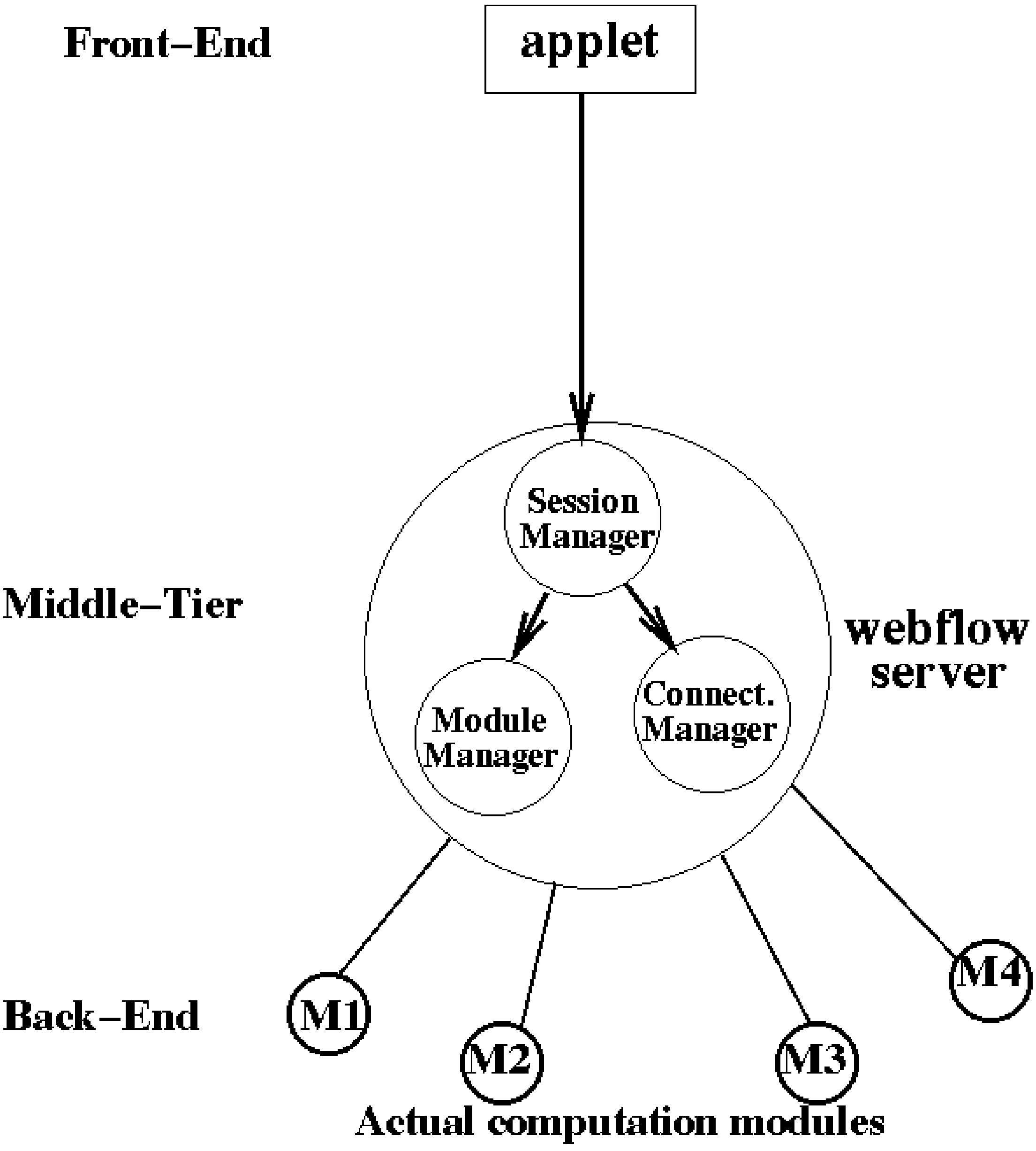
1. COMPONENTS OF WEBFLOW SERVER



Interaction of Client(Applet) and Webflow Server.

Client communicate with Webflow trough Session manager

Figure 1. Components of webflow server

WebFlow is dataflow system which can be accessed through internet and is using flow pattern from the point of view of object-based design. In usual dataflow systems, there are multiple stages each one of which transforms or processes data and send the result to its connected stages. In Webflow, there are modules corresponding to stages. As seen in the (figure 1.), webflow has three-tier architecture .User is building his computation graph from visual components at front-end(inside applet). Each visual component has corresponding actual running module at Back-End. Independent of modules running at back-end, in the middle-tier we have webflow server which consists of three main components:Session manager, module manager and connection manager. User communicates with WebFlow server through Session manager which keeps one distinct session for each user who login into the system. We will describe the webflow architecture and its componenets at the same time in the following chapters.

2. PREPARING THE MODULES

User has to write his own modules by implementing Module interface before doing anything . Module interface is as the following:

```
public interface Module {

  /** initialize the module, put all module information in a
  * MetaModule, and return that information.  */
  public MetaModule initialize();

  /* run the module code  */
  public void run();

  /* stop the execution of the module.   */
  public void stop();

  /* deallocate all resources allocated by the module. Destroy the module  */
  public void destroy();

}
```

In initialize method, user has to create one MetaModule object and initialize its Input and Output ports. But actual connections of ports will be done by Connection manager. This will be explained

in later chapters.Currently , there is no distinction between Input and Output ports in WebFlow.
Initialize method will be called by Webflow during its creation on the behalf of the user. Usually,user will put his computation codes inside run method which is called after he sent Run method to WebFlow server. Stop(Destroy) method is called when he send Stop(Destroy) command to the server.

3. LOGIN INTO THE SYSTEM

User stars his session with sending "start-session" command to Session Manager(SM) which returns to the user SMHost(on which SM runs) and SMport (SM port). It also returns URL file moduleListURL which includes descriptions of user modules and user unique identifier ClientID. This step is done automatically when user downloades the applet inside browser.

1

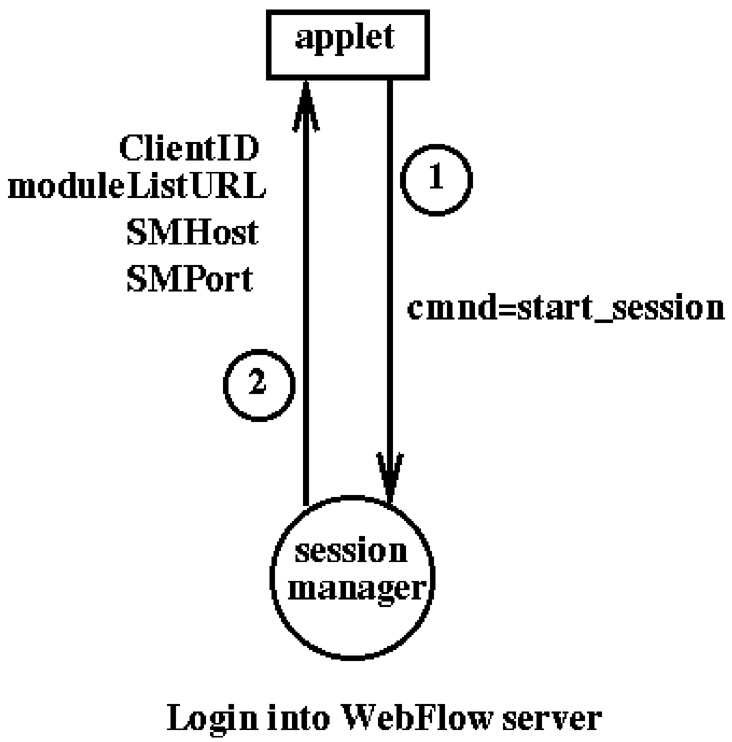


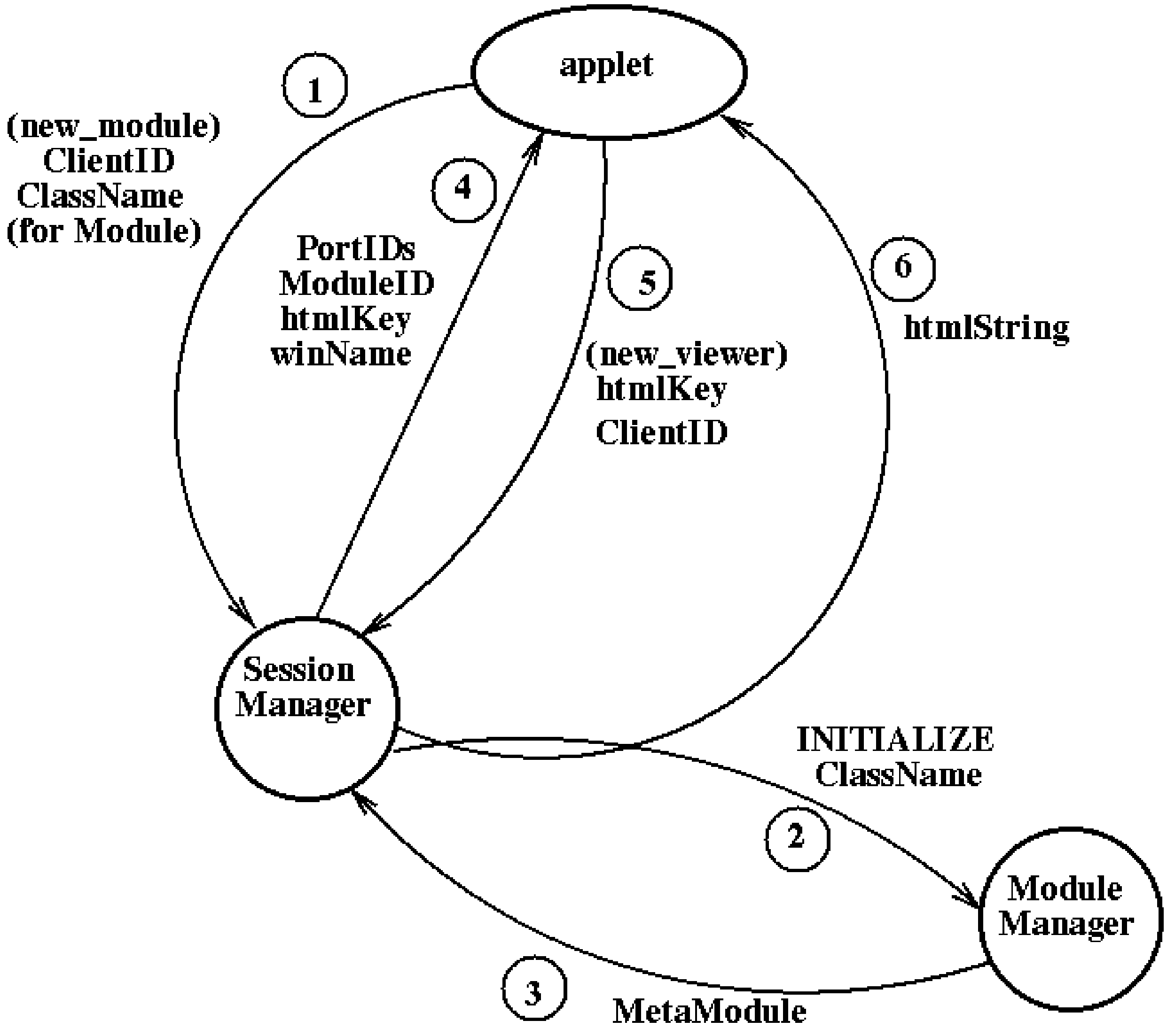
Figure 2. Login into system

4. CREATING NEW MODULE

As we describe about how to create new module, we will also give necessary details of Session manager and module manager . User always initiates the commands for SM by visual authoring tools inside applet at Front-end(click and drag-drop mouse operations) and commands are sent automatically on the behalf of the user.

SM holds WFlowSession list (indexed by UserID) each of which includes ModuleRepresentation table (indexed by ModuleID) and ViewerList-html Strings-(indexed by htmlKey) for user modules and module description list. ModuleRepresentation object for each module keeps its CMPort(its Connection manager Port) ,MMPort(its ModuleManager Port) and Host on which it resides. MM (Module Manager) holds ModuleWrapper table (indexed by ModuleID). ModuleWrapper object wraps user defined module. MM sends the commands to ModuleWrapper(runnin as seperate thread), which forwards them to the Module itself. The advantage is that ModuleWrapper can return control to the MM instantly,especially when running the Module because ModuleWrapper runs the actual module run method as seperate thread.

New-module command with the parameters, his ID,ClientID and ClassName for the module he wants to create is sent to SM after user drops visual icon for the module to Graphic Editor Frame. Then SM finds corresponding MM for this module and sends INITIALIZE command with parameter module ClassName to this MM. Then SM gets MetaModule object for created module and creates ModuleRepresentation object for this module by using information inside MetaObject and inserts this created object into ModuleRepresentation table . Also SM inserts htmlString of this module into ViewerList table. MetaModule object actually includes state (description), ID ,port list and viewerWinName for new module. SM forwards the contained information inside received MetaModule object to applet. If there is matching front-end applet for this module, applet send "new_viewer" command with parameters ,htmlKey and ClientID and receive html string and pop up module applet.



Creating and initializing one module

Figure 3. Creating new module

5. MAKING CONNECTION BETWEEN MODULES

Creating connection between modules are more complex than initializing modules. As seen in (Figure 4.),first, if user wants to make connection from fromModule to toModule, port Ids(fromPortId and toPortID) and Modules IDs (fromModuleID and toModuleId) of module pairs(these port IDs and module IDs are received during initialization of modules) and ClientId are sent with Connect command to SM. Then SM finds CMHost(CM host) and CMPort(CM Port) for module

toModule and send them with port IDs (fromPortID and toPortID) of two modules inside ESTABLISH command to CM
for module fromModule.

CM keeps portList table, indexed by port ID, which includes PortRepresentation for each module port. also CM has two ports :one for requests and other for connections which is drawed in th figure 5. as filled small circle at left CM. In connection process, it got fromPortID (for source module) and toPortID(target module). As you remember from chapter 2 (Preparing Modules), user put module Ports by creating OutputPort and InputPort objects(inside initialize method), where PortRepresentation object is created for each port and registered into CM by calling CM's static method registerPort. The CM job is just to look up PortRepresentation objects for port pairs by means of fromPortID and toPortID and to make connection between these two ports and set storedPort field(type of Port object) of PortRepresentation objects.

As we return to connection process, then CM at fromHost send RECEIVE command with arguments fromPortID , toPortID,CMHost and connections port for fromHost.Then
CM at toHost make socket connection (as client) to connections port for fromHost which is used just only to make connections and set PortRepresentation of toPort to connection and send OK message to CM at fromHost which then set PortRepresentation of fromPort to returned connection from accept call of ServerSocket and send OK message to SM which also send OK to Applet.

This scenario is general case . If two CMs are on the same host, steps 3 and 5 are not needed. Here user doesn't send any command to CM to register its ports but gives its Port objects to CM as method calls that means the host name of CM and MM of one module has to be same .

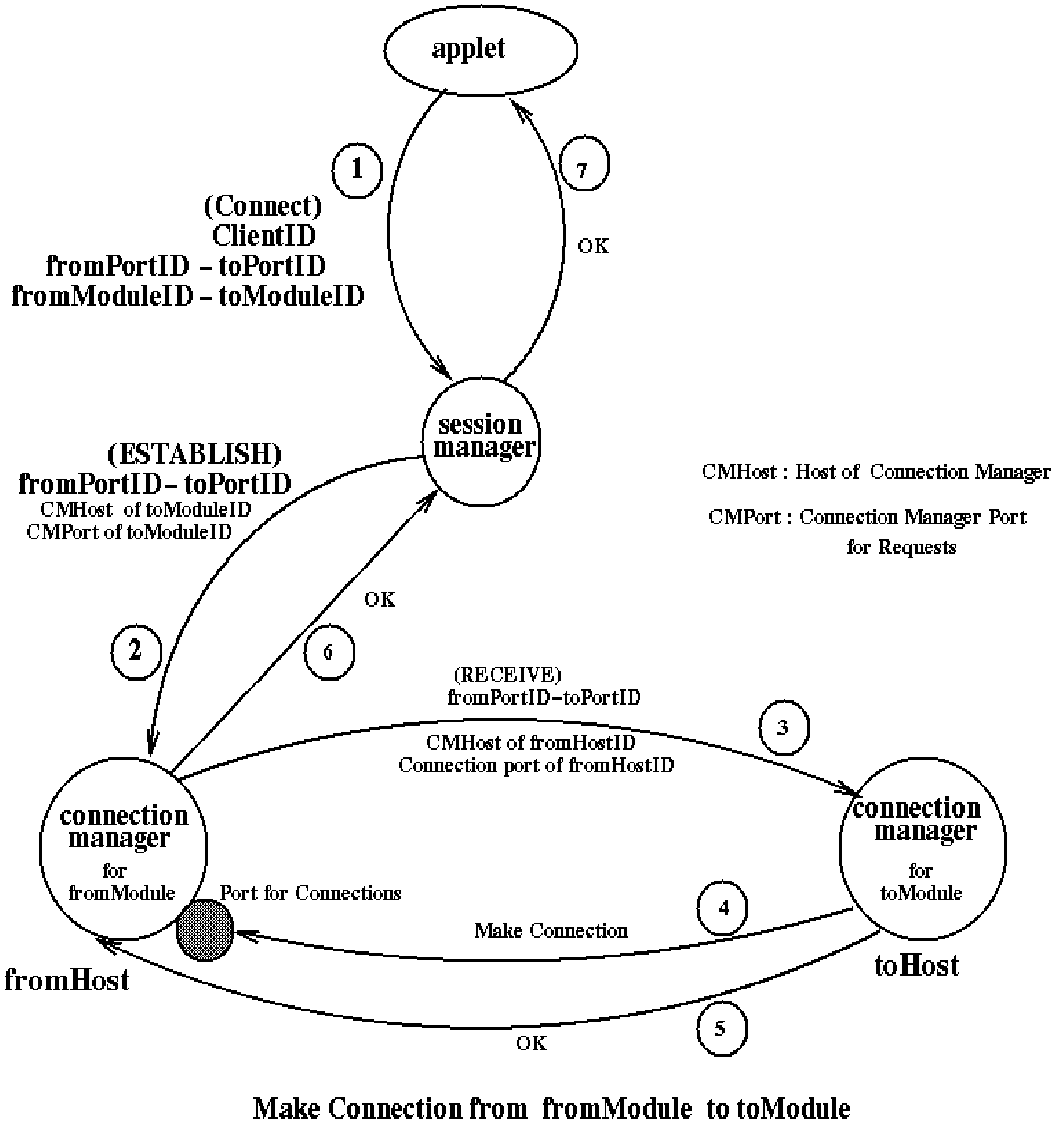
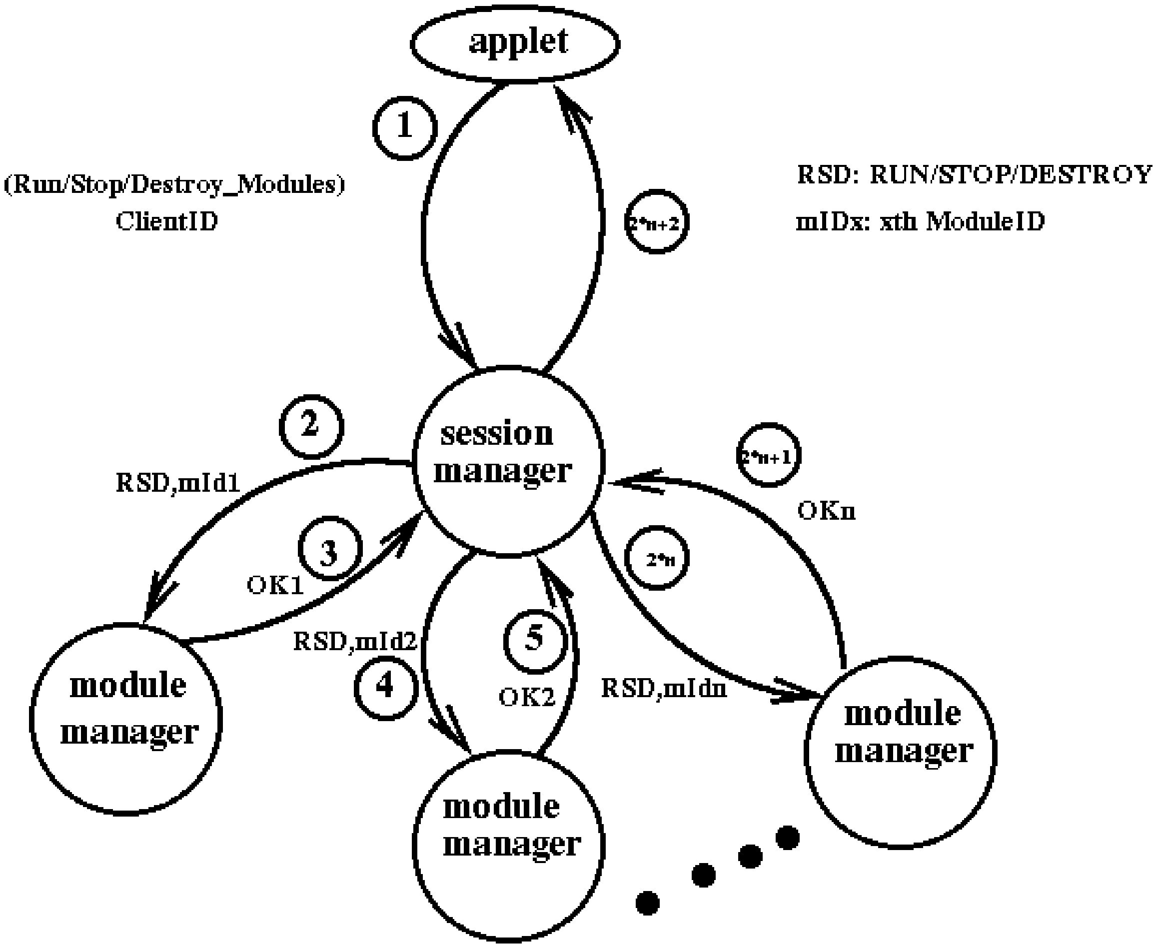


Figure 4. Creatin connection between two modules

6. RUN/STOP/DESTROY OF MODULES

Current WebFlow supports only running modules all together but not separately.After user finished building visual graph, he may send run command to SM with parameter clientID.As seen in the (Figure 5.) ,SM looks up the WFlowSession object .For each ModuleRepresentation inside this object, SM sends to appropriate MM RUN command with ModuleID. Each MM finds ModuleWrapper correponding to received ModuleID and calls its runModules method which then calls actual run method of user module implementation. Stop/Destroy commands work same sa Run command.



Run/Stop/Destroy of Modules. Here user has n modules. Session manager sends the commands to module managers. Some of module managers may be same.

5. Run/Stop/Destroy of modules