CECS 327 Project Presentation

Riana Jara, Guanyu Ding, Andrew Chheang

Implementation

Our implementation was done by having the nodes communicate through events.

By using different event types, we are able to design a network system where nodes know when a new node joins and when one leaves, and files between the nodes are synchronized as long as these nodes are under the same network.

There is also an event to respond to a new node joining if one is already in the network.

Post connection events include removing files and sending files for the nodes.

Events

There are different types of events to show the different actions done. The following events can occur on the network as follows:

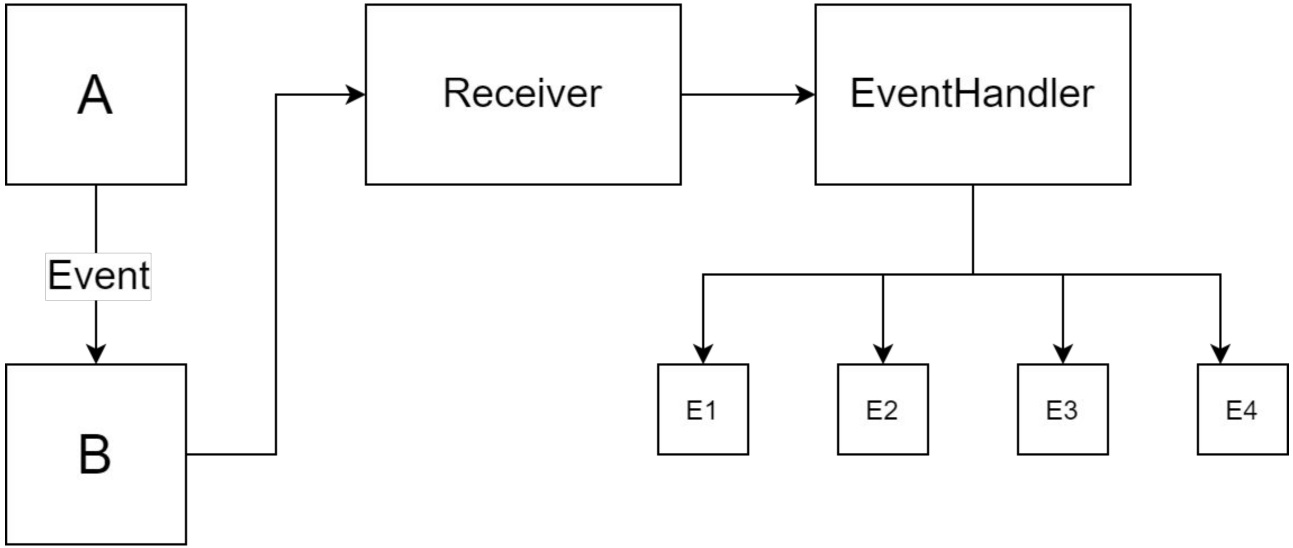
- Join Event
- Join Response Event
- Leave Event
- Send File Event
- Remove File Event
- Create Directory Event
- Remove Directory Event

Event Handler

The event handler will work somewhat like a queue whereas the events will be created one at a time.

The event handler is integrated in the Node entity. When a node receives data from other nodes, the Event Handler resolves the data, and determines what kind of event has arrived and handles what operation to do next.

How Events & Event Handler work



If Node A sends a file to Node B, a SendFileEvent is created. The SendFileEvent holds information such as the path, filename, data, etc;. Node B will receive the SendFileEvent, and pass it to the Event Handler to determine what operation needs to be done. Once, the EventHandler recognizes that the Event is a SendFileEvent, it will unpack the data, and create the files that were sent to Node B on that respective node.

File Controller

A part of the program that repeatedly scans local files in order to know what files are on the computer. Each node has its own file controller.

If a file/folder on the local node is added, updated or removed, the file controller would detect the change.

After the change is detected, it would trigger an event where the nodes would be signalled to either delete or send file(s).

File Controller: Directory

The File Controller also has a object type Directory variable called nodeRecordDir. This object is used to track the status of the files stored in the local node(/machine).

The Directory structure has two HashMaps: subDirs and dirFiles. The subDirs HashMap has the key that is the name of the sub-directory under the current Directory, and the value is the type of directory. The dirFiles HashMap has the key as the file name, and the value is the Custom Type of file.

We use the Directory to easily track the information of the local node. This structure allows for tracking both files, and folders containing files.

File Controller: syncFiles()

File Controller also has a HashMap object called clientsMap which holds the ID and IP of the nodes within the network.

FileController is where syncFiles() happens. In syncFiles, the comparison between the latest local file directory and the program's recorded directory is done.

If the local and recorded have the same files, they compare SHA256 and the timestamp of the same file, if one is different, the local is updated and the clientsMap sends the file to the other nodes as a broadcast event for update as well.

If the local doesn't have the file, the clientsMap is updated to tell the other nodes that the file has been deleted, and will broadcast the RemoveFileEvent for the other nodes to resolve within their own.

If the local has added files and the record doesn't have said files, the files will be added into the record, and the files will be sent to the other nodes through the SendFileEvent in clientsMap.

How Does The Client Discover Other Clients On The Network: Register, Sender, Receiver, ReceiverThread

The client can discover other clients on the network by using Register to to join the network using a User Datagram Protocol (UDP) connection.

Upon successful connection, a sender and receiver are created in order to send events and then receive the orders from one.

The ReceiverThread is the connection between the nodes where it will take in all events and send all the events to the event handler.

The Sender is used to send the events to the event handler, while the Receiver takes in event processes from the event handler.

DEMO