



# Information Management Research

**Lipyeow Lim**

Assistant Professor

Information and Computer Sciences

<http://www2.hawaii.edu/~lipyeow/>

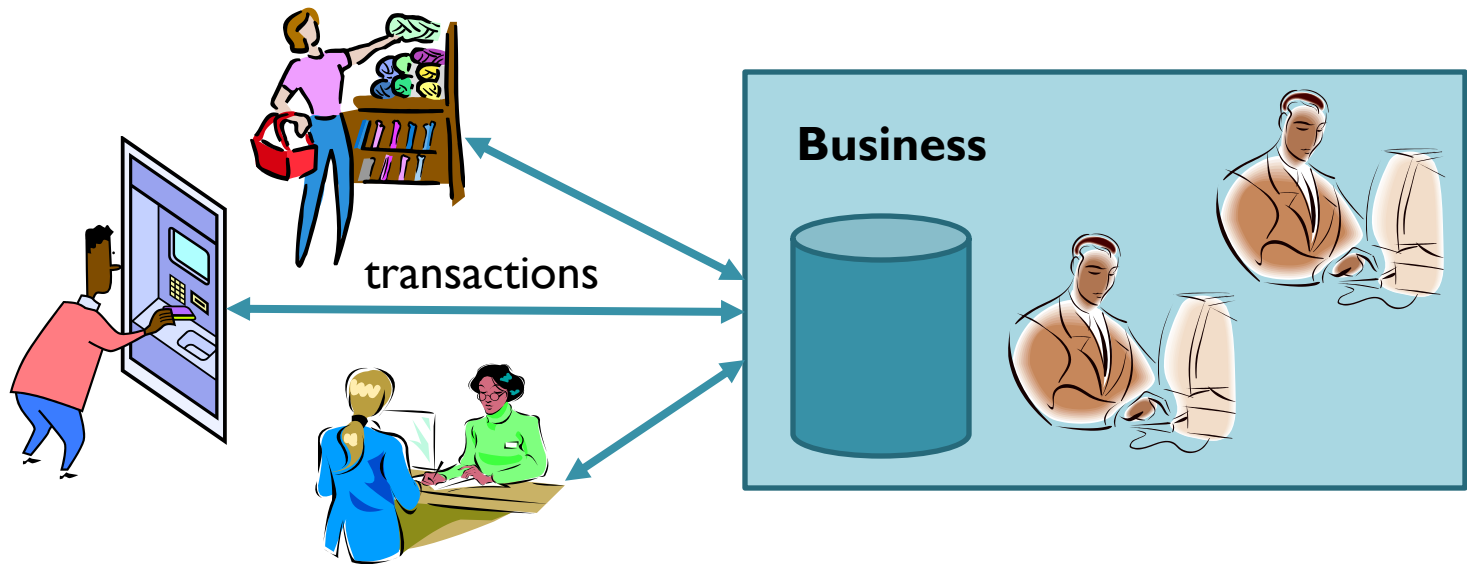
# Agenda

- Motivating Videos
  - <http://www.youtube.com/watch?v=EVL3I2zbEKg&feature=relmfu>
- Overview of Information Management
- Research Topic #1
  - Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search
- Research Topic #2
  - Optimizing Sensor Data Acquisition for Energy-Efficient Smartphone-based Continuous Event Processing

# Information Management

- Data = Information ?
- Core questions:
  - What is the best way to store data?
  - How do we query and/or update the data?
  - How to speed up queries
- Research Drivers:
  - New applications.
  - New data types. New query types

# Traditional Information Management



Pick a business or application

- What data needs to be stored ?
- What operations are needed on the data ?
- How should the data be stored ?
- How should the operations access the data ?

# A Bit of History

- 1970 **Edgar F Codd** (aka “Ted”) invented the **relational model** in the seminal paper “A Relational Model of Data for Large Shared Data Banks”
  - Main concept: relation = a table with rows and columns.
  - Every relation has a schema, which describes the columns.
- Prior 1970, no standard data model.
  - Network model used by Codasyl
  - Hierarchical model used by IMS
- After 1970, IBM built System R as proof-of-concept for relational model and used **SQL** as the query language. SQL eventually became a standard.

# Data, Database, DBMS

- A **database** : a collection of related data.
  - Represents some aspect of the real world (aka universe of discourse).
  - Logically coherent collection of data
  - Designed and built for specific purpose
- **Data** are known facts that can be recorded and that have implicit meaning.
- A **data model** is a collection of concepts for describing data.
- A **schema** is a description of a particular collection of data, using the a given data model.

# DBMS

- A **database management system (DBMS)** is a collection of programs that enables users to
  - **Create** new DBs and specify the structure using data definition language (DDL)
  - **Query** data using a query language or data manipulation language (DML)
  - **Store** very large amounts of data
  - Support **durability** in the face of failures, errors, misuse
  - Control **concurrent** access to data from many users

# Types of Databases

- On-line Transaction Processing (**OLTP**)
  - Banking
  - Airline reservations
  - Corporate records
- On-line Analytical Processing (**OLAP**)
  - Data warehouses, data marts
  - Business intelligence (BI)
- Specialized databases
  - Multimedia
  - XML
  - Geographical Information Systems (GIS)
  - Real-time databases (telecom industry)
- Special Applications
  - Customer Relationship Management (CRM)
  - Enterprise Resource Planning (ERP)
- Hosted DB Services
  - Amazon, Salesforce



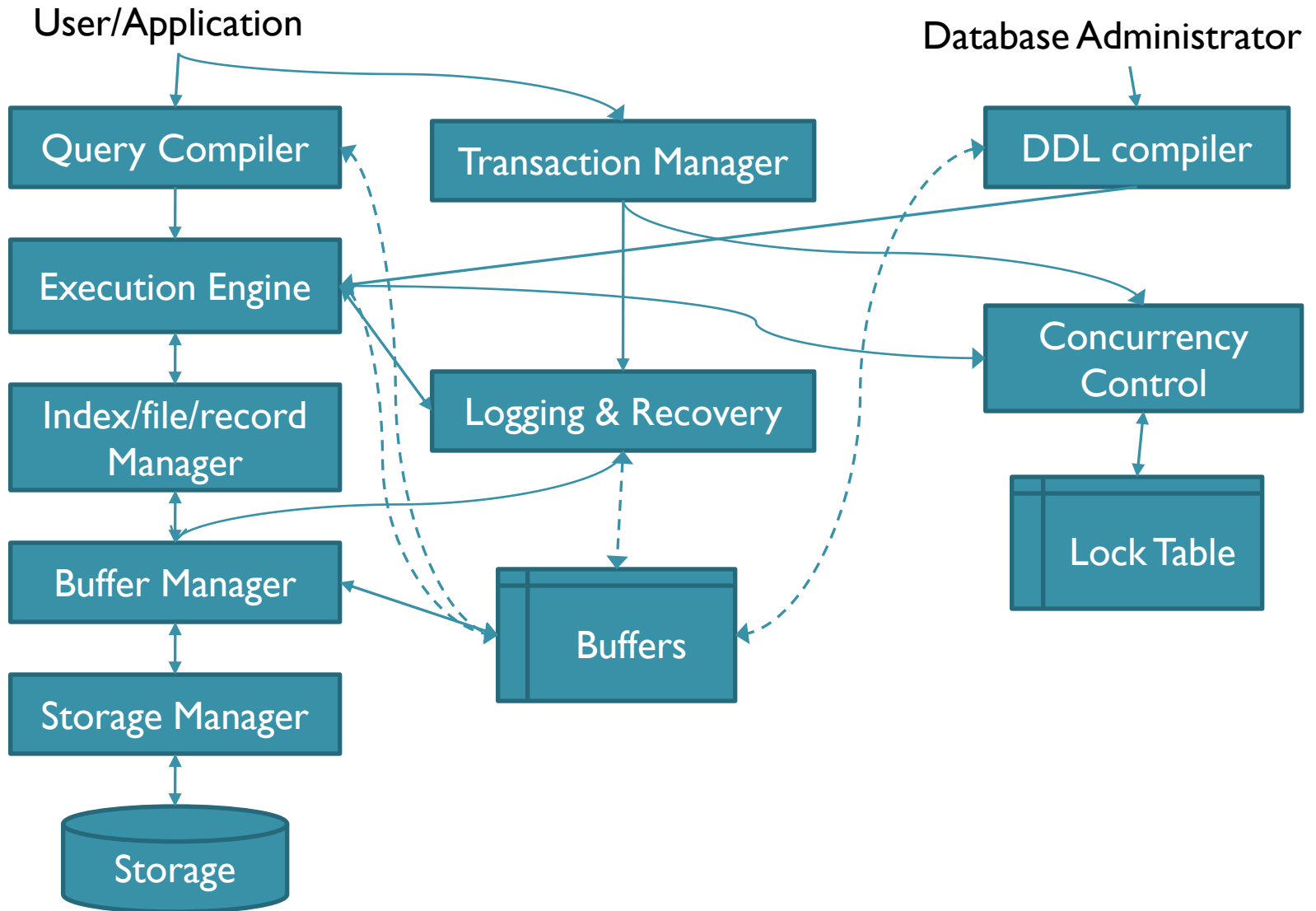
# Files vs DBMS

- Swapping data between memory and files
- Difficult to add records to files
- Security & access control
- Do optimization manually
- Good for small data/files
- Run out of pointers (32bit)
- Code your own search algorithm
  - Search on different fields is difficult
- Must protect data from inconsistency due to concurrency
- Fault tolerance – crash recovery

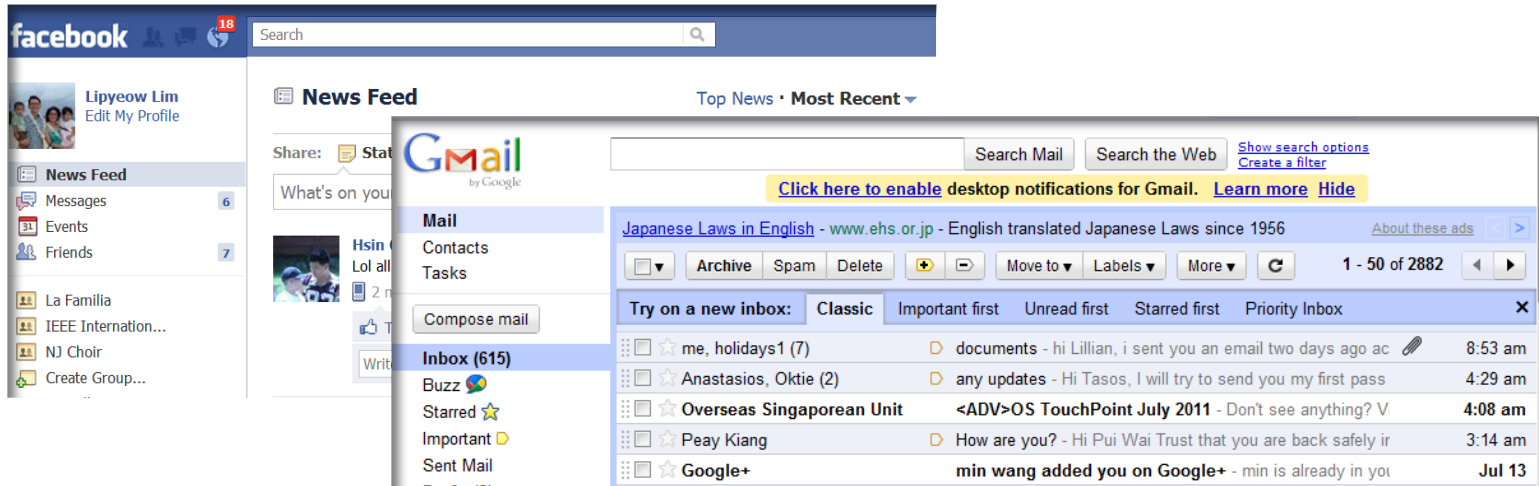
# Why use a DBMS ?

- Large datasets
- Concurrency/ multi-user
- Crash recovery
- Declarative query language
- No need to figure out what low level data structure
- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.

# DBMS Components

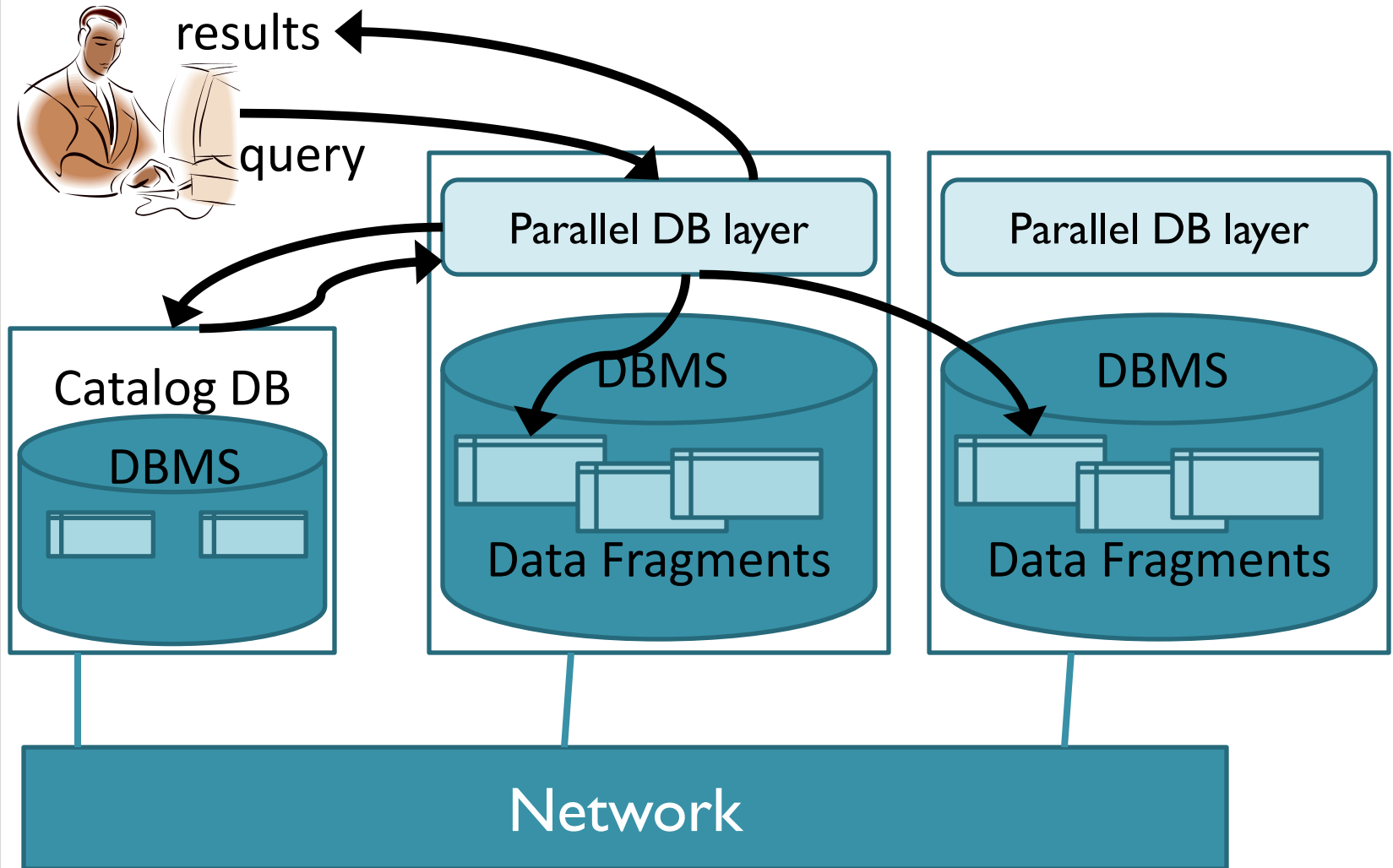


# Internet Information Management



- Millions of users
- Lots of data (peta-bytes =  $10^{15}$  bytes!)
- Startups prefer cheap computers
- Store data in very large clusters of cheap computers

# Parallel DBMS Architecture



# Horizontal Fragmentation: Range Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

## Range Partition on rating column

- Partition 1:  $0 \leq \text{rating} < 5$
- Partition 2:  $5 \leq \text{rating} \leq 10$

Partition 1

sid	sname	rating	age
29	brutus	1	33
32	andy	4	23

Partition 2

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35
64	horatio	7	35

# Range Partition: Query Processing

- Which partitions?
- Better than non-parallel ?

```
SELECT *  
FROM Sailors S
```

```
SELECT *  
FROM Sailors S  
WHERE rating = 2
```

```
SELECT *  
FROM Sailors S  
WHERE age > 30
```

```
SELECT *  
FROM Sailors S  
WHERE rating < 2 and age < 30
```

Partition 1

sid	sname	rating	age
29	brutus	1	33
32	andy	4	23

Partition 2

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35
64	horatio	7	35

# Horizontal Fragmentation: Hash Partition

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
31	lubber	8	55
32	andy	4	23
58	rusty	10	35
64	horatio	7	35

Partition 1

sid	sname	rating	age
31	lubber	8	55
32	andy	4	23
58	rusty	10	35

Partition 2

sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
64	horatio	7	35

- Hash partitioning using hash function
  - $\text{Partition} = \text{rating} \bmod 2$



# Hash Partition: Query Processing

- Which partitions?
- Better than non-parallel ?

```
SELECT *  
FROM Sailors S
```

```
SELECT *  
FROM Sailors S  
WHERE rating = 2
```

```
SELECT *  
FROM Sailors S  
WHERE age > 30
```

```
SELECT *  
FROM Sailors S  
WHERE rating < 2 and age < 30
```

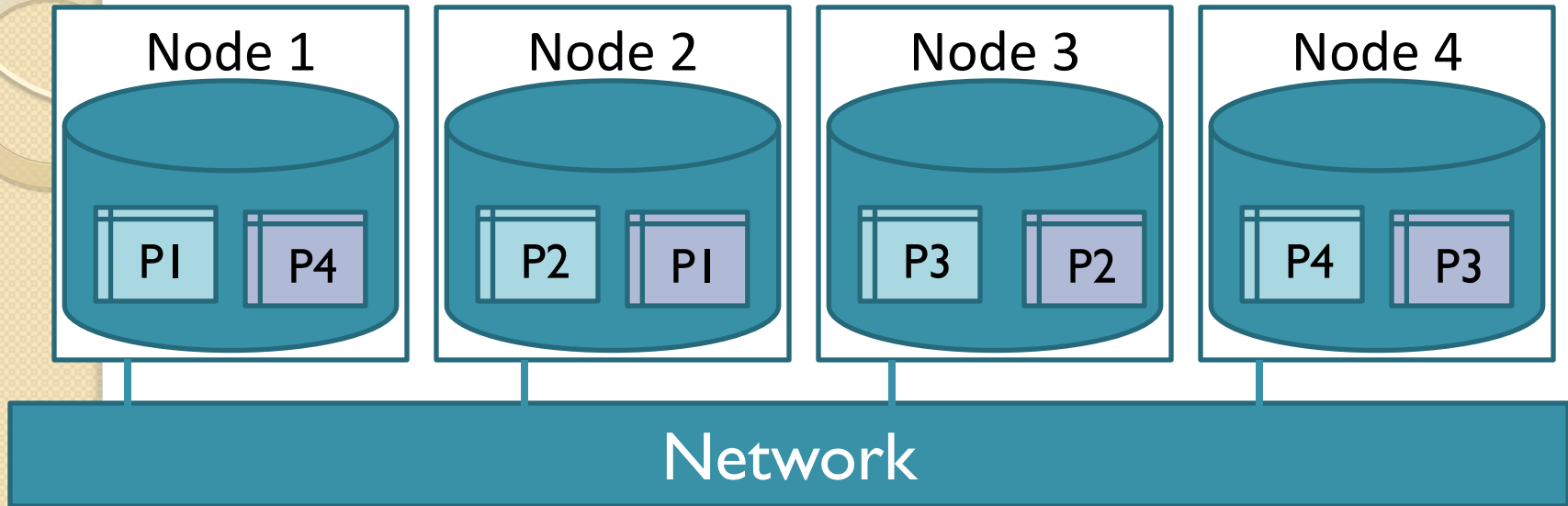
Partition 1

sid	sname	rating	age
31	lubber	8	55
32	andy	4	23
58	rusty	10	35

Partition 2

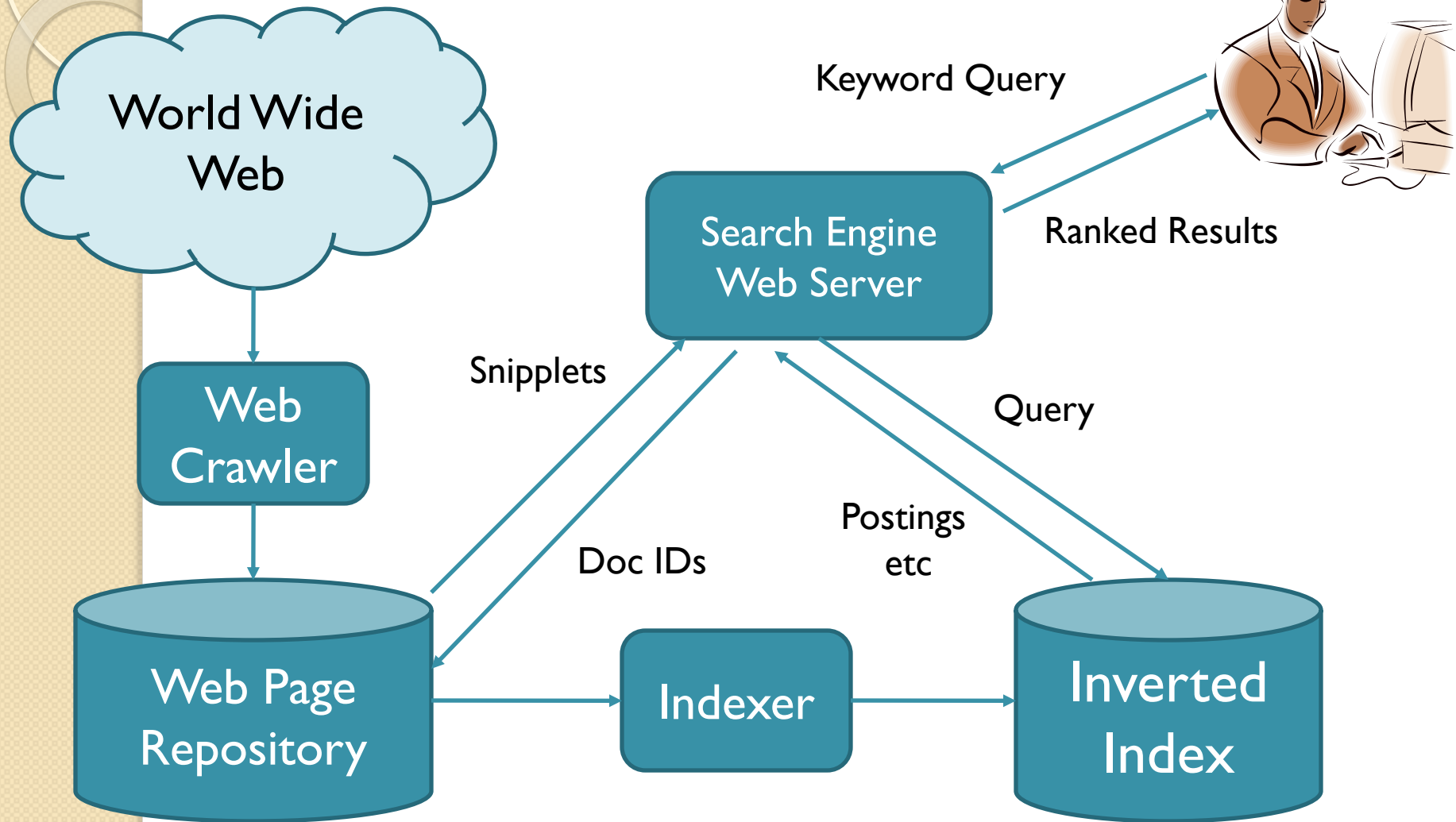
sid	sname	rating	age
22	dustin	7	45
29	brutus	1	33
64	horatio	7	35

# Fragmentation & Replication



- Suppose table is fragmented into 4 partitions on 4 nodes
- Replication stores another partition on each node
  - What happens when 1 node fails ? 2 nodes ?
  - What happens when a row needs to be updated ?

# Internet Search Engines



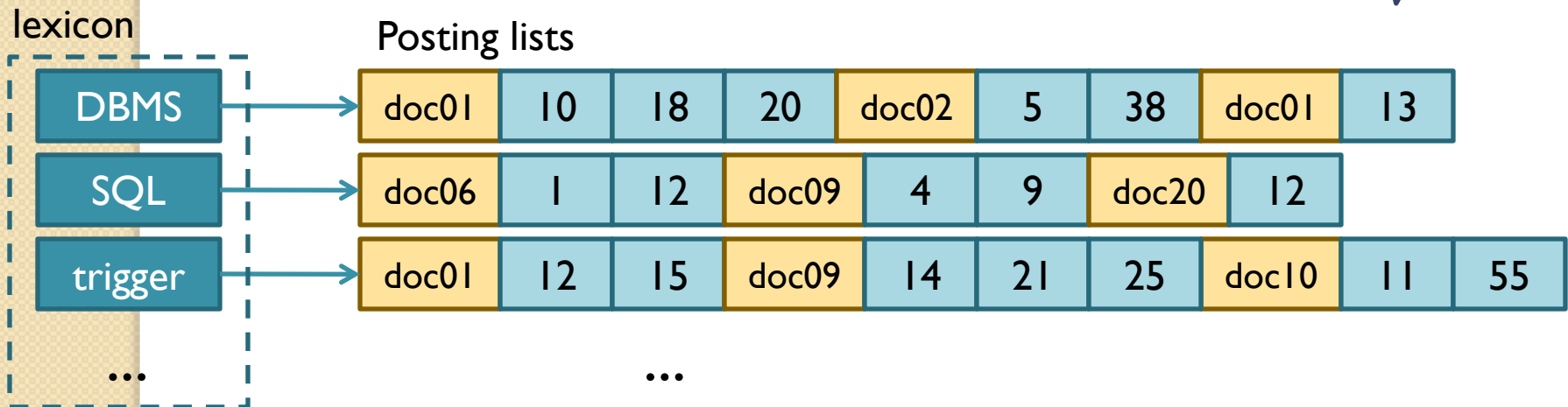
# Unstructured Text Data

- Field of “Information Retrieval”
- Data Model
  - Collection of documents
  - Each document is a **bag of words** (aka terms)
- Query Model
  - **Keyword** + Boolean Combinations
  - Eg. DBMS and SQL and tutorial
- Details:
  - Not all words are equal. “**Stop words**” (eg. “the”, “a”, “his” ...) are ignored.
  - **Stemming** : convert words to their basic form. Eg. “Surfing”, “surfed” becomes “surf”

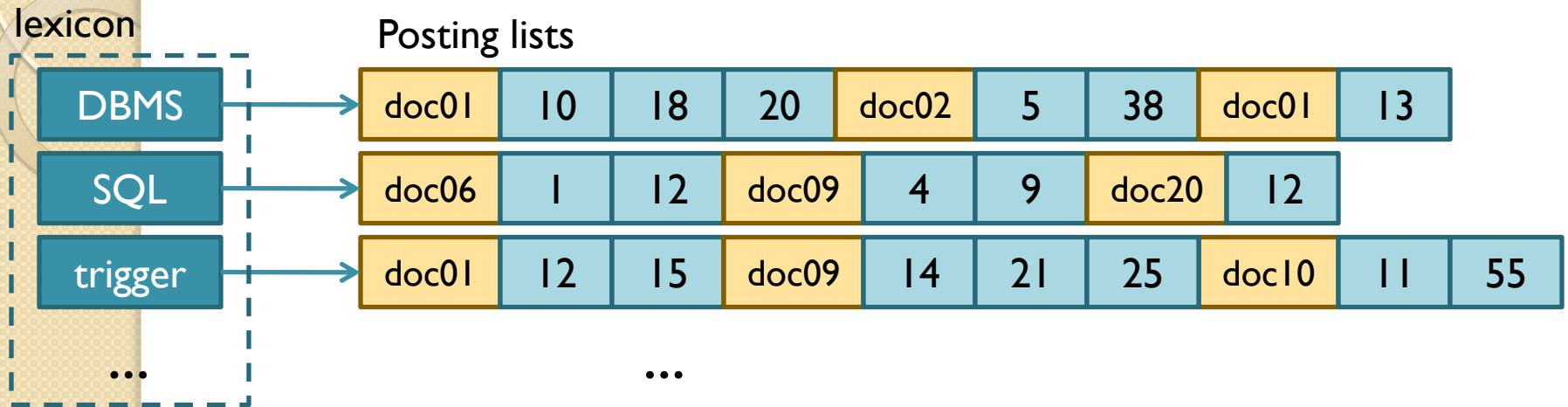
# Inverted Indexes

- Recall: an index is a mapping of search key to data entries
  - What is the search key ?
  - What is the data entry ?
- Inverted Index:
  - For each term store a list of postings
  - A posting consists of <docid,position> pairs

What is the data in an inverted index sorted on ?



# Lookups using Inverted Indexes



- Given a **single keyword query “k”** (eg. SQL)
  - Find k in the lexicon
  - Retrieve the posting list for k
  - Scan posting list for document IDs [and positions]
- What if the query is **“k1 and k2”** ?
  - Retrieve document IDs for k1 and k2
  - Perform intersection

# Too Many Matching Documents

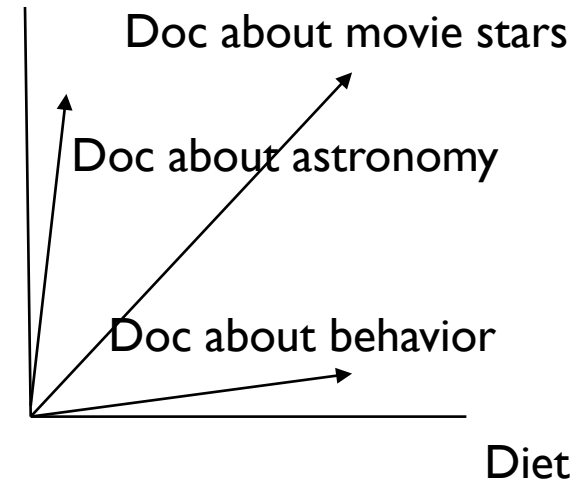
- Rank the results by “relevance”!

- Vector-Space Model

- Documents are vectors in hi-dimensional space
- Each dimension in the vector represents a term
- Queries are represented as vectors similarly
- Vector distance (dot product) between query vector and document vector gives ranking criteria
- Weights can be used to tweak relevance

- PageRank (homework)

Star



# Streaming Data Management

<http://www.youtube.com/watch?v=trvoc6GDAqo>



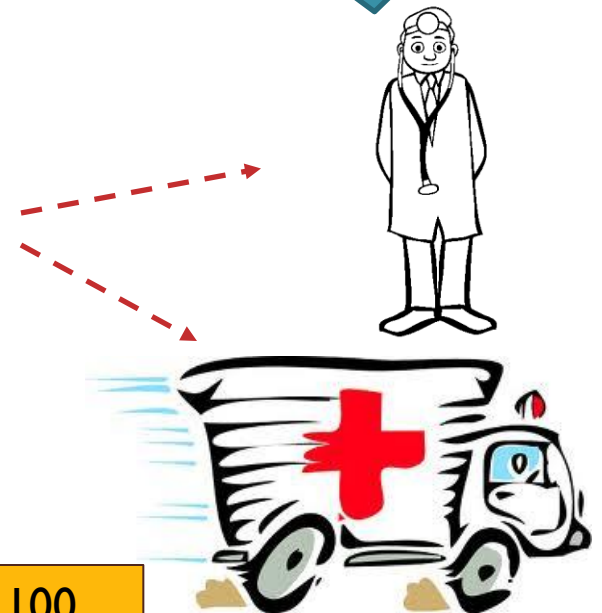
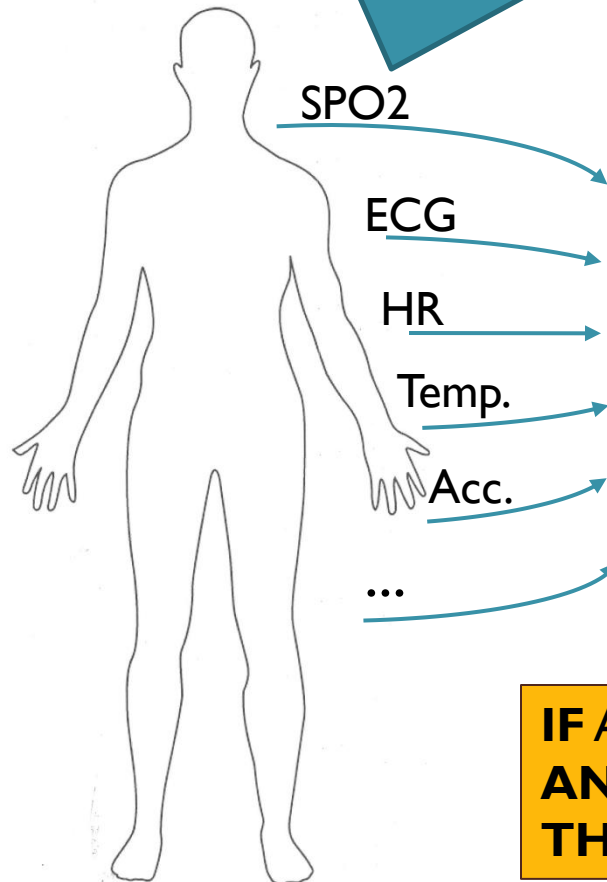


# Mobile Information Management

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)

# Discussion

- What are some applications in your country where data or information management can make a difference?
- <http://www.youtube.com/watch?v=bedVDouyMYE>



## Research Topic #1

# Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search

Mohan Yang (Shanghai Jiao Tong University),

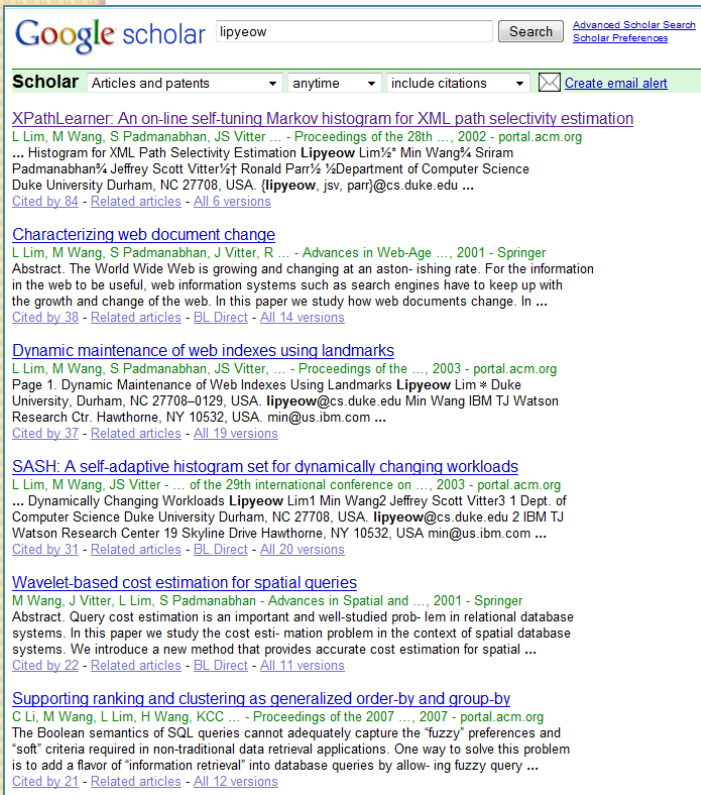
Haixun Wang (Microsoft Research Asia),

**Lipyeow Lim (UHM)**

Min Wang (HP Labs China)

# Motivating Application

- Management at a prominent research institute wanted to analyze the impact of the publications of its researchers ...



Google scholar lipyeow Search Advanced Scholar Search Scholar Preferences

Scholar Articles and patents anytime include citations Create email alert

[XPathLearner: An on-line self-tuning Markov histogram for XML path selectivity estimation](#)  
 L Lim, M Wang, S Padmanabhan, JS Vitter ... - Proceedings of the 28th ..., 2002 - portal.acm.org  
 ... Histogram for XML Path Selectivity Estimation **Lipyeow** Lim<sup>1\*</sup> Min Wang<sup>2</sup> Sriram Padmanabhan<sup>2</sup> Jeffrey Scott Vitter<sup>2†</sup> Ronald Parr<sup>2</sup> <sup>1</sup>Department of Computer Science Duke University Durham, NC 27708, USA. (**lipyeow**, jsv, parj)@cs.duke.edu ...  
 Cited by 84 - Related articles - All 6 versions

[Characterizing web document change](#)  
 L Lim, M Wang, S Padmanabhan, J Vitter, R ... - Advances in Web-Age ..., 2001 - Springer  
 Abstract. The World Wide Web is growing and changing at an astonishing rate. For the information in the web to be useful, web information systems such as search engines have to keep up with the growth and change of the web. In this paper we study how web documents change. In ...  
 Cited by 38 - Related articles - BL Direct - All 14 versions

[Dynamic maintenance of web indexes using landmarks](#)  
 L Lim, M Wang, S Padmanabhan, JS Vitter, ... - Proceedings of the ..., 2003 - portal.acm.org  
 Page 1. Dynamic Maintenance of Web Indexes Using Landmarks **Lipyeow** Lim \* Duke University, Durham, NC 27708-0129, USA. **lipyeow**@cs.duke.edu Min Wang IBM TJ Watson Research Ctr. Hawthorne, NY 10532, USA. min@us.ibm.com ...  
 Cited by 37 - Related articles - All 19 versions

[SASH: A self-adaptive histogram set for dynamically changing workloads](#)  
 L Lim, M Wang, JS Vitter - ... of the 29th international conference on ..., 2003 - portal.acm.org  
 ... Dynamically Changing Workloads **Lipyeow** Lim<sup>1</sup> Min Wang<sup>2</sup> Jeffrey Scott Vitter<sup>3</sup> 1 Dept. of Computer Science Duke University Durham, NC 27708, USA. **lipyeow**@cs.duke.edu 2 IBM TJ Watson Research Center 19 Skyline Drive Hawthorne, NY 10532, USA min@us.ibm.com ...  
 Cited by 31 - Related articles - BL Direct - All 20 versions

[Wavelet-based cost estimation for spatial queries](#)  
 M Wang, J Vitter, L Lim, S Padmanabhan - Advances in Spatial and ..., 2001 - Springer  
 Abstract. Query cost estimation is an important and well-studied problem in relational database systems. In this paper we study the cost estimation problem in the context of spatial database systems. We introduce a new method that provides accurate cost estimation for spatial ...  
 Cited by 22 - Related articles - BL Direct - All 11 versions

[Supporting ranking and clustering as generalized order-by and group-by](#)  
 C Li, M Wang, L Lim, H Wang, KCC ... - Proceedings of the 2007 ..., 2007 - portal.acm.org  
 The Boolean semantics of SQL queries cannot adequately capture the "fuzzy" preferences and "soft" criteria required in non-traditional data retrieval applications. One way to solve this problem is to add a flavor of "information retrieval" into database queries by allowing fuzzy query ...  
 Cited by 21 - Related articles - All 12 versions

Employee	Publication	Citation
Lipyeow	XPathLearner ...	84
Lipyeow	Characterizing...	38
Haixun	Clustering by ...	308
Haixun	Mining concept ...	424
...	...	...

# The Simple Solution

Loop

$Q$  = set of keyword queries

Foreach  $q$  in  $Q$

Send  $q$  to Google Scholar

Scrape the first few pages into tuples

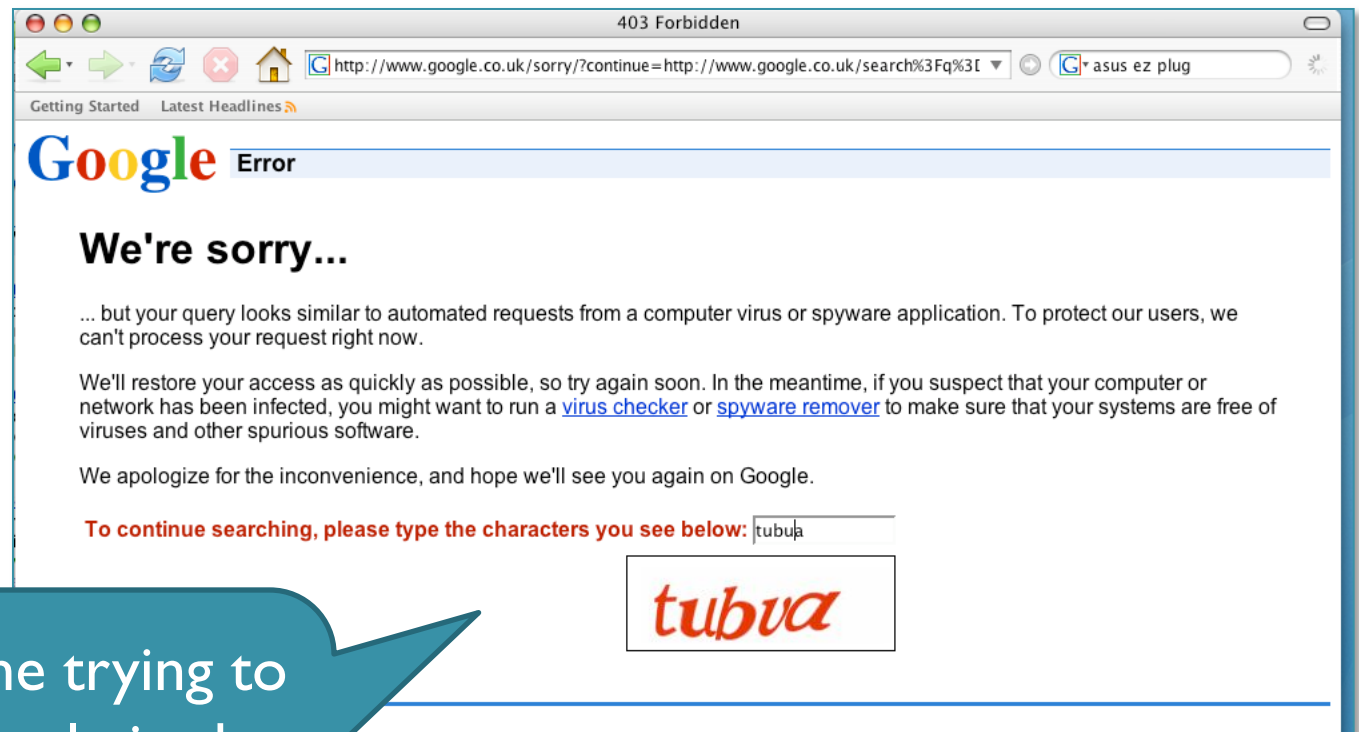
Update local relation using scraped tuples

Sleep for  $t$  seconds

End Loop

- Query Google Scholar using researcher's name and/or publication title to get
  - new publications and
  - updated citation counts

# Problem with the Simple Solution



Everyone trying to use Google in the building got this screen !

# The Elegant Solution

- All this hacking (including the solution I am about to present) could be avoided if there was an API to get structured relations from Google Scholar.



Employee	Publication	Citation
Lipyeow	XPathLearner ...	84
Lipyeow	Characterizing...	38
Haixun	Clustering by ...	308
Haixun	Mining concept ...	424
...	...	...

Linking Open Data effort might address this issue...

# But ...

- Such API's don't exist (yet?)
- And ...

I need those  
citation counts by  
next week!





# Problem Statement

- Local database periodically synchronizes its data subset with the data source
- Data source supports keyword query API only
- Extract relations from the top k results (ie first few result pages) to update local database

At each synchronization,  
find a set of queries that will maximize the  
“content freshness” of the local database.

- Only relevant keywords are used in the queries
- Keywords cover the local relation
- Number of queries should be minimized
- Result size should be minimized

NP-Hard by  
reduction to  
Set Cover

# Picking the Right Queries ...

Loop

**$Q$  = set of keyword queries**

Foreach  $q$  in  $Q$

Send  $q$  to Google Scholar

Scrape the first few pages into tuples

Update local relation using scraped tuples

Sleep for  $t$  seconds

End Loop

- The simple algorithm is fine, we just need to pick the right queries...
  - Not all tuples are equal – some don't get updated at all, some are updated all the time
  - Some updates are too small to be significant

# Greedy Probes Algorithm

1.  $Q$  = empty set of queries
2. NotCovered = set L of local tuples
3. While not **stopping condition** do
4.     K = Find all keywords associated with NotCovered
5.     Pick  $q$  from PowerSet(K) using **heuristic equation**
6.     Add  $q$  to  $Q$
7.     Remove tuples associated with  $q$  from NotCovered
8. End While

Could be based on size of  $Q$  or coverage of L

- What should greedy heuristic do ?
  - **Local coverage** : a good query will get results to update as much of the local relation as possible
  - **Server coverage** : a good query should retrieve as few results from the server as possible.
  - A good query updates the **most critical** portion of the local relation to maximize “**content freshness**”

# Content Freshness

Server	Employee	Publication	Citation
	Lipyeow	XPathLearner	87

Local	Employee	Publication	Citation
	Lipyeow	XPathLearner	84

- Weighted tuple dissimilarity
  - Some tuples are more important to update
  - $= w(local) * d(local, server)$
- Content Freshness

$$D(L, S) = \frac{1}{|L|} \sum_{l \in L} w(l) \cdot d(l, s)$$

- Example:
  - $w(l) = l.citation = 84$
  - $d(l, s) = |l.citation - s.citation| = 3$

**Catch:** local DB does not know the current value of citation on the server!

# Content Freshness (take 2)

- Estimate the server value of citation using an update model based on
  - Current local value of citation
  - Volatility of the particular citation field
  - Time elapsed since last sync.

$$D(L, S) = \frac{1}{|L|} \sum_{l \in L} w(l) \cdot F(l, t)$$

$F(l, t)$  estimates the dissimilarity between the local tuple and the server tuple at time  $t$  assuming an update model

# Greedy Heuristic

- Query efficiency

$$q = \arg \max_{q \in P(K)} \frac{|LocalCoverage(q)|}{|ServerCoverage(q)|}$$

- To give higher priority to “unfresh” tuples, we weight the local coverage with the freshness

$$q = \arg \max_{q \in P(K)} \frac{\sum_{l \in LocalCoverage(q)} w(l) \cdot F(l, t)}{|ServerCoverage(q)|}$$

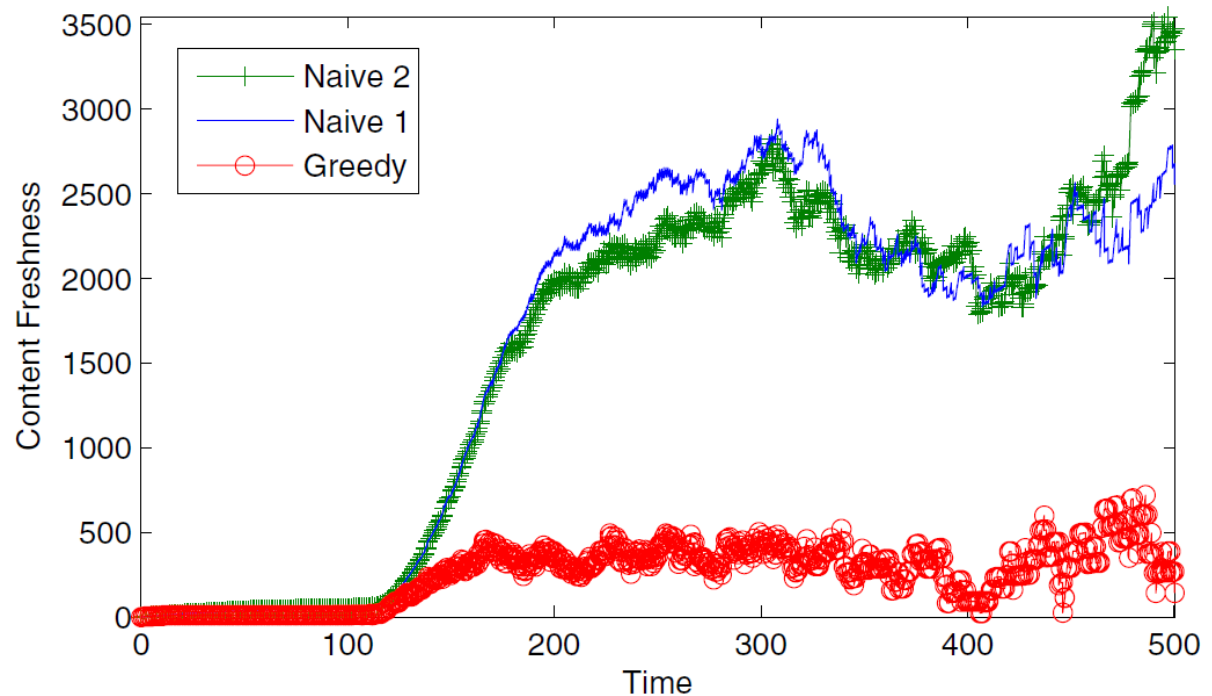
- Catch: local DB does not know server coverage!
  - Estimate server coverage using statistical methods
  - Estimate server coverage using another sample data source (eg. DBLP)

# Experiments

- Data sets:
  - Synthetic data
  - Paper citations (this presentation)
  - DVD online store
- Approximate Powerset(K) with all keyword pairs
- Result Extraction
  - Method 1: scan through all result pages
  - Method 2: scan only the first result page

# Content Freshness

- Synthetic citation data based on known statistics
- A Poisson-based update model used to estimate freshness
- 10 queries are sent at each sync
- Naive 1 & 2 sends simple ID-based queries





# Optimizations

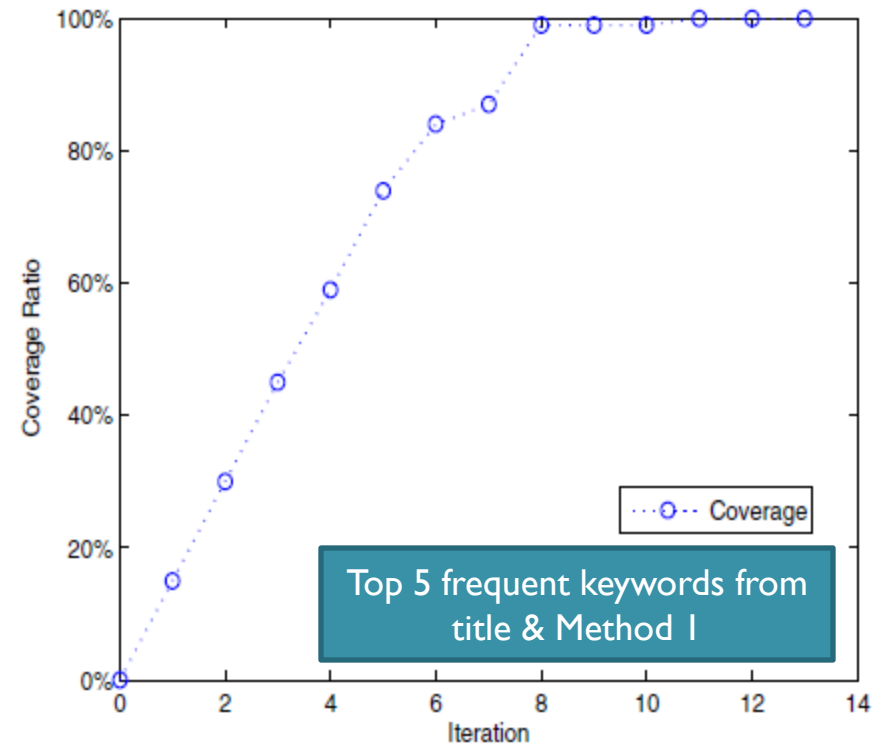
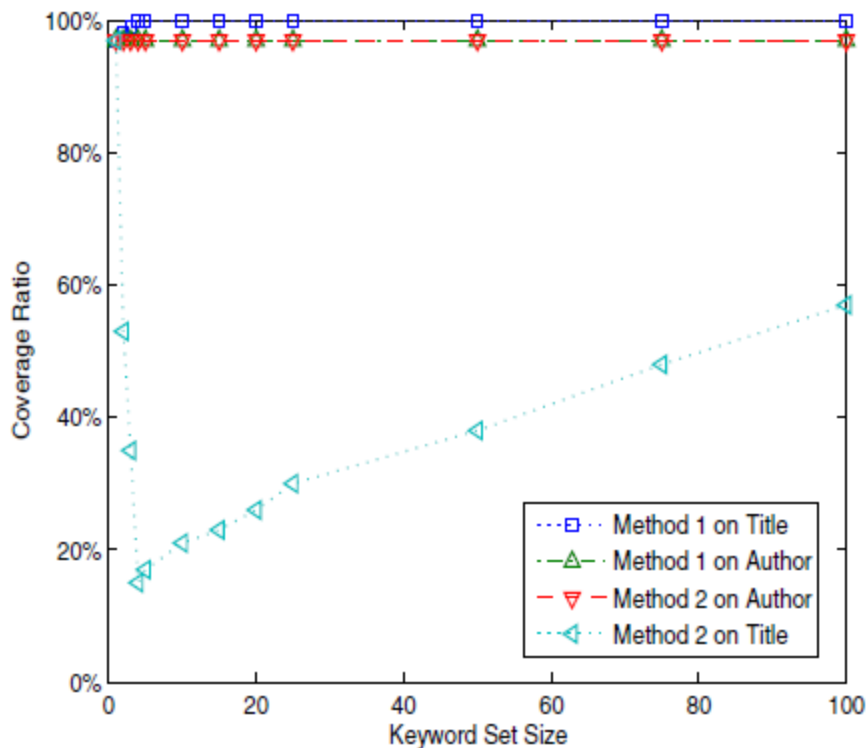
1.  $Q$  = empty set of queries
2. NotCovered = set L of local tuples
3. While not **stopping condition** do
4.      $K$  = Find all keywords associated with NotCovered
5.     Pick  $q$  from PowerSet( $K$ ) using **heuristic equation**
6.     Add  $q$  to  $Q$
7.     Remove tuples associated with  $q$  from NotCovered
8. End While

$K$  can be large

Power Set is exponential

- Approximate  $K$  using most frequent  $k$  keywords
- Approximate Power Set using subsets up to size  $m=2$  or  $3$ .

# Coverage Ratio



- Coverage ratio is the fraction of local tuples covered by a set of queries
- Result Extraction
  - Method 1: scan through all result pages
  - Method 2: scan only the first result page

# Summary

- Introduced the problem of maintaining a local relation extracted from a web source via keyword queries
- Problem is NP-Hard, so design a greedy heuristic-based algorithm
- Tried one heuristic, results show some potential
- Still room for more work – journal paper



## Research Topic #2

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

**Lipyeow Lim**

University of Hawai`i at Mānoa

**Archan Misra**

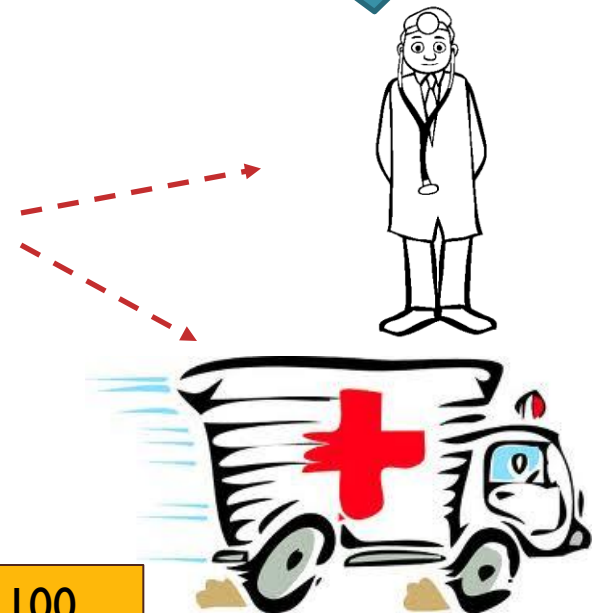
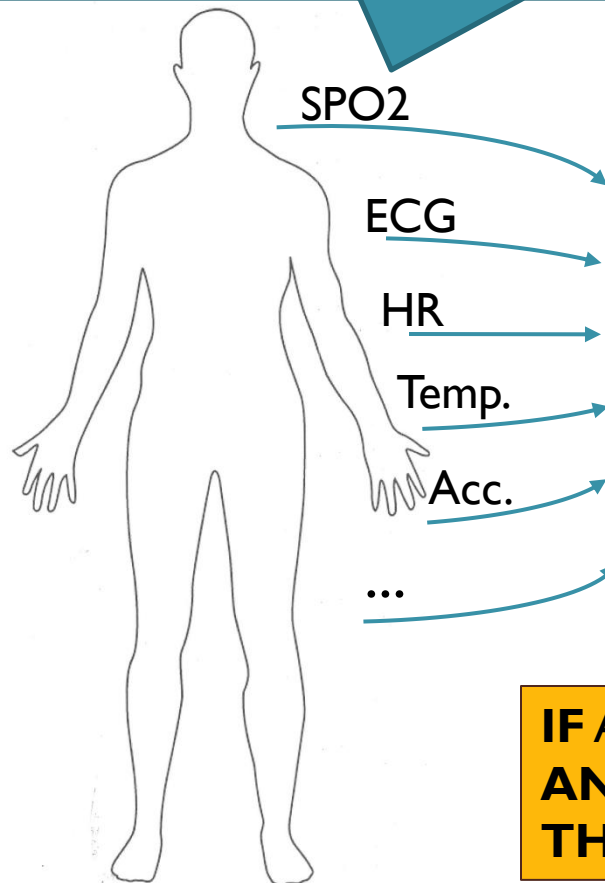
Singapore Management University

# 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

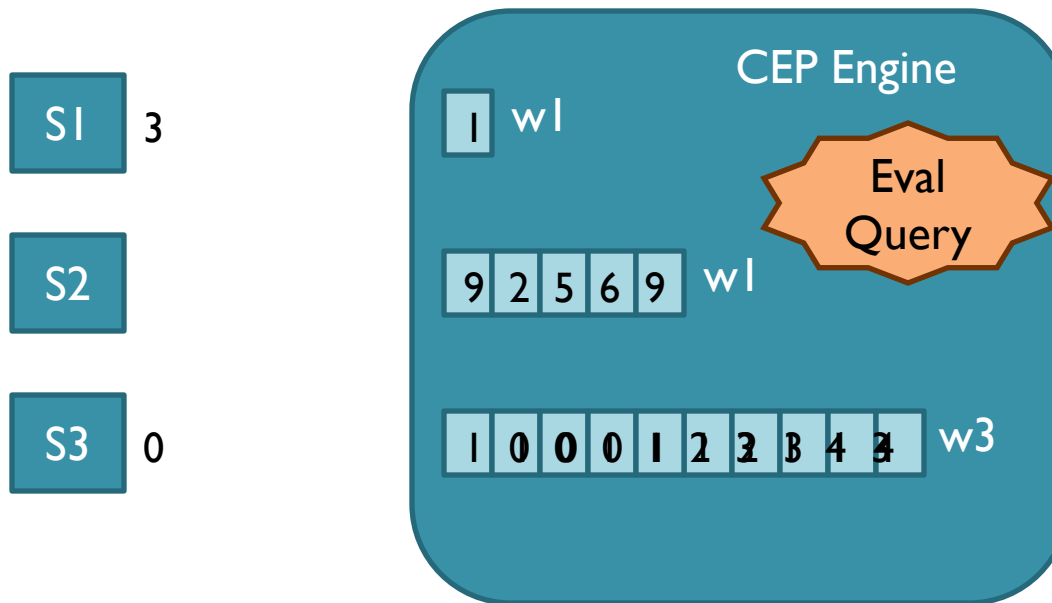
Alerts can  
notify  
emergency  
services or  
caregiver



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

# Continuous/Streaming Evaluation

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

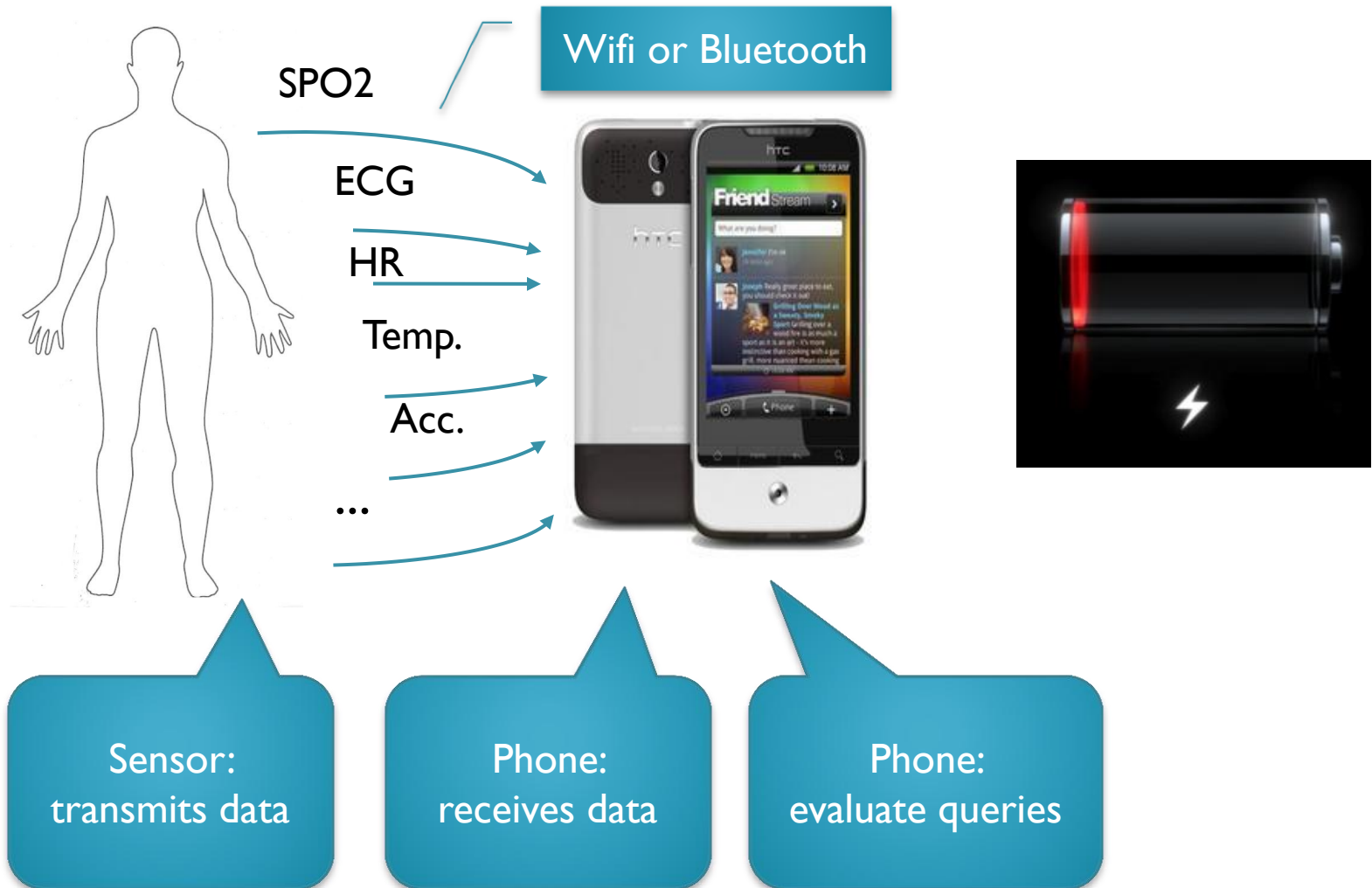


## Algorithm

When  $t_i$  of  $S_i$  arrives  
Enqueue  $t_i$  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.

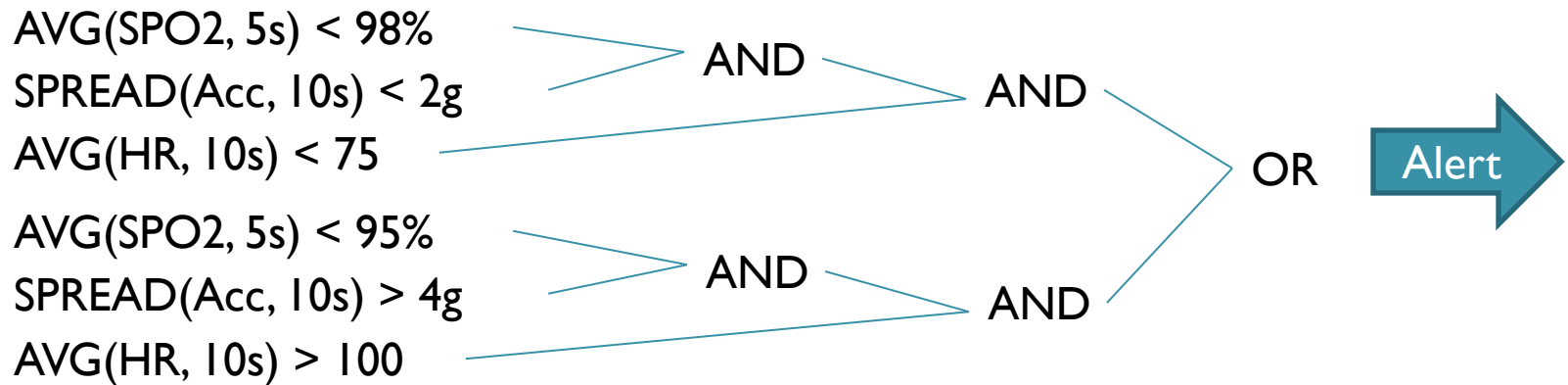


# Key Ideas

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

*Assuming fairly smart sensors capable of buffering and supporting “pull”*

# Query Model



- A **query** is a boolean combination of predicates
- Predicates
  - **Aggregation functions** over a **time-based window** of sensor data

# Sensor Data Acquisition

3D acc.  
ECG,  
EMG, GSR



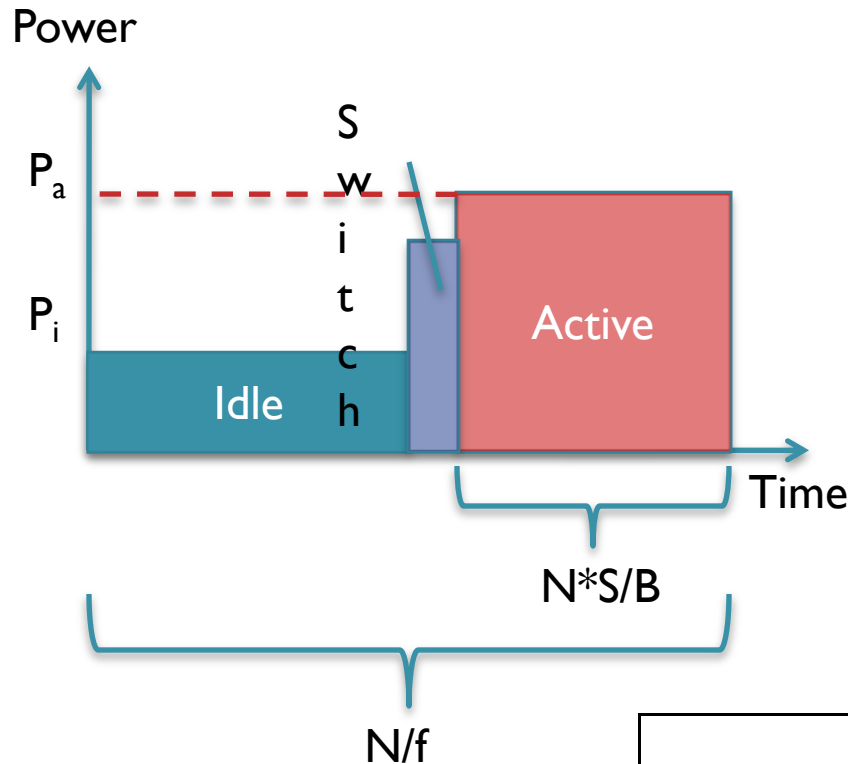
Bluetooth  
Or 802.11  
Or 802.15



- 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
Accelerometer	64	3	100 Hz
Temperature	20	1	256 Hz

# Pulling N Tuples from Sensor

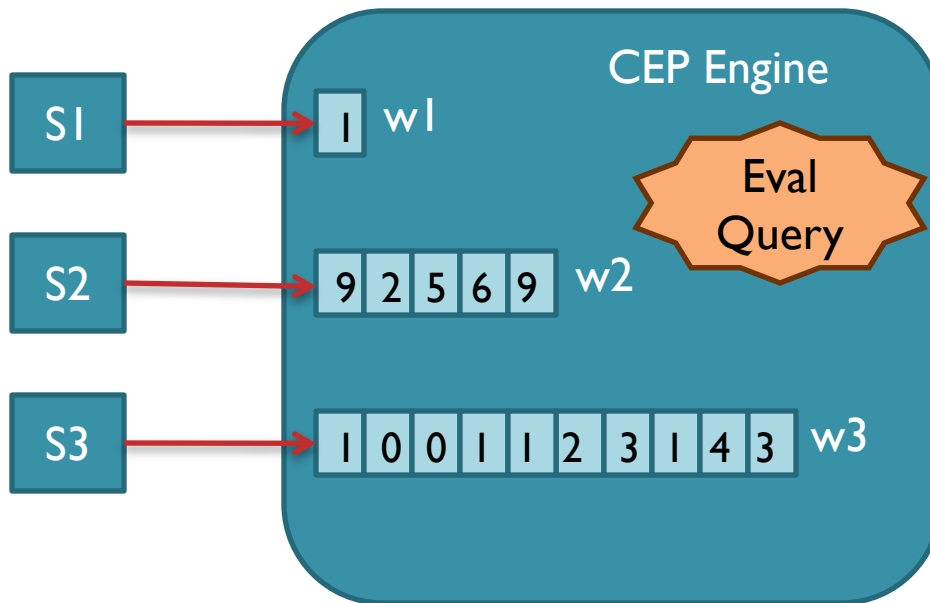


- Idle mode consumes  $P_i$  mW
- Active mode consumes  $P_a$  mW
- Sensor rate is  $f$  Hz
- A tuple is  $S$  bits
- Bandwidth is  $B$  Mbps

	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 $\mu$ Joule	–
$T_{idle}$	100 ms	–
$T_{switch}$	–	6 msec

# Pull-based Evaluation

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

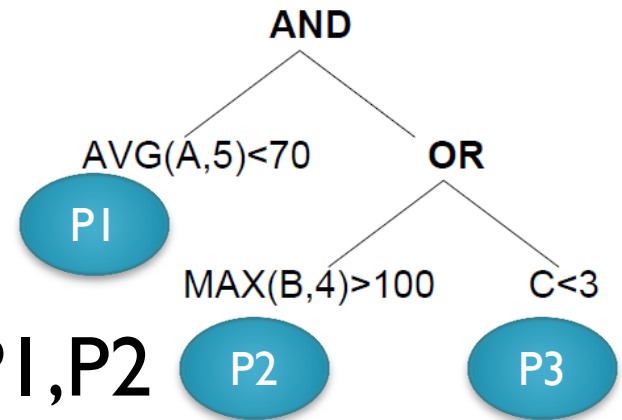


## Pull

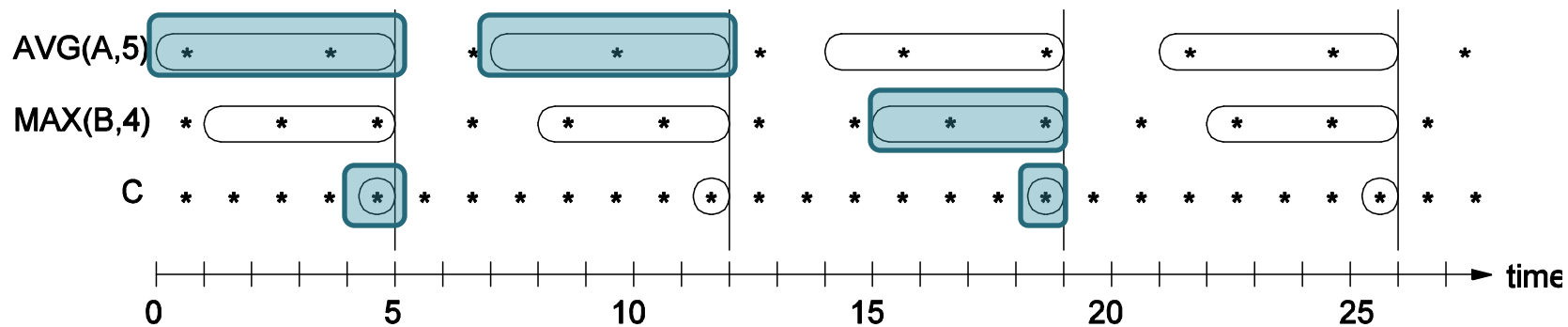
```
Loop every  $\omega$  seconds
For each sensor  $S_i$ 
  Acquire data for  $S_i$ 
  Enqueue data into  $W_i$ 
EndFor
If  $Q$  is true,
  Then output alert
End loop
```

- Complex interaction between  $\omega$ , stream rates, and predicate windows
- If predicate  $S1 < 10$  is **false**, why bother to acquire data for S2 and S3?

# Example: $\omega=7$



- **Time 5:** eval order is P3,P1,P2
- **Time 12:** eval order is P1,P2,P3
- **Time 19:** eval order is P2,P3,P1



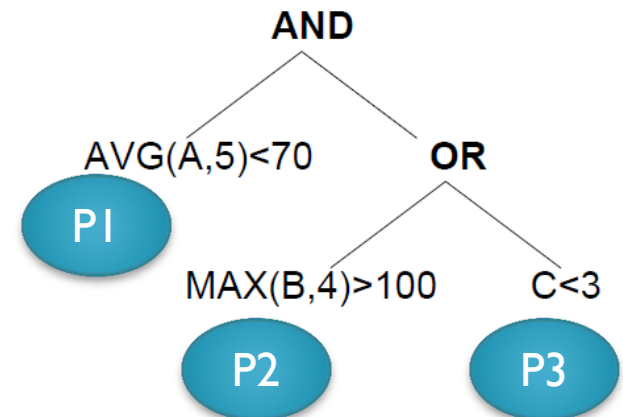
# Evaluation Order

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

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

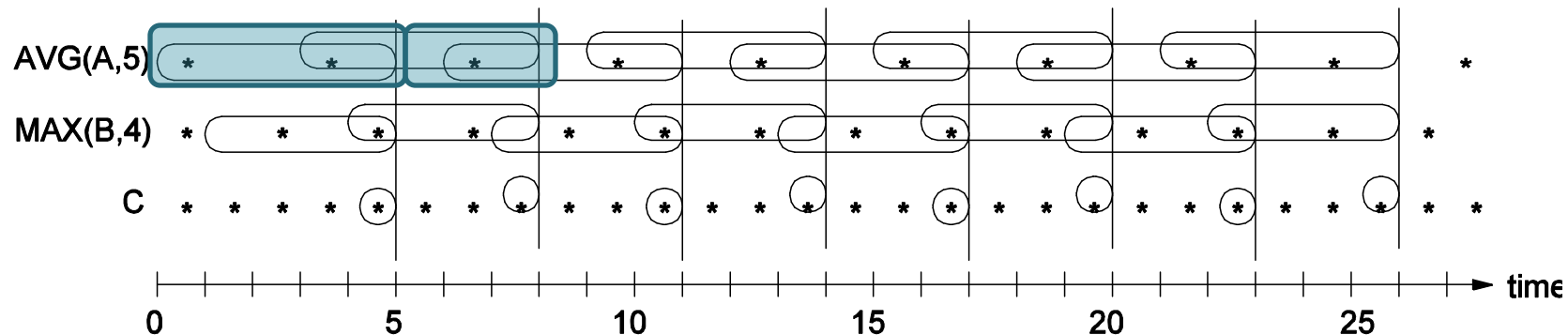
- Evaluate predicates with lowest energy consumption first
- Evaluate predicates with highest false probability first
- Evaluate predicate with lowest normalized acquisition cost first.

# Example: $\omega=3$



- **Time 5:** P1,P2,P3
- **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 Sketch

At each  $\omega$

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

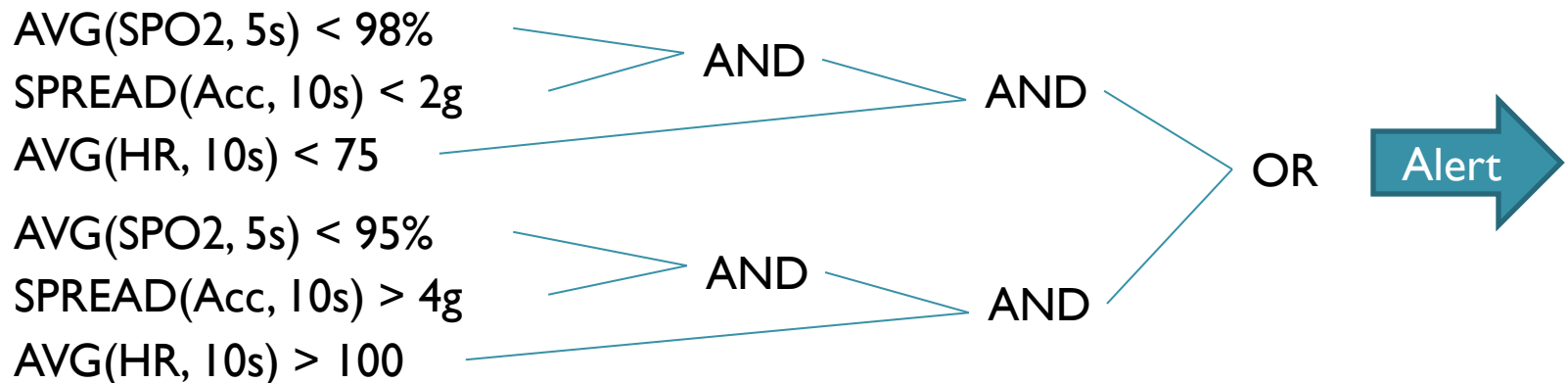
What happens if  $>2$  predicates operate on the same sensor data stream?

# 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  $\omega$  time period.
- Simulation results averages 5 1-hour traces with 95% confidence intervals.
- **P(pred=true)** distributions obtained from half the data streams themselves

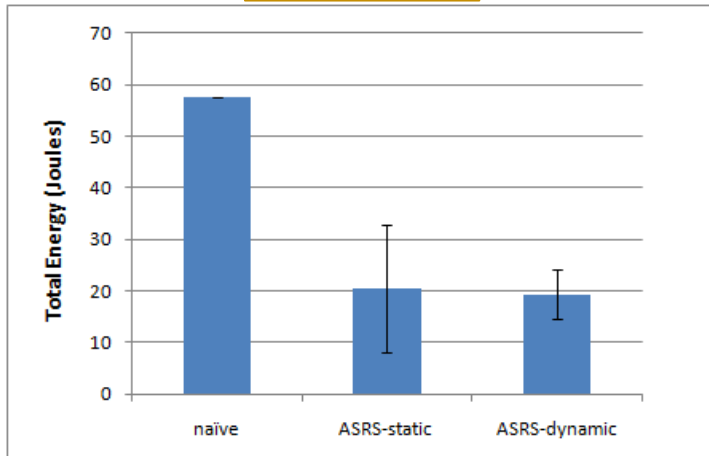
# Simulation Data & Query

- 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

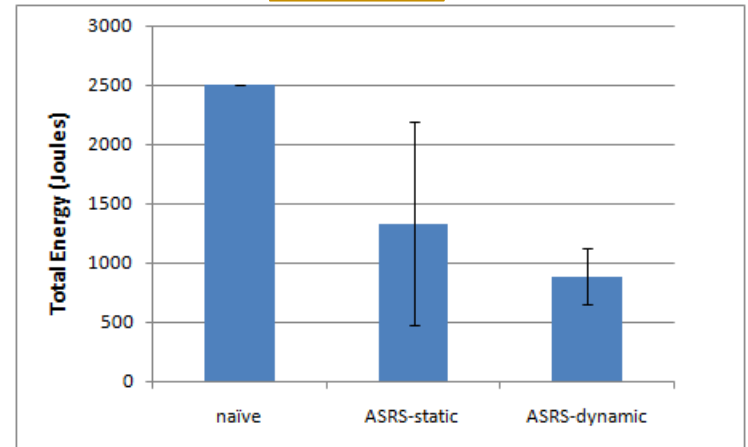


# Simulation Results

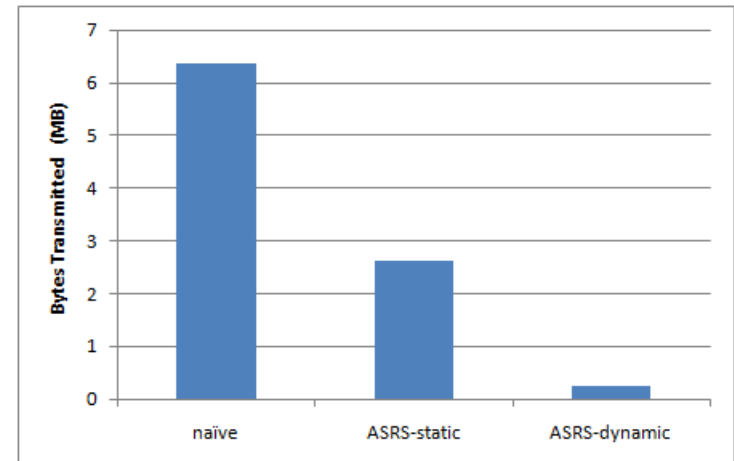
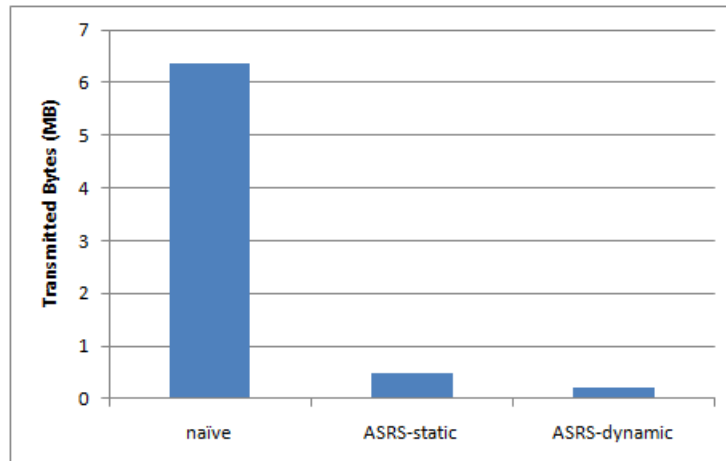
Bluetooth



802.11



Energy



Bytes

# Summary

- Pull-based processing paradigm can have a significant impact on data acquisition energy consumption
- Ordered evaluation of predicates can help shortcircuit the evaluation and avoid costly data acquisition
- We proposed evaluation algorithms based on these two observations to minimize data acquisition cost at CEP engine
- Results on synthetic traces show that savings up to 70% are possible.

# Real-time Data Analytics for Wind Power Management

- Key problem in renewable energy is the variability in supply
- Demand is predictable
- Accurate and continuous forecasting can help utility company balance the load
- Weather forecasting algorithms in streaming mode ?



# Scientific Data Warehouses

- Massive amount of data (petabyte range)
- No updates, append only
- Interactive queries + long running analytical queries
- Commodity clusters and/or virtualized “cloud” environment
- Data-intensive vs Compute-intensive infra-structures ?

