Real-Time Video Streaming Server with GStreamer and REST API

Objective

Develop a comprehensive real-time video streaming application that utilizes GStreamer for handling video capture and streaming, with a REST API for external control. Integrate video compression to enhance stream quality and network efficiency. Include MQTT as an optional component for advanced users interested in exploring messaging systems for synchronization and control.

Overview

The project aims to simulate a real-world media streaming service. It will feature a server capable of streaming compressed video content to clients using modern protocols and technologies. The service will be controlled remotely via a REST API, allowing dynamic interaction during operation. MQTT can optionally be used for enhanced control and synchronization.

System Architecture

Key Components:
- Video Source: Captures real-time video from a webcam or file.
- Streaming Server: Manages video capture, processing, and distribution.
- REST API Server: Provides an interface for external control commands.
- Client Application: Receives and displays the video stream.
- Optional: MQTT Broker for advanced messaging and synchronization.

Key Components

1. GStreamer Pipeline

Constructs a pipeline to handle the encoding and streaming of video. This includes setting up video capture (from a webcam for example), applying compression techniques, and preparing the video for network transmission.

2. REST API

Provides endpoints for starting/stopping the stream, changing settings, and retrieving system status, allowing for external control and monitoring of the streaming service.

3. Optional: MQTT System

Implement a messaging system to handle playback controls and system monitoring, ensuring all components are synchronized and responsive to user inputs and system commands.

4. Client Implementation

Develops a simple application that connects to the streaming server, displaying the streamed video and responding to control messages relayed via the REST API, with optional MQTT support.

Expected Outputs

The project will result in a fully functional video streaming service with the following capabilities:
- Real-time video capture and streaming (from webcam).
- **Optional** Video compression to optimize bandwidth.
- External control via REST API (start, stop).
- **Optional** advanced control via MQTT.
- Client-side video reception and display.

Tools and Technologies that you may use

- **GStreamer**: For video capturing and streaming. Download and install GStreamer from its official website.
- **Python**: For writing the server and client logic. Install Python and pip (Python package installer).
- **Flask or FastAPI**: For creating the REST API. Install using pip.
- **MQTT (optional)**: Install an MQTT broker like Mosquitto and the Paho MQTT Python client.

Deliverables

Comprehensive source code and documentation on setup and usage.

A demonstration video showcasing the system in operation will have a bonus point.

Implementing the optional parts (MQTT and video compression) will have a bonus point.

There will be bonus point if you apply a person detection (like an object detector like YOLO) for each frame and show in the client side.

Duration: 25 days