information from the database using data handler functions.
4. If (there is resource information in Database)
5. BEGIN
6. For each grid in database do
7. BEGIN
8. Create a topology object with object type as GRID
9. Initialize other attributes of GRID object with the retrieved information
10. Get all clusters information from the database for this grid.
11. For each cluster in database do
12. BEGIN
13. Create a topology object with object type as CLUSTER
14. Initialize other attributes of CLUSTER object with the retrieved information.
15. Initialize the parent id with the GRID ID
16. Get all node information for this cluster.
17. For each node in database do
18. BEGIN
19. Create a topology object with object type as NODE
20. Initialize other attributes of NODE object with the retrieved information
21. Initialize the parent id with CLUSTER ID
22. END
23. END
24. END
25. For each node added in the topology
26. BEGIN
27. Send a request to Data Collection Manager to poll the resource information
28. Get the request id and start timer
29. Mark as request retry as 1.
30. END
31. END
32. else /* no resource information in DB */
33. BEGIN
34. Create a default grid topology object.
35. Create a default cluster topology object under the default grid topology object.
36. END
37. If timer expires then
38. BEGIN
39. If request retry is less than 3
40. BEGIN
41. Send another request to Data Collection Manager to poll the resource information
42. Increase the request retry to 1
43. END
44. else
45. BEGIN
46. Get the parent cluster object
47. Remove the node object from topology
48. Delete node information in database
49. Delete all services/applications information under this node
50. Get the list of children of cluster object
51. If (number of children is zero)
52. BEGIN
53. Get the parent grid object
54. Remove the cluster object from topology
55. Delete cluster information in database
56. Get the list of children of grid object
57. If (number of clusters is zero)
BEGIN
Remove the grid object from topology
Delete grid information in database
END
END
END
For each message received do
BEGIN
if (message is from Discovery Server)
BEGIN
Parse the message and retrieve parent id.
If (parent id exists)
BEGIN
Retrieve parent cluster information.
Add the new node under this cluster in topology
Insert the new resource information in database.
END
else
BEGIN
Add the new node under the default cluster in topology.
Insert the new resource information in database
END
END
else if (message from Data Collection Manager)
BEGIN
if (message type is “POLL_RESPONSE”) BEGIN
Update the topology and the database
END
else if (message type is “RESOURCE_UPDATE”) BEGIN
if (resource already exists in topology) BEGIN
Update the topology and the database.
END
else
BEGIN
Retrieve the parent id of resource
if (parent id exists in topology)
BEGIN
Get the cluster topology information using parent id
Create a new object in topology under this cluster.
END
else
Create a new resource object under the default cluster object.
END
END
END
106. END
107. else if(message from resource selector)
108. BEGIN
109. Create a job object in cache memory.
110. END
111. else Ignore the messages
112. END

4.3.2 Data Model for Topology

The data model for topology takes care of grouping and maintaining logical and hierarchical relations of varied information available with different types of resources that will be supported. The objects in each grid are grouped into scheduled poll-able managed object groups and non-pollable object groups. The poll-hierarchy of the managed objects is also modeled to take care of handling poll responses and event handling.

Each resource type is associated with a listId for the list of ManagedObjectGroups. Topology manager reads the data model and for each type of resource, creates resource objects, which will contain a list of ManagedObjectGroups. For those ManagedObjectGroups, which need to be schedule-polled, the flag is set to TRUE in the data model and all such ManagedObjectGroups are registered with the Data Collection Manager to be polled.

The following figure 4.4 shows the data model for topology. The representation is done as class diagram for convenience.

Example usage: To initialize all nodes and their managed objects, the ResourceTable will have a row {Resource_Name = NODE; ManagedObejctGroupListId = 100(some unique id); defaultStatus=UP (so that DCM can start polling without waiting for link checking)}. This row is read and for resource type NODE all instances will be retrieved from Database and as many Node Instance objects are created in the topology with their IP addresses etc.

Next, for the ListId = 100, the ManagedObejctGroupList is read to get all the ManagedObject groups. Note that each resource can have one or more Managed Object
groups. E.g., Node may have two managed object groups: {NodeState – poll every 3 minutes}, {LoadState – poll every 5 minutes}.

![Data Model for Topology](image)

**Figure 4.4: Data Model for Topology**

Using the ManagedObjectGroupIds create ManagedObjectGroups by taking data from ManagedObjectIdTable. For each ManagedObjectGroupId, a list of MO classes is given like NodeState, LoadState. If the number of instances is fixed then the source shall say 'FIXED' else need to look at database to get the number of instances.

Each managed object can be a container object like Node state, Hardware container etc. For all such objects, their child objects are added to the object repository by looking at MO_Contained table. For Node state, the child MO_Group will contain HW_Container. These are created recursively until there are no child objects. Later if the OS_container is added to NODE, just a new entry in the MO_Contained and appropriate group details in ManagedObjectIdTable should be sufficient to create the new objects without change in the code.
4.4 Database

Each GMIS has a Database. This database includes information about the local managed resources. Whenever a resource is added/deleted or modified in grid, then the database gets updated by the topology manager.

Data handler provides an interface between GMIS components and the database. GMIS components can perform query operations on the database and update database tables using data handler. Data handler provides the following data handling functionalities.

- Handle the parameters associated with each resource.
  - Create the instance of resource.
  - Retrieve the instance of resource.
  - Modify the parameters associated with resource.
  - Retrieve the parameters associated with resource.
- Handle the hierarchy of the resources
  - Traversing through the hierarchy
    - Retrieve the parent of a given resource.
    - Retrieve the children of a given resource.

4.4.1 Data Handler Algorithm

Algorithm: DataHandler

1. Initialize IPC library and create two communication channels to send and receive messages.
2. Check whether GMIS database exists or not.
3. If GMIS database does not exist
4. Create GMIS database with the GMIS DB schema
5. For each received message
6. BEGIN
7. Parse the message.
8. Connect to the database.
9. if (connection is successful)
10. BEGIN
11. if (request type is “RESOURCE_ADD”)
12. Insert the resource information in the appropriate table.
13. else if (request type is “RESOURCE_UPDATE”)
14. Update the resource information based on resource id.
else if (request type is "RESOURCE_DELETE")

Delete the resource information based on resource id.

else if (request type is "RESOURCE_RETRIEVE")

Query the resource information and send the information.

END

else

Send DB_FAILURE message to the requestor component.

END

4.5 Data Collection Manager

Data Collection Manager (DCM) collects resource status periodically. DCM performs resource data collection requests based on the contents of topology object. Agent software running on the resources will respond to these requests. After creating a topology object, it schedules a data request for that object. When topology object is deleted, it stops collecting data from the resource.

DCM is responsible for finding faults and failures of managed resources. It polls the status information of the resources, handles events and updates the database through topology manager.

4.5.1 Polling and Parsing

Resource polling is a mechanism to periodically check the status of any resource connected to a GMIS managed grid network with the help of timer. In GMIS polling is done for resource node for every 10 minutes to get the status of the node. After response is received, it parses response and based on received response, topology and database gets updated, so that these changes will get reflected on resource monitoring view of viewer. If received response for a node is not operational, then it polls immediately its descendants i.e., memory, CPU, load and applications.

DCM polls each of the managed resource periodically. DCM gets the resources configuration information from topology manager. It sends resource status requests through Communicator Server to all managed resources to obtain their latest status. Upon receiving the new status information, it sends this information to the viewer through topology manager, data base and monitor service. DCM interacts with topology manager, and communicator server using Inter Process Communication (IPC) messages. This process is shown as a data flow diagram in the following figure 4.5.
Figure 4.5: Data Flow Diagram for Periodic Monitoring of Resources

1. Data Collection Manager (DCM) requests Topology Manager (TM) for schedule poll able resources and their information.
2. TM responds to the DCM request, with the resources information like FQDN, Internet protocol address, object identifier and hierarchy.
3. DCM sends a message to Communicator Server (CS) to poll for current status of resources using information provided by the TM.
4. CS sends a poll request to resource.
5. Resources’ agent software collects resource information and responds to the poll request.
6. CS forwards the poll request response to DCM.
7. DCM parses the response received from the resources’ agent software and sends updated resource information through internal message to TM.
8. TM updates topology and database with the latest resource information.

4.5.2 Event Handling

Event handling provides a mechanism at the GMIS for managing events generated by resource agents which are routed to the GMIS Communicator Server. Typical events are:

- Resource update information (addition/deletion/modification)
- Indication of a resource change of status
- Indication of overload
- Utilization of resource components i.e. threshold values exceeded
- Job execution information

Communicator Server provides centralized reporting of all events generated by resource agents across the grid network. Communicator Server sends events to Data Collection Manager (DCM). DCM correlates various events generated by resources to get the latest status of resource. This status is passed on the viewer through topology manager, database and monitor service.

The following types of events are supported in GMIS. Communicator Agents generate these types of events.

- **Resource Update Event** – This event will be generated whenever a component or service is added/deleted/modified to resource.
- **Resource Status Change Event** – This event indicates that one of resource component's exceeded a threshold value.
- **Equipment Failure Event** – This event will be generated whenever there is a resource failure due to hardware problems.
- **Availability Event** – This event is generated whenever there is a change in resource availability to the grid users.
- **Job Execution Start Event** – This event is generated whenever job execution starts.
- **Job Execution Completed Event** – This event is generated whenever job execution completes.
- **Job Execution Failure Event** – This event is generated by resource when resource is unable to access input files or unable to upload the output files. If it is unable to access the input file for job execution then, the status of job execution set to Job_Input_Download_Failed. If the resource is unable to upload the output file, then Job_Output_Upload_Failed status will be set to the status of job execution.
These events will be generated by the resource agents on behalf of resources. These resource agents are referred as Communicator Agents. For instance, whenever a resource is added, modified or deleted an event will be generated by resource agent software. This event reaches to Data Collection Manager (DCM) through Communicator Server of GMIS. DCM parses the message and updates Topology Manager (TM) to update GMIS topology. Topology manager updates topology and database about the updated information. This process is called as dynamic event based discovery. Dynamic event based discovery process data flow diagram is shown in the following figure 4.6.

![Figure 4.6: Data Flow Diagram of Event Handling in GMIS](image)

1. When a new resource or application is added/deleted/modified a RESOURCE_UPDATE event sent by the agent software which is running on resources to Communicator Server (CS) of GMIS.
2. CS forwards the RESOURCE_UPDATE event to Data Collection Manager (DCM).
3. DCM parses the RESOURCE_UPDATE event and sends a message to Topology Manager (TM) to update topology.
4. TM constructs/updates hierarchical resource management tree and stores in cache storage and updates the database with resource information.
4.5.3 Data Collection Manager Algorithm

**Algorithm: Data Collection Manager:**

1. Initialize IPC library and open two communication channels for sending and receiving messages.
2. Get all instances of pollable objects from the topology manager.
3. For each object do
4. BEGIN
5. Get IP address of resource
6. Prepare the list of attributes to the message parameters list along with object identifiers
7. Set a transaction id to the poll request
8. Prepare a message structure
9. Send message to communicator server
10. Start timer
11. END
12. If timer expires again start polling of resources, follow steps 3 to 10.
13. For each message received do
14. BEGIN
15. Parse the message
16. if (message is from Communicator Server)
17. BEGIN
18. if (message type is “POLL RESPONSE”) BEGIN
19. Delete the transaction id
20. Forward message to topology manager to update topology and database
21. END
22. else if (message type is “EVENT”) BEGIN
23. Parse the event
24. Prepare a message
25. Send message to Topology Manager
26. END
27. END
28. else if (message is from Topology Manager)
29. BEGIN
30. Poll resources using step 3 to 10.
31. END
32. else Ignore the messages
33. END
34. END
35. END

4.5.4 Data Model for Poll Response and Event Handling

Poll-response handling and event handling for Status Change Events (SCE) or resource update events have a common behavior based on the new state of the managed object. Such behavior is captured in the following data model figure 4.7.
In the first part of the data model general event handling is captured, where-in event acknowledgement is covered. If an event required to be acknowledged, then it shall contain an acknowledgement Object Identifier (OID) and value. If it is a status change event the status of the resource Managed Object is updated, else, based on the appropriate action code specific processing is done.

![Event Handle Table](image)

**Figure 4.7: Data Model for Poll Response and Event Handling**

Second part of the data model looks at updating the status of a managed object, which is common for both status change event and poll-response. Whenever an object status changes, Topology Manager updates topology and as well as database.

**Example usage:** The Event Handle Table contains the list of all types and events and the action that needs to be taken. For instance, a Node Availability event has an entry in the table with \{NE_type = NODE, Event_Type = Node_Avail_event, Ack_Required = TRUE, ACK_OID = 1.2.3.4.3.2.2.22.0, ACK_Value = 1; isSCE = FALSE, actionCode = 0\}. Based on this the event handler shall handle the availability event by picking up the Ack_OID and sending the Ack_Value to that Node.

If the event is a status change event, which does not require acknowledgement like a Node-Hardware-Container status change event, it will have appropriate values and the isSCE flag will be TRUE. Based on that the Event Handler looks at the SCE Handle Table and finds that for the MO_Class Node-Hardware_container, and update the object status. The Managed object instance’s updateState() method is invoked by Data Collection Manager.
The updateState() method extracts the new status of resource managed object and constructs an internal message. This message will be sent to the Topology Manager to update the topology and database.

### 4.6 Monitor Service

Monitor Service provides inputs to the viewer to show the status of resources. The main functionalities of the Monitor Service are User Management and to retrieve GMIS managed resource information. When a grid user logged into the GMIS using user interface provided on viewer, authentication request comes to monitor service. Monitor service authenticates user information by checking user details in database through data handler. Monitor Service validates and updates user information in GMIS database for new users. Once the user authentication is done, GMIS managed resource information is provided to the viewer to display the resource information. Monitor Service periodically responds to viewer's resource information request. The basic Monitoring Service process data flow diagram is shown in the following figure 4.8.

![Figure 4.8: Data Flow Diagram for Monitor Service Process](image)

1. User provides user credentials to access GMIS managed resource information. Viewer sends this request to Monitor Service.
2. Monitor Service validates the user information, and if the user is a valid user, then it sends a request to Database through data handler for the resource information.
3. Monitor Service retrieves resource information from the database.
4. Monitor Service updates the Viewer with the available resource information.

4.6.1 Monitor Service Algorithm

Algorithm: Monitor Service

1. Initialize IPC library and create two communication channels, one for receiving and other one for sending messages.
2. For each message received do
3. BEGIN
4. Parse the message
5. If (message is from Viewer)
6. BEGIN
7. if (message type is USER_AUTHENTICATION)
8. BEGIN
9. Validate the user information, by querying user information from database
10. If (validation is successful)
11. Retrieve the resource information
12. Convert all resource information into XML format
13. Send to the viewer
14. else
15. Send error message to viewer.
16. END
17. if (message type is RESOURCE_INFO)
18. BEGIN
19. Retrieve the resource information from Database using Data Handler
20. Convert retrieved information in XML format
21. Send the retrieved resource information to viewer
22. END
23. END
24. else
25. Ignore the message.
26. END

4.7 Brokerage Service

The Brokerage Service performs a number of functions. The first step is the identification and selection of resources that best fits the needs of grid application. The broker will then submit jobs in the application to the chosen machines. The broker handles submission of jobs but not how the job is actually executed on the resource, as that is part of the resource management system that resides on the resource involved.
Resource Selector component of GMIS takes care of the resource broker functionality. Once jobs are being executed Data Collection Manger monitors resources and the progression of the jobs. The following figure 4.9 shows the brokerage service design in GMIS framework.

![Figure 4.9: Resource Selector Service Design](image)

User can submit the jobs from the user interface provided on viewer. All job requests are processed by the Resource Selector. Resource selector selects the appropriate resources for job execution, provides resource information to create activation graphs and provide interface to execute the jobs. Resource selector gathers all resource information from the database. Job execution request go to the resource agent through the communicator server. Data collection manager monitors the job execution status and status will get updated in database through topology manager. In GMIS framework, three types of job execution are provided:
• **Auto Job Execution** – In this type, user is not bothered which resource executes job. User selects a particular job to be executed and based on the resource availability Job execution will be done by the resource selector.

• **Job Execution on user defined resource** – In this type, user selects a resource on which a particular job requires to be executed.

• **Job Execution through activation graphs** – If a job can be divided into multiple tasks and tasks are to be executed on different nodes, then the user can create activation graphs. Generation of activation graphs can be done by using two methods. One is brokerage service itself which will create activation graphs automatically based on user inputs and other is creation of activation graphs by the user manually, by selecting different resources and applications available.

4.7.1 Resource Selection

Resource selection by grid user process data flow diagram is shown in the following figure 4.10.

![Figure 4.10: Data Flow Diagram for Resource Selection Process](image)
1. Grid user requests for resources with specific job requirements like type of processor, memory size, application name by using user interface on viewer. Viewer sends this request to Resource Selector.

2. Resource Selector queries the database based on the user requirements using Data Handler methods.

3. Data Handler methods retrieves resource information from the database and sends to Resource Selector.

4. Resource Selector updates the Viewer with the available resource information.

**4.7.2 Creation of Activation Graph**

In a grid computing environment various services work in collaboration with each other. The output of one service acts as an input for another service. Each service after processing a input file will produce an output file. A graph can be created specifying the order in which various services are to be executed. The condition on execution and the input/output files of various services could also be specified. Such a graph is called as an Activation Graph.

A user interface is provided on viewer to invoke grid services. The following steps are required to be followed to create an activation graph:

- Choose a node on which task need to be executed.
- Choose a service on this node which is used for task execution.
- Specify input/output files
- Specify the dependency of execution for each task.

Activations graphs can be created manually or automatically. In Manual creation, user requires to select resources and has to specify all the required inputs, whereas in auto generation, activation graph will be created automatically by just specifying a minimal inputs like application names, input files and output files.

**4.7.3 Job Execution**

Job execution process data flow diagram is shown in the following figure 4.11.
1. User initiates job execution using user interface on viewer. Viewer sends this request to Resource Selector (RS).
2. RS stores the job information in database.
3. RS retrieves resource information from the database.
4. Resource Selector sends a request to the Topology Manager (TM) with job details.
5. TM sends a request to the communicator server to submit the job on resource.
6. CS sends a request to Agent software to execute job.
7. Agent software on resources executes the job and sends the status of job to the CS.
8. CS forwards the status of job to the Data Collection Manager (DCM).
9. DCM sends update to the TM.
10. TM updates the database with the latest status of the job.

The above mentioned job execution process is a simple job execution process. For efficient utilization of resources, fuzzy based scheduling algorithm [Pramod, 2006] can be used in GMIS job execution process.
4.8 Communicator Server

Communicator Server (CS) provides GMIS the ability to communicate with resources in grid. CS listens to responses and events from the resources, converts them into internal GMIS messages and sends them to the requested GMIS components. Communicator server maintains all the transactions data in a hash memory. The functionalities of communicator server include:

- Provides interface to send a request message to a resource on behalf of GMIS
- Creates and initializes the transaction and session tables
- Creates UDP/IP sessions
- Handles all communicator related messages like request/responses from GMIS components and events originated from the resources.
- Acts as an communicator agent for GMIS

Communicator server gets requests from Discover Server, Data Collection Manager, Topology Manager and Resource Selector. Communicator server communicates with resource agent with the help of Management Information Base (MIB) generated by the Topology Manager. This process is shown in the following figure 4.12.

![Figure 4.12: Communicator Server](image-url)
4.8.1 Algorithm for Communicator Server

Algorithm: Communicator Server

1. Initialize IPC environment and create two communication channels for sending and receiving messages to/from GMIS components.
2. Open two UDP/IP sessions one for listen and other for send messages from/to communicator agents.
3. Initialize the head and tail shared memory pointers to mib tree head and tail respectively.
4. For each message received do
   5. BEGIN
      6. Parse the message
      7. if (messages from GMIS components)
      8. BEGIN
      9. Initialize the transaction data
     10. Interpret the IP address of resource from the header of message
     11. Add IP address to the transaction data
     12. Get the MIB of requested resource
     13. Update the appropriate hash table to accommodate this request
     14. Create an communicator Packet Data Unit (PDU) for the request
     15. Send message to the resource communicator agent using IP address and port 261.
      16. Start timer
      17. END
     18. else if (messages from Resource agents)
     19. BEGIN
     20. Check whether transaction id exists in the message
     21. If (transaction id exists)
     22. BEGIN
     23. Retrieve the transaction information based on transaction id from hash table
     24. Create internal message
     25. Send message to the GMIS components
     26. Delete the transaction details from hash table
      27. END
     28. else
     29. BEGIN
     30. Validate the received Packet Data Unit
     31. Retrieve the Managed object instance (MOI) from PDU
     32. Retrieve the resource type and message type
     33. Construct internal message
     34. Mark the message as event message
     35. Send message to the Data Collection Manager
      36. END
      37. END
     38. else if (message is from remote GMIS)
     39. BEGIN
     40. if (message type is "JOB_EXECUTION")
     41. BEGIN
Initialize the transaction data
Retrieve the IP address and port of resource from the header of message
Add IP address to the transaction data
Get the MIB of requested resource
Update the appropriate hash table to accommodate this request
Create an communicator Packet Data Unit (PDU) for the request
Send message to the resource using IP address and default port 261.

END
END
else (ignore the message)
END
if timer expired
BEGIN
Retrieve the transaction from the hash table
Build transaction failed message
Send transaction failed message to requested GMIS component
END

4.9 Viewer

Grid users can use the viewer to access resource information, to discover resources on grid, for adding resources to grid and to submit jobs. Viewer is a web client which can be opened on any web browsers like Netscape or Mozilla firefox. Viewer uses Hyper Text Transfer Protocol (HTTP) to interact with the GMIS components through web server. Apache tomcat web server is used as a web server in GMIS. All classes related to interface between viewer and GMIS components are hosted by this web server. The following user interfaces are provided on the viewer.

1. User Management
2. Resources Discovery (Discover at node, cluster or grid level, add node, cluster or grid, add service, discover service and resource availability).
3. Job execution (Activation graph creation and execution, Job submission)
4. Resources monitoring at grid, cluster and node level.
5. Job execution monitoring

4.10 Agent Software

Agent software runs at each physical resource to provide resource information. Three types of agents are included as part of agent software.
• **Communicator Agent (CA)** – CA responds to communicator server requests includes job execution requests.

• **Resource Discover Agent (RDA)** – RDA collects resource specific details and generates events whenever there is a change in resource configuration.

• **Critical Information Agent (CIA)** – CIA sends any critical component change of the resource i.e., when CPU exceeds 80% CPU.

All agents use UDP/IP protocol to communicate with the GMIS communicator server as discussed in Chapter 3.

In operational setup these agents should be kept in the startup so that whenever the resource is booted these agents start automatically. In case the resource provider wants to use the system in a dedicated mode or wants to remove the resource from grid, for some reasons, then the resource provider could stop or kill these agents. If the resource provider stops the agents manually then before shut down RDA generates a resource delete event. Upon receiving this event GMIS updates its topology and database with this information. In case of killing agents, for the GMIS periodic request there will not be any response, so it will be treated as unavailable resource to the grid. Whenever resource provider wants to add the resource to grid, the resource owner can start this agent, by providing inputs that specifying which GMIS it needs to be added.

**4.10.1 Algorithm for Communicator Agent**

*Algorithm: Communicator Agent*

1. Initialize IPC environment and create two communication channels for sending and receiving messages to/from RDA and CIA.
2. Open two UDP/IP sessions, one for listening and another for sending from/to GMIS communicator server.
3. For each message received do
4. BEGIN
5. if (message is from GMIS communicator server)
6. BEGIN
7. Parse the Packet Data Unit (PDU)
8. Extract the communicator server IP address and port number
9. if (request type is RESOURCE_DISCOVER)
10. BEGIN
11. Invoke Resource Discover Agent (RDA) to collect both static and dynamic resource information
12. END
else if (request type is PHYSICAL_ RESOURCE_POLL)
BEGIN
    Invoke Resource Discovery Agent (RDA) to collect only dynamic resource information
END
else if (request type is JOB_EXECUTION)
BEGIN
    Create a new job id
    Validate the job request details
    If validation fails return appropriate error.
    Create an entry in a hash table about the job execution with job id as a hash index
    Invoke the corresponding application to execute
    Send job execution information to CIA
    Prepare a PDU with “JOB_EXECUTION_STARTED” with job id.
    Send the PDU to GMIS communicator server
END
else if (request type is JOB_EXECUTION_POLL)
BEGIN
    Retrieve the job id from PDU
    Retrieve the job status from hash table based on job id
    Prepare a PDU with job status
    Send the PDU to GMIS communicator server
END
else if (message is from RDA or CIA)
BEGIN
    if (message type is RESOURCE_DISOCOVER or PHYSICAL_ RESOURCE_POLL or EVENT)
BEGIN
    Pack the information into a PDU
    Send the PDU to GMIS communicator server
END
else if (message type is JOB_EXECUTION)
BEGIN
    Retrieve the job details based on job id
    Update it’s job status in hash table
    Pack the information into a PDU
    Send the PDU to GMIS communicator server
END
END

4.10.2 Algorithm for Resource Discovery Agent

Algorithm: Resource Discovery Agent

1. Initialize IPC library and create two communication channels one for receiving and other for sending messages.
2. For each message received do
3. BEGIN
4. Parse the message.
5. If (message is from Communicator Agent)
6. BEGIN
7. if (message type is “RESOURCE_DISCOVERY”)
8. BEGIN
9. Get static information like operating system, processor family, number of processors using operating system level commands.
10. Get dynamic information like memory, disk space available, and CPU idle time using system commands.
11. Pack both static and dynamic information into the internal message format.
12. END
13. else if (message type is “POLL_REQUEST”)
14. BEGIN
15. Get the component based on the object identifier (OID) of the request
16. Get the information of that particular component.
17. Construct an internal message with the resource/component information.
18. Send this message to communicator agent.
19. END
20. else
21. Ignore the message.
22. END
23. else if (message is from Controller handler)
24. BEGIN
25. if (message type is “START” or “STOP”)
26. BEGIN
27. Prepare an internal message with message type as RESOURCE_UPDATE event.
28. Send this message to communicator agent.
29. END
30. else
31. Ignore the message.
32. END
33. END
34. END

4.10.3 Algorithm for Critical Information Agent

Algorithm: Critical Information Agent

1. Initialize IPC library and create two communication channels to send and receive messages to/from the communicator agent.
2. Get the threshold values for different components from the threshold configuration file. Currently only two components are monitored, one is CPU usage and other is memory usage.
3. Get CPU and memory usage using the system commands.
4. Start timer
5. If CPU or memory usage is exceeding the threshold values then
6. BEGIN
7. Prepare an internal message with event type as CRITICAL_EVENT.
8. Send this message to communicator agent.
9. END
10. If timer expires
11. BEGIN
12. Repeat steps 3 to 10.
13. END
14. For each received message
15. BEGIN
16. Parse the message
17. If (message is from Communicator Agent)
18. BEGIN
19. Store the job information details in internal memory.
20. Start execution of the job.
22. END
23. END
24. If job timer expires
25. BEGIN
26. Get the status of job execution.
27. Prepare a message with the job execution status.
28. Send the message to Communicator Agent.
29. If (job execution status is not “JOB_EXECUTION_COMPLETED) then
30. BEGIN
31. Start job timer again.
32. Repeat steps 24 to 33.
33. END
34. END

4.11 Inter Process Communication

The components or processes of the GMIS communicate with each other through a set of library calls which contain the Inter Process Communication (IPC) library called message-queue IPC library. This IPC library is created to use message queues, which are resident as shared memory in the kernel of operating system. This new library handles communication among the GMIS components.

A socket-based IPC library has developed to communicate between GMIS communicator server and resource agent software. The following figure 4.13 shows the GMIS IPC mechanism.
4.12 GMIS-GMIS Communication

To manage multiple grid network resources and increase resource availability, multiple GMISs can be used across the grid network. In multiple GMIS architecture the advantages are:

- Network traffic and managing resources can be distributed among GMISs.
- When two or more GMISs are connected and need to transfer resource information, then one Communicator Server of GMISs acts as agent and other acts as manager.
- Grid users access resources and act on consistent management information, without regard to location.
- All information is gathered via management requests to GMIS.
- Any GMIS can locate requested managed information and execute requests, thereby distributing the processing load.

To achieve the GMIS – GMIS communication the following pre-conditions should meet:
1. Both running GMIS servers should have trusted host permissions.
2. GMISs users account should exist in both the machines.
3. Ensure that /etc/hosts of the GMIS contain the hostnames and IP addresses of all other GMISs to be connected.

The design of GMIS-GMIS is shown in the following figure 4.14. In this figure, GMIS-1 connected to GMIS-2 and all resource information of GMIS-2 is mounted to the Resource Management Information Tree (RMIT) of GMIS-1. An Application Programming Interface called GMIS-GMIS communicator is provided to connect the multiple GMIS-GMIS. This process is part of Communicator Server of GMIS.

4.13 Summary

This chapter discussed about the design of GMIS components with data flow diagrams and data models. Algorithms for various tasks are discussed in this chapter. This chapter also discussed about the communication methodologies between...
Communicator Server of GMIS and grid resources. Finally it discussed about the GMIS-GMIS communication design.

Next chapter discusses about GMIS processes and interface modeling with experimentation results.