What is VNC?

Virtual Network Computing (VNC) is a graphic desktop-sharing system that users Remote Frame Buffer Protocol (RFB) to remotely control another computer (or in our case watch another computer). It transmits the keyboard and mouse events from one computer to another, relaying the graphical-screen updates back in the other direction, over a network (Wikipedia).

VNC is the name of a product and a trademark of RealVNC Ltd but the term ‘VNC’ got denoted as the informal name for this category of products/software.

RFB:

RFB is a protocol designed specifically for VNC systems is a simple protocol for remote access to graphical user interfaces that allows a client to view and control window system on another computer. It stands for “Remote Framebuffer” (StackOverflow).

Client & Server:

For client and server we will be referencing hep.phy (see citations) and form post found on StackOverflow. A VNC system consists of two parts, a client and a server. With the use of RFB, it acts as the protocol that governs the format used for passing data between the two. RFB is designed to make the client as simple as possible, so it is usually up to the server to perform any necessary translations.

Client:

* Lightweight
* Reliable transport (usually TCP/IP)
* A way of displaying pixels (either directly writing to the framebuffer, or going through a windowing system)

Server:

* More complex
* Must provide pixel data in the format the client wants
* The Windows server (WinVNC) is a little more difficult to create, because there are fewer places to insert hooks into the system to monitor display updates, and a less clearly-defined model of multiuser operation. Our current server simply mirrors the real display to a remote client, which means that the server is not 'multiuser'. It does, however, provide the primary user of a PC with remote access to their desktop.

API’s and Libraries:

VNC Developer API: <https://www.realvnc.com/en/developer/docs/latest/api/python/index.html>

Twisted Framework: [www.twistedmatrix.com](http://www.twistedmatrix.com) : Network Programming Framework

Pygame: Good graphics performance (Techtonik Github)

Project Idea:

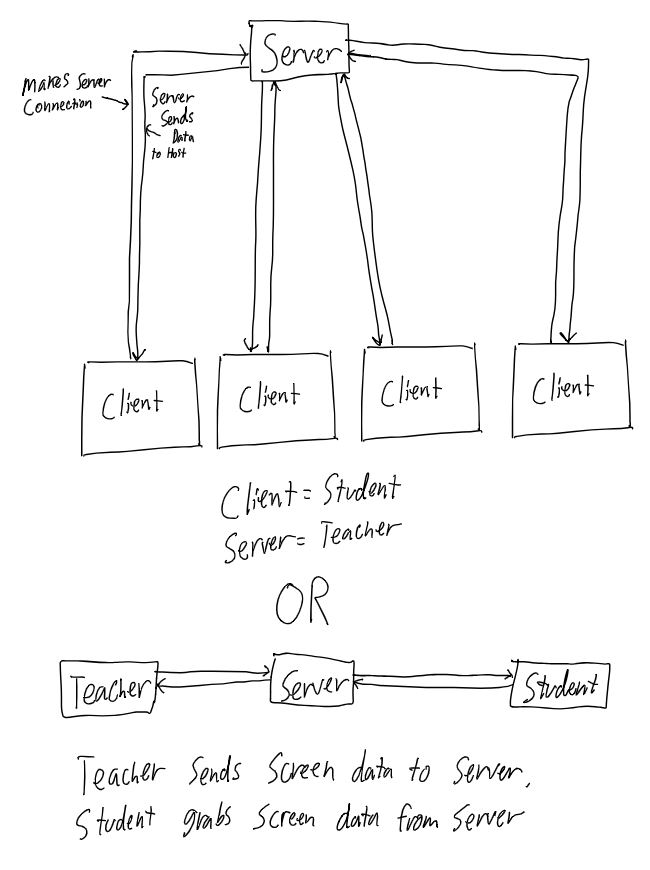
Implementation 1:

What is video? A digital video recorder takes 30 frames per second each of them a full scan of its sensor and without the use of compression or processing the raw data it is just a bunch of photos. With this knowledge, we can use Pythons MSS (multiple screenshots) library to first make a screen recorded that will record the screen of the teacher which will act as our ‘server’. Students or clients will then connect to the server if they wish to view the teachers screen from their computer.

Steps:

* + First we will build the screen recorder for the host wanting to share his/her screen. In more detail further in the report we will explain how we will do just that. To kick start the idea, we can use Python libraries to assist us with that.
    - Python Imaging Library (PIL): adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.
    - Multiple Screenshots (MSS): An ultra fast cross-platform multiple screenshots module in pure python using ctypes.
    - NumPy: used as an efficient multi-dimensional container of generic data. (Maybe)
    - Pygame: Free and Open Source python programming language library for making multimedia applications such as games, but not limited to games
  + Next we will need to work on compression as we will be sending this over the network. Finding a way to take the data recorded from the host and compressing that down to a reasonable size.
  + Next will be sending that data to the clients. Some sort of networking programming framework could be used such as the Twisted Framework or the use of sockets. Also explore RBF protocol and see what the advantage would be using it / using it in python.
  + After that setting up the client side application. The client will be running Pygame to display the information sent by the server and it will also need to decompress that data in order to read it.

This implementation is subject to change, but will provide a rough outline of what we can expect and need to accomplish. Breaking it down into pieces and then connecting the working pieces together will make the development more reasonable and efficient.



Citation:

<https://en.wikipedia.org/wiki/Virtual_Network_Computing>

<https://stackoverflow.com/questions/35731986/what-is-the-connection-between-rfb-protocol-and-vnc-protocol>

<https://github.com/techtonik/python-vnc-viewer>

<https://www.hep.phy.cam.ac.uk/vnc_docs/howitworks.html>

<https://stackoverflow.com/questions/48950962/screen-sharing-in-python>

<https://stackoverflow.com/questions/35097837/capture-video-data-from-screen-in-python>