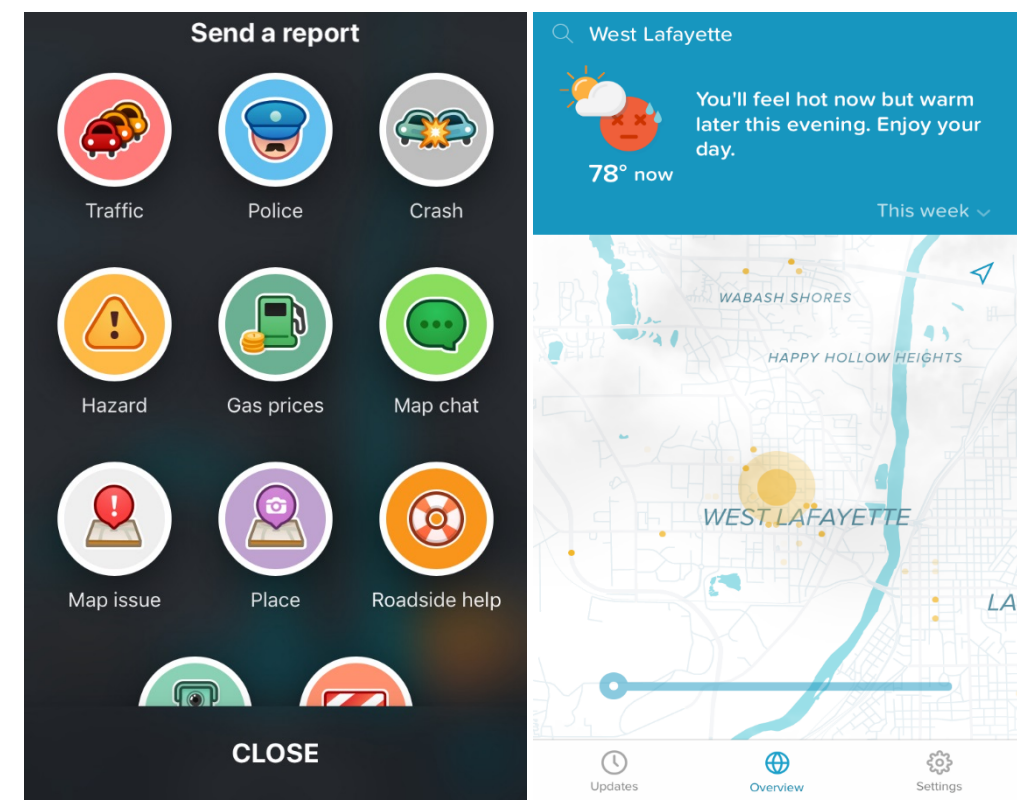


What is mobile crowdsensing (MCS)?

- Mobile crowdsensing is a new way to distribute computing and sensing workload to mobile devices.
- Participatory crowdsensing
 - Need user interaction with devices
- Opportunistic crowdsensing
 - Automatic sensing, collecting and sharing

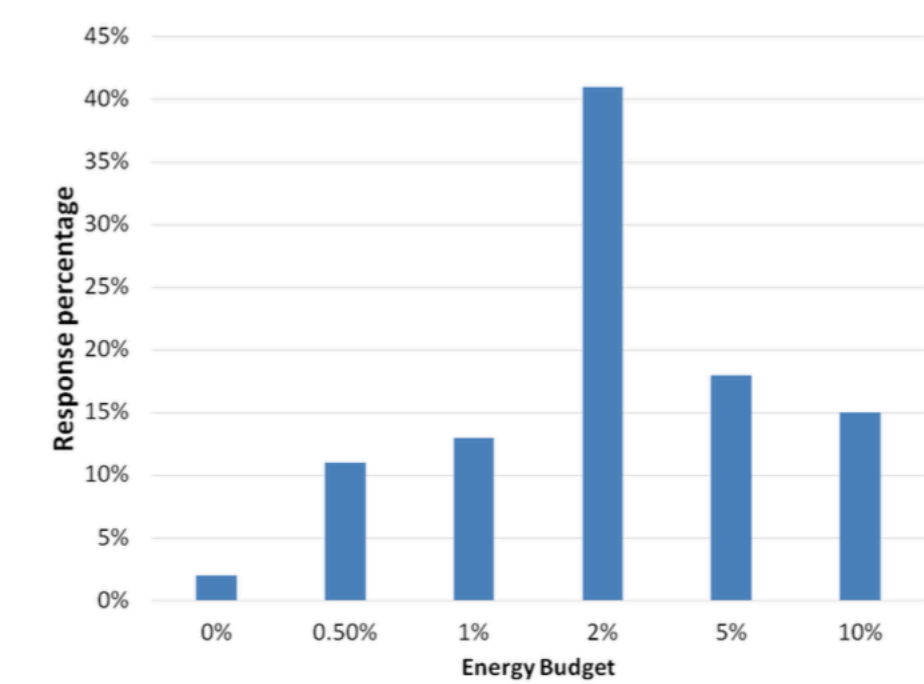


Why mobile crowdsensing is getting popular

- Ubiquitous usages of smartphones
- Sophisticated sensors embedded
- Accuracy and real-time delivery
- Lower cost than many dedicated infrastructures
- Fast network support

User's incentive

- Current MCS has problem encouraging more participants
 - One concern is energy drain
 - Second concern is privacy
- Most users would allow 2% energy drain for MCS activities
- But energy cost is higher in today's MCS solutions for most MCS tasks



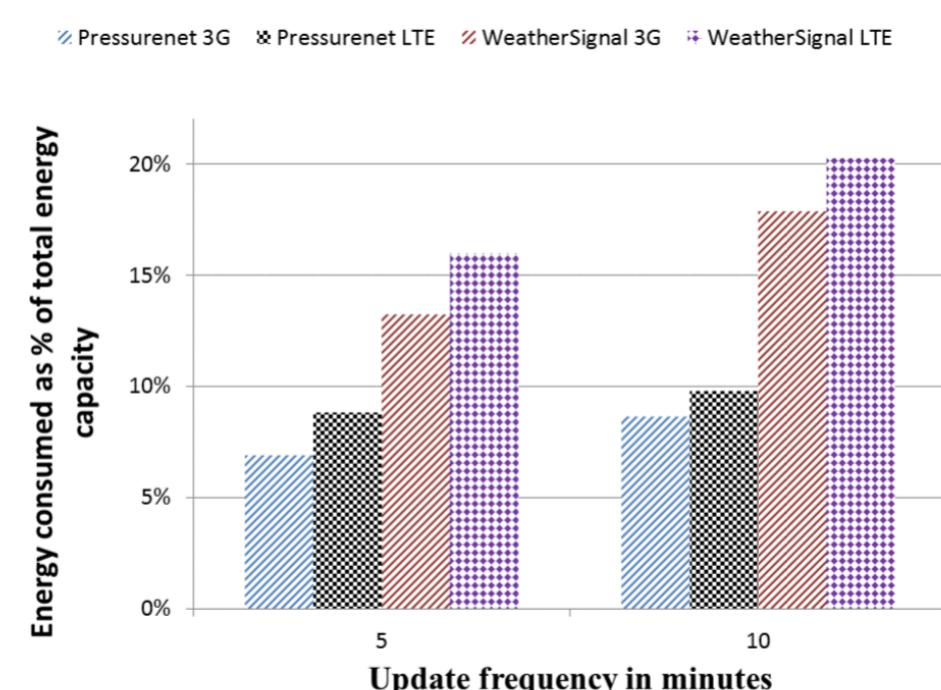
Component	Power Consumption
Accelerometer	21mW
Gyroscope	130 mW
Barometer	110 mW
GPS	176 mW
Microphone	101 mW
Camera	>1000 mW
LTE Module	594 mW to 1700 mW

Figure 1. Energy usage expectations from 109 university students. Majority of the users are willing to spend up to 2% Galaxy S4 (In the case of camera, the value of power consumption is dependent on what activity user is doing with camera.)

Periodic and Piggyback crowdsensing

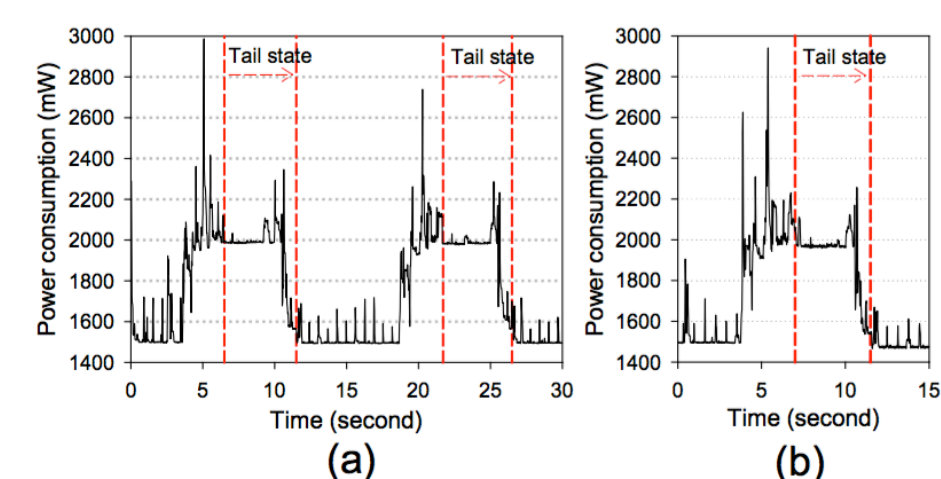
Periodic

- Periodically sample sensors and send sensor values to remote servers.
- Intuitive and easy to implement.
- Energy costly and large redundancy (some devices at the same location are scheduled the same tasks)



Piggyback crowdsensing

- Piggyback crowdsensing data packet onto regular traffic
- Predict user's app usage pattern to determine piggyback opportunity



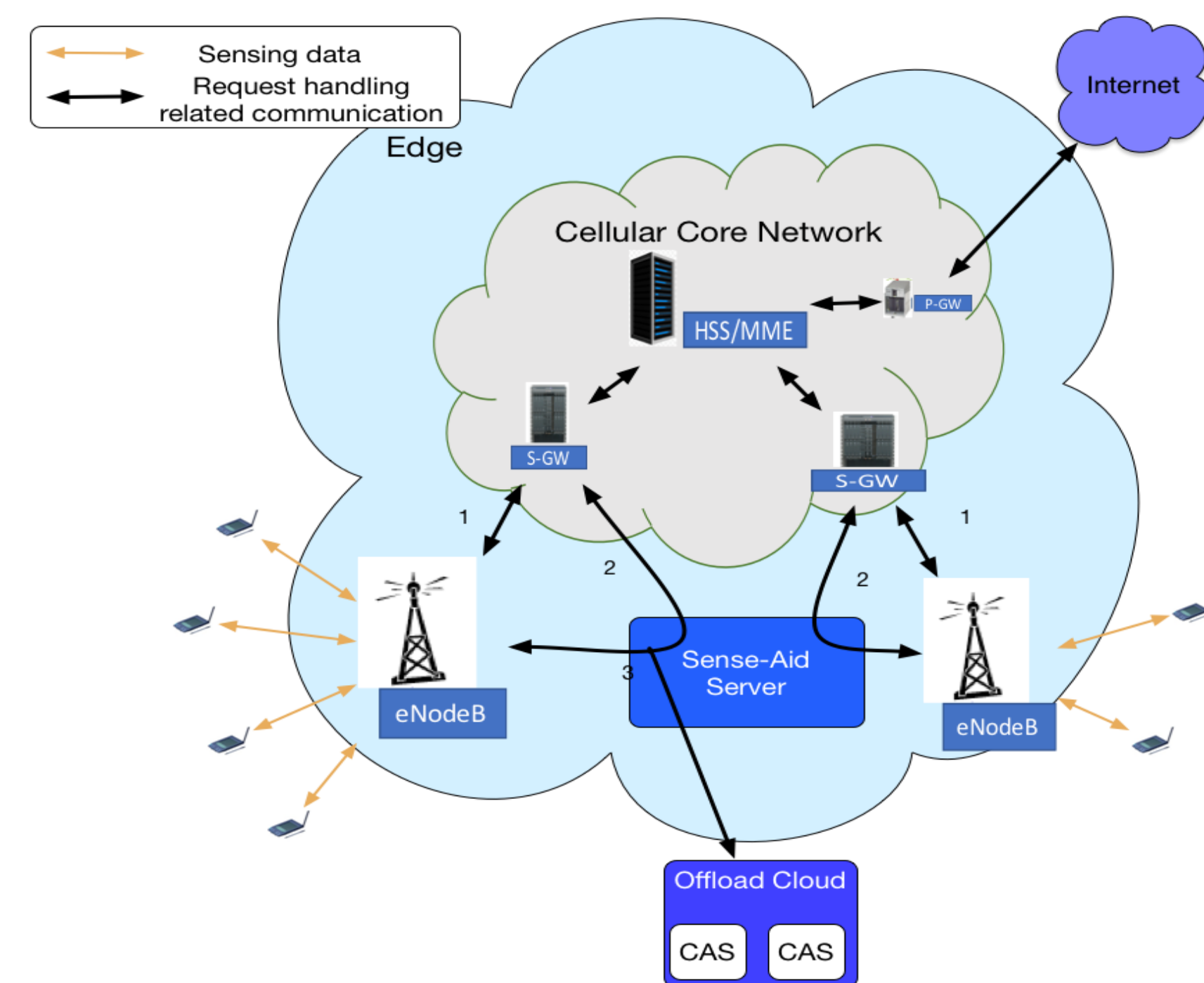
Problem Statement

- Current approaches to mobile crowdsensing activities need to be improved with respect to energy consumed and fairness
 - Mobile crowdsensing client: High energy consumption.
 - Mobile crowdsensing server: High overhead and low task distribution efficiency;
 - Need a generic mobile crowdsensing framework to schedule mobile crowdsensing tasks as well as pick mobile crowdsensing participants.

Solution Approach

- Sense-Aid is a distributed software framework for mobile crowdsensing
- Embedded at the edge of 4G LTE network:
 - Fetch device information from LTE Core network
 - Offload mobile traffic to Sense-Aid Server
 - Schedule mobile crowdsensing tasks
- Approach
 - Orchestrate among many devices
 - Piggyback mobile crowdsensing data packets
 - Utilize cellular radio state opportunity

What We Built: A mobile crowdsensing framework



- Crowdsensing application server (CAS) describes crowdsensing task to Sense-Aid server
- Sense-Aid server lies at the edge of 4G LTE network and schedules crowdsensing tasks onto mobile devices
- Mobile devices send scheduled sensor data back to Sense-Aid Server
- Sense-Aid server forwards crowdsensing traffic and reply result to CAS

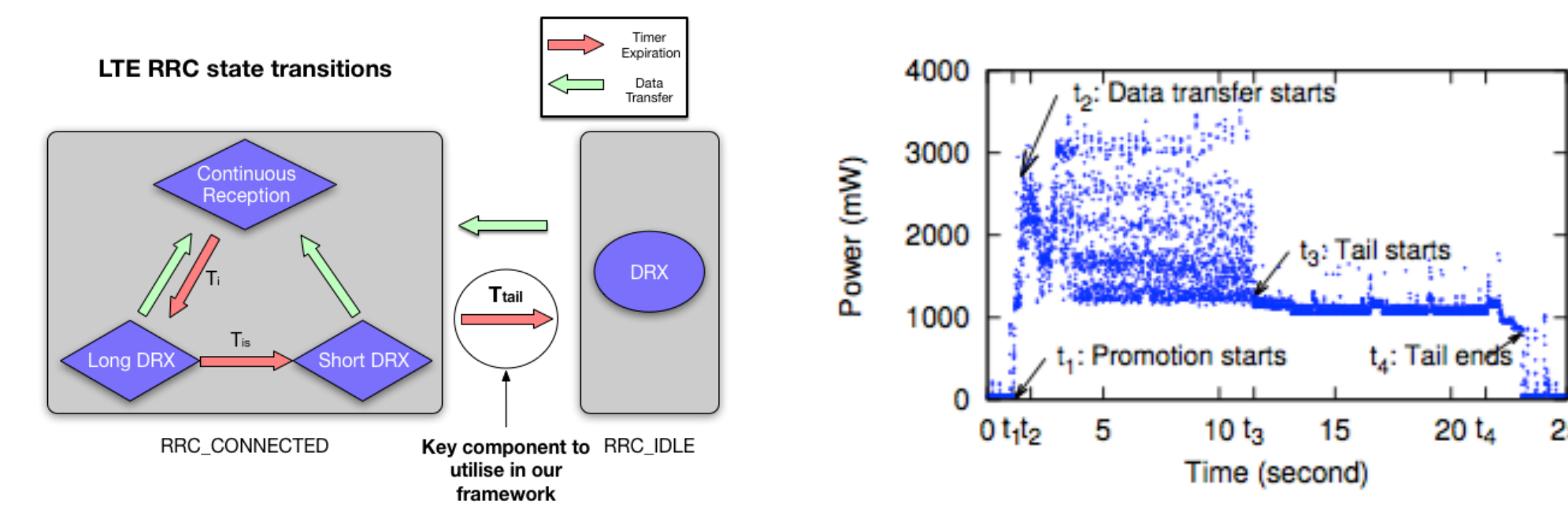
Our Contributions

- Propose a framework for developing crowdsensing applications
 - Saving device as well as energy for all mobile crowdsensing devices
 - Fairly choose crowdsensing devices
 - Deploy Sense-Aid server in 4G LTE network edge
 - Have a global view of crowdsensing devices' locations and cellular radio state
 - Evaluated by a user study to compare with baseline periodic sensing and state-of-art piggyback crowdsensing

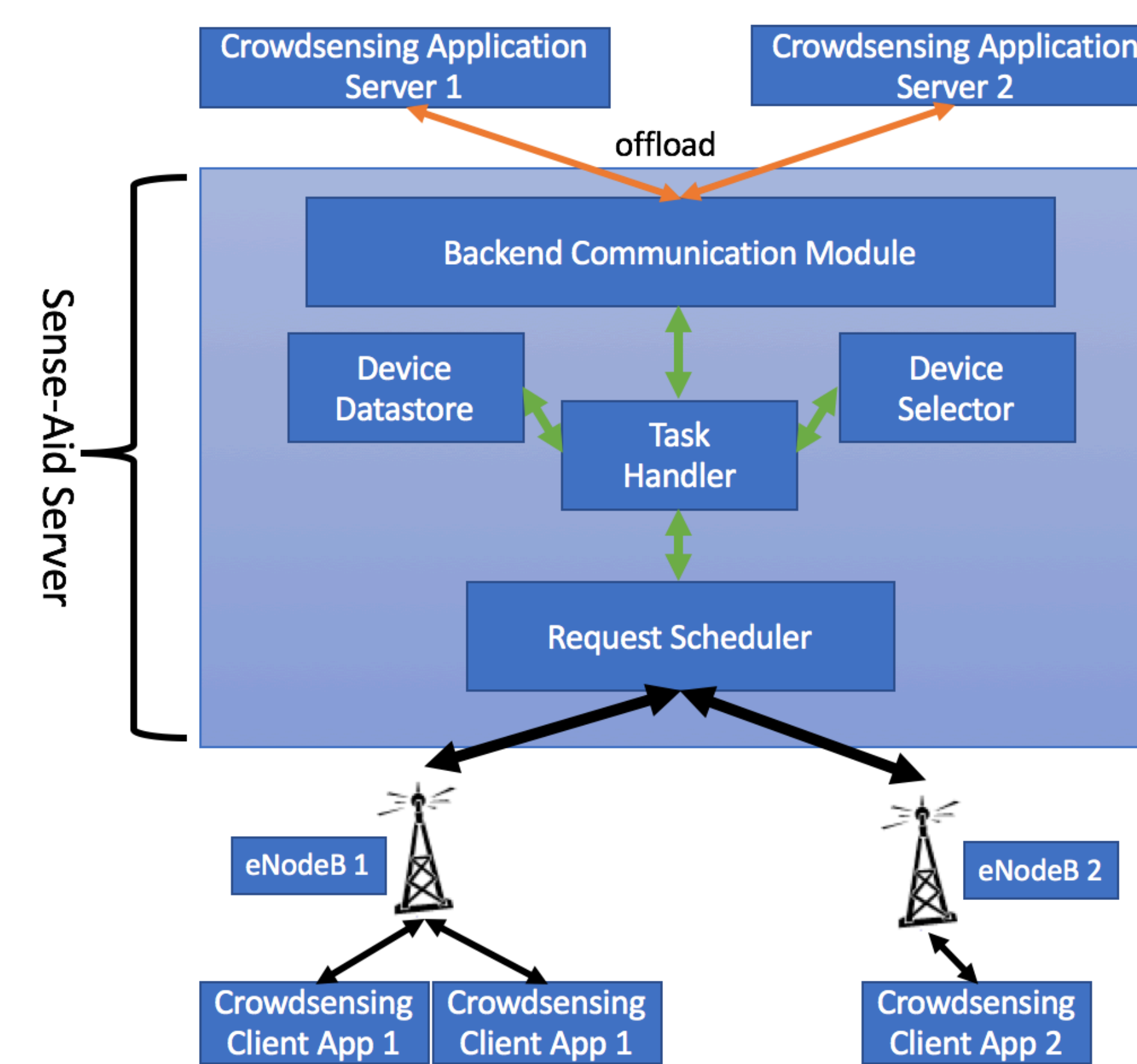
Cellular radio state

Cellular Radio state transition is energy costly

- State promotion is unnecessary to just send a few hundred bytes of crowdsensing data packet.
- Cellular tail has no data transmission between device and cellular tower.
- Tail timer is reset if client sends or receives data in tail time.



Design



- Request run queue: sort all MCS requests based on start time
- Pop request queue (e.g. 5 devices / Location 1): $n = 5$ (required number of devices)
- N : total number of devices that are qualified in the request region.
- if $n < N$, run device selection algorithm.
- Otherwise push request back to wait queue.
- Separate thread to periodically check if a request is satisfiable then move it to run queue.
 - E_i : energy consumed by device i for the crowdsensing task.
 - U_i : number of times device i has been used
 - CBL_i : current battery level in percentage
 - TTL_i : current timestamp - timestamp of most recent radio communication (set to 11.5 if $TTL_i > 11.5s$)

The scoring function is defined as follows.

$$Score(i) = \alpha \cdot E_i + \beta \cdot U_i + \gamma \cdot (100 - CBL_i) + \phi \cdot TTL_i$$

- Run the above scoring function over N
- Choose the lowest scored n devices
- Maintain overall fairness among N devices.
- Maintain low total energy cost. PCS and Periodic will choose all N devices but Sense-Aid only selects minimum required number of devices.

User study

Sense-Aid Basic

- Reset tail timer if data transmitted in tail time.

Sense-Aid Complete

- Not reset tail timer

Periodic sensing and Piggyback sensing

Tasks span 4 buildings at Purdue University,

The user study runs for 6 days.

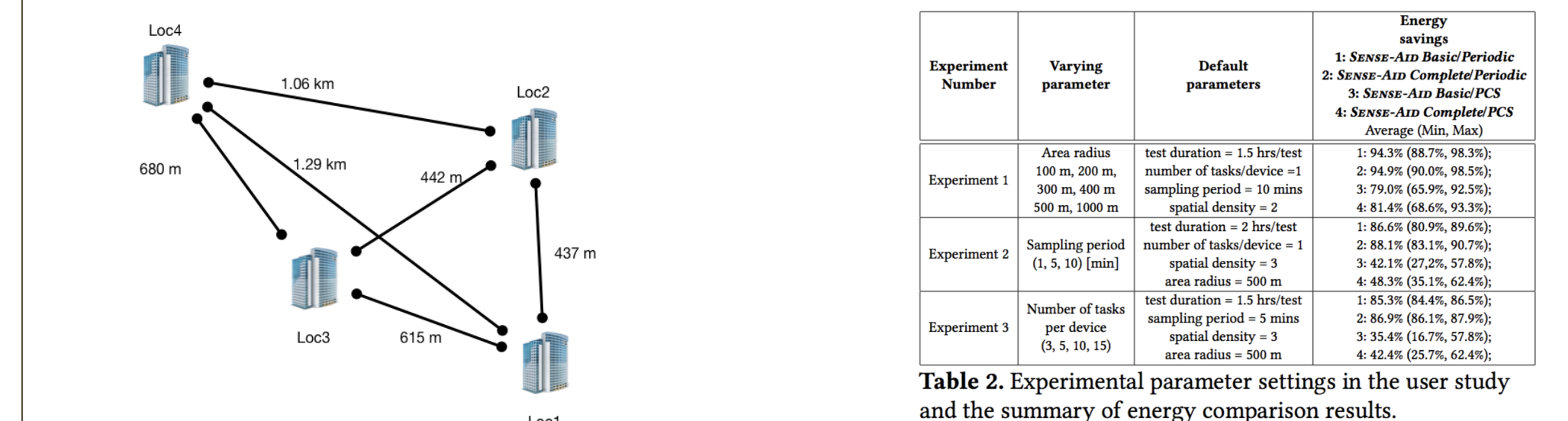
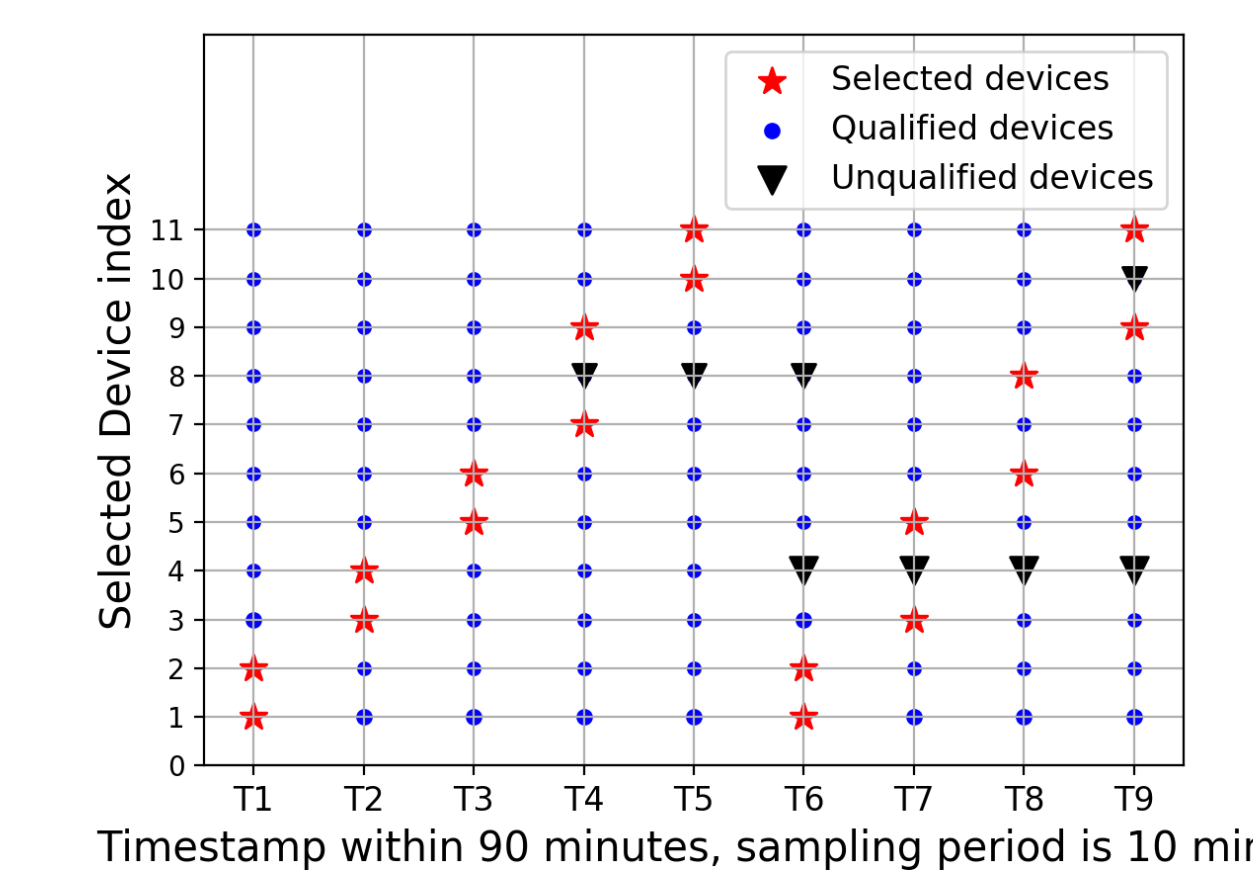


Table 2. Experimental parameter settings in the user study and the summary of energy comparison results.

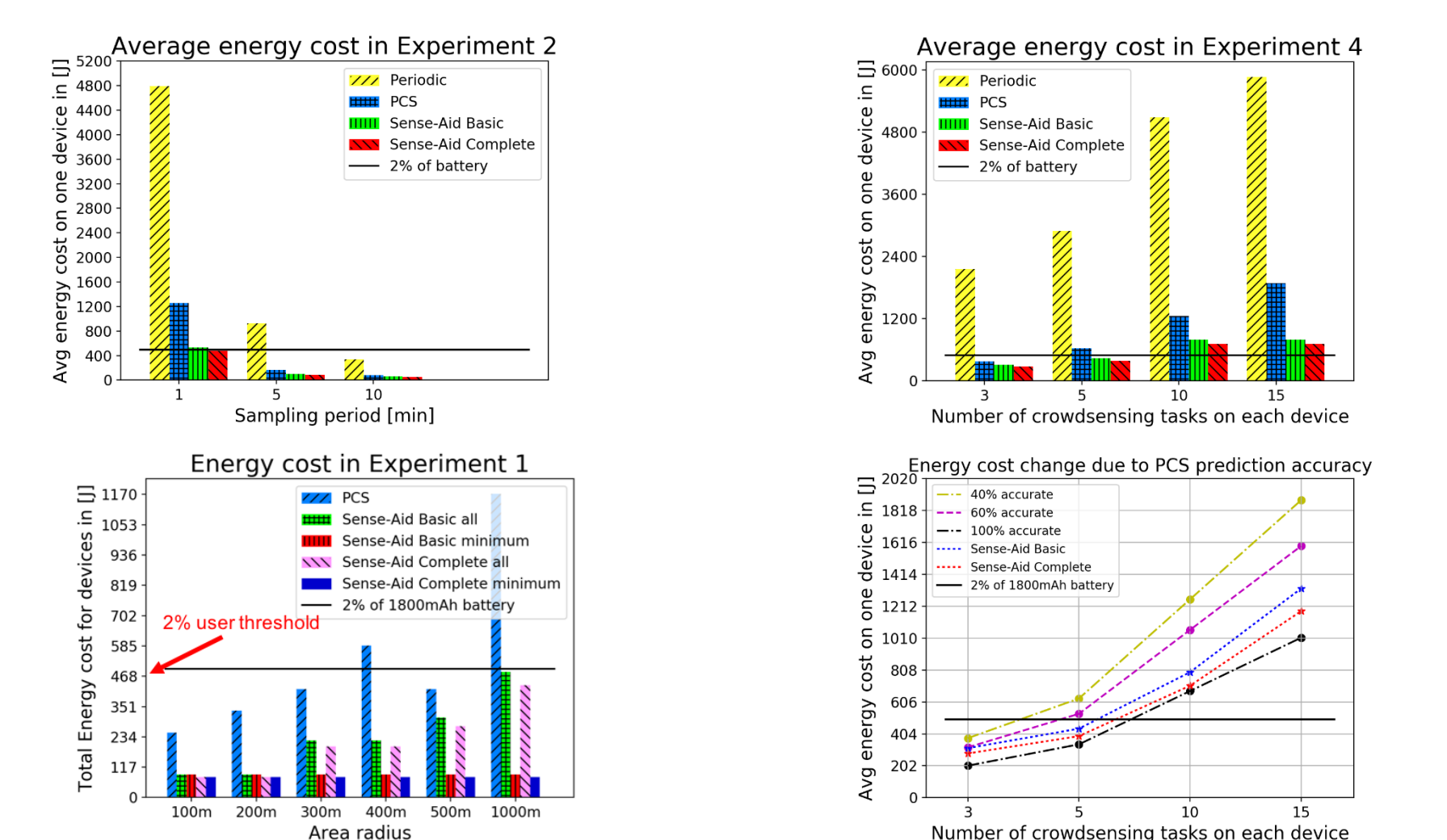
Energy: Fairness

Sense-Aid device selection with radius 1000m



- Fairness is maintained even as some users move out and into the request region.

Result: Energy Saving



- Sense-Aid out-performs Periodic and PCS in all scenarios
- Sense-Aid shows good scalability with respect to concurrent MCS tasks running on one device.
- Even with near perfect prediction of PCS, Sense-Aid still saves more energy than PCS.

Acknowledgments

This work was supported by NSF grant CNS- 1409506 and AT&T. The views expressed represent those of the authors and do not necessarily reflect the views of the sponsoring agency.

Reference

H. Zhang, N. Theera-Ampornpunt, H. Wang, S. Bagchi, RK. Panta, "Sense-Aid: A Framework for Enabling Network as a Service for Participatory Sensing," in Proceedings of Middleware '17, Las Vegas, NV, USA, December 11-15, 2017, 13 pages.