**Introduction**

Traditional video streaming is based on the client-server architecture in which the central server will store all the videos and clients will download the video after connecting to the server. As the demand for large-scale video streaming increases, the capacity of servers is limited and cannot provide efficient service to a large amount of users. To tackle this, peer-to-peer networks are introduced in this model. In this model, the demand is distributed among all the peers and one peer can request any part of any file from other users. In order to achieve this, the peer first needs to know what the resources on the network are and who have the resources. This adds difficulty to the administration of the system, so a central server is needed to store the video list and the peer list.

**System Block Diagram**

Server side
- Listen for incoming client
- Accept client connection
- Send peer list and video list to client

Client side
- Set up connection to server
- Receive peer list and video list
- Receive peer and video list
- Get video segments from other peers
- Restore the segments and play the video

**Methodology**

**Part I: Set Up Network Connection**

Server side setup
- Create a socket
- Bind the socket to an address
- Listen for and accept a client
- Receive and send data
- Disconnect

Client side setup
- Create a socket
- Connect to the server
- Send and receive data
- Disconnect

**Part II: Video Segmentation**

The format of the video is ASF (WMV), which consists of header object, data object, and data packets. Peers need to get segments from different peers and restore video by time stamp. We need to know the length of header object, the length and position of data object, and the size of each packet.

**Part III: Video Streaming**

Sending a Frame
- Main thread responsible for video compression
- Sending video frames
- Receive data
- Transmit empty frames

Receiving a Frame
- Look for the start of a frame
- Read the size and other features of a frame
- Get the main video data

**Conclusion and Further Development**

The system is designed using C++ in Win32 platform. It consists of the server side and the peer side. The server side provides lists to peers and then the peer gets video segments from other peers to increase the download rate of the whole system. A user interface is created for the user to use the functions and the media player is also embedded in the user interface.

Up to now the peer can only act as either server or client, but in real-world applications, after a P2P program starts to run, it serves as both client and server, downloading video for the current client while uploading what it has for other peers. Some system exist has a policy that the more you contribute in uploading, the faster your downloading speed will be. So further improvement can be made by combining the server and peer function into one program, and every peer is making use of what is contributed to the system. This will be more efficient.