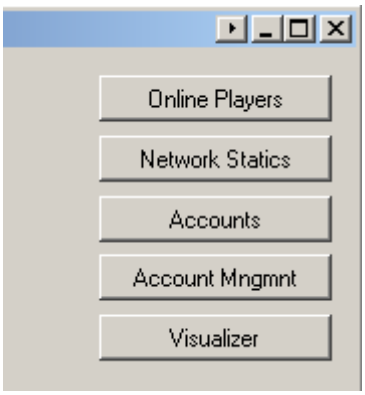
MECAC WEEKLY REPORT (March 25 – March 31)

This week we have worked on visualization of network traffic. The test clients have no graphics to decrease the memory usage. In order to verify the server network component, we needed to visualize the network traffic on server side.



The screen shot on the left is taken from server administration tool. Visualizer button is added to this panel. The visualizer is responsible for showing the locations of the clients their address information and the messages in between. The visualizer class is consumes the information from event matrix class. On each message push from one client to another visualizer is responsible for drawing an arrow in between.

Following properties of network visualize are important as they provide the infrastructure for the functionality of the class.

```
private Dictionary<String, Pair> clients = new Dictionary<string, Pair>();
```

Each client is identified by a string. The pair stores graphics objects associated with the a client. These graphics objects are windows forms classes.

```
public List<Pair> arrows = new List<Pair>();
```

Arrows is a public property that allows EventMatrix to pass connections between clients. On every event flush the EventMatrix clears the arrows of NetworkVisualizer and setups connections.

Our Network visualizer class has following functions to interface with the event matrix on server side:

```
public void NewClient(String identifier, Vector3 position)
```

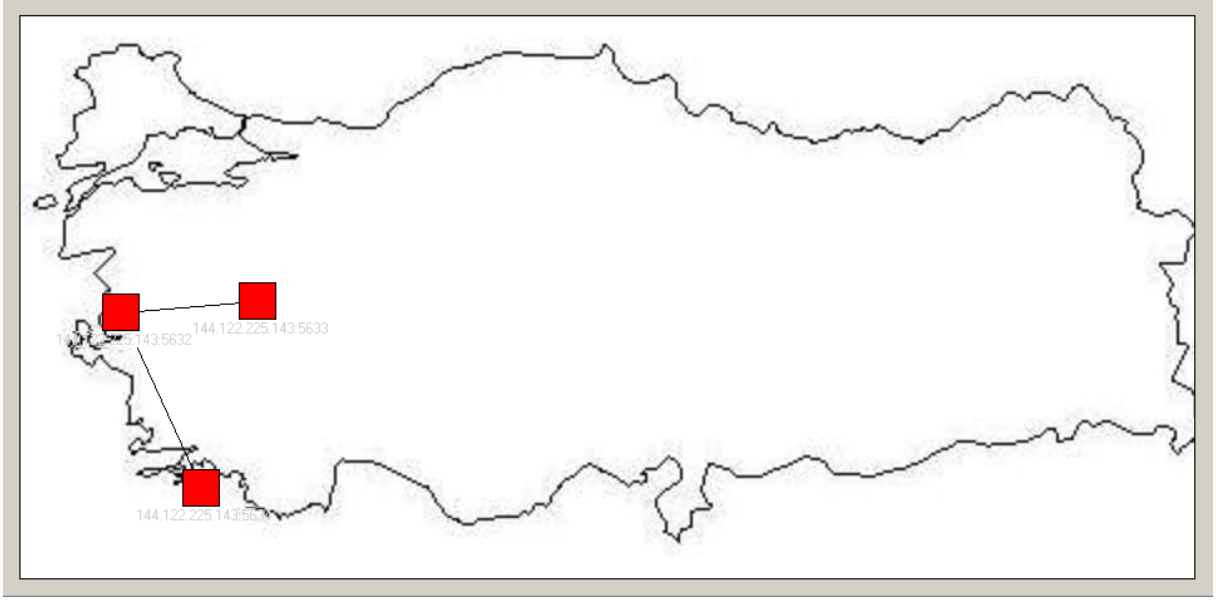
New client function adds a new client to canvas of the NetworkVisualizer. It takes a string identifier and vector position. String identifier is of the form ip:port. Although position is a 3D vector, our visualizer draws clients on 2D and discards Y information.

```
public void RemoveClient(String identifier)
```

This function is called when a client disconnects from the server. In order to find out which client has been removed, the method takes the identifier as an argument.

```
public void UpdateClient(String identifier, Vector3 position)
```

This function changes the position of a client in visualized canvas. Position argument replaces the new position of the client specified by the identifier.



The screenshot above is taken from network visualizer component. There were trivial problems in implementation of this component. Windows forms do not use double buffering and doesn't offer any easy solution to this problem inherently. We have implemented our custom double buffering panel. On each paint event we have drawn the background, clients, and arrows to a back buffer image, and switch images afterwards.

With this implementation of Network Visualizer component, we were able to simulate clients on the server side. The image above is taken when network bandwidth is 2 update messages. The simple area of interest calculation on EventMatrix class computes Euclidian difference between clients to score the messages urgency. As seen in the figure, the server sends update information of near-by clients when there is not enough bandwidth to send all updates to every client.

Next week, we will deal with following work items:

- Configuring SVN, and uploading the code base to CENG server.
- Building MSI installer for the client project, and uploading the installer to MECAC website.
- Increasing the number of test clients that can be run concurrently by reducing memory usage of a single client. We aim to discard unused graphics objects in test clients.