TECHNICAL MANUAL

Table of Contents

Technical Manual edited by Grant King, and written by Grant King except where otherwise noted
All non-module-specific diagrams by Jenna Thomson

Introduction

jSummit is a peer-to-peer, multi-platform conferencing suite for Local Area Networks. Written in Java, and using only the Sun Java Media Framework as an external package, jSummit is trivially portable to any platform for which the standard desktop Java Runtime Environment is available. It is designed with flexibility and modularity in mind, and implements, in a modular fashion:

-A global text chat room that encompasses all in the conference ("Summit")
-Private one-to-one text chats
-A shared whiteboard
-A polling/voting system
-File sending to single or multiple participants in the Summit
-Video and/or Audio streaming from a webcam and microphone

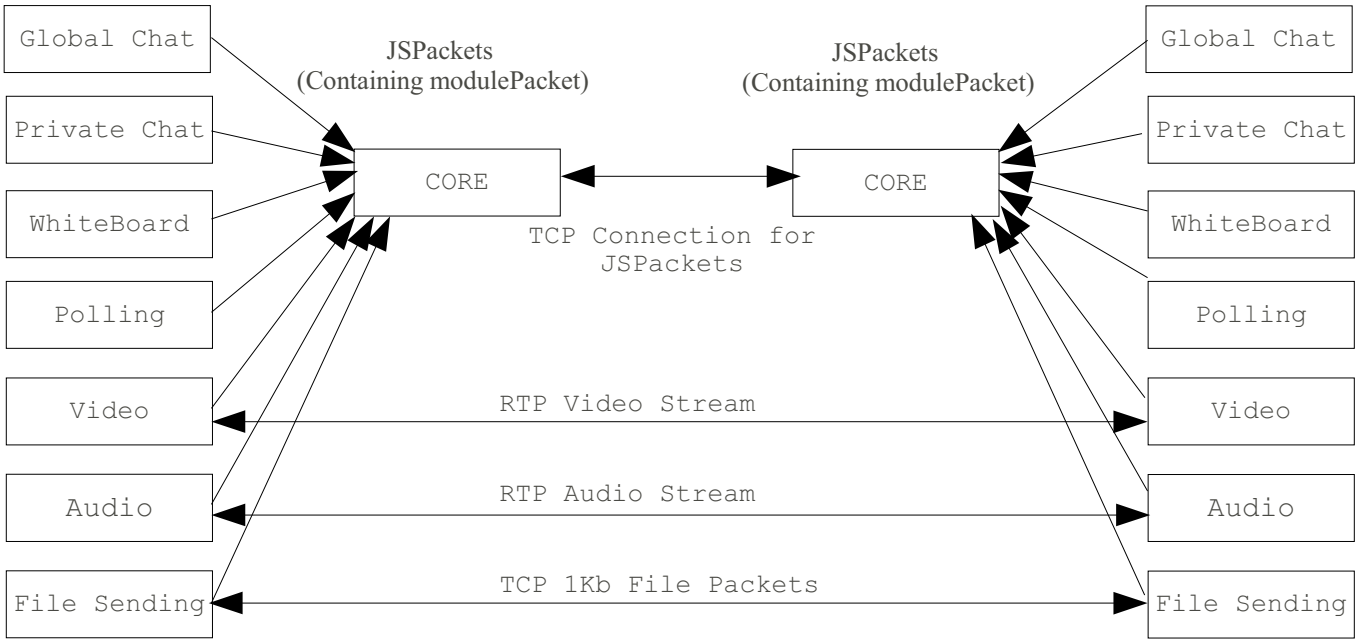
This document is the Technical Manual for jSummit, describing its internal structure, classes, and communications systems.

Overall View of jSummit

jSummit is a "peer-to-peer" conferencing package. Like most communications systems that carry that description, however, the layout is not entirely serverless. The initiator of a Summit becomes the authority and entry point to the Summit, and can hence be seen as the "Server". All participants of the Summit, however, communicate to all other peers via their own peer-to-peer connection. Summit communication, mainly initiated by the Core's Modules that handle each of jSummit's main features, is passed through these connections via a "Core" running on each client, which maintains the connections and manages the modules.

Data is transmitted between Cores in a Summit using Java's ObjectInputStream and ObjectOutputStream classes. These provide a simple interface for sending Serializable classes through a socket.

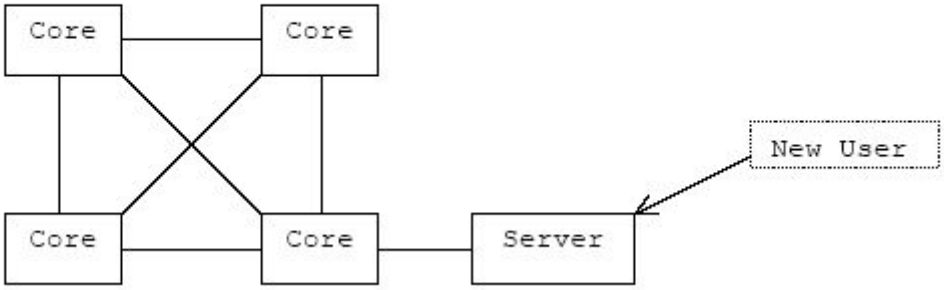
Below is a diagram of the module communications system used in jSummit



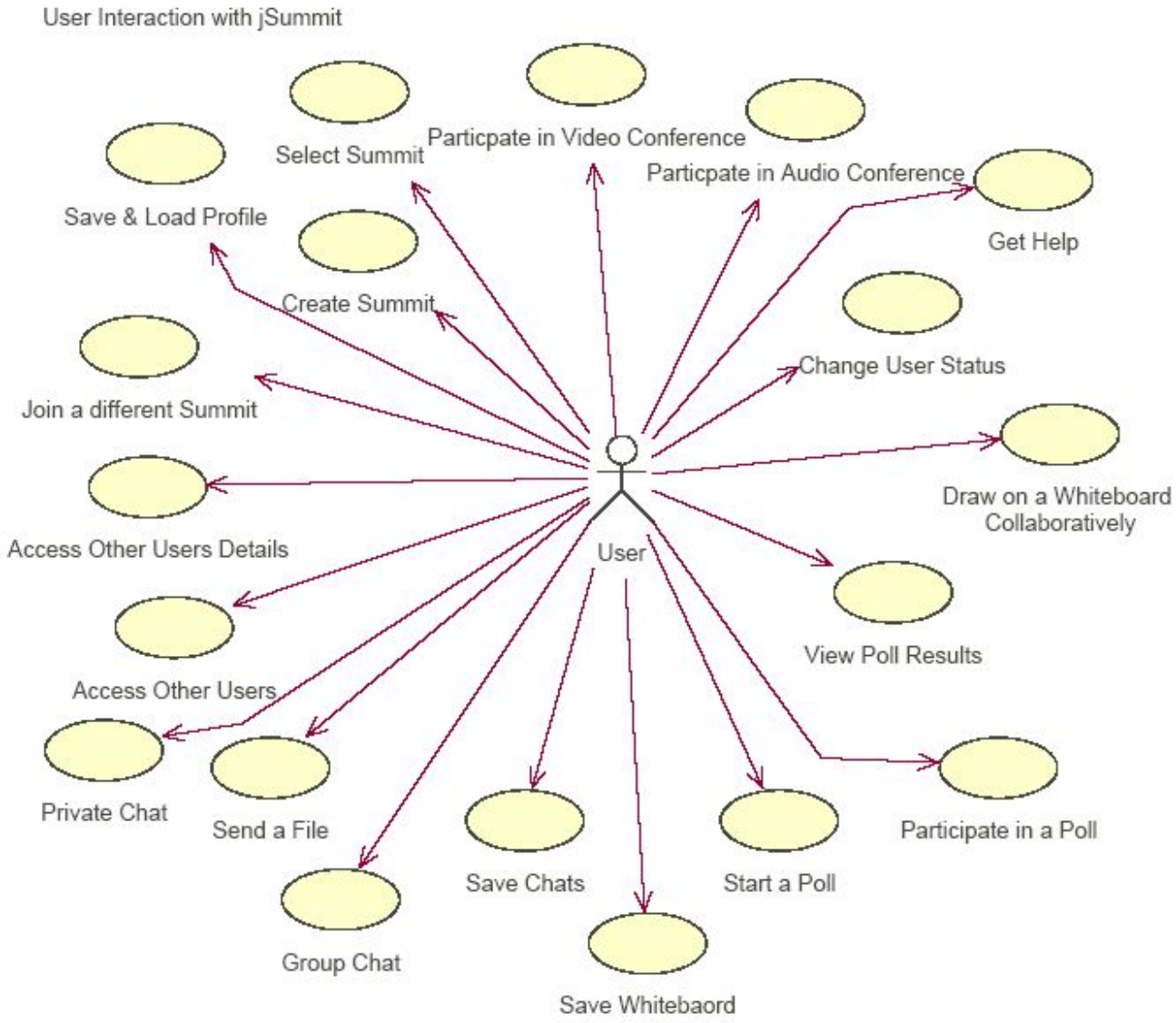
Note: the Video, Audio and File Sending modules use jsPackets to negotiate communications via direct connections.

Below is a representation of the peer-to-peer aspect of jSummit. Note that the Server is running "behind" one of the Cores, and acts only as an entry point and authority to the Summit.

Core Connections



Below is the use case diagram representing a typical user's possible interactions with jSummit



Data Representation

public class JSPacket implements Serializable

Data is transmitted across the network using a standard class with publicly accessible components (similar to a struct), called the "JSPacket":

public String moduleName
A string representing the module sending the packet. This is set to the return value of the class' static getModuleName() method.

public Object modulePacket
An arbitrary class set by the module, used by the module to package the data it is sending. The only restriction on this object is that it implements the Serializable interface.

public Person destination
Represents the recipient of the packet.

public Person author;
Set by the recipient, this represents the sender of the packet, wich may be used by the recipient module.

public class Person implements Serializable

The Person class is a representation of a user profile, and is used in profile selection, as well as to represent the users in a currently active Summit. Below are the data members that are currently used in jSummit:

private String username
The user's chosen nickname. This is unique in a Summit (it is modified by the Server upon joining until it is unique), and userlists are generally keyed against it.

private String realname
The user's real name.

private String location
The user's geographic location.

private String email
The user's email address.

private String company
The user's company or affiliation.

private InetAddress ip
The InetAddress of the user, determined dynamically.

private int status
A representation of the user's status. Definitions of the status values are given in the the Globals class as follows:

```
    public static final int ONLINE = 0;
    public static final int OFFLINE = 1;
    public static final int AWAY = 2;
    public static final int BUSY = 3;
    public static final int IGNORED = 4;
```

If a user is active and in the Summit, their status is user-set to be one of ONLINE, AWAY or BUSY. A value of OFFLINE or IGNORED is never actually set in the Person class, but can be set in the Main Gui's userlist if the Person has been removed from the Summit, or added to an Ignore list, respectively.

public class SummitInfo extends Thread implements Serializable
---
The SummitInfo class is a description of a currently running Summit. It is primarily used to store the Summit's name and description, as well as the list of users currently connected to the Summit.

private String sName
A name given to the Summit by the creator.

private String sDesc
A text description of the Summit.

private InetAddress inetSummit
The InetAddress that the Server of the Summit can be contacted at.

private boolean boPassword
An indication of whether the Summit requires a password to connect to it.

private Hashtable htMembers
A Hashtable containing information on all the users currently connected to the Summit. It contains instances of the Person class, and is keyed by the Person's unique username.

private Vector vToAdd
Reference to an outside Vector to store the description of this summit if it's used in a Marco/Polo discovery (see below)

public void run()
A threaded function to fill in a SummitInfo and add it to a Vector. If the SummitInfo is given inetSummit and a reference to the external Vector vToAdd, the function will query the jSummit Server at inetSummit, receive all information about it, and add the new SummitInfo to vToAdd. See "Service Discovery" for more information.

+------------------+
|Service Discovery|
+------------------+

Marco/Polo

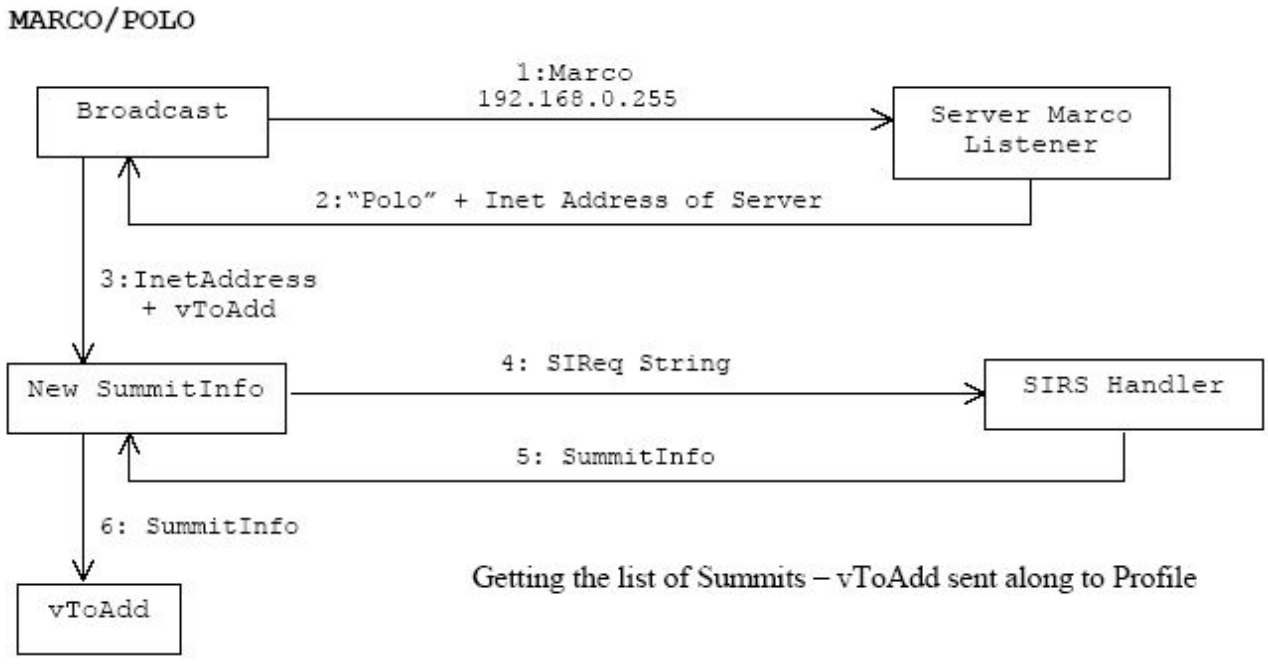
The "Marco/Polo" service discovery system is used to detect Summits on the local area network, in order to gather addresses to query the respective servers at. Upon starting jSummit, a "Marco", or "discovery request" packet is sent to the broadcast address of the network, at a well-known port for this service discovery. The Marco packet consists simply of a sequence of bytes that uniquely identify the version of jSummit running and the type of request. Each server has an instance of the Threaded class "MarcoListener" running, which listens for these requests. When a MarcoListener detects one of these Marco packets, it responds to the original sender with a Polo packet, which is another constant sequence of bytes as in the case of the Marco packet. As the client detects Polo packets being returned from Summits, it uses the addresses of the incoming Polo packets to construct empty SummitInfo classes ready to be started to perform SummitInfo Requests.

SummitInfo Request System

The SummitInfo Request System allows jSummit to retrieve information about all currently running Summits on the network. The client side, that of the newly started jSummit process, creates an empty Vector for SummitInfo classes returned by Servers. It then creates an empty SummitInfo for every Polo response returned to it, fills in these SummitInfos with the address of the Server and a reference to the Vector, and starts each SummitInfo's threaded "run()" function. After a specified timeout, the threads are interrupted and shut down, and the Vector is used in the Summit Selection process.

The threaded SummitInfo class connects to its specified address at a well-known port, and uses Java's ObjectOutputStream to send a String representing the SummitInfo Request to the server. All servers run a SIReqServer class, which listens to this port and delegate incoming connections to threaded SIReqHandler classes. These read and check the incoming request, and respond by using Java's ObjectOutputStream to send the current SummitInfo of the server back to the client. The client's SummitInfo then adds the new information to the vToAdd Vector passed to it.

Below is a diagram of the Marco/Polo sequence and SummitInfo Request System:



+--------------+
|jSummit Server|
+--------------+
If a user decides to create a Summit, an instance of JSServer is created to provide an authority for the Summit (for booting/excluding users and maintaining authoritive lists of users, etc), and to give new members a point of entry to join the Summit.

public JSServer(String sName, String sNewPassword, String sDesc, String newMe)

The constructor creates an empty server with the given Summit Name (sName), Description (sDesc), and a String representing the username of the user creating the Summit (newMe). sNewPassword is the password used to connect to the Summit, if it is null or blank there is no password for the Summit.

The constructor instantiates all classes owned by the JSServer except the JSCore and SummitInfo, and creates a new SummitInfo from the constructor's arguments, as well as the local InetAddress. Once the JSServer is constructed, a JSCore can be constructed with the JSServer's reference. The JSServer is useless until the JSCore has been constructed, and its reference passed to the JSServer using the setCore(JSCore newcore) function.

public void startSummit()

Effectively begins the summit, starting each of the Threaded listener classes.

public boolean addPerson(Person pNew)

Called by a ConnHandler, this function adds a user to the Summit's current SummitInfo and notifies all other participants of the Summit of the new user. Since the username must be unique in the Summit, this function appends "_1" to the incoming user's username until it is unique.

public void boot(String bootUser)

Boots (removes) the user with a Username matching bootUser from the Summit, and notifies all participants to disconnect from the user.

public void addExclude(String sExcluded)

Adds the InetAddress of the user with username sExcluded to the Exclude Vector. Users from this address will NOT be allowed to connect to the Summit, and the MarcoListener will NOT respond to Marco packets from this address

public void remExclude(InetAddress ina)

Removes the InetAddress ina from the Exclude Vector. Users from this address will be allowed to connect to the Summit, and the MarcoListener will respond to Marco packets from this address

+----------------------+
|Server Helper Classes|
+----------------------+

ConnListener

Threaded class, listens to a known port for incoming clients attempting to connect to the summit, and spawns a ConnHandler to handle them.

Class Declaration:
public class ConnListener extends Thread
Constructor:
public ConnListener(JSServer newJss)
   newJss: A reference to the JSServer creating this ConnListener

ConnHandler

Threaded class, spawned and run by a ConnListener. This negotiates with a client attempting to connect to the Summit. If successful, it begins the process of adding the user to the Summit

Class Declaration:
public class ConnHandler extends Thread
Constructor:
public ConnHandler(Socket newClient, JSServer newjss)
   newJss: A reference to the JSServer
   newClient: The Socket retrieved from the ConnListener's ServerSocket

DisconnListener

Threaded class, periodically checks and cleans up connections for disconnected or possibly unauthorised) client sockets.

Class Declaration:
public class DisconnListener extends Thread
Constructor:
public DisconnListener(JSServer newJss)
   newJss: A reference to the JSServer creating this DisconnListener

MarcoListener

Threaded class, listens for and responds to Marco (discovery) requests, and replies with a Polo confirmation if the user is not currently excluded

Class Declaration:
class MarcoListener extends Thread
Constructor:
public MarcoListener(JSServer newJss)
   newJss: A reference to the JSServer creating this MarcoListener

SIReqServer

Threaded class, listens to a known port for Summit Information Requests, spawns SIRSHandlers to deal with them.

Class Declaration:
public class SIReqServer extends Thread
Constructor:
public SIReqServer(JSServer newjss)
   newJss: A reference to the JSServer creating this SIReqServer

SIRSHandler

Threaded class, spawned and run by a SIReqServer, responds to requests for this Summit's SummitInfo, used after the Marco/Polo discovery sequence.

Class Declaration:
public class SIRSHandler extends Thread
Constructor:
public SIRSHandler(Socket theClient, SummitInfo newSi)
   theClient: The Socket retrieved from the SIReqServer's ServerSocket
   newSi: The SummitInfo to respond to the request with, fetched via the SIReqServer's JSServer reference.

ServerPacket

Class used as a "struct" (container for a few values, with each data member public) for passing information to/from Servers and Cores. The JSCore also uses this for Core-Core communication

Class Declaration:
public class ServerPacket implements Serializable

Data members:
public String sCommand
   A String representing the command to perform

public Object oData
   Any object to be used as the operand for the command

+---
|Server Commands|
+---

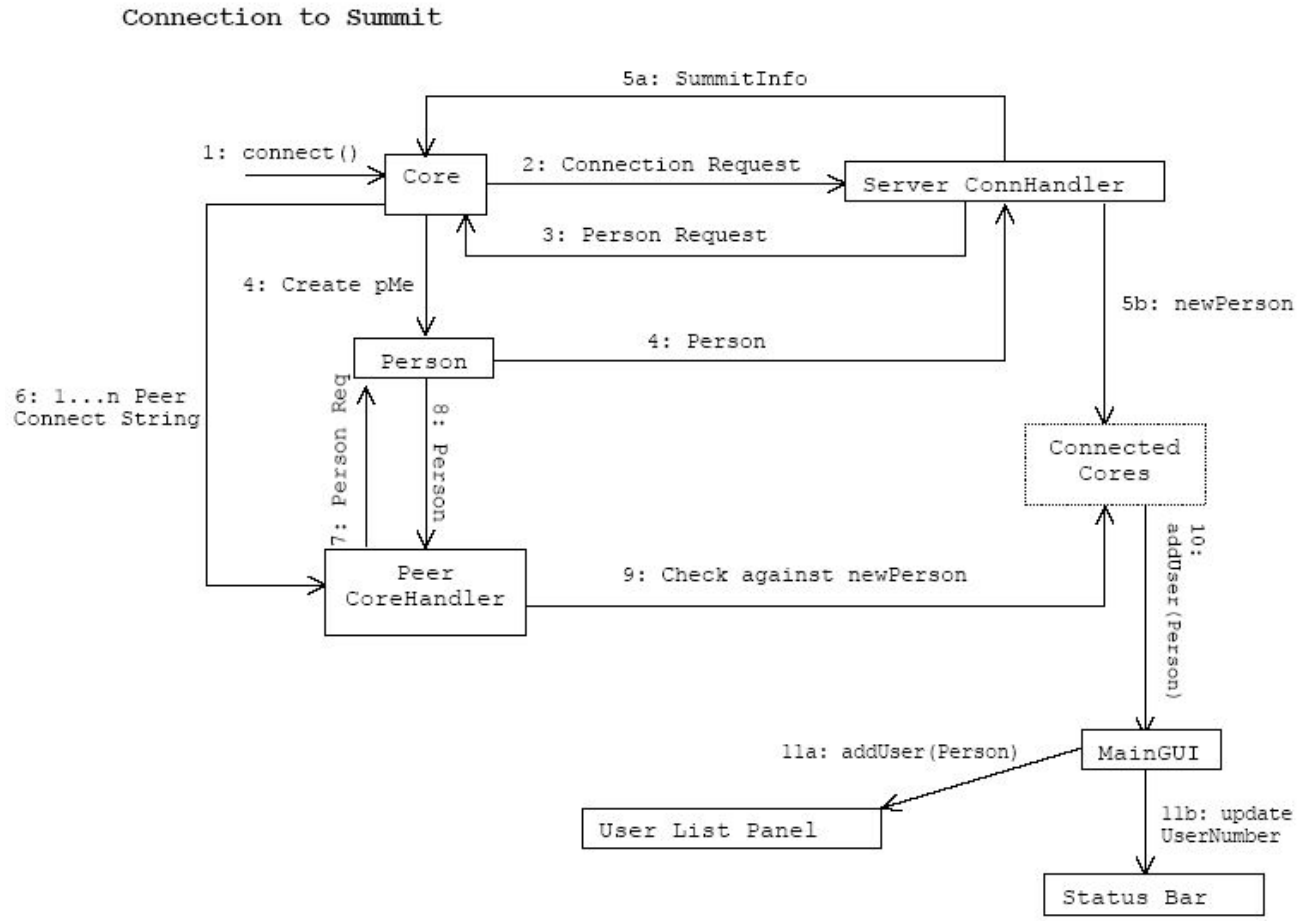
The following commands are sent from JSServer instances using the ServerPacket to package a command and data.

sCommand: "New User"
Sent to JSCores, indicates that a new user should be added to the current SummitInfo
oData: (Person) pNew
The Person to be added to the SummitInfo

sCommand: "Remove User"
Sent to JSCores, indicates that a user should be removed from the SummitInfo, and any connections to the user be closed.
oData: (String) bootUser
The Username of the user to be removed from the Summit.

+------------+
|jSummit Core|
+------------+
The JSCore class serves as the main communications engine in jSummit. Its main functions are to send JSPackets to other peer Cores in the Summit, and to maintain the state of the Summit as consistently as possible with the other Cores. It handles connections to other Cores and the transmission of data through these connections.

Below is a diagram of the connection sequence a Core will initiate after the Marco/Polo and SummitInfo Request System



public JSCore(SummitInfo siNew, Person newMe, ProfileFrame newPf)

Construct a "Peer" core, that connects to an existing Summit. siNew is the SummitInfo retrieved for the Summit, and newMe is the Person selected by the user in the Profile Selection process. The reference to the ProfileFrame (newPf) is so that the JSCore/MainGUI can redisplay the Frame should the user wish to connect to a different Summit.

public JSCore(JSServer newjss, Person newMe, ProfileFrame newPf)

This constructor will create a "Server" core, one that owns and manages a JSServer. The JSServer must be already constructed, and its reference passed in as newjss. Once the JSCore is constructed, its reference may be passed to the JSServer, and the Summit started.

public boolean sendPacket(JSPacket jspToSend)

Called by modules (as well as some Core functions) to send packets to another core specified by jspToSend's "destination" attribute. Returns true if the packet was successfully sent, false if there was a connection error preventing the packet from being sent. If there was an error sending the packet and this Core is runnign a Server, the recipient is booted from the Summit to maintain Summit integrity.

public void broadcastPacket(JSPacket jspToSend)

Similar to the JSModule's implementation of broadcastPacket, this is a convenience function for the Core to broadcast jspToSend to all participants in a Summit. It overwrites the value of jspToSend.destination as necessary.

public void receivePacket(JSPacket jspReceived)

Forwards a received packet to the appropriate module. If the packet is addressed to the Core, this function processes the packet.

public void addUser(Person pNew)

Called by the Server, this function adds a newly connected user to the "Server" Core's SummitInfo

public boolean Connect()

Connects to a Summit, negotiates with the JSServer, connects to peers, initiates modules, and displays the Main GUI. Returns true on successful connection, false if there was an error connecting to the Summit.

public boolean Connect(String sNewPassword)

Connects to a Summit with the specified password. This function simply sets the password to be sent and calls Connect().

public void disconnect()

Disconnects this core from Summit, closes all sockets, and stops all listeners. Displays a warning dialog to the user to inform them they have been disconnected. If this core is already disconnected from a Summit, this function does nothing.

public JSModule getModule(String sModuleName)

Returns a reference to the module that matches sModuleName (as returned by the module's getModuleName function). Returns null if the module does not exist.

public void changeStatus(int newStatus)

This function updates the Main GUI's userlist when the status of a user has changed.

public void exclude(String sExclude)

If this Core is running a server, calls the Server's addExclude function.

public void boot(String bootUser)

If this Core is running a server, calls the Server's boot function.

public void ignore(String sUsername)

Adds a user with the username matching sUsername to the Ignore Vector. Packets will not be received from, or sent to, this user. If the user is already in the Vector, this function does nothing.

public void unIgnore(String sUsername)

Removes a user with the username matching sUsername from the Ignore Vector. Packets will nowbe received from and sent to this user. If the user is not in the Vector, this function does nothing.

Core Helper Classes

ConnectionManager

A ConnectionManager is kept by the Core to maintain peer connections, it stores and handles the peer Sockets, and their associated output streams and ReadServers (input streams)

Class Declaration:
public class ConnectionManager

Data members:
private JSCore jscCore
A reference to the JSCore that created this ConnectionManager.

private Hashtable htOutputStreams
Hashtable of the OutputStreams retrieved from peer sockets, keyed by the Peer's username.

private Hashtable htReadServers
Hashtable of the ReadServers which maintain and listen to the InputStreams retrieved from peer sockets, keyed by the Peer's username.

private Hashtable htSockets
Hashtable of the sockets held for each Peer, keyed by the Peer's username.

Constructor:
public ConnectionManager(JSCore newCore)

Other functions:

public void addUser(Person pTemp, Socket sNew, ObjectInputStream newOis, ObjectOutputStream newOos)
Sets up a new peer (pTemp) in the Summit, and calls the appropriate Core functions to add them to the Summit. sNew, newOis, and newOos are the socket and its associated ObjectStreams that have already been established in the negotiation process when the peer has first contacted this core to connect.

public void remUser(String sTemp)
Removes a disconnected user with username sTemp from the ConnectionManager. Closes any sockets and ReadServers, and removes appropriate entries from the maintained Hashtables.

public boolean sendPacket(JSPacket jspToSend)
Called by the JSCore's sendPacket function, this does the actual writing of the

packet through the appropriate socket. Returns true if the packet was successfully sent, and false if there was an error sending the packet.

public void receivePacket(JSPacket jspReceived)
Called by a ReadServer, this simply passes jspReceived to the JSCore's receivePacket function.

public void disconnect()
Called by a Core that is disconnecting from the Summit, effectively performs a remUser on all other users in the Summit.

PeerListener

Threaded class, listens to a known port for newly connected peers attempting to connect to this Core, and spawns a PeerHandler to handle them.

Class Declaration:
public class PeerListener extends Thread
Constructor:
public PeerListener(JSCore newJsc)
   newJsc: A reference to the JSCore creating this PeerListener

PeerHandler

Threaded class, spawned and run by a PeerListener. This negotiates with a peer attempting to connect to the this Core. If successful, it updates the Main GUI's userlist.
Class Declaration:
public class PeerHandler extends Thread
Constructor:
public PeerHandler(Socket newClient, JSCore newjsc)
   newJsc: A reference to the JSCore
   newClient: The Socket retrieved from the PeerListener's ServerSocket

ReadServer

Threaded class. One ReadServer per peer is created by the ConnectionManager, and listens to the socket established to the peer for an incoming packet (which is then forwarded through the ConnectionManager to the Core.

Class Declaration:
public class ReadServer extends Thread

Constructors:
public ReadServer(ConnectionManager newCm, Person newPClient, ObjectInputStream newOis)
Constructs a ReadServer to listen for packets incoming from another peer (newPClient). Also properly removes a user if the socket closes unexpectedly. newCm is a reference to the ConnectionManager that is creating this ReadServer, and newOis is the socket's ObjectInputStream.

public ReadServer(ConnectionManager newCm, Socket sToServer, ObjectInputStream newOis)
Constructs a ReadServer used only to monitor the connection to a Server (sToServer), to ensure that this Core is properly connected to the Summit. If the socket closes, this will notify the ConnectionManager to cleanly disconnect from the rest of the Summit.
newCm is a reference to the ConnectionManager that is creating this ReadServer, and newOis is the socket's ObjectInputStream.

+---+
|Pre-Summit GUI Systems – By Jenna Thomson|
+---+

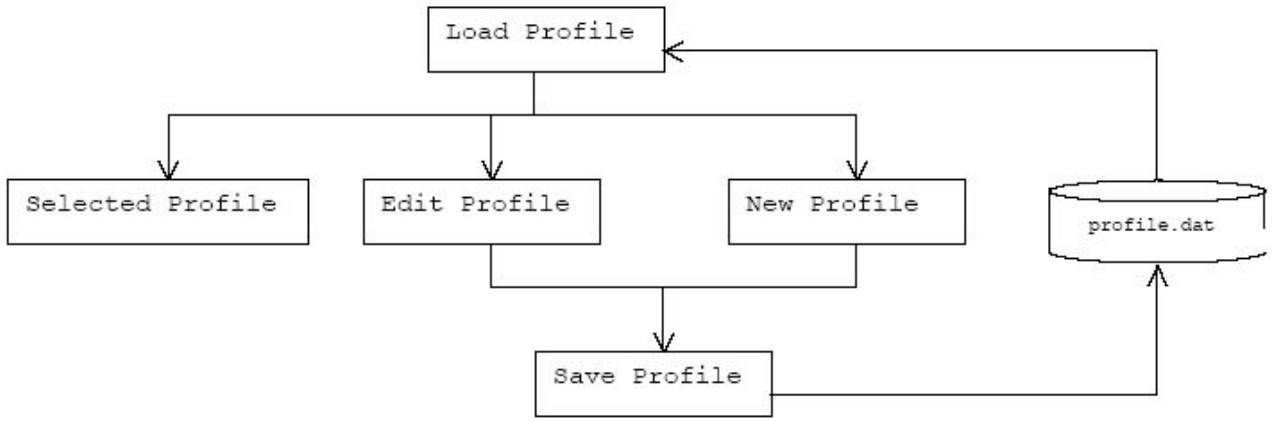
[Diagram of the Classes and how they interact with each other and the CORE]

Profile

This is the JSModule class that takes in the available Summit information from the Broadcast and opens the Profile Frame ready to be used
- the only JSModule classes that it actually uses is the constructor and getSummitName()
It acts as a link between the Broadcast and the Profile Frame - it has no GUI components apart form calling the Profile Frame

Below is a diagram of the Profile selection sequence

Profile Creation/ Selection



Profile Frame

The Profile Frame class is the GUI designed to house the two separate Panels as well as having a Help Button, Back and Next Buttons as well as a Cancel that prompts the user to make sure that they really want to quit. The class consists of GUI components plus the following important members:
//MAJOR MEMBERS (minus GUI components)
    private RoomSelectionPanel paRoomSelection
       - this is the Room Selection Panel class that is called after the user has selected or created a profile then hit the "Next" button
    private ProfilePanel paProfile
    - the profile selection/ creation panel class
    private Person pProfile
       - the profile chosen by the user
    private Profile ProfileReference
       - the JSModule Profile reference
    private SummitInfo siSummitChoosen
       - the SummitInfo class that was chosen by the user - ready to be passed through to the CORE
    private Vector vSummits
       - the Summits available

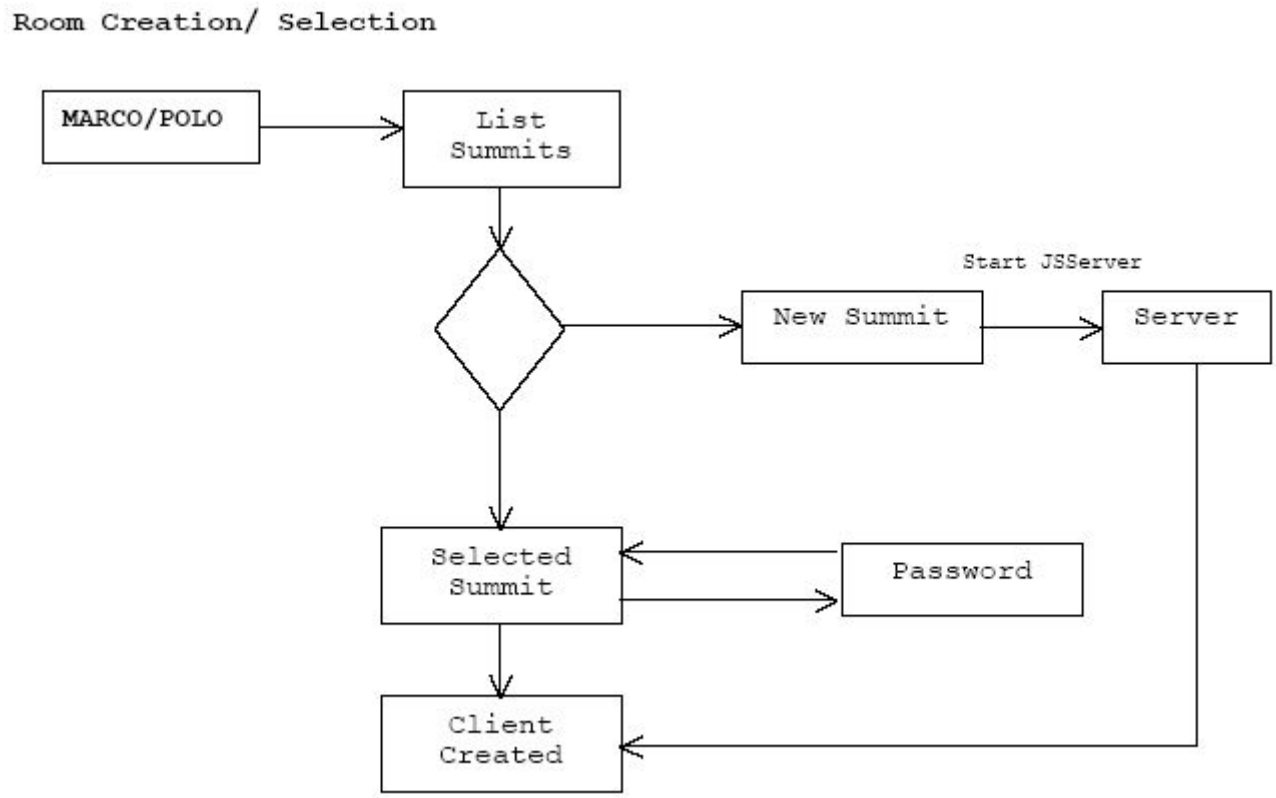
```
//MAJOR FUNCTIONS
    public ProfileFrame( Vector vSummInfo, Profile newPR ) {}
       - the constructor for the initial Frame
    public Person getCurrentProfile() {}
       - gets the current Profile that has been chosen
    public void actionPerformed(ActionEvent e){}
       - for the 4 different buttons - each calling a different action - Back and
Next work differently depending on which panel is currently visible
       - if the Next is to create a summit it calls the current Core's 'connect
()' function to connect to the chosen summit - if it can not connect to a summit
an error dialog is shown
    public JSCore getCore() {}
       - gets the Core associated with the profileFrame
    public void createSummit(JSServer jss) {}
       - called by room creation and starts a summit with a new JSServer and then
connects to this


Profile Panel (Selection And Creation)
-------------

Profile Selection Panel that grabs profile information from a data file
(profile.dat) and adds it to a vector of Person classes - it then iterates
through allowing selection
Profiles can be edited and created by selecting the correct button  and adding
the information into the Textboxes provided in the GUI - if the profile.dat file
is not available it is created when a profile is Saved
Restrictions on the length of UserName, RealName, Company and Location is 24
characters whilst the Email is a maximum of 36 characters
Every time a profile is Saved the entire Profile list (Person objects) are
written to the file - a user can not continue if the profile name is empty or the
<default> profile has been selected
The main members of this class (it implements ActionListener, Serializable for
writing to files and a key Listener) are:
//MAJOR MEMBERS (minus GUI components)
   private Vector profiles - the vector of availble profiles from the profile.dat
file
   private Person currentProfile - the current profile chosen
   private ProfileFrame profileFrame - a reference to the calling Profile Frame
//MAJOR FUNCTIONS
   public ProfilePanel( ProfileFrame pf ){}
      - the constructor that loads the profiles into the combo box ready to be
selected and sets up the GUI
   public void actionPerformed(ActionEvent e){}
      - on the Buttons and the combo box changing
   public void getUserProfiles(){}
      - loads the profiles from the profile.dat file
   public void writeUserProfiles(){}
      - writes all of the current profiles to the profile.dat file
   public Person getCurrentProfile() {}
      - loads the current profile chosen;s details into the textfields
   public void keyTyped(KeyEvent e){}
      - makes sure that the textfeilds don't exceed their limits
```

Room Selection Panel

Below is a diagram of the Room Selection/Creation sequence



The Room Selection Panel gets the available Summits from the Marco-Polo broadcast (as passed through the Profile Module and the ProfileFrame) and displays the information for the user to choose
- the user can choose to select the desired summit from the table or by selecting from the combo box - it wont let the user continue if their are no summits available to choose from they must create a new one in order to continue
A new Summit can be made by hitting "new" (calls the Room Creation Frame)
It's most important members and functions are:

```
//MAJOR MEMBERS (minus GUI components)
   private JFrame fRoomCreation - the Room Creation Frame
   private Vector vSummitInfo - the Summit Info's collected from the Braodcast
   private SummitInfo CurrentSummit - the current summit choosen
   private ProfileFrame pf - reference to the profile frame
//MAJOR FUNCTIONS
   public RoomSelectionPanel( Vector vSummits, ProfileFrame newPf ){}
      - the constructor that sets up the GUI components and their action
listeners
   public void createSummit(JSServer jss) {}
      - called by room creation frame and calls the profile frame create summit
with the new JSServer details
   public ProfileFrame getProfileFrame() {}
      - gets the reference to the profile frame that owns the panel
   public SummitInfo getCurrentSummit() {}
      - returns the current Summit chosen
   public void actionPerformed(ActionEvent e){}
      - calls the RoomCreationFrame constructor
```

Room Creation Frame

Simple little frame that enables the user to create a Summit with the information given - there is a maximum number of characters in each field and a new summit is

created with these parameters by linking back to the Room Creation Panel which links back to the Profile Frame that called it - a new JSServer is created with the given parameters (Summit Name , Password (Optional) and Description (Optional)
The class will not allow the user to continue without first entering something into the name field - the other two fields are optional
//MAJOR MEMBERS (minus GUI components)
   private RoomSelectionPanel rsp - the reference to the Room Creation Panel that called it
//MAJOR FUNCTIONS
   public RoomCreationFrame(RoomSelectionPanel newRsp)
      - constructor to set up the GUI
The Summit Name has a maximum of 24 characters, the password 8, description 64 and all are case sensitive

Password Frame

If the Summit has a password then this frame is called and it gets the password input from the user. It is a simple GUI consisting of a text field and "OK" "Cancel" buttons.
It has a reference to the Profile Frame and uses this to try and connect to that by using: profileFrame.getCore().Connect(sPassword) in the CORE and passing along the password given - if it fails to connect it shows an error dialog telling the user they failed to connect.

+--------------------------------------+
|jSummit GUI Systems – By Jenna Thomson|
+--------------------------------------+

JSFrame

[Diagram of the window list and the module's GUI components using this]

Instead of extending JFrame Modules will extend this child class instead so that the Window list can be implemented easily
The WindowList from the MainGUI calls the bringtoFront and setCentered functions when the JSFrame is clicked
The class instance adds itself to the Window List or removes from when the window is setVisible true or false
//MEMBERS
   protected MainGui mainGuiRef - mainGUI Reference for the windowlist

   public JSFrame(MainGui mgRef){}
      - the constructor setting up the common parameters such as the style and the icon of the frame
   public void setCentered(){}
      - is called when the windowlist item is clicked and puts the GUI frame into the center of the screen
   public void bringToFront(){}
      - used at the same time as the setCentered and brings the frame into the foreground
   public void setVisible(boolean bool){}
      - calls the JFrame setVisible as well as adding the name of the frame to the windowlist in the mainGui Reference
   public void dispose(){}
      - calls the setVisible(false) as well as the super dispose function

+----------------------------+
| Main GUI - By Jenna Thomson|
+----------------------------+

Base Listener

This is an abstract class that all of the action listeners extend off of - it has a mainGui Reference and a way of getting that reference. It doesn't have any abstract classes that need implemententing.
//CLASS
```
abstract class BaseListener implements ActionListener{
	protected MainGui mainGuiRef;

	public BaseListener(MainGui theRef){
		mainGuiRef = theRef;
	}

	public MainGui getMainGui(){
		return mainGuiRef;
	}
}
```

Edit Menu Listener

Child class of the Base Listener that implements the actions of the Edit Menu - Change Profile and Change Room (only if it is a server is this option availble) - the actions don't do anything because these functions were deemed a low priority and will be tackled further in the future
FUTURE
- edit of profile: will call up the profile frame and panel and allow the user to either change their details of the profile that they are using - this new profile will need to be sent down through the core's to inform the other members of the summit of the change (via the user list)
- edit of room: this is only availble to the owner of a summit and it will call up a similiar frame to the create room and enable the owner to change the password and description - the name of the summit will have to remain the same

Environment Menu Listener

Another child class of the Base Listener that responds to the Environmentment Items - the environment menu is not visible in this version of jSummit and will be implemented in the future - please visit: http://jsummit.sourceforge.net for more information on Future Tasks

File Menu Listener

Extends the Base Listener and is the he File Menu Event Handling for all of the File Menu options
It calls core functions to change the status of the current user and if they want to Sign out allows then to save settings where as Exit simply signs out and exits the program.
//ACTIONS
"Change Profile..." - is not implmented but will call up the profile panel in Future releases
"Appear Online" - calls the core's changeStatus function with the Globals.ONLINE static variable by using the mainGui Refernece
"Appear Away" - core's changeStatus with Globals.AWAY
"Appear Busy" - core's changeStatus with Globals.BUSY
"Save Global Chat..." - calls the Global Chat's save function (it routes through the mainGuiRef to the core's reference to the global chat
"Sign Out..." - calls the core's disconnect
"Exit" - calls the core's disconnect followed by a quit and shuts down the program

Diagram of the Change Status sequence:

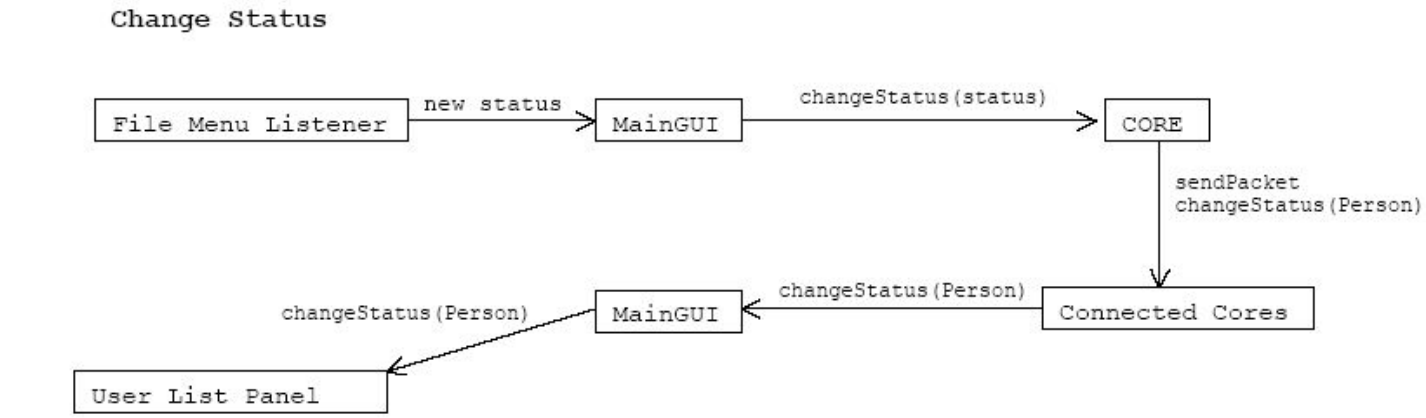
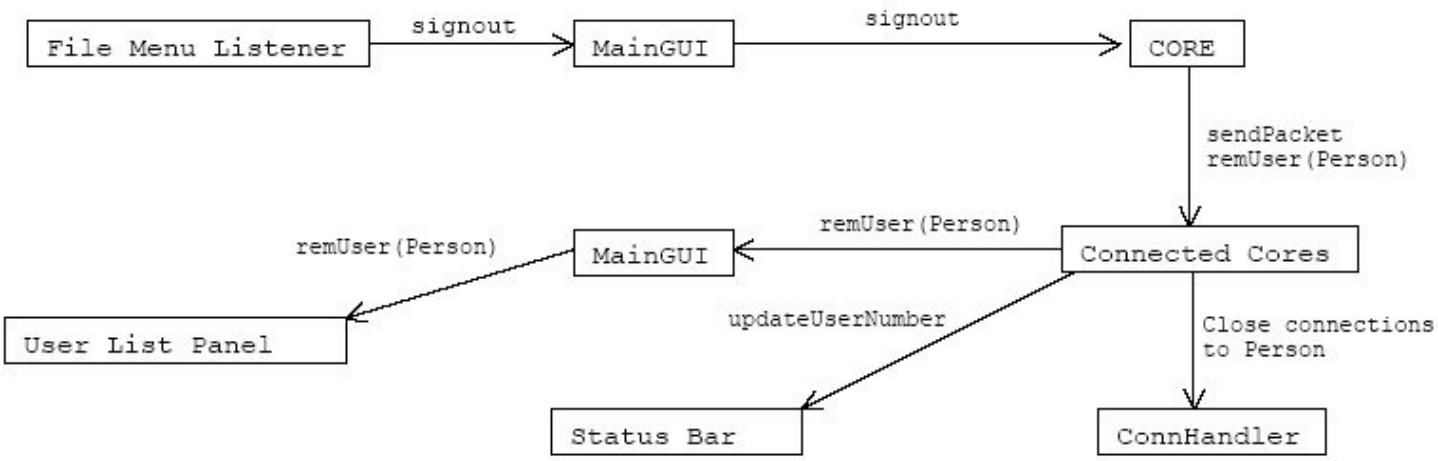


Diagram of the Sign Out sequence:



Help Menu Listener

Child class of Base Listener that calls up the Help Menu Frame and the lovely jSummit about box - it is accessed in the same way as the other Modules in the window list box
The Help Menu has two menu items:
- Main Help - which calls the HelpFrame
- About jSummit - which calls up the about jSummit box

Options Menu Listener

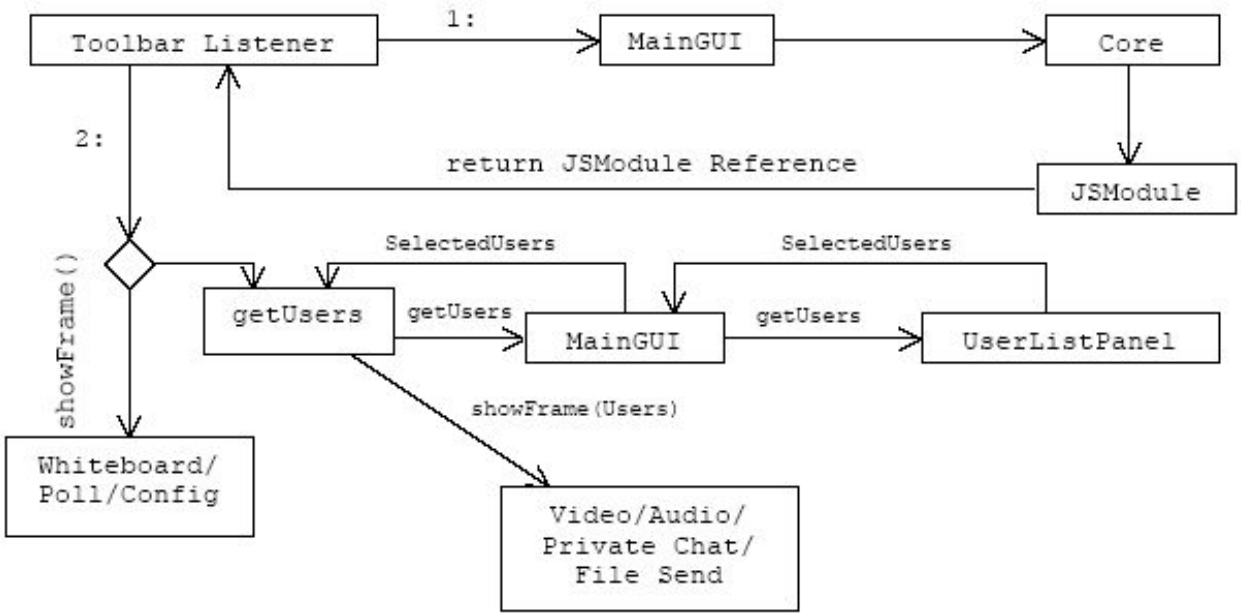
This is not used and extends the base class. It is for future tasks where jSummit works over the Internet rather than just a LAN it enables the user to turn various bandwidth intensive modules on and off such as Video and Audio

Toolbar Listener

Child class of the Base Listener that implements the Toolbar actions. The toolbar listener that does a variety of functions to do with thebuttons and makes the icons change colour as well to indicate that they have been click'd
When the icon has been click the mainGUI calls the showFrame(vPersons) of the Module from its CORE reference - that is all it does - the modules do what they want with multiple calls of the showFrame function - it gets the user's selected fromt eh user list but routing through the mainGUI to call a Userlist.getSelectedUsers() function
An example:
- the video button has be clicked

```
   AudioVideo video = (AudioVideo) mainGuiRef.getCore().getModule
(AudioVideo.getModuleName()); //gets the AV module reference from the core
   video.showVideoFrame( mainGuiRef.getSelectedUsers() ); //calls the AV module's
showFrame function
```

Toolbar Actions



Config Frame

A simple GUI window displaying the various details of the summit such as jSummit Version number, current profile choosen and current summit connected to. If the person is running a server it tells them. It is passed a mainGUI reference and just gets the profile information from the core's reference: myProfile = (mainGuiRef.getCore()).getMe();
It gets the Summit information (one that the user is currently connected to): summInfo = (mainGuiRef.getCore()).getSummitInfo();
And to see if it is a server it calls the isServer() function in the core as well.
The version of jSummit that they are currently using is also displayed (dervived from the Globals variable).
This class is only used for display purposes and has no actual functions.

Global Chat

[see Modules]

Help Frame

Simple GUI that has all of the help information in string arrays. This is the help frame that contains a string list of all of thedetails needed for quick help in the program - Future Developments will have this information read from a file and constructed dynamically but at this point it is easy to do it this way. When one of the topics is selected from the JList the apprpriate array string is selected and displayed in the display area

Main Gui

This is the mainGUI loader that contains all of the mainGUI elements and is what is called by the core in order to start the main component
the program up and running It creates the toolbar, menubars and sets up the panels used by the various frames. Most imporantly it has a core reference which is used by the various classes that have a mainGUI reference in order to access core functions.

MainGUI --> User List
        --> Window List
      --> Global Chat
      --> Status Bar
The mainGUI calls the various GUI components mentioned above and all of them have refernce back to the mainGUI that called them - the mainGUI also has the ability to call the public functions of these classes which are used by various things such as accessing the selectedUsers from the UserList and accesseing the Window List for the JSFrame.setVisible() function.

The Menu consists of a File, Edit and Help menu's all which use the children listener classes for their actions. Whilst the toolbar uses the jSummit icons and has rollover icons and this uses the ToolbarListener class in order to complete the actions performed when the icons are pressed.
//MAJOR MEMBERS
   UserListPanel pUserList;
   StatusBarPanel pStatusBar;
   JSCore jscRef;
   WindowList pWindowList;
//MAJOR FUNCTIONS
   public MainGui(JSCore jsc){}
      - the constructor that sets up the GUI componets
   public void paint(Graphics gc){}
      - repaints the screen to a minimum size - doesn't work on a Mac though
   void createMenuBar(){}
      - sets up the Menu bar
   void createIconToolbar(){}
      - sets up the toolbar icons and rollovers
   void createMainPanel() {}
      - sets up the area to add the other panels onto
   void createStatusBar(){}
      - creates a StatusBarPanel object and loads it to the bottom of the screen
   void createCenterWindow(String ProfileName){}
      - creates the Panel where the Global Chat and the WindowList objects are placed
   public void updateStatusBar(int UserNum){}
      - called by the core when a user either enters or leaves the summit
   public void addUser(Person newPerson){}
      - called by the core when a new user enters the summit - calls the userlist add user function
   public void remUser(Person remPerson){}
      - called by the core when a user leaves the summit - calls the userlist remove user function
   public void changeStatus(Person pPerson){}
      - called by the core when a user changes status and passes this along to th userlist change status function
   public void doQuit(){}
      - ends the jSummit program
   public String getTime(){}
      - gets the current time for diplay in the status bar
   public void addToWindowList(JSFrame frameRef, String frameTitle){}
      - called by the JSFrame and adds the frame to the windowlist by calling the windowlist function
   public void remFromWindowList(JSFrame frameRef, String frameTitle){}
      - opposite of the addToWindowList - it removes it by calling the windowlist remFrame fucntion
   public void ignoreUser(String UserName){}

- called by the userlist when a person needs to be ignored - this is then passed through to the core's ignore user function
public void changeUserStatus(String UserName){}
- called by file manu to change the person's status and passes along to the core's changeStatus

Status Bar Panel

Status Bar Panel that contains the name of the summit, date and time the client connected to it and a current list of users. There is a function that changes the display of the number of users which the mainGUI calls when it is informed that there has been a change in the number of users.

User List Panel

This is the UserList panel that has its own version of the summit info and a JList of all of the users and their status after them - when the list is clicked the display panel shows the information of that particular user
- there are a number of options that can be selected upon right clicking one or more of the users such as Ignore and Un-Ignore
- there are more functions if the summit is owned by that particular person such as Exclude and Boot Users
//MAJOR MEMBERS
private MainGui mainGuiRef - The reference
//MAJOR FUNCTIONS
public UserListPanel(SummitInfo summInfo, MainGui mgr){}
- constructor sets up the mainGUIReference and the SummitInfo used to generate the original list
void ChangeDisplay(String userName){}
- changes the top display to be the current user that has been clicked (or the first if multiple users have been clicked)
public void addUser (Person newPerson){}
- called by the mainGUI to add a user to the list
public void changeStatus(Person pPerson, int Status){}
- called mainGUI to change the status of a user
public void removeUser (Person remPerson){}
- called by the mainGUI when a user has left the summit - their statur is now set to offline - in the future after 5 minutes they would be removed from the summit
void DisplayProfile(String UserName, String RealName, String Company, String Location, String Email, int Status){}
- changes the current displayed profile to the given parameters
public Vector getSelectedUsers(){}
- called by the mainGUI to get those user names of the people currently selected in the user list panel
public static String extractUsername(String nameAndStatus) {}
- gets rid of the statuses after the users names so can search the member list accurately
public void actionPerformed(ActionEvent e){}
- for the right click functions that call the appropriate core function through the use of the mainGUI Reference

Window List Panel

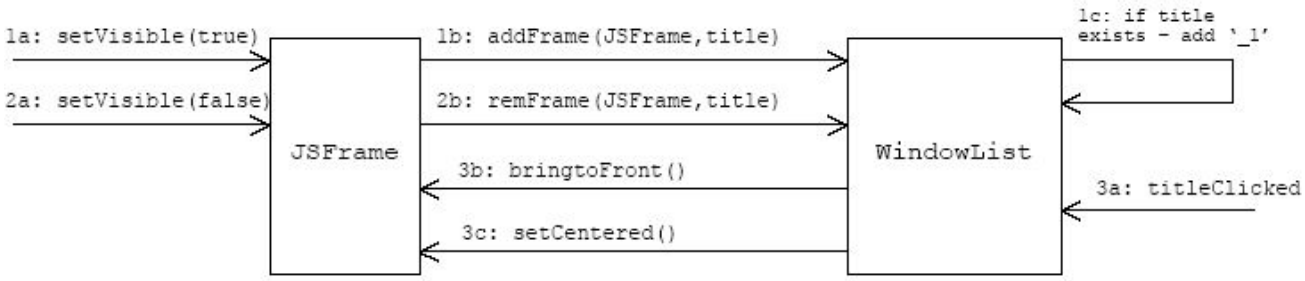
The window list is used by clicking on one of the items - this brings he desired frame to the front and center - the class is used by the mainGUI and accesses the various JSFrame's and the appropriate function in them

```
    jsFrame.bringToFront(); //bring Frame to the front
    jsFrame.setCentered();  //bring the Frame to the center
```

It only has three functions: the constructor - that has the mouse listener on the list

    removeFrame and addFrame that either removes or adds the frame to the list

Window List



+---------------+
|jSummit Modules|
+---------------+
jSummit Modules implement the end-user functionality of jSummit - all primary communications are initiated through and processed by these modules. In order to ensure a consistent interface, and to add some standard functionality in each module, each module extends a base class called the JSModule

JSModule

Class Declaration:
public abstract class JSModule extends Thread

Data members:
protected JSCore jscCore
Reference to the Core that created this module.

Constructor:
public JSModule(JSCore aCore)
Creates a new JSModule with an appropriate Core reference

Other functions:
public static String getModuleName()
In most cases this would need to be overridden (the only case where it needn't be is a Module that does not send or receive packets). As long as the function returns a String unique to this module, the contents are not important, however this generally serves as a descriptive name for the Module.

public abstract void run()
Module's threaded run function. If a module needs to do intensive threaded processing at startup, this function allows it to do so.

public abstract void receivePacket(JSPacket jsPacket)
When the Core receives a JSPacket with a moduleName that matches this module's getModuleName return value, it will pass the JSPacket into this function. In other words, this function deals with received JSPackets.

public void broadcastPacket(JSPacket jspToSend)
This is a convenience function for the Module to broadcast jspToSend to all participants in a Summit. It overwrites the value of jspToSend.destination as necessary.

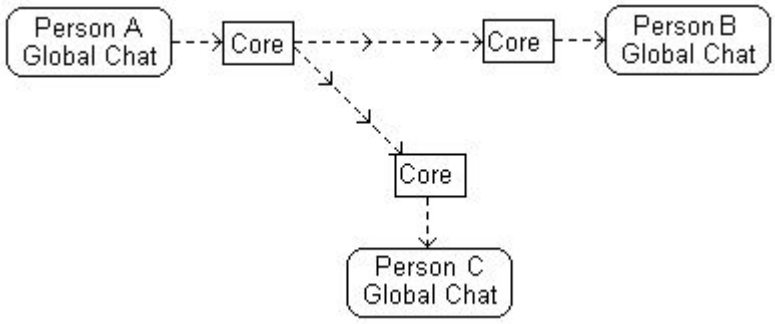
public void vectorCastPacket(JSPacket jspToSend, Vector vPersons)
This is a convenience function for the Module to send jspToSend to each user contained in vPersons. It overwrites the value of jspToSend.destination as necessary.

public void showFrame()
public void showFrame(Vector vSelectedPersons)
This is another function that is generally overridden (the only time it wouldn't be is if a Module does not need a button on the Main Gui). It is the function called by the Main GUI when the user clicks on the Module's button. Generally, the intention is for the Module to show a JSFrame of some sort, and can be used to start Module communication with other users. vSelectedPersons represents the users whose names were selected in the Userlist, so the Module can use this information to communicate with only those users.

+----------------------------+
|Global Chat - By Tim Goodwin|
+----------------------------+

Global Chat Module

The general chat module is a simple object, which receives and sends simple text to and from other users. The general make up of the global chat, is as follows,



The global chat module acts as a simple stub. When you send a message, it sends it's to the core, and by using the cores broadcast wrapper function, it send it to each person in the summit. When receiving the packet, the module takes the packet, and simply displays it to itself.

The chat packet used to send messages, is declared as follows

```
public class ChatPacket implements Serializable {
            public String message;
            public String sender;
}
```

It is a simple, public class, which holds 2 members: a Message, and a Sender. The GlobalChat module receives it, gets a timestamp from the systems current time, and then adds it to the global chat in the following style

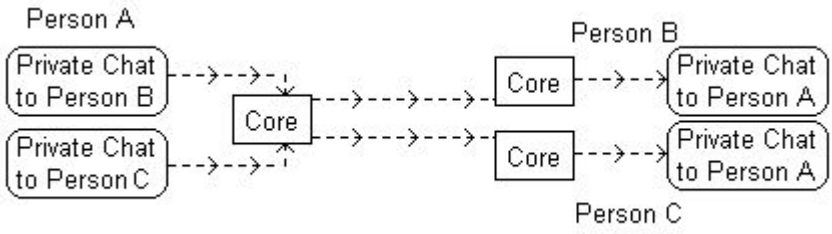
[HH:MM] <Sender> Message

Of course, unlike the private chat, that global chat does not support font faces, sizes, and styles.

+------------------------------+
|Private Chat - By Tim Goodwin|
+------------------------------+

Private Chat

The private chat is slightly more complex than that the global chat. The private chat is similar in protocol, but instead of broadcasting, it is sent to the one person, for example,



In this example,

- Person A is having a private chat to Person B and Person C
- Person B is having a private chat to Person A
- Person C is having a private chat to Person A

As you can see, traffic is individual, and not broadcasted. The traffic that is sent, is a Private Chat Packet which has the following declaration

```
public class PrivateChatPacket implements Serializable {
      public String sender;
      public String message;
      public Font font;
      public Color color;
}
```

The packet is a simple structure, with the following members
sender: The person who the message/packet is from
message: The message/text that being sent
font: The font that the person is using
colour: The colour that the person is using.

The PrivateChat module receives it, gets a timestamp from the systems current time, and then adds it to the global chat in the following style

[HH:MM] <Sender> Message

The message however, unlike the global chat, has the font, which includes a face, a size, and a style, and a colour, which is applied to it.

Because the private chat is traffic is individual, every new private chat opened is a new instance of the private chat class.

Whiteboard

The Whiteboard module implements a shared Whiteboard that is global to the Summit, where freehand and vector-based objects can be drawn.

WhiteBoard

The Whiteboard class is the actual Module, which in general simply organises and manages the different other objects owned by it.

Class Declaration:
public class WhiteBoard extends JSModule

Data Members:
private WhiteBoardFrame wbf;
The WhiteBoardFrame that displays the Whiteboard and BoardControl

The following values represent different draw modes, and are used together as an enumerated type.

```
public static final int DOT           = 0;
public static final int LINE          = 1;
public static final int RECT          = 2;
public static final int ELL           = 3;
public static final int RECTFULL      = 4;
public static final int ELLFULL       = 5;
public static final int IMAGE         = 6;
public static final int FONT          = 7;
public static final int COL           = 8;
public static final int CLEAR         = 9;
```

Constructor:
public WhiteBoard(JSCore newCore)

Functions overridden in JSModule:
public void receivePacket(JSPacket jsPacket)
Passes the jsPacket to the WhiteBoardFrame for processing, unless the packet originated from this core (in which case any objects have already been drawn on the whiteboard)

public void showFrame()
This shows or hides the WhiteBoardFrame

Other Functions:
public void clearWhiteBoard()
This prompts the user with a confirmation dialog, then clears the Whiteboard of all drawings.

WBPacket

The WBPacket is the modulePacket for the Whiteboard, consisting only of publicly accessible data members.

Data members
public String sCommand
The command to perform. Currently valid commands are "Draw Object" to draw a WBObject on the whiteboard, or "Clear Whitevoard" to clear the whiteboard.

public Object oData
The Object (generally a "Draw Object" command's associated WBObject) to go with the command.

WhiteBoardFrame

The WhiteBoardFrame is the window that displays the Whiteboard and its associated control panel.

Class Declaration:
public class WhiteBoardFrame extends JSFrame

Data members:
private WBoard mainBoard
A WBoard panel to display

private BoardControl theBoardControl
A BoardControl to display on the left side of the frame

private WhiteBoard wb
A reference to the WhiteBoard module creating this WhiteBoardFrame

private WBMenuListener wbml
A WBMenuListener for the BoardControl

Constructor:
public WhiteBoardFrame(MainGui newMg, WhiteBoard newWb)
Constructs a WhiteBoardFrame with references to the MainGui and WhiteBoard.

Other Functions:
public void receivePacket(JSPacket jsPacket)
Processes the received packet, forwarding a "Draw Object" command to the WBoard, or clearing the whiteboard

WBoard

The WBoard class is the actual graphical whiteboard, on which the different objects are drawn.

Class Declaration:
class WBoard extends JPanel implements MouseListener, MouseMotionListener

Relevant Data Members
private Graphics fixedg
private BufferedImage fixed
The BufferedImage representing the Whiteboard space, and its associated Graphics object

private WBObject currentObject
The WBObject currently being drawn

private int drawMode
The current draw mode on the whiteboard

private BoardControl bc
A reference to the WhiteboardFrame's BoardControl

private Color cBGColor
private Color cFGColor
The background and foreground colours currently being used on the Whiteboard

private WhiteBoardFrame wbf
A reference to the WhiteBoardFrame

Constructor:
public WBoard(WhiteBoardFrame newWbf)

Other functions:
protected void paintComponent(Graphics g)
Overridden version of JPanel's paintComponent, this redraws the whiteboard and any objects currently being drawn on it.

public void clear(int newBoardWidth, int newBoardHeight)
Clears the whiteboard, and resizes it to newBoardWidth and newBoardHeight (currently fixed at 600x400)

public void mousePressed(MouseEvent e)
Creates a new WBObject depending on what the current draw mode is, and calls the WBObject's mousePressed function

public void mouseReleased(MouseEvent e)
Calls the current WBObject's mouseReleased function, and broadcasts the WBObject if its isSendable() function returns true.

All other MouseListener and MouseMotionListener functions simply pass the mouse events to the current WBObject

WBObject

All drawable objects in the Whiteboard module extend a base WBObject class. This allows a standard "Draw Object" command to be sent in a WBPacket, and the included WBObject will be able to handle its own drawing given the Whiteboard Graphics object. The following base elements are included in the WBObject class

Class Declaration:
abstract class WBObject implements MouseListener, MouseMotionListener, Serializable

Data members:
protected Color colour
The colour that this object should be drawn in

Constructor:
public WBObject()
Simply sets the colour to a suitable default

Other Functions:
public boolean isSendable()
Returns true if the object is currently suitable for transmission across the network. The base WBObject version of this function returns false.

public void draw(Graphics g)
Given the Graphics object g of the Whiteboard, each object should use this function to draw itself onto the canvas.

public void mouseClicked(MouseEvent e) { }
public void mouseEntered(MouseEvent e) { }
public void mouseExited(MouseEvent e) { }
public void mousePressed(MouseEvent e) { }
public void mouseReleased(MouseEvent e) { }
public void mouseDragged(MouseEvent e) { }
public void mouseMoved(MouseEvent e) { }
These are the functions required to implement the MouseListener and MouseMotionListener. Any WBObject may choose to implement any of these.

WBNull

"Blank" WBObject, used when no shape should be drawn on the Whiteboard.

WBDot

This WBObject implements a freely drawn sequence of points ("Dots"), connected together form a free-form line.

Data Members:
private Vector x
private Vector y
Vectors containing the x and y coordinates of the drawn points. x[0] and y[0] refer to the first point drawn.

Implemented functions:

public void draw(Graphics g)
Conforms to WBObject standard, draws this object onto the supplied Graphics.

public boolean isSendable()
Returns true if there are points to draw.

public void mousePressed(MouseEvent e)
Adds the coordinates of the mouse press to the Vectors of points

public void mouseDragged(MouseEvent e)
Adds the coordinates of the mouse drag to the Vectors of points

WBLine

This WBObject implements a straight line joining two points.

Data Members:
private int x1, x2, y1, y2;
The coordinates of the start and end points for this line

Implemented functions:

public void draw(Graphics g)
Conforms to WBObject standard, draws this object onto the supplied Graphics.

public boolean isSendable()
Returns true if the values of x1, x2, y1, y2 are valid

public void mousePressed(MouseEvent e)
Sets the coordinates of the mouse press to be the start point of the line

public void mouseDragged(MouseEvent e)
Sets the coordinates of the mouse drag to be the temporary end point of the line, used to show a live rendering of the currently drawn line

public void mouseReleased(MouseEvent e)
Sets the coordinates of the mouse to be the end point of the line

WBEll

This WBObject implements an Ellipse, both filled and empty

Data Members:
private int x1, x2, y1, y2
The x and y coordinates of the top left and lower right corner of the rectangle that binds this ellipse

private boolean boFull
True if this ellipse is filled with colour. If this is false, only the outline of the ellipse will be drawn

Implemented functions:

public WBEll(boolean newFull)
Constructs a WBEll that is filled if newFull is true.

public void draw(Graphics g)
Conforms to WBObject standard, draws this object onto the supplied Graphics.

public boolean isSendable()
Returns true if the values of x1, x2, y1, y2 are valid

public void mousePressed(MouseEvent e)
Sets the coordinates of the mouse press to be one corner of the ellipse's binding rectangle

public void mouseDragged(MouseEvent e)
Sets the coordinates of the mouse drag to be the opposite corner of the ellipse's binding rectangle, used to show a live rendering of the currently drawn ellipse

public void mouseReleased(MouseEvent e)
Sets the coordinates of the mouse to be the opposite corner of the ellipse's binding rectangle

WBRect

This WBObject implements a rectangle, both filled and empty

Data Members:
private int x1, x2, y1, y2
The x and y coordinates of the top left and lower right corner of the rectangle.

private boolean boFull
True if this rectangle is filled with colour. If this is false, only the outline of the rectangle will be drawn

Implemented functions:

public WBRect(boolean newFull)
Constructs a WBRect that is filled if newFull is true.

public void draw(Graphics g)
Conforms to WBObject standard, draws this object onto the supplied Graphics.

public boolean isSendable()
Returns true if the values of x1, x2, y1, y2 are valid

public void mousePressed(MouseEvent e)
Sets the coordinates of the mouse press to be one corner of the rectangle

public void mouseDragged(MouseEvent e)
Sets the coordinates of the mouse drag to be the opposite corner of the rectangle, used to show a live rendering of the currently drawn rectangle

public void mouseReleased(MouseEvent e)
Sets the coordinates of the mouse to be the opposite corner of the rectangle

WBText

This WBObject implements a written string of text on the whiteboard

Data Members:
private int x, y
The coordinates to draw the text at.

private String sText
The text to draw on the whiteboard.

Implemented functions:

public WBText()
Constructs a WBText, and displays a panel prompting the user for text to enter.

public void draw(Graphics g)
Conforms to WBObject standard, draws this object onto the supplied Graphics.

public boolean isSendable()
Returns true if there is text to draw and proper coordinates

public void mousePressed(MouseEvent e)
Sets the coordinates to draw this text to the position of the mouse

BoardControl
BoardControlListener

The button panel at the side of the WhiteBoardFrame, and its associated listener. These allow the user to choose a draw mode, and they set it for the WBoard appropriately.

+----------------------------+
|Polling – By Phillip Street|
+----------------------------+

Introduction

The Vote module (also known as the 'Polling' module) is an addition to *jSummit* that further enhances the functionality and the comprehensive nature of the application to meet business and conferencing needs. *jSummit* is an application that aims to fulfil and exceed users online meeting requirements and to be more than the average 'instant messaging' application.

The Vote module can be used for many practical applications while in the online meeting environment. Some advantages of using the *jSummit* Vote module include:

- Fair and unbiased voting.
- Anonymous vote responses and results.
- Highly configurable interface to meet any voting demand.
- Instant results once votes are all in to all participants.

Some examples of when voting may be of use include some of the following:

- Voting on executive or meeting decisions.
- Assessing majority responses to several responses.
- Resolving meeting disputes.

The potential of the Vote module within *jSummit* is only limited by the user. It was designed with dynamic flexibility in mind and ease of use. Further on in the manual, explanations of the classes used and diagrams detailing how they interact will be shown then also details on the protocol used and how the module works in concert with the *jSummit* Core will be explained.

Please read on…

Class Details

1. Polling

The Polling class (contained in *Polling.java*) is the main class where the module is instantiated. The class prototypes are as follows:

```
public class Polling extends JSModule{
      Boolean doQuit;
      PollDataPacket createdPollData;
      PollResponseGui aRespGui;
      Vector pollResponses;

      public Polling(JSCore);
      public static String getModuleName();
      public void receivePacket(JSPacket);
      public void shutdown();
      public void showFrame();
      public void cleanup();
      public PollDataPacket getCreatedPoll();
      public JSCore getCoreRef();
}
```

When the module is loaded within the *jSummit* engine it sits and waits until either it receives a packet from some other summit member through the *receivePacket* call (which is called by the Core when it's passing a packet through) or until it has its *showFrame* function called (by the summit user clicking on the Vote icon on the toolbar).

The global variables defined in the prototype above have their functions explained here;

PollDataPacket createdPollData:
This variable is used to hold the information on a poll that has been created by the current user. This variable is either *null* (meaning no vote is currently being created by the user) or it contains a PollDataPacket object detailing the current Vote configuration. Only one vote can be created at a time by the user, so if this variable isn't *null* then a vote is currently in operation.

PollResponseGui aRespGui:
This variable contains a reference to an instantiated PollResponseGui object. This is a handle for certain calls made within the *receivePacket* function.

Vector pollResponses:
A vector of PollResponseGui references used for cleanup and bookkeeping. Keeps a record of current polls open awaiting a response from the user to other votes from remote summit members.

The functions defined within *Polling.java* are described below, further on at the end of this chapter will be diagrams detailing the interrelation between the classes:

public static String getModuleName():
The call returns a basic static string containing the identifier name of the Vote module so that it can be used by the Core to reference the module instance.

public void receivePacket(JSPacket):
This function call within *Polling.java* is how the Core of *jSummit* passes off data packets to the running Vote module. The protocol implemented by the Vote module is handled within this function. The details of the protocol will follow in the Communications chapter.

public void shutdown():
This function hasn't really been implemented but was put in for future use when full thread implementation of the Vote module is implemented. Currently is a place holder.

public void showFrame():
This function call is where the beginnings of a Vote occur. When the Core receives a request (via a mouse click on the Vote icon in the toolbar) to create a Vote, the Core gets the module instance of the running Vote module (using the *getModuleName* call) and then runs this function. The function prepares variables for vote creation and if a vote isn't already in creation it will get a PollCreateGui instance running.

public void cleanup():
This call is used to cleanup a finished vote instance. When this call is made it will allow the user then to prepare another vote to be sent out over the summit.

public PollDataPacket getCreatedPoll():
A simple accessor function to return the current instance of a running vote (or null if there isn't a vote in operation from this user).

public JSCore getCoreRef():
Another simple function call to return a reference to the modules *jSummit* Core reference. This allows for callback through this module so that classes with a reference to this module object instance can easily also get a Core reference.

2. PollDataPacket

The *PollDataPacket.java* file contains the source implementation for the PollDataPacket class. This is the packet class which extends JSPacket for easy transport through the Core to other summit members running the Vote module. The class is similar to a *JavaBean* but doesn't strictly adhere to the standards.

The PollDataPacket contains the following data:

1. the packet creator (determined by 'username')
2. a packet type (types defined in the class)
3. a vote type (also defined in the class)
4. question string
5. poll options available (Vector reference of Strings)
6. voters
7. voters responded
8. number of votes taken
9. current vote count
10. selected options
11. a created date
12. an end date

Most of the functions defined within the class are simple accessor and setting functions to interoperate with the encapsulated data. Also (for stack processing reasons) the PollDataPacket implements the Cloneable interface to allow for easy creation of packet duplicates.

3. PollCreateGui

PollCreateGui is an implementation for the creation gui that the summit client will use to create an active Vote within the running summit. The class is mostly comprised of gui elements with most of the processing done with the *actionPerformed* function call. Error checking is performed when a user is ready to proceed with the vote by clicking on 'Ok', if the components are valid the vote is created and then sent off through the core to the other summit members.

4. PollResponseGui

The PollResponseGui class is similar to the PollCreateGui in many regards. Once again processing is performed within the *actionPerformed* call and error checking is again taken into consideration when the users are preparing to respond to a vote, if the vote is a valid response it will be sent back to the vote's creator through the senders Core.

5. PollResultsGui

The PollResultsGui class is a simple gui just to display the final results from a finished vote. The results are sent out from the creating vote users machine where the results are kept and tallied (this should be fixed in future releases as vote results may possibly be modified). A very basic gui used simply for displaying results and doesn't do any real processing whatsoever.

6. ErrorDialog

An extremely small utility class basically an abstract with a static function call that creates a JOptionPane error message with a passed in string. Very, very simple and self explanatory when you review the class file itself.

Module Communications

1. Protocol

The protocol is very simple, based on peer to peer negotiation and state progression. There are only three states defined and used within the Vote module, these being:

- PACKET_CREATED:
  This communication type is sent out to participating clients by the vote creator to let them know that the packet being received has only just been created, thus meaning a new vote has been created, and that the sending creator will be expecting a PACKET_RESPONSE from the receiving client.
- PACKET_RESPONSE:
  This is the communication type used within a PollDataPacket to let the receiving user know that a response has been sent back to their created vote, which should be awaiting such a response by the responding client.
- PACKET_RESULT:
  Finally, this is the end of communications, being sent from the vote creator (vote server) to all the clients, including itself, once all responses have been received for the given vote. This packet reception will kick in the production of a PollResultsGui showing the users the vote results.

The protocol tightly depends on the above interaction, basically it progress in a simple chain of conversation with the following occurring:

1. The vote creator (or server - for this example) creates a vote by setting it up in the PollCreateGui then clicking 'Ok'. When this is done a PACKET_CREATED PollDataPacket is sent out to all participating users in the summit (this includes the creator, as they can participate in there own vote as well).
2. Next, once the Cores of each of the participants receives a PACKET_CREATED PollDataPacket, the Vote module then brings up a PollResponseGui based on the PACKET_CREATED information. When the user finally finishes with their response and clicks 'Ok' or 'No Vote' it will send the results back to the 'server' that sent the original PACKET_CREATED PollDataPacket.
3. The server will be waiting for all PACKET_RESPONSE packets from each of the participants it sent a PACKET_CREATED packet to (new summit members who just join after the vote is sent out, won't be able to participate), Once all the PACKET_RESPONSE packets are returned the server will compile the responses and the send out a PACKET_RESULT packet containing the results to all the participants (including the server itself).
4. Finally when a PACKET_RESULT packet is received the PollResultsGui is shown containing the information held within the PACKET_RESULT packet.

Simple and straightforward. The protocol is not entirely efficient, nor is it error free. Further development time could improve this method significantly and reduce overheads and improve error handling.

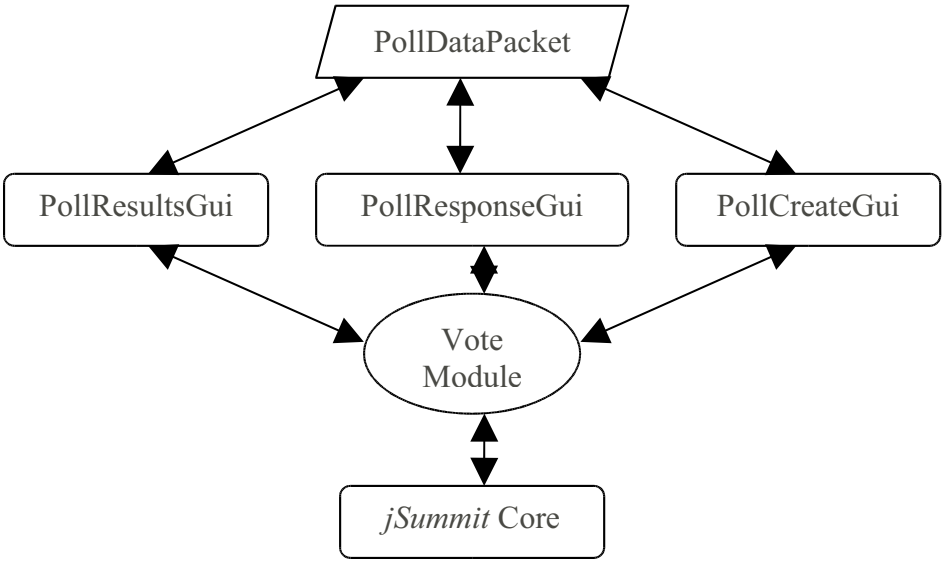
2. Core Interactions

The interactions with the Core are simple and not complex in the least. The module sends packets with the Core's implementation of *sendPacket* and the Core in turn, passes packets to the module through the modules *receivePacket* implementation.

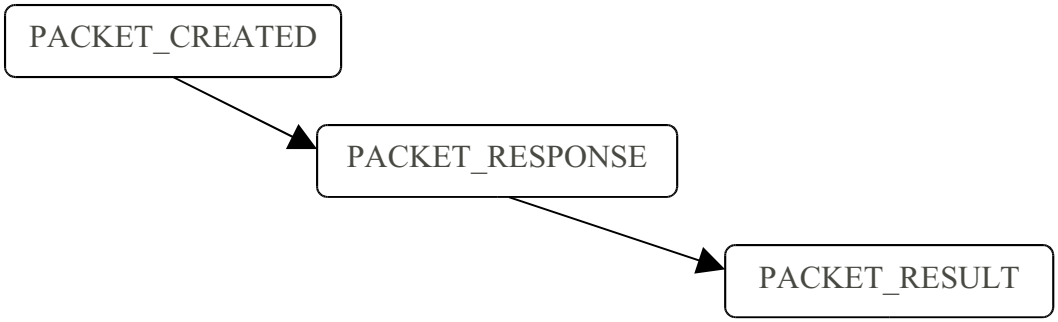
Each of the gui's in turn interact with the same methods to get the information across to other members.

Diagram Reference

1. Core Interaction



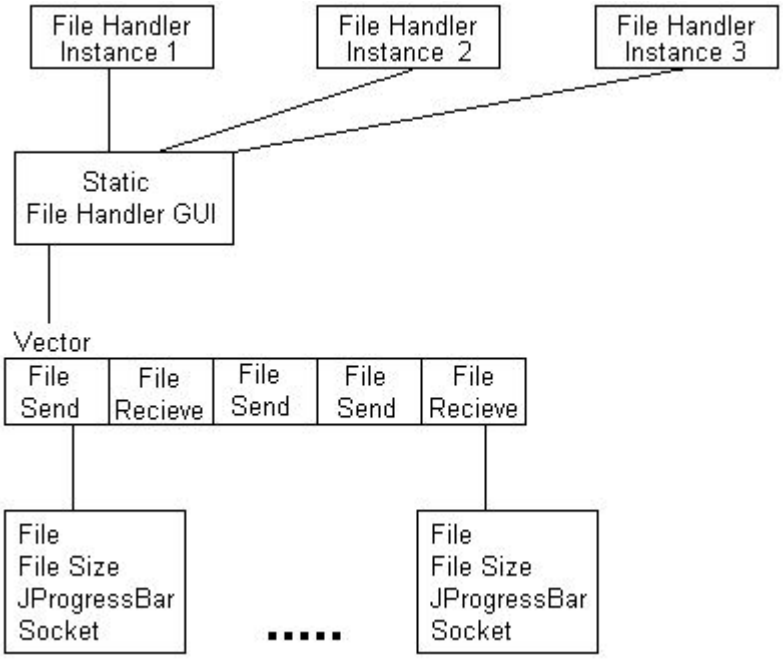
2. Protocol - Communication Chain



+------------------------------+
|File Sharing - by Tim Goodwin|
+------------------------------+

File Sending

The layout of the classes is rather large, however can be displayed graphically.



The GUI of the file handler is declared as static, so all instances of the file handler will hold the same GUI, the same records, and the same sends and receives. Inside the GUI, the JPanel, which displays the file send/receive, is a dynamic JPanel. Every time a new object is selected in the list, the same JPanel is taken, cleaned, and then all information is obtained from that file send/receive and added to the JPanel.

File sending is done in 2 parts. The first is the initialization process, and the second is the actual sending of the file.

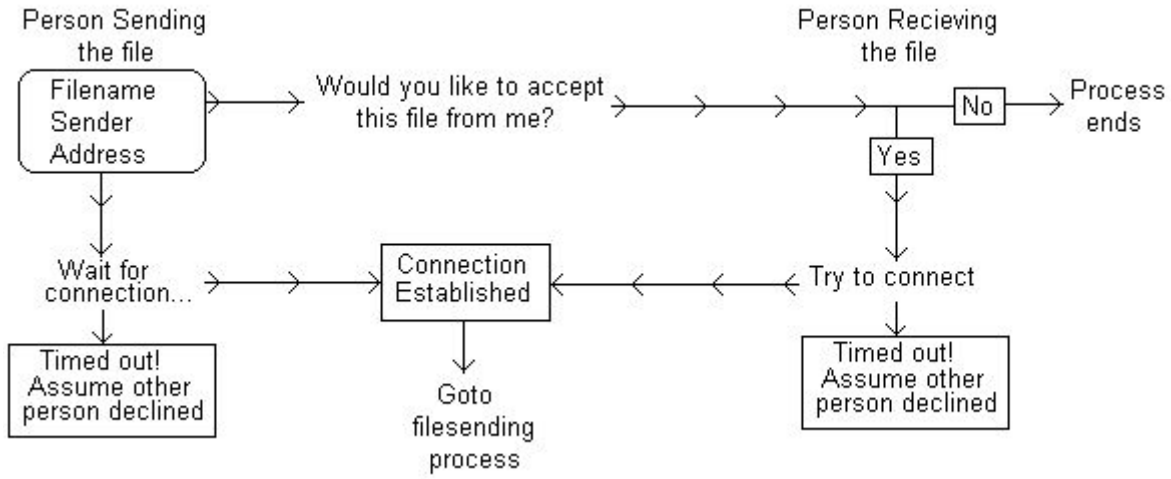
Initialization process

The person who wishes to send the file initializes the whole process. They send a packet, the other user, which is of the following make up,

```
public class FileInitPacket implements Serializable {
      public String sender;
      public String filename;
      public InetSocketAddress address;
}
```

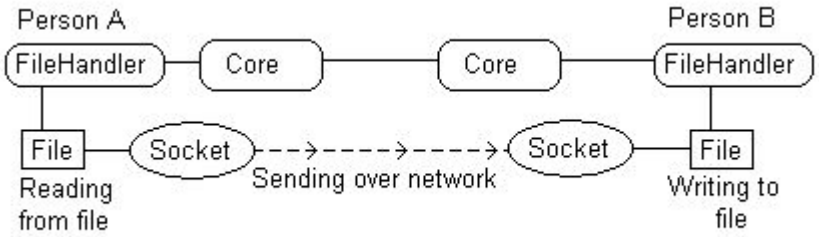
It has the following members
sender: Who the packet last came from
filename: the name of the file being sent
address: the IP and port number of the sender

The general protocol of how the initialization works, is shown below



File Transfer Process

This process handles the actual transfer of the file, from the sender to the receiver. Unlike most modules, and more like the video and audio streaming, the file handler manages its own sockets, for ease of traffic on the core. While sending, the layout of a person to person is as follows,



During the transfer process, in order to not flood the TCP layer, to allow monitoring of its status, and to allow resumes of file sending to occur, the file handler sends files in 5k size packets.

+--------------------------------+
|Video/Audio – By Phillip Street|
+--------------------------------+

Introduction

Audio and Video functionality is provided within *jSummit* through the addition of this module. Multimedia conferencing capabilities are a defining factor in today's conferencing software tools. The audio and video functionality was designed with ease of use and yet powerful capability in mind. The only downside to including audio and video functionality is that it can be very resource intensive for a system to run and requires a computer with moderate power to operate useably.

Advantages to using the *jSummit* audio and video module within a meeting environment include:

- Face to Face conversation.
- Auditory expression.
- Ease of communication.
- No typing skills necessary.

The majority of the implementation was done using Sun's Java Media Framework. Without the high level API and functionality provided by the JMF this module would not have been a possibility.

Further into the manual there will be given more detailed explanations on class operation and interactions. Also, the protocol used to communicate between cores to get audio video conferencing operational will be detailed and explained and some diagrams will be used to give a visual representation.

Class Details

1. AudioVideo

The AudioVideo class (contained in *AudioVideo.java*) is the main class where the module is instantiated. The class prototype is as follows:

```
public class AudioVideo extends JSModule{
      public static final int VidServerPort;
      public static final int AudServerPort;
      public static final int VidClientPort;
      public static final int AudClientPort;
      public static final int BasePortStart;
      public static final String VIDEO;
      public static final String AUDIO;
      public static final String REQUEST;
      public static final String STOP;
      public static final String SENDING;
      public static final String SENDABORT;
      AVTransmitter theTransmitter;
      AVReceiver theReceiver;
      Vector audPorts;
      Vector vidPorts;
      Hashtable assAudPort;
      Hashtable assAudPerson;
      Hashtable assVidPort;
      Hashtable assVidPerson;

      public AudioVideo(JSCore);
      public void run();
      public void sendPacket(Person,String);
      public void receivePacket(JSPacket);
      public void sendStop(Person);
      public void sendVideoStop(Person);
```

```
        public void sendAudioStop(Person);
        public void showVideoFrame(Vector);
        public void showAudioFrame(Vector);
        public JSCore getCoreRef();
        private boolean transmitterEnabled();
        private void createAVTransmitter();
        private boolean receiverEnabled();
        private void createAVReceiver();
    }
```

Rather than going into great detail on each of the sections within the class prototype we will instead focus on the interactions and a step through of the module initialisations.
Only one module instance should ever be running within the Core or problems will be encountered with socket binding and callback problems (to do with module referencing).
During the initial construction the variables are preset and initialised and the module then proceeds to kick things off by creating its transmitter and receiver sub modules. The audio and video module will then await packets received from the core to handle and also for function calls to its *showVideoFrame* and *showAudioFrame* functions.

The module also has its own *sendPacket* function to aid in sending out packets which basically have the same details in most respects and then passing them off to the core to handle. Its *receivePacket* function is used to handle all protocol interactions from other summit members to enable audio and video conferencing to proceed.

Whenever the audio or video icon is clicked in *jSummit's* toolbar it will make a function call to this modules respective showFrame which will then handle the interaction between members necessary to start the streaming (this will be explained in detail further on).
If, however, it occurs that there are no physical devices to connect to (being webcam or microphone) the transmitter for that device will be disabled and the user won't send that particular stream out upon request, instead sending abort messages. Yet again this will be covered in detail further on in the Module Communications section.

2. AVTransmitter

The AVTransmitter class basically provides the full server functionality of the conferencing streams sent out to listening clients. Audio and Video streaming tends to work in reverse of the standard client-server methodology. In streaming, the server is the one who initiates connection with the client and proceeds to push data down the connection while the client waits for the server data to arrive and then proceeds to process it and pass it along for visual or auditory output.

At the start of the AudioVideo modules run it will create an instance of this class to handle the streaming from the WebCam and Microphone hardware devices. If either of these devices can't be configured at start then that device stream is disabled. Something odd that may be noticed is that when the server has the devices configured and begins to stream, it targets itself as a receiver, even though there isn't a client listening for the data on the targeted port, this is to enable the multiple streaming capability to more than one remote target.

The transmitter keeps its own record of sessions currently being transmitted out to remote hosts, so that when a stop request is received it can remove that session target. Fairly simple in concept, think of it as the way in which television stations transmit information out (although this is more targeted than spread broadcast) and local television units receive the signal.

Because the streams are being sent out using a protocol (RTP) over the network using the UDP transport method, it is semi independent from Core operations and

uses it's own port ranges for the socket bindings.

3. AVReceiver

The stream reception functionality is provided within the AVReceiver class. This class doesn't technically handle stream reception as such but handles the record keeping necessary to look after the multiple conference connections that can occur in a *jSummit* meeting. The AVReceiver has function calls within it to enable audio and video streams that are incoming to be sent to the appropriate playing AVFrame owned by the sending individual.

4. AVFrame

All incoming streams (both video and audio) from users within a *jSummit* session each receive an instance of this class which takes all streams from one particular user, per instance, being sent to this user. The AVFrame instance keeps a record of the incoming streams, who the user is that is transmitting them, the running players for the streams, data source records and backward references to the AVReceiver that is keeping a record of the frame and a reference to the AudioVideo module itself.

The AVFrame will combine the audio and video stream output into one nice frame that shows on the screen for ease of use and simple layout. When the frame is closed, the frames dispose method will handle proper disconnection and bookkeeping within the module.

Module Communications

1. Protocol

The protocol used within the AudioVideo module is simple yet effective. It is basically a very simple handshake protocol between two summit members to agree on a transmission or reception of streaming media.

A *jSummit* user initiates a stream request to another user by clicking on their name in the userlist and then clicking on the audio icon or the video icon. What actually occurs is as follows:

13. The client wants a stream from someone serving
14. The client queries the selected user asking it to start streaming to a given port e.g. "REQUESTVIDEO 59002" or "REQUESTAUDIO 59004", the way that a client port is found is by keeping a vector (or array) of currently assigned ports (starting from the baseport), generating a new port that isn't already recorded and sending that as the request port while then recording the new port number.
15. The client then records the server in two cross referenced hashtables (a hashtable with the port as a key, the other with the username) until it receives a SENDING or SENDABORT response from that server.
16. The client then waits until it gets the SENDING message for a request it made or a SENDABORT e.g. "SENDINGVIDEO" or "SENDINGAUDIO" at which point it will check its hashtable to ensure the server sending the message is a server that the client made a request to.
    i. If the client didn't make a request to the responding server, a STOP is sent to that server.
    ii. If the response satisfies a request from the client, then it proceeds to call a play request in the AVReceiver for that server and stream.
17. However, if a SENDABORT is receiver it notifies the client that the server can't stream the requested data or is experiencing problems and won't stream.
18. Once the user is finished and closes the AVFrame associated with a server, the AVFrame will then send a STOP request through to the server sending the streams.

The formats used in transmission of the streams are as follows:

Video - H.263
: Ideal for video conferencing where there is little to no action. It is lossy but still ideal as a low bandwidth transport format for live video capture.

Audio - G.723.1
: This format is similar to low bit rate telephone transmission of speech. G.723.1 handles verbal communications satisfactorily over the streaming medium.

Appendix A - jSummit Project Diary (Newest to Oldest)

Thu, 21 Oct 2004 Trade Show
As Tim would say "ZOMG!" We did it! Trade show was awesome - we didn't get the $500 but oh well - checking out some of the other groups work - well done guys!

But ours went well - we demo-ed it to various people, got grilled by others but all in all it was successful. We had some issues in the morning but we had it working on not only 2 computers but THREE! It was really good - people were impressed. The Poster looked awesome, the commercial was really good and wonderful and all and now we are going to get PIZZA!

Okay we haven't entirely finished - the website and tech man and demos need completing but its almost over! Whoo hoo!
Sun, 17 Oct 2004 Well well well
The code is all finished - Tim is just putting the final 'tweeks' to his File Sharing module but everything else is working rather nicely I must say.

The end is in site - Trade show is on Thursday and we honestly can't wait - jSummit is awesome - we are so proud of it.

Personally I have finished the User Manual ready for printing, the Commercial is done (just needs recompressing and putting up on the website), the Powerpoint Demo has been emailed to Koren for the Trade Show, Daniel already has the poster and the Technical Manual is well underway but it will be fun trying to get it finished by Trade Show - hopefully we can but I have a Printed in Draft Mode version of the User Man to take and I revamped the website today changed some of the bios to more reflect the roles for the members of jSummit and added up the documents mentioned above. The rest of today will be devoted to getting a demo of jSummit done and the Tech Manual more completed with diagrams and all.

We shall be putting a Linux/Windows version of jSummit up on the website by the end of this week though the Mac version will not be available because I don't know precisely what needs to be tweaked in order to get it to work properly and I don't have a Mac that has Java on it to test it thoroughly (it is mainly GUI changes the underlying code will work the same)

Sat, 9 Oct 2004 More stuff
Okay Phil's having fun with the Video and Audio (not sure if Pete's doing the same thing? No contact....)
Tim's bug fixing his File Sending whilst Grant fully integrated the Private Chat into jSummit.

Grant did some minor bug fixes and 'touch-ups' whilst Jenna completed the Help menu and did some MainGUI 'touch-ups'

User Manual is coming along nicely (if Open Office will co-operate) and should be completed tomorrow. The graphics on the website will be overhauled tomorrow as well.

Also contacted Daniel about our 'presentation' on Monday but he's going out of town - we are a bit lost as what to do about it... Going to e-mail Koren.

Also also we are going to film some more stuff for our 'commercial' on Monday with a LAB COAT

Thu, 7 Oct 2004 Late session code scramble...
The deadline is looming....

The past week has been pretty hectic for all involved. Jenna & Grant have been working on getting final features and tweaks fixed and in place within the Gui and Core. A lot of little last minute things keep popping up.

The poster has been printed (and we must say it's a fantastic effort on Jenna's part). The user manual is coming along nicely, just waiting on a few of the modules to have their manuals submitted.

Tim has been working solidly on his filesharing and private chat code (Grant has been helping out with a few code tweaks and bug fixes). The project is starting to come together, conceded it's not a complete nor fully polished product but it will still be a worthy competitor at the trade show.

On a side note, My polling/vote module has been completed, some things had to be changed and/or dropped from the initial spec. There are some known issues but they can be rectified or worked around. With the deadline fast approaching and no definite submission of the audio/video module coming into action, I've taken the challenge on to make up a backup module, just in case Pete encounters difficulty with the integration and development (JMF is a nasty little beast to work with - likes to bite).

All in all, an extremely productive week on my own part, the group members hard work and perseverance is starting to come to fruition. Nerves will be the last thing to contend with...

Regards,
Phil.

Thu, 30 Sep 2004 Thursday Meeting
Jenna and Grant went to see Daniel to discuss a variety of things including Tim's justified outburst at Pete, the poster (which Jenna completed the night before), the commercial and 'demonstration' as well as getting the information on what we will be getting during the tradeshow. It has come to our attention that we wont have access to the Poject lab during week 12 so we need to have project completed by end of next week at the absolute latest.

Grant has almost finished the whiteboard, the Private Chat plugs into the mainGUI, File sending is almost complete (in it's beta format at least and being plugged into the CORE).

Grant and Tim abstracted out a Image Sending Module that needs to be created so that the profile images and whitebaord images can be sent as Java doesn't have a serializeable image class for some reason

Tim gave Jenna a piece of the User Manual and has decided that if the video module is not completed by monday as promised he will see what he can do to finish it off.

Wed, 29 Sep 2004 Holidays? What's holidays...precious, what's holidays?
Sunday:
- Tim went home then spent the night coding away on the 3rd module he has created
- Jenna worked on the Help Menu and wrote more on the User List to get it to function properly
- Grant fiddled around with the CORE some more making it even more robust
- Jenna mapped out a design for the poster

Monday:
- Phil (for a time), Jenna and Grant went to Tim's house and we got jSummit going 4 ways in the Global Chat - after dinner J,T & G continued some more (whilst Jenna's computer had windows reinstalled she nicked Grant's second computer) - testing various bits of the socket code and changing various GUI items and the like. Grant wrote the disconnection manager as well whilst Tim fixed up some minor things in his Private and Global Chat, Jenna worked on the mainGUI some more and wrote a Help Frame
- They also filmed some footage for the commercial and discussed the poster
- Pete got sick. He got a medical certificate.

Tuesday
- J,T & G met up at uni and got some more commercial footage whilst trying to find Daniel to ask him what he thought of the poster design
- Tim had fun at a UNI LAN (which he deserved) whilst Jenna and Grant worked on the whiteboard one doing the icons the other doing the actual code ;) Though both taking as long as the other

Wednesday (as of 4.30)
- Tim got beta version of his File Sharing working with robust port cycling and is proceeding to plug it into the core
- Jenna successfully added her icons to the Whiteboard, got UserList opoup menu working and fiddled a little with the Modules and showFrames for Private CHat and Audio to get them integrated into the core
- Grant is working on changing the user status, removing a user and general core functions that need to be plugged into the mainGUI menued items

Sun, 26 Sep 2004 LANTek
Grant: Password for the Summit was implemented and working and tested. Peer-to-Peer code was tested and modified

Jenna: Put restrictions on number of characters in Profile area, also did the Help GUI and list functionality

Tim: Wrote Private Chat JSModule and got the Private Chat connecting through the core

We also played Rune, Phil insulated Jenna so she swore at him. Tim established how to use the video camera and fell asleep at his computer at 4am - yes we have photos

Also Grant got a RoboSapien - we are so putting jSummit on it : P

Sat, 25 Sep 2004 Too excited
O.M.G. jSummit actually works its amazing I can't believe it! We got it going through the core and the global chat works!
All that hard work and late nights have paid off for Jenna and Grant - Tim just needs to made a module wrapper for his private chat and it too will work ;)

Fri, 24 Sep 2004 Quick Update
Another late night coding. Took out the Demo code for the toolbar - need all modules to extend JSModule in order for the toolbar to function properly. Superflous menu items and toolbar icons have been removed.

Grant is churning away at removing all of the runtime errors and we got the users connected - go us! this close to sending legitimate packets through the core - Marco -Polo work like a charm and user incrementation if user already exists in list is working (though needs a little tweaking) - password yet to be finished but shouldn't take very long to complete.

Jenna wrote the JSFrame class and got it integrated into the Window List - still a few bugs but they will fix them in the afternoon. Her Userlist details display is still a little buggy too but she's working on fixing that up too.

Only 24 hours till jSummit prototype needs to be complete.

Jenna and Grant will put together a demo poster (after Tim gives us the example) on Sunday to show the meeting on Monday - Phil and Tim will be looking after video taping our big test for the 'commercial' - wonder if we can get any really cheesey images ;)

Note: Jenna just updated the index page of the website with current news and updated the bios to better reflect the roles taken on by members (plus a joke one - guess which)

All module writers are to write their section of the UserManual by Wednesday so that Jenna can put it together and get that part out of the road - Grant will be updating the website whilst that is going on and the others will be tweaking their code to make it completely finished by the end of the holidays.

Thu, 23 Sep 2004 Minutes
Okay Tim and Phil met briefly with Daniel in the morning a quick hi.

On Monday Tim, Jenna and Grant met up after Jenna's tut and talked about jSummit and she wrote the code that changes the user profile details with debugging help from Grant and Tim

On Tuesday night Tim finished off the Private Chat and sent it along to Jenna whilst Phil updated his poll stuff as well - later on that night Pete got his AV Video code up with a few minor bugs that the rest of the team helped him with - Jenna integrated the new Module classes into the jSummit workspace and made them link through the toolbar icons

On Wednesday Jenna and Grant fixed up the Private Chat's minor bug and Grant got the addition of users to the Summit all worked out and working, he also helped Jenna debug her code and finished off his Peer-to-peer connections. They also thought about design issues of the Poster, User Manual and Tech Manual which will be written over the holidays. Tim almost finished his File Sharing.

On Thursday Phil, Pete and Tim stayed up in the computer lab working bugs out of each others code. Tim churned away at his File Sharing code and as of this time almost has it complete whilst Jenna fixed up the Status Bar and Main GUI to remove the superflous menu's and icons. It's been a lot of hard work implementing ALL of the functional main GUI code (although Phil typed the first few bytes they were just a few rough frames which didn't do anything, and have since been overhauled completely and replaced with actual functional code. Currently the only file/class that is of any real use in the current project is Phil's "BaseListener"). Grant did a tiny bit (the "on-press" icon code), and is grinning stupidly with honour at adding something to Jenna's mammoth effort. He wanted to claim credit for the cool icons Jenna did, but would never take credit for effort as huge as that ;)

Also on Thursday, Grant censored a diary post by a really, (and completely justifiably) angry Jenna :)

Sun, 19 Sep 2004 Weekend Coding
Tim: Worked on getting Global and Private Chat completed - and did!
Jenna: Did the MainGUI - the ENTIRE MainGUI based on the outline code Phil had done months ago - added appropriate graphics and integrated into the jSummit package as well as fixing up the Profile and Room Selection plus a Password File and a variety of new pressed icons
Grant: Did MORE socket code including the peer-to-peer conenctions between the cores as well as completing the rest - it only needs to be runtime tested to make sure it does what it should

The others: ??? No clue.

Thu, 16 Sep 2004 Meeting
Okay no Daniel Meeting again though he did say on Monday not to make the project un-completeably robust and some ideas on how to simplify what we have done. Though we still seem to be in the dark about the poster (which I have asked the rest of the team to think about but they seem to be more worried about how much it will cost rather than who is going to contribute *sigh*)
Pete was at work though as I understand it he is currently working on getting the Video and Audio components finished - well enough so it can be integrated any way.
Phil played around with his Poll and showed us what it looks like and has decided how it is going to work and time-outs and the like
Tim finished his File Sharing earlier in the week and has moved onto the Whiteboard - he also tweaked the Global Chat making sure it was ready to be plugged into the mainGUI
Grant got more of the core Server code going (at the moment you can create a new summit and displays it in the profile/room selection and creation that has been fully completed -> by sat night it will create a MainGUI and have packets sending through the core for the global chat)
Meanwhile Jenna got the profile and room creation/selection fully finished and is working on the MainGUI and what it needs to contain - she will be finishing it on staurday and getting functionality to the buttons by sunday so that it can call all of the modules.

We have decided that we need to start 'plugging' in our modules and hence the core needs to be finsihed so that is why Tim, Phil and Pete have to get their modules finsihed by end of the weekend whilst Jenna and Grant get the core and mainGUI finished --> on Monday a large coding session has been organised to do some integration

Daniel sent to us:
Project Diary (5%)
Prototype and final web site demonstration (due week 11) (5%)
Peer review of process and group collaboration (due week 13) (5%)
Final Product (due week 13) :
- code (25%) documentation (20%)
- mpeg tradeshow presentation + final project web site (10%)

No-where is the poster mentioned and I am curious why the website is worth about 25% overall - it is a very static thing - seems a little over assessed to me - shrugs; anyway, only 1 week to get our code ready for the big LAN test!

(I never did mention that it is Jenna who writes these diary entries)

Sun, 12 Sep 2004 Sunday "Crunch Down"
Jenna and Grant sat down with two computers and did some heavy coding. Grant made a lovely Globals class for all our static variables whilst Jenna cursed and sweared at the GUI making abilities of Java but managed to fix up the profiles and retrieving the summitinformations passed through from the Broadcast frame

They also abstracted out the connection manager of the cores as well as worked on completing this code (with the aid of a multitude of diagrams)

After 12 hours of coding they realised that jSummit is a lot of work but it is slowly coming together - that and the ISP (Inspirational Socket Puppet) is a real help ;)

More coding tomorrow - only 12 days till LAN testing

Thu, 9 Sep 2004 Minutes
Yet again we didn't have a meeting with Daniel which is I suppose a good thing in that we don't have anything to actually show (it's going to proabably be nothing-nothing-nothing-nothing -jsummit when all the underlying network code is fully written) - it has all been designed out and the following schedule of events confirmed:

By the 11th CORE code will be completed along with user-lists and beginning profile and summit selection (to be completed by Jenna and Grant)
Pete is to finish his Audio and Video (he was not present at this meeting)
Tim has the File Sharing code working nicely and was working on a File Sharing dialogue - he will most likely end up taking over the whiteboard whilst Grant handles the CORE instead
Phil has decided on one poll per person approach and will be working on OS specific subnet mask 'getting' programs

- All the modules are to be completed by the end of Week 9
- By the end of Week 10 jSummit will be in its beta testing stage
- On the 25th of Sept - jSummit will be tested at a LAN party to see how it holds up
- 25-4 Poster, User Manual and Technical Manuals will be completed
- Poster is due on the 5th of October
- Video will be created and manuals 'tweaked' in Week 11
- Week 12 will probably be our presentation week

On the 21st of October jSummit will be at the UoW Trade Show;

We have our work cut out for us but if we sit down and do some hard coding we will get it done! Only 2 weeks to go!

Thu, 2 Sep 2004 Minutes
Another productive meeting - boy we are on a roll - though yet again we didn't meet up with Daniel (is there a pattern forming here? Nah just kidding), Daniel was in Perth - pity we didn't have jSummit available to conference with him...Though on Monday there was a brief meeting after 322 where Daniel suggested how we might capture the LAN's available on the client so both Grant and Tim got a little gung-hoe and had a crack at it based on a suggestion by Pete.

In the meeting Pete set up CVS on his machine at home so we actually have a proper repository for our files - which is really good as things start to come together it is working out rather nicely.
Tim started working on file sending across the CORE whilst Jenna fixed up her profiles code getting it to read available profiles from a file and both Edit and Save them. Grant got stuck into more of the CORE after a little designing session with Jenna earlier in the week and Phil churned away at the Poll.

Overall whilst we still don't have anything to 'show' in a sense the networking code and the under lying design and structure is coming together very nicely - the chat is almost complete it just need beautifying and integration and the CORE is a few steps away from being ready to pass module data through

Whilst we only have 3 weeks to get this finished - we can do it *chants jSummit cheer*; (okay so we don't have a cheer but it would be nice....)

Thu, 26 Aug 2004 Minutes
Most productive meeting that we have had yet which is a good thing. Jenna, Grant, Tim and Pete sat down in front of a whiteboard to abstract out the details of the Core and how it insteracts with the modules in a more precise detail and to diagramise the API that was written two weeks ago.

Also a shutdown procedure was written how the program might handle itself in case of a 'crash'. As well as a pseudocode version of the start-up procedures and what has already been written - a time line for the completion of the project has been set for end of week 10.

Jenna, Grant and Tim then sat down and started thinking out the design of the CORe and how the module frames might want to work togther comparing it to the Marratech way of 'doing things' and some other possible ideas. As well as how the USerlist will be handled by the 'server' Core and how this might interact with the other modules in the program. The trio then went into the Java lab to try and get the SIR packets working and debug the code to make it work, Tim then took this home and churned on it all night to get it right
The plan is for Jenna to have Profiles and Userlist functioning by next meeting, Grant will be working on the CORE design with Jenna and Tim, Tim will be the 'debugger' of this code finding all the silly errors and correcting them whilst Pete and Phil continue with the video/audio/poll.

Sun, 22 Aug 2004 Website Updated!
Okay took several hours of Grant and Jenna swearing at the 'tables' (okay just Jenna and Grant laughing at her. But but it works now and I am incrediably happy and the new logo up the top makes it lok even better, plus with a new SourceForge link (that really should've been there since the begining) and updated roles in everyone's profile as well as modifying some fo the screen shoots to show newer versions of the code including the new icons for the Main Gui :)

And a nice new look for the diary to more suit the new colour scheme for the website

Thu, 19 Aug 2004 Minutes
Today we created some use case diagrams for all aspects of our project. Also made some class definitions for the modules as well as who will be implimenting the classes. Core packet headers were finalised as well as wrapping a core packet and sending it through. As a group we also viewed the other project websites and decided to spruse our up a little, as we think ours is lacking especially after seeing one of Koren Ward's project group's website.
Module packet API and abstract classes was also reviewed and updated along with

the Summit Info class and the implemented functions. We have also made a considerable amount of progress on the Splash screen program that broadcasts to find jSummits.

For this week we will have that part implemented and the basic sending of a module packet with the appropriate header that can be decoded and send off to the receiving module.

Thu, 5 Aug 2004 Minutes
Incrediably un-productive meeting though some more of the braodcast code was written - but more procrastinating - saw Daniel said Hi, listened to some chit chat, got roped into some movie thing - you know.

Thu, 29 Jul 2004 Minutes
Had first "official" meeting of the session with everyone. Showed Daniel all of our groovy new guis and what we had been doing over the session break. He was mighty impressed - well actually he just got annoyed with Phill taking over his desk space. After that meeting jSummit went up into the project lab and for the first time ever we all got coding - actually doing work - though Pete got annoyed with the firewall in the project lab that was preventing him from fully testing out his video code - but he got the web cam sending images through his program. Tim was very happy with the way his code was going and Grant had a whole red vs blue thing going on in the whiteboard.

Phill is going to create the networking protocol and with Grant ironed out some of the details of network packets and how they are to communicate whilst Jenna put up all of the current code on Phill's home machine for easy access and started developing the splash screen program that uses the broadcast code written during the holidays.

All in all one of the most productive jSummit meeting that we have ever had. :)

Thu, 22 Jul 2004 Minutes
Phil, Grant, Tim and Jenna met up in the Porject lab and discussed various wonderful things including how great the website looks and how great the GUI's look. That and we discussed various things like sending objects using the ObjectStream classes and wrote some code to test this that actually works!

Tim had fun (and much swearing then rejoicing) getting different font types and colours for the different lines of text in his private chat modules.

Also we decided to not make things plugin yet - though that might change in the future.

Tim was swindled into joining the new SITACS student society but Donkey saved the others. None of the team members are on drugs (we swear we didn't know that package contained anything illegal please let us go to the Olympics).

Mon, 19 Jul 2004 Diary's Back!
The Diary's back again - everyone is rejoicing in the streets - singing, dancing - oh wait no they aren't.....

Mon, 12 Jul 2004 Server Code
Some of the client server code was written and the idea of using XML as packaging structure fully fleshed out - the C-S is currently being written and more of the GUIs are being fleshed out to have functionality - the Chat program has changing images and really good resize capabilities :)

Wed, 7 Jul 2004 New Contact Form on the website
No more mailto: links for us!
http://jsummit.sourceforge.net/index.php?p=contact

Fri, 2 Jul 2004 Mini Meeting
Jenn, Grant, Tim and Pete met up to fix up the bits of code that we had created for the GUIs (which are now avalible on our website under 'Screenshots') - we went for a blue coloured look for the most part and we discussed what more would

we need to do. Though we were planning on getting a small socket program written that really didn't happen but it is to be done by the end of next week.

We really have been trying to get some work done but in reality it is really hard as so many things keep on popping up.

The project is progressing forward which is always nice and hopefully we can put everything together in the near future and have something working by the end of the holidays - 2 weeks? Yeah we can do it

Fri, 25 Jun 2004 Website
Yay finally updated to make it look super neat rather than neat :) Added a nice background, new colours, updated screens, new "Releases" and Downloads areas, still waiting on Phil's photo though

Also the top image was changed from a smaller version of the splash screen to something a little more presentable. I'm quite proud of it if I do say so myself. Coming soon will be all new Screen shots of the Prototype.

Fri, 18 Jun 2004 Minutes
Minutes Fri 18/6/2004
-Grant, Jenna, Pete and Phil met up in the project lab to discuss what we are going to do over the holidays
-Looked over the idea that Jenna had set up about the user list and IP adress and what sort of protocols that will be used for the network traffic (such as UDP, TCP)
-Then started looking at user interfaces and using JCreator to write our projects - every one got a copy of JCreator
-Deligated tasks as follows to come up with the GUI of the various models:
Jenna: Profile/ Room Creation and Main GUI
Grant: Whiteboard
Tim: General Chat, Chat and File SHaring
Pete: Audio and Video Conferencing
Phil: Poll and start working on ideas for network socket codes
-Decided to have another meeting in a weeks time to put together what we have and to make sure that we can get it all compatiable

Tue, 8 Jun 2004 Diary's Back
Yay the diary is back and after some tweaking is now fully functional and back to what it was like before oh happy dance. Pity about exams though as nothing is going to get done this week at least.

Wed, 26 May 2004 Minutes
It is amazing how when one member is sick and can't make the meeting that hardly anything gets done - but the website was played with, the tech manual put up though the diary went down and is in the process of being relocated to a new server (hence the 'stuck in text format'). We decided that we will be getting a prototype with chat and minial whitebaord functionality by the end of our holidays and that we need to continue coding it and getting the cores to connect to one another. But with exams coming up we are putting the project 'on hold' until they are over before having a major coding session to get stuck into it.

Peer Assessments were due and the like - lots of bribing. But really the group works well together and everyone does their part so we have good group dynamics at the moment.

Ah the diary went down and we have decided to move it to Pete's machine instead

Wed, 19 May 2004 Minutes
Very unproductive meeting, we did dicuss some more about the CORE and its interactions and we started writing an abstract class for the modules themselves as well as dealing with the issues of when a core drops out and what this might entail.

Wed, 12 May 2004 Minutes
Had another meeting today though this one was much more productive than the last

as we started talking about the exact classes and abstractions that we will need to the Modules and the Main GUI as well as having a look at the Java SDK in order to see what functionality we would need in the abstract class for the modules and how we would load them exactly.

Jenna also brought in some possible icon ideas - but the 'look' is now on the back burner we are instead concentrating on the Core and how it is going to interact with the modules (we had a Polling vs Event debate and it is still going on now).

We have decided (with Daniel's insistance of course) to have a working prototype with CORE, CHAT and minimal WHITEBAORD to be ready by the end of mid session holidays - so this means that each of the group are going to think about the interactions and specifics before the meeting on Wednesday where the whole team will sit down and get some coding down (preliminary code will also be brought to the meeting as well)

Fri, 7 May 2004 Tech Manual
After a last minute scramble, coffee and MSN meeting the prelim tech manual got completed as of 12.51 today.
Most parts were completed (minus the audio which is much the same as video anyway)
Some pretty diagrams and layout was completed and made into pdf format
This was then sent along to Daniel. [Click on the Image to see larger version - it is the image that features in the Tech Manual]

TO DO: Big coding session where all members start coding the CORE program - there are many assignments due for members this weekend so they probably wont have time spare then but hopefully a coding session will be arranged

Wed, 5 May 2004 Minutes Wed 5th
Went though a lot of things in both meetings and came up with lots of plans, ideas and tasks that must be done.One thing though the Prelim Tech Manual needs to be submitted on

Friday so we have to get it done.First thing in the meeting we talked about the Core and some ideas on how it would work - what sort of packets would need to be sent as well as the types of connections we will be using:
1. Have a 'server' that sits in the middle, CORE programs will wrap packets and send them onto the destination and then 'unwrap' and send tot he appropriate module
2. Video conferencing and audio make their own connections
3. Wrapper of CORE will include:
UserID (Inet Address)
Module ID (enum)
Data (object)
[Wow serious dejuvu]
We then discussed some of the 'layout' of the

Tech Manual that Jenna had come up with - though Daniel told us not to bother with UML - but it makes it look pretty
Also discussed possible exception and error handling such as if a CORE drops out then the video connections to this core are also severed - Video uses UDP and it wouldn't notice

Also Threaded server (issues with Swing will have to be addressed later on if they arise)

Wed, 28 Apr 2004 Wed 28th Minutes
Need to look at possible functions and the data structures and protocol - BASIC core protocol
- State (intitate, terminate, continue) in the chat - chat structure wrapped and sent through TCP fromt eh core
-core needs src and destination
-articulate the protocol and how it actually does these sorts of things

How are the modules going to converse with one an other? Need to think about this as well

Need the following in the Tech Manual (after looking at a few examples)
Components, Algorithms (not yet though)
- assumptions and psuedocode, Datastructures - explain and the members, Configurations, functions inc pre and post conditions
- what has to offer
How components interact with the core and the types of information that is exchanged

Daniel's
'wants': Spinning icons and windows especially tween the users in video conferencing - nice icons that bounce and generally do groovy things

Mon, 26 Apr 2004 MSN meeting Monday 26th April: Grant, Tim, Jenna and Phil and Pete
Grant: well we can start with the modularity... i'm guessing there'll be a core that'll handle connections to the room, and modules'll pretty much just be independent applications that'll submit packets to a core scheduler thingy which'll pass them onto the room, and the room'll pass named packets to a module... since the modules open seperate windows there's no real need for other comms. okay, well it shouldn't be too hard to whip up a basic template for the modules to submit stuff to the core... basically just module ID, user and/or room, and data... the core doesn't need to know what the data is, it just needs to be a dynamic size

Hmm how does the module distinguish valid users/rooms?
Grant says:it should be able to ask the core
Phil Street says: Fair enough
Phil Street says: Hmm ok easily done, just need to use a common interface class for modules.. easy.

Putting Zero COnfig on hold

protocol would just be the net traffic... core packet containing headers and a module packet as data

MUST HAVES:
Profile and Room Creation
Main GUI (inc User List and Status Bar)
Whiteboard
General Chatting
Private Chatting
Video Conferencing
Audio COnferencing
File Sharin

OPTIONALS (1)
Poll
Environment Settings
Options Panel

ONLY IF HAVE PERFECT
Skins
Desktop Sharing

Drew pridy picture of the interfacing of the 'modules'

Taking a good gander at JMF on wednesday meeting

Tue, 20 Apr 2004
- Tech Manual needs: ER diagrams, Data models, and descriptions of protocols
Descriptions of components: what parts of JMF are we going to use and how
SIMPLIFY

Complex data structures
Authentication
Objects executing what functions
CORE FUNCTIONALITY ONLY
How we are going to go about doing things
Modules and how they are going to interact with one another
Decided to get it working on one platform before worrying about porting it onto another
TASKS: Play with JMF and get basics done of Tech Manual
- Webpage was updated with some of the new screen shots and the prelim user manual

EDIT: Changed year from 2005 to 2004 - took me a while to figure out why it was always up the top but I did - go

Tue, 6 Apr 2004
- Jenna showed prelim User Manual - tweaked it a little and did some more screen shots and the like
- Tech Manual: look at codec - put on hold till user manual completed
- Suggestion to limit functionality - only do core things (Chatting, File Sharing)

Tue, 30 Mar 2004
-Buttons we need up top: WebCam (to see who is broadcasting), Log Chat, All File Transfers and transferring a file
-Grant came up with the mountain idea (on the logo) and to coin the rooms as 'Summits'
-Prelim Webpage was put up
-Signed up at Sourceforge
-Tasks: Jenna & Phil to finish Mockups
Tim found a sample Whiteboard written in JMF = is possible - will try and use what we can of it
Pete VNC code for windows (c++ though)

? only if have time - VNC on the 'back-burner' so to speak
- Daniel told us not to worry too much about security issues such as encryption

Thu, 25 Mar 2004
Talked about access to labs and the user accounts which we don't have yet.

Tue, 23 Mar 2004
- looked at Marratech Options window (and decided against huge garish buttons) what we could have:
  Identity ->Alias and Person details
  Viewer ->Font Size, Button Size, (skins), (cookies)
  Audio ->Volume, Device, Enable/ Disable
  Video ->Volume, Video Device, E/D, Refresh rate

Supervisor Meeting 24th Wed
- familiarise with coco, Carbon - macOSX programs - we might need access to OS stuff
- find modules of things we might already have
- id functionality - Java modules that might already provide) ->JMF?
- find whiteboard 'component'
- critiquing other programs such as ICQ, MSN, Marratech, NetMeeting, etc
- put project diaries in a blog
- get sourceforge site up and running
- what we want to achieve in more detail
- User Manual - inc explanations and mock interfaces and make it really pretty
- Tech Manual - general arch, req, percieve to work
- elect a leader

Thu, 18 Mar 2004
Lecture:
- Prelim user manual needs: bulleted sub areas, overviews, objectives and the like in the (Tech Man need a description of the functions), common methods as

well as user interface mockups (QT Designer?)
- have layers of description (CD @ back with code on it)
Why have we done things a certain way (ie comments)
- tutorials (not really in prelim one though)

Tue, 16 Mar 2004
Encryption issues - public and password protected rooms - the admin is the person who started up the room
- Marratech review (uses lots of memory when the whiteboard is in use) - also supports pdf and word doc viewing
- INTERFACE DESIGN:
Open Screen Shots - Splash Screen (when searching for possible broadcasts and assessing connection details)
 - multicasting out using Zero-config
 - int/libraries/config files or set up screen
 - device probing
Profile Select / Creation
 - settings, name/personal info, default profile
Room Selection/ Creation
 - initial rooms + "refresh"
 - join-authentication
 - create -room settings
Room Creation
 - name, max connections, password (none if want) - encryption optional - security reporting to room owner
- can "Back" at any stage in the process too
- Plugin loader
- VNCing is now pushed down as a low priority - whiteboard as highly compressed jpg - will see if JMF or another module already has a built-in whiteboard

Wed 17th (Daniel - Supervisor Meeting)
- take a look at iChalk, Filesharing, Video Conferencing, IRL, Join all constructs in a nice simple and elegant package
- also wants multi editing of a word document
HMWRK: Roles for Group members, sort of features a user would want to see
-jSummit was proposed as a name and was accepted - get a Source Forge site

Mon, 15 Mar 2004
Discussed draft of the interface and possibilities of things we can omit at this stage - decided to call our project Aelgy, set up meeting time with Daniel, client/server and zero config discussion and started thinking about what will happen when the user first connects to creating profiles and rooms.
- when join already formed group an IP is assigned as part of the zero config protocol
- desktop sharing: are we emulating or full access like VNC
- tabbed chatting
- hideable windows - dock verses floating windows debate - support docking later on
- benchmarking the settings for connection speed and the like to optimise performance (ie turning of Video conferencing for those with dial up)

Thu, 11 Mar 2004
Possible Project names: Ringcast, GlobeMeet, Binky, Aelgy, Z-CAMP, VirCla, Occurous

Tue, 9 Mar 2004
Made up list of requirements (draft version), started doing some research (Zero-config), plug in debate, and that we should come up with the name for our project.
Mon, 8 Mar 2004
Discussed possible meeting times and looked over schedule

Thu, 4 Mar 2004
Met with Daniel Saffioti about the "Peer-to-peer Computing Environment for PC and Mac" - decided to do this

Mon, 1 Mar 2004
Met on Tuesday 2nd March, went through project list and emailed about

5 of the

Fri, 27 Feb 2004
Group formed Friday 27th Feb