

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

Introduction

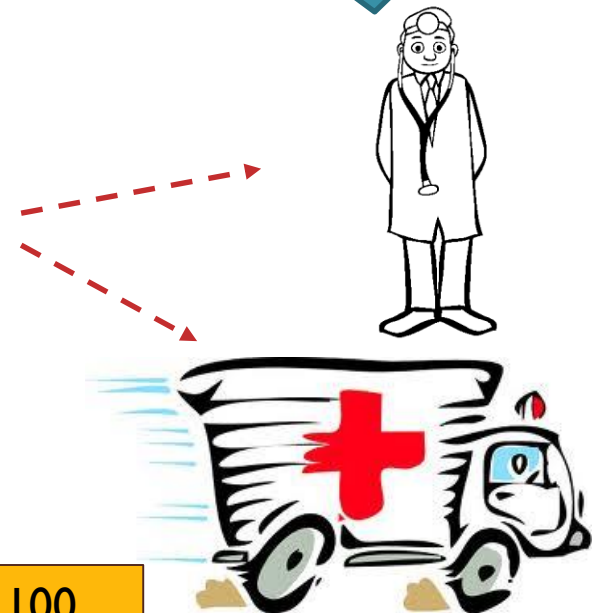
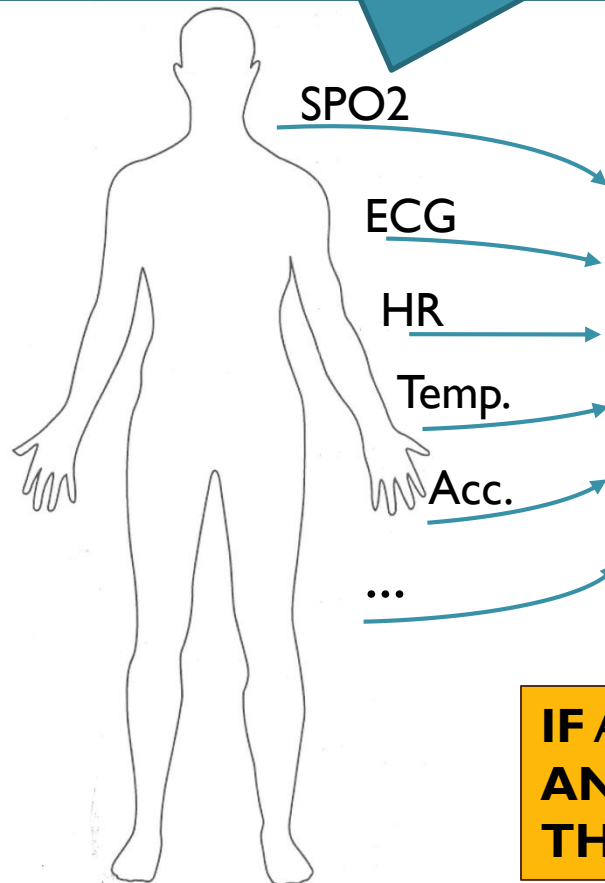
- Smartphones already have several on-board sensors (e.g., GPS, accelerometer, compass and microphone)
- But, there are many situations where the smartphone aggregates data from a variety of other specific (a) external medical (e.g., ECG, EEG, SpO₂) or (b) environmental (e.g., temperature, pollution) sensors, using a Personal Area Network (PAN) technology, such as Bluetooth™, WiFi direct etc.

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

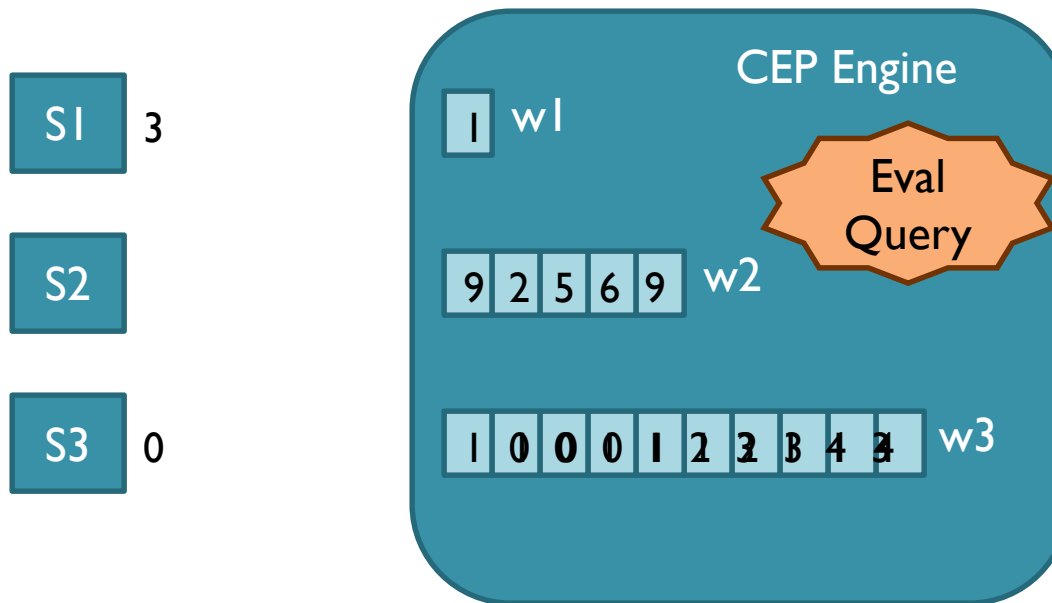
Alerts can
notify
emergency
services or
caregiver



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

Continuous/Streaming Evaluation

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

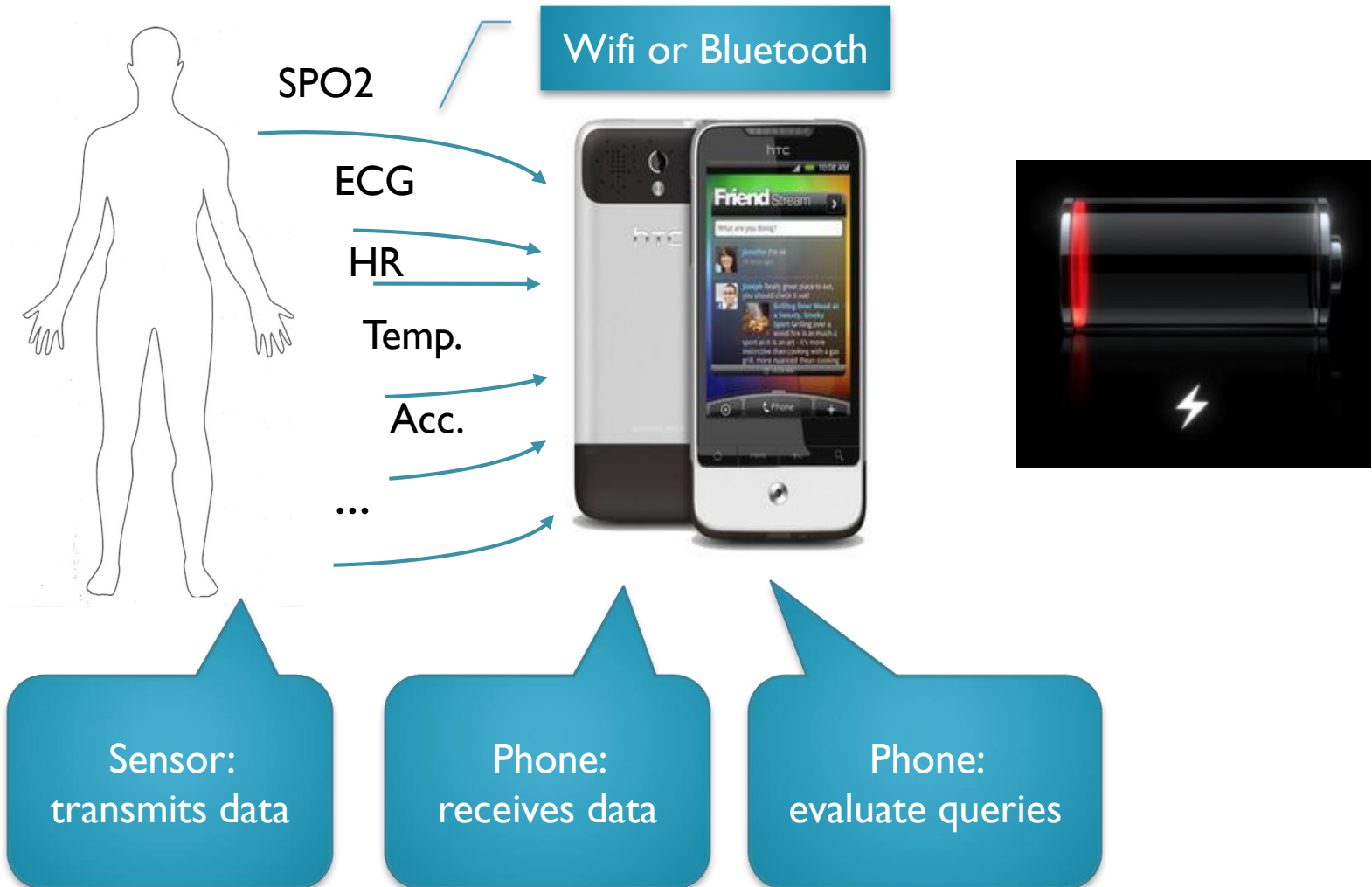


Algorithm

When x_i of S_i arrives
Enqueue x_i into W_i
If Q is true,
Then output alert

“Push”
model

Energy Consumption



Research Question



Is there a better way to perform such complex event processing that

- Minimizes energy consumption at the phone, and/or
- Maximizes operational lifetime of the system.

Key Idea

- This work explores an approach to reduce the energy footprint of such continuous context-extraction activities, primarily by **reducing the volume of sensor data that is transmitted wirelessly over the PAN interface** between a smartphone and its attached sensors, without compromising the fidelity of the event processing logic.
- We aim to replace the “push” model of sensor data transmission,
 - where the sensors simply continuously transmit their samples to the smartphone,
- **Use** “phone-controlled dynamic pull” model, where the smartphone selectively pulls only appropriate subsets of the sensor data streams.

Sensor Data Acquisition

3D acc.
ECG,
EMG, GSR

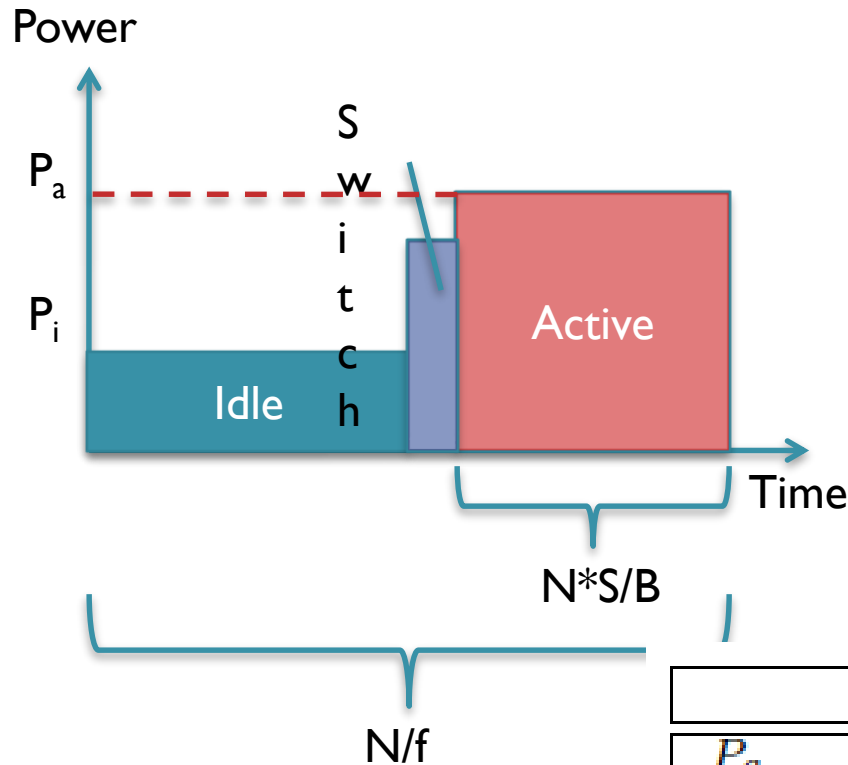


Bluetooth
Or 802.11



- Constant sampling rate
- 802.11 (wifi) uses 2 power modes: active, idle
- Bluetooth has 3 modes: active, idle, sleep.
- Time needed to switch modes
- Energy spend to switch

Pulling N Tuples from Sensor



Data generation rate
 $R = f * S$

- Idle mode consumes P_i mW
- Active mode consumes P_a mW
- Sensor sampling rate is f Hz
- A sample is S bits
- Bandwidth is B Mbps

	IEEE 802.11	Bluetooth 2.0+EDR
P_a	947 mW	60mW
P_i	231 mW	5 mW
B	54 Mbps	1 Mbps
E_{switch}	14 μ Joule	–
T_{idle}	100 ms	–
T_{switch}	–	6 msec

Accordingly, it follows that the total transmission time for the N samples, generated over a time interval of $\frac{N}{f}$, equals $\frac{N*S}{B}$

IEEE 802.11:

1) *IEEE 802.11*: Commercial IEEE 802.11 radios can operate in two states—a normal 'active' mode (when the radio interface receives or transmits packets) and a Power Save Mode (PSM), where the radio periodically wakes up to check if there any pending transmissions or receptions. The following are two key relevant properties associated with 802.11 hardware:

- Due to the switching characteristics of the radio hardware, there is typically a lower bound on the minimal idle time Th_{idle} , below which the radio cannot enter the PSM mode (typically, this is around 100 ms) [9].
- There is a fixed, *duration-independent* switching energy E_{switch} spent when a radio transitions from the PSM to the 'active' mode.

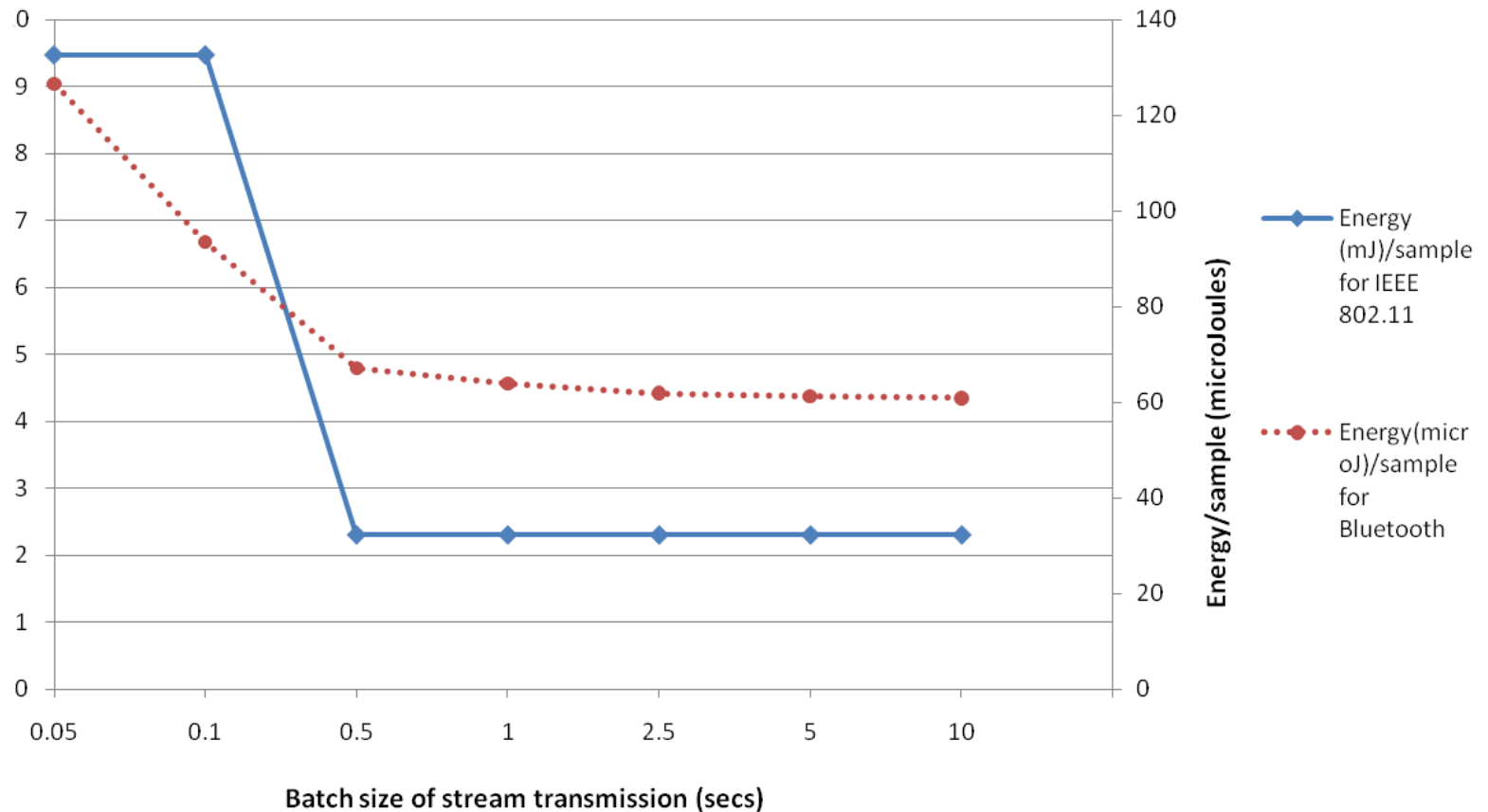
$$E_t = \begin{cases} P_i * \left(\frac{N}{f} - \frac{N*S}{B}\right) \\ \quad + P_a * \frac{N*S}{B} + E_{switch} & \text{if } \frac{N}{f} - \frac{N*S}{B} > Th_{idle} \\ P_a * \frac{N}{f} & \text{otherwise} \end{cases}$$

Bluetooth:

the smartphone, which primarily receives data from an external sensor, we denote its active energy consumption P_a as the energy spent in actively receiving data. We consider the Bluetooth version 2.0+ EDR and assume, for analytical tractability, that a single sensor device attaches as a slave to the master located on the smartphone. While the low-power mode results in significantly low power consumption, note that there is a latency T_{switch} involved in switching from the non-associated low-power mode to the associated-active mode. Accordingly, any data transfer duration would consist of the total time spent in transfer $\frac{N*S}{B}$, plus the additional time T_{switch} . Accordingly, the total energy consumed in transmitting the sensor stream in batches of N samples is given by:

$$E_t = P_i * \left(\frac{N}{f} - \frac{N * S}{B} - T_{switch} \right) + P_a * \left(\frac{N * S}{B} + T_{switch} \right) \quad (2)$$

The Impact of Batched Transmissions on the Transmission Energy Overhead per Sample



representative accelerometer sensor, with $S = 192$ bits/sample and $f = 100$ Hz. It is clear that the choice of the batch size N , for a given radio technology, has a significant effect on the

ACQUA

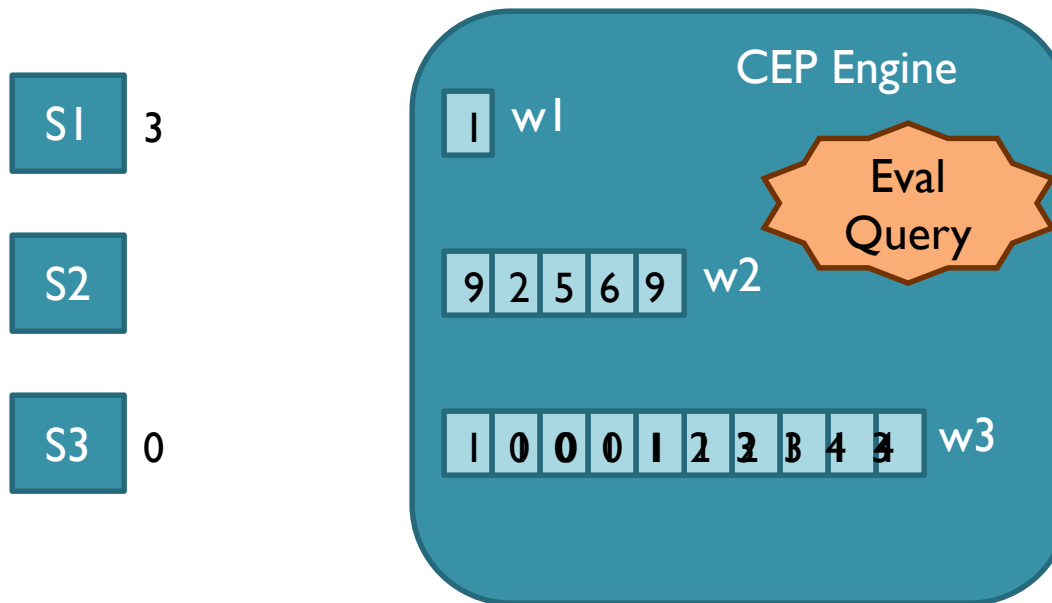
- Introducing a new continuous stream processing model called ACQUA (Acquisition Cost-Aware Query Adaptation)
- Which (a) first learns the **selectivity properties** of different sensor streams
- and then (b) utilizes such **estimated selectivity values to modify the sequence** in which the smartphone acquires data from the sensors.

Normalized acquisition cost

- We observe that the choice of the best acquisition sequence should take into account both (i) the acquisition energy cost and (ii) the selectivity properties.
- We should ideally retrieve the data chunk from the sensor that should have a (i) low acquisition cost and (ii) also a high likelihood of helping to terminate the predicate evaluation.
- For the **conjunctive query**, we note that a single 'FALSE' predicate implies that the complex predicate is FALSE and that the subsequent steps of predicate evaluation can be aborted.
- Accordingly, in our formulation, we first compute the 'normalized acquisition cost' (NAC) as a ratio of the acquisition cost normalized by the 'predicate being FALSE' probability

Continuous/Streaming Evaluation

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




Algorithm

When x_i of S_i arrives
Enqueue x_i into W_i
If Q is true,
Then output alert

“Push”
model

Query (I)

- Consider a hypothetical activity/wellness tracking application that seeks to detect an episode where an individual
 - (i) walks for 10 minutes, AND
 - (ii) while being exposed to an ambient temperature (95th percentile over the 10 minute window) of greater than 80F, AND
 - (iii) while exhibiting an AVERAGE heart rate (over a 5 minute window) of > 80 beats/min.

- 
- Assume that this application uses an external wrist-worn device, equipped with
 - accelerometer (sensor S1, sampling at 100 samples/sec),
 - heart rate (S2, sampling at 5 sample/sec) and
 - temperature (S3, sampling at 10 sample/sec) sensors.

Probability of Success

- Consider a the probability of the given conditions occurring at that time be as follows;
- walks for 10 minutes, -- 0.95 (success)
- while being exposed to an ambient temperature (95th percentile over the 10 minute window) of greater than 80F, -- 0.05 (success)
- while exhibiting an AVERAGE heart rate (over a 5 minute window) of > 80 beats/min. = 0.2 (success)

Energy Consumption

- Assume the sensor energy costs as follows;
- walks for 10 minutes, -- $E(S1) = 0.02\text{nj/sample}$
- while being exposed to an ambient temperature (95th percentile over the 10 minute window) of greater than 80F, -- $E(S2) = 0.02\text{nj/sample}$
- while exhibiting an AVERAGE heart rate (over a 5 minute window) of > 80 beats/min. $E(S3) = 0.01\text{nj/Sample}$

Calculation:

NAC (Normalized Acquisition Cost)

- = Sample rate * Energy/Failure Rate
- Considering Conjunctive Query (AND)
 - $NAC(S1) = 100 * 0.02 / 0.05 = 40$
 - $NAC(S2) = 5 * 0.02 / 0.95 = 0.105$
 - $NAC(S3) = 10 * 0.01 / 0.8 = 0.125$
 - Best Sequence = {S2, S3, S1}

Query (2)

- Consider a hypothetical activity/wellness tracking application that seeks to detect an episode where an individual
 - (i) walks for 10 minutes, OR
 - (ii) while being exposed to an ambient temperature (95th percentile over the 10 minute window) of greater than 80F, OR
 - (iii) while exhibiting an AVERAGE heart rate (over a 5 minute window) of > 80 beats/min.

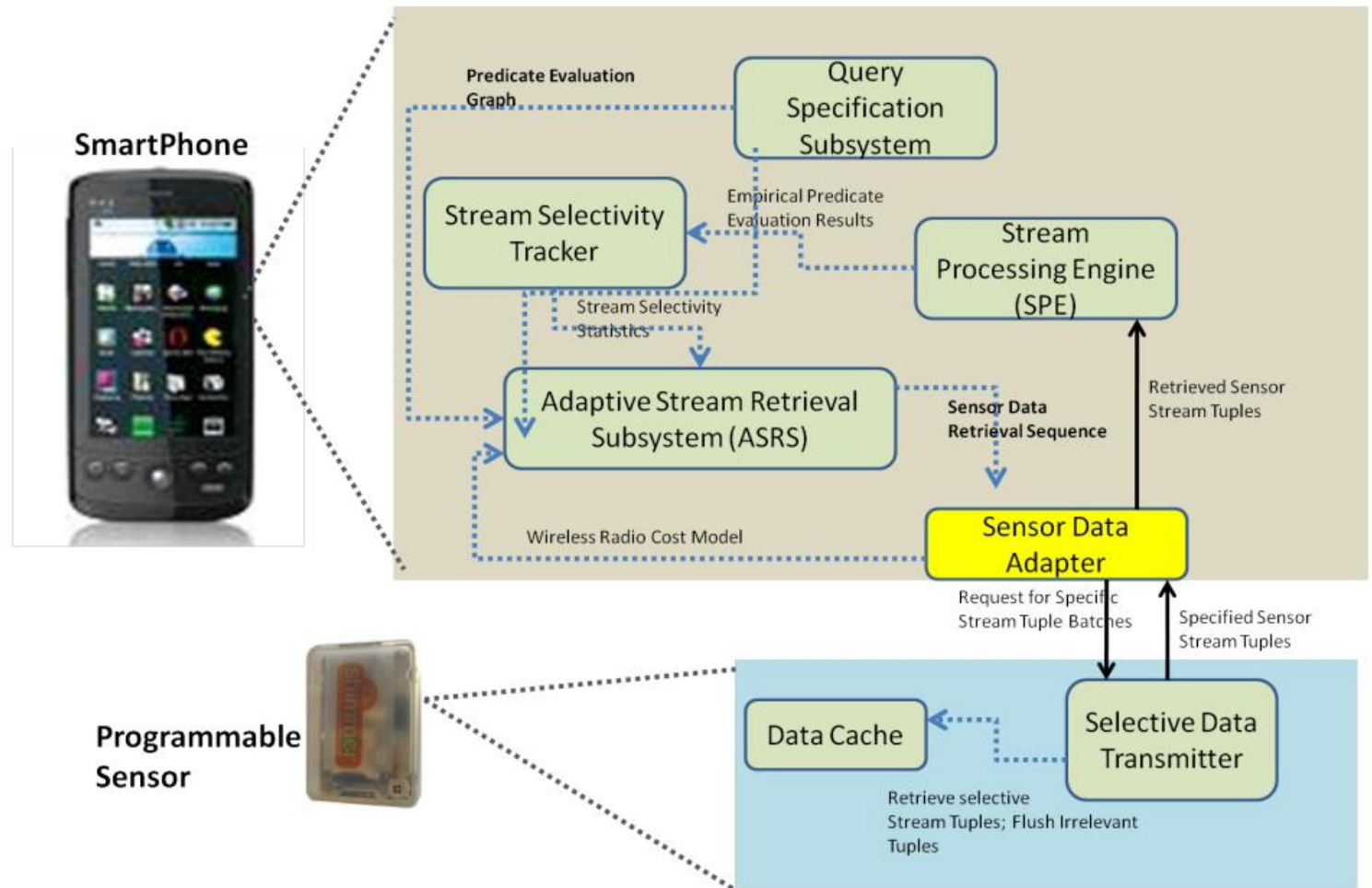
Probability of Success

- Consider a the probability of the given conditions occurring at that time be as follows;
- walks for 10 minutes, -- 0.95 (Success)
- while being exposed to an ambient temperature (95th percentile over the 10 minute window) of greater than 80F, -- 0.05 (Success)
- while exhibiting an AVERAGE heart rate (over a 5 minute window) of > 80 beats/min. = 0.2

Normalized acquisition cost

- Considering Disjunctive Query (OR)
 - $NAC(S1) = 100 * 0.02 / 0.95 = 2.11$
 - $NAC(S2) = 5 * 0.02 / 0.05 = 2$
 - $NAC(S3) = 10 * 0.01 / 0.2 = 0.5$
 - Best Sequence = {S3, S2, S1}

System Architecture



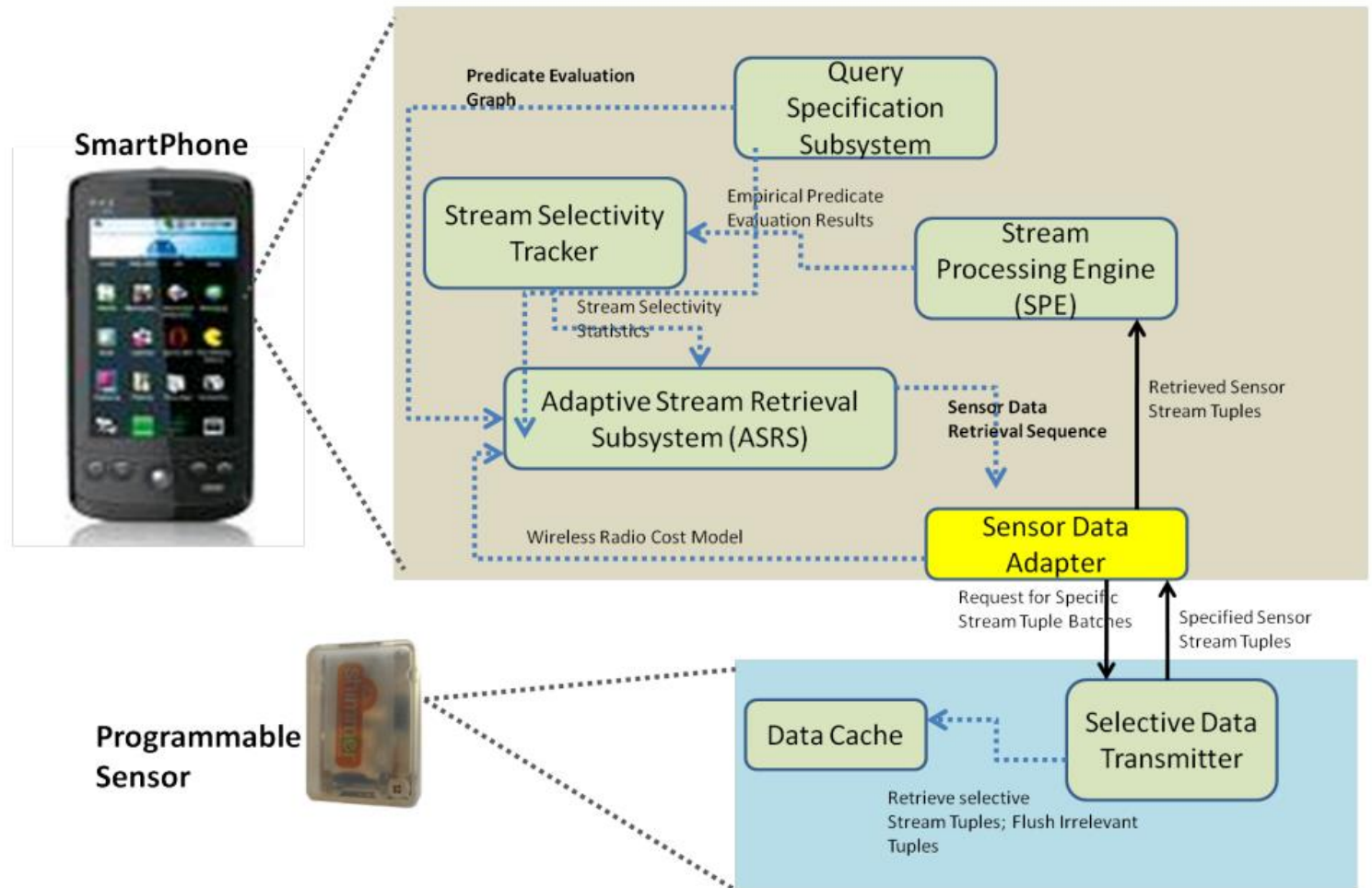
System Architecture

The heart of the ACQUA framework are the *Stream Selectivity Tracker (SST)* and the *Adaptive Stream Retrieval Subsystem (ASRS)* components. The SST is responsible for computing and establishing the selectivity properties of different sensor streams—in effect, computing the likely probability distribution of the values of each individual stream elements. To compute these values, ACQUA requires the SST to interface with the embedded Stream Processing Engine (SPE) to obtain the empirical observations of how the stream elements (individually or time-windows) satisfy different query predicates. The ASRS component is responsible for dynamically computing the sequence in which different (batches of) stream elements are retrieved by the smartphone from the locally-connected sensor. Note that the Stream Processing Engine (SPE) and the Sensor Data Adapter are pre-existing and non-ACQUA specific components needed to perform the basic functionality of a)

System Architecture

The heart of the ACQUA framework are the *Stream Selectivity Tracker (SST)* and the *Adaptive Stream Retrieval Subsystem (ASRS)* components. The SST is responsible for computing and establishing the selectivity properties of different sensor streams—in effect, computing the likely probability distribution of the values of each individual stream elements. To compute these values, ACQUA requires the SST to interface with the embedded Stream Processing Engine (SPE) to obtain the empirical observations of how the stream elements (individually or time-windows) satisfy different query predicates. The ASRS component is responsible for dynamically computing the sequence in which different (batches of) stream elements are retrieved by the smartphone from the locally-connected sensor. Note that the Stream Processing Engine (SPE) and the Sensor Data Adapter are pre-existing and non-ACQUA specific components needed to perform the basic functionality of a)

System Architecture



System Architecture

To algorithmically determine the best evaluation sequence, the ASRS also requires the knowledge of the energy per sample profile associated with different sensor devices and radios—it receives these specifications from the corresponding Sensor Data Adapter. As mentioned before, the ACQUA framework requires some degree of embedded data processing and storage capability on each individual sensor. In particular, the sensor-resident ACQUA components include the Data Cache, which acts as a temporary local repository for the stream tuples that may or may not be eventually pulled by the smartphone, and the Selective Data Transmitter, which is responsible for receiving requests for specific subsets of the stream tuples and for transmitting (in batches) these requested subsets.

System Architecture

sensor. Note that the Stream Processing Engine (SPE) and the Sensor Data Adapter are pre-existing and non-ACQUA specific components needed to perform the basic functionality of a) performing the appropriate query execution on the incoming data streams and b) interfacing with the sensor to retrieve the appropriate sensor samples. The Query Specification Subsystem

appropriate sensor samples. The Query Specification Subsystem is another ACQUA component that is responsible for receiving the various query specifications (associated with multiple applications) and for compiling them into a common Predicate Evaluation Graph. This graph is the data structure used by the ASRS algorithms to determine the preferred sequence in which data is pulled from individual sensor streams—the formal model for this graph will be presented shortly (in Section IV).

Query Specification and Representation

- We consider complex stream of queries
- Expressed as arbitrary conjunction or disjunction predicates over a set of stream-oriented SQL aggregate (e.g., MAX or AVG) or
- user defined functions,
- defined over a time-window of each individual sensor stream.

Q1: $\text{AVG}(A, 5) < 70 \text{ AND } (\text{MAX}(B, 4) > 100 \text{ OR } C < 3)$,

Q2: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR } (C < 3 \text{ AND } \text{Speed}(D, 2) < 1.0)$,

Q3: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR } (C < 3 \text{ AND } \text{MIN}(B, 7) < 80)$,

Query Specification and Representation

$Q ::= \text{Predicate} \mid (Q \text{ AND } Q) \mid (Q \text{ OR } Q)$
 $\text{Predicate} ::= \text{AggFunc} (\text{SExp}, w) \text{CmpOp Const} \mid$
 NOT Predicate
 $\text{SExp} ::= \text{StreamName} \mid$
 $\text{StreamExp ArithmeticOp Numeric}$

SQL function

Applied over a time window (t-w, t)

Sensor stream

Q1: $\text{AVG}(A, 5) < 70 \text{ AND } (\text{MAX}(B, 4) > 100 \text{ OR } C < 3)$,
Q2: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR}$
 $(C < 3 \text{ AND } \text{Speed}(D, 2) < 1.0)$,
Q3: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR}$
 $(C < 3 \text{ AND } \text{MIN}(B, 7) < 80)$,

Query Trees

All such queries are compiled into a uniform Query Tree representation,

Internal node is associated with a boolean conjunction or disjunction operator.

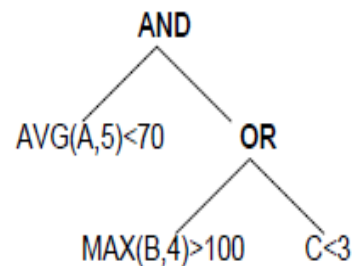
leaf node is associated with a predicate.

This query tree provides us the unifying application-independent query representation framework;

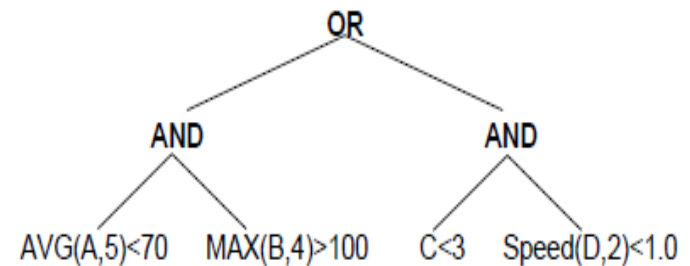
hence, the specific ASRS algorithms are defined in terms of such a query tree.

Query Trees

Q1: $\text{AVG}(A, 5) < 70 \text{ AND } (\text{MAX}(B, 4) > 100 \text{ OR } C < 3)$,

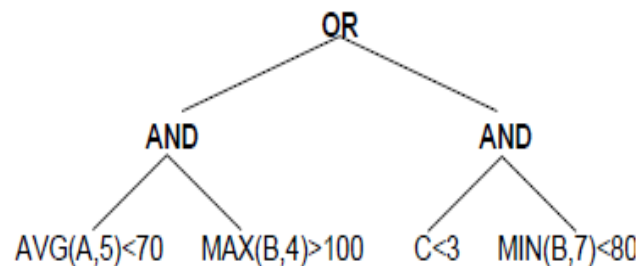


(a)



(b)

Q2: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR } (C < 3 \text{ AND } \text{Speed}(D, 2) < 1.0)$,



(c)

Q3: $(\text{AVG}(A, 5) < 70 \text{ AND } \text{MAX}(B, 4) > 100) \text{ OR } (C < 3 \text{ AND } \text{MIN}(B, 7) < 80)$,

Query Evaluation period (ω)

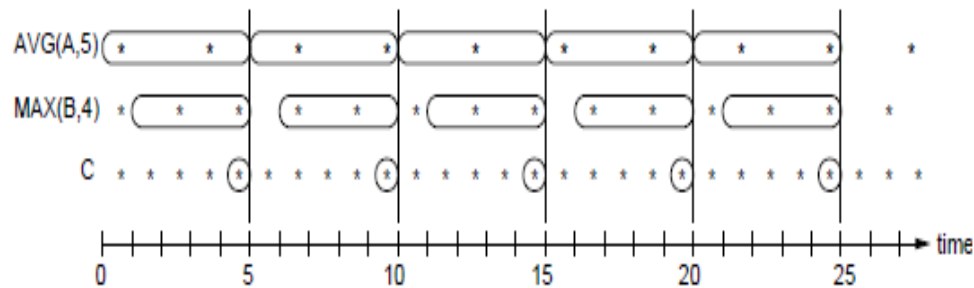
We consider queries defined over **tumbling windows** of the individual sensor data streams.

Formally, this implies the notion of a 'time shift' value $\omega(Q)$ associated with a query Q , such that the query is evaluated repeatedly at the time instants $t = (\omega; 2\omega; 3\omega; \dots)$.

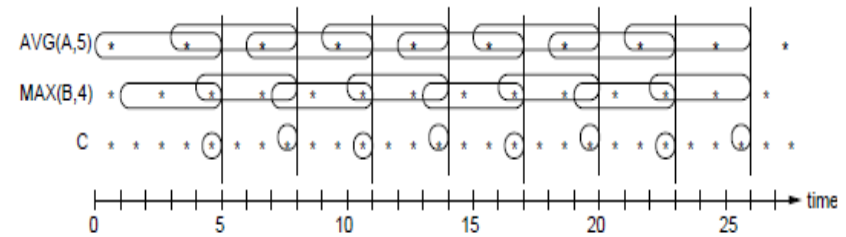
Note that the time-shift value ω is distinct from the time windows associated with the individual predicates and operators of the query Q .

For example, a specific query may be defined to perform an $\text{AVG}(5)$ operation i.e., an average of the last 5 seconds worth of sensor data with $\omega = 7$; in this case, the query would be evaluated at $t = 0$ over the stream tuples belonging to the time window $(-5; 0)$, and then again at time $t = 7$ over the time window $(2, 7)$.

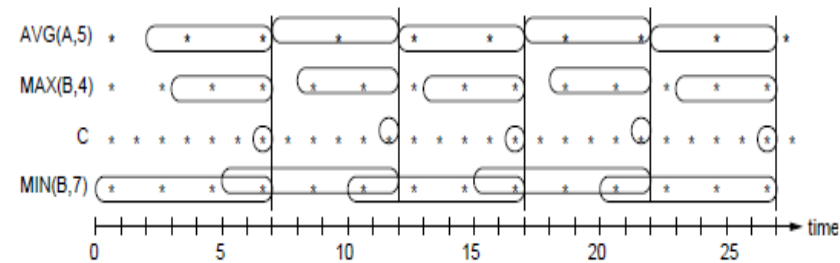
Relationship between query evaluation period, predicate time windows, and stream rate



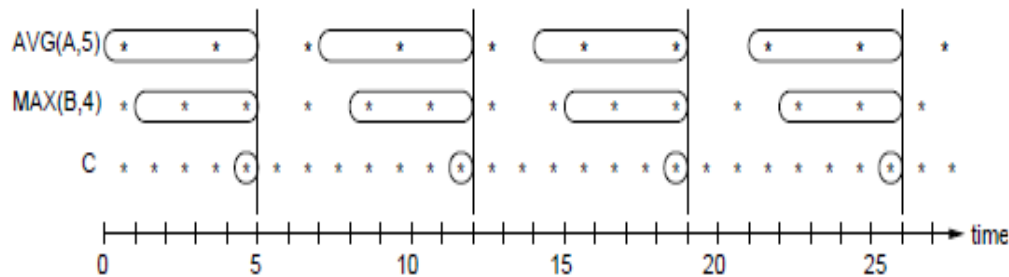
(a) $Q1 : \omega=5$



(b) $Q1 : \omega=3$

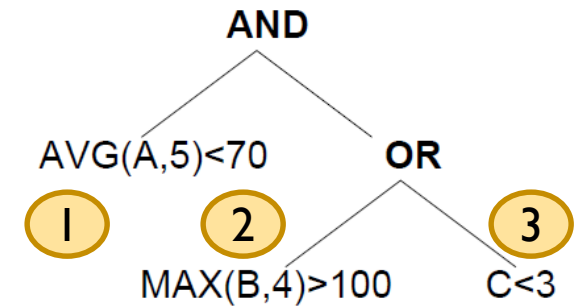


(d) $Q3 : \omega=5$



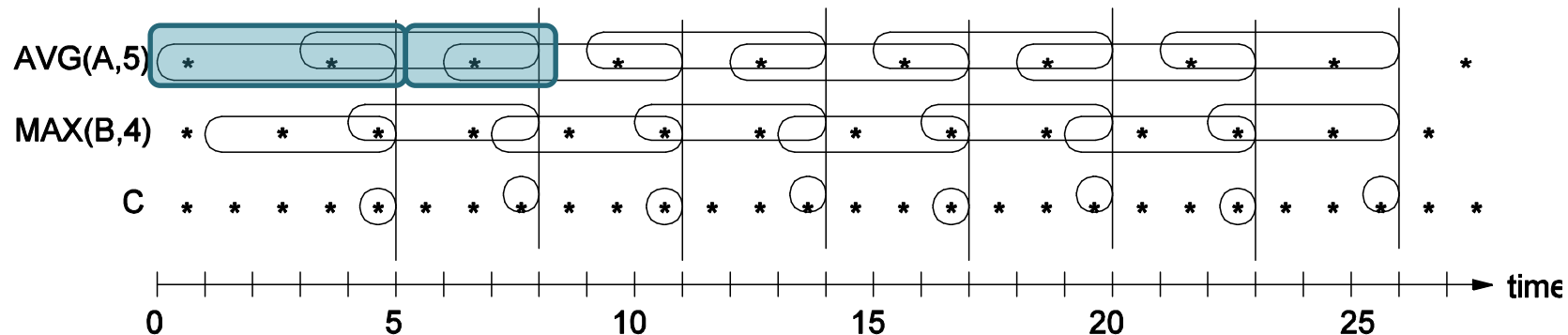
(c) $Q1 : \omega=7$

Example: $\omega=3$



- **Time 8:** acquisition cost for A becomes cheaper, because some tuples are already in buffer

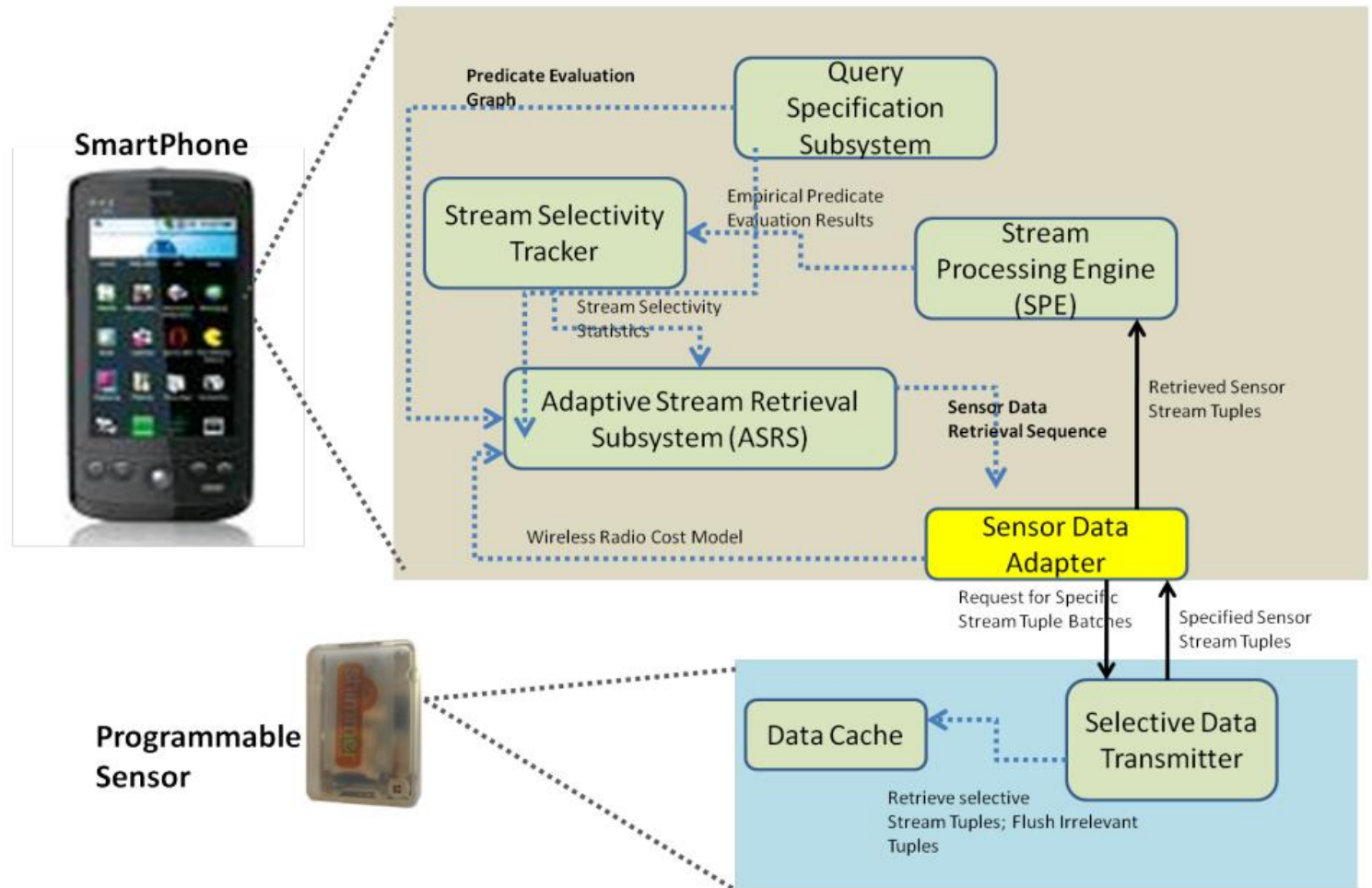
Acquisition cost depends on state of the buffer at time t



The **acquisition cost for a particular sensor stream may be different** at different evaluation instants,

Depending upon the data tuples that may have been acquired during prior event processing.

System Architecture



ASRS sequential retrieval Algorithm:

The algorithm first computes the lowest expected cost of evaluating different portions of the query sub-trees, and thereby determines (using the recursive Algorithm CALCACQUISITIONCOST) the optimal sequence for retrieving the data from the different sensor streams.

Subsequently, the actual query is evaluated using the recursive Algorithm 3 EVALUATEQUERY, which essentially follows the specified sequence to evaluate the sub-trees

Algorithm 1 PROCESSQUERY(q, P, ω)

Input: Query tree q , probability P of each subquery evaluating to true/false, evaluation period ω

Output: Alert Stream

- 1: **loop**
 - 2: $t \leftarrow$ current time
 - 3: CALCACQUISITIONCOST(q, t, P, C)
 - 4: **if** EVALUATEQUERY(q, t, P, C) = *true* **then**
 - 5: output alert tuple
 - 6: sleep ω seconds
-

Algorithm 2 CALCACQUISITIONCOST(q, t, P, C)

Input: Query tree q , current time t , probability function P , data acquisition cost function $C(\cdot)$

Output: Updates cost function $C(\cdot)$

- 1: **if** q is a predicate node **then**
 - 2: let s be the stream that q operates on, w be the window size of q , t_s be the latest time the buffer for s was updated.
 - 3: $C(q) \leftarrow$ Calculate cost for acquiring the samples in time interval $(\max(t - w, t_s), t]$ for stream s using Eqn [1](#) or Eqn [2](#).
 - 4: **else**
 - 5: CALCACQUISITIONCOST($q.left, t, P, C$)
 - 6: CALCACQUISITIONCOST($q.right, t, P, C$)
 - 7: **if** $q.op = \text{AND}$ **then**
 - 8: $C(q) \leftarrow$ Eqn. [3](#)
 - 9: **else**
 - 10: $C(q) \leftarrow$ Eqn. [4](#)
-

IEEE 802.11:

1) *IEEE 802.11*: Commercial IEEE 802.11 radios can operate in two states—a normal 'active' mode (when the radio interface receives or transmits packets) and a Power Save Mode (PSM), where the radio periodically wakes up to check if there any pending transmissions or receptions. The following are two key relevant properties associated with 802.11 hardware:

- Due to the switching characteristics of the radio hardware, there is typically a lower bound on the minimal idle time Th_{idle} , below which the radio cannot enter the PSM mode (typically, this is around 100 ms) [9].
- There is a fixed, *duration-independent* switching energy E_{switch} spent when a radio transitions from the PSM to the 'active' mode.

$$E_t = \begin{cases} P_i * \left(\frac{N}{f} - \frac{N*S}{B} \right) \\ \quad + P_a * \frac{N*S}{B} + E_{switch} & \text{if } \frac{N}{f} - \frac{N*S}{B} > Th_{idle} \\ P_a * \frac{N}{f} & \text{otherwise} \end{cases}$$

Bluetooth:

the smartphone, which primarily receives data from an external sensor, we denote its active energy consumption P_a as the energy spent in actively receiving data. We consider the Bluetooth version 2.0+ EDR and assume, for analytical tractability, that a single sensor device attaches as a slave to the master located on the smartphone. While the low-power mode results in significantly low power consumption, note that there is a latency T_{switch} involved in switching from the non-associated low-power mode to the associated-active mode. Accordingly, any data transfer duration would consist of the total time spent in transfer $\frac{N*S}{B}$, plus the additional time T_{switch} . Accordingly, the total energy consumed in transmitting the sensor stream in batches of N samples is given by:

$$E_t = P_i * \left(\frac{N}{f} - \frac{N * S}{B} - T_{switch} \right) + P_a * \left(\frac{N * S}{B} + T_{switch} \right) \quad (2)$$

Expected cost

For AND

$$C(q) = \begin{cases} P(q.left) \times [C(q.left) + C(q.right)] \\ \quad + P(\neg q.left) \times C(q.left) & \text{if LR} \\ P(q.right) \times [C(q.left) + C(q.right)] \\ \quad + P(\neg q.right) \times C(q.right) & \text{if RL} \end{cases} \quad (3)$$

For OR

$$C(q) = \begin{cases} P(\neg q.left) \times [C(q.left) + C(q.right)] \\ \quad + P(q.left) \times C(q.left) & \text{if LR} \\ P(\neg q.right) \times [C(q.left) + C(q.right)] \\ \quad + P(q.right) \times C(q.right) & \text{if RL} \end{cases} \quad (4)$$

Algorithm 3 EVALUATEQUERY(q, t, P, C)

Input: Query tree q , current time t , probability function P , data acquisition cost function $C(\cdot)$

Output: Truth value of q

```
1: if  $q$  is a predicate node then
2:   let  $s$  be the stream that  $q$  operates on,  $w$  be the window size
   of  $q$ ,  $t_s$  be the latest time the buffer for  $s$  was updated.
3:   Acquire the samples in time interval  $(\max(t - w, t_s), t]$  for
   stream  $s$ .
4:   Update  $C(\cdot)$  if  $s$  is used in multiple predicates
5:    $truthval \leftarrow$  evaluate predicate  $q$ 
6:   return  $truthval$ 
7: else
8:   if  $q.op = \text{AND}$  then
9:      $leftshortcircuits \leftarrow P(\neg q.left)$ 
10:     $rightshortcircuits \leftarrow P(\neg q.right)$ 
11:     $shortcircuitval \leftarrow false$ 
12:   else
13:      $leftshortcircuits \leftarrow P(q.left)$ 
14:      $rightshortcircuits \leftarrow P(q.right)$ 
15:      $shortcircuitval \leftarrow true$ 
16:    $evalorder \leftarrow (q.left, q.right)$ 
17:   if  $\frac{C(q.left)}{leftshortcircuits} > \frac{C(q.right)}{rightshortcircuits}$  then
18:      $evalorder \leftarrow (q.right, q.left)$ 
19:   for all  $q' \in evalorder$  do
20:      $truthval \leftarrow$  EVALUATEQUERY( $q', t, P, C$ )
21:     if  $truthval = shortcircuitval$  then
22:       return  $truthval$ 
23:   return  $\neg shortcircuitval$ 
```

Simulation Setup

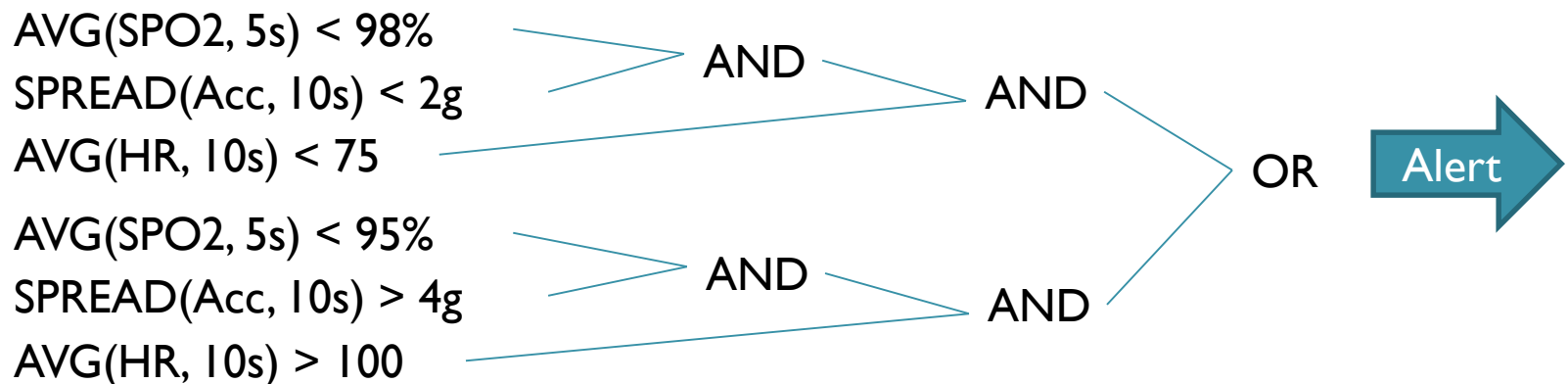



2. Sensor tuples are generated from probability distribution

Results are presented by averaging over 5 one hour long traces

Simulation Data & Query

- Data streams generated using independent Gaussian distribution
 - $SPO2 \sim N(96,4)$, 3 Hz, 3000 bits
 - $HR \sim N(80,40)$, 0.5 Hz, 32 bits
 - $Accel \sim N(0,10)$, 256 Hz, 196 bits





Intuitively, this query generates alerts either if the user's SpO₂ values drop below 98% while the user is resting, or if the SpO₂ values drop below 95% while the individual is engaged in vigorous activity (e.g., running). The accelerometer and SpO₂ sampling rates and data sizes are adapted from Fig. 3, while the heart rate sensor has a sampling frequency of 0.5 Hz and a sample size of 32 bits. We experimented with both 802.11 and

Simulation Setup

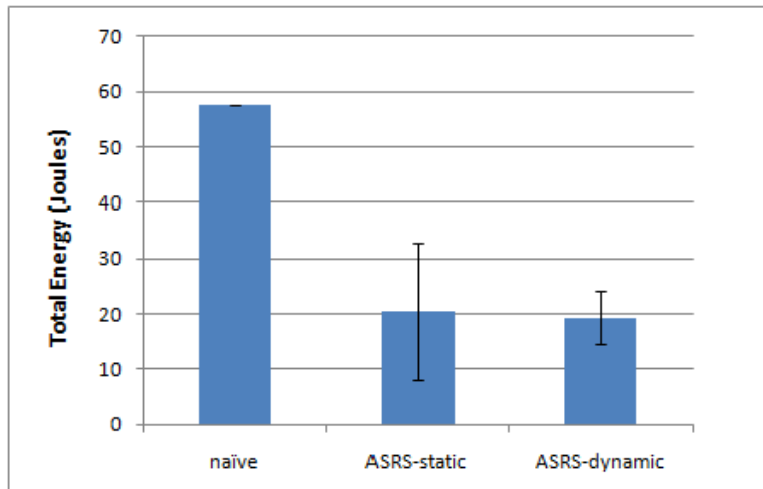
- **Naive**
 - Upload data from all sensors acquired in batches to SPE
- **ASRS-static**
 - Evaluation order determined once at initialization and never changes
 1. It computes an optimal sequence only once (at the beginning of the simulation) based on the selectivity characteristics and the communication costs,
 2. then applies the EvaluateQuery() procedure to evaluate the query tree at successive 'time shift' instants.

Accordingly, it does not perform the dynamic update of NAC values, based on the dynamically evolving state of the query processing state.
- **ASRS-dynamic**
 - Evaluation order determined at each ω time period.

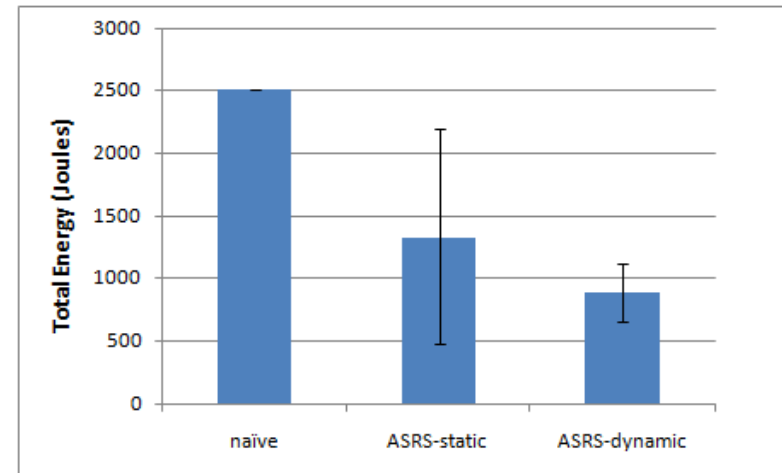
Dynamic modification of the acquisition cost functions after each data retrieval and evaluation, to account for

 - (a) the stream tuples already present in the smartphone buffer and
 - (b) the already-resolved ('shortcircuited') query subtrees.

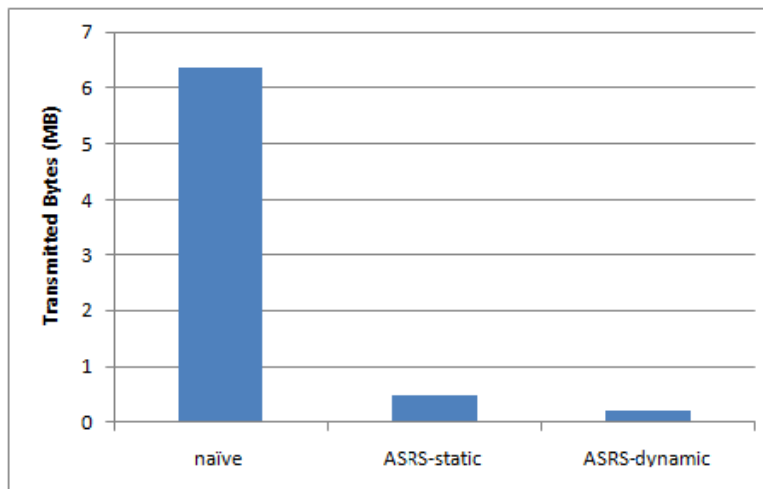
Evaluation: Fixed Time-Shift Value for Each Stream



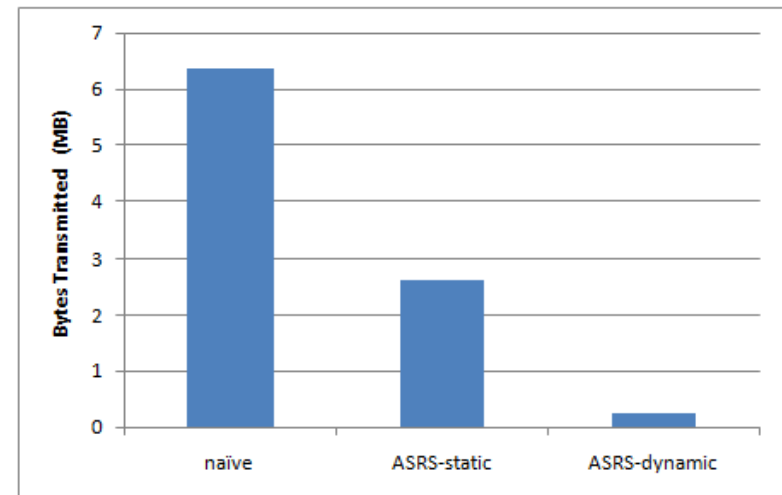
(a) Total Energy (Bluetooth)



(b) Total Energy (WiFi)



(c) Total Bytes of Sensor Data (Bluetooth)



(d) Total Bytes of Sensor Data (WiFi)

Evaluation: Fixed Time-Shift Value for Each Stream

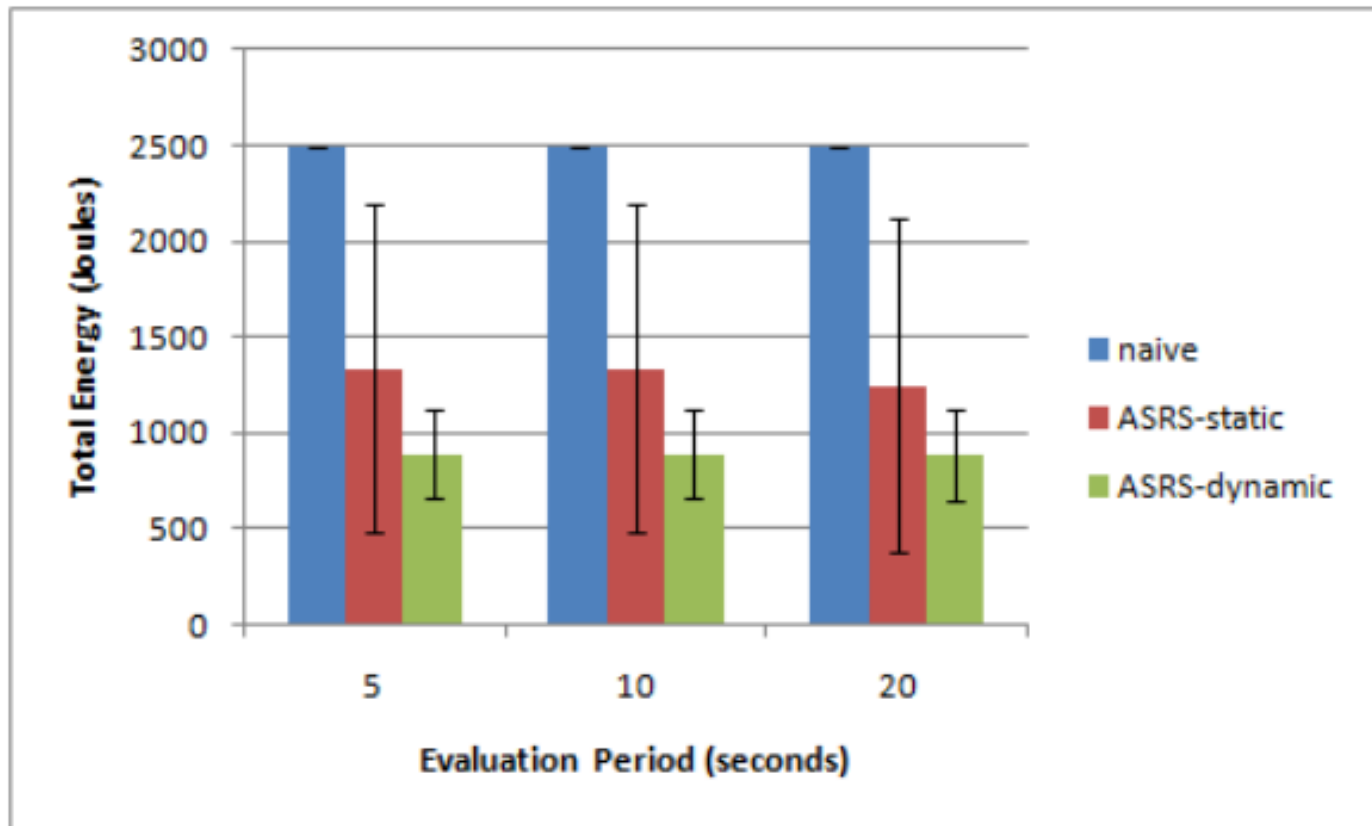
Figures 7(a) and 7(b) plot the total data acquisition energy (in Joules, over the 1 hour evaluation duration) for each of the three strategies, for the case of Bluetooth and 802.11-based PAN technologies respectively. These results correspond to a query with a time-shift value of $\omega = 10secs$. It is easy to see that our approach of sequential retrieval and evaluation of individual sensor streams, while taking into account their respective acquisition costs and selectivity characteristics, results in significant energy savings, compared to the naive approach where the data is pushed (albeit in batches) from each sensor. In particular, for 802.11 based transmissions, ASRS-static and ASRS-dynamic result in $\sim 50\%$ and $\sim 70\%$ reduction in energy overheads compared to the Naive scheme. For Bluetooth-based data transfers, the energy reductions are equally dramatic, with ASRS-static and ASRS-dynamic both achieving $\sim 70\%$ savings in energy overheads.

Evaluation: Fixed Time-Shift Value for Each Stream

The results also demonstrate the benefits of ASRS-dynamic: by taking the dynamic state of a query and the contents of the data buffer into account, this approach is able to further reduce the energy overhead, compared to the static counterpart. The gains are, however, not as dramatic for the Bluetooth interface (even though ASRS-dynamic has significantly lower variance

Figures 7(c) and 7(d) similarly plot the total data overhead (in bytes). While the ASRS algorithms clearly require an order-of-magnitude less communication than the Naive counterpart, it is interesting to note that the energy savings are not directly proportional to the communication overheads. For example, with Bluetooth, ASRS-dynamic requires about 50% fewer bytes of sensor data than ASRS-static, but has only a $\sim 10\%$ lower energy overhead.

Evaluation: Varying Time-Shift Values



The relative gains are fairly independent of the time-shift values

data tuples. Nonetheless, the ASRS-dynamic algorithm is able to outperform the static variant, by better adapting its data acquisition sequence to take account of the intermediate query evaluations state (i.e., by eliminating data acquisition for those sub-trees that have already been ‘short-circuited’).

Functional Requirements (I)

- Accommodate Heterogeneity in Sensor Data Rates, Packet Sizes and Radio Characteristics
 - Sensor data streams exhibit significant heterogeneity in terms of
 - (i) data rates (sensor samples/sec)
 - (ii) data sizes (bytes/sample)
 - (iii) energy cost with radio interfaces

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
Accelerometer	64	3	100 Hz
Temperature	20	1	256 Hz

Communication energy cost depends on sensor type as well as specific wireless radio implementation

Functional Requirements (2)

Adapt to Dynamic Changes in Query Selectivity Properties:

To apply ACQUA, it is extremely important to have **correct estimates for the query selectivity properties** of different **sensor data streams**.

However, we need to keep in mind that these selectivity properties are **not only individualized**, but also **vary dramatically over time** due to changes in an individual's activity.

For example, the **likelihood** of HR samples exceeding 80 might be **very low** when a person is engaged in **office activity**, but will be **very high** when the person is **walking** or working out in the gym.

Accordingly, the ACQUA framework must be capable of using context to **accurately predict** (albeit statistically) the selectivity characteristics of different sensor streams.

Functional Requirements (3)

Take into Account other Objectives Besides Energy Minimization:

Operating with a **heterogeneous set of sensors** implies that energy minimization, while important, might not be the only objective of interest to a user of ACQUA.

For example, it is possible that one of the N sensors might have very little battery capacity

In such a case, **to extend the overall operational lifetime** of the context detection activity,

preferentially retrieve and process data from an alternative sensor, even though the selectivity characteristics of the alternative sensor may not be the highest.

Functional Requirements (4)

Support Multiple Queries and Heterogeneous Time Window Semantics:

Sensor-based context extraction is becoming an intrinsic feature of a variety of smartphone applications that may be executing concurrently.

Different applications may specify distinct predicates over a shared set of sensor streams

for example, the **accelerometer sensor** may be used to both evaluate **step-counts** in a **wellness monitoring application** and to understand the user's current **mode of transport** in a separate **social networking application**.

The query predicates would differ not just in their **predicate logic**, but also in the **time windows** over which the stream query semantics are expressed.

Accordingly, ACQUA must support a **unified application-independent query representation framework** that is able to optimize the evaluation sequence across all concurrently executing stream queries.