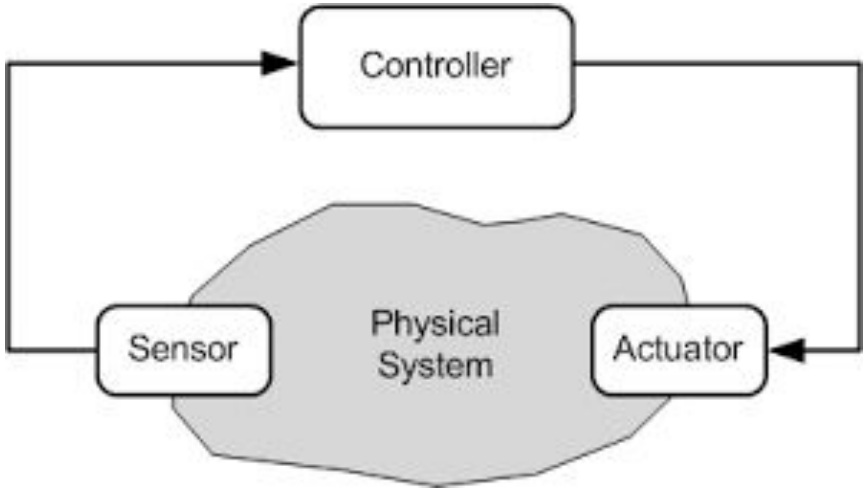
Decision-driven Execution: A Distributed Resource Management Paradigm for the Age of IoT

Tarek Abdelzaher, Tanvir Al Amin, Amotz Bar-Noy, William Dron, Ramesh Govindan, Reginald Hobbs, Shaohan Hu, Jung-Eun Kim, Shuochao Yao and Yiran Zhao

Presented by: Henry Jaensch

Introduction: IoT Feedback Loop

- Event Based:
  - Sensor receives a reading and sends it to the controller
- Periodic:
  - Sensor collectes a reading at a defined interval



Decision-Defined Execution

- What if we managed system resources based on the decisions that applications tasks needed to make?
- This kind of system needs to be able to decide what data to collect and when to collect it.

What is a Decision?

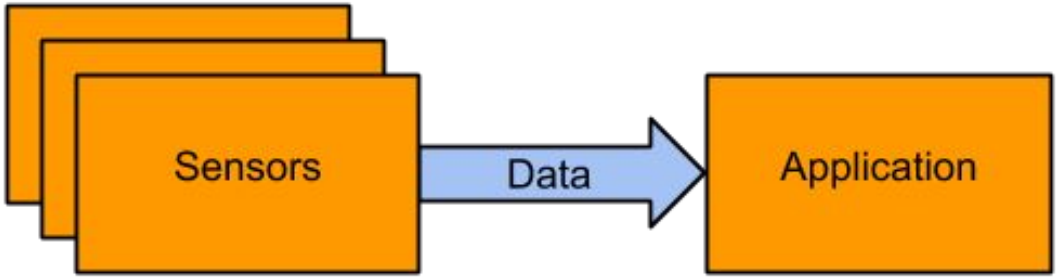
- The paper defines a decision “choice of a course of action among multiple alternatives”
- Decision deadline: temporal span with which you can act

What Kind of Data is Needed for Decision Making?

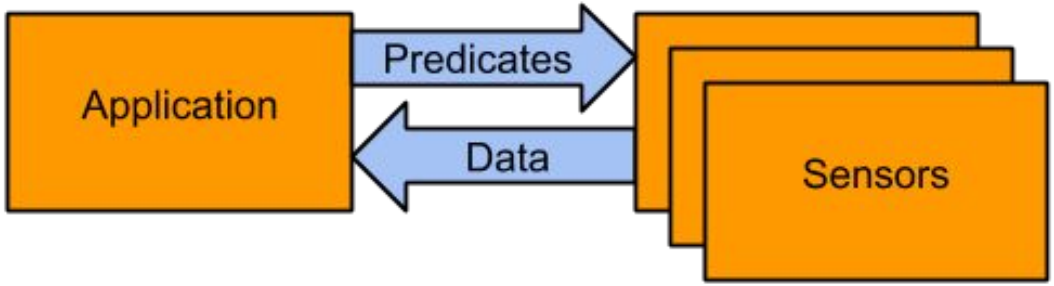
1. Has the quality threshold for decision making been met?
2. Is the data still relevant?
3. Can the decision be made within a certain deadline?

Problem Definition

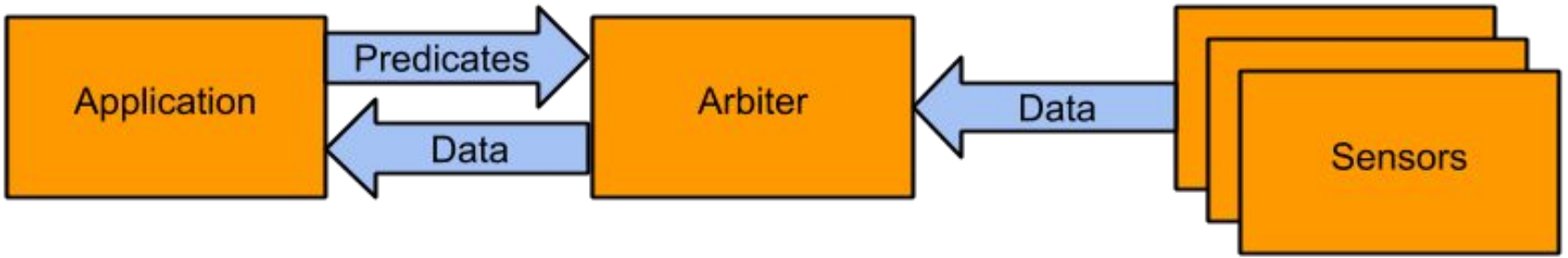
How do we go from this...



to this?



Problem Definition

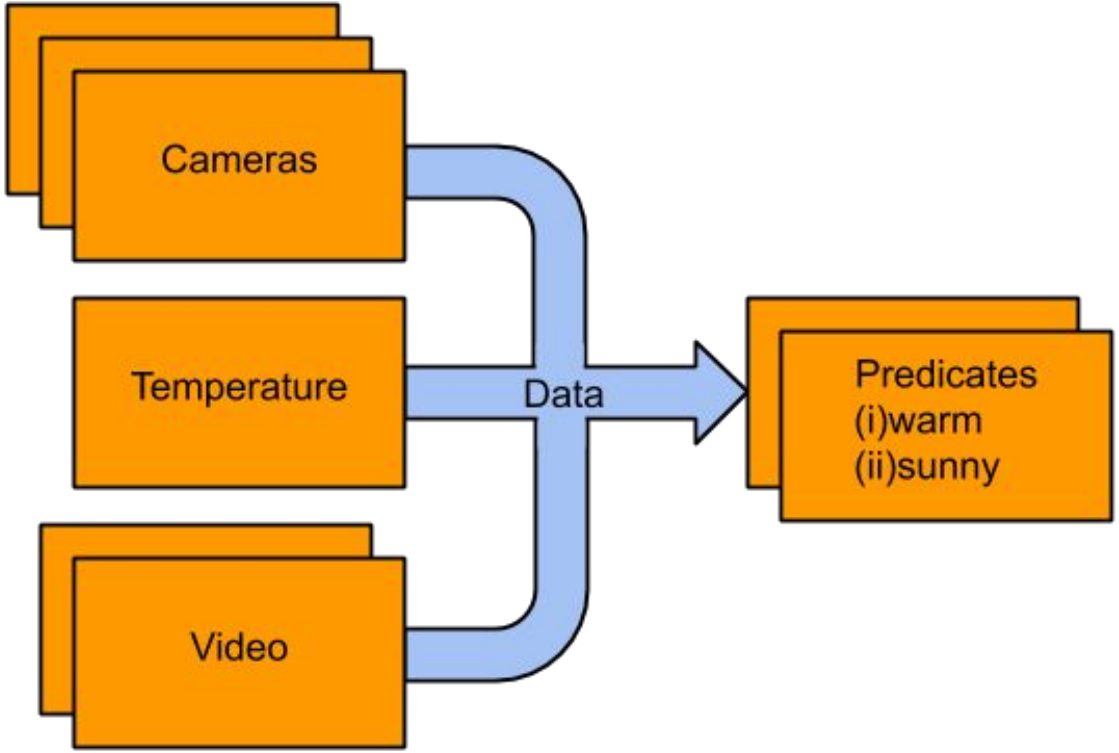


System Design: Components

- Predicates:
  - Requirements for a specific choice to be made
- Evidence(Data Objects):
  - Used to evaluate the truth value of predicates

Example:

"If it is sunny and warm, I will not wear a sweater."



Decisions Can be Translated to Predicates

- This breakdown is exciting because logic acts in ways we know how to manipulate. We can leverage this structure to optimize resource management.
- Defining application tasks in boolean terms can allow for optimizations

Predicate Logic Optimizations: Example

- Earthquake rocks the city, and medical air support is unavailable
- Ambulance can take routes A-B-C or D-E-F to reach an injured person
- $(viable(A) \wedge viable(B) \wedge viable(C)) \vee (viable(D) \wedge viable(E) \wedge viable(F))$
  - Where viable(x) means route segment x is viable

- Potential Optimizations:
  - Only one route needs to be viable
  - Only one segment needs to be not viable to disqualify a route
  - We can check minimal sensor data to fulfill these predicates.

Retrieval Cost Optimization: Short Circuiting

- More than eliminating unnecessary data collection we can use heuristics to optimize data collection.
- If the logic for a decision is connected by the logical and the predicate that is most likely to be false should be checked first. If in fact this predicate is determined to be false the remaining predicates do not need to be checked.

Short Circuiting Example

- In this example the predicates are joined by a logical and(∧)
- Short-Circuit Probability:
  - C: retrieval cost
  - p: Success probability
- Example: **h**∧**k**
  - **h**: C = 4mb, p = 60%
  - **k**: C = 5mb, p = 20%
  - **k**-prob: 16% > **h**-prob: 10%

$$a_i = b_{i_0} \wedge b_{i_1} \wedge b_{i_2} \wedge \ldots,$$

$$\frac{1 - p_{i_j}}{C_{i_j}}.$$

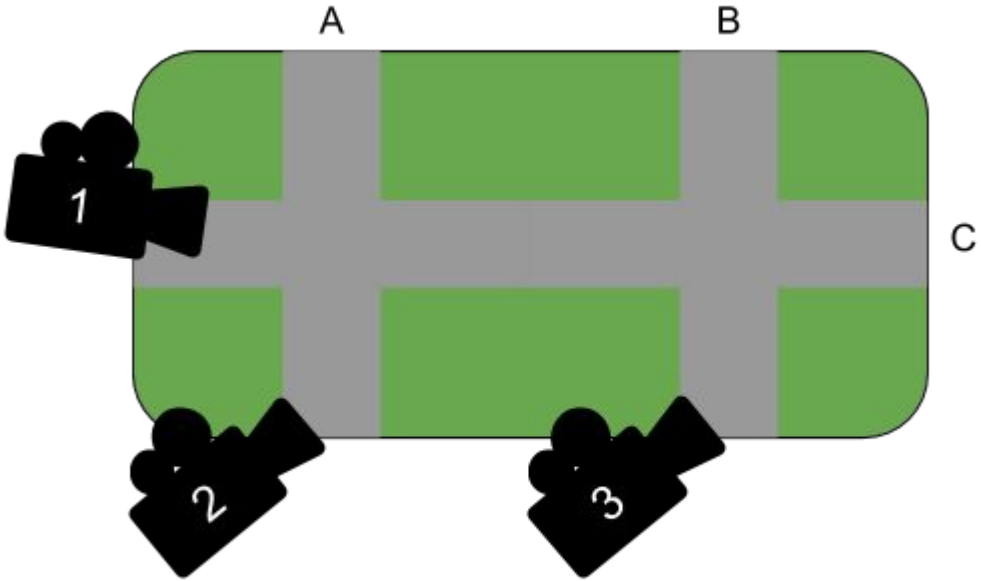
$$\underbrace{\frac{1 - 0.2}{5}}_{0.16} > \underbrace{\frac{1 - 0.6}{4}}_{0.1}.$$

k h

Retrieval Cost Optimization: Coverage

- There is a cross section of predicates that multiple decisions rely on
- More than that sensors can help evaluate a subset of predicates
  - Example: One camera can verify the viability of multiple route segments

Predicate: A ∧ B ∧ C



Decision-Driven Scheduling

- Is the data still fresh(reliable)?
- Can we meet the decision deadline?

$$\textbf{Data freshness: } t_i + I_i \geq F \ \ (\forall i, 1 \leq i \leq N),$$
$$\textbf{Decision deadline: } t + D \geq F,$$

$$\min\left(\min_{1 \leq i \leq N}(t_i + I_i),\ \ t + D\right) \geq F$$

Remaining Challenges: Scheduling

- **Non-Independent Queries:** Queries that rely on similar or the same predicates should not be processed in isolation
- **Noisy Sensor Data:** How to accomodate for execution wasted on a sensor that cannot provide a confident predicate? More simply, how do we schedule something that might fail?
- **Event Triggered Decision-Making:** some tasks work best under the event driven model this can be accounted for when using decision-based scheduling

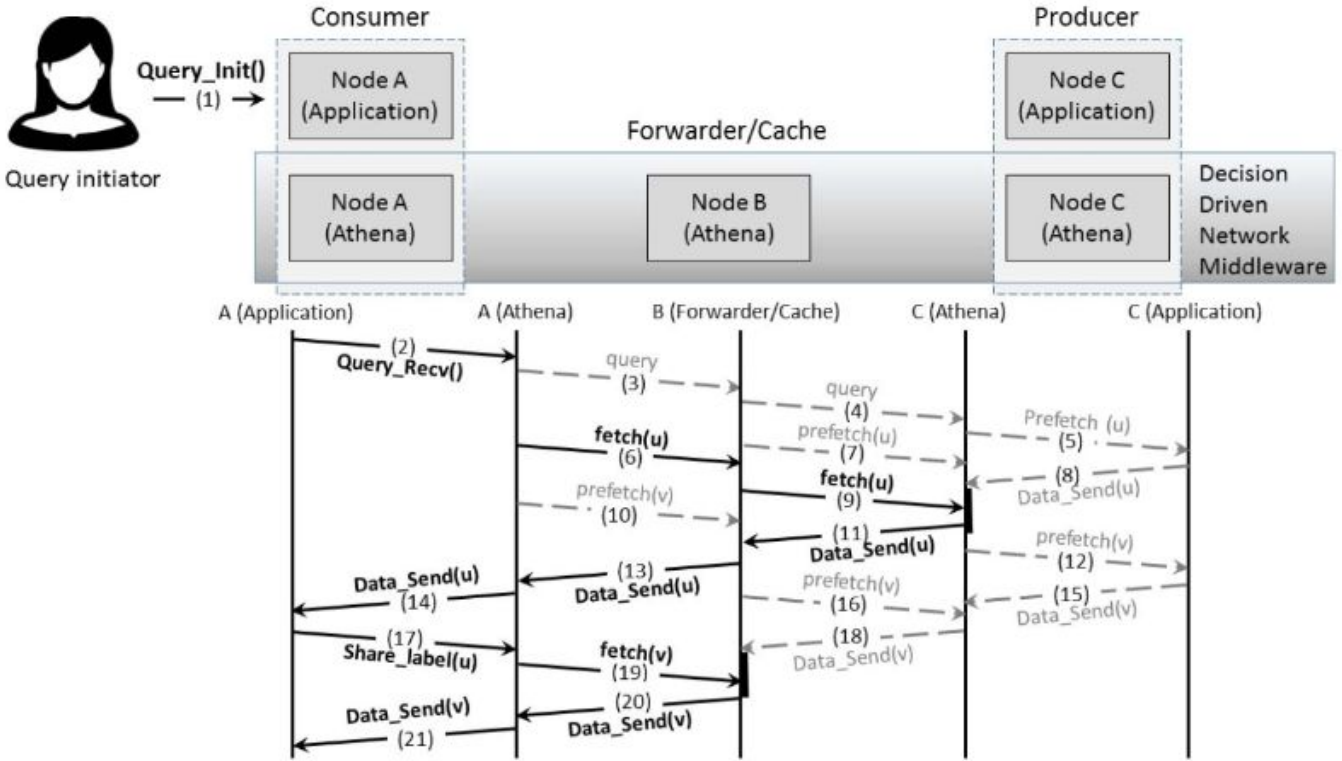
Networking Challenges: Namespacing

- Hierarchical Semantic Naming and Indexing
  - Information centric networks use data names rather than IP addresses
  - Namespacing in a network like this is UNIX-like:
  - `/city/marketplace/south/noon/camera1/`
- Advantages:
  - Automatic substitution when approximation is allowed
  - Graceful degradation with overload
- Criticality:
  - Namespacing can allow a network to easily support mixed criticality
  - Different levels of this namespace can be assigned criticality

Network Challenges: Publish-Subscribe

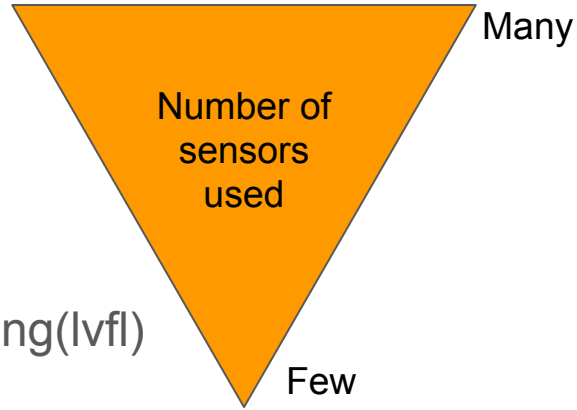
- Data is sub-additive:
  - An image of a bridge being flooded is only useful once. Ten of the same image are wasted resources.
- If the network could triage data upon receiving it could reduce overload
  - A naive implementation of this filter catches redundant data and doesn’t send it upstream
- Two cameras watching a street for pedestrians. Only one person is needed to fulfill “People in street” predicate

Athena System Design: Query Request



Evaluation

- Using the earthquake emergency response example the Athena implementation of this paradigm was tested under a variety of information retrieval strategies
  - Comprehensive retrieval (cmp)
  - Selected sources (slt)
  - Lowest Cost Source First (lcf)
  - Variational Longest Validity First (lvf)
  - Variational Longest Validity First with Label Sharing(lvfl)



Evaluation: Fast Changing Object Ratio

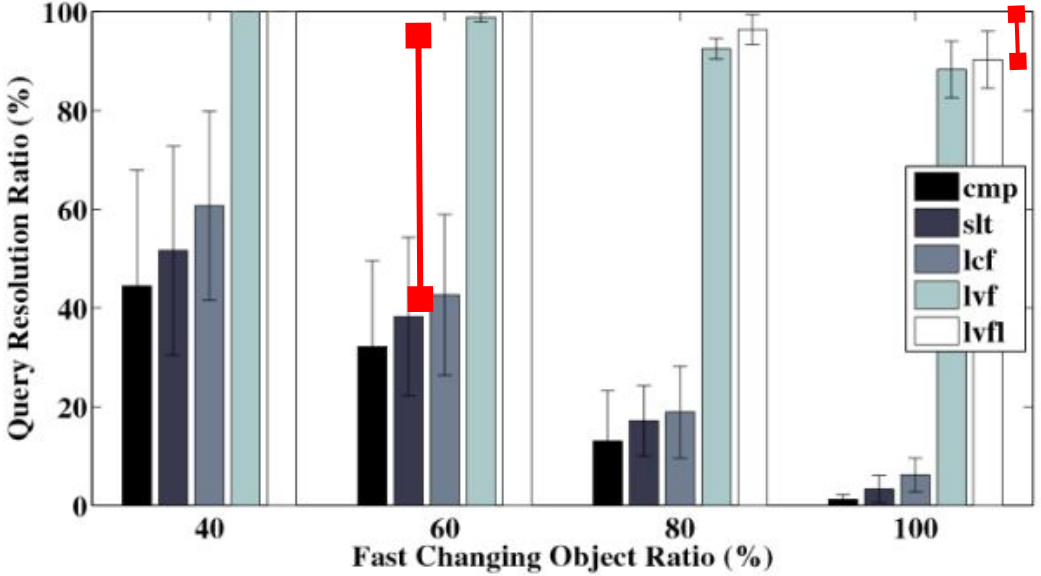


Fig. 2. Query resolution ratio at varying levels of environment dynamics (ratio of fast changing objects).

Evaluation: Bandwidth Consumption

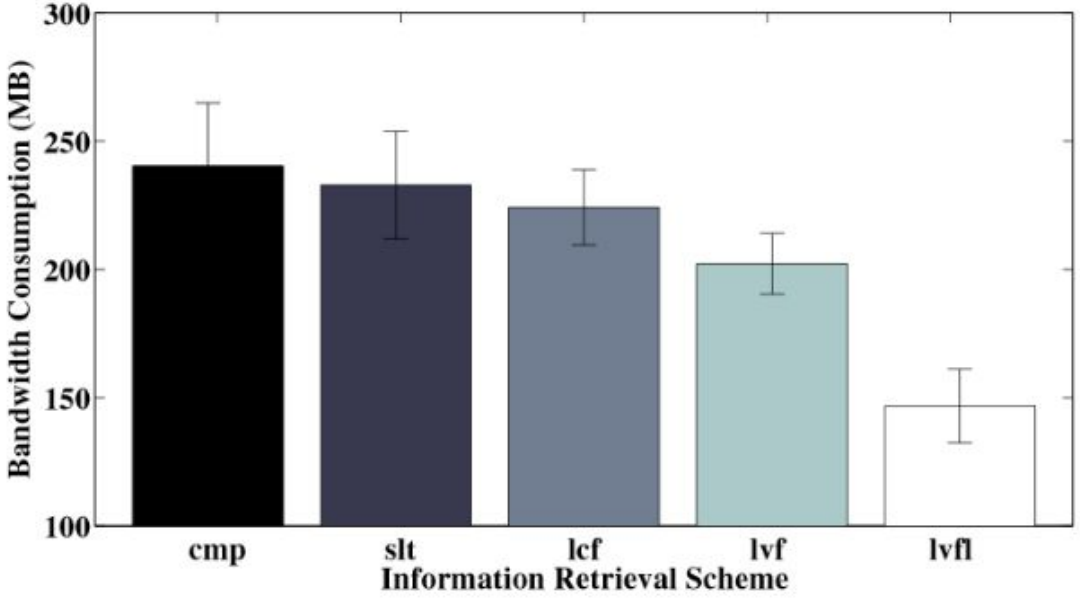


Fig. 3. Total network bandwidth consumption comparison of all schedule schemes (with 40% fast changing objects).

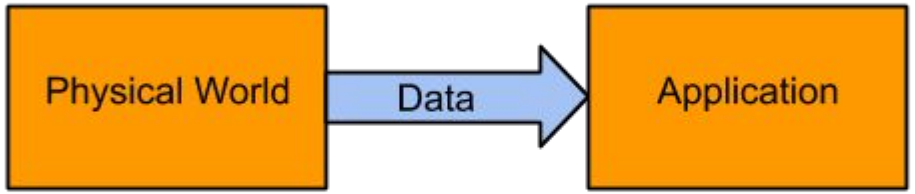
Critiques

- This is an interesting way to change our thinking about resource management but it is clear that there's a lot to explore. This paper may have gotten ahead of itself by trying to discuss all the possibilities rather than the implementation.
- **@pcodes:** Given how many components there are to this paper, it would have been nice to see a more comprehensive example that demonstrated all functionality.
- **@samfrey:** Even though the paper focuses on a higher-level change execution flow, the data offers proof of concept and got me excited about the idea.
- **@Bushidocodes:** Some decisions require historical context and baseline information to discover trends? How might this architecture support this sorts of queries?

Conclusions

This new paradigm is interesting and worth exploring. Understanding what different components of the system need to make decisions and optimizing with that information is useful.

Old:



New: