



Workshop on Big Data Analytics in CPS: Enabling the Move from IoT to Real-Time Control

Scalable and Energy-Efficient Cloud-Sensor Architecture for CPS

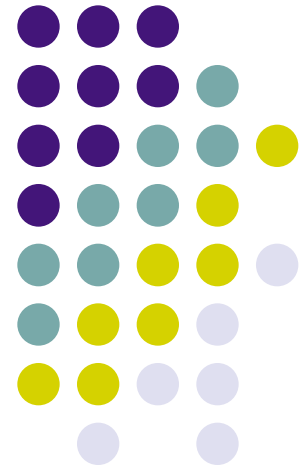
Sumi Helal and Yi Xu*

Mobile and Pervasive Computing Laboratory
Computer & Information Science & Engineering Dept.

University of Florida

www.icta.ufl.edu

* Currently with Google



UF UNIVERSITY of
FLORIDA
The Foundation for The Gator Nation



April 13, 2015
Seattle, WA

About Smart City Applications



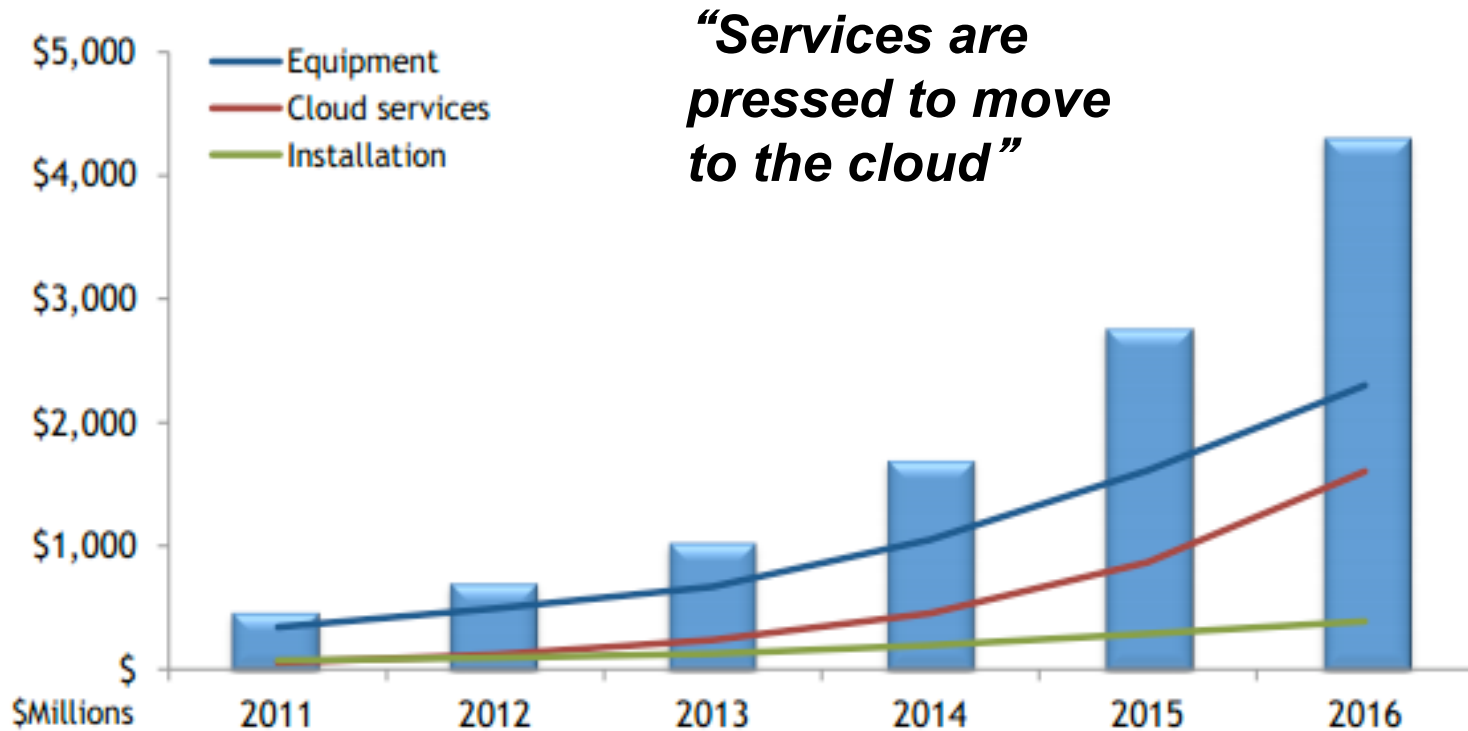
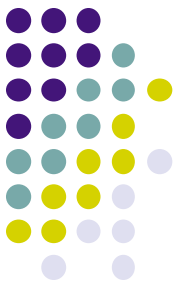
- Real-time sentience
- Real-time closed-loop control
 - Service, hardware or human actuations
- Ecosystem
 - Numerous stakeholders (citizens, city, government, private, ...)
 - Developers
 - Cyber infrastructure
 - Physical deployments

Why Cloud-Sensor Systems?



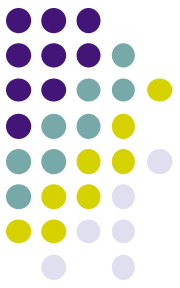
- As smart spaces and smart cities proliferate into a massive scale, applications & their sensor data will be *pressed to move to the Cloud*.
 - Economies of scale and highly anticipated reductions in service cost
 - A neutral and common platform where various stake holders can have access to the sensor-based services
- As a result, we will have Cloud-Sensor Systems
 - Cyber-physical systems spanning the clouds and the sensors.
- ***Dilemma!*** Sensors are *external* resources that cannot be farmed or provided on demand by the cloud

Cloud Involvement in the Smart City: Energy Sector



Global Home Energy Management Revenues by Segment (2011-2016)
(OnWorld Report on Smart Homes)

Smart Cities – An economical pressure and an impending development lacking *Cloud-Sensor Infrastructure*



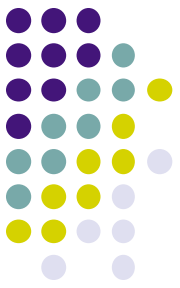
U-City, Korea, A 7-years, \$25B Smart City Investment




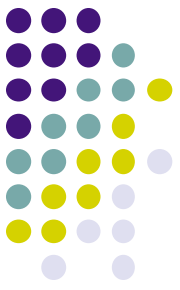
Talk Overview

- **Why cloud-sensor systems?**
- **Key challenges**
- **Our approach**
 - Optimization-centric architecture: **Cloud-Edge-Beneath (CEB)**
 - Programming models under CEB
 - Bi-directional Waterfall Optimization Framework
- **Agenda for future research**
- **References**

Key Challenges in Cloud-Sensor Systems



- How can we support open-ended, friction-free, **continuous integration** of sensors and devices into the cloud? *Alfas*
- How can we **program the cloud-sensor** system? Do we have an ecosystem that can.. unleash powerful app development? 
- How can we use the cloud wisely? How can we keep the **cloud economically scalable**?
- How to sustain high system dependability despite **sensor energy limitation**?



Cloud Scalability Challenge

- Extensive external interactions between cloud services and the physical sensors.
- Expensive cloud “attention”, not only per sensor, but per each sensor duty cycle.

If sensors push data once every minute, then millions of sensors will produce billions of sensor-cloud interactions, daily.

- Requires tremendous processing power, memory resources and huge incoming/outgoing cloud traffic.
- As a result, the cloud economies of scale per sensor will not stand. ***That is, even though the Cloud is elastic, its auto-scaling rate will be cost-prohibitive, if it is left unrestrained.***



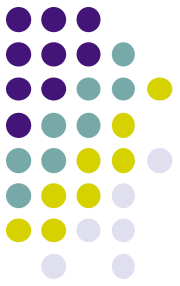
Sensor Energy Challenge

- **Limited energy of sensor devices**

- Limited battery capacity, or inadequate harvest supply rate.
- In smart city scenarios, a sensor may be queried by multiple *independent* cloud applications each of which requires periodical readings of the sensor (classical redundant data acquisition problem).
 - Deplete sensor energy rapidly → unreliable and unavailable sensor-based services.

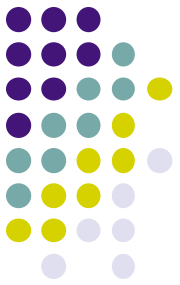
Without involving the applications in the optimization problem, minimizing sensor energy consumption will remain blind-sided and limited.

From Challenges View to the Value View: What is the Value of the IoT?



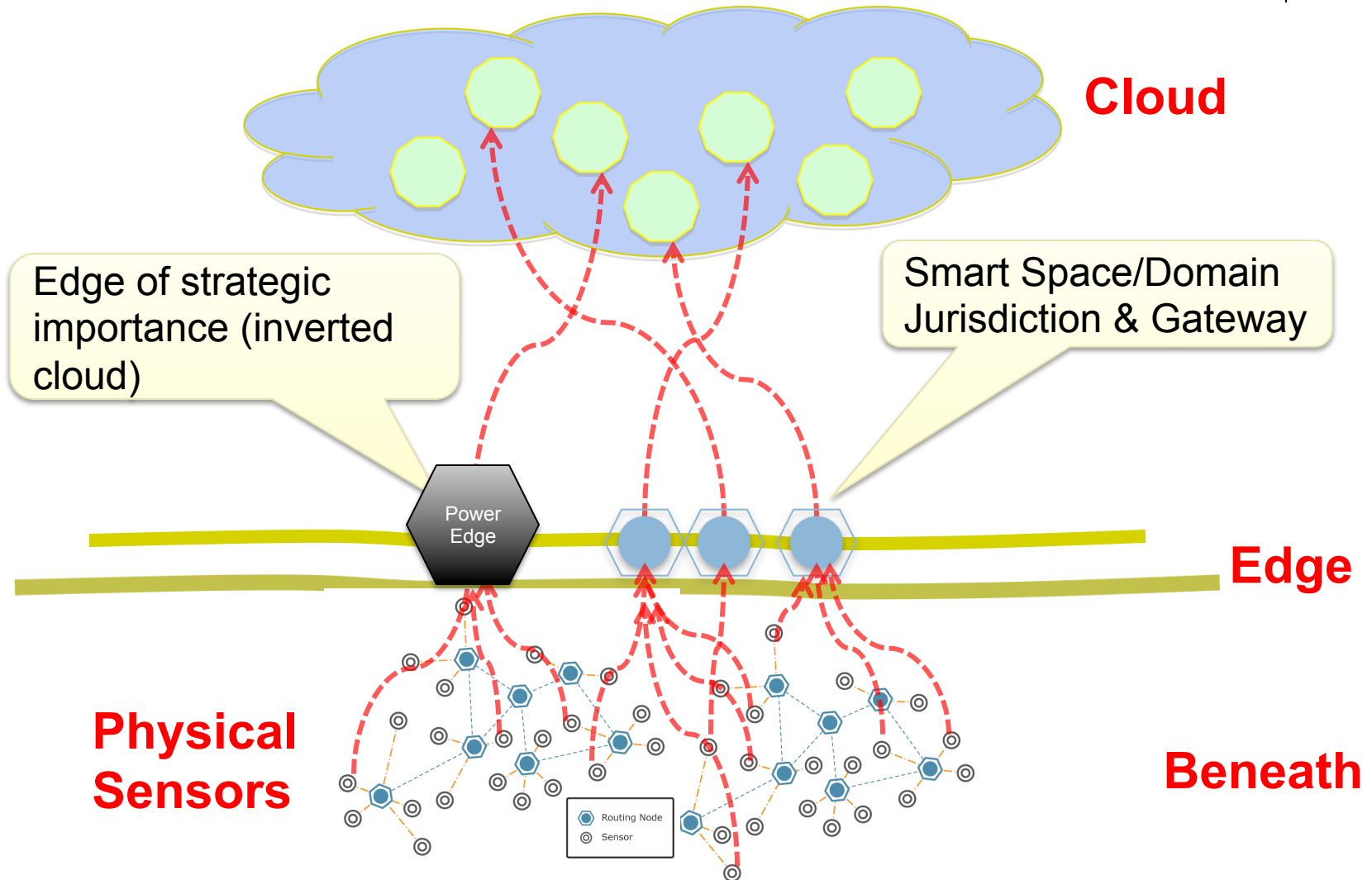
- As the value of the network was once estimated, the value of the IoT will need to also be estimated.
- The value of the IoT will depend on how programmable and manageable the IoT is in face of its massive growth rate, heterogeneity, and stringent energy requirements.

$$V_{IoT} \sim \frac{P}{M}$$

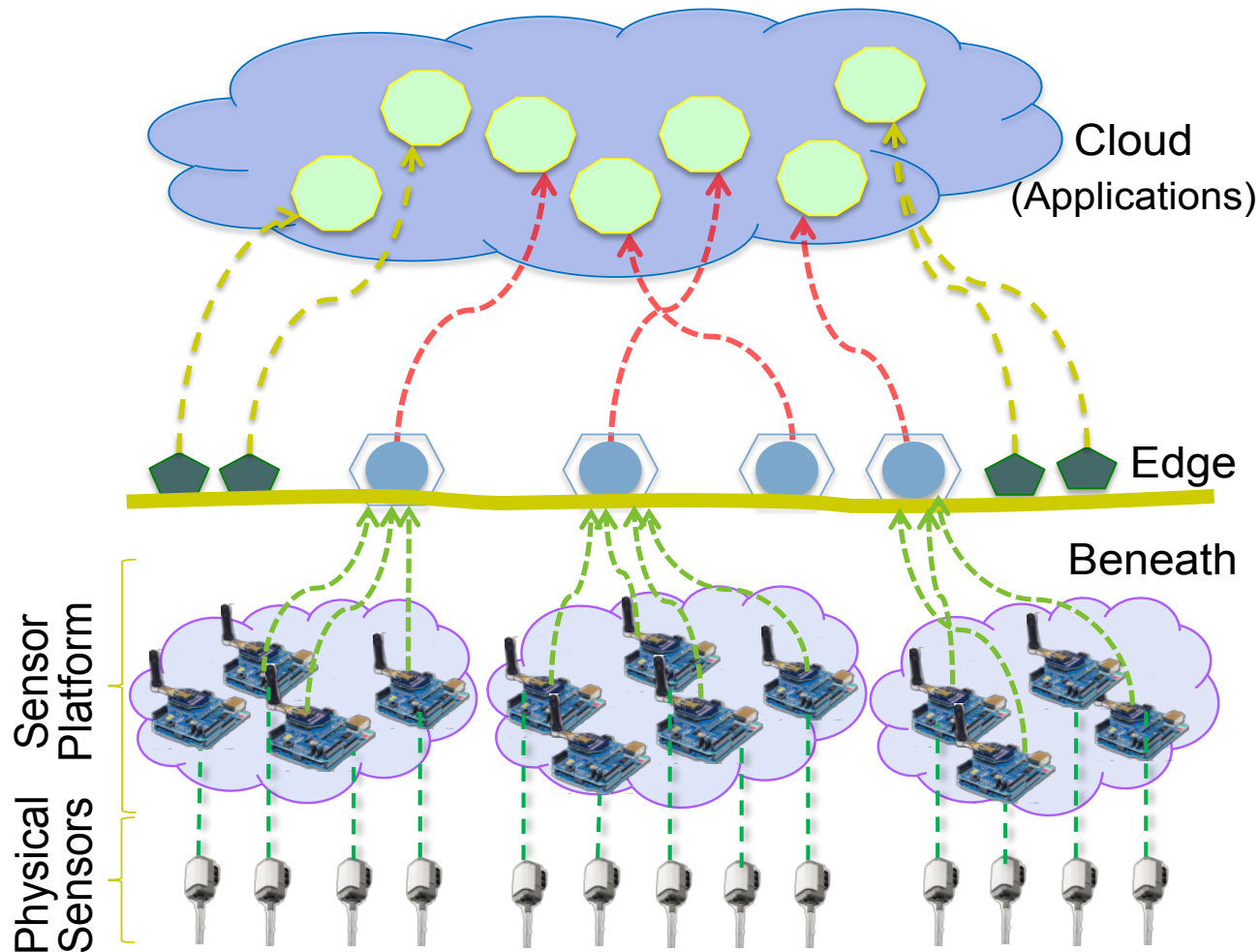
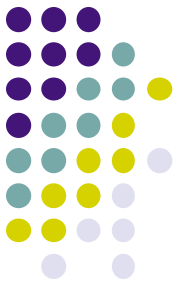


Cloud, Edge and Beneath Architecture (CEB)

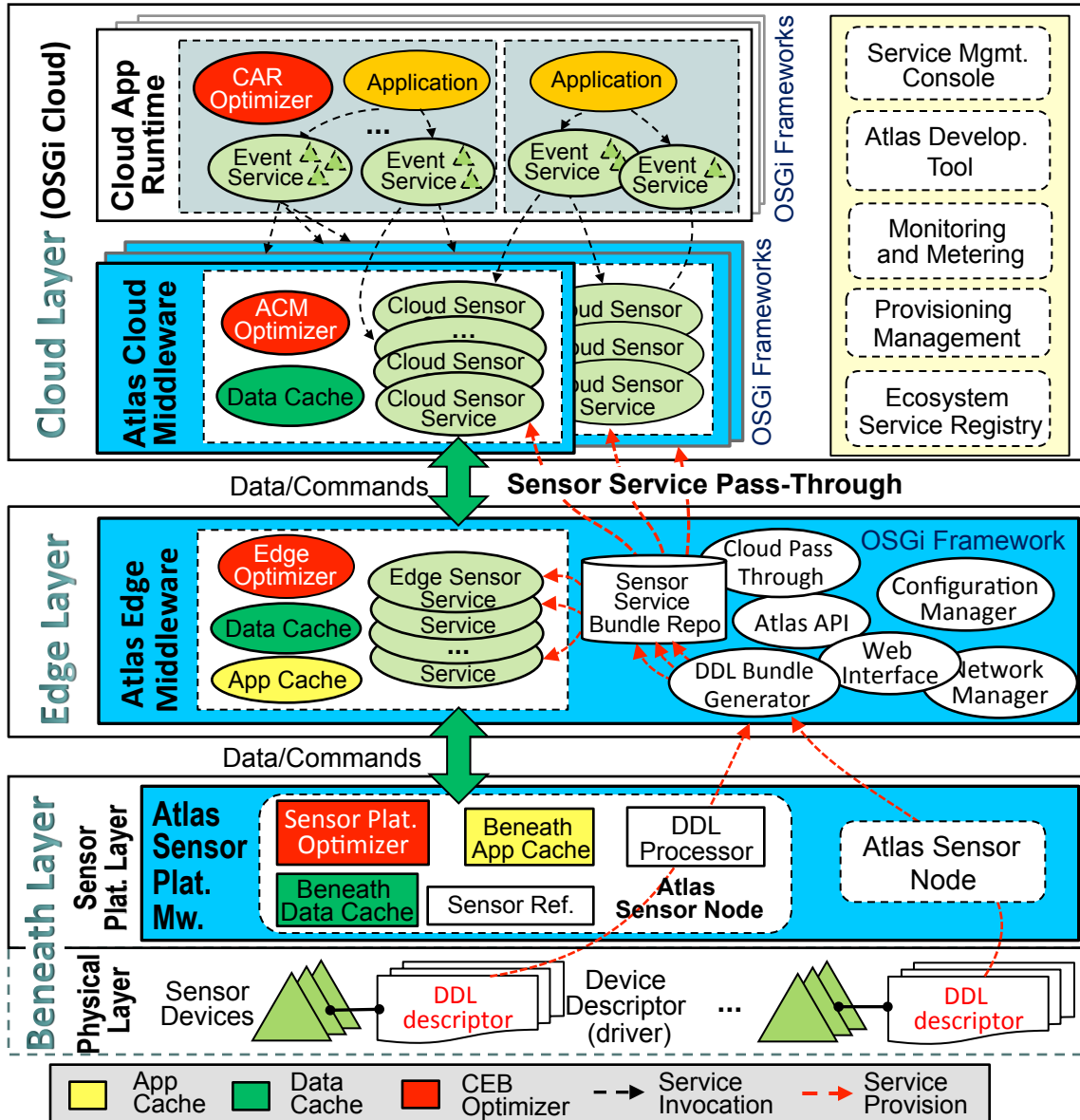
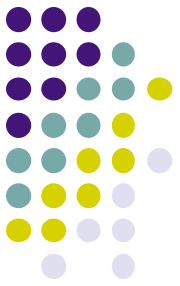
How Sensors could be Connected to the Cloud?



Sensors & Sensor Platforms



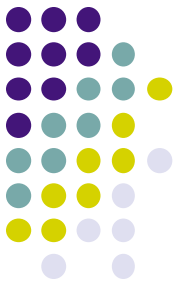
Cloud, Edge & Beneath (CEB)



Goals of CEB:

- Autonomic Integration platform
- Programming platform & Ecosystem
- **Optimization enabling platform:**
 - **Data Caching**
 - **App Caching**
 - **Energy Efficiency**
 - **Sentience Efficiency**

CEB Optimization Principles



- **Energy Efficiency**

- Degree to which total energy used to provide sensor readings is minimized over a period of time

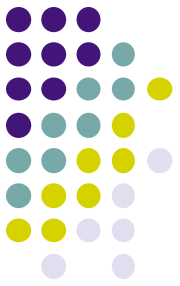
- **Sentience Efficiency**

- Degree to which unnecessary sentience of a sensor reading is minimized over a period of time

- **Suppressed System Dynamics**

- Minimize all movements of requests and data (requests by apps down to sensors, and data by sensors up to the apps).

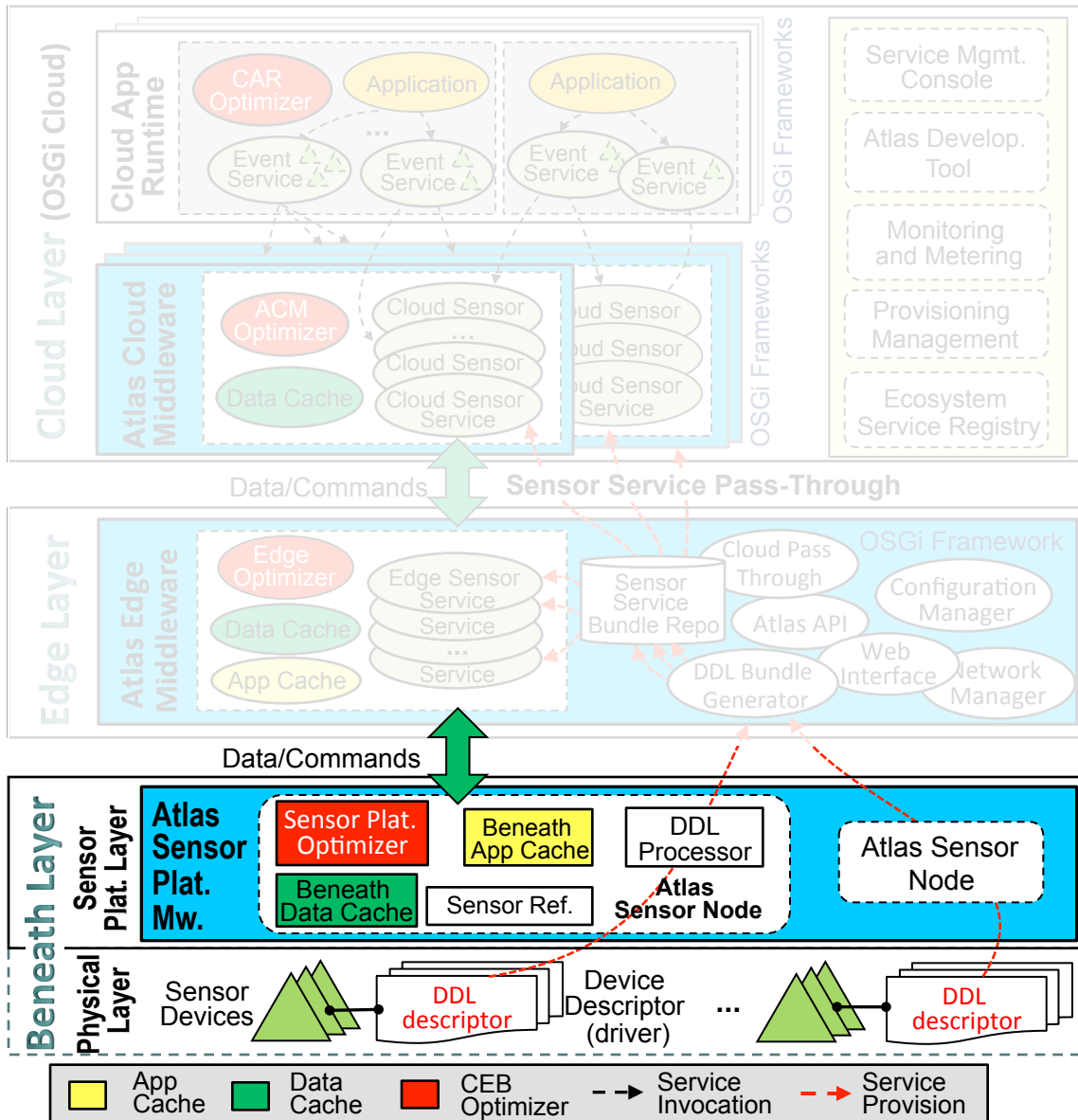
Beneath Layer in CEB



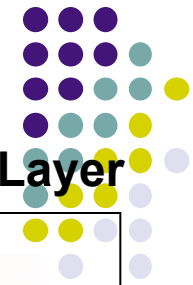
Atlas

Beneath Layer:

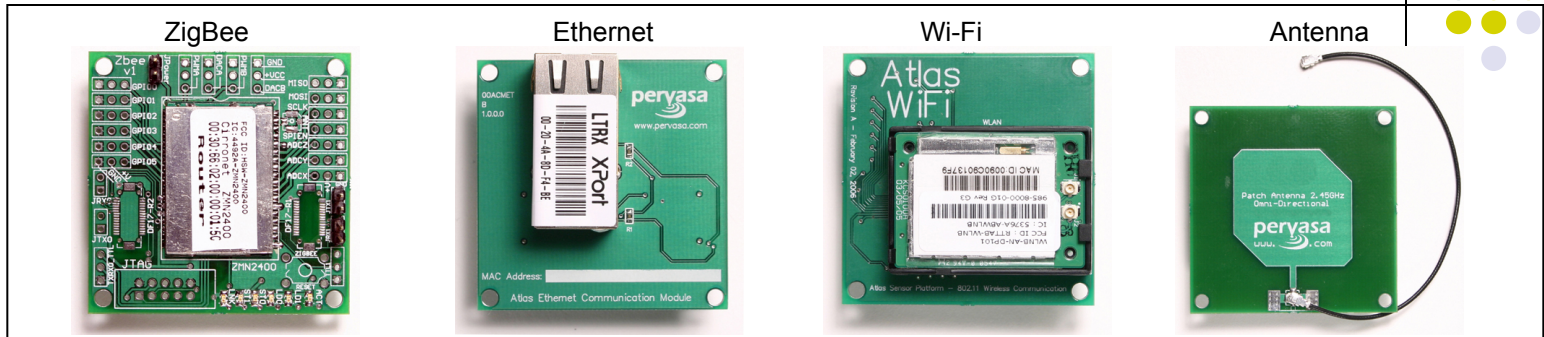
- CEB is built on top of Atlas which is an implementation of the service-oriented device architecture (SODA)
- Physical layer refers to the sensors and their descriptions written in Device Description Language (DDL).
- Sensor platform layer hosts first prong of the Atlas middleware which is responsible for sensor integration, configuration, init., data acquisition, caching, and device control.



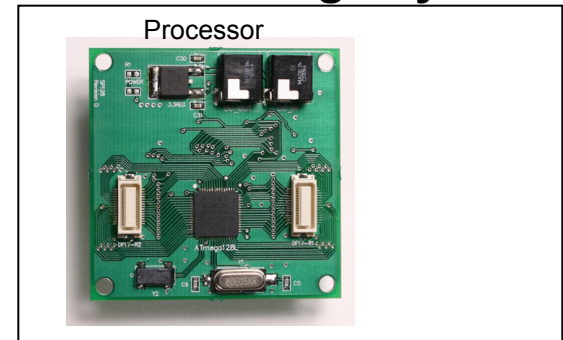
ATLAS Sensor platform & Middleware



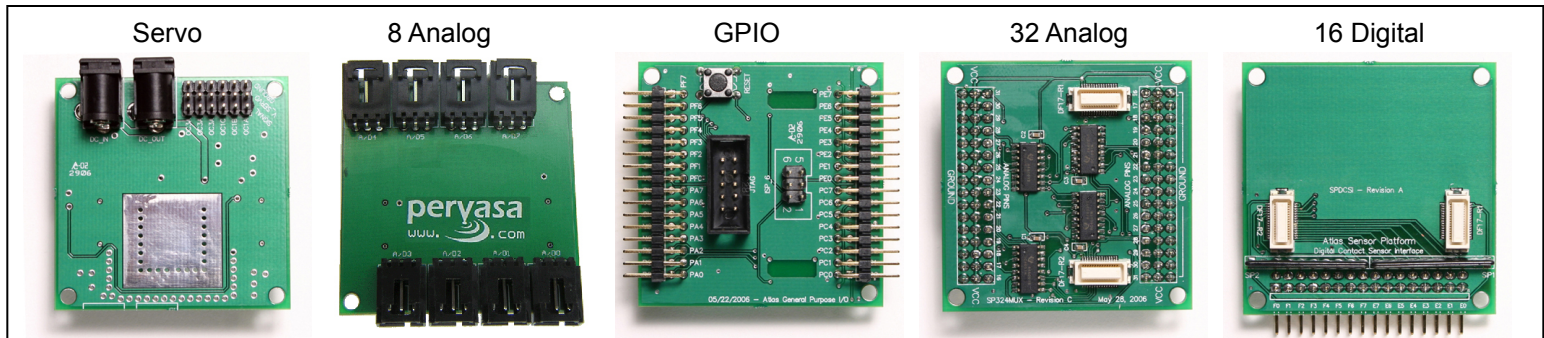
Atlas Communication Layer



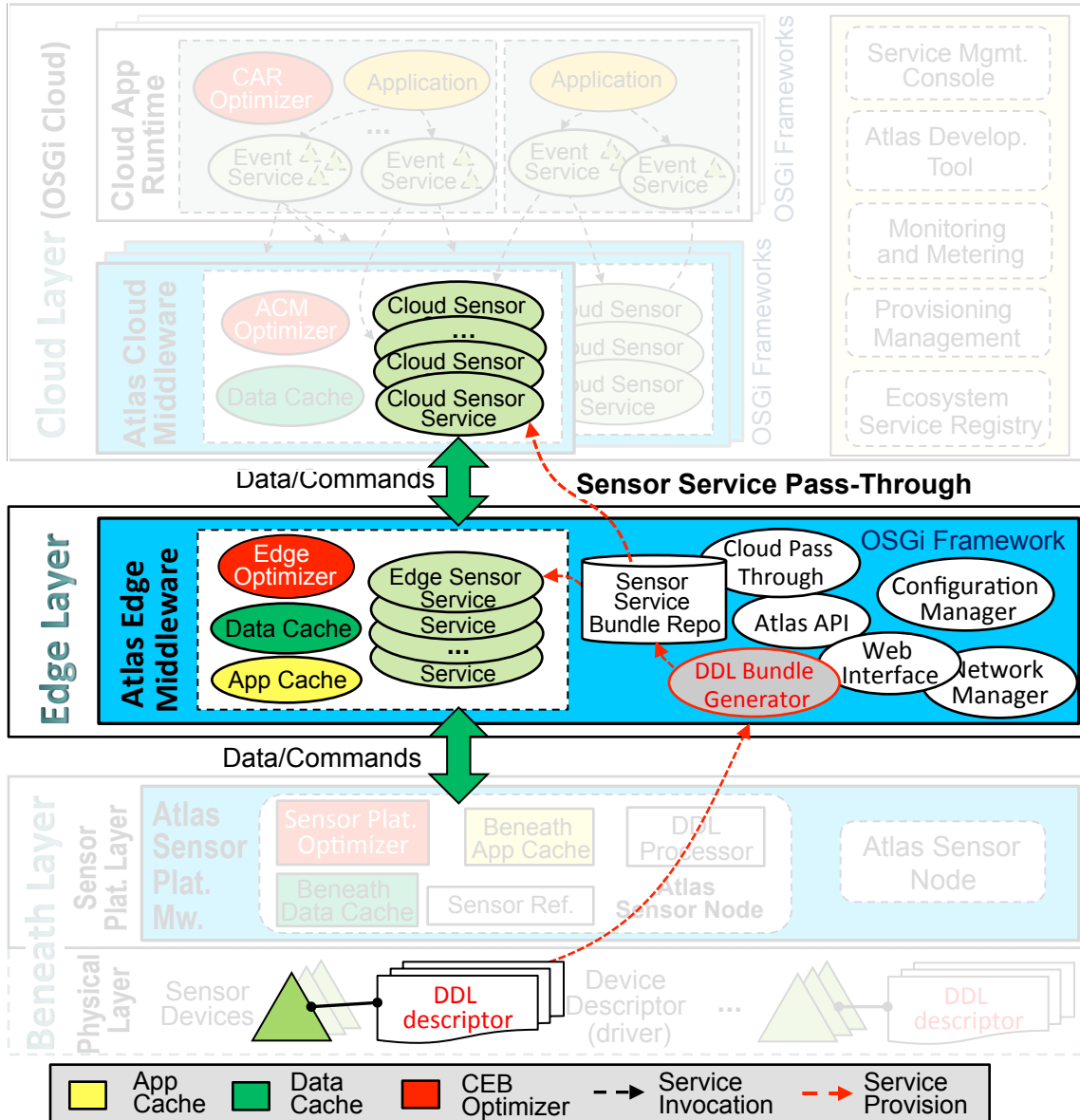
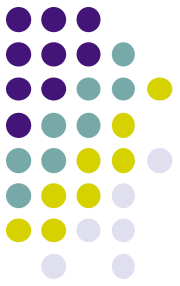
Atlas Processing Layer



Atlas Device Interface Layer



Edge Layer in CEB

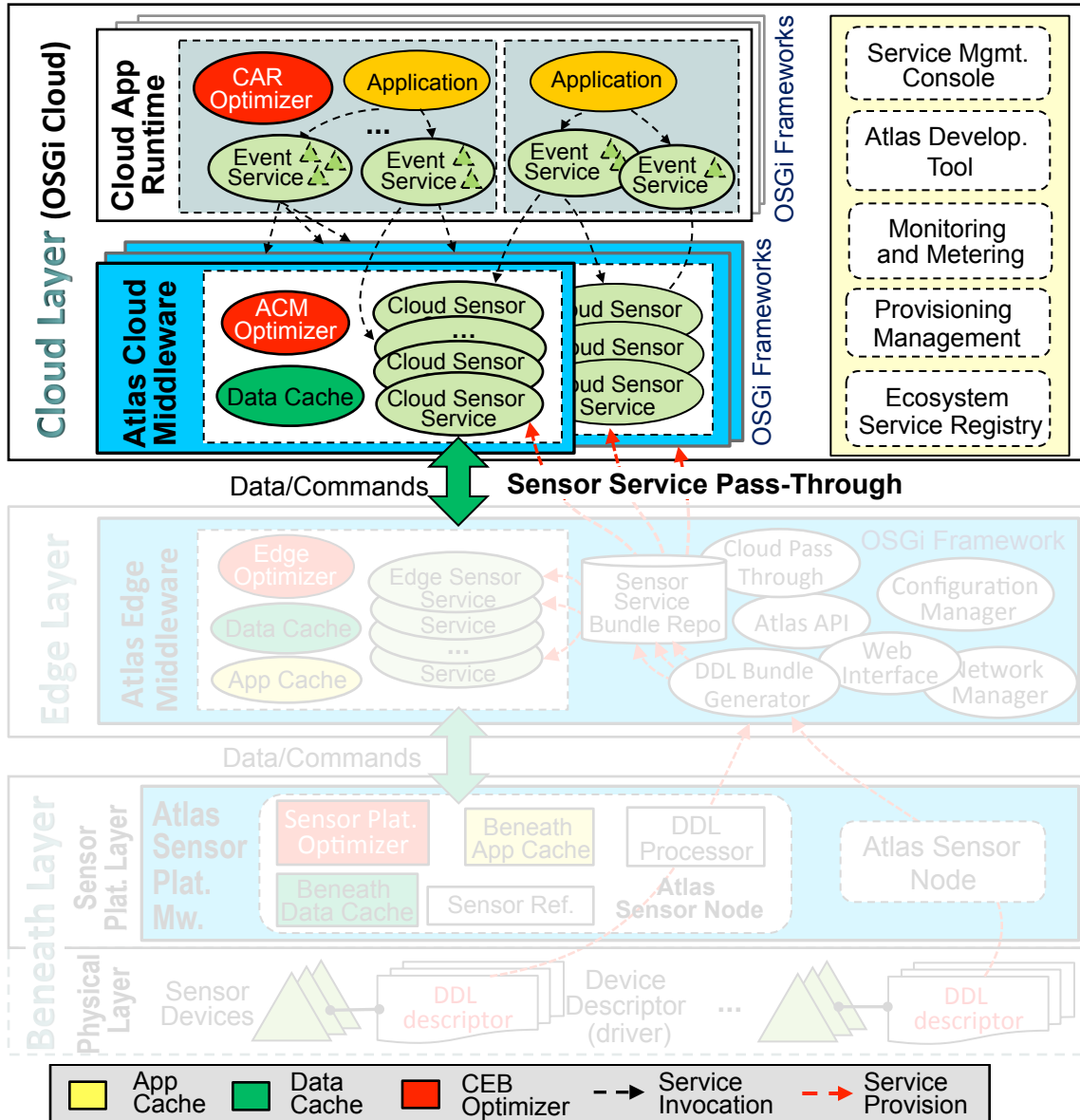


Edge Layer:

- Runs second prong of the Atlas middleware which uses **OSGi** as its basis to provide service discovery and configuration.
- The bundle generator creates a pair of software bundles for each sensor: 1) *edge sensor service* to be hosted at the Atlas edge middleware, 2) *cloud sensor service* to be uploaded to the Atlas cloud middleware.

Device Description Language
<http://www.icta.ufl.edu/atlas/ddl/>

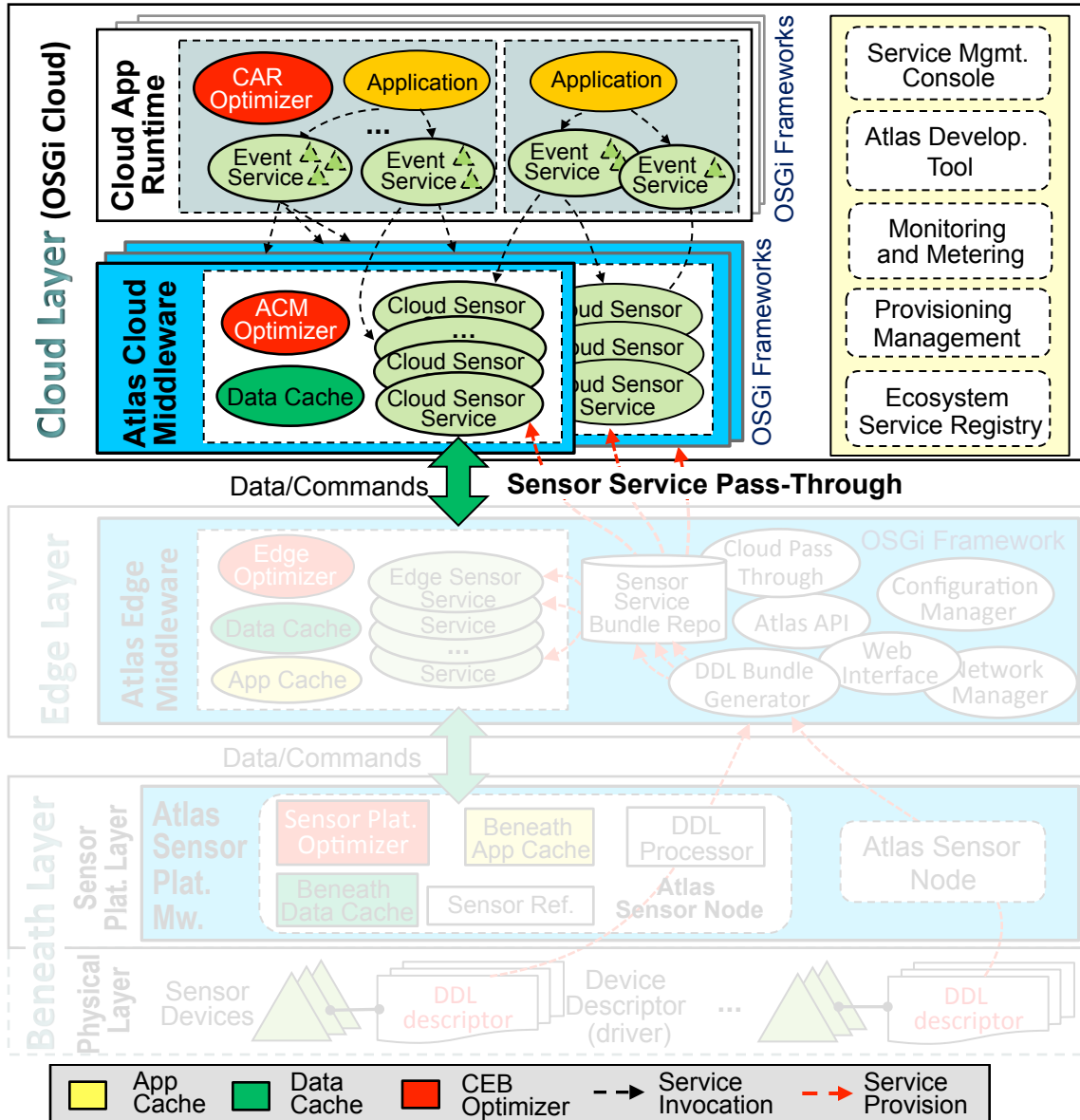
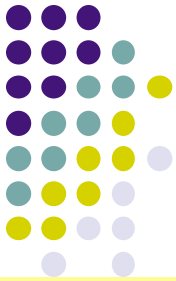
Cloud, Edge & Beneath (CEB)



Cloud Layer:

- Based on **OSGi Cloud** in which application is composed by loosely-coupled modules as OSGi services hosted at a distribution of cloud nodes.
- Third prong of the middleware
- Provides cloud-wide discovery, configuration and 'wire-up' of services across different OSGi frameworks in the dynamic cloud environment into applications and services.

Cloud, Edge & Beneath (CEB)

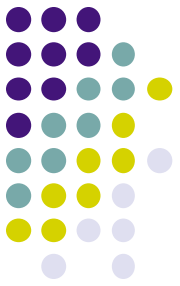


Atlas Cloud Middleware (ACM):

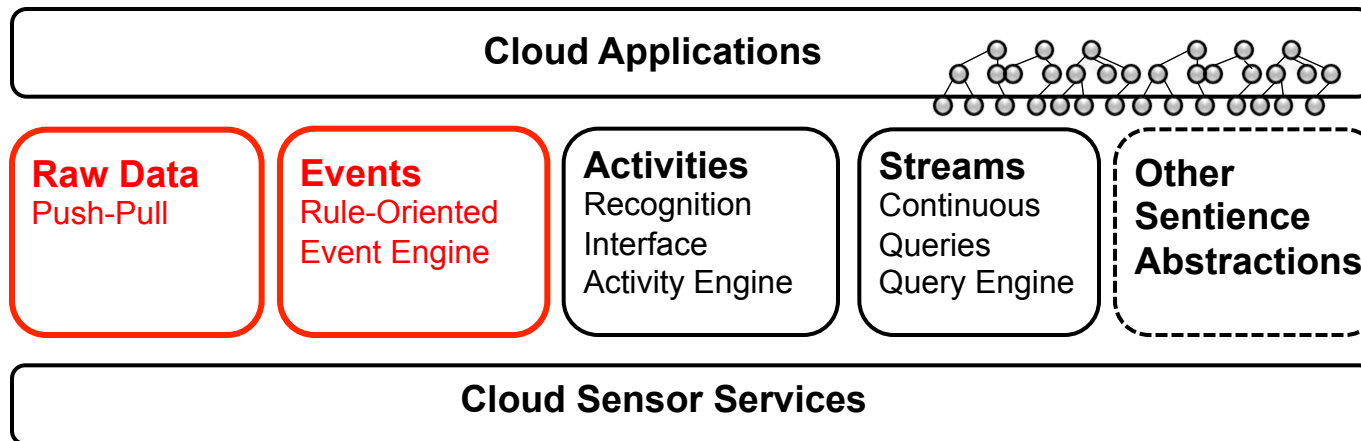
- For every edge, there exists a corresponding ACM at the cloud. It hosts the cloud sensor service (passed from the edge) to be subscribed to by other cloud services.
- It acts as the cloud gateway to the lower layers, and meanwhile, it hosts the most basic “clouding” of sensors based on which sensor-based cloud applications can be built.

Cloud Application Runtime (CAR):

- It is the container where application services are deployed and managed.
- An application makes invocation to the cloud sensor services at the ACM to acquire raw sensor readings from the physical deployment.

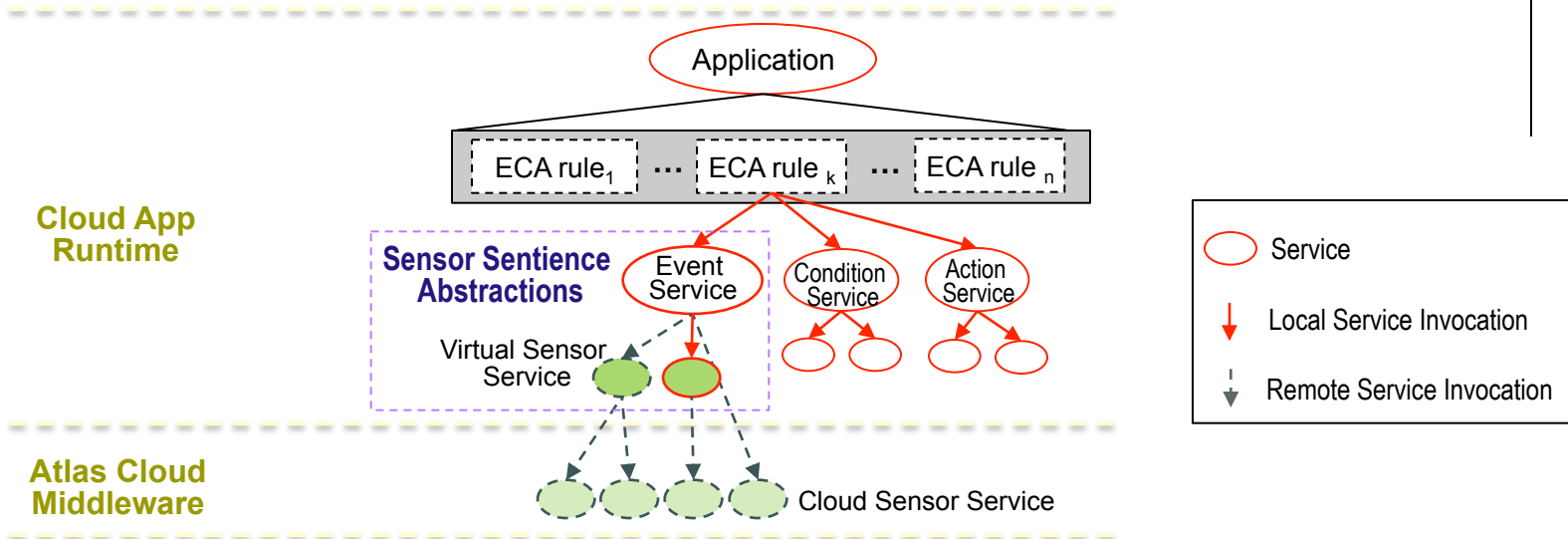
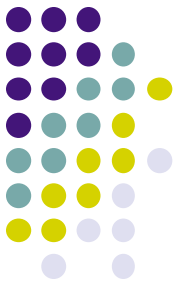


Programming in CEB

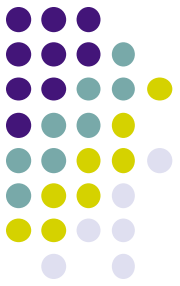


Event-driven SODA Application Model (E-SODA)

E-SODA Application Model



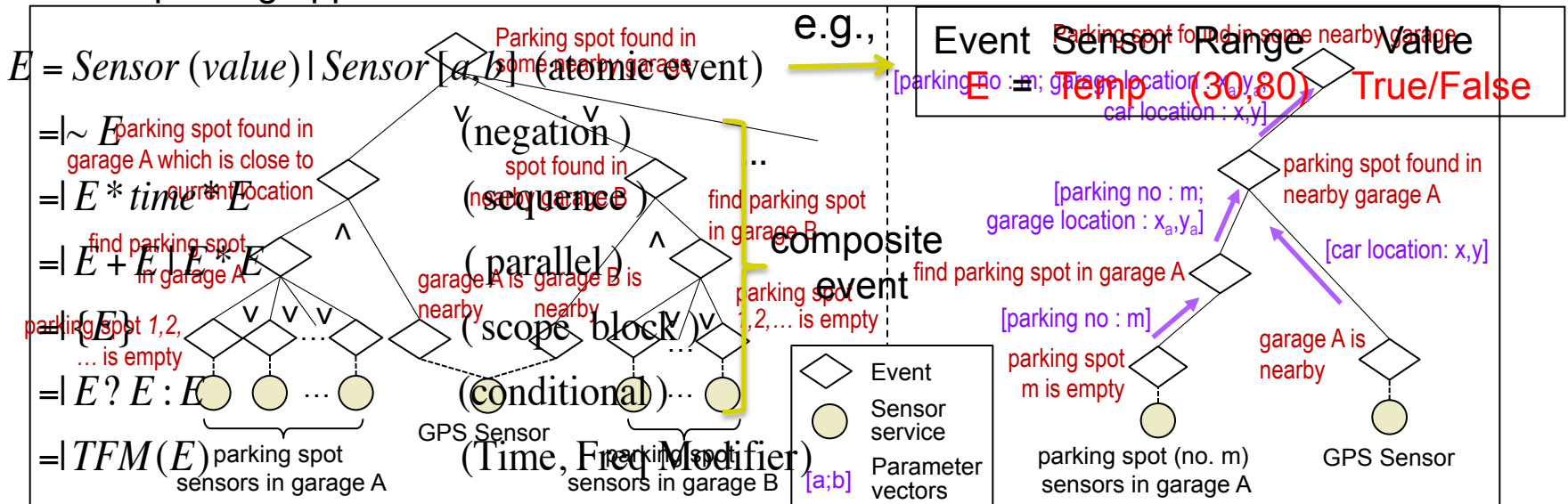
- Rule-oriented event processing paradigm
 - Defines a set of Event/Condition/Action (ECA) rules.
 - An application is a composition of interrelated services together performing the function of rule evaluation.
- **Event Services**
 - **Subscribe to and invoke the cloud sensor services from the ACM to implement event-level abstractions of sensor data.**

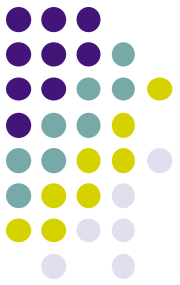


Event Services in E-SODA

- An event service listens to the occurrence of a particular event which is a logical expression over sensor values.
- The event has a Boolean value which is evaluated to true when the event occurs otherwise to false.
- To provide information about an event that occurred, event parameters propagate along an *event representation tree* (ERT).

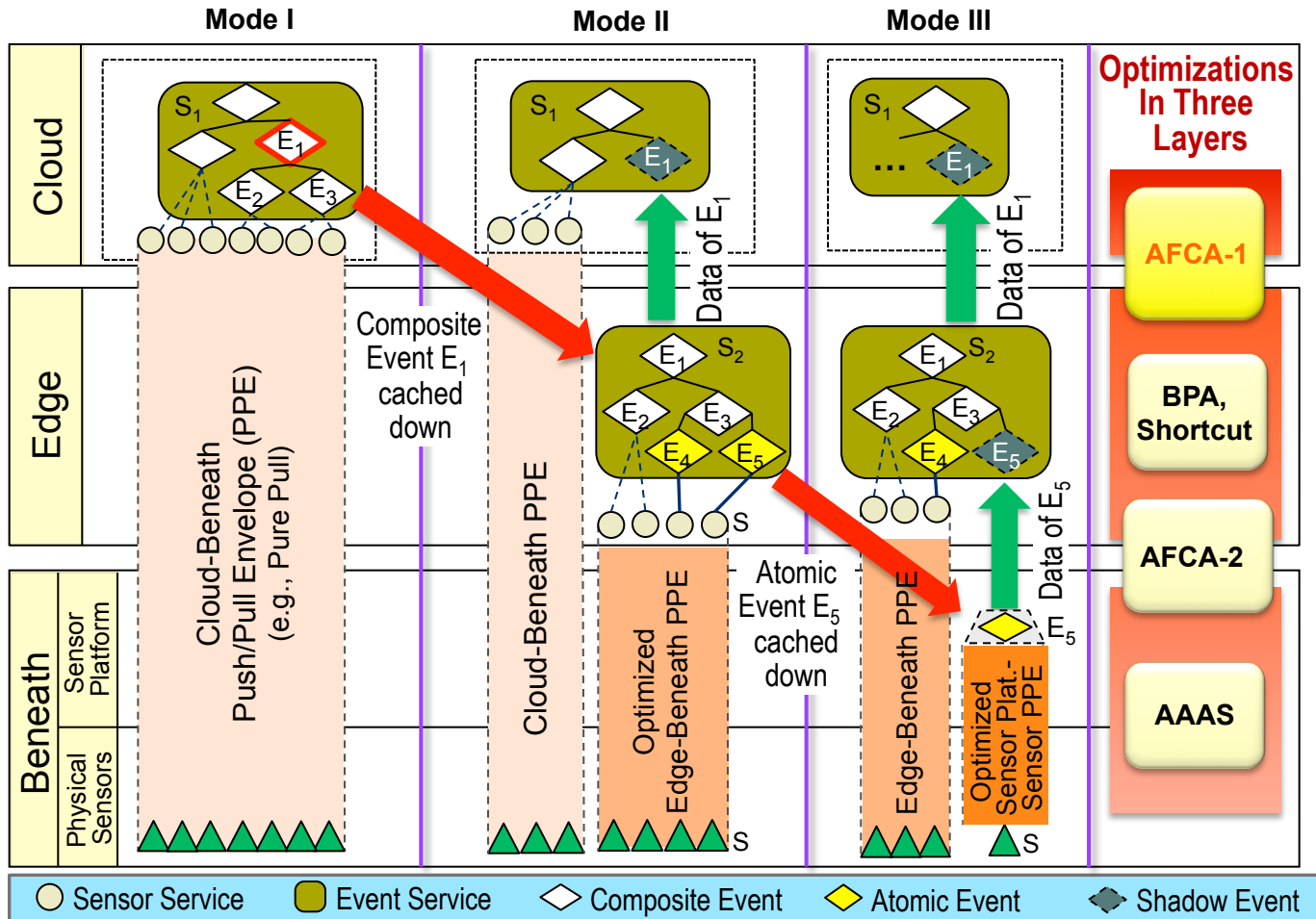
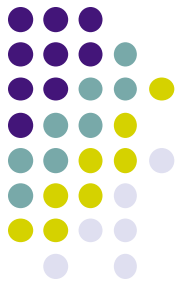
Smart parking application





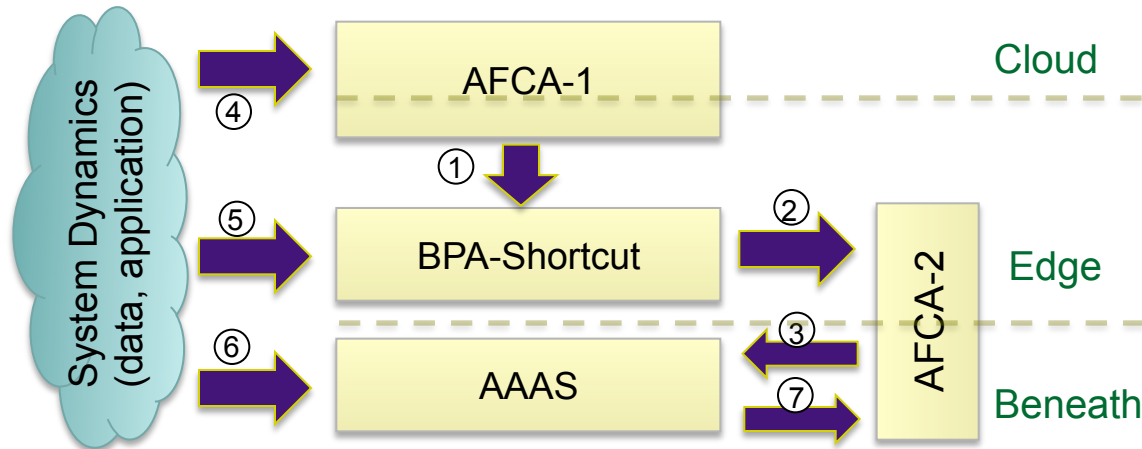
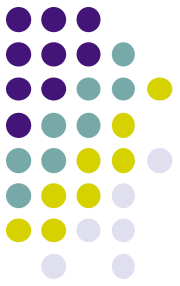
Bi-directional Waterfall Optimization Framework

Bi-directional Waterfall Optimization Framework



Metaphorically, such optimization framework allows for most interesting data to flow up and for most curious applications to propagate down

Optimization Algorithms Interplay

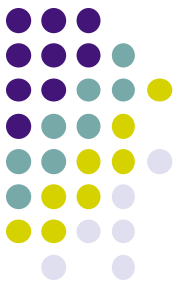


- **AFCA-1:** Application Fragment Caching Algorithm 1 (Cloud-Edge)
- **BPA-Shortcut:** Branch Permutation & Shortcut Algorithm
- **AFCA 2:** Application Fragment Caching Algorithm 2 (Edge-Beneath)
- **AAAS:** Application-Aware Adaptive Sampling Algorithm



A Cloud-Sensor Data/Application Benchmark

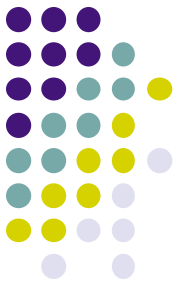
- Requirements:
 - Large number of sensors (city-scale) of diverse sensor types.
 - Diversity of sensor data characteristics (e.g., data type, variance, and tolerance to time fidelity).
 - Large number of events with different event characteristics (e.g., rare, static, dynamic, simple, complex).
- Two types of cloud-sensor system expansion:
 - Horizontal: Applications are built on dedicated, proprietary or exclusive sensors. Deploying a new application requires installing its associated sensor devices.
 - Vertical: Applications are built on pre-existing, sharable sensors which are not bounded to any individual application. Adding new services does not require installing additional sensors.



A Cloud-Sensor Data/Application Benchmark

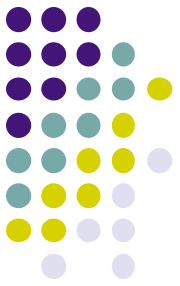
- PlaceLab Dataset:
 - Contains sensor data from 178 wired and wireless sensors for a 2.5 month period when a couple stayed in the PlaceLab*.
 - Sensors: location, humidity, switch, pressure, light, temperature, gas, current, flow sensor and body-worn accelerators.
- Horizontal Expansion:
 - Manually created basic events based on household sensors (activity recognition, emergency detection, security, etc.).
 - The same event set is used in every synthesized smart homes.
 - Synthetic data generator:
 - PlaceLab dataset and basic event set* →
 - Probability of event occurrence and event dynamics* →
 - Sensor data* for all other synthesized smart homes
 - Irregular events (e.g., house intrusion, fire alarm)

A Cloud-Sensor Data/Application Benchmark

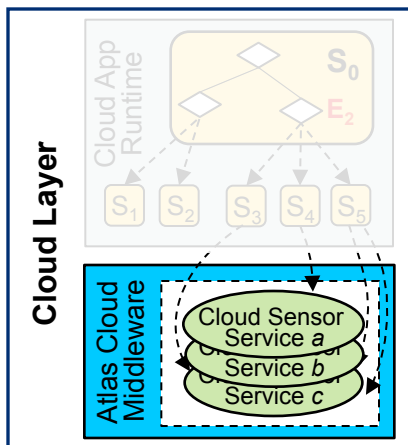


- Vertical Expansion:
 - Randomly generated composite events using the basic event set, connected by randomly chosen operators.
 - In addition, the TFM parameters are also modified in these synthesized events to manipulate different activeness of event evaluations.

Dominant Resource in ACM

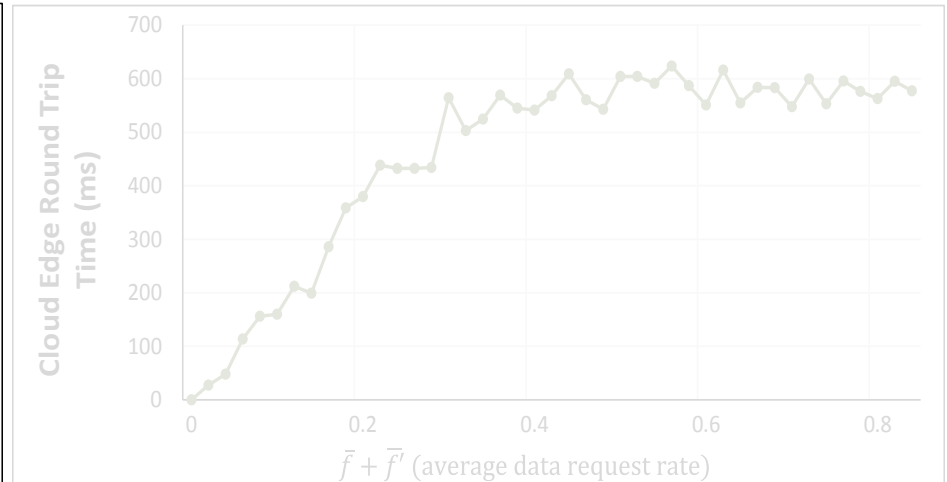
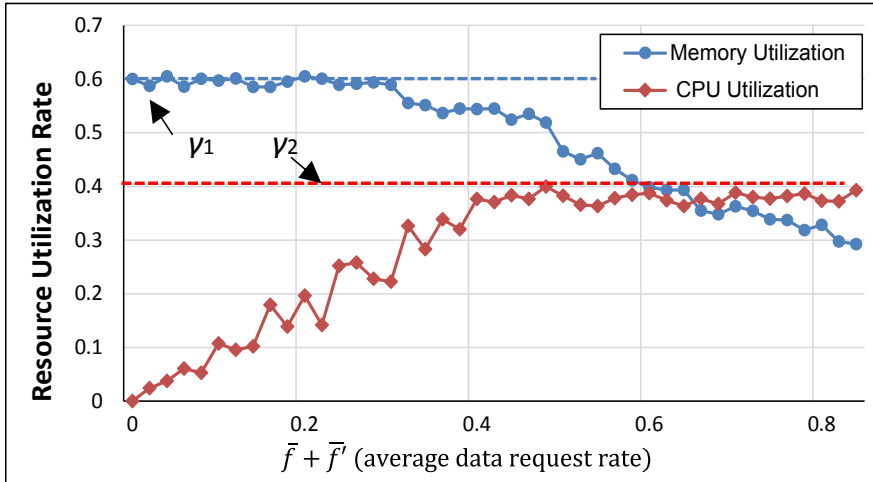
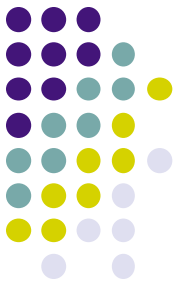


- Analyzing resource usage in ACM is relatively simple because it hosts mostly the cloud sensor services which are very simple and similar in their sizes and functionalities – no more than data transporters between cloud applications and edges with minimum data processing.



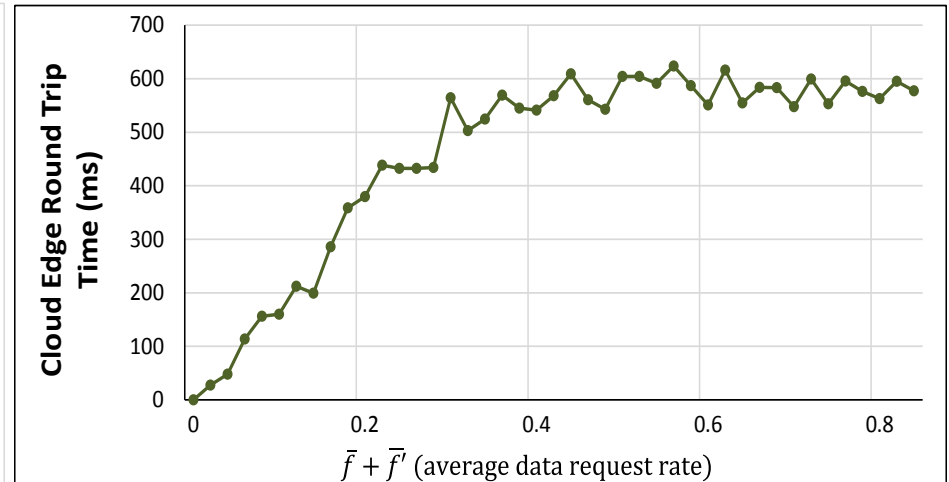
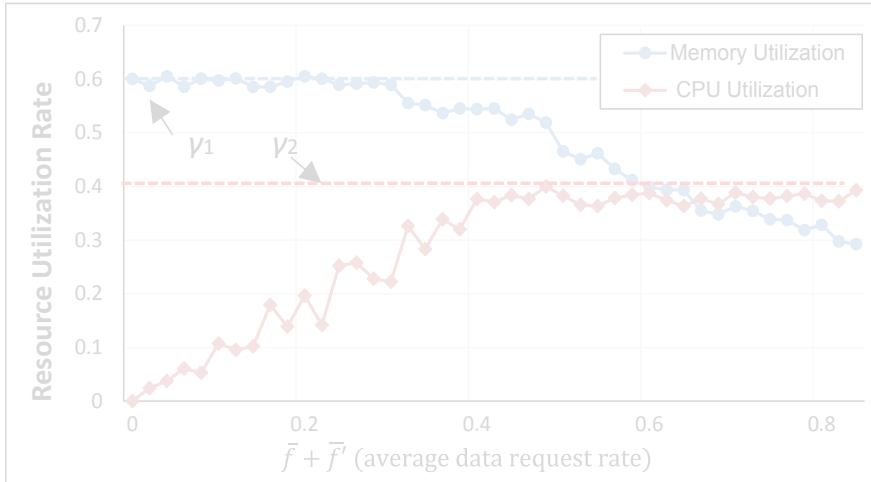
- For memory resource, most of its consumption comes from the deployment of cloud sensor services.
- For processing resource, most of its consumption is attributed to the processing of the cloud sensor services of which we reasonably consider only the data transmitting and receiving as the major contributions while ignoring others (e.g., data processing and reading/writing cache).

Dominant Resource in ACM



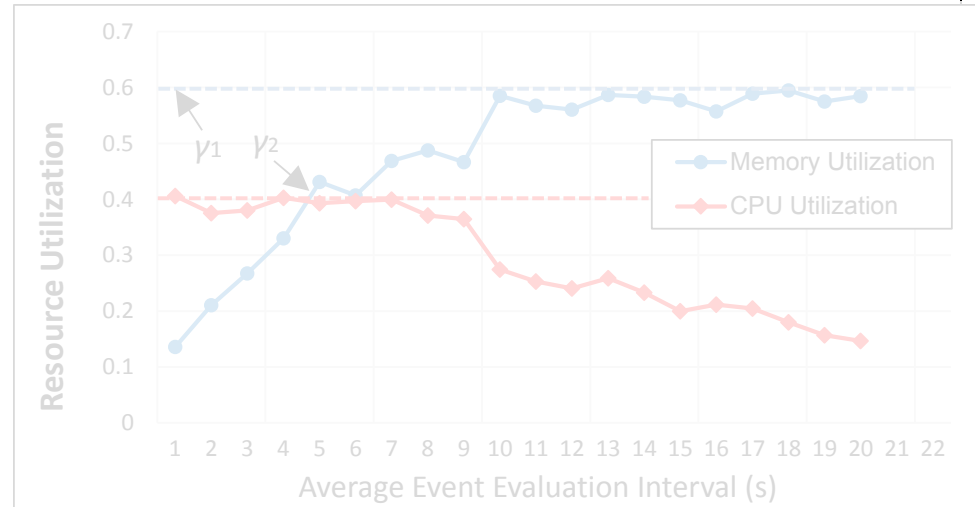
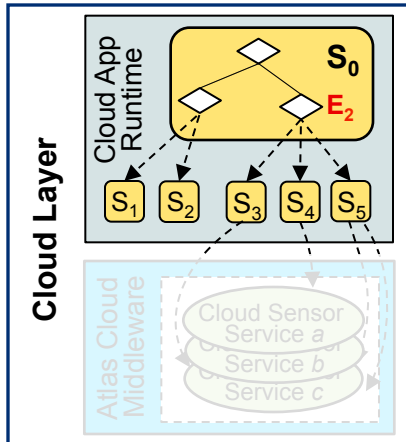
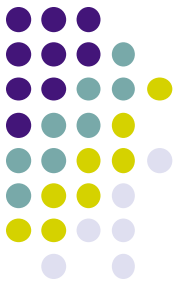
- With increasing $\bar{f} + \bar{f}'$ the dominant resource of the ACM changes from memory to CPU
- Network delay first increases as the data request frequency increases and eventually stops at an acceptable level (~600ms) due to performing cloud auto-scaling (decreasing the number of sensor services per cloud instance). Therefore, we exclude the bandwidth from being ACM's dominant resource.

Dominant Resource in ACM



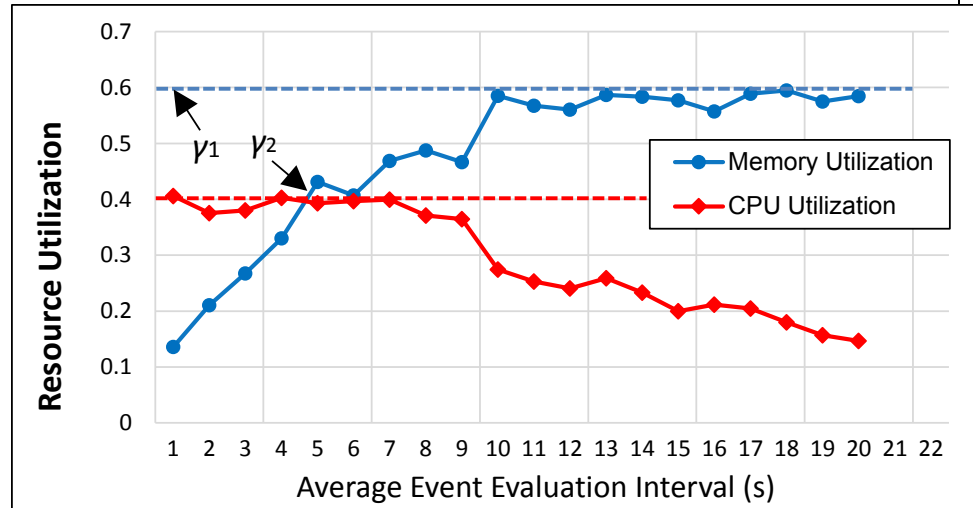
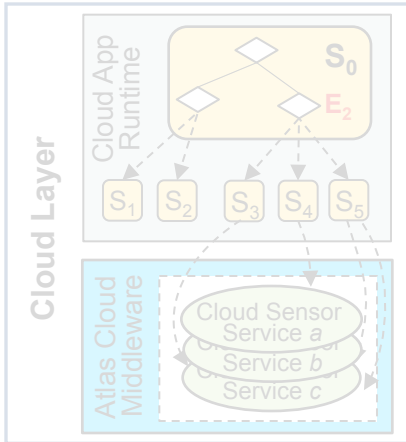
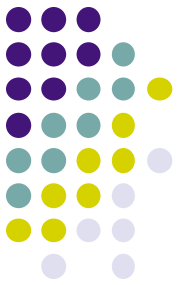
- With increasing $\bar{f} + \bar{f}'$ the dominant resource of the ACM changes from memory to CPU
- Network delay first increases as the data request frequency increases and eventually stops at an acceptable level (~600ms) due to performing cloud auto-scaling (decreasing the number of sensor services per cloud instance). Therefore, we exclude the bandwidth from being ACM's dominant resource.

Dominant Resource in CAR



- CARs are more processing intensive and less transmission intensive than ACMs. In addition, CARs communicate with ACMs (i.e., in-cloud instances) which are allocated much higher bandwidth capacity than outbound interfaces. Consequently, like ACM we exclude bandwidth from being a dominant resource of the CAR.
- With increasing event evaluation intervals, the dominant resource of the ACM changes from CPU to Memory

Dominant Resource in CAR



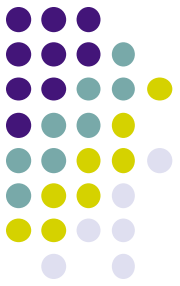
- CARs are more processing intensive and less transmission intensive than ACMs. In addition, CARs communicate with ACMs (i.e., in-cloud instances) which are allocated much higher bandwidth capacity than outbound interfaces. Consequently, like ACM we exclude bandwidth from being a dominant resource of the CAR.
- With increasing event evaluation intervals, the dominant resource of the ACM changes from CPU to Memory

Dominant Resource in Cloud Components



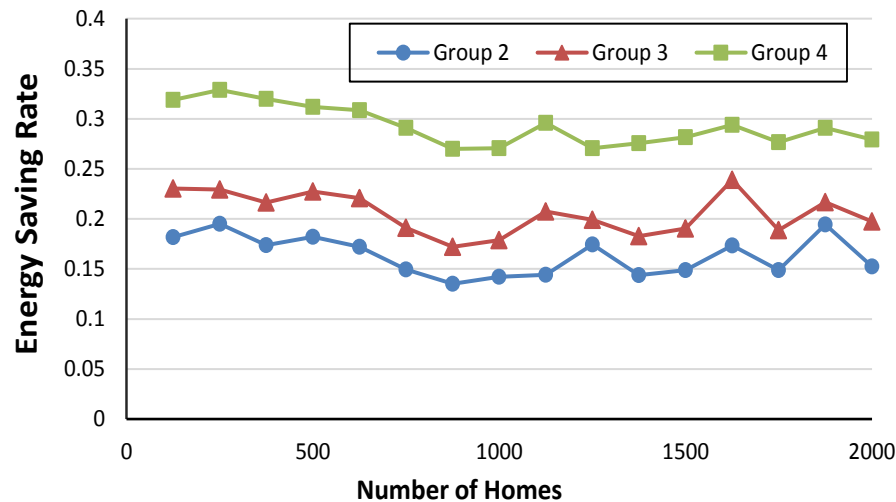
For both ACM and CAR, the dominant resources which determine the cloud dimension can be either processing or memory, but not bandwidth. Which one dominates the other depends on the overall sensor sampling rate in the cloud-sensor system and the maximal quota of resource usage allocated in the cloud instance (i.e., the auto-scaling threshold).

Combined Optimizations



FOUR EXPERIMENT STUDY GROUPS

Experiment Groups	Conversion from Gaussian and CGS EMU to SI ^a
1	Pure Pull
2	Add Shortcut Evaluation to group 1
3	Add BPA to group 2
4	Add AFCA-2 (selective push, AAAS) to group 3

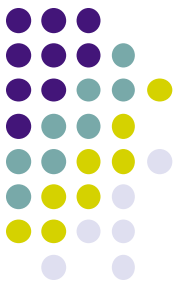


Average energy saving rate for three experiment groups with different number of homes.



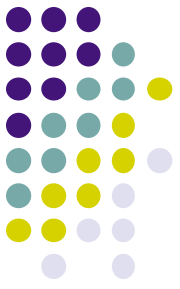
Concluding Remarks

- More focus is needed on High Velocity Big Data and its real-time application involving actuations
- Let us not run before we walk: we started off focusing mobile cloud: cloud-sensor systems represents a major class of applications in the Smart City, mobile devices certainly a major player.
- Let us marry optimization with architecture
- Let us search (or build) together for massive, truly city-scale datasets for better validation.



References

1. Y. Xu and A. Helal, “Scalable Cloud-Sensor **Architecture** for the Internet of Things,” Accepted for publication in the **IEEE Internet of Things Journal**. Accepted February 2015.
2. Y. Xu and A. Helal, “An **Optimization Framework** for Cloud-Sensor Systems,” in Proceedings of the 6th IEEE International Conference on Cloud Computing (**IEEE CloudCom’14**), Singapore, Dec 2014.
3. Y. Xu and A. Helal, “**Application Caching** for Cloud-Sensor Systems,” in Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (**ACM MSWiM’14**), Montreal, Canada, Sept. 2014.
4. Y. Xu, A. Helal, M. Thai and M. Schmalz, “Optimizing Push/Pull Envelopes for Energy-Efficient Cloud-Sensor Systems,” In Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (**ACM MSWiM’11**), Miami, Florida, Dec. 2011.



References

- Under review: Y. Xu and A. Helal, “Optimizing Cloud-Sensor Systems: A Bi-directional Waterfall Approach,” Submitted March 2015.
 - Detailed optimization algorithms except for the application caching algorithm which is published in ACM MSWiM in 2014.