CSE471: DATA COMMUNICATIONS AND COMPUTER NETWORKS

YEDITEPE UNIVERSITY

FALL 2025

TERM PROJECT – DUE DATE JANUARY 5TH, 2026

As a term project, this semester you are expected to develop a peer-to-peer (P2P) video streaming application that will operate without a need for a centralized server. P2P nodes should be able to UDP flood the network (Fig. 1) to locate other overlay network users. However, your application should not overwhelm the network and bring on to a crash state. To prevent this, you should consider capping the number of packets by implementing limited scope flooding. Any of the nodes should be able to share as many videos as s/he wants. On the other hand, any peer should be able to request and stream a video's chucks from multiple sources.

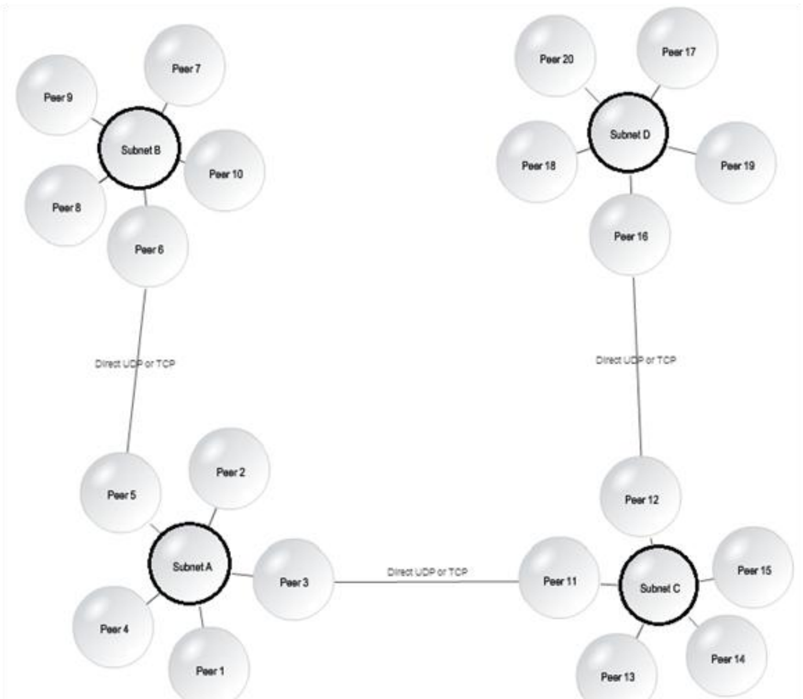
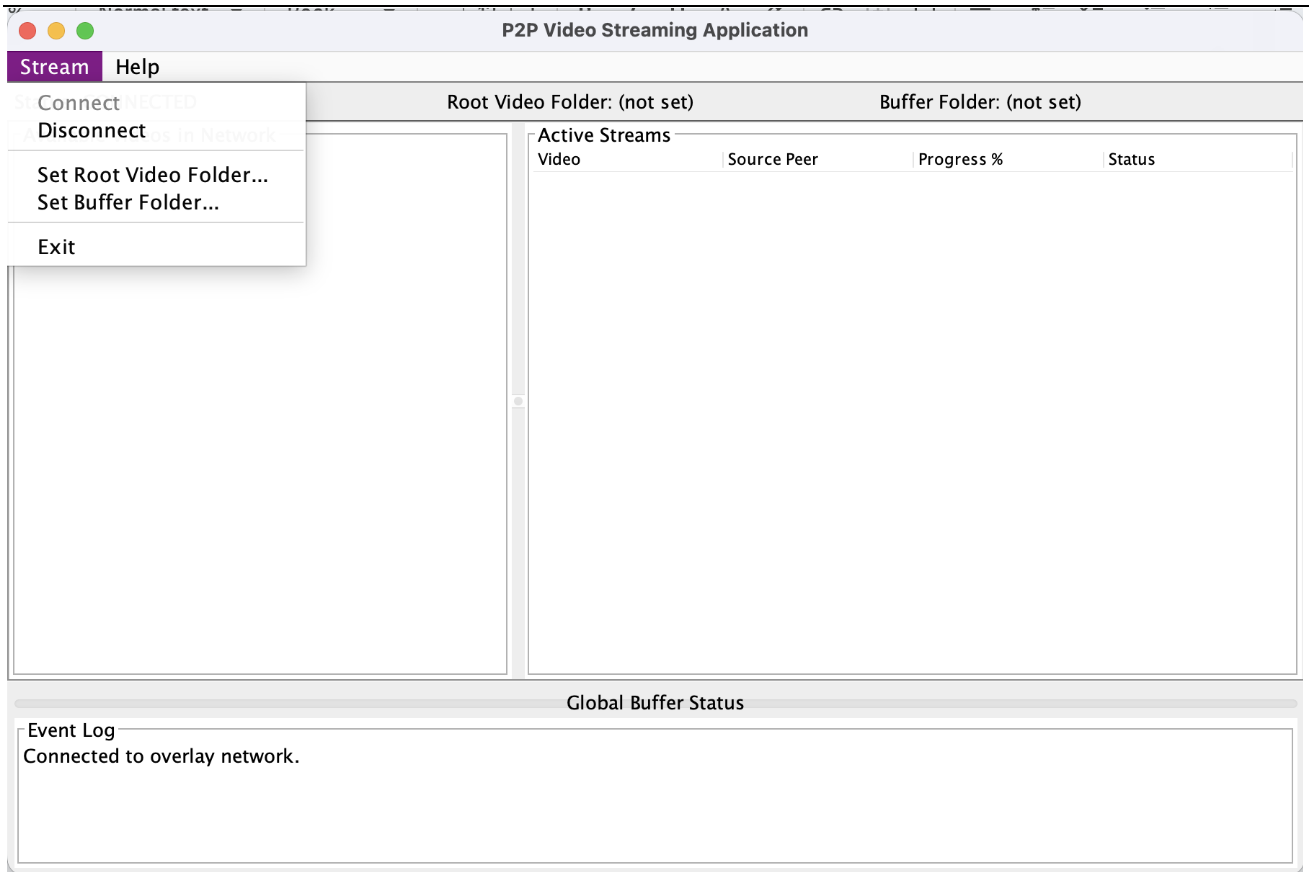


Fig. 1: Peer-to-Peer (P2P) Video Streaming System Overlay Network

Instead of downloading a full file beforehand, the playback should start as soon as the required number of chunks of the video data is buffered. The video chuck exchange could be facilitated using TCP or UDP depending on your protocol design. Accordingly, your application should be able to discover other peers, request to download a specific file from multiple source nodes and play the video while the data transfer is in progress.

For usability purposes, you are also required to implement a graphical user interface. Accordingly, your application should provide the following (Fig. 2):

- Your application must contain two menu bars - namely ***Stream*** and ***Help***.
- The Network menu should have a menu item called **Connect** (*Stream->Connect*) and ***Disconnect*** (*Stream->Disconnect*), which will respectively enable the application to connect to the overlay network and disconnect from it.
- The Network menu should also contain ***Set Root Video Folder*** (*Stream->Set Root Video Folder*) and ***Set Buffer Folder*** (*Stream->Set Buffer Folder*) where the first one would set the folder which be shared with the overlay network users and the second would set the download destination folder for the file that is being streamed.
- On the other hand, The ***Help*** menu should contain information about the developer of the application.



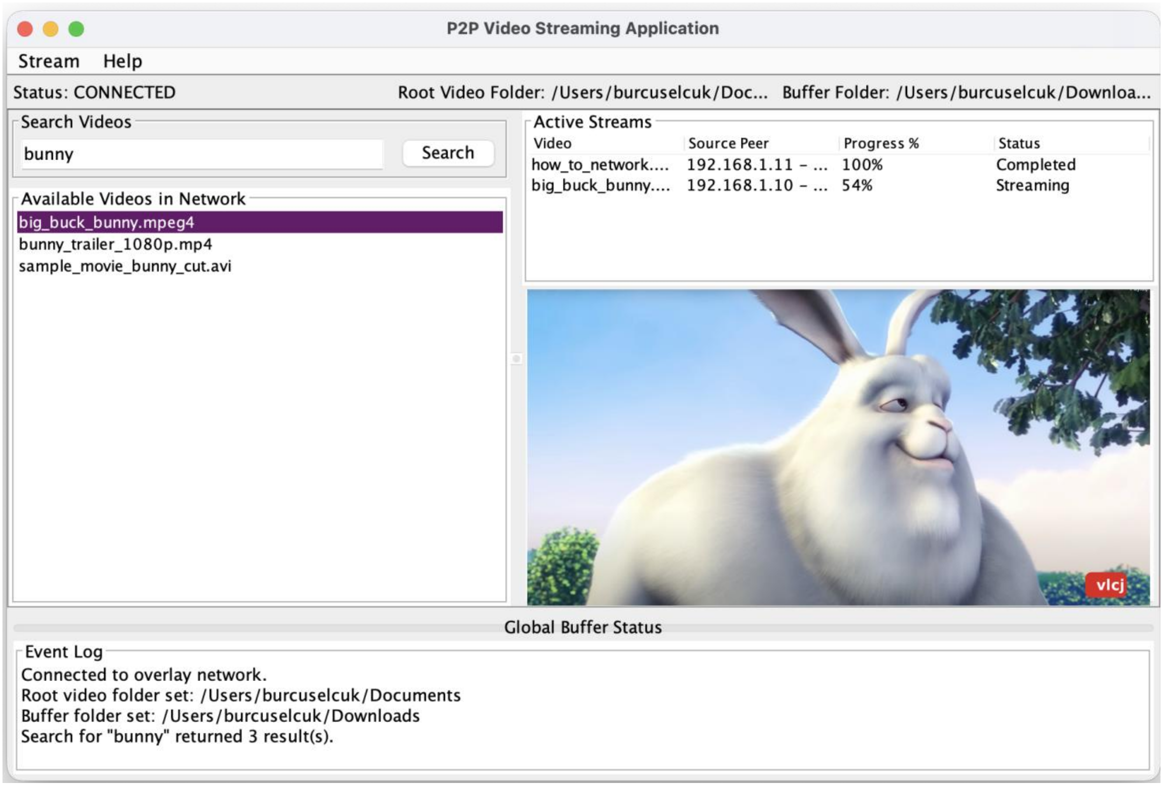
```
Connected to overlay network.
```

Fig. 2: P2P Streaming Application screenshot with menus

Additionally, your application design should have the following information GUI elements:

- A text box with a search button which should enable users to search for file name
- A list box entitled ***Available Videos on Network***, which will be listing the files found on the network according to the search criteria.

- Another list box entitled **Active Streams** that would list active streams and peers that the current video is being downloaded from - containing information about source IP, chunk number and percentage and status.
- A video player that can be used popped up to full screen but also can be used as part of the main window.
- An indicator for the overall buffer status - namely Global Buffer Status.



Status: CONNECTED   Root Video Folder: /Users/burcuselcuk/Doc...   Buffer Folder: /Users/burcuselcuk/Downloa...

big_buck_bunny.mpeg4
bunny_trailer_1080p.mp4
sample_movie_bunny_cut.avi

Video	Source Peer	Progress %	Status
how_to_network....	192.168.1.11 – ...	100%	Completed
big_buck_bunny....	192.168.1.10 – ...	54%	Streaming

```
Connected to overlay network.
Root video folder set: /Users/burcuselcuk/Documents
Buffer folder set: /Users/burcuselcuk/Downloads
Search for "bunny" returned 3 result(s).
```

Fig. 3: Screenshot with video streaming

From the functionality point of view your streaming application should be able to support the following:

- The application should operate using a fully **P2P model**.
- Peers must use a **limited-scope UDP flooding** method to announce themselves and detect other active peers in the network.
- Your application should avoid overwhelming the network; appropriate **TTL (time-to-live)** or hop-count limitations should be implemented.
- After discovery, peers maintain a list of available nodes and their shared video catalogs.
- Video content must be transmitted in **fixed-size chunks of 256 KB**.
- The application should support **out-of-order streaming**, meaning video chunks may arrive in any order from multiple sources/nodes.
- Your system should be able to (a) Request missing chunks; (b) Detect duplicate chunks; (c) Reassemble all chunks into a playable video file (d) Allow **playback while downloading** (buffering mechanism).

- The system must distinguish identical video files with different filenames using a hashing mechanism (similar to real torrent systems).
- You do not need to implement your own video players (see Fig. 3) as it is out of our networking scope. So you can use existing Java libraries such as ***VLCj*** (https://github.com/caprica/vlcj) and ***Video4j*** (https://github.com/metaloom/video4j)
- [**BONUS**] Exclusion filters - your application will allow user to filter filenames, folders, file masks/extensions
- [**BONUS**] Implement a dynamic buffering strategy that increases or decreases buffer size based on packet loss or peer latency.
- [BONUS] You can implement your video streaming application and network using Docker and Docker Compose.

You should export your Eclipse project folder (for other IDEs please see how you can export) and submit your term project as a single zip file through the YULEARN (https://yulearn.yeditepe.edu.tr/) latest by the end of Monday, January 5th, 2026.