URMILA: A Performance and Mobility-Aware Fog/Edge Resource Management Middleware

Shashank Shekhar, Ajay Chhokra, Hongyang Sun, **Aniruddha Gokhale,** Abhishek Dubey and Xenofon Koutsoukos

Professor of Computer Science and Engineering, Dept of EECS, Vanderbilt University, Nashville, TN, USA

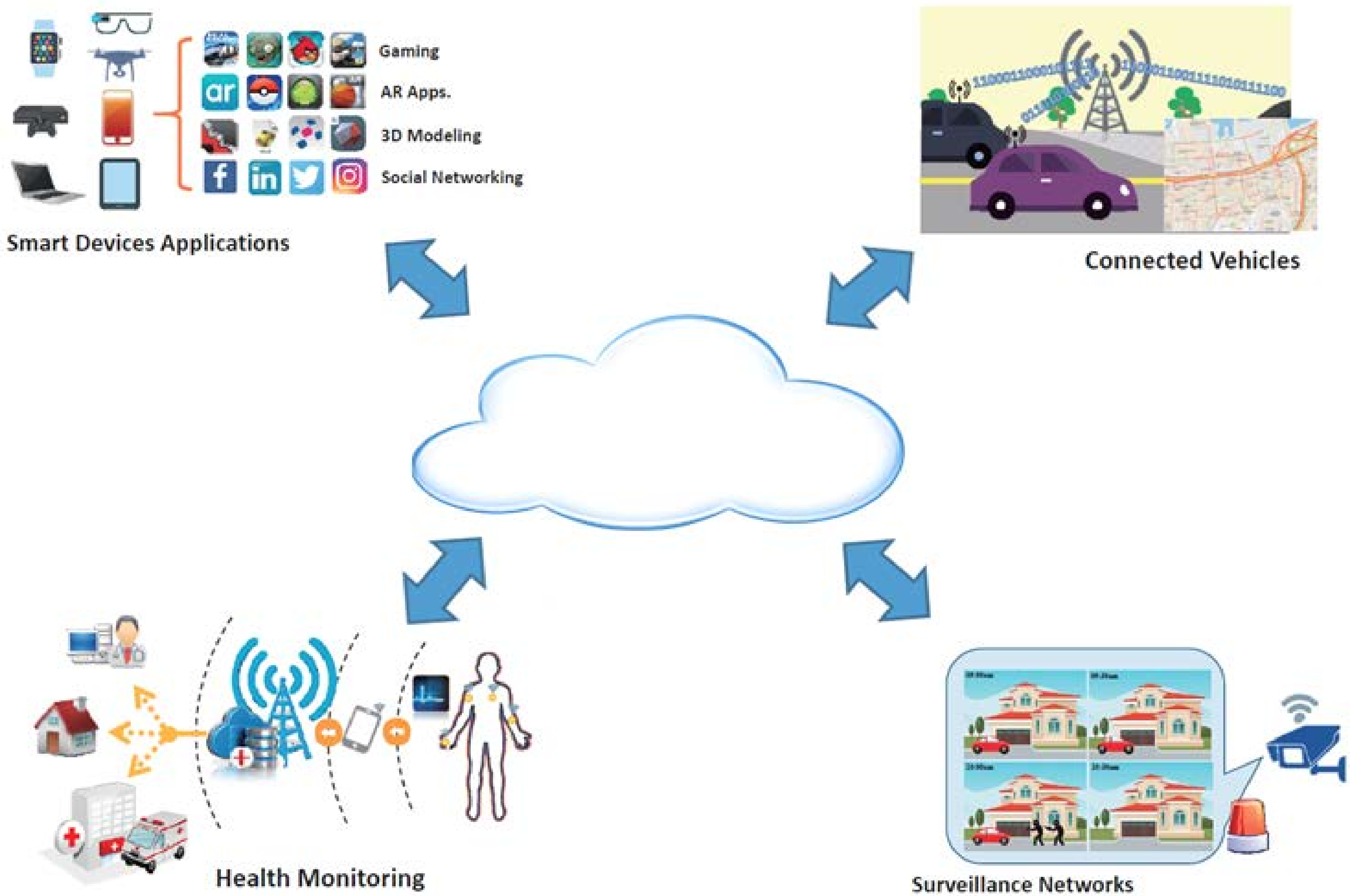
a.gokhale@vanderbilt.edu

http://www.dre.vanderbilt.edu/~gokhale

Presented at ISORC 2019, Valencia, Spain

May 7-9, 2019

IoT/CPS Applications & Cloud Computing

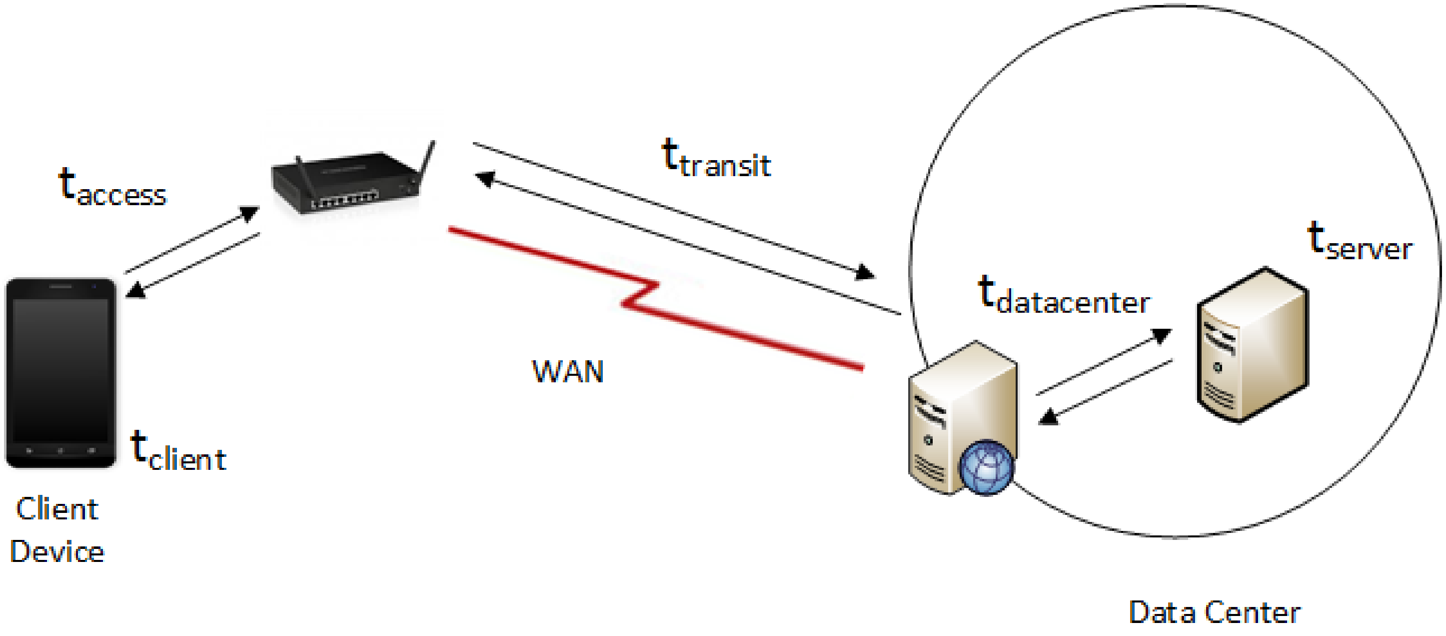


- Soft real-time Cyber-Physical Systems (CPS) /Internet of Things (IoT) applications are increasingly using the cloud for Reliability, Scalability, Elasticity, Cost benefits

Cloud Latencies can be Hurtful to CPS/IoT

- End-to-end (round trip) latency for cloud-hosted IoT applications is computed as:

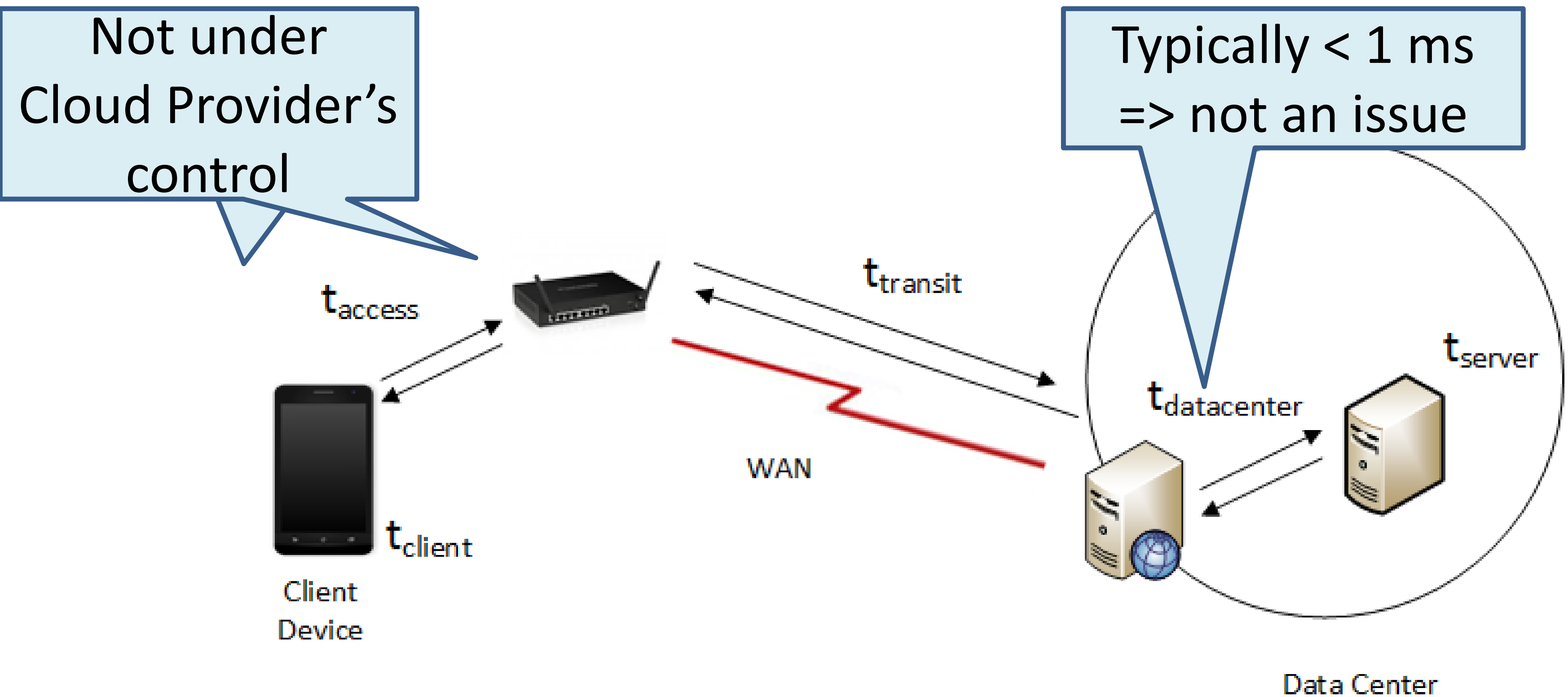
$$t_{total} = t_{client} + t_{access} + t_{transit} + t_{datacenter} + t_{server}$$



Cloud Latencies can be Hurtful to CPS/IoT

- End-to-end (round trip) latency for cloud-hosted IoT applications is computed as:

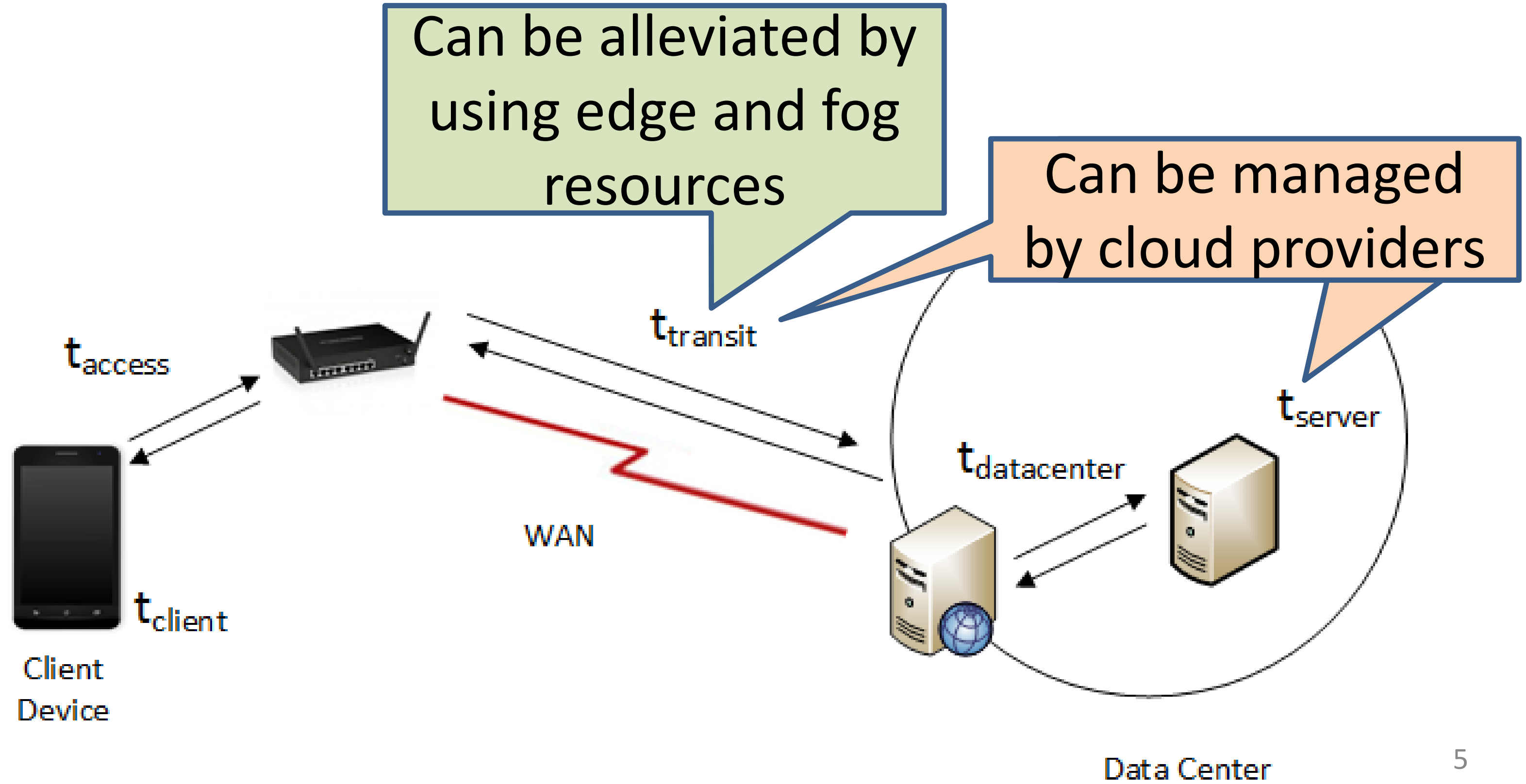
$$t_{total} = t_{client} + t_{access} + t_{transit} + t_{datacenter} + t_{server}$$



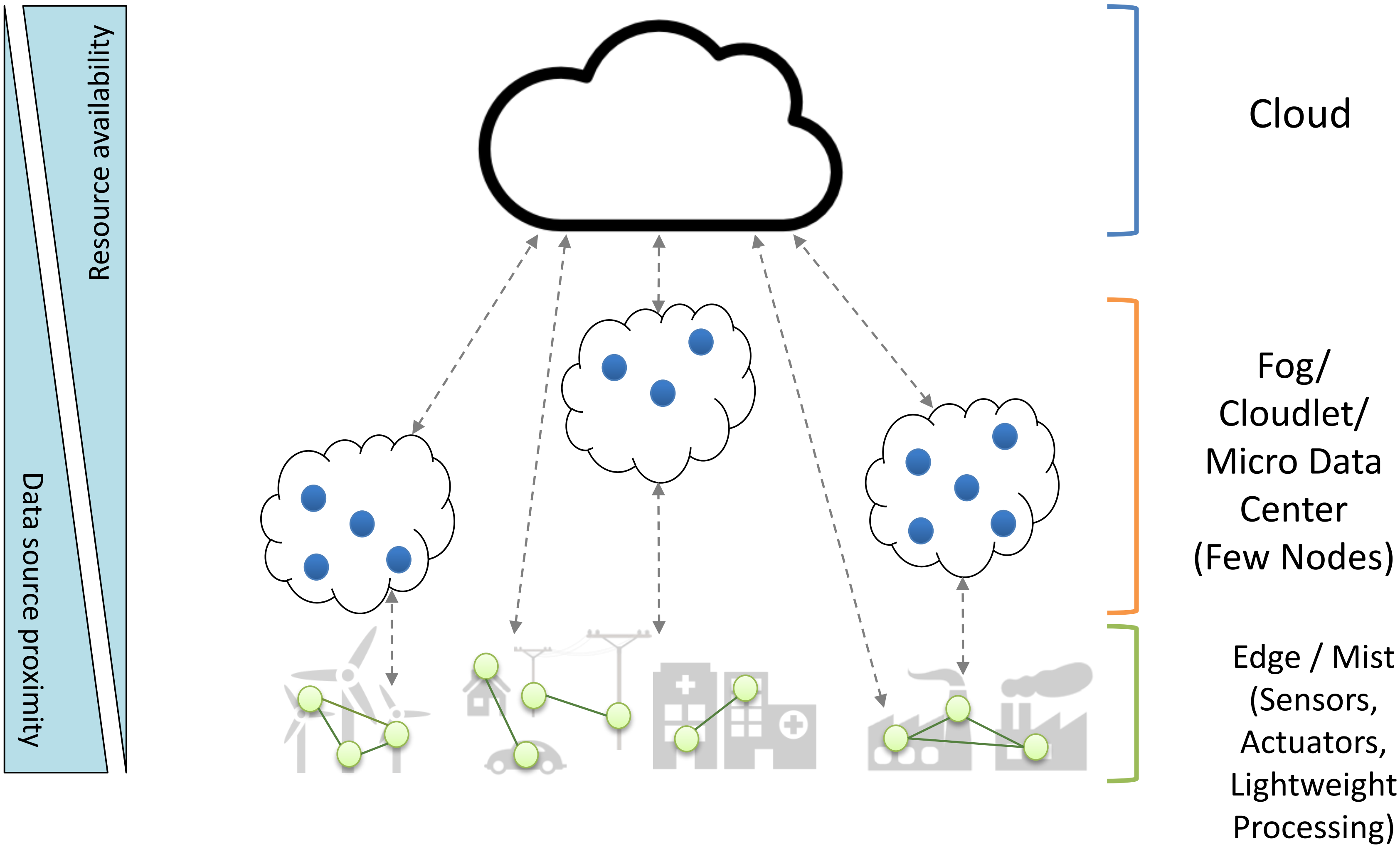
Cloud Latencies can be Hurtful to CPS/IoT

- End-to-end (round trip) latency for cloud-hosted IoT applications is computed as:

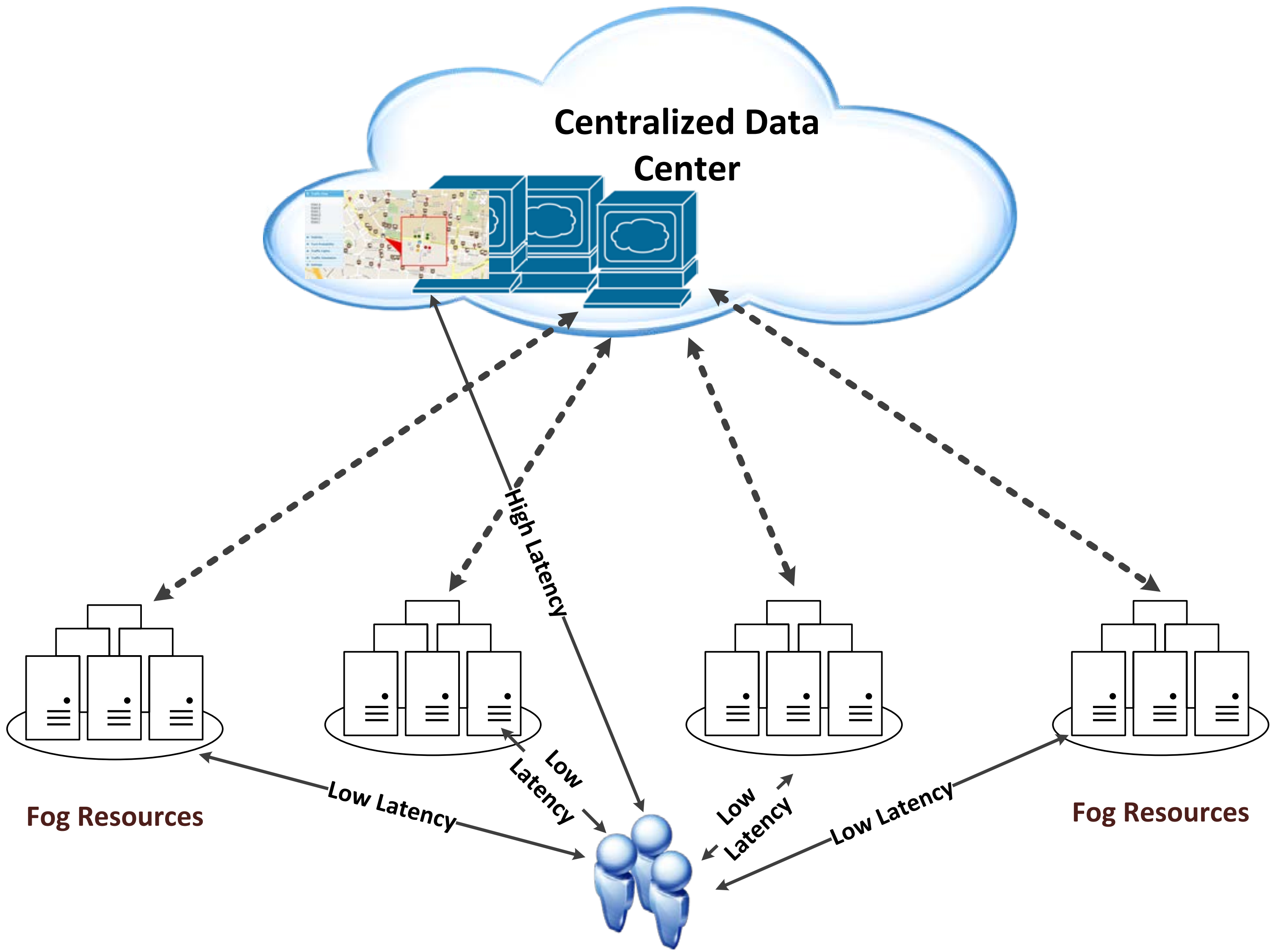
$$t_{total} = t_{client} + t_{access} + t_{transit} + t_{datacenter} + t_{server}$$



A 3-tiered Cloud Architecture for CPS/IoT



Cloud-Fog-Edge Computing Model



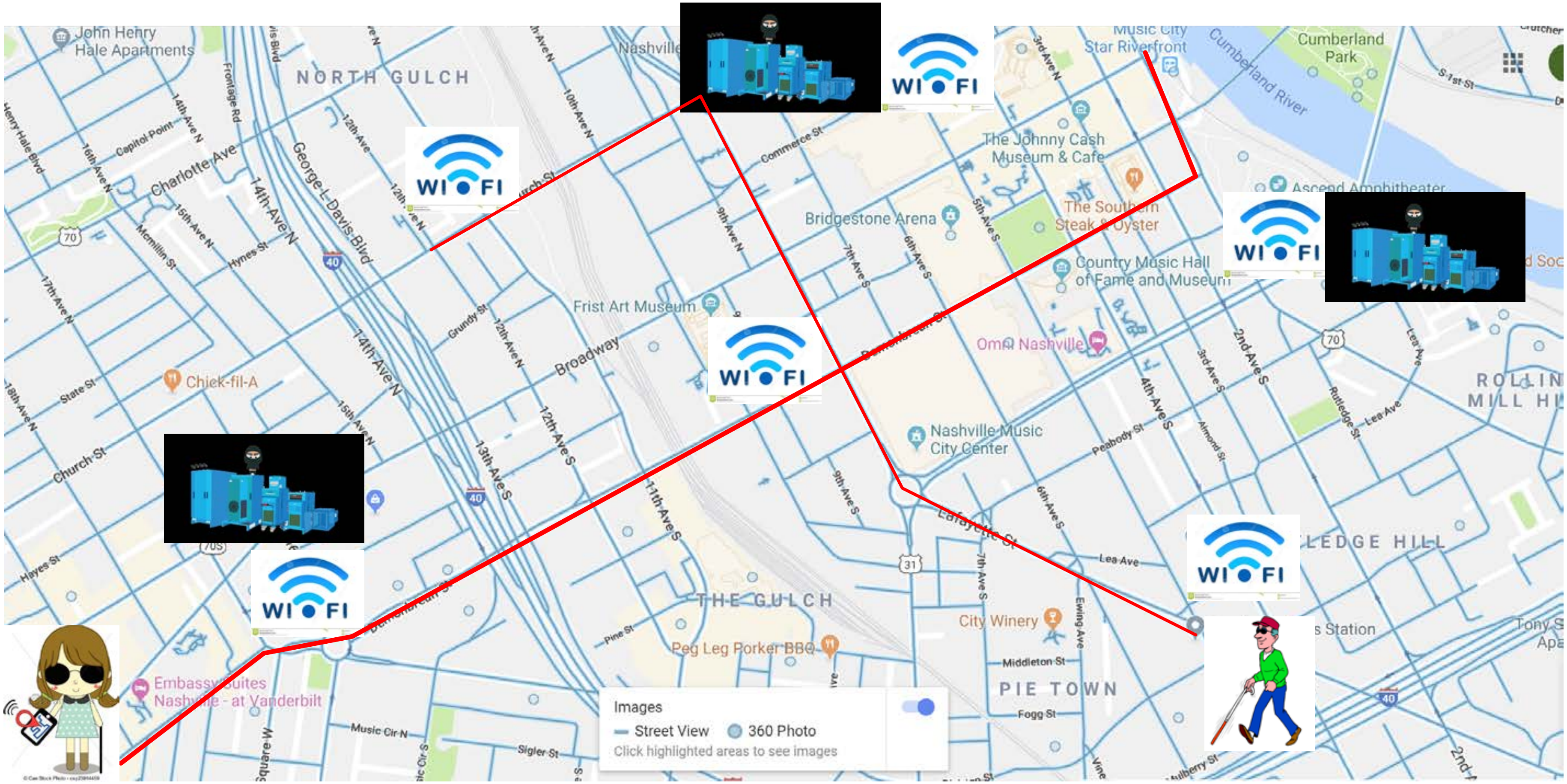
Motivational Use Case: Real Time Object Detection

- A number of fast and accurate algorithms based on convolutional neural networks for object detection have been developed in the last few years
  - YOLO, SSD, MobileNet, ResNet, Inception

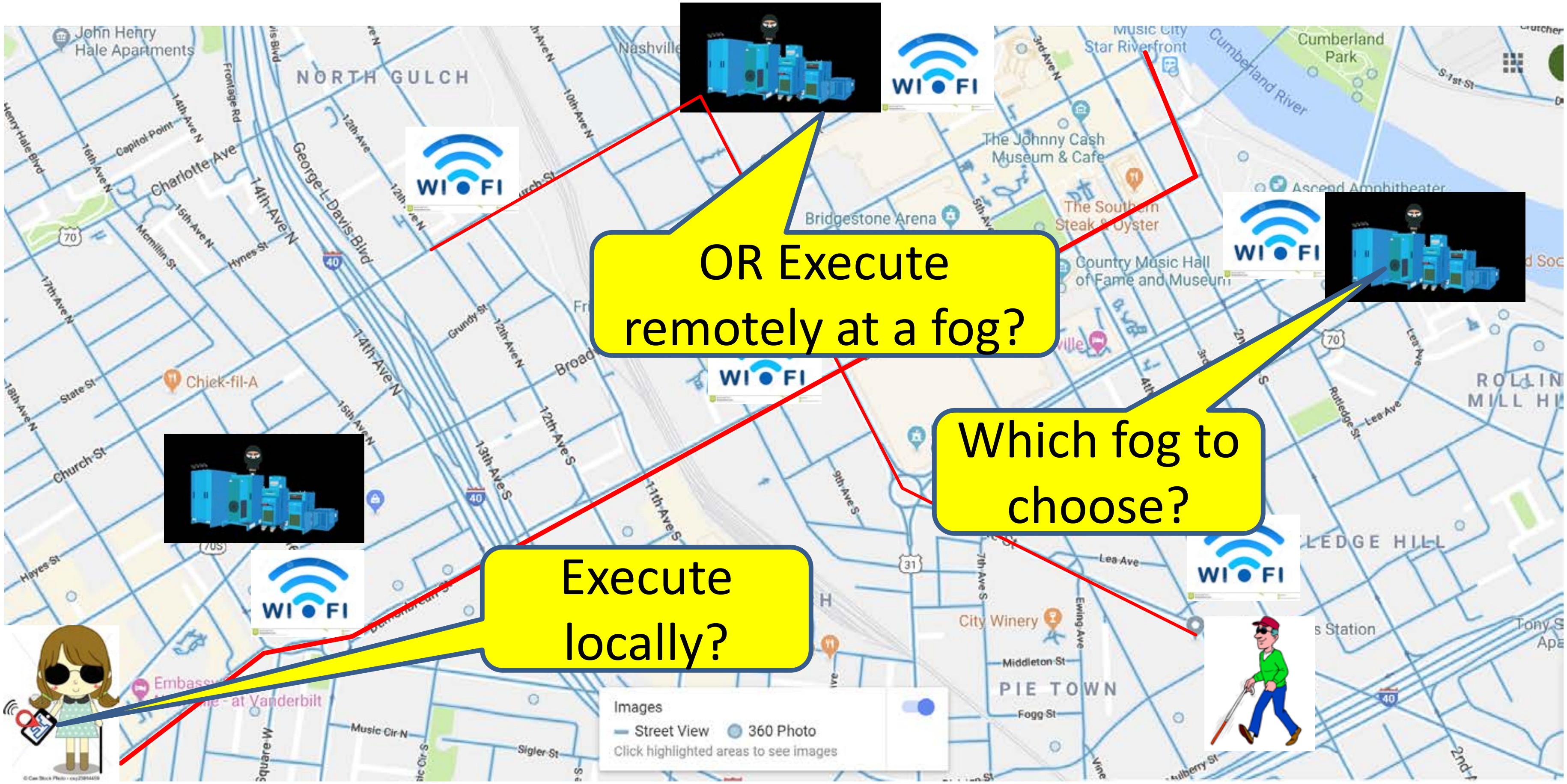
Object Identification using ResNet


Image Credit: Microsoft Seeing AI

Must Now Address User Mobility



Runtime Decisions to be Made

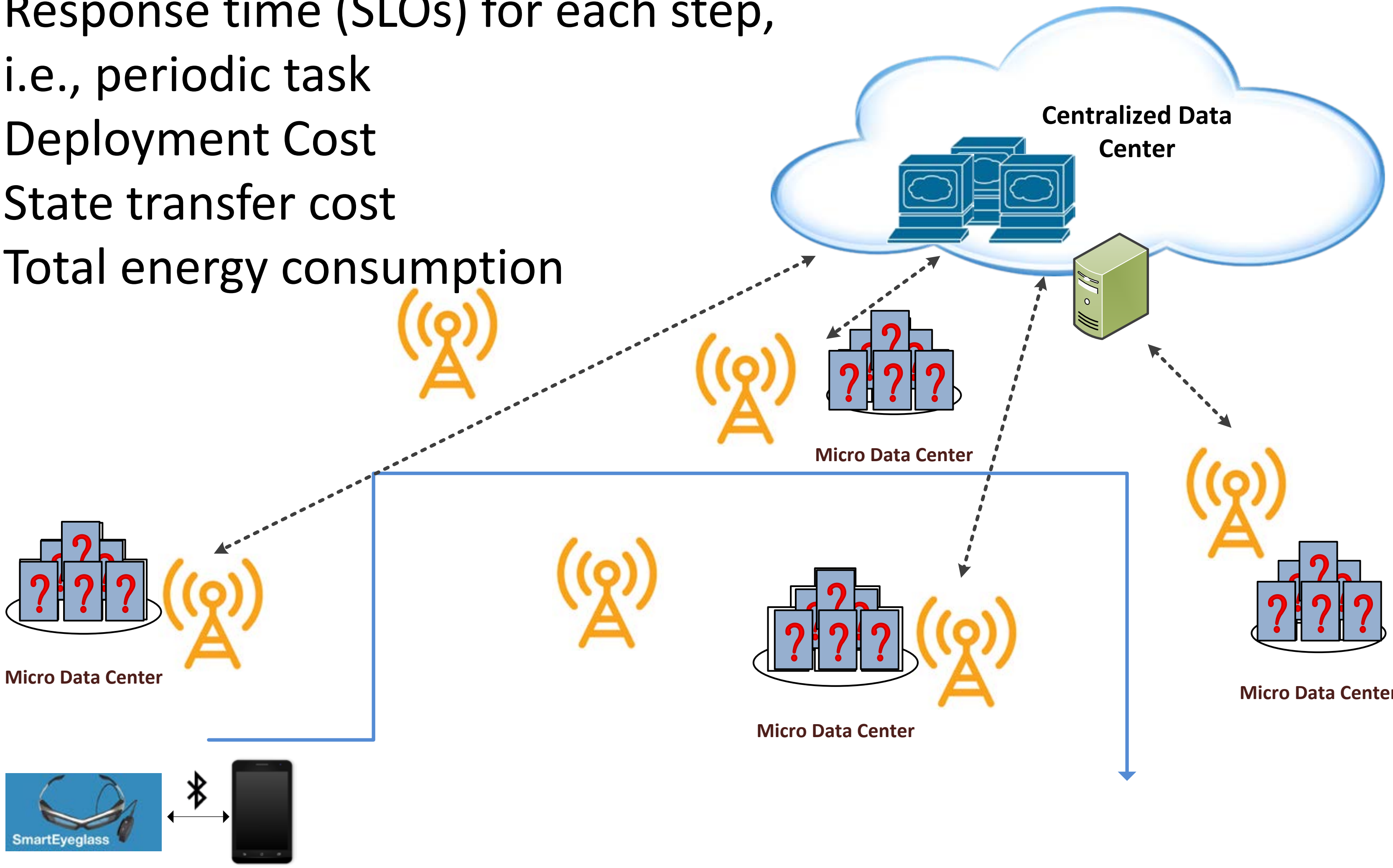


Multi-objective Solution Requirements

1) Meeting Service-Level Objectives for the service is critical
2) Conserving battery resources on edge devices is paramount => leverage fog as much as possible
    - But which one? Or do we keep handing off from one fog to another?
3) High service availability is critical => during durations of bad wireless signals, edge device must be leveraged
    - But the duration for which edge is used should be kept as low as possible
4) Overall cost of deployment and operation must be kept at a minimum

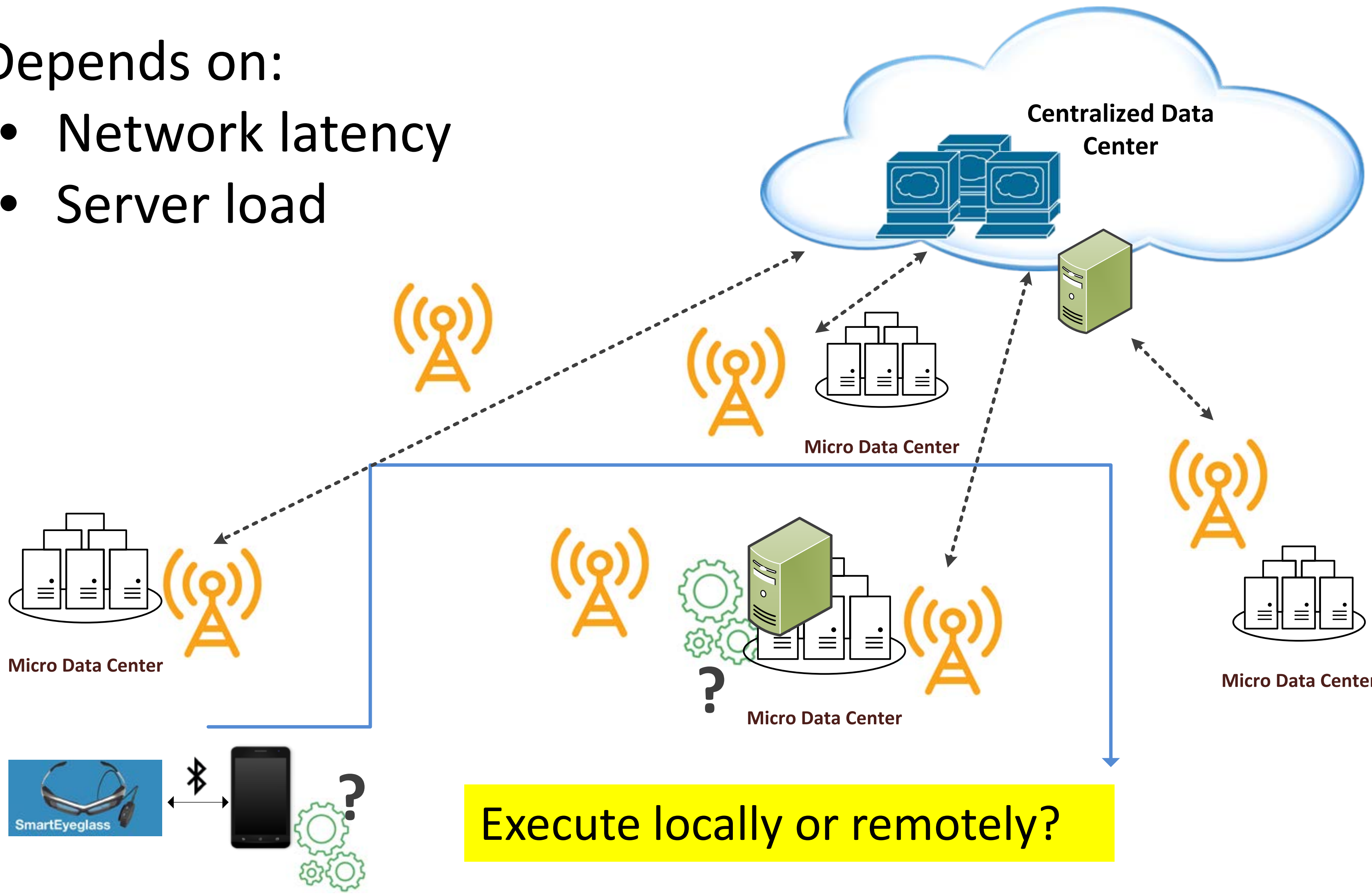
Problem: Choosing a Fog Resource

- Depends on:
  - Response time (SLOs) for each step, i.e., periodic task
  - Deployment Cost
  - State transfer cost
  - Total energy consumption



Problem: Local or Remote Execution?

- Depends on:
  - Network latency
  - Server load

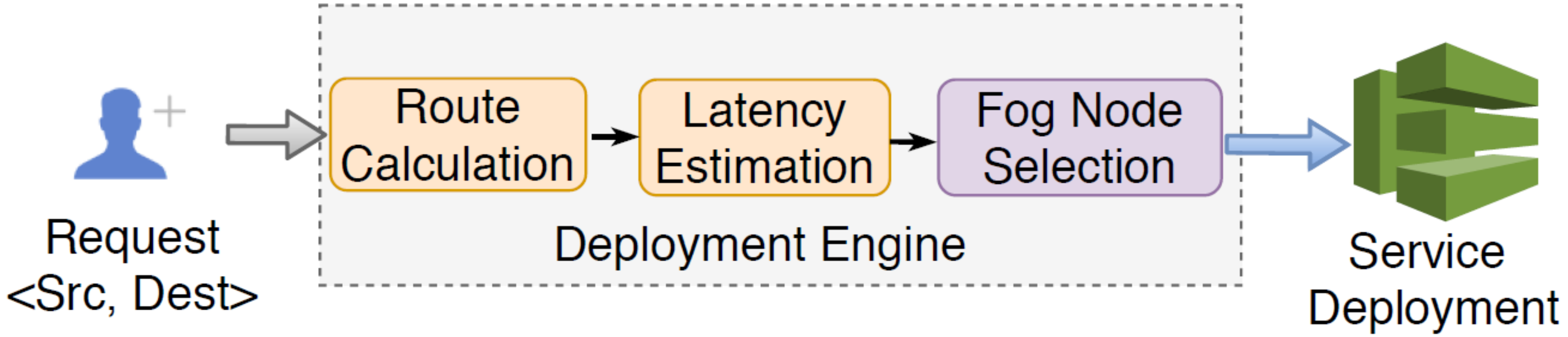


Solution Approach

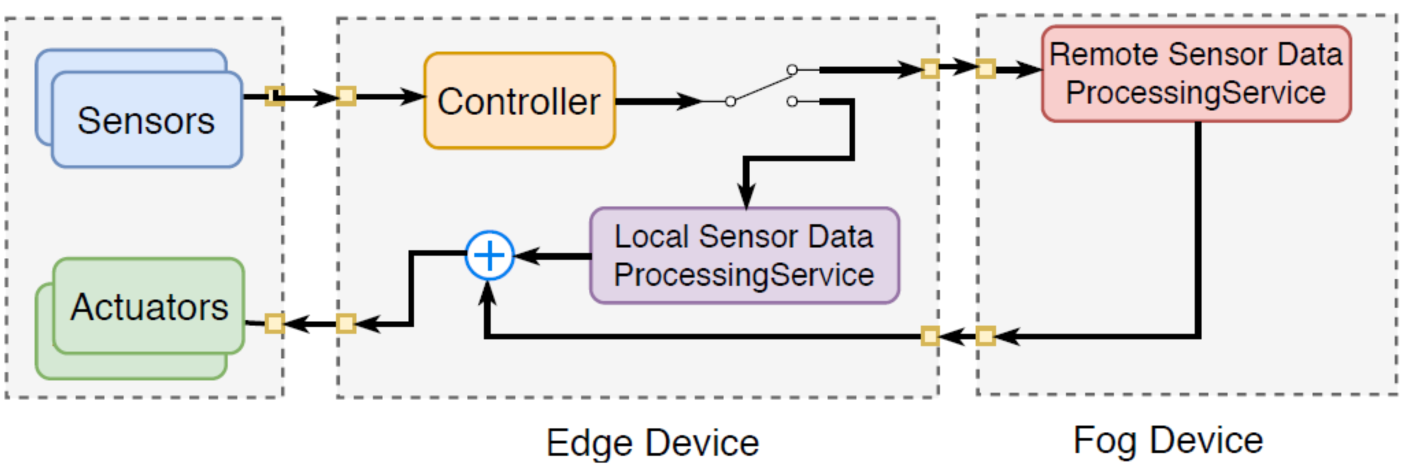
- Exclusively offline solution?
  - No, because the instantaneous loads on fog resources and density of users in the wireless areas cannot be known ahead of time
- Exclusively online solution?
  - No, because collecting all the information needed to make informed decisions from distributed sources and making those decisions in near real-time is not feasible
- Our approach: hybrid solution comprising partly offline and partly online
  - Offline part uses machine learning techniques to build models of the system
  - Online part relies on just the most critical information needed at runtime which is then used in conjunction with the learned models to make decisions on whether to use the fog or the edge to keep the service available

Solution – Ubiquitous Resource Management for Interference and Latency-Aware services (URMILA)

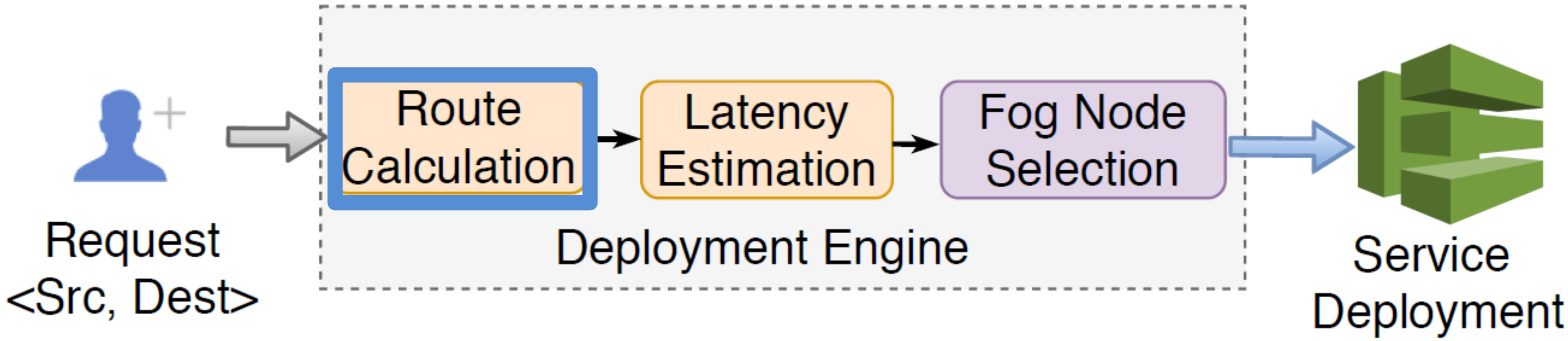
- **Deployment Phase:**



- **Runtime Phase:**

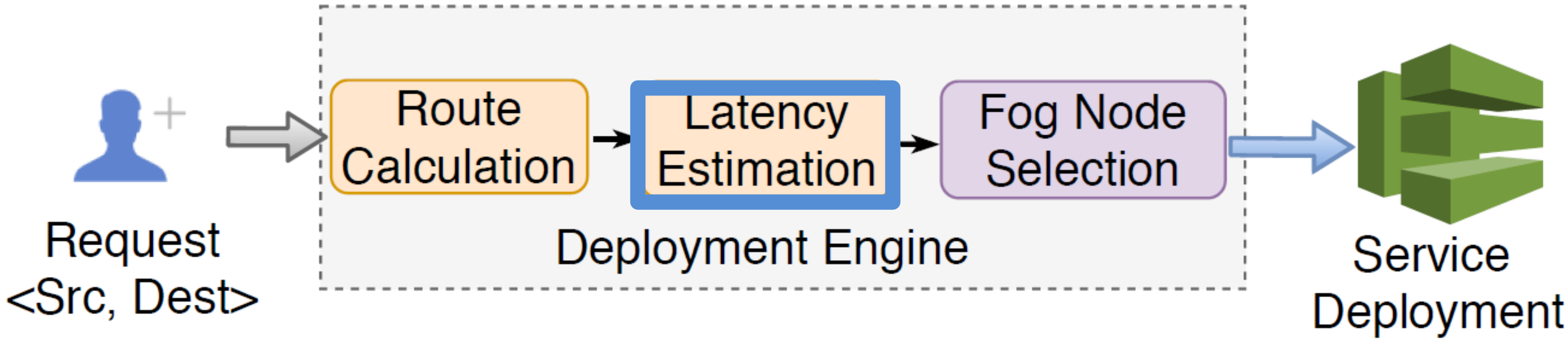


Deployment Phase – Route Calculation



- Techniques:
  - Probabilistic - data driven techniques
    - substantially data intensive
    - lacks generality
  - Deterministic - user's input and a navigation service
- Our Choice: Deterministic using Google Maps API
  - Routes are divided into small segments receiving same signal strength
  - Constant speed model

Deployment Phase – Latency Estimation



Last hop latency

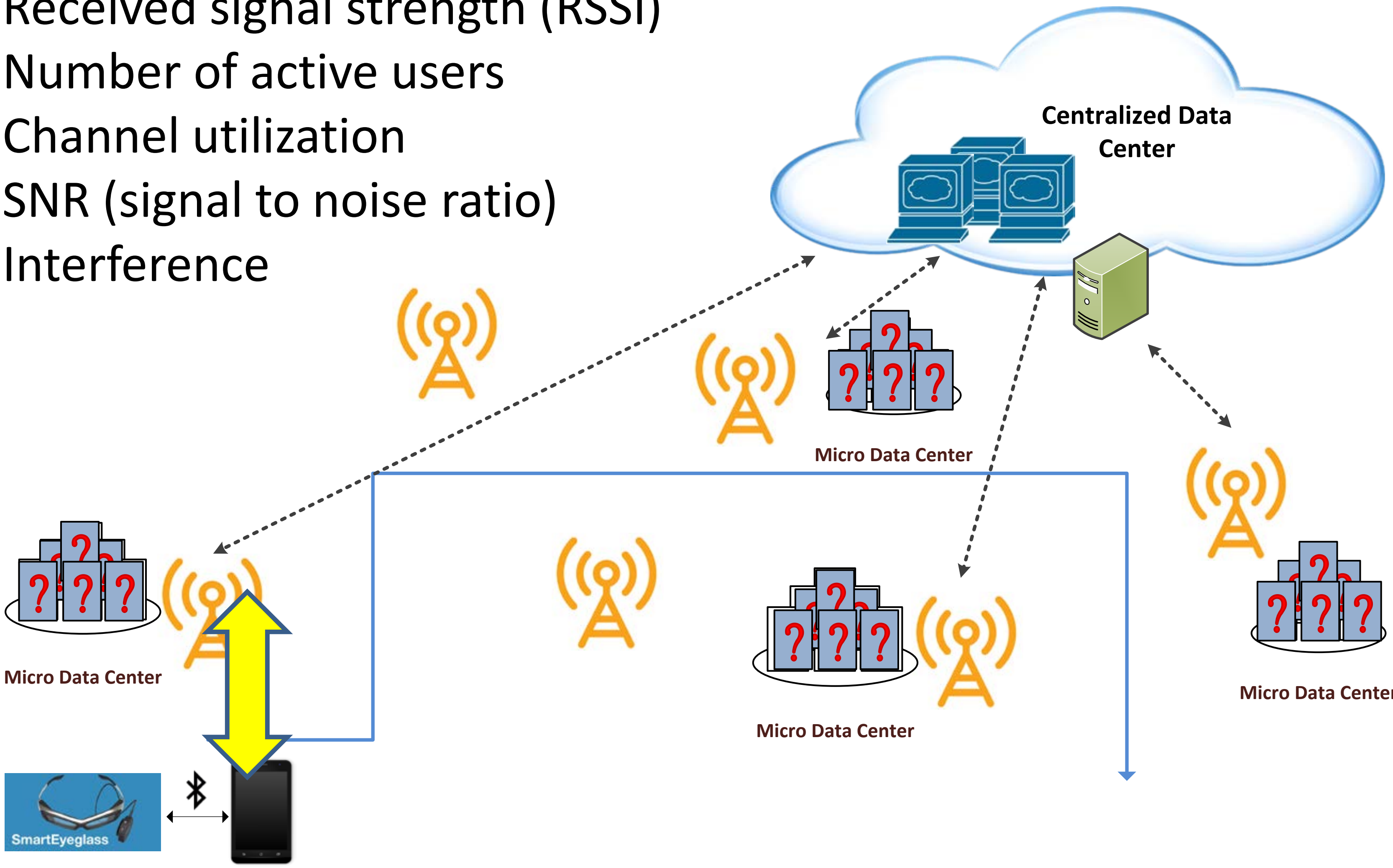
WAN latency

$$t_{network}(u, s_{i,h}, \ell) = t_{u,ap(u,\ell)} + t_{ap(u,\ell),ap_i} + t_{ap_i,s_{i,h}}$$

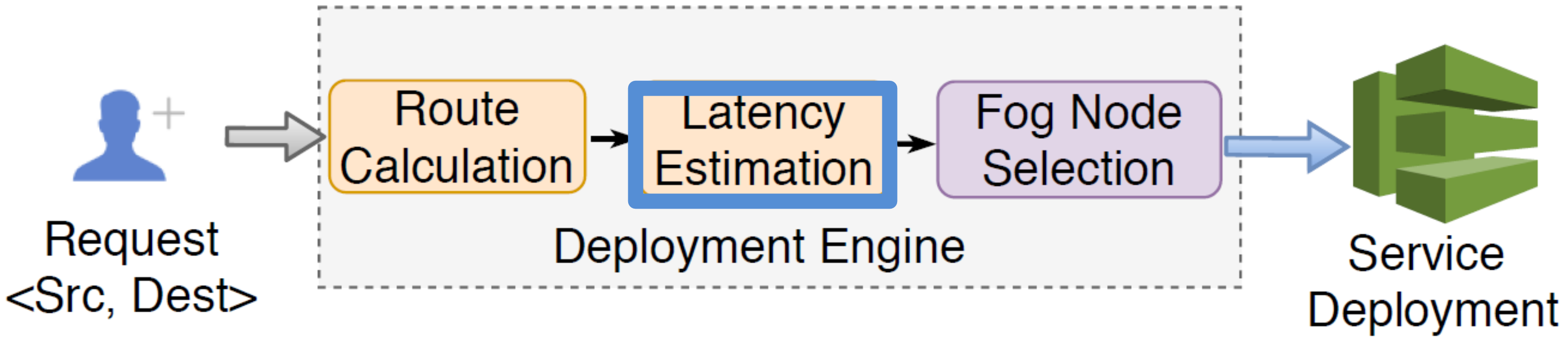
Data center latency (negligible)

Deployment Phase – Latency Estimation

- Factors affecting last-hop latency:
  - Received signal strength (RSSI)
  - Number of active users
  - Channel utilization
  - SNR (signal to noise ratio)
  - Interference



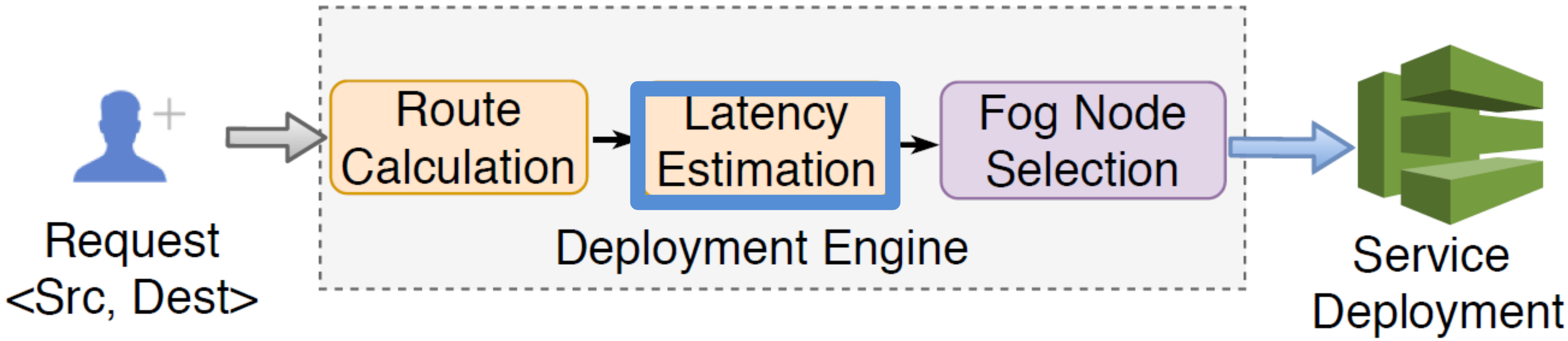
Deployment Phase – Latency Estimation: Selecting WAP



$$t_{network}(u, s_{i,h}, \ell) = \boxed{t_{u,ap(u,\ell)}} + t_{ap(u,\ell),ap_i} + t_{ap_i,s_{i,h}}$$

- We apply standard handover policy based upon the received signal strength
  - Client device selects an access point with the highest signal strength
  - Lazy handover - sticks to WAP till RSSI drops below threshold (-67)
  - Can be swapped with other policies, e.g. strongest RSSI, WAP assisted roaming, multiple WAP association etc.
  - Handover duration depends on client device and WAP
    - Apply measurement-based approach

Deployment Phase – Latency Estimation: Maintaining Knowledgebase

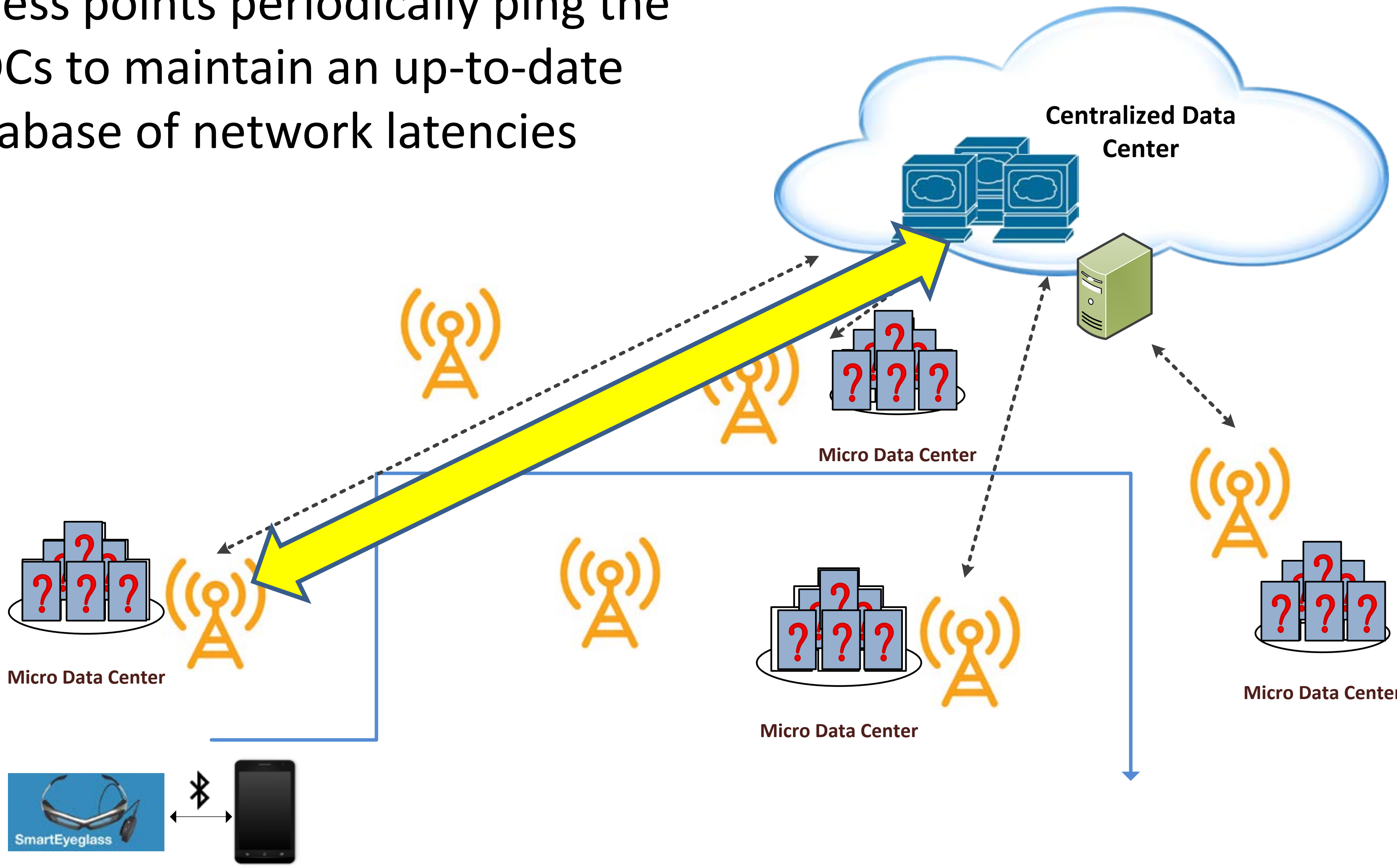


$$t_{network}(u, s_{i,h}, \ell) = t_{u,ap(u,\ell)} + t_{ap(u,\ell),ap_i} + t_{ap_i,s_{i,h}}$$

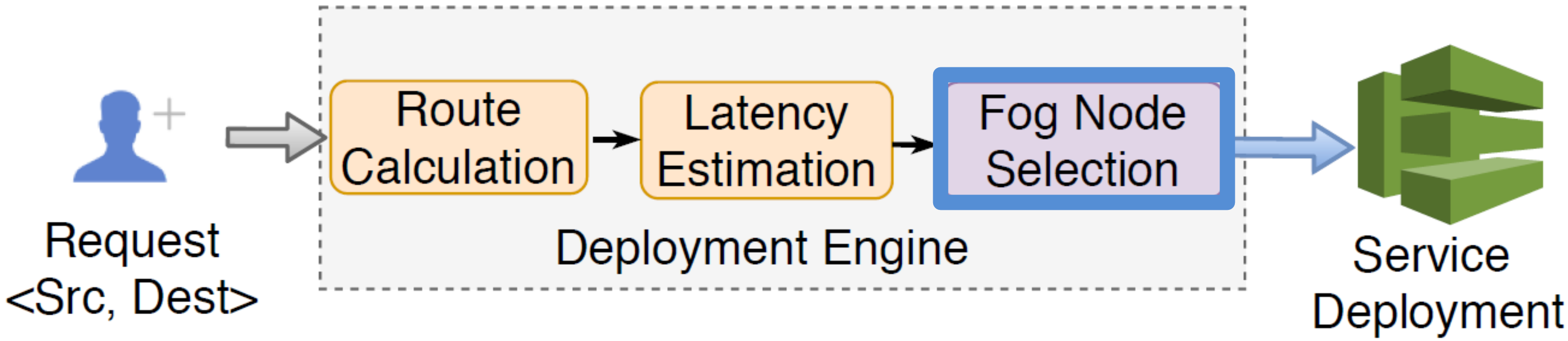
- Route segments in a given geographical region are profiled
- We created a database of coordinates, time of day and latency
- Latency is a function of location and time of day
- Perform lookup

Deployment Phase – Latency Estimation: WAP Latency

- Access points periodically ping the MDCs to maintain an up-to-date database of network latencies



Deployment Phase – Fog Node Selection: via Performance Interference Estimation



$$t_{total}(u, \ell) \le \phi_u, \ \forall \ell$$

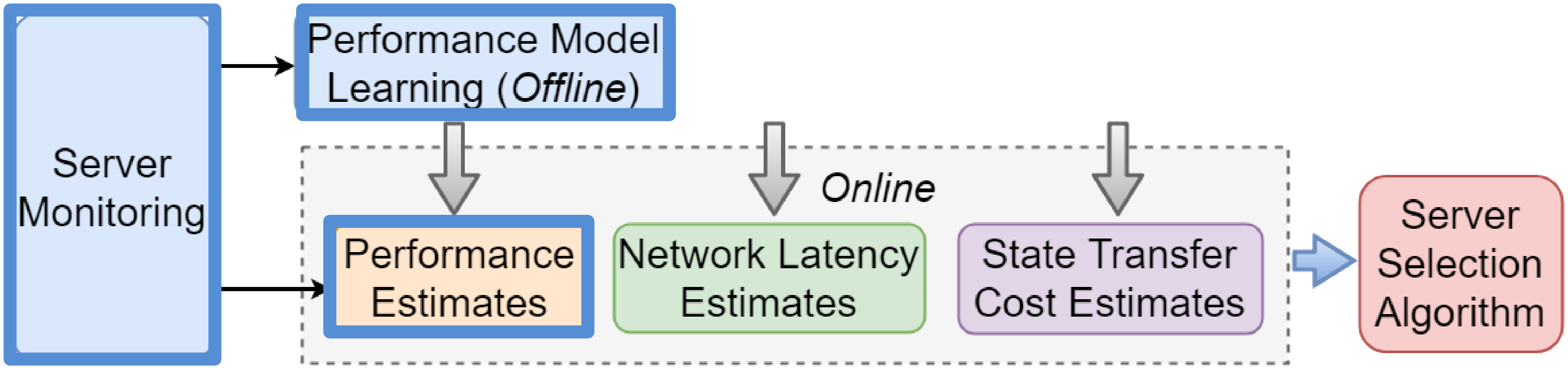
Sensitivity

$$t_{total}(v, \ell) \le \phi_v, \ \forall \ell, v \in V$$

Pressure

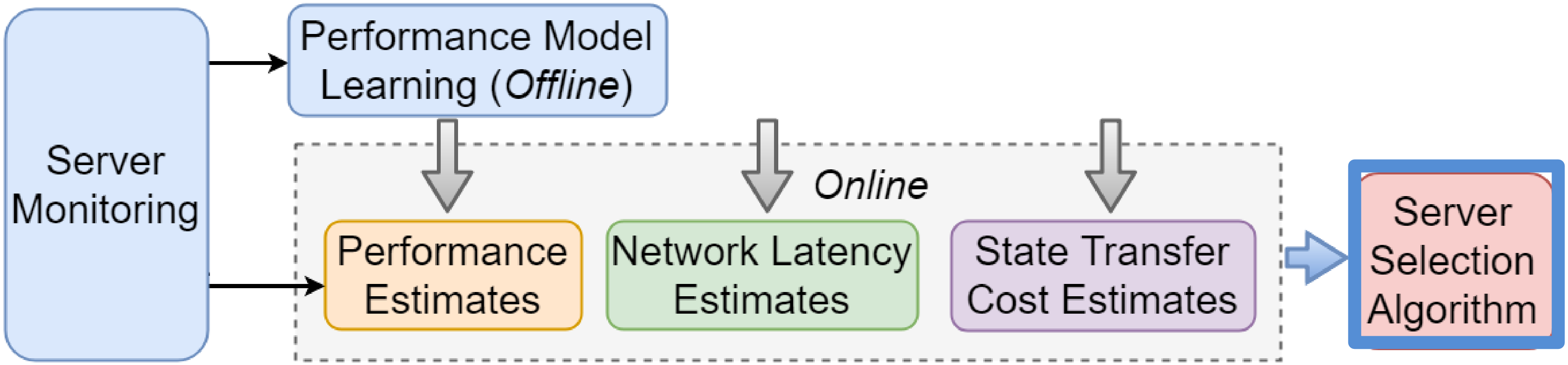
- Interference Profile of an application consists:
  - **Sensitivity**: Performance degradation of an application due to interference from other applications
  - **Pressure**: Performance degradation of other co-located applications on the host due to the application

Deployment Phase – Fog Node Selection: via Performance Interference Estimation



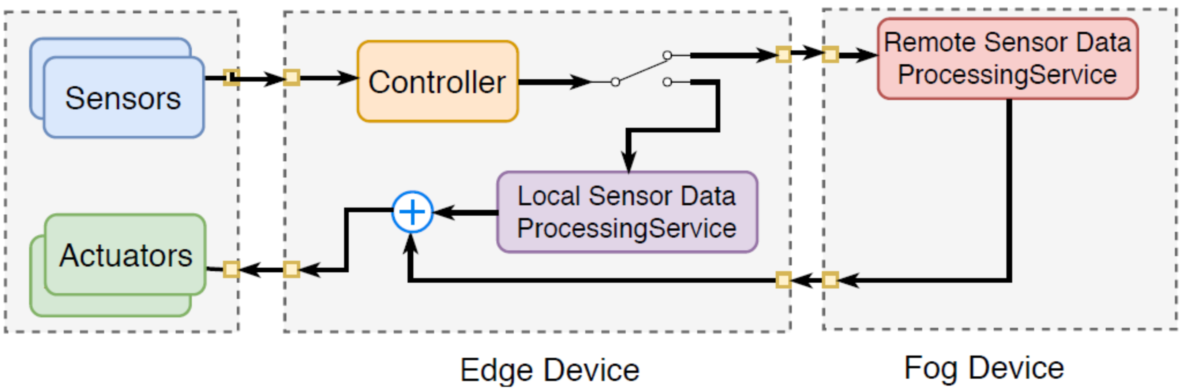
- Apply our FECBench data collection and model learning
  - Collectd, AMQP, InfluxDB
  - Gradient tree boosting curve fitting
- Enhanced for:
  - Docker containers
  - NUMA architecture
  - Intel Cache Monitoring Technology (CMT)

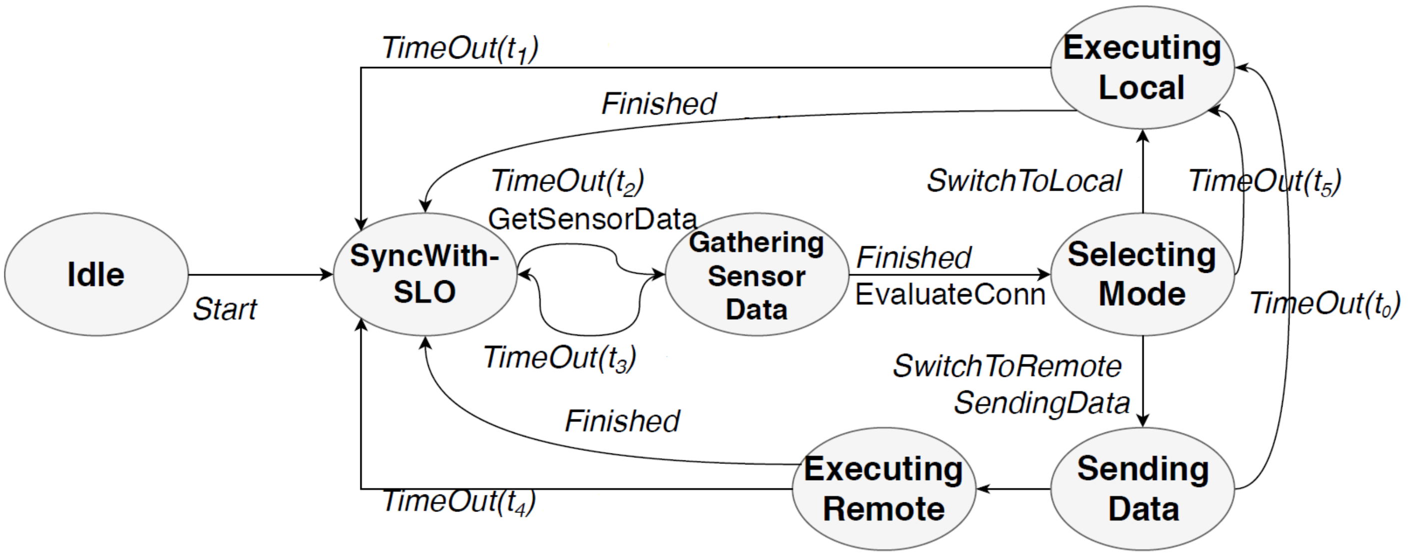
Deployment Phase – Fog Node Selection: Server Selection Algorithm



- Problem is then formulated as an optimization problem
- Solved using a runtime heuristic approach

Runtime Phase





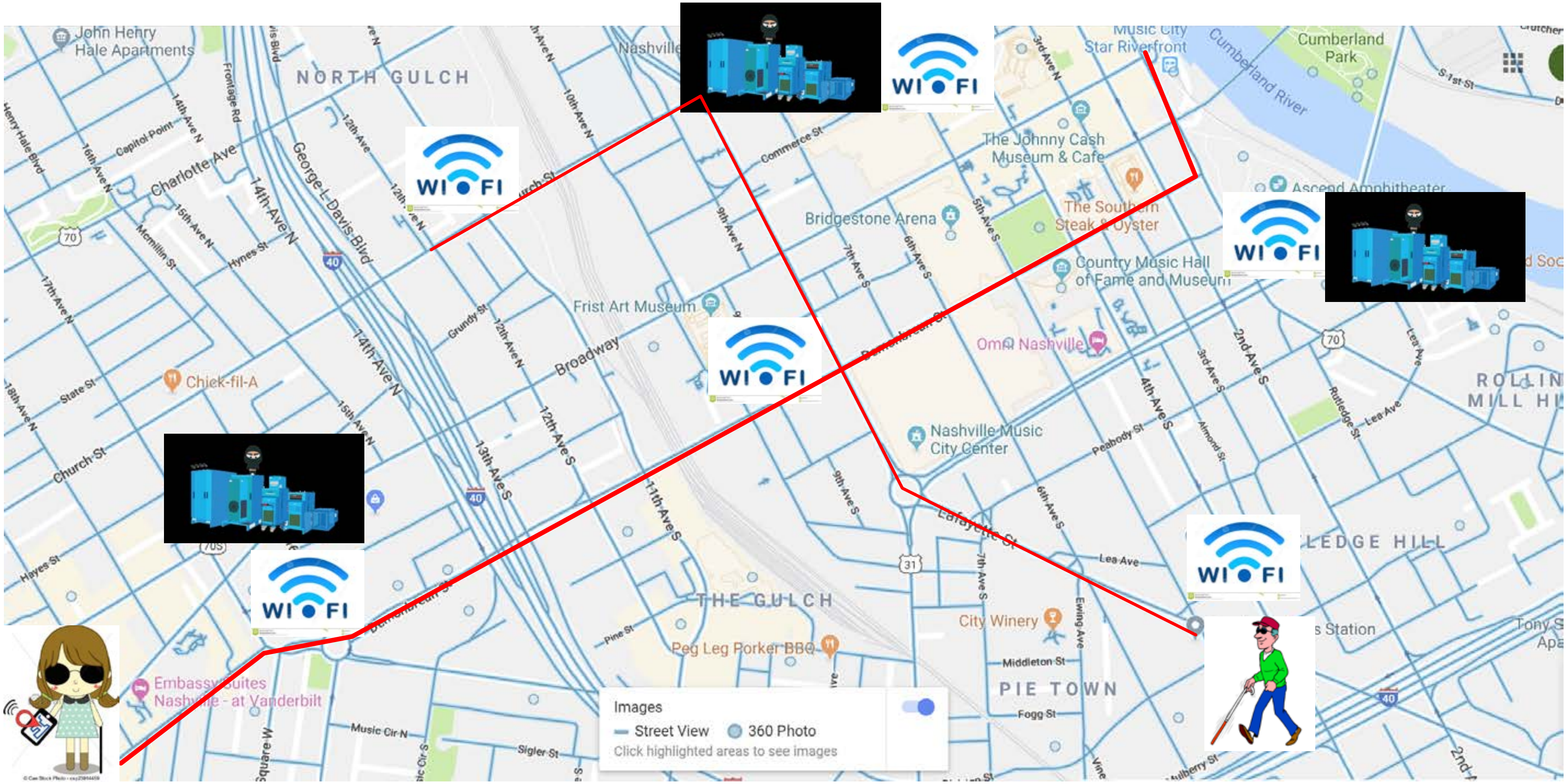
- EvaluateConn accounts for both initial decision and current received signal strength to select the execution mode

Experimental Setup

- Wireless Access Points:
  - Raspberry Pi 2B
  - OpenWRT 15.05.1
  - 2.4 GHz Channel frequency
  - -67 dBm threshold

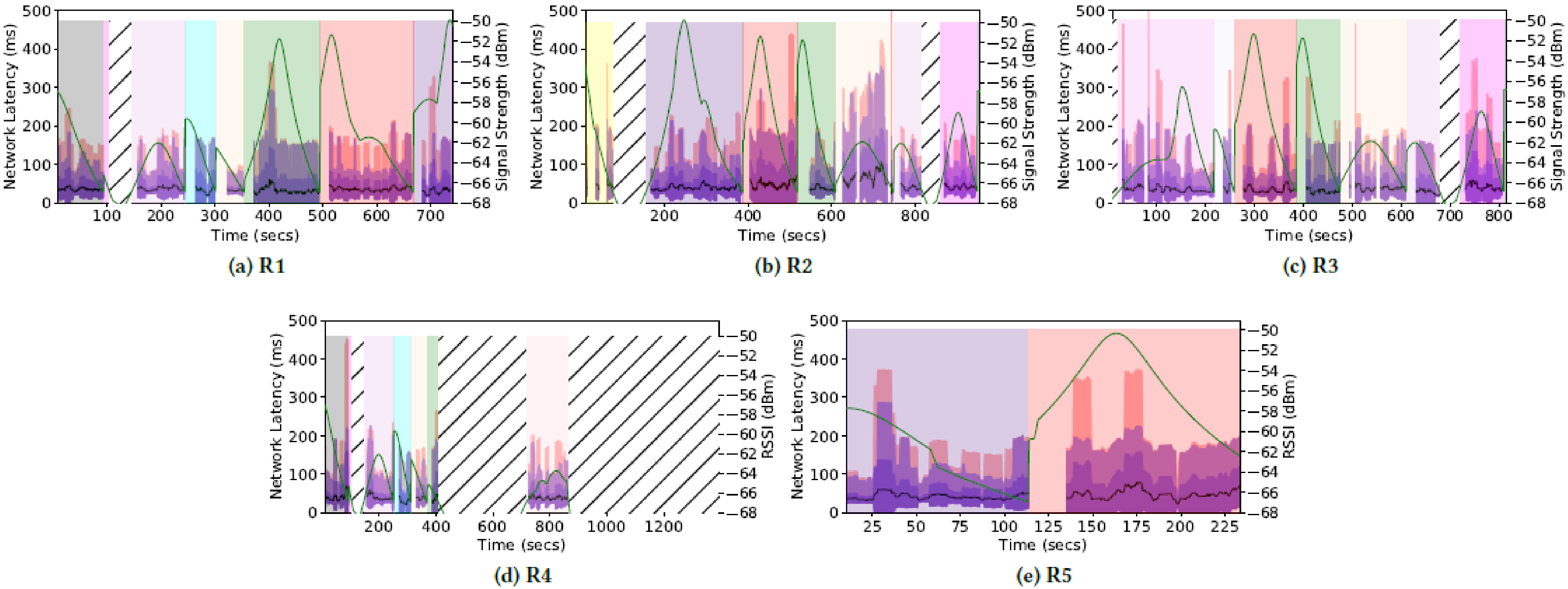
- Clients:
  - Android client:
    - Motorola Moto G4 Play – Quad-core CPU, 2 GB memory, 2800 mAh battery
    - Android version is 6.0.1
    - User walks at a brisk walking speed (expected to be close to 1.4 mps)
  - Linux client
    - Minnowboard Turbot - Quad-core CPU 1.91 GHz, 2 GB memory
    - Ubuntu 16.04.3
    - Creative VF0770 webcam, Panda Wireless PAU06
    - Connected to Watts Up Pro power meter for energy measurements
    - 2 fps (500 ms deadline), 224X224 frame, ≈30 KB size

Emulating a Geographic Region



Experimental Setup

- Route and MDC setup:
    - 18 WAPs and 4 MDCs
    - 30 ms ping latency among the WAPs
    - 5 routes



(a) R1 (b) R2 (c) R3 (d) R4 (e) R5

Observed mean, standard deviation, 95th and 99th percentile network latencies and expected received signal strengths on different emulated routes

Experimental Setup

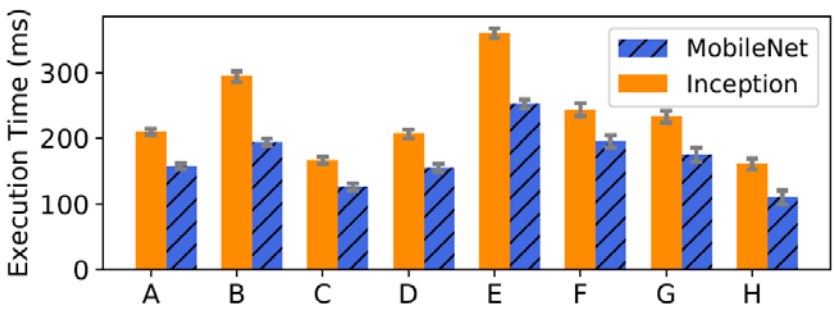
- Applications:
  - Real-time object detection algorithms: MobileNet and Inception V3
  - Application on Android device: Tensorflow Light 1.7.1
  - Application on Linux device: Ubuntu 16.04.3 container, Keras 2.1.2, Tensorflow 1.4.1

- Fog Setup: 4 MDCs - each has 4 servers (randomly assigned)
- Each server has medium to high interference load

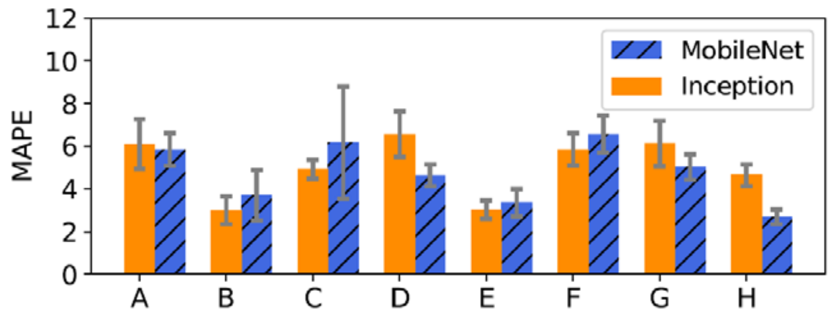
Server Configurations

Conf	sockets/cores/ threads/ GHz	L1/L2/L3 Cache(KB)	Mem Type/ MHz/GB	Count
A	1/4/2/2.8	32/256/8192	DDR3/1066/6	1
B	1/4/2/2.93	32/256/8192	DDR3/1333/16	2
C	1/4/2/3.40	32/256/8192	DDR3/1600/8	1
D	1/4/2/2.8	32/256/8192	DDR3/1333/6	1
E	2/6/1/2.1	64/512/5118	DDR3/1333/32	6
F	2/6/1/2.4	32/256/15360	DDR4/2400/64	1
G	2/8/1/2.1	32/256/20480	DDR4/2400/32	2
H	2/10/1/2.4	32/256/25600	DDR4/2400/64	1

Evaluations - Performance Estimation

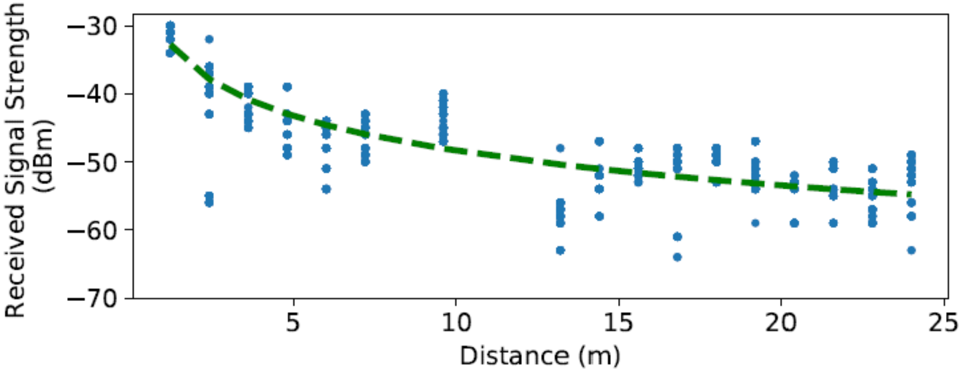


(a) Execution time in isolation



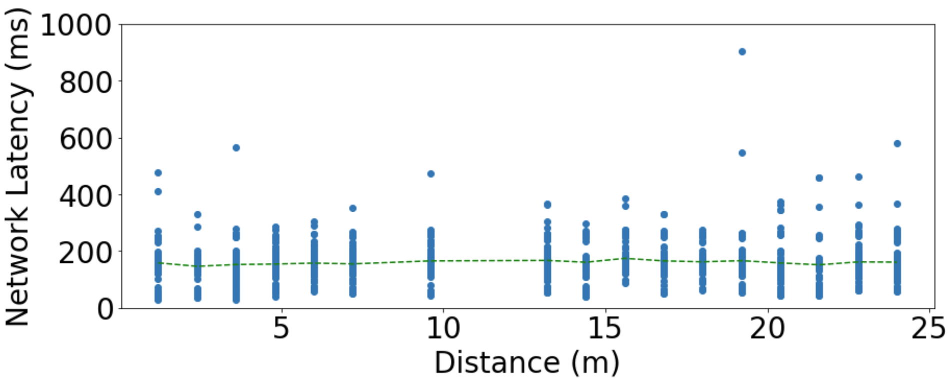
(b) Mean Absolute Percentage Error

Evaluations - Network Latency, RSSI, Distance



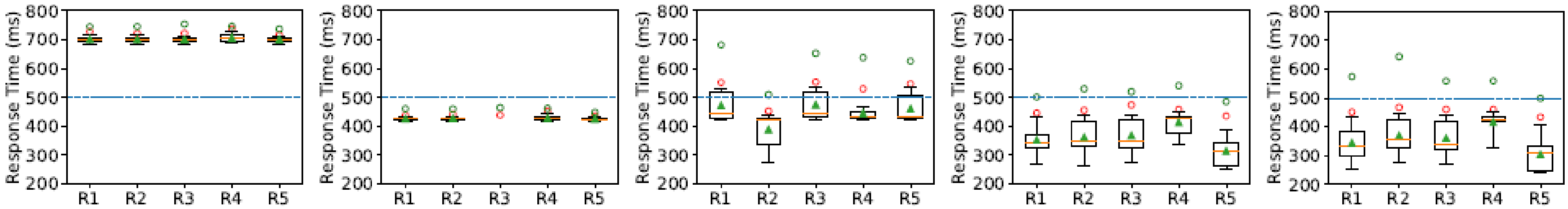
(a) RSSI

$\hat{\gamma} = 1.69$



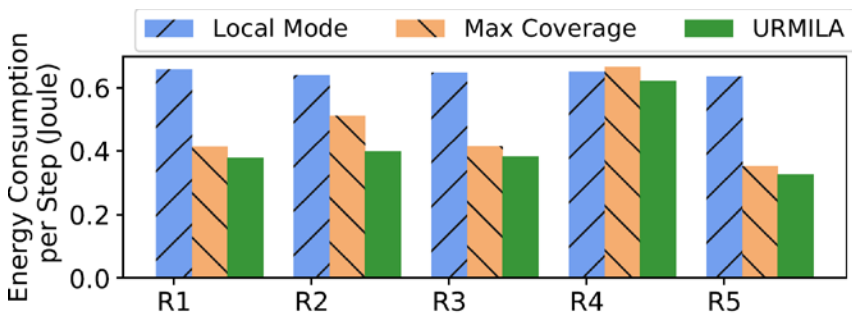
(b) Response Time

Evaluations – Comparison with Least Loaded & Max Coverage

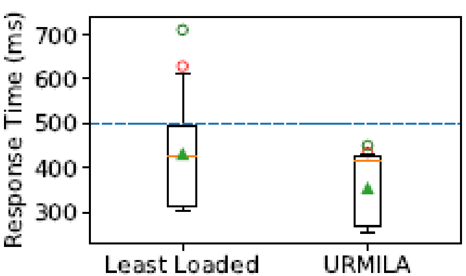


(a) Local Inception (b) Local Mobilenet (c) Max Coverage (d) Least Loaded (e) URMILA

Response time for different techniques on the routes. ○ and ○ depict 95th and 99th percentile respectively



Energy Consumption Comparison



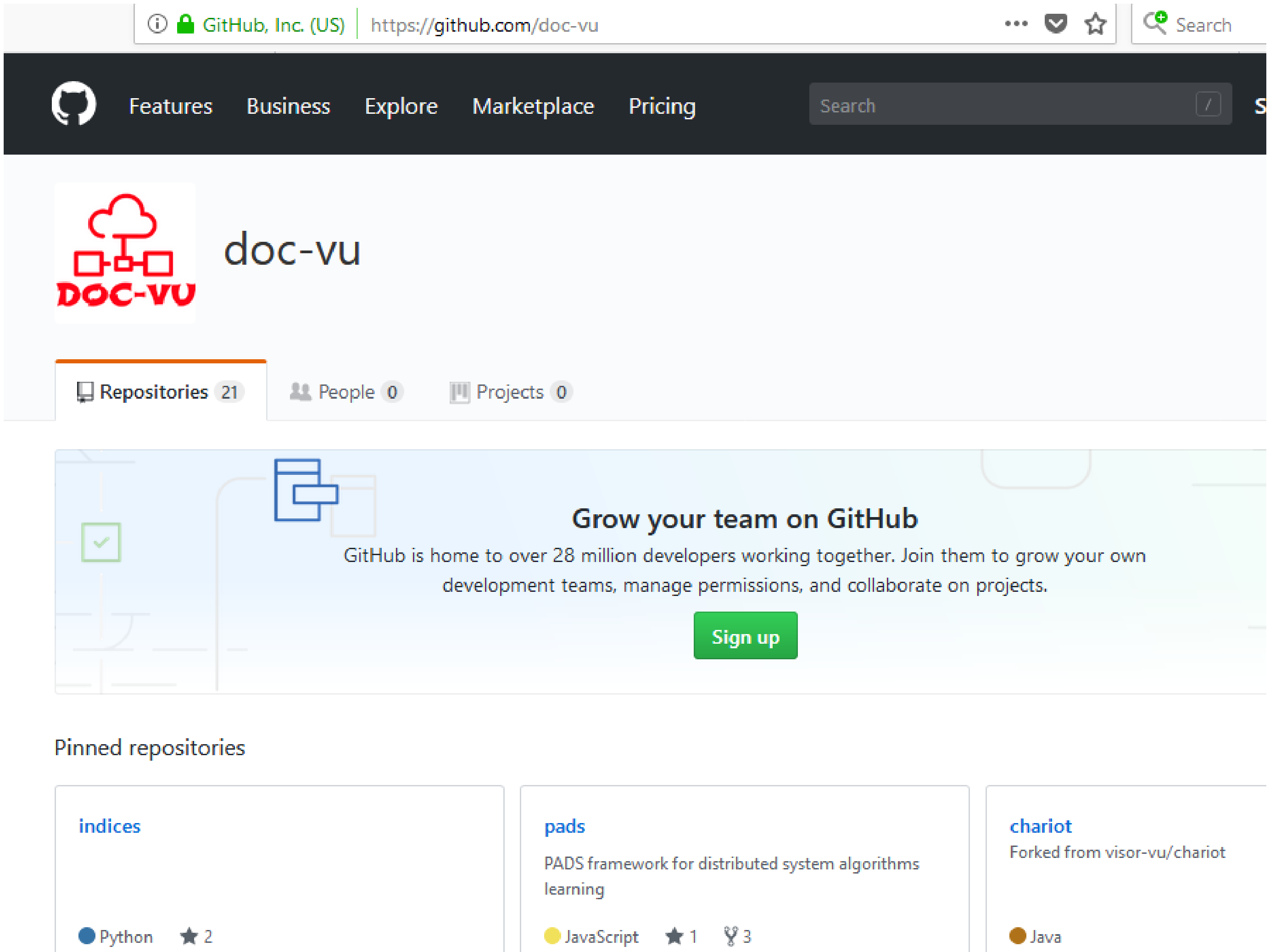
Response time comparison for route R5 when one of the WAPs is experiencing larger latency

SLO = 95%

Lessons Learned

- Performance interference problem for traditional cloud data centers extend to fog resources
- User mobility amplifies the problem further since choosing the right fog device becomes critical
- Executing the applications at all times on the edge devices is not an alternative due to severe battery constraints and limited resources
- URMILA validated for two client applications for cognitive assistance applications
- Solution needs to be advanced to account for wireless access point load, deviation from constant speed mobility model
- Serverless computing architecture fits nicely
- Can be extended to route selection, wireless handover policy
- Trust, privacy, billing, fault tolerance and workload variations are still not addressed

https://github.com/doc-vu



THANK YOU

QUESTIONS?