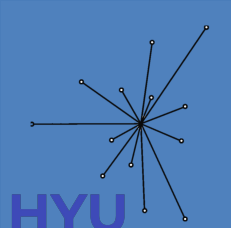



HYU IoT Lab

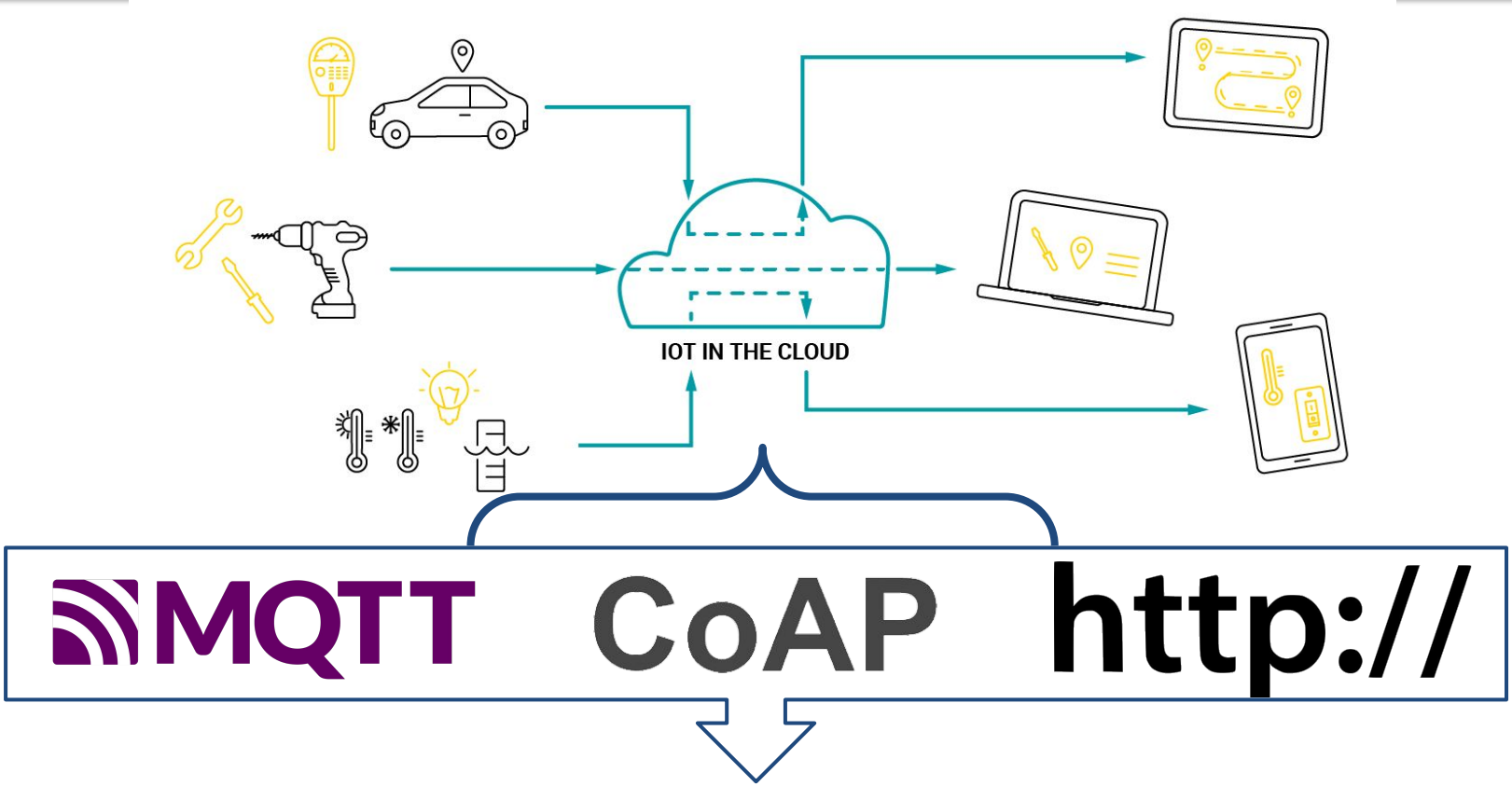
# Reliable Event Detection Using Time-Synchronized IoT Platforms

***Byeong-gil Jun***, *Dongha Kim, Marten Lohstroh, and Hokeun Kim*

**HYU IoT Lab & LF Project Team**

*Time-Centric Reactive Software*
*San Antonio, TX*
*May 9, 2023*

# Introduction

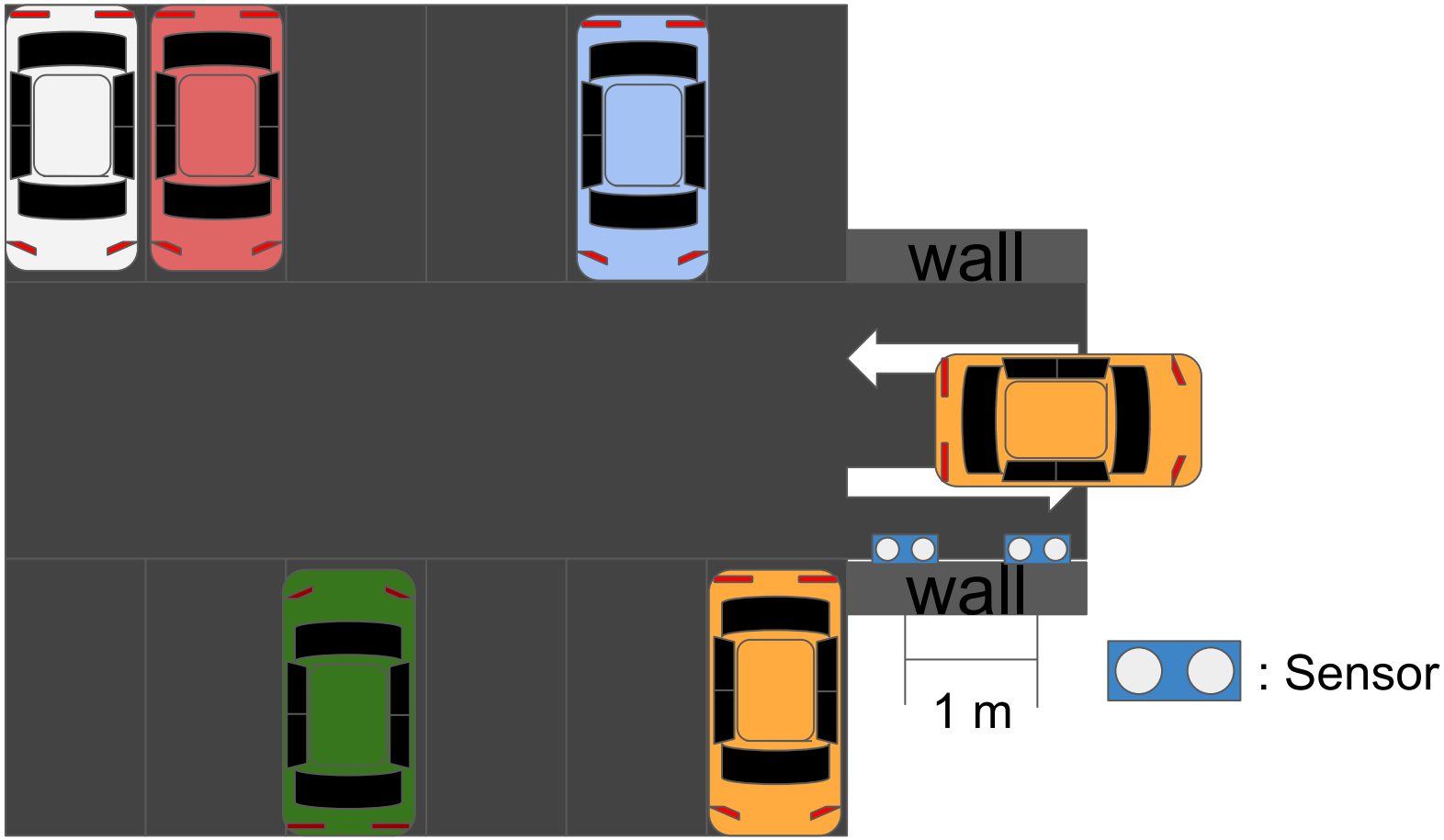



**Order of events?**

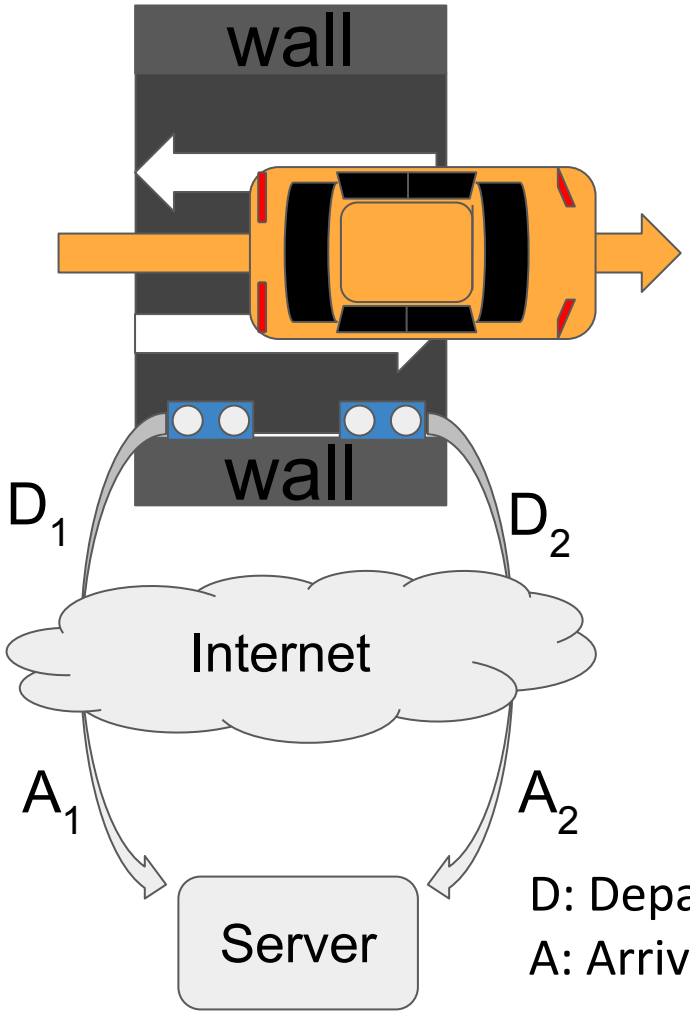
Image source: https://www.avsystem.com/blog/iot-cloud-platform/ https://mqtt.org/mqtt-specification/ https://docs.loriot.io/display/LNS/CoAP+Push https://www.onlinewebfonts.com/icon/139564

# Motivation - Practical Example

- The Parking Garage Occupancy Monitoring



# Motivation - Naive Approach

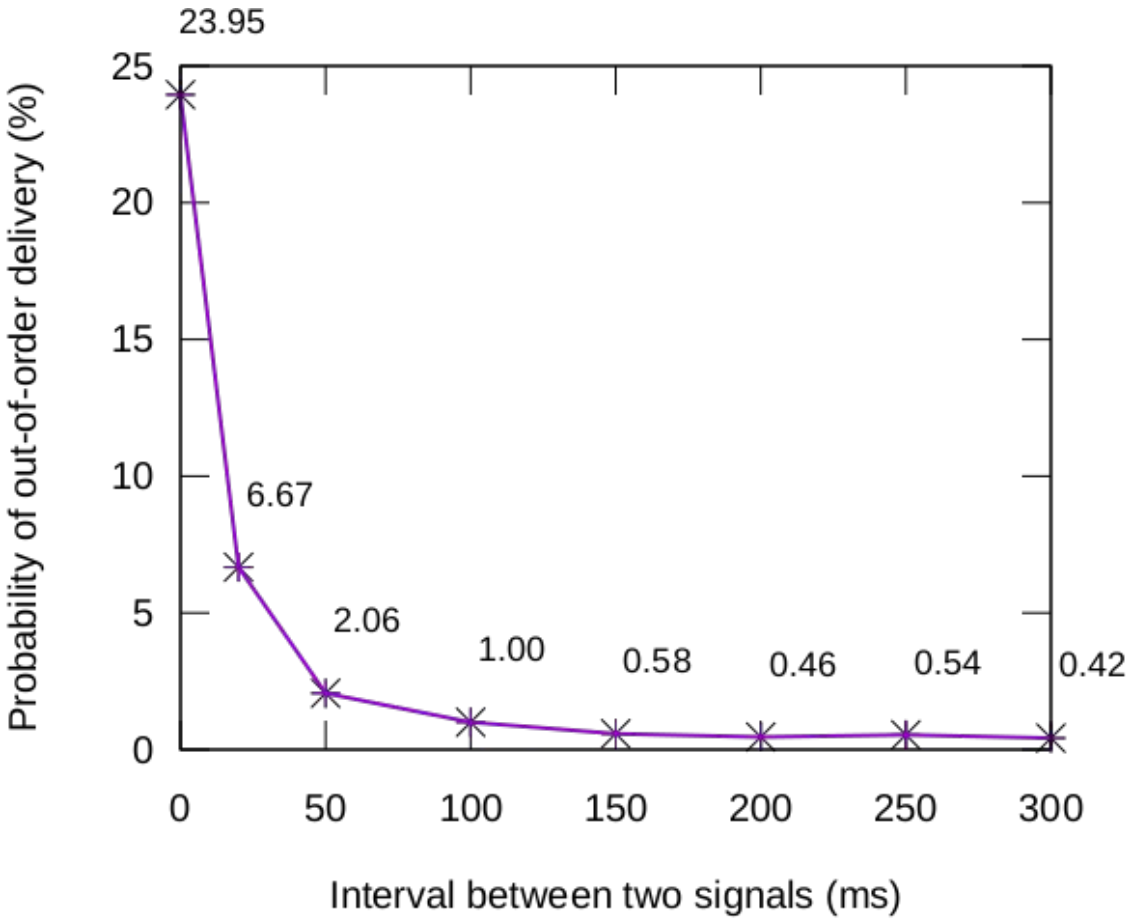



- Using Internet to report the event
- It is all about the **order**!!
- Will this work with AWS?

D: Departure Time of a signal
A: Arrival Time of a signal

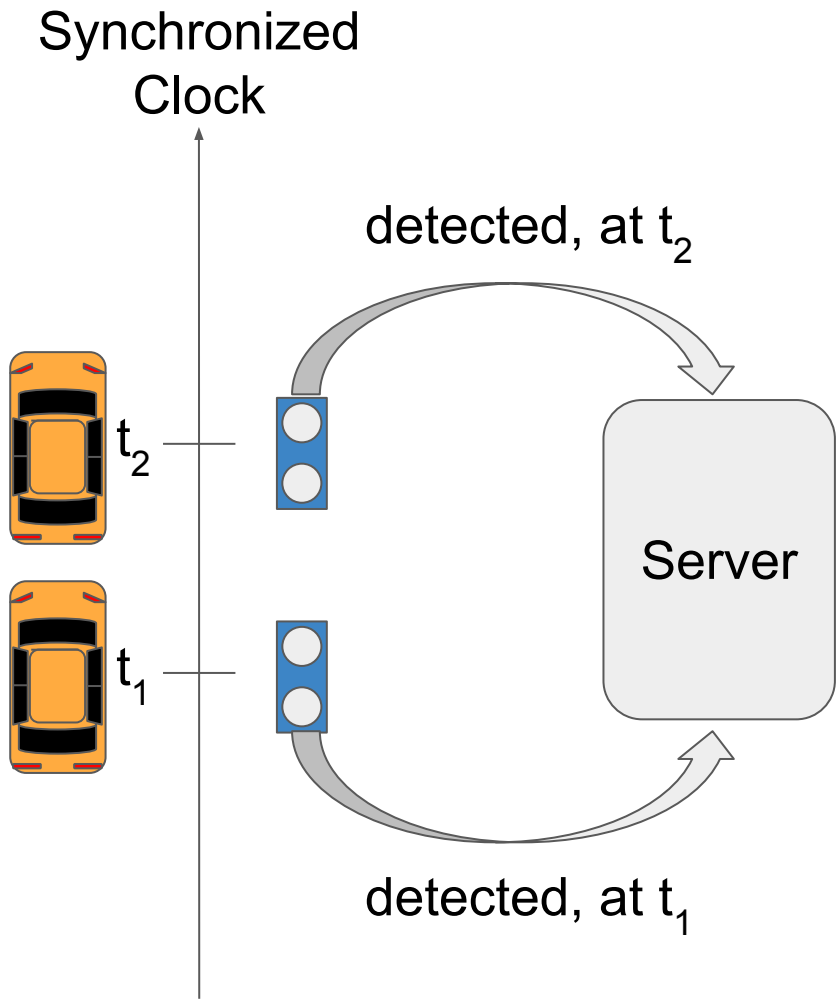
# Motivation - AWS DynamoDB

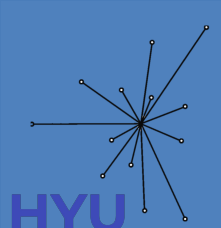
- The implementation with AWS (SToA) is not enough!
  - Using Raspberry pi 4 board to send signals
  - Endpoint is AWS DynamoDB



# Key Ideas of Our Solution

- Clock synchronization
- Tag-based ordering

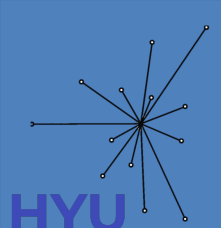





# Background

- Reactor Model
  - Ordering events using a logical timeline and a synchronized clock
  - Can be easily applied to the distributed environment

- Lingua Franca
  - A coordination language for reactor programs
  - Supporting C, C++, Python, Rust, and TypeScript

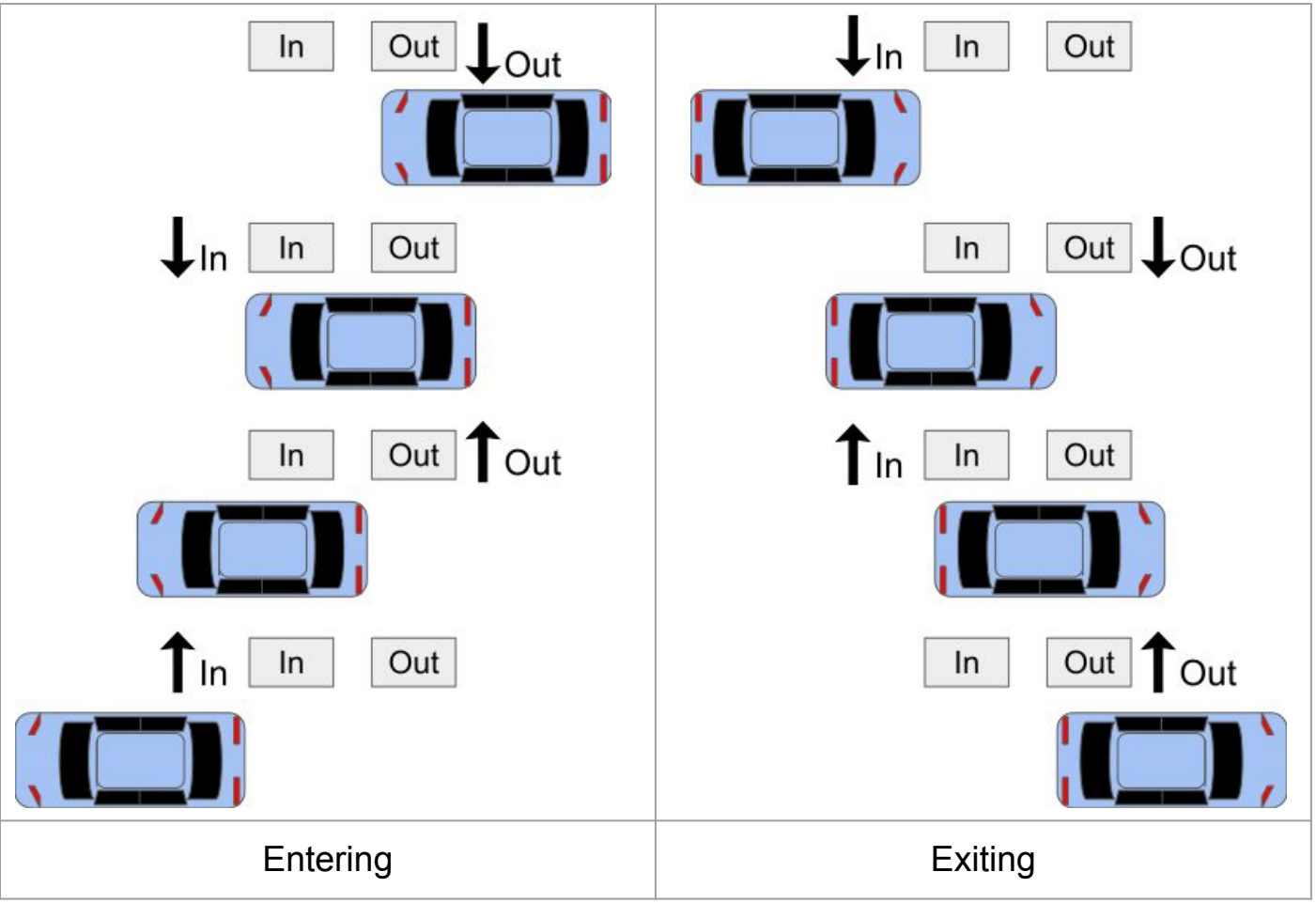



# Background (cont'd)

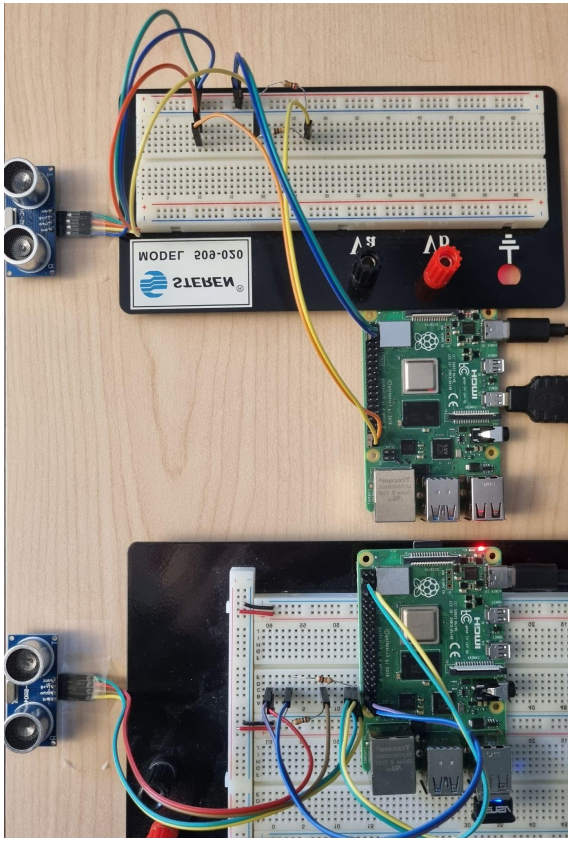
- Federated (distributed) execution
  - Lingua Franca supports federation of multiple reactor programs
  - Clocks are synchronized using the Runtime InfraStructure (RTI)
  - TypeScript target of LF only provides the centralized coordination

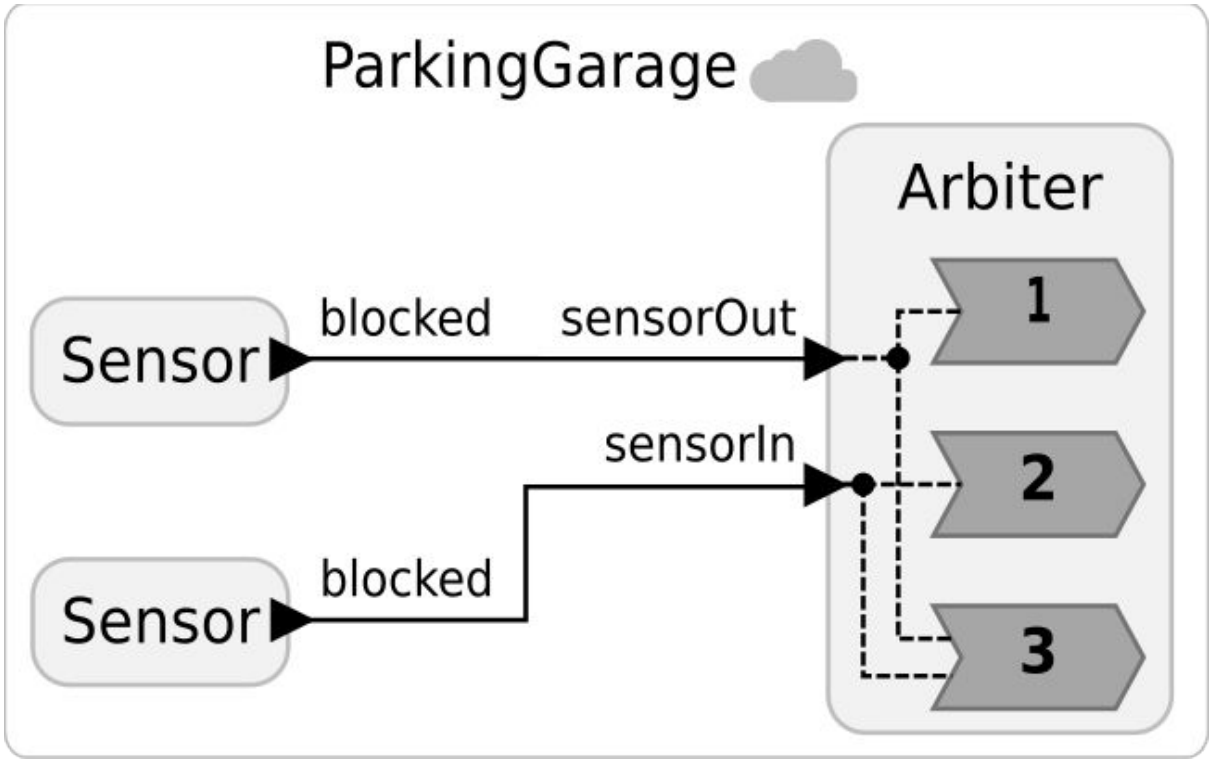
# Implementation of Our Solution using sensors

- How to specify entering or exiting



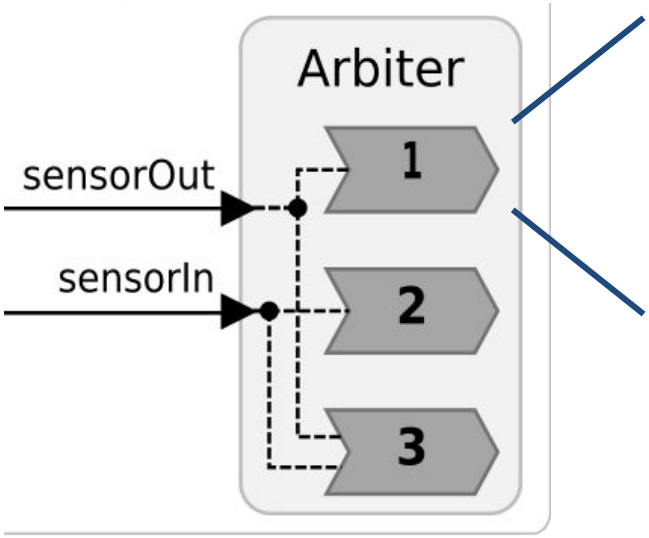
# Lingua Franca Implementation of Our Solution





# Lingua Franca Implementation of Our Solution (cont'd)

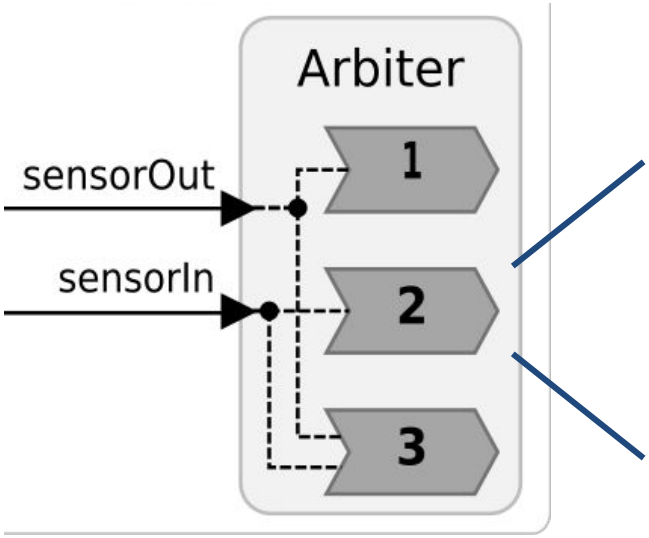
- Arbitration Logic




```
reaction(sensorOut) {=
  if (sensorOut)
    // Store the time that an object appears at the outside sensor
    outsideAppeared = util.getElapsedLogicalTime()
  else
    // Store the time that the object disappears at the outside sensor
    outsideDisappeared = util.getElapsedLogicalTime()
=}
```

# Lingua Franca Implementation of Our Solution (cont'd)

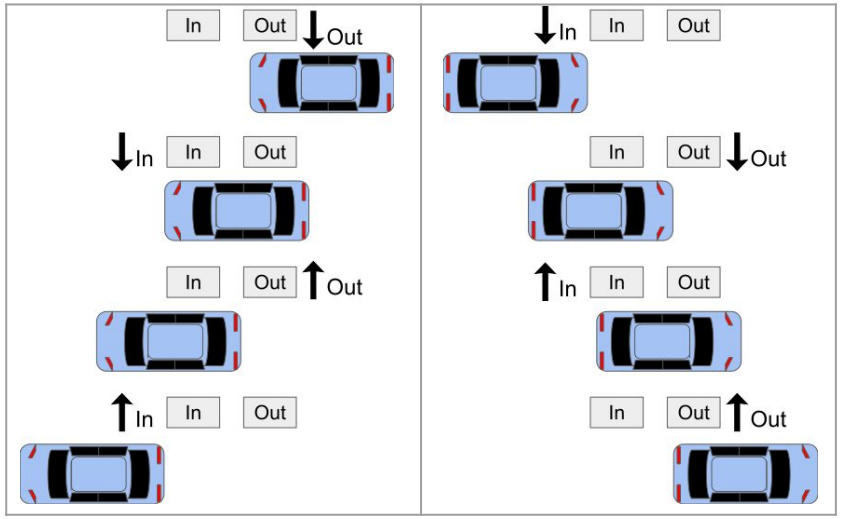
- Arbitration Logic

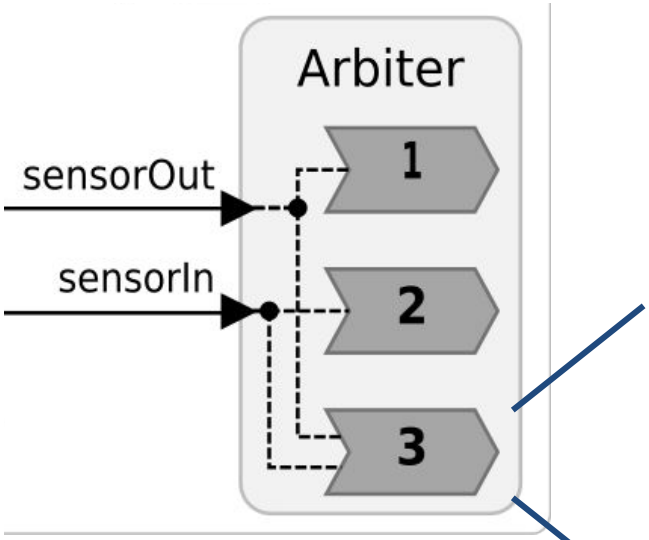



```
reaction(sensorIn) {=
  if (sensorIn)
    // Store the time that an object appears at the inside sensor
    insideAppeared = util.getElapsedLogicalTime()
  else
    // Store the time that the object disappears at the inside sensor
    insideDisappeared = util.getElapsedLogicalTime()
=}
```

# Lingua Franca Implementation of Our Solution (cont'd)

- Arbitration Logic




```
reaction(sensorOut, sensorIn) {=
    if      (outsideAppeared.isEarlierThan(insideAppeared)
             && insideAppeared.isEarlierThan(outsideDisappeared)
             && outsideDisappeared.isEarlierThan(insideDisappeared))
        // The car is entering
    else if (insideAppeared.isEarlierThan(outsideAppeared)
             && outsideAppeared.isEarlierThan(insideDisappeared)
             && insideDisappeared.isEarlierThan(outsideDisappeared))
        // The car is exiting
=}
```

# Experiments - with Synthesized Sensor Events

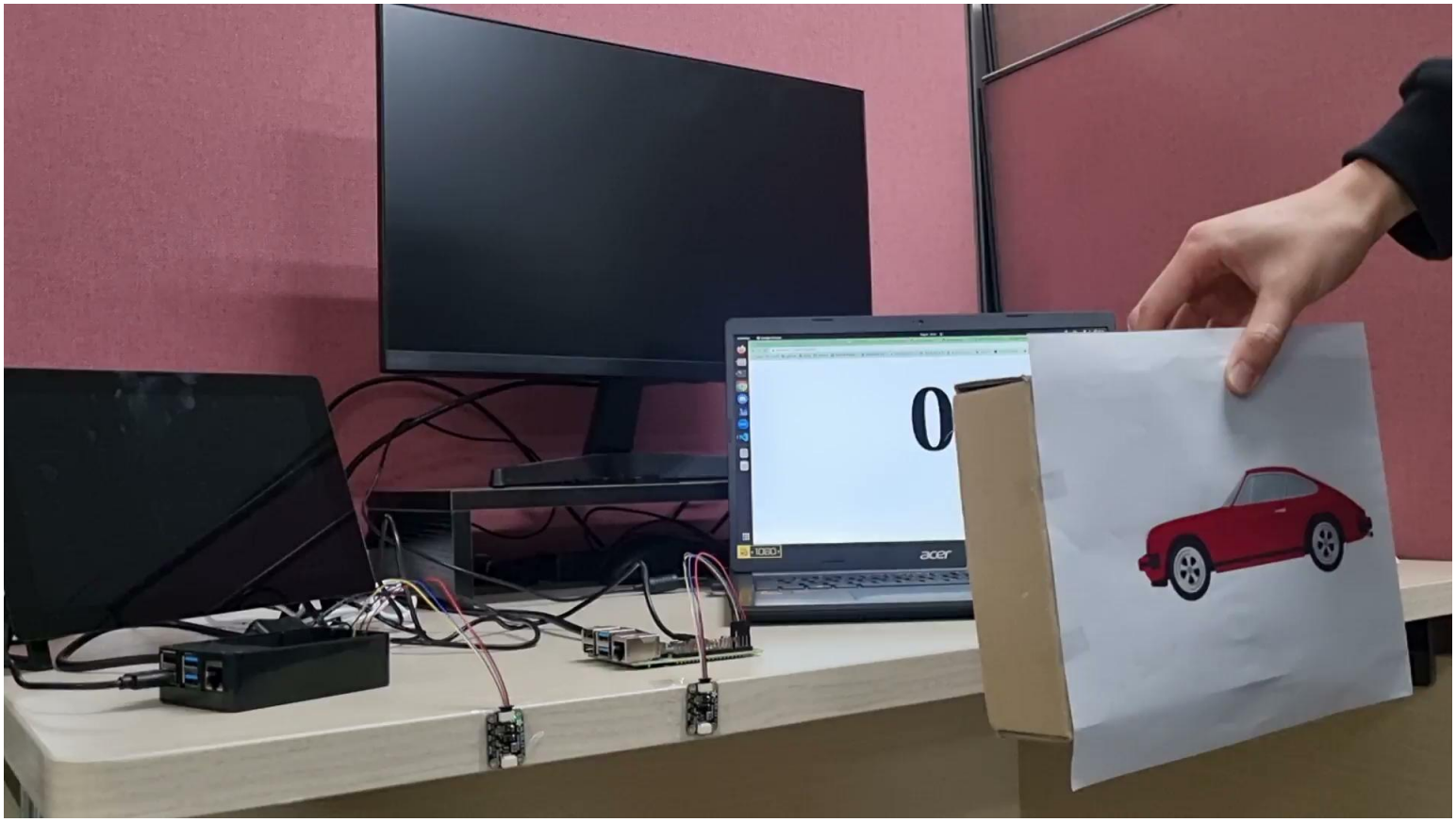
- Generating sensor events using LF timers
- **No misclassification detected** in Arbiter
- With LF, physical events are converted into the tagged events
- (Recap) the duration of events: 90 ms

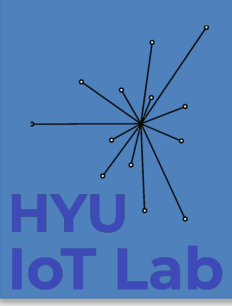
-> No out-of-order classification will happen!!

HYU IoT Lab

# Demo of Real Implementation

- Using Time of Flight Sensors



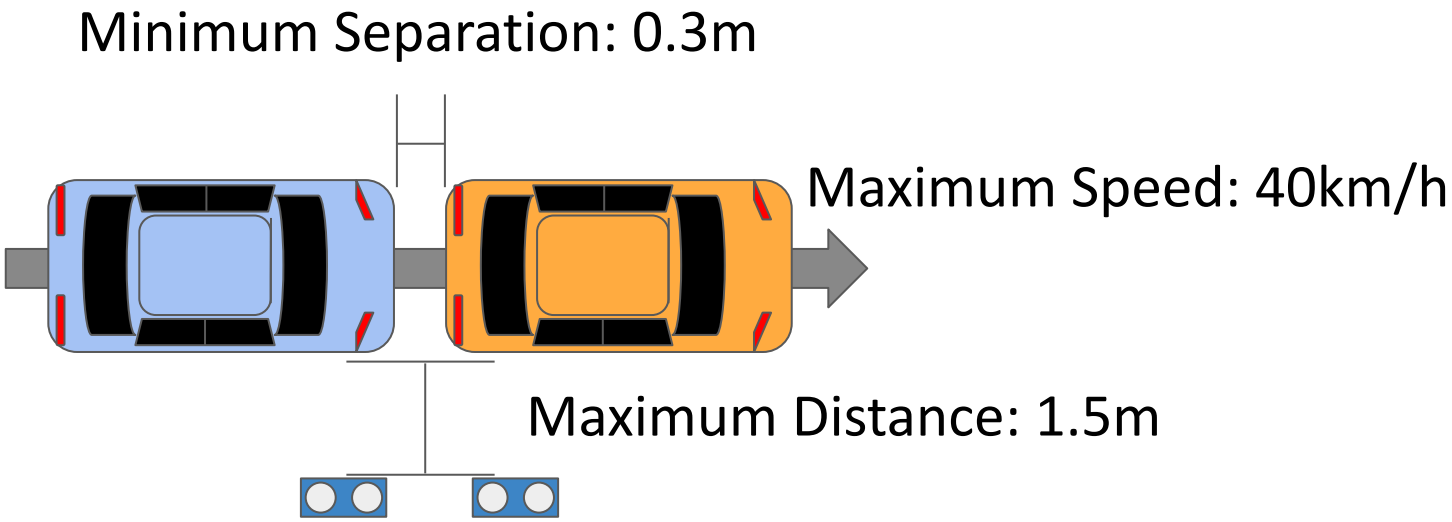



# Different Sensor Options

- Light Sensor
  - Cannot check reflected wave manually
  - Physical actions are required

- Ultrasonic Sensor
  - Need to check reflected sound wave (340 m/s)
  - Timers are required

# Assumptions & Requirements

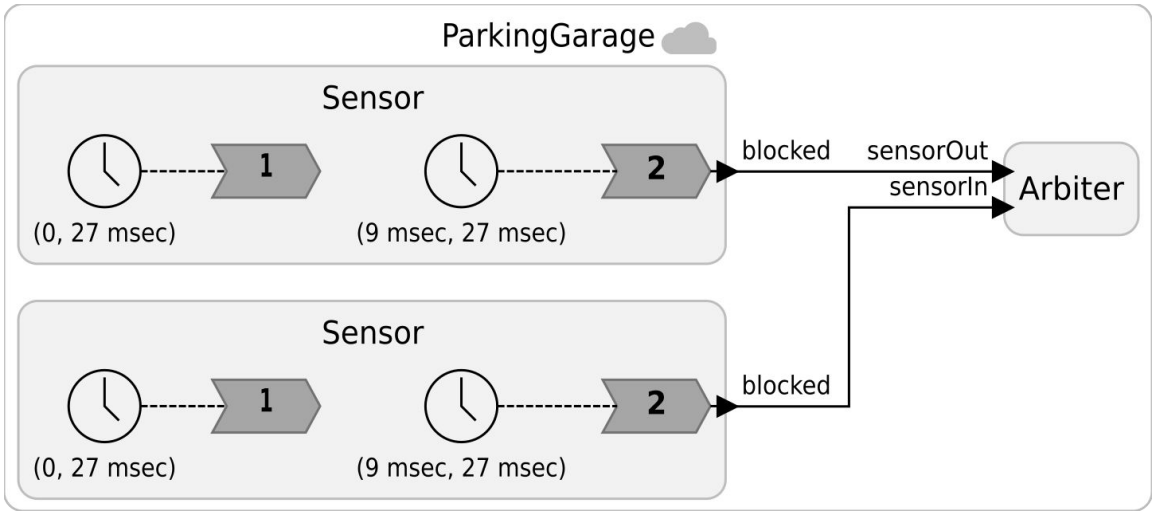



=> Maximum sensing period : 27 ms

=> Time for Ultrasonic to return : 9 ms

# Alternative Implementation

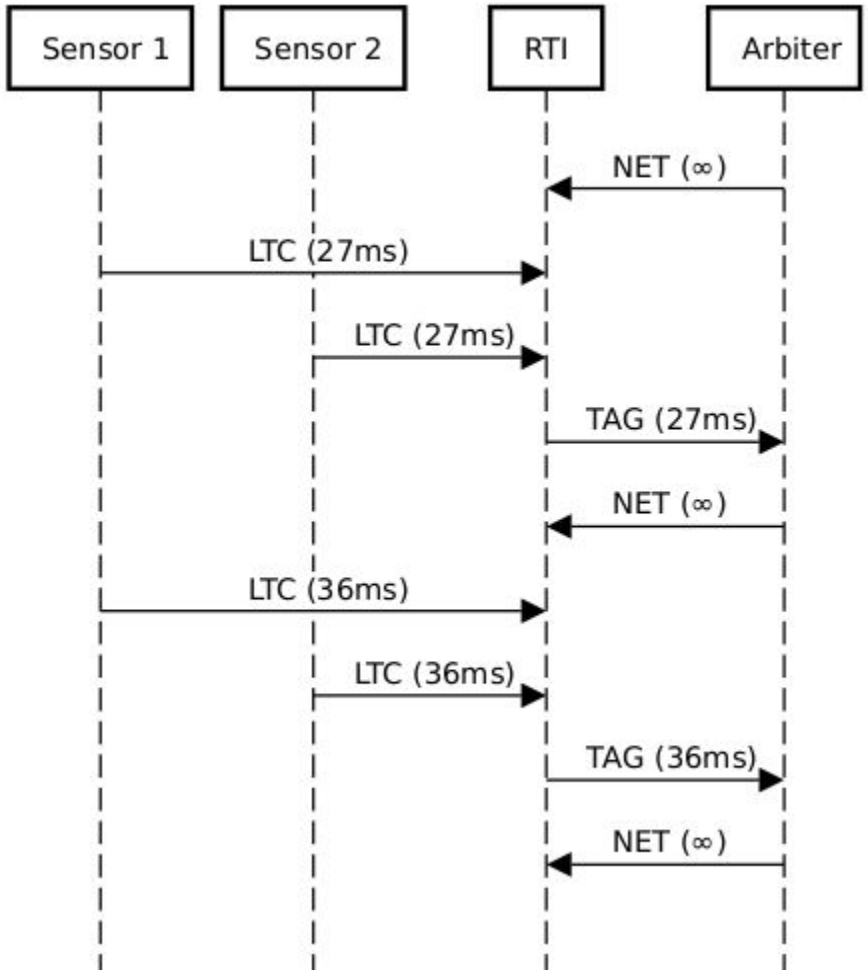
- Polling an Ultrasonic Sensor with Two Timers
  - Emit an ultrasonic wave every 27 ms
  - After 9 ms, check whether the signal is reflected or not



HYU IoT Lab

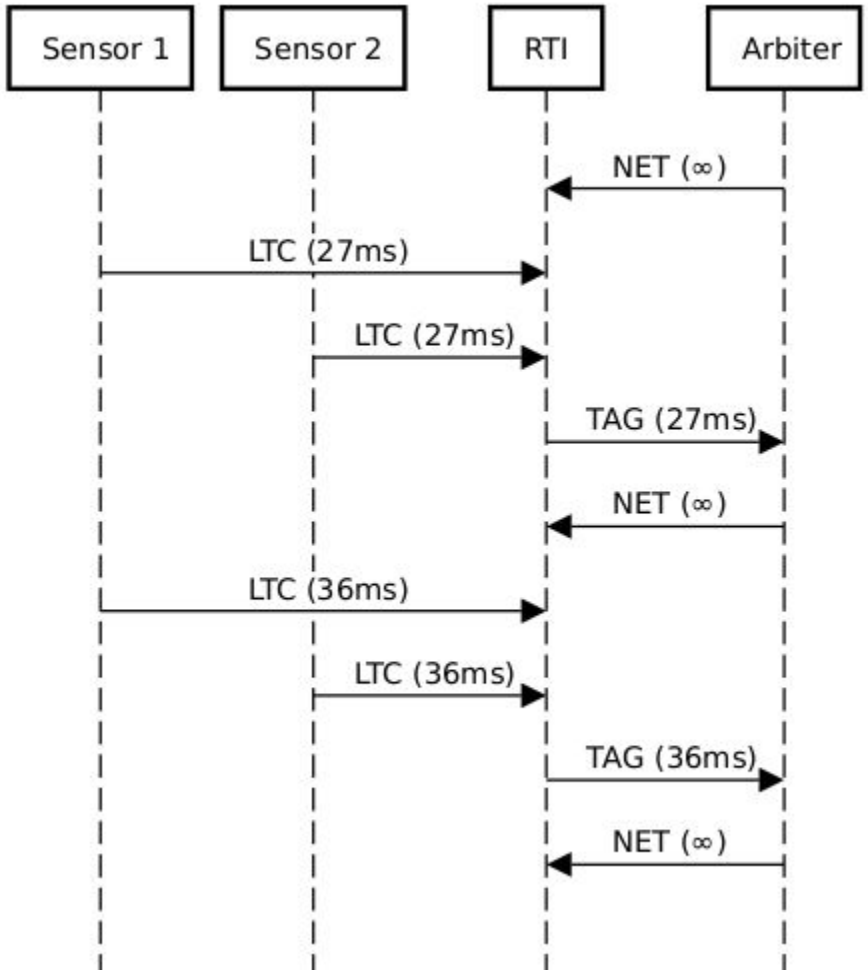
# Message Exchange

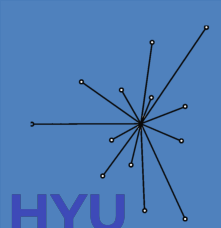
- Time Advance Grant (TAG)
- Logical Tag Complete (LTC)
- Next Event Tag (NET)



# Discovered Issue

- 74 messages are sent every 1 sec
- Actual events occurring very sparsely
- Sensors cannot handle all events

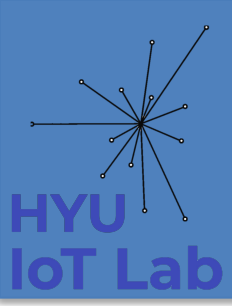





# Discovered Issue (cont'd)

- LF Typescript only supports centralized coordination
- Numerous redundant messages are exchanged
  - Those messages cause an network overhead

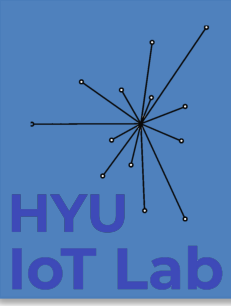
-> How can deal with these?




# Future Work

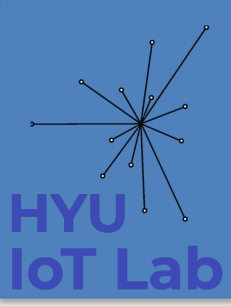
- Optimizing the message-exchanging protocol in centralized coordination
  - A new message type, Next Downstream Event Tag (NDET) is proposed

- Implementing the decentralized coordination to the TypeScript runtime




# Summary

- Processing fine-grained events in order is problematic
- LF perfectly handles the order of fine-grained events
- Future works - to reduce overhead at the current implementation




# Thank You!

**HYU IoT Lab & LF Project Team**

*Time-Centric Reactive Software*
*San Antonio, TX*
*May 9, 2023*