P3 - NETWORKED PRODUCER AND CONSUMER

S12 group discm buddies

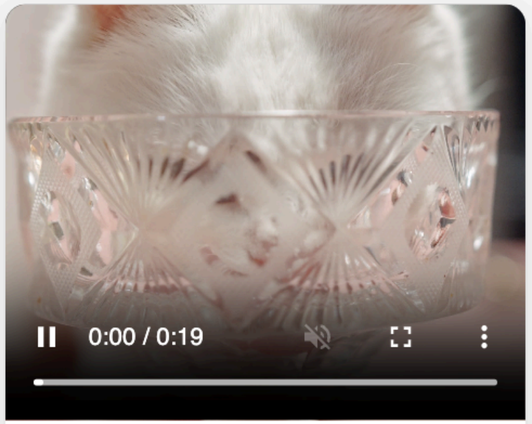
BERNARDO, ERICA MONTEMAYOR
ATIENZA, MARIELLE ANGELENE
RIVERA, JOSE CARLOS IGNACIO
TIGHE, KAITLYN PATRICIA

📤 Upload a Video

Select an MP4 file below and upload it to preview.

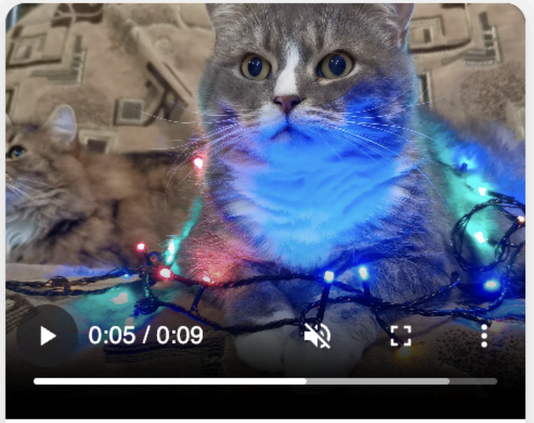
Choose File | No file chosen | Upload

🎞️ Uploaded Videos



video_174[illegible]331471.mp4

🔍 Previewing...



video_1744434341243730138.mp4

Docker Architecture - Containers

Web App	Producer	Consumer
• Serves the frontend via HTTP. • Lets users upload and view video previews. • Mounted to web/uploads for live access to uploaded files.	• Reads video files from shared folders. • Compresses and sends videos to the consumer using TCP sockets. • Multiple producer threads enable concurrent uploads.	• Listens for incoming TCP connections. • Saves uploaded videos to web/uploads. • Uses multithreading with a queue to manage simultaneous uploads.

- All containers are connected to a shared **Docker network (media-net).**
- Producers and consumer communicate via TCP, while the web app uses HTTP for uploads and viewing.

Key Implementation Steps

Producer (sender)

- Reads video files from multiple folders using threads.
- Each thread establishes a TCP socket connection to the consumer.
- Sends video data in chunks and shuts down socket after transmission.

Consumer (receiver)

- Starts a TCP server that listens on port 8080.
- Accepts incoming socket connections and pushes them to a thread-safe queue.
- Consumer threads dequeue sockets and write incoming data to .mp4 files.

Queueing Details

- Custom **thread-safe queue** (queue.h) handles incoming connections safely.
- Implements a **leaky bucket model:** if the queue is full, connections are dropped and producer is notified.
- Ensures **fair load distribution** and avoids system overload during high traffic.

Breakdown of Producer and Consumer Concepts

Producer

- Each producer thread reads all video files from a specific folder (shared/videosX).
- Sends videos one at a time in 1MB chunks over a TCP socket.
- Automatically skips duplicates and retries on queue full.

Consumer

- A multi-threaded consumer using a thread-safe queue.
- Listens on port 8080 and processes incoming sockets.
- Each thread dequeues and writes video to web/uploads.

Producer-Consumer

- Producers and consumer communicate via TCP inside a shared Docker network.
- Uploads are parallelized, and the queue ensures controlled concurrency (leaky bucket).
- Supports retry + duplicate detection + compression simulation.
-

Synchronization Mechanisms Used

Thread-safe Queue	Ensures only one thread accesses the queue at a time (avoids race conditions).
Socket handling	Proper use of shutdown() and close() to end communication safely.
Thread joins	Ensures all producer/consumer threads complete execution before program ends.