Optimizing Sensor Data Acquisition for Energy-Efficient Smartphone-based Continuous Event Processing

Lipyeow Lim
University of Hawai`i at Mānoa

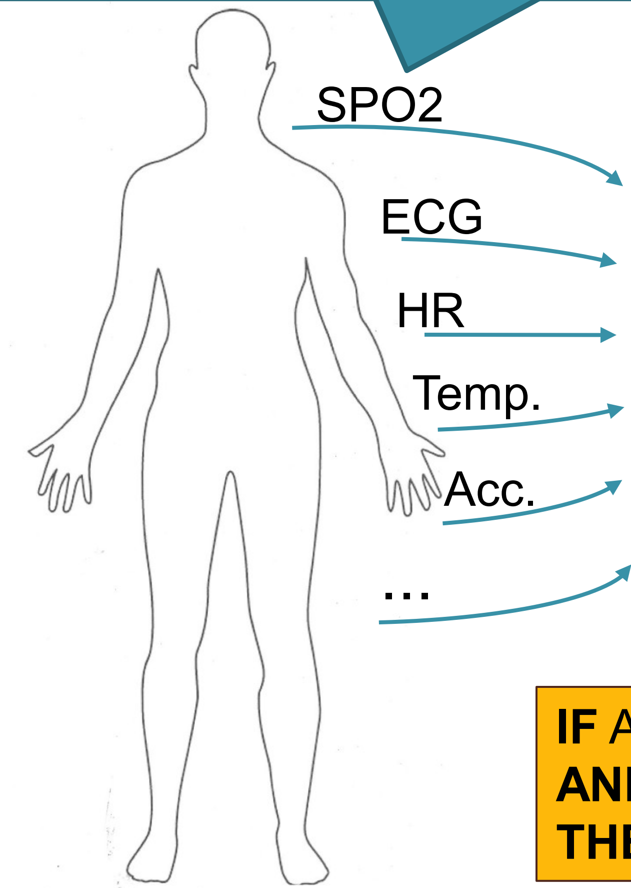
Archan Misra
Singapore Management University

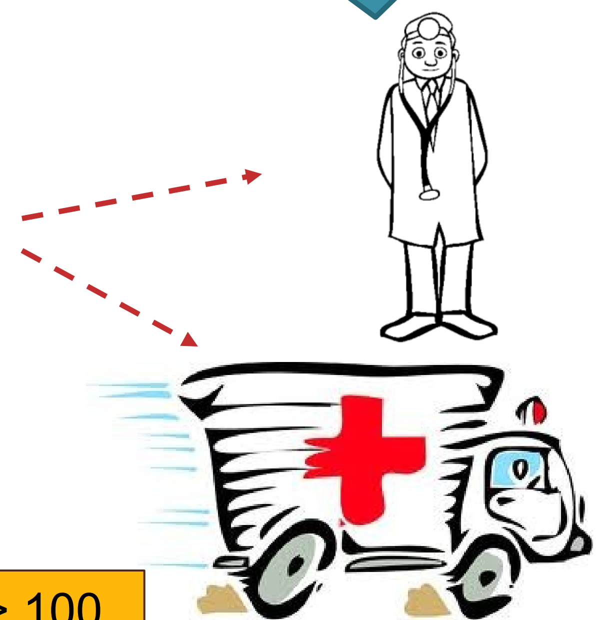
Telehealth Scenario

Wearable sensors transmit vitals to cell phone via wireless (eg. bluetooth)

Phone runs a complex event processing (CEP) engine with rules for alerts

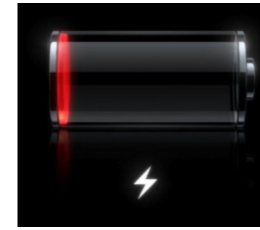
Alerts can notify emergency services or caregiver





IF Avg(Window(HR)) > 100
AND Avg(Window(Acc)) < 2
THEN SMS(doctor)

Energy Efficiency



- Energy consumption of processing

This Talk

  - Sensors: transmission of sensor data to CEP engine
  - Phone: acquisition of sensor data
  - Phone: processing of queries at CEP engine

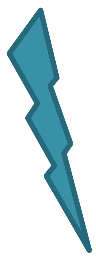
This Talk

- Optimization objectives
  - Minimize energy consumption at phone
  - Maximize operational lifetime of the system.

Sensor Data Acquisition

3D acc.
ECG,
EMG,
GSR

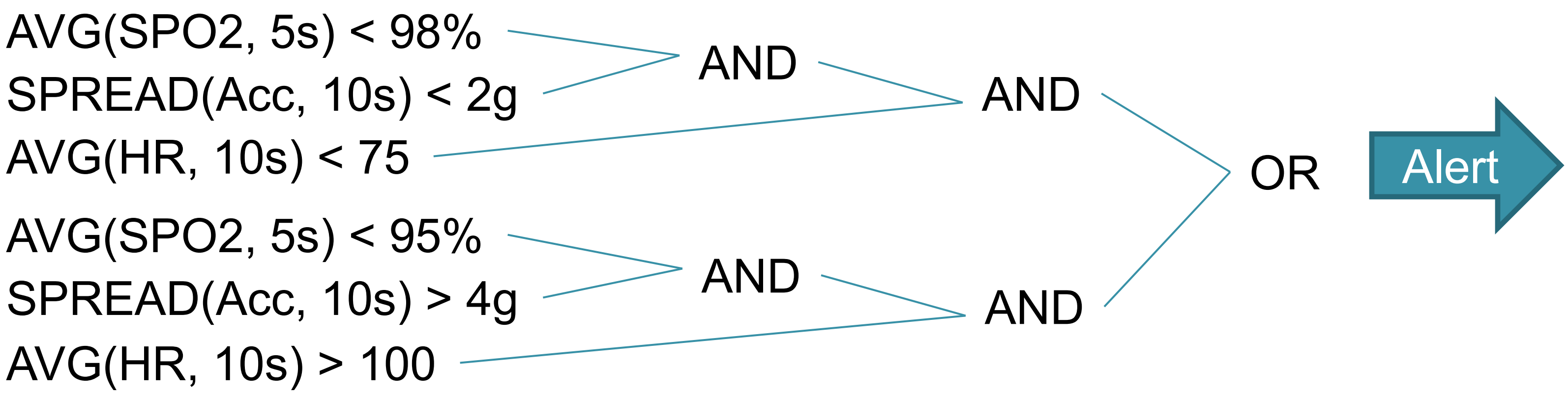
Bluetooth
Or 802.11
Or 802.15



- Constant sampling rate
- 802.11 (wifi) uses 2 power modes: active, idle
- Bluetooth has 3 modes: active, idle, sleep (not relevant).
- Time needed to switch modes
- Energy expended to switch

Sensor Type	Bits/ sensor channel	Channels/ device	Typical sampling frequency (Hz)
GPS	1408	1	1 Hz
SpO2	3000	1	3 Hz
ECG (cardiac)	12	6	256 Hz
Accelero-meter	64	3	100 Hz
Temperature	20	1	256 Hz

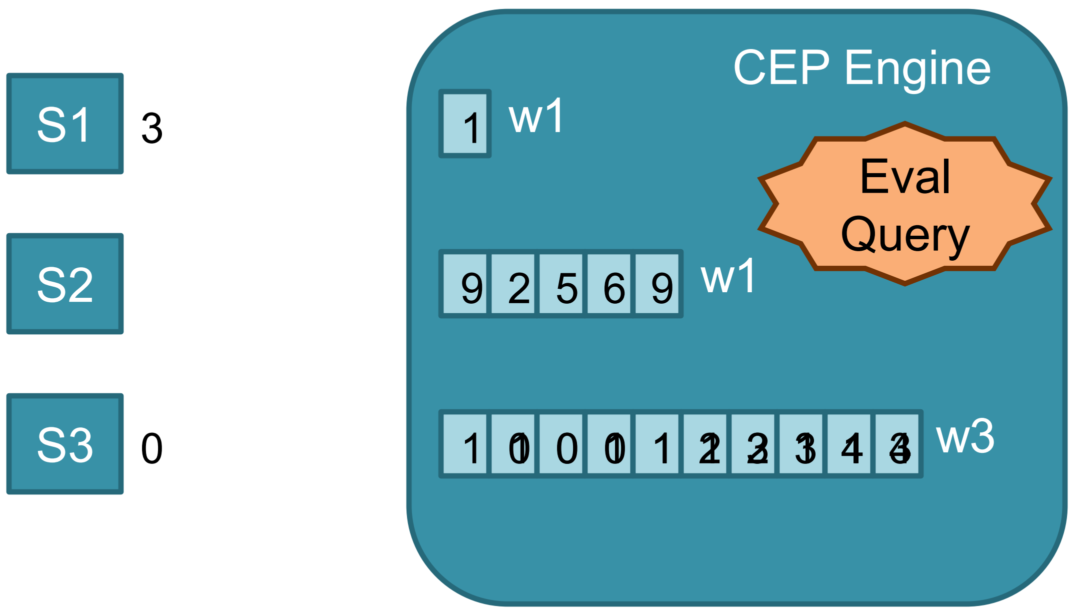
Query Model



- A query is a boolean combination of predicates
- Predicates
  - Aggregation functions over a time-based window of sensor data
- Traditional push model
  - A given query is evaluated whenever a new sensor reading arrives

Continuous Evaluation

if Avg(S2, 5)>20 AND S1<10 AND Max(S3,10)<4 then email(doctor).



Push

When t_i of Si arrives
- Enqueue t_i into W_i
- If Q is true,
- Then output alert

Pull

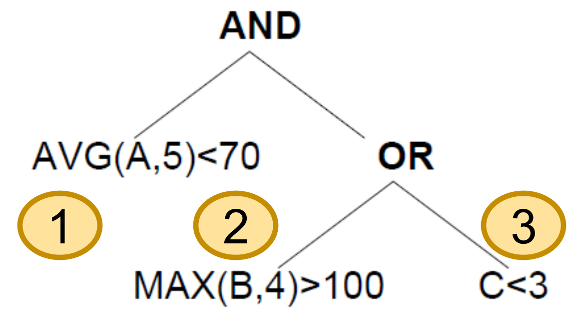
Loop
- Acquire t_i for Si
- Enqueue t_i into W_i
- If Q is true,
- Then output alert

End loop

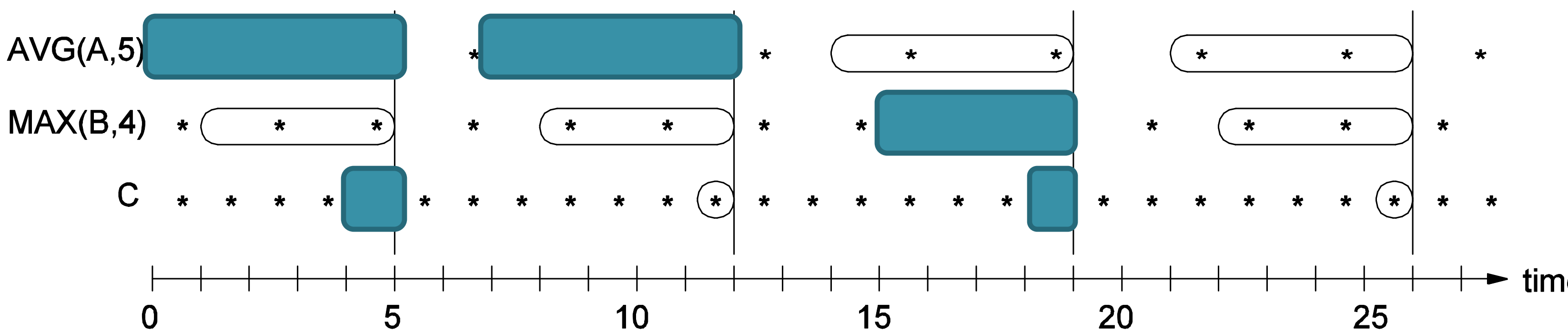
Key Ideas

- Pull model
  - Evaluate a query every ω seconds
  - Acquire only data that is needed
- Evaluation order of predicates matter!
  - Shortcircuiting can avoid data acquisition
- Batching

Example: ω=7



- Time 5: eval order is 3,1,2
- Time 12: eval order is 1,2,3
- Time 19: eval order is 2,3,1



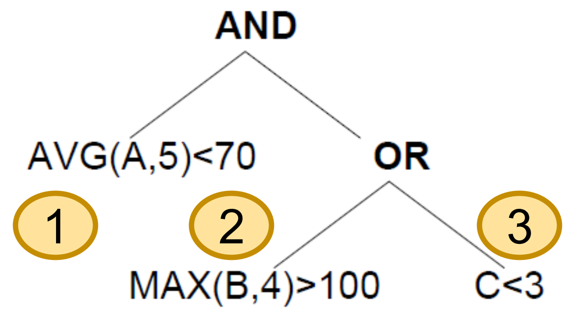
Evaluation Order

if Avg(S2, 5)>20 AND S1<10 AND Max(S3,10)<4 then email(doctor).

Predicate	Avg(S2, 5)>20	S1<10	Max(S3,10)<4
Acquisition	5 * .02 = 0.1 nJ	0.2 nJ	10 * .01 = 0.1 nJ
Pr(false)	0.95	0.5	0.8
Acq./Pr(f)	0.1/0.95	0.2/0.5	0.1/0.8

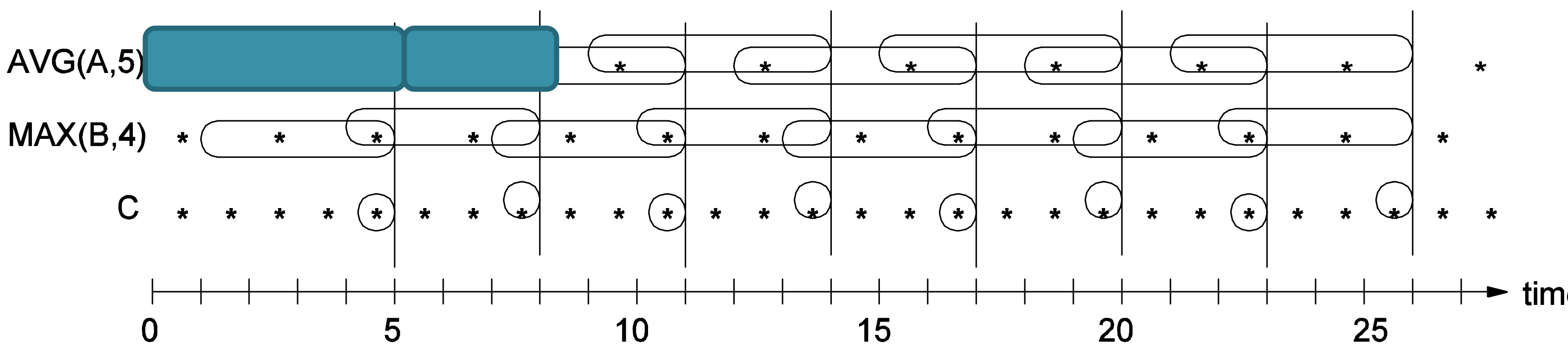
- Evaluate predicates with lowest energy consumption first
- Evaluate predicates with highest false probability first
- Hence, evaluate predicate with lowest normalized acquisition cost first.

Example: ω=3



- Time 5: 1,2,3
- Time 8: acquisition cost for A becomes cheaper, because some tuples are already in buffer

Acquisition cost depends on state of the buffer at time t



Algorithm

At each ω

1. Calculate normalized acquisition cost (NAC) based on buffer state and P(pred=true)
2. Find evaluation order using NAC
3. Acquire sensor data and eval pred using eval order with shortcircuiting.

What happens if >2 predicates operate on the same sensor data stream?

Simulation Setup

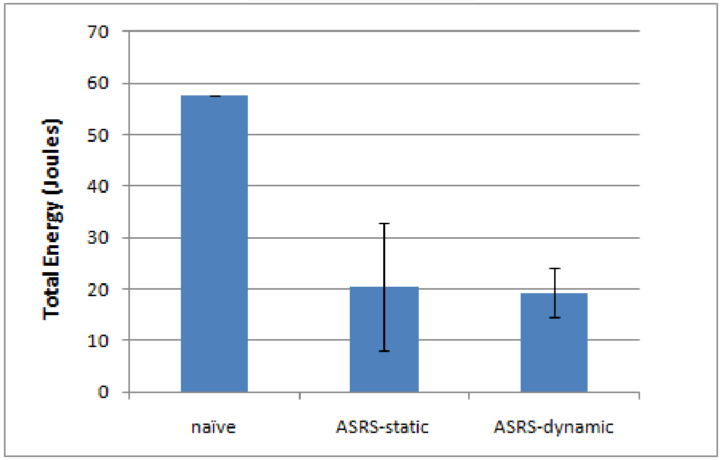
- Naive
  - data from all sensors acquired in batches
- ASRS-static
  - Evaluation order determined once at initialization and never changes
- ASRS-dynamic
  - Evaluation order determined at each ω time period.
- Data generated using independent Gaussian distribution

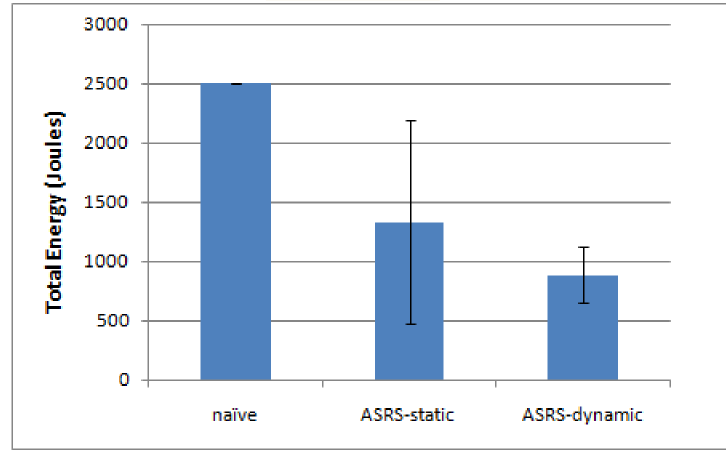
Simulation Results

Bluetooth

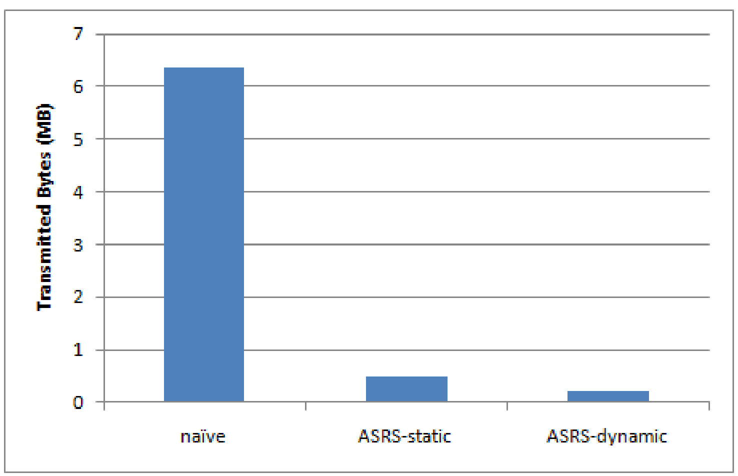
802.11

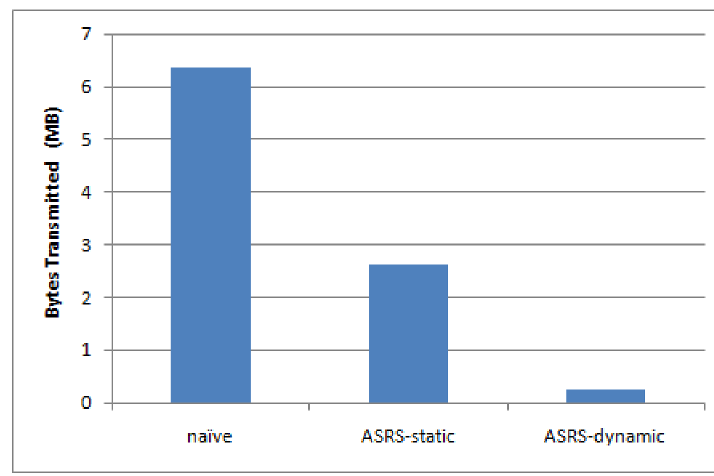
Energy





Bytes





A Lot More Work Needed

- Improve simulator
  - Disjunctive normal form query representation
  - More realistic data generators
- Estimation algorithms for P(pred=true)
  - Condition on context
- Batching: wait say 3 ω before query evaluation
  - Design and implement the algorithm
  - Evaluation via simulation
- End-to-end evaluation on Android phone
  - Maximize operational lifetime of phone+sensors

Other projects

- Cloud-based SQL Processor for Scientific Applications
  - Benchmarking work
  - Query optimization for parallel SQL processing
  - Elastic & dynamic parallelization

- Develop a journal version of: *Optimizing Access Across Multiple Hierarchies in Data Warehouses*

- Data compression of Join Query Result Sets

ACM SIGMOD Programming Contest

- Task: **A Durable Main-Memory Index Using Flash**
- Contest ends Mar 31 2011
- Skills: C, OS, Algorithms, DB concepts
- Why do it?
  - Great learning experience
  - Looks good on your resume

Free trip for two to SIGMOD 2011 in Athens!

USD 5000