Intruder Detection - Distributed Systems

Alan Gallo, Cedric Sillaber, Frantisek Sova

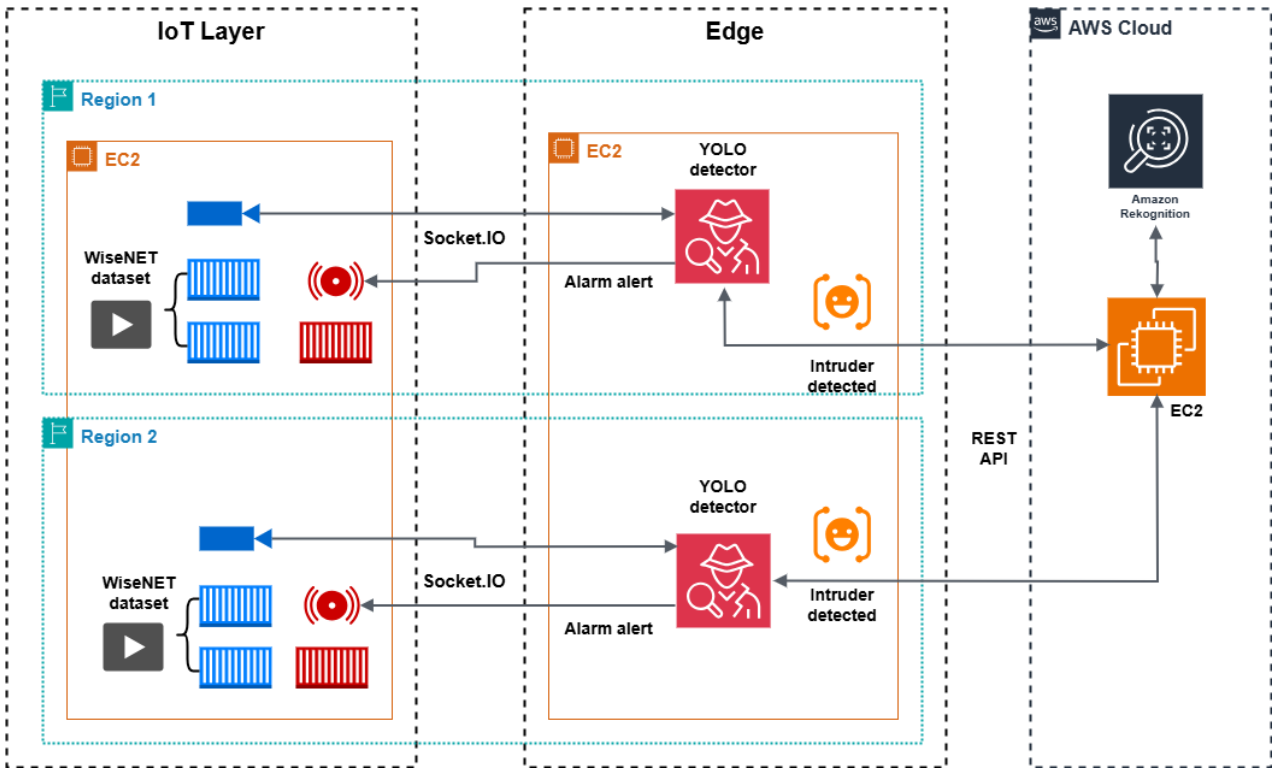
January 2025

Content

- System architecture
- Implementation details
- Demo
- Evaluation

System architecture

Final system architecture:



IoT Implementation Details

Video files stored on iot, opened with opencv

Frame Processing

- extract 1 frame per second from 30 fps video stream
- main bottle neck!

Communication using Socket.IO

- websockets, built ontop of application layer
- Good for continuous dataflow, persistent connection
- Setup once, then just send data

Edge Implementation Details

Async Processing Pipeline

- two workers working asynchronously:
  - worker 1: frame buffer queue
  - worker 2: process buffer: YOLO person detection, cloud communication

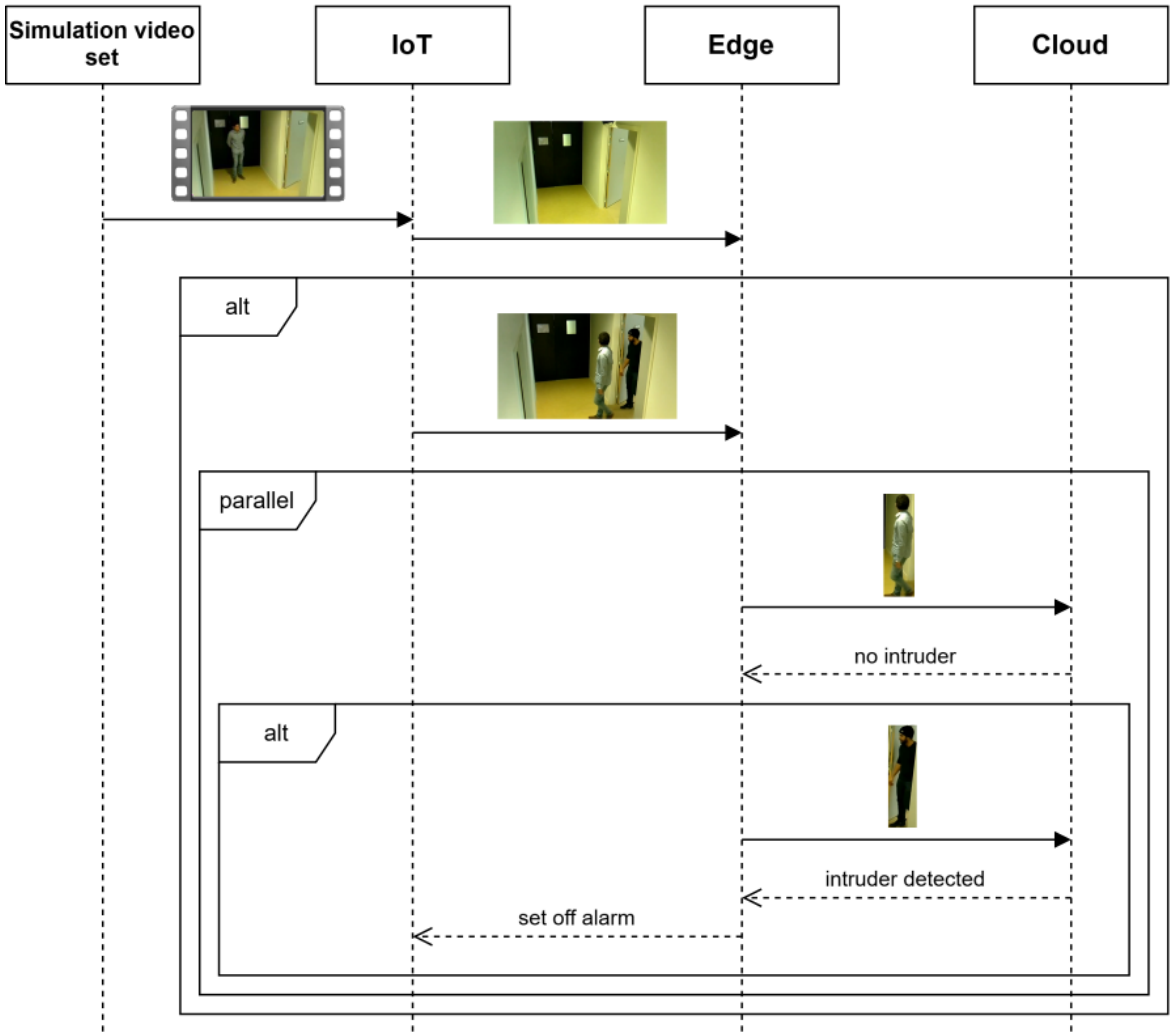
Communication: Flask REST

- REST API for communication between edge and cloud
- Edge sends HTTP request to Cloud $\rightarrow$ Cloud sends HTTP response

Cloud Implementation Details

- REST API endpoint
- Wrapper for AWS Rekognition

Example Workflow/ Controlgraph



Evaluation

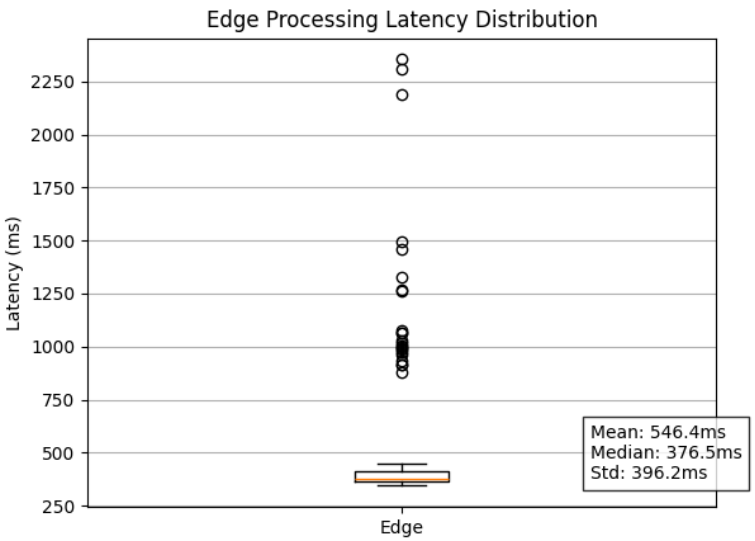
Experiment Configuration:

- *set3* from WiseNET dataset
- 2 edge devices
- 2 cams + alarm per edge
- cloud instance

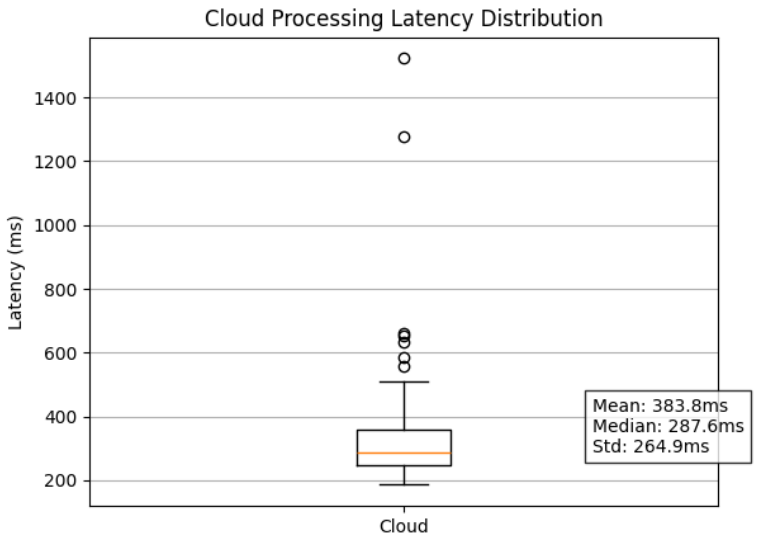
Metrics:

- Edge processing latency (yolo processing)
- Cloud latency (AWS Rekognition)
- Round-trip time from camera to alarm

Evaluation - Edge & Cloud Latencies



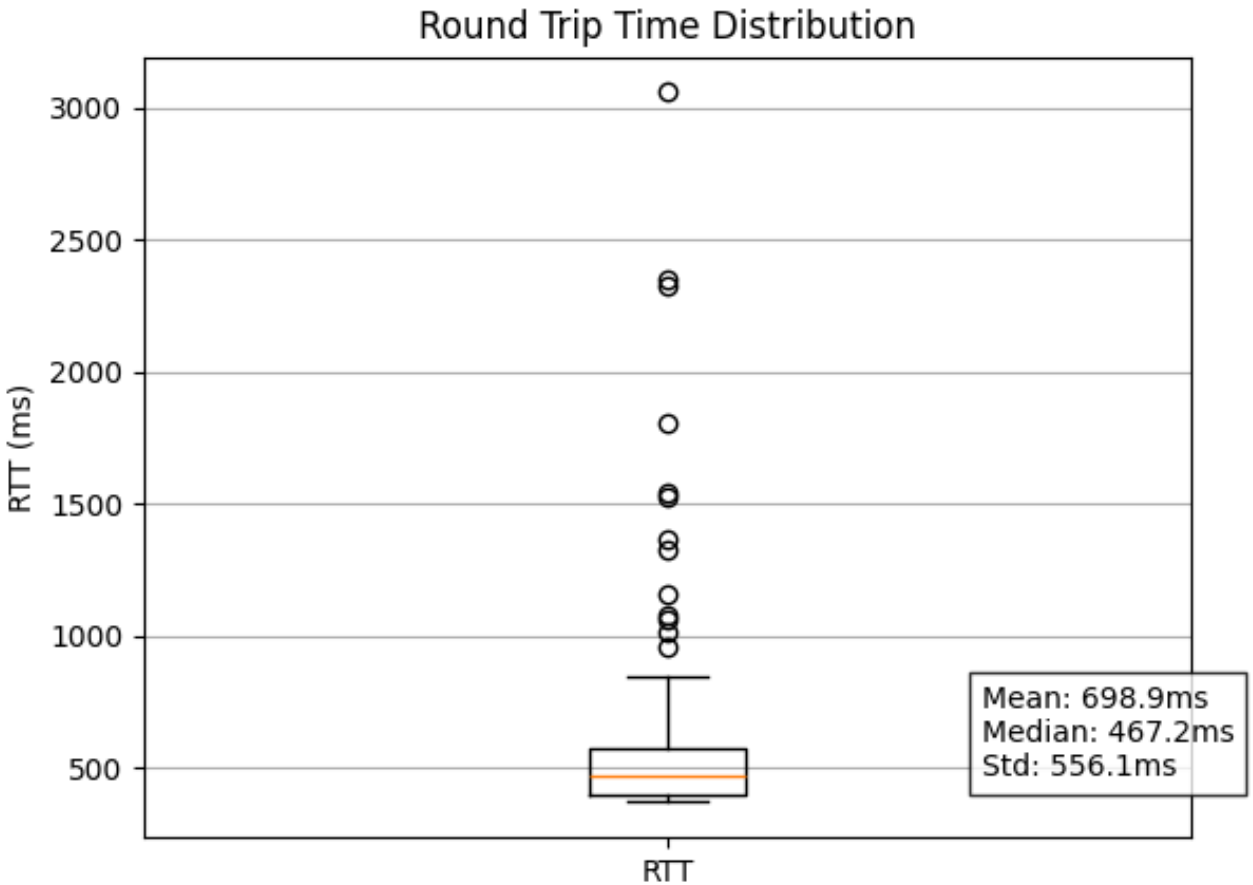
YOLO person detection latency



Cloud latency (AWS Rekognition)

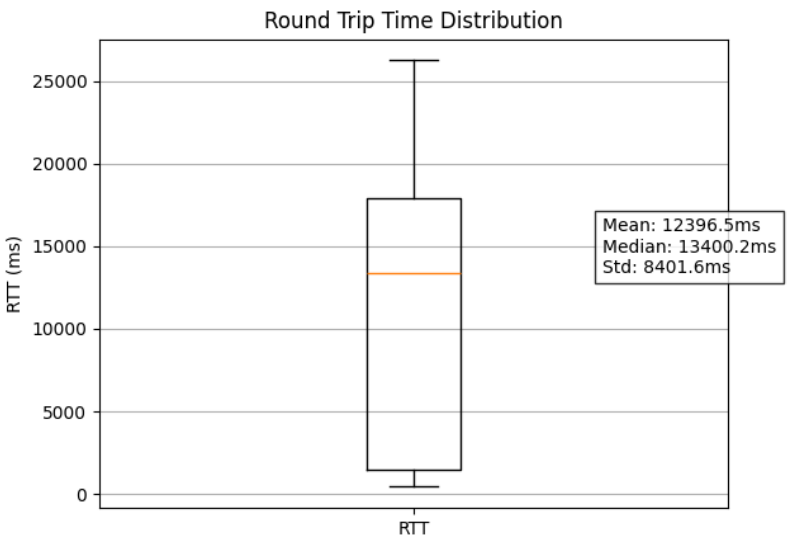
Evaluation - Round Trip Time

Round-trip time: camera frame to alarm.

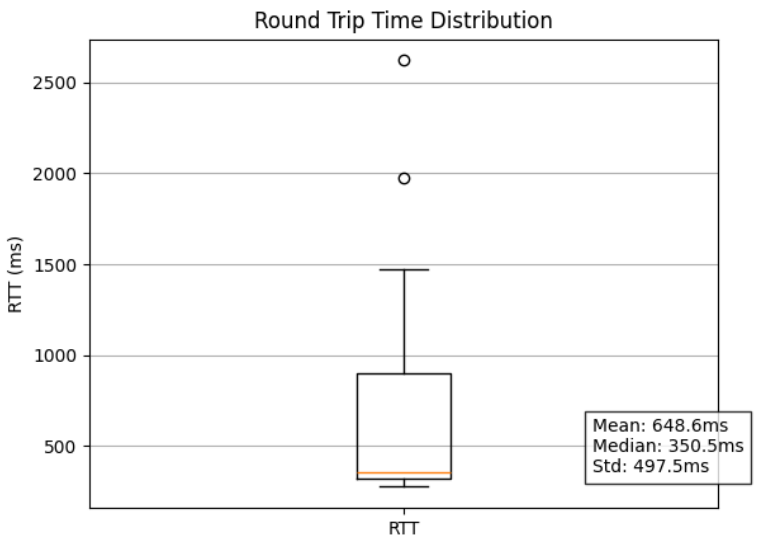


Evaluation - System Bottleneck

- System bottleneck on edge
- New frames / second $>$ frame processing rate $\rightarrow$ very high latency



RTT for 2 frames per second sent



RTT for 0.5 frames per second sent