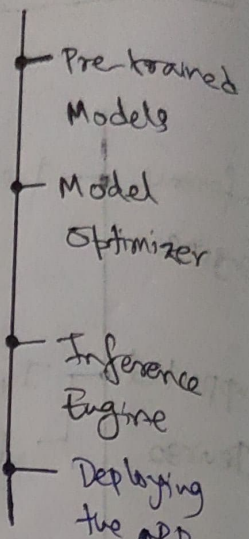
(5) Deploying an Edge App



Deploying an app involves:

- handling i/p streams (OpenCV)
- Processing model outputs

and more.

Handling Input Streams

Steps:

1. Handle the cmd-line args in case of image / video inputs.
   - In case of image inputs, handle for webcam / video paths.
2. Get the video capture from cmd-line args, and open it using OpenCV. Also, define the output video if video is input.
3. Repeat steps (4) through (6) until the capture is opened.

④ Resize the video to a suitable size, detect edges using Canny algorithm, and make a 3-channel image of the frame read.

⑤ Watch out for key presses. For example, if ESC key is pressed, break the loop.

⑥ Write out the frame depending on whether it is an image or a video.

⑦ Release all resources before closing the captured object.

Handling Output Streams

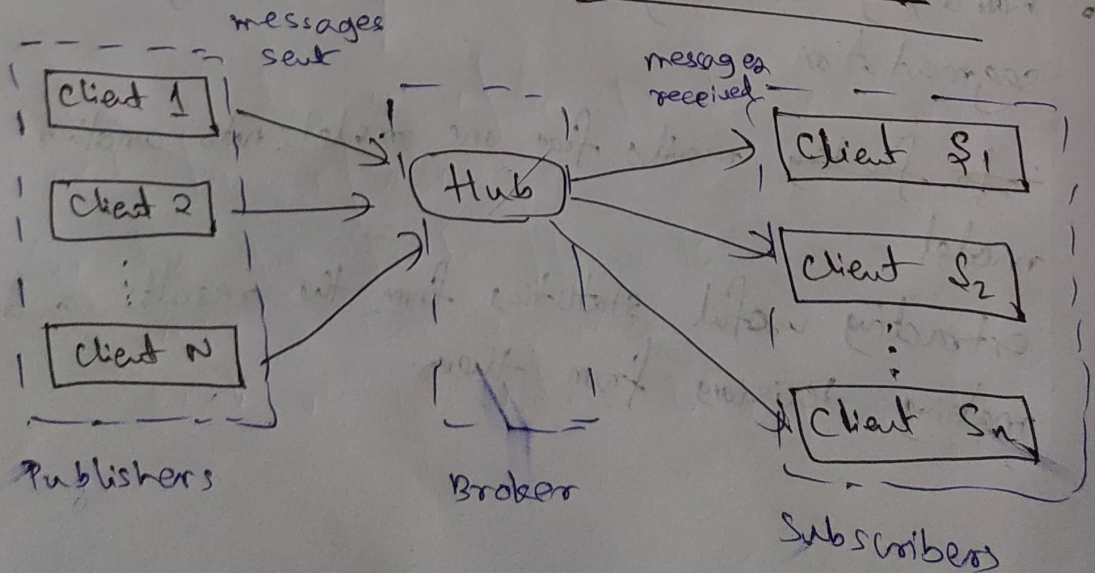
This involves:

- handling bounding boxes as a result of semantic segmentation;
- ~~info~~ feeding results from one model into another model
- extracting useful statistics from the results and making decisions from them.

MQTT

- Stands for MQ Telemetry Transport, where:
  - MQ comes from Message Queues
  - But, MQTT does not use queues.
- It is a lightweight publish-subscribe architecture.
- It is designed for:
  - resource-constrained devices,
  - low-bandwidth setups.
- It is used for M-2-M communication like IoT
- Port: 1883 is reserved for MQTT.

Publish-Subscribe ~~Architectures~~ Architectures



- In this architecture, there are:
  i) Publishers: Clients that send messages to a topic.
  ii) Subscribers: Clients that subscribe to the topic so as to receive its messages.
  iii) Broker: Handles passing of messages on the topic to its subscribers.

- In this way, the clients receive the messages they subscribe to without having to know each other.

- Examples of publish-subscribe architecture:
  - MQTT: publish info like the count of bounding boxes (lightweight) but NOT the actual video frames (heavy weight)
  - Self-driving cars: Used in Robot Operating System (ROS).

- paho-mqtt is a library in Python for working w/ MQTT.

Ffmpeg

- It is a library used for video streaming.
- The command-line tool ffserver will be used for this.
- The sys library in Python is used to send frames to a given server.
  - sys.stdout.buffer.write(frame)
  - sys.stdout.flush()

  } code for sending frames

- In one terminal, run the config server;
  sudo ffserver -f ./ffmpeg/server.conf
- In another terminal, send the frame to the above server.
  python app.py -m model.xml | ffmpeg -<options>

End-User Needs

- Their needs can inform app design & models used for inference.
- Consider the trade-offs with:
  - more or less resources
  - high v/s low network latency envs.

Summary

- **Deploying an Edge app**
  - Handling input streams (OpenCV)
  - Processing model outputs
- **MQTT**
  - Message Queues Telemetry Transport
  - Lightweight publish-subscribe arch. for M-2-M comm
  - Port 1883 used for this
- **Publish-Subscribe Arch.**
  - Publisher → posts to topics
  - Subscriber → receives from topics subscribed
  - Broker → Handles this transmission
- **FFMPEG**
  - Tool for video streaming
- **Misc. Edge Topics**
  - Analyzing Performance
  - Model Use Cases
  - End User Needs