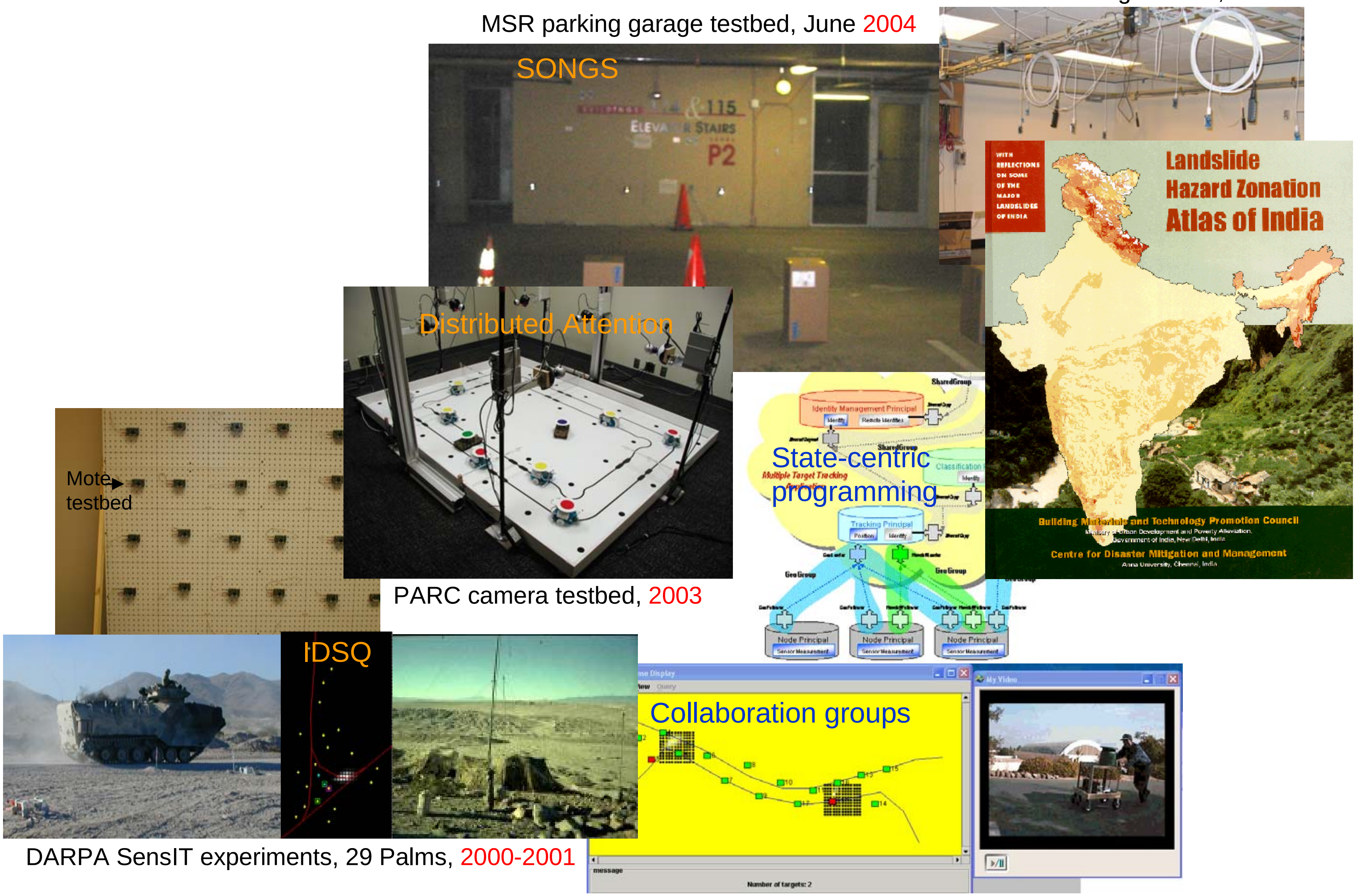
Challenges in Programming Sensor Nets

Feng Zhao
Networked Embedded Computing Group
Microsoft Research

Acknowledgment

- Joint work with Jie Liu, Alec Woo, Elaine Cheong, Kamin Whitehouse, Siddharth Seth, Prabal Dutta
- Discussions with Jeremy Elson and Leo Guibas

From Mojave Desert to India Hills …



MSR in-building testbed, 2005

MSR parking garage testbed, June 2004

PARC camera testbed, 2003

DARPA SensIT experiments, 29 Palms, 2000-2001

Outdoor experiment, 2002

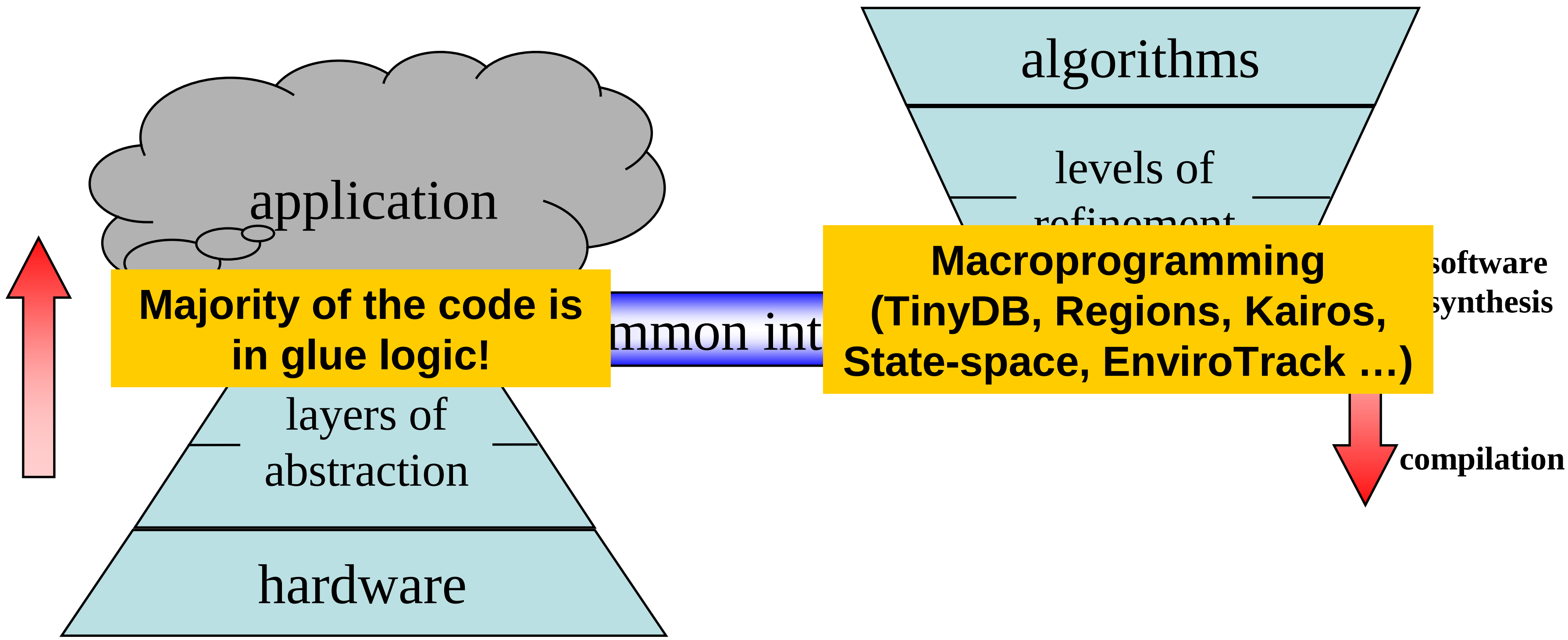
Lessons Learned

OS/Network-centric view

Designing component-level abstractions

Information-centric view

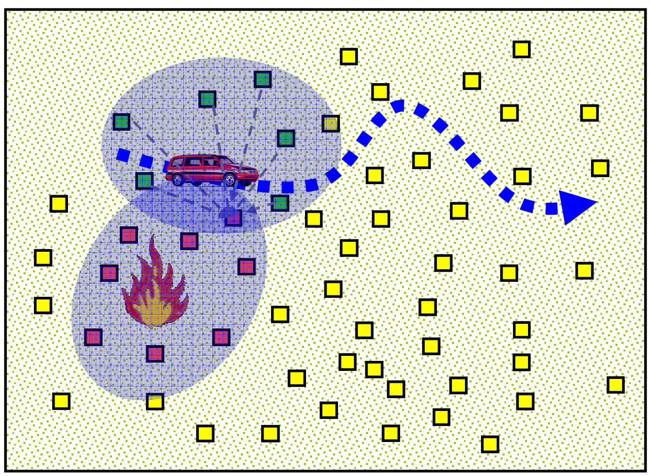
Designing application-level abstractions



Programming Challenges

You've heard all of this:

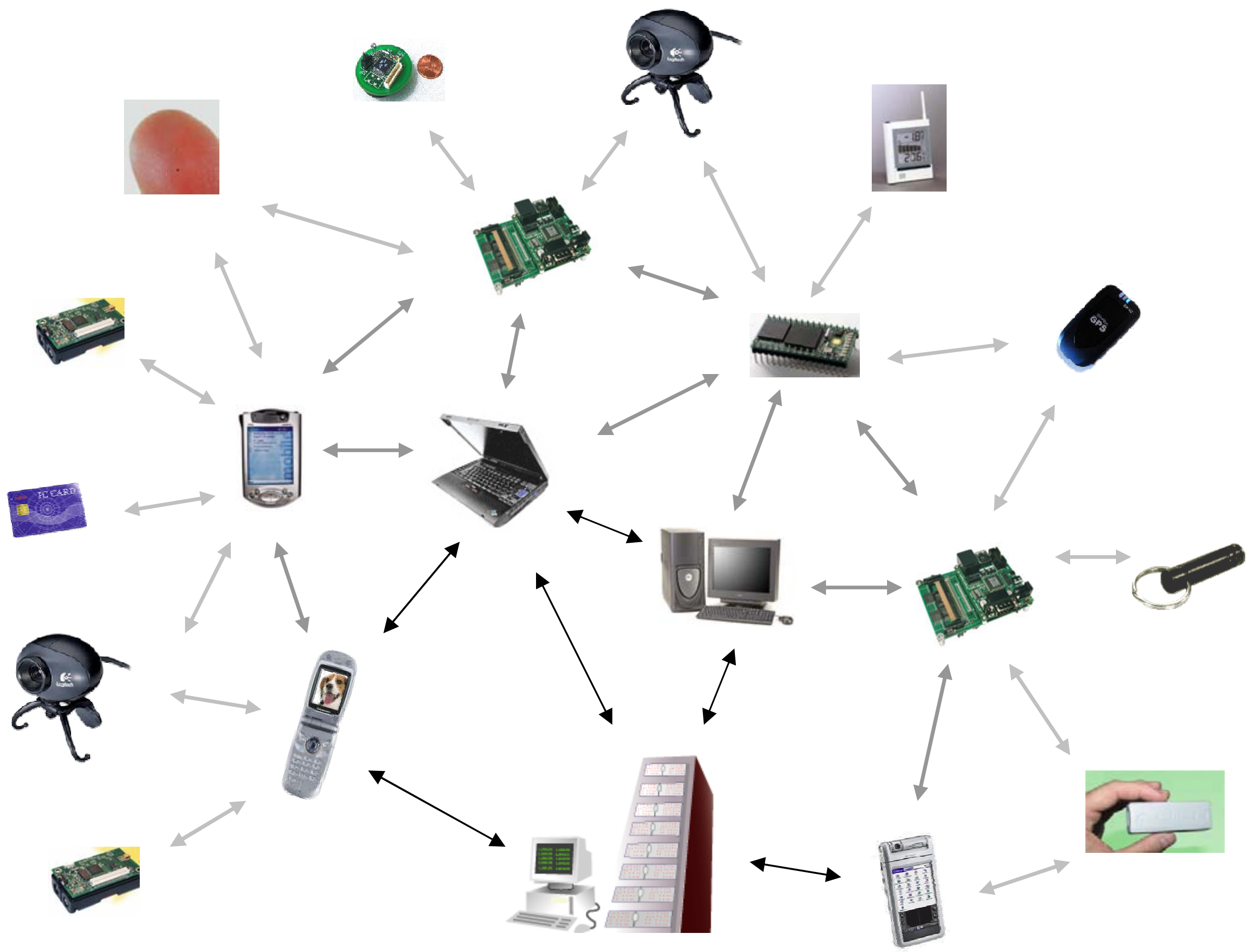
- Limited node capability
- Limited, uncertain connectivity
- Complex collaborative processing tasks



More importantly:

- Sensors are not just a random collection of objects
- Rich structure in system (e.g., connectivity, link quality, node density)
- Rich structure in sensing (e.g., spatio-temporal coherence, data reliability)
- Programming models should provide abstractions for these structures
- Permit couplings between system and information processing design

Proliferation of networked devices



Characteristics:

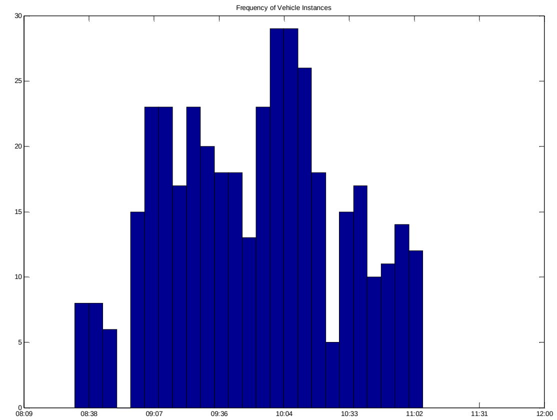
- Heterogeneous devices
- Disparate capabilities
- Physically embedded (energy, size, noise, real-time events, …)
- Dynamic topology
- Large scale
- Inherent uncertainties (in systems and environment)
- Concurrent user queries

Desired properties:

- Easy to program, deploy, and manage
- Robust to failure
- Responsive
- Re-taskable
- Scalable
- Secure

A Scenario: What’s happening in the garage?

Arrival statistics

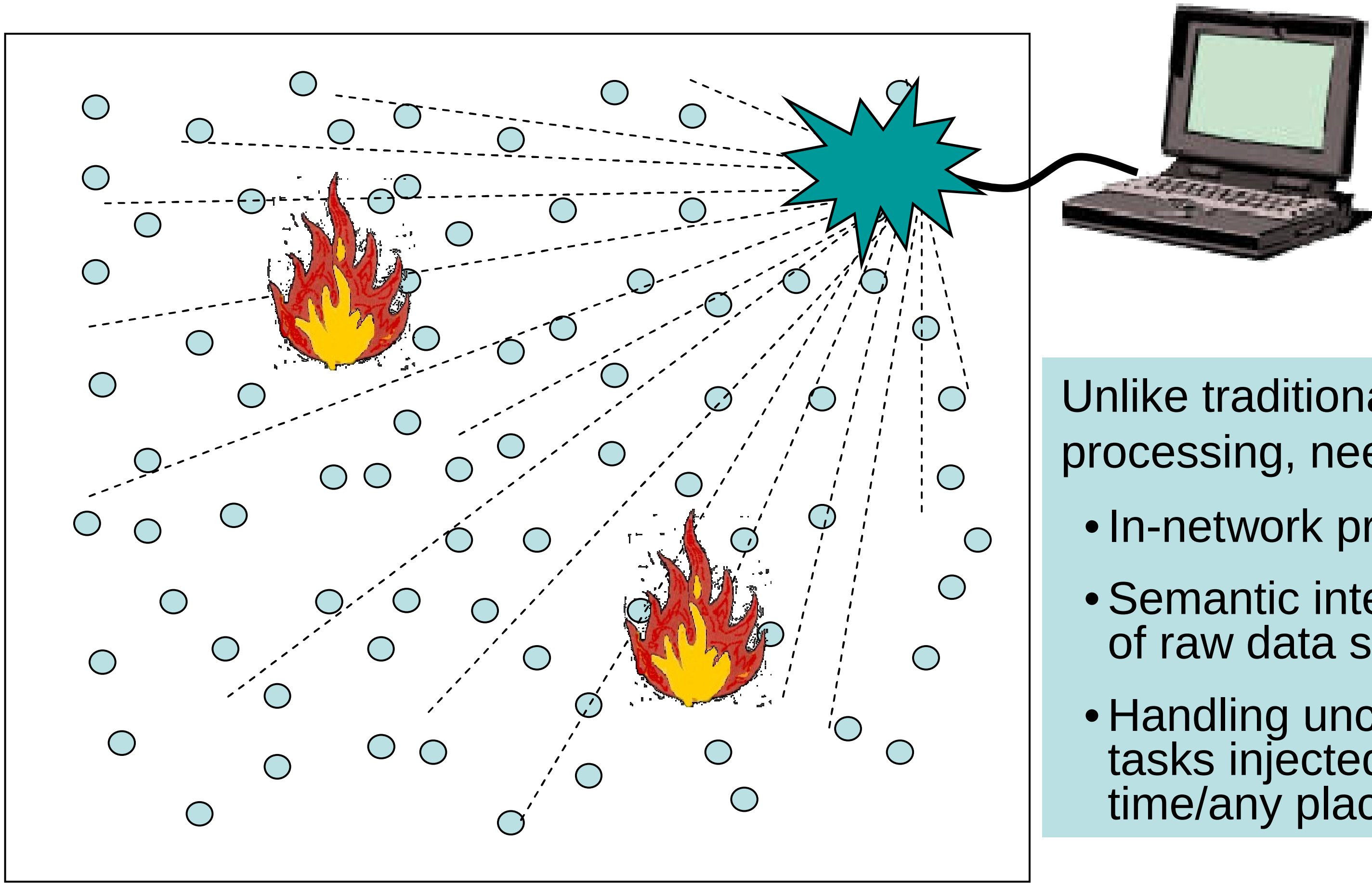


8am 11am

Multiple, concurrent queries:

- Traffic engineering: where can one find a parking spot?
- Security: what is going on down there?
- Corporate health services: when should the air exhaust fan run?

One may argue for central data collection and server programming

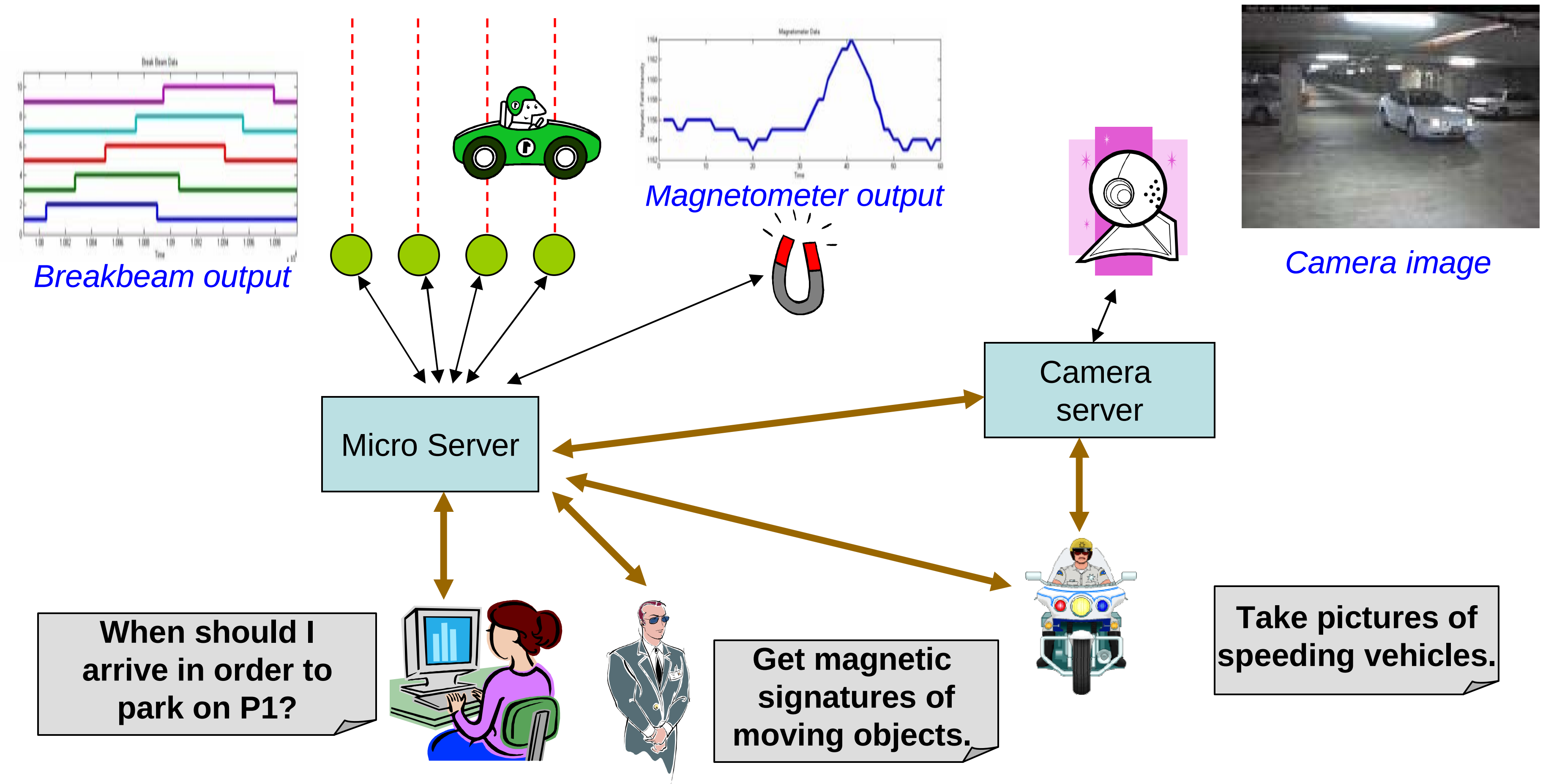


Unlike traditional query processing, need:

- In-network processing
- Semantic interpretation of raw data streams
- Handling uncoordinated tasks injected any time/any place

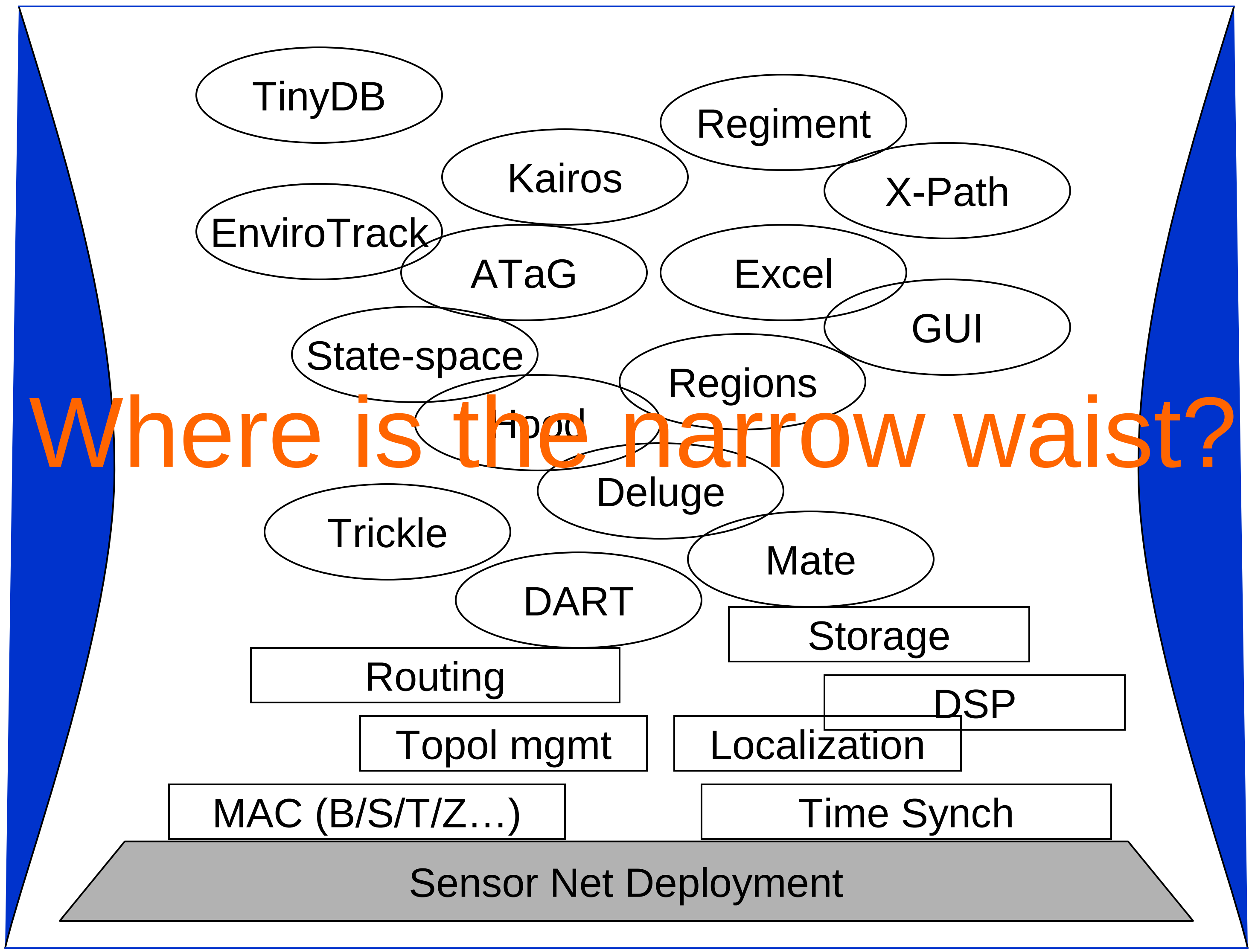
But centralized data collection is not scalable with the number of nodes.

Parking garage testbed/scenarios

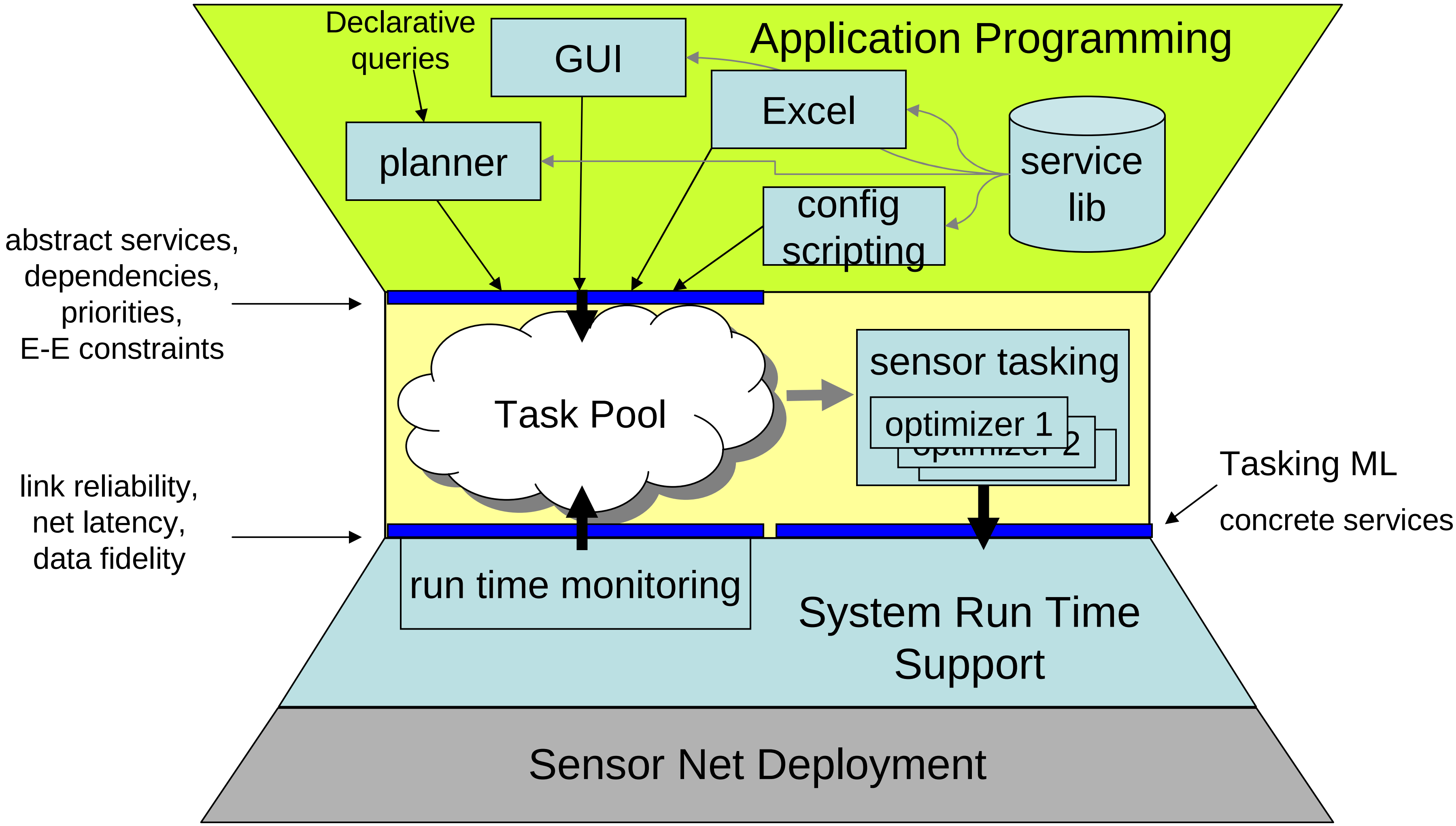


But the problem is made hard by …

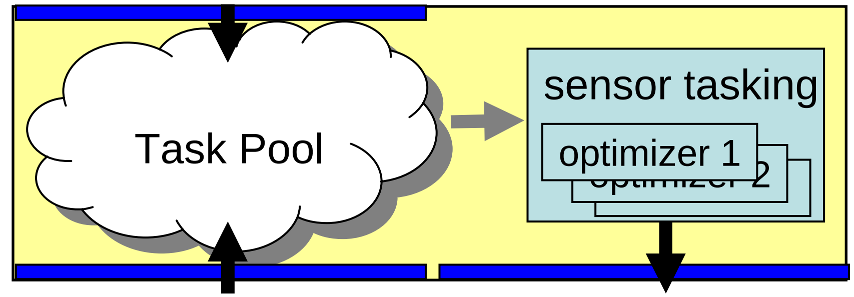
- A great deal of uncertainty
  - System dynamism and unreliability
    - Nodes/links come and go
    - Possibly non-replenishable resources
    - Parts of system may be deployed over time by different vendors, using different technologies (e.g., network protocols)
  - Data uncertainty
    - Uncertain data due to sensor noise, packet loss
    - Incomplete information due to partial observability of the world
- User tasks often specified in high-level languages
  - E.g., “Tell me if you see a red car”, or, “Doing this with that data”.
- As a result, a number of systems have been built vertically
  - With own system abstractions and data models
  - Make sense from efficiency pov, but with relatively small code reuse


Where is the narrow waist?
TinyDB
Regiment
Kairos
X-Path
EnviroTrack
ATaG
Excel
GUI
State-space
Regions
Hood
Deluge
Trickle
Mate
DART
Storage
Routing
DSP
Topol mgmt
Localization
MAC (B/S/T/Z…)
Time Synch
Sensor Net Deployment

Proposed Common Interfaces

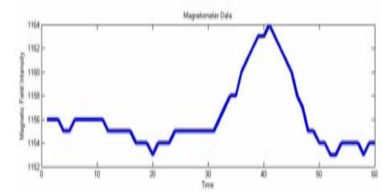


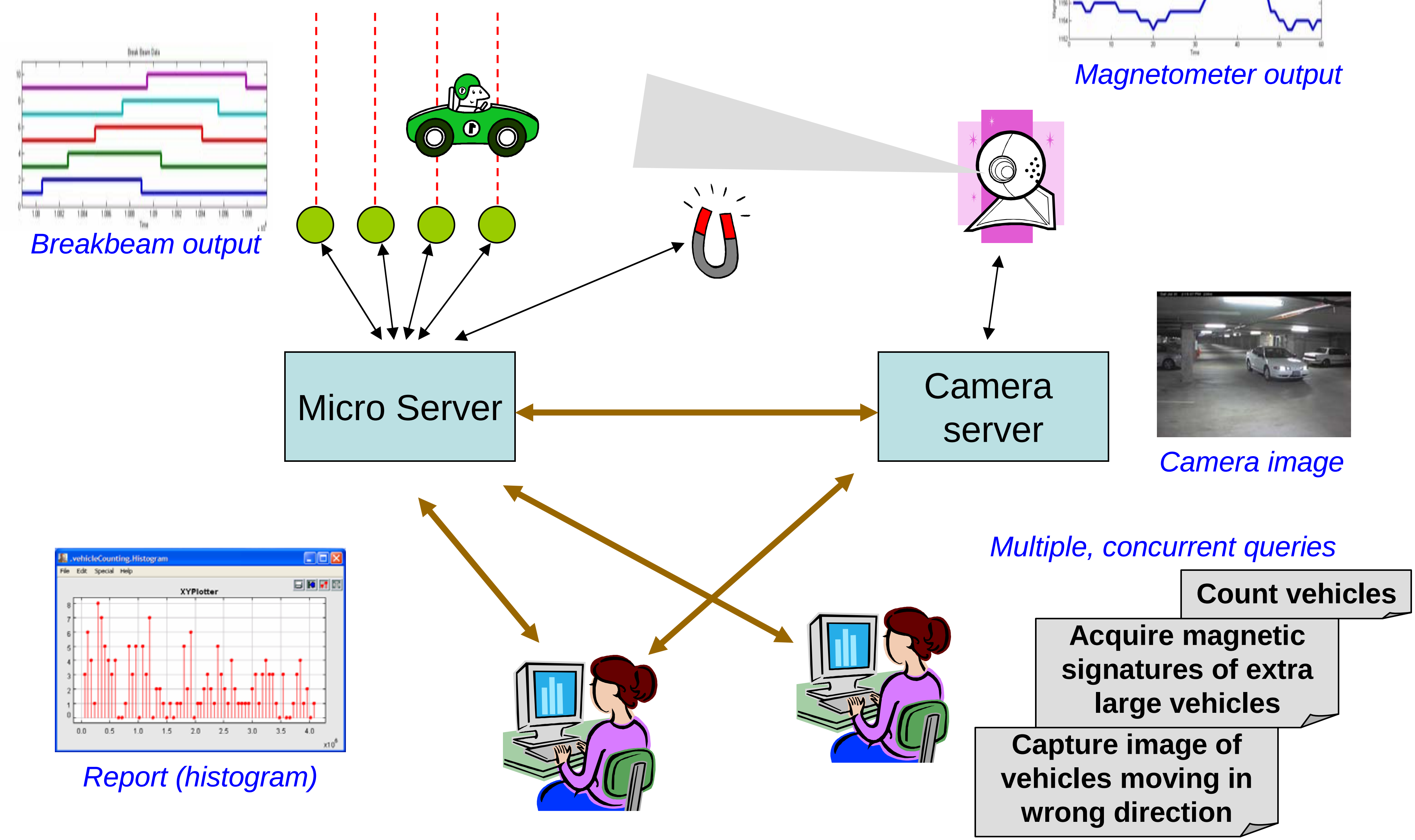
Task Pool Abstraction



- Application programs inject abstract services and their dependencies into task pool, specifying what/how
  - Services have priorities
  - Applications have end-to-end constraints such as latency, data quality, energy usage
- Sensor Tasking embed services onto nodes, instantiate where/when
  - Need information about physical topology, link quality, latency, data fidelity from run time
  - Search for optimal assignment satisfying EE constraints
    - Using a variety of tools such as CLP(R), LP, or Monte Carlo
  - Tasking ML describes concrete services instances
- Task pool is agnostic to application programming environments and run-time system support, provided that
  - Services are described in a common intermediate language
  - Run-time provides system and data reliability info

Concurrent Uncoordinated Tasks



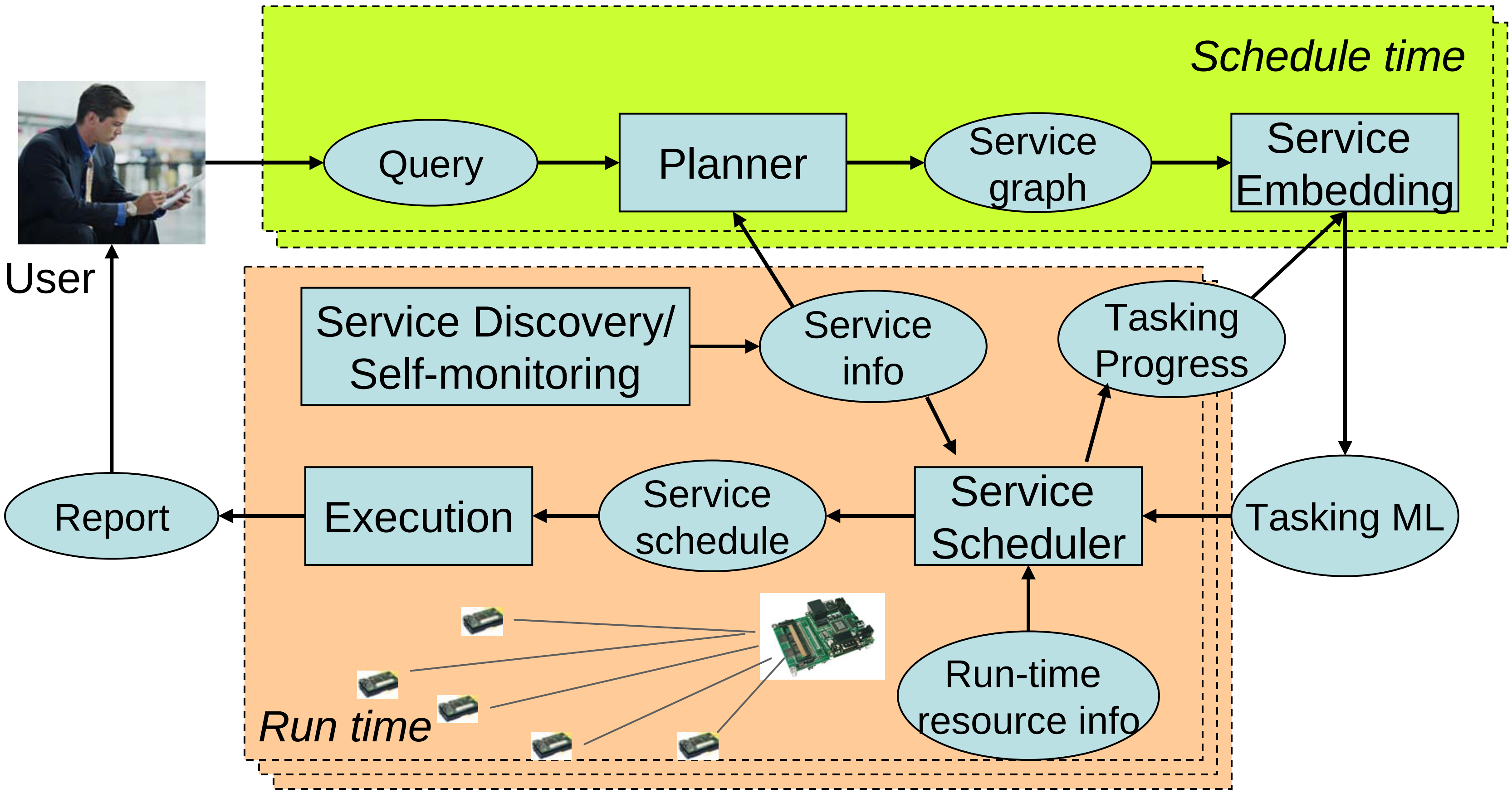


Tasks are sent to microservers at uncoordinated times, running for unpredictable duration. Tasks may partially overlap.

A Service Model for Sensor Net

Sensor net as provider of services

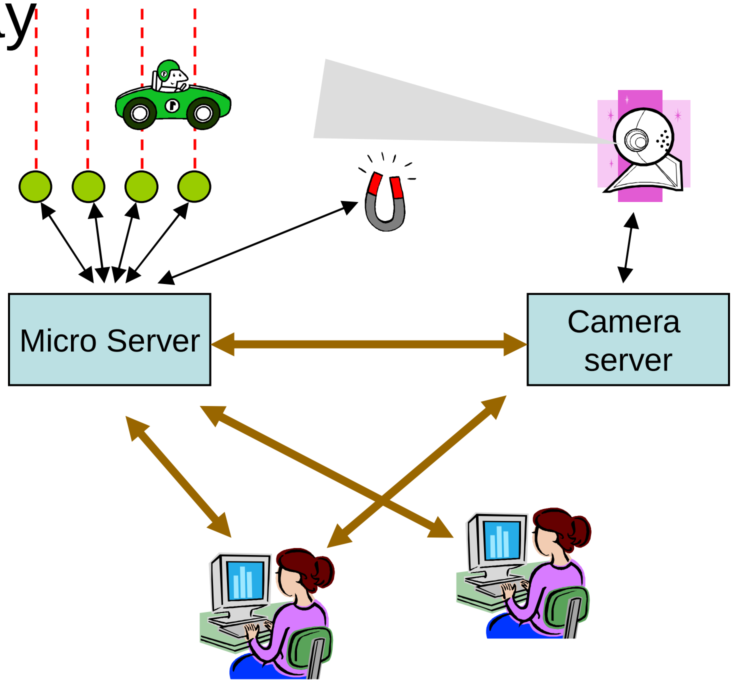
- Services encapsulating data and computation
- Service discovery, composition, execution
- Tasking description

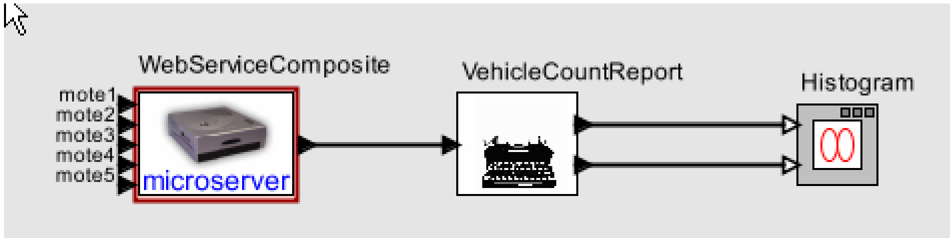


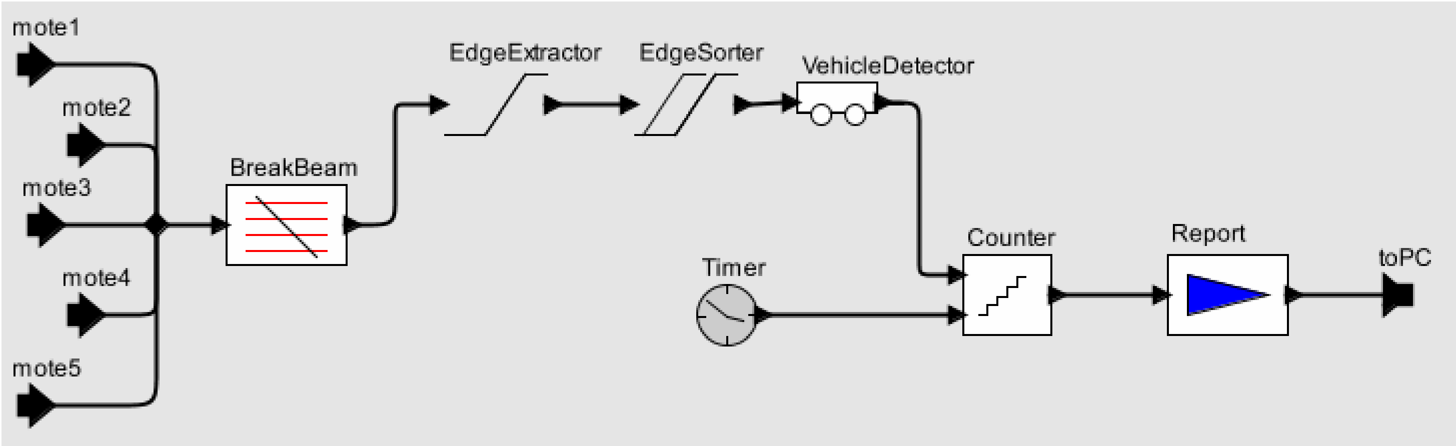
Example of services and their composition

Counting vehicles with a sensor array

- – Extract edges from break beam detections
- – Sort edges into consecutive detections
- – Detect vehicles based on timing relations among detections
- – Count vehicles
- – Generate an arrival histogram report

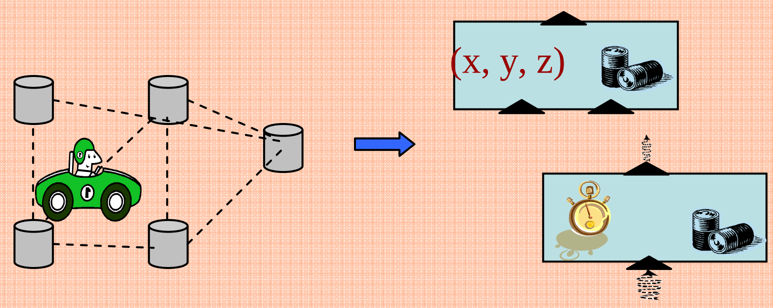




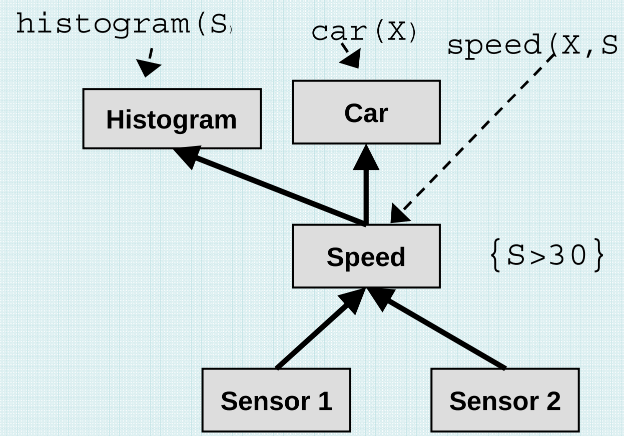


SONGS: Service Oriented Networked ProGramming of Sensors

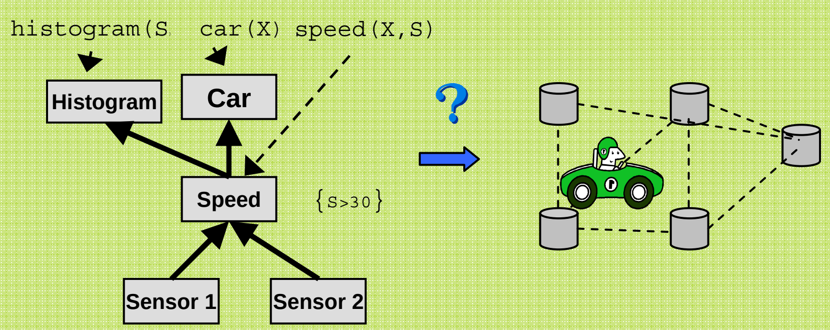
Service Abstraction and Interface



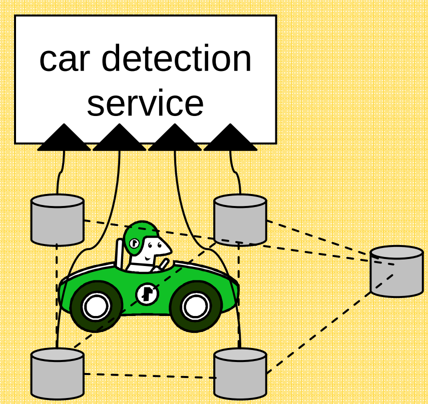
Service Planning



Service Embedding



Service Scheduling and Execution

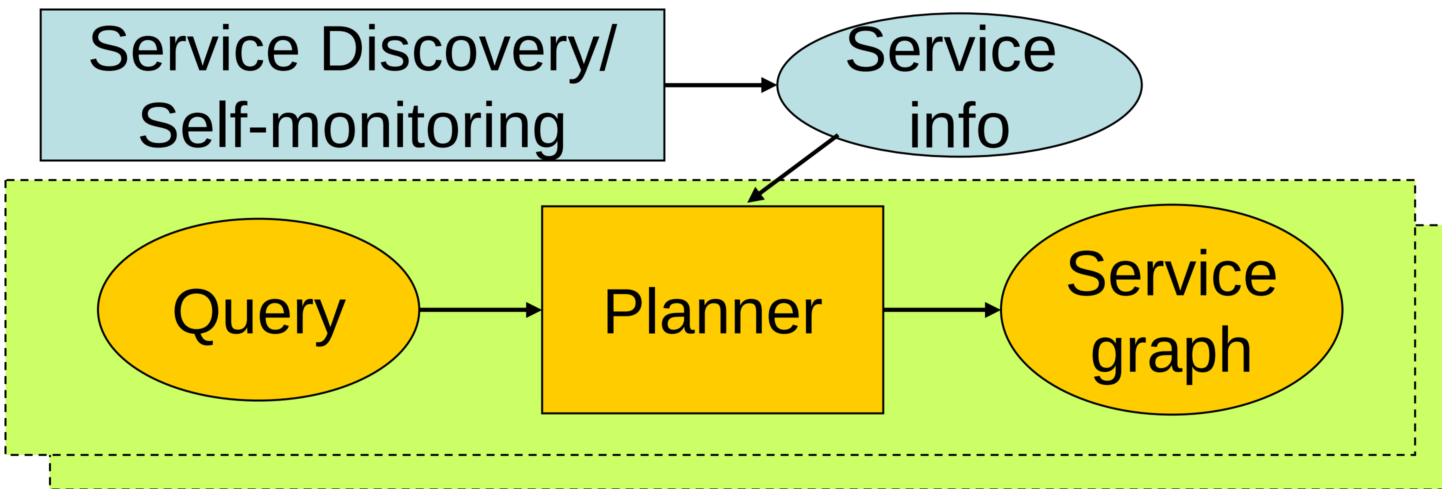


Service Abstraction and Interfaces

- In embedded sensing systems, service abstractions are more than just invocation interfaces.
- A rich service interface may contain:
  - Interaction interface
    - data I/O
    - data semantics (e.g. sensing modality, sampling rate, uncertainty)
  - Location
  - Resource requirements (e.g. in power and bandwidth)
  - Fidelity specifications (e.g. uncertainties in processing)

Service Planning

- High-level user interface eases programming
  - Declarative or Excel-like
- Automatic service planning uses goal-directed inference
- Resource sharing and reuse supports system retaskability
- Constraint-based specification allows flexibility in design trade-off



Automatic Service Planning

- Declarative specification
- Automatic resource sharing and reuse
- Reasoning about space/time constraints
- Execution monitoring and re-planning
- Constraint-based trade-offs

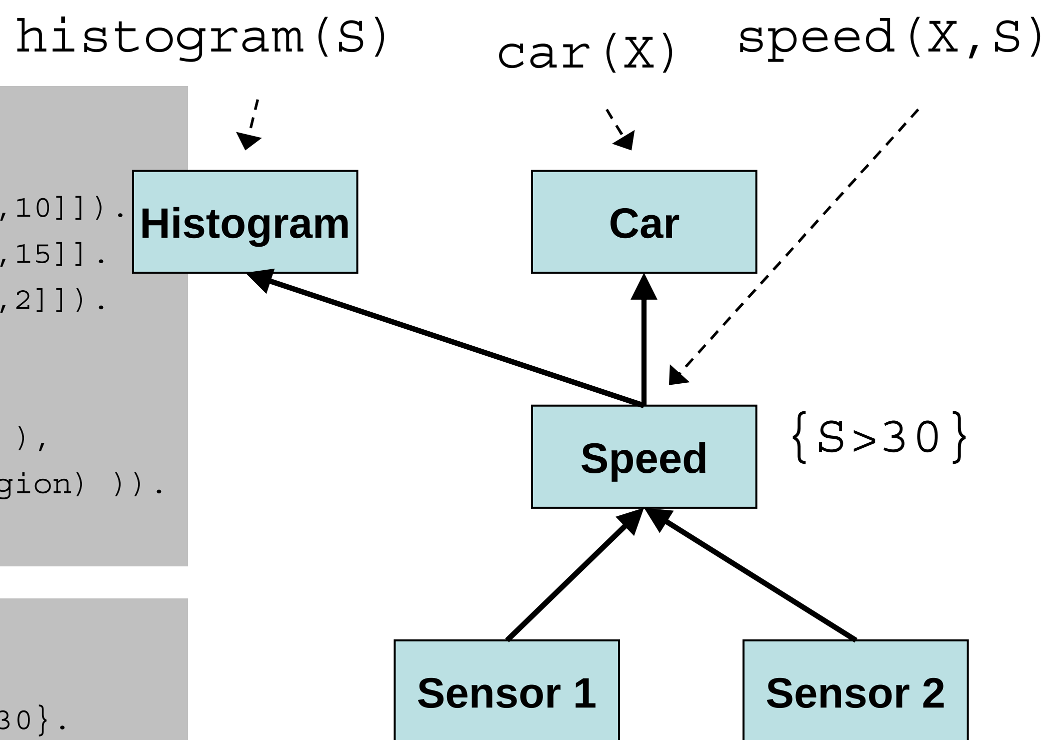
Service Description

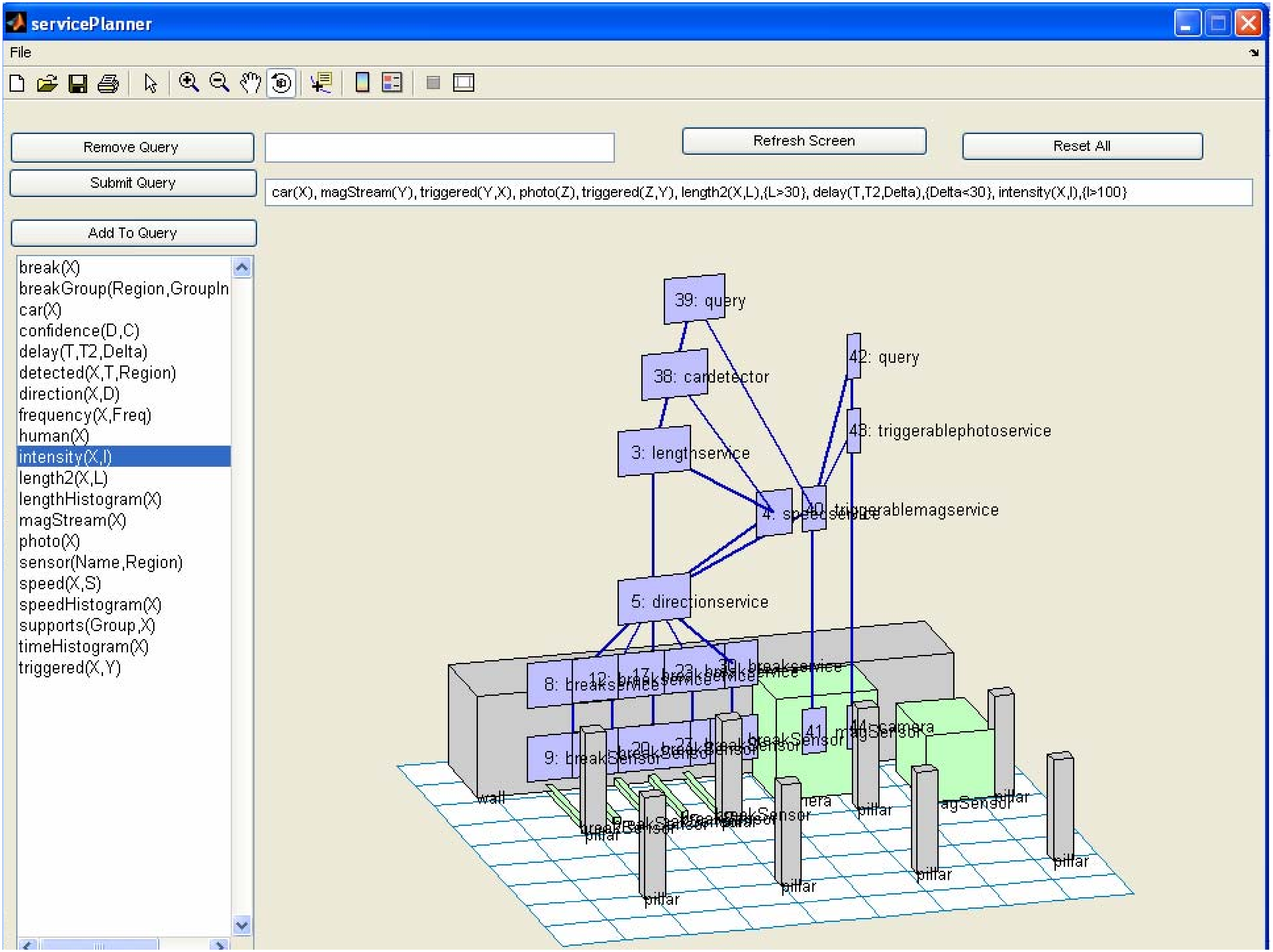
```
sensor( magSensor,    [[60,0,0],[70,10,10]]).
sensor( camera,       [[40,0,0],[55,15,15]].
sensor( breakSensor,  [[10,0,0],[12,10,2]]).

service(  breakService (Region),
needs(    sensor (breakSensor,Region) ),
creates(  break (X), detected (X,T,Region) )).
```

User Query

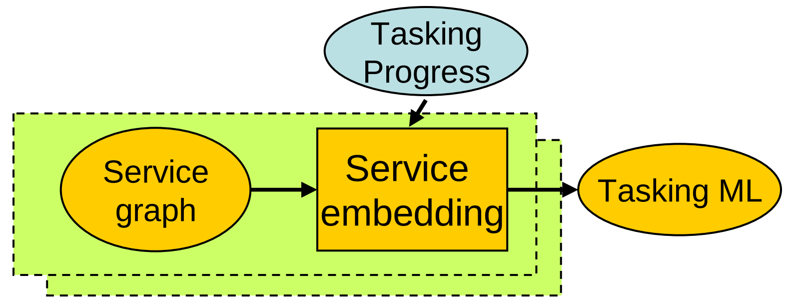
```
car(X), speed(X,S), histogram(S), {S>30}.
```

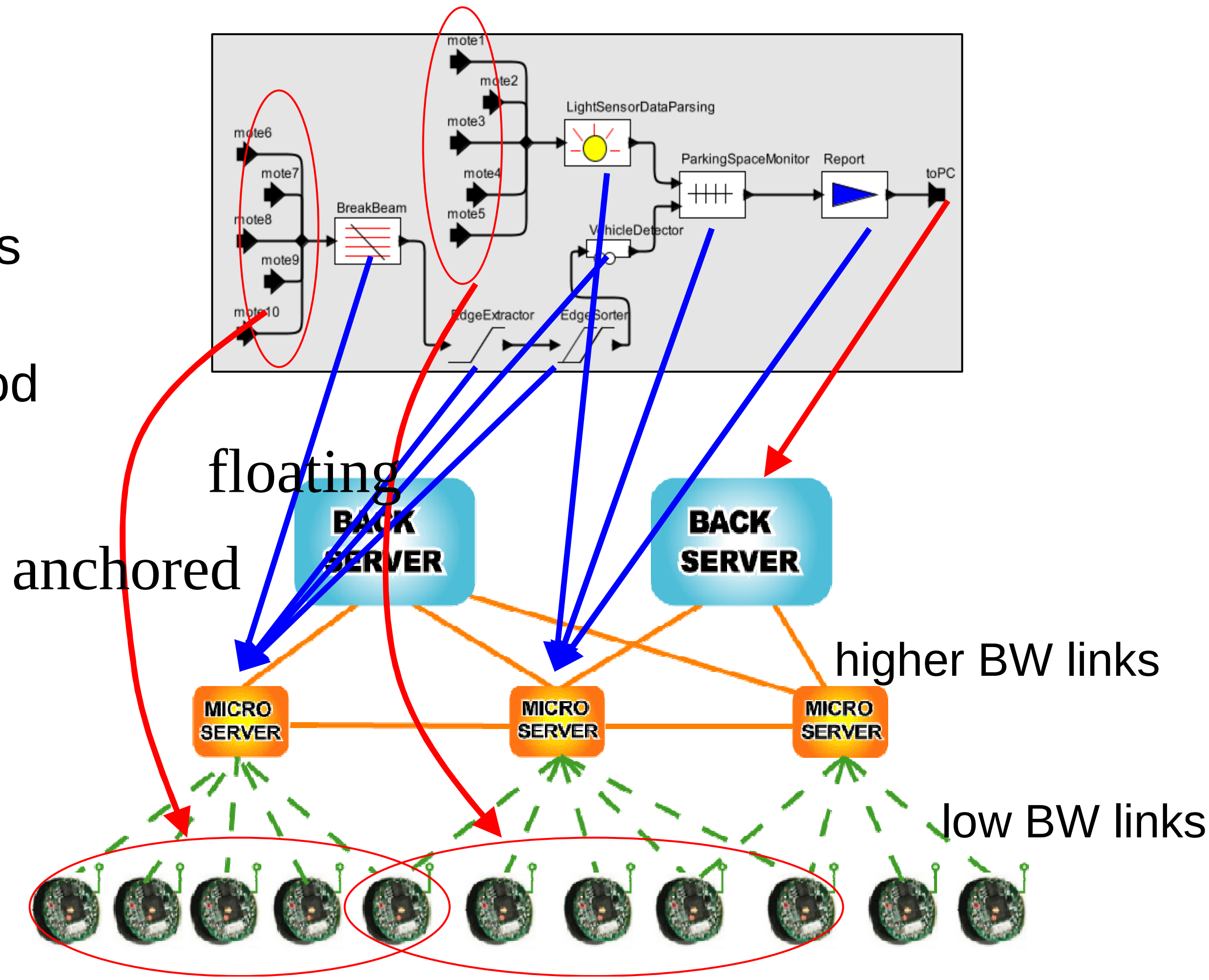



servicePlanner
File
Remove Query
Submit Query
Add To Query
Refresh Screen
Reset All
car(X), magStream(Y), triggered(Y,X), photo(Z), triggered(Z,Y), length2(X,L),{L>30}, delay(T,T2,Delta),{Delta<30}, intensity(X,I),{I>100}
break(X)
breakGroup(Region,GroupIn
car(X)
confidence(D,C)
delay(T,T2,Delta)
detected(X,T,Region)
direction(X,D)
frequency(X,Freq)
human(X)
intensity(X,I)
length2(X,L)
lengthHistogram(X)
magStream(X)
photo(X)
sensor(Name,Region)
speed(X,S)
speedHistogram(X)
supports(Group,X)
timeHistogram(X)
triggered(X,Y)
39: query
38: cardetector
42: query
43: triggerablephotoservice
3: lengthservice
40: triggerablemagservice
5: directionservice
8: breakservice
9: breakSensor
wall
pillar

Service Embedding

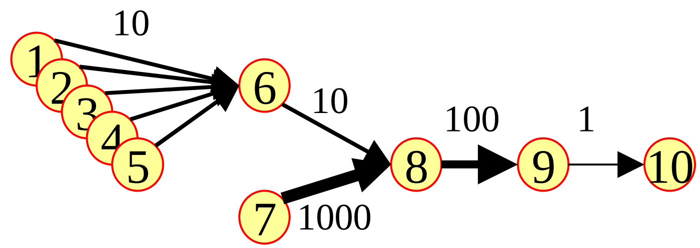
- Embed service graph onto physical nodes
- Preserve proximities in data flows
- Optimize for resource usage, load balance, latency, and robustness
- Exploit physical abstractions to find good mapping

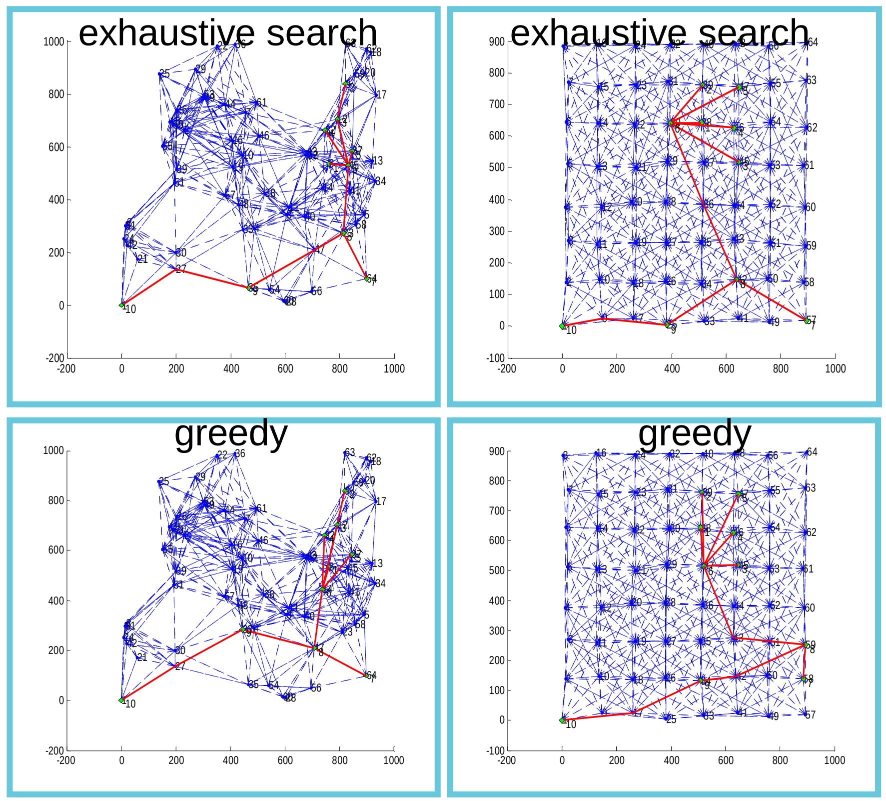




Proximity Based Greedy Search

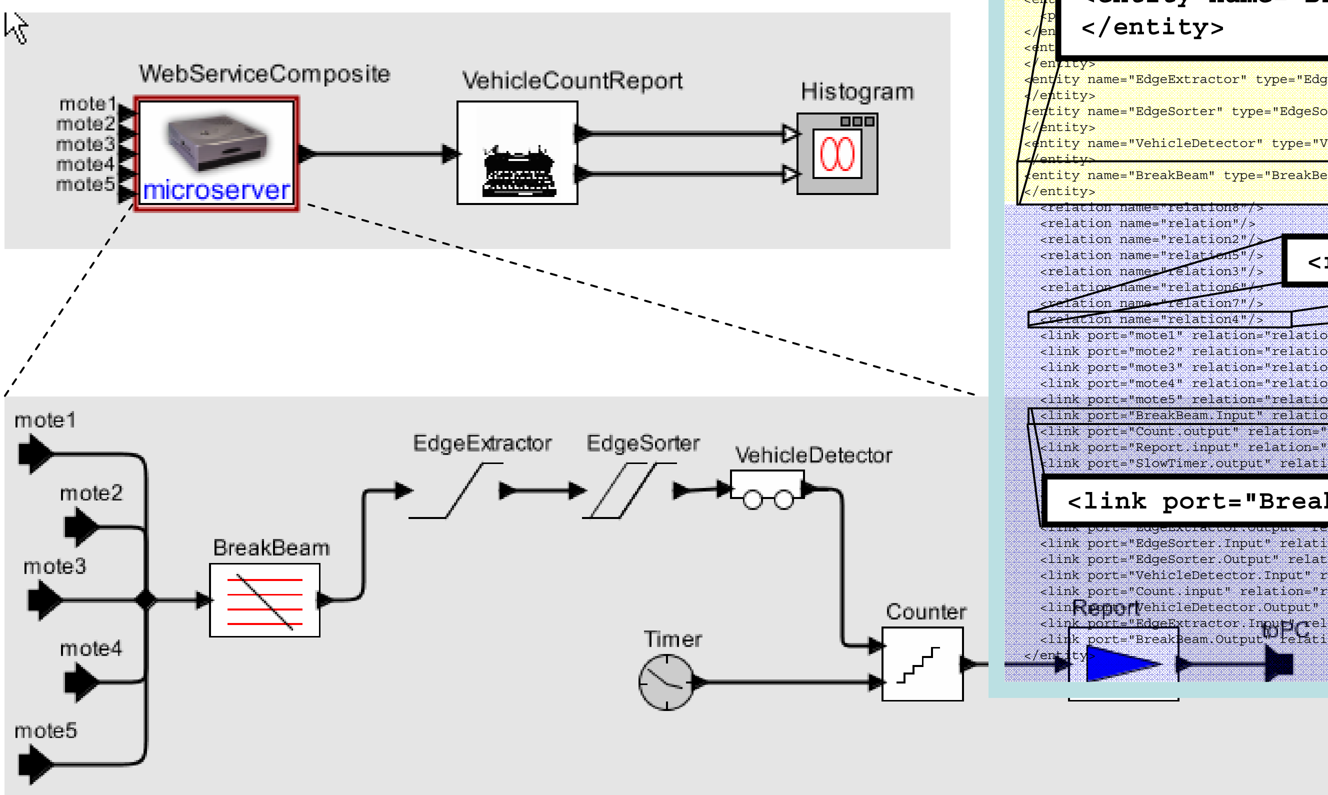
- Assume nodes close in physical space are also close in the communication graph
- Sort nodes in SCG into levels, where sensor nodes at bottom level are anchored.
- Starting from the bottom, for each floating service, place it at the weighted geographical mean of child nodes
- Repeat until all services are placed.
- Map each service at a virtual location to nearest physical node

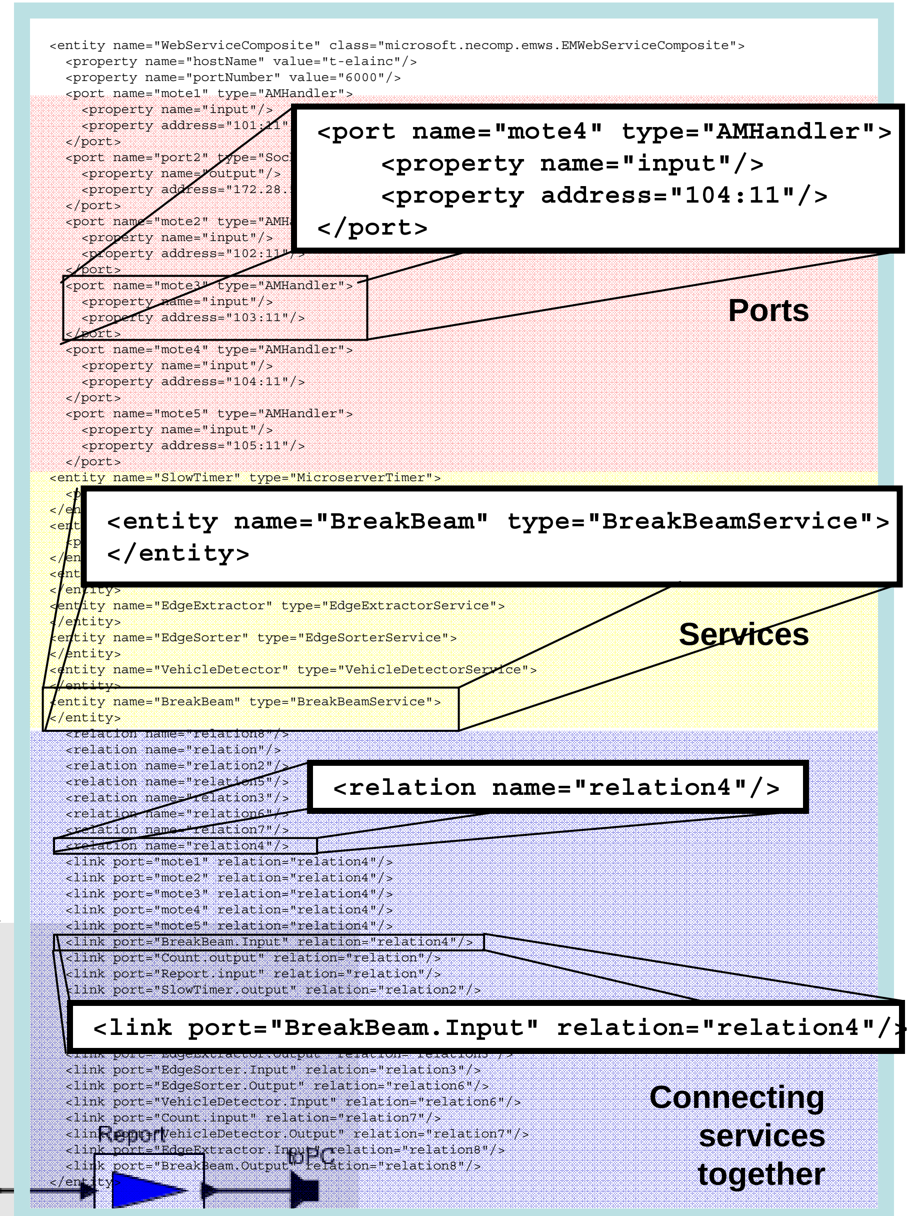




An Example of TML

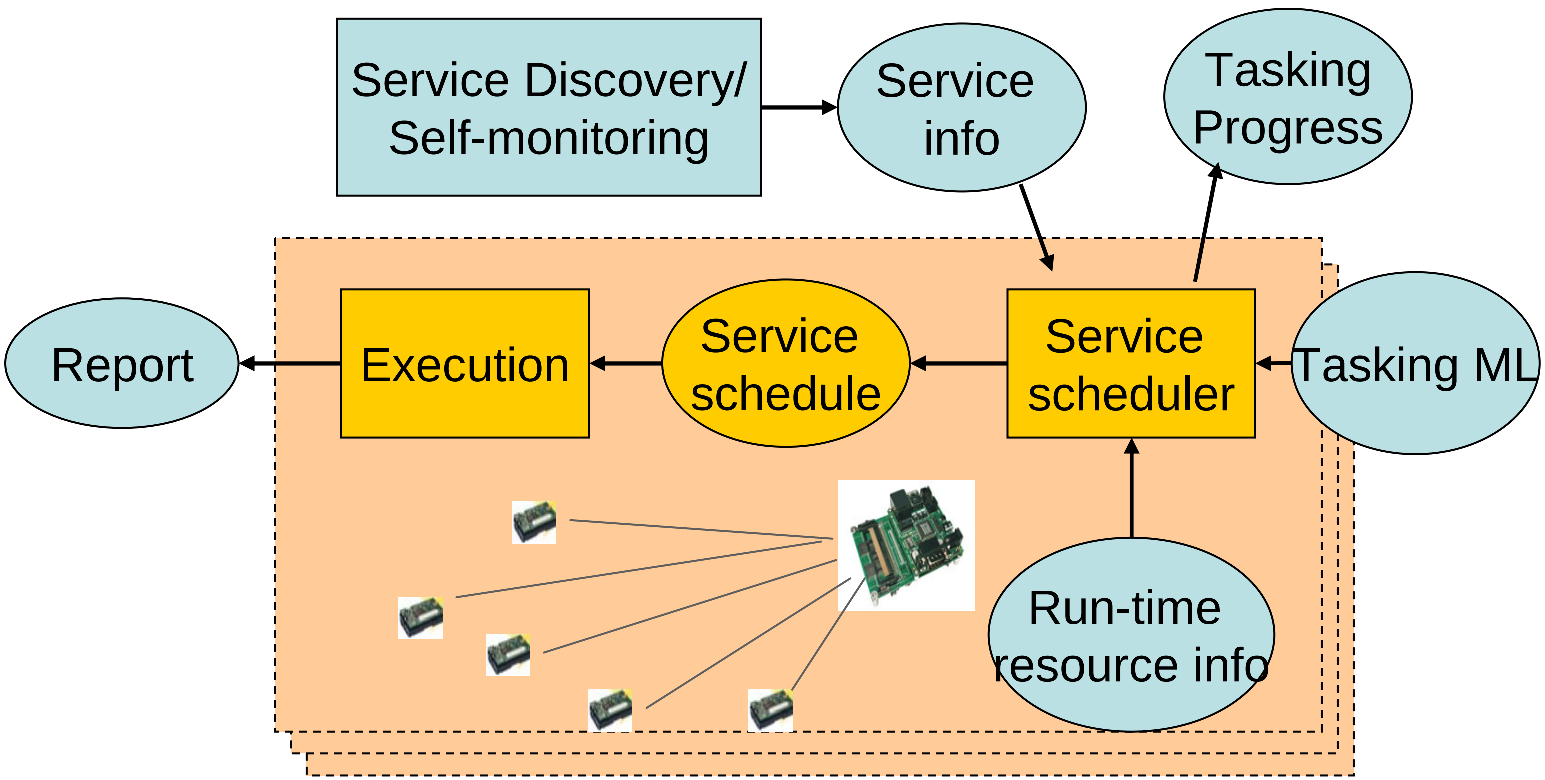
- Task graph for counting vehicle query
  - Ports
  - Services
  - Wiring services together
- Description in Tasking ML (micro-server tasking markup language)



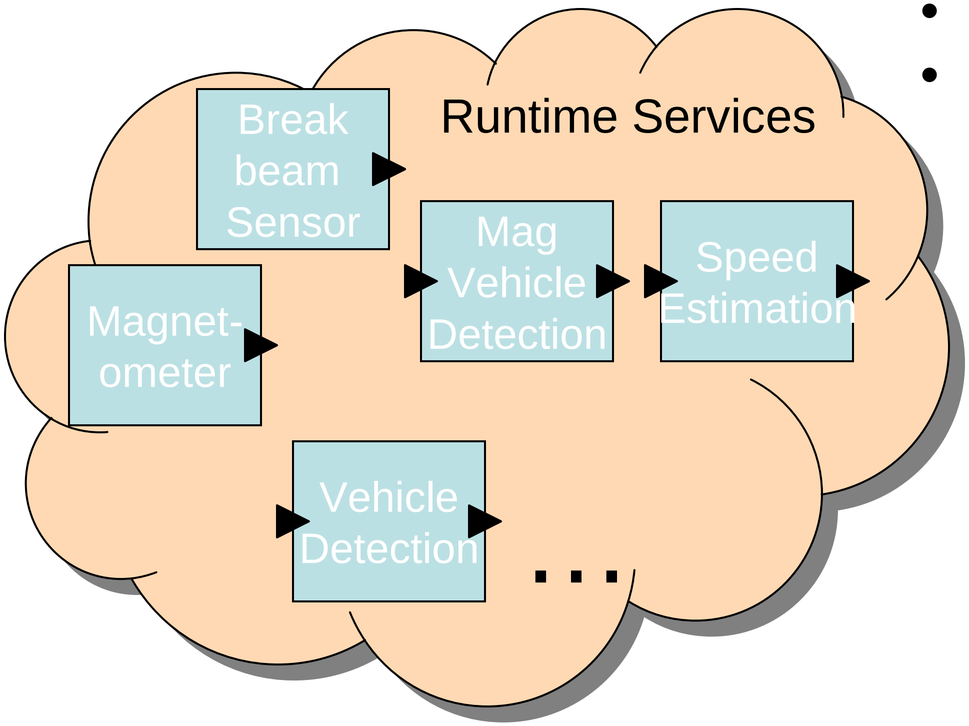


Service Run Time (SERUN)

- On-demand instantiation
- Adaptive to resource availability
- Collaboration among peers and between tasks
- Dynamic service migration

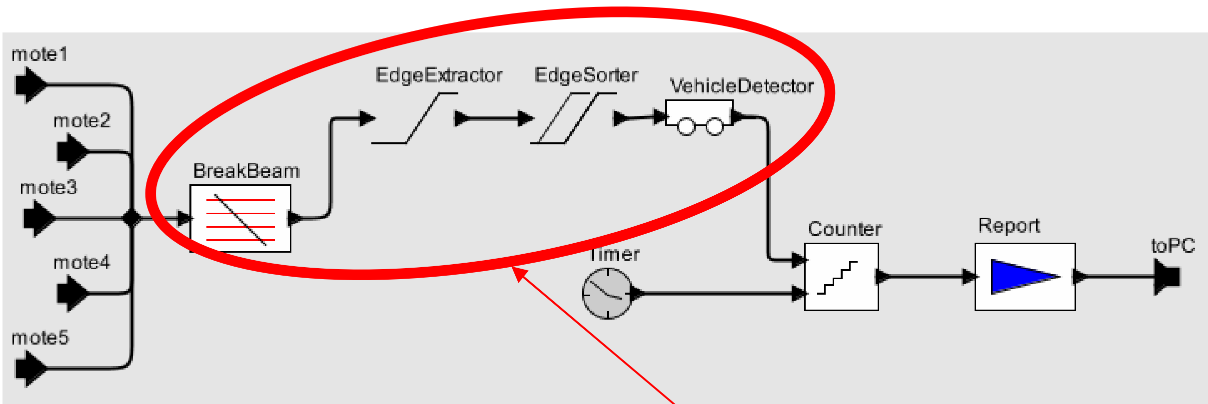


A Service Sharing Runtime Environment



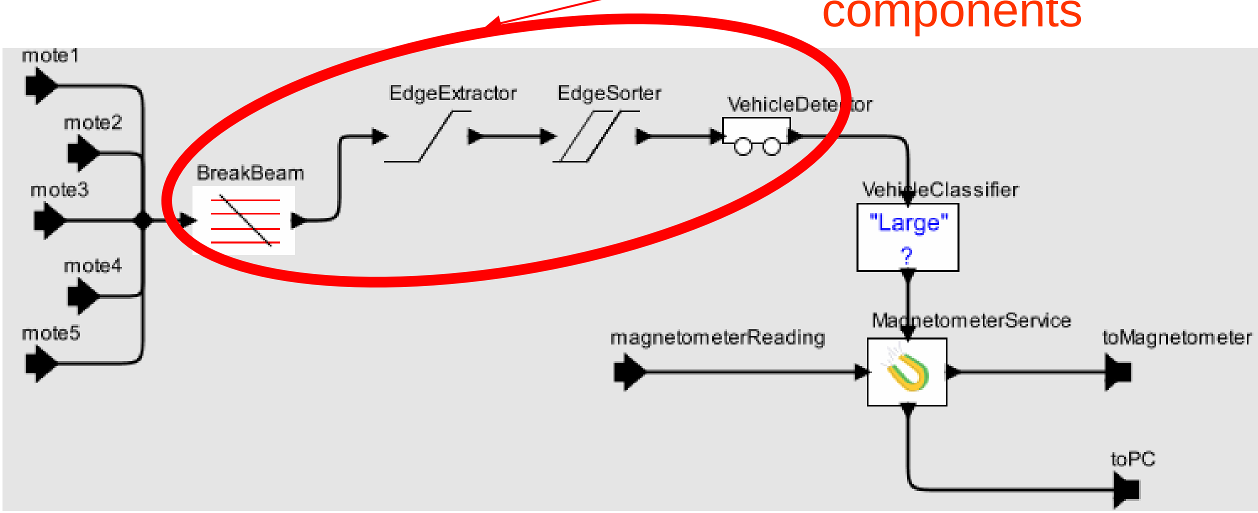
- Event-driven execution
- Publish/subscribe composition

Optimization in run-time service composition



Vehicle counting services

Optimize out common components

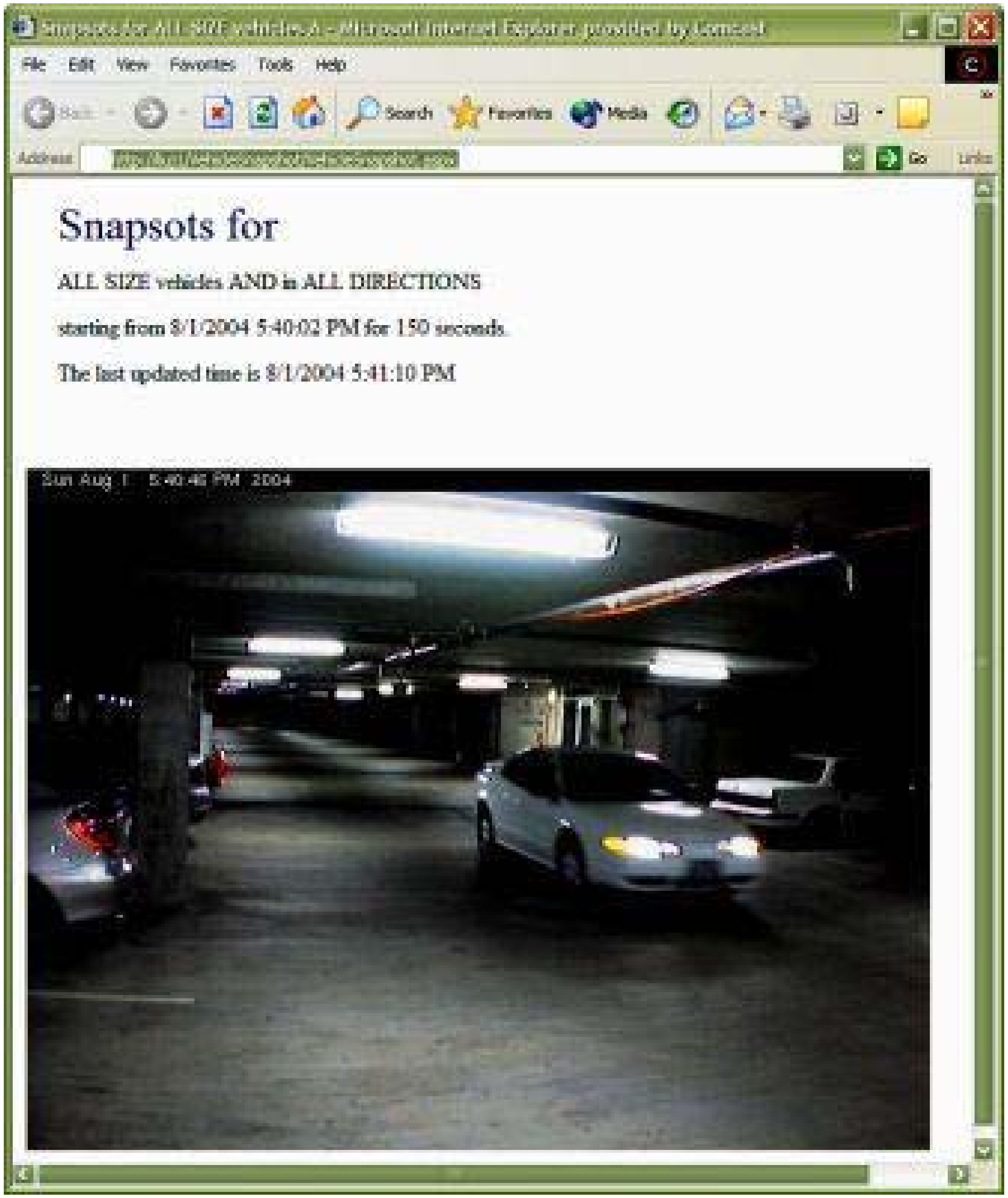


Large vehicle detection services

- Syntactic analysis
  - Graph analysis to remove duplications
- Semantic analysis
  - Analyze information content to determine reuse or sharing (e.g., sampling)

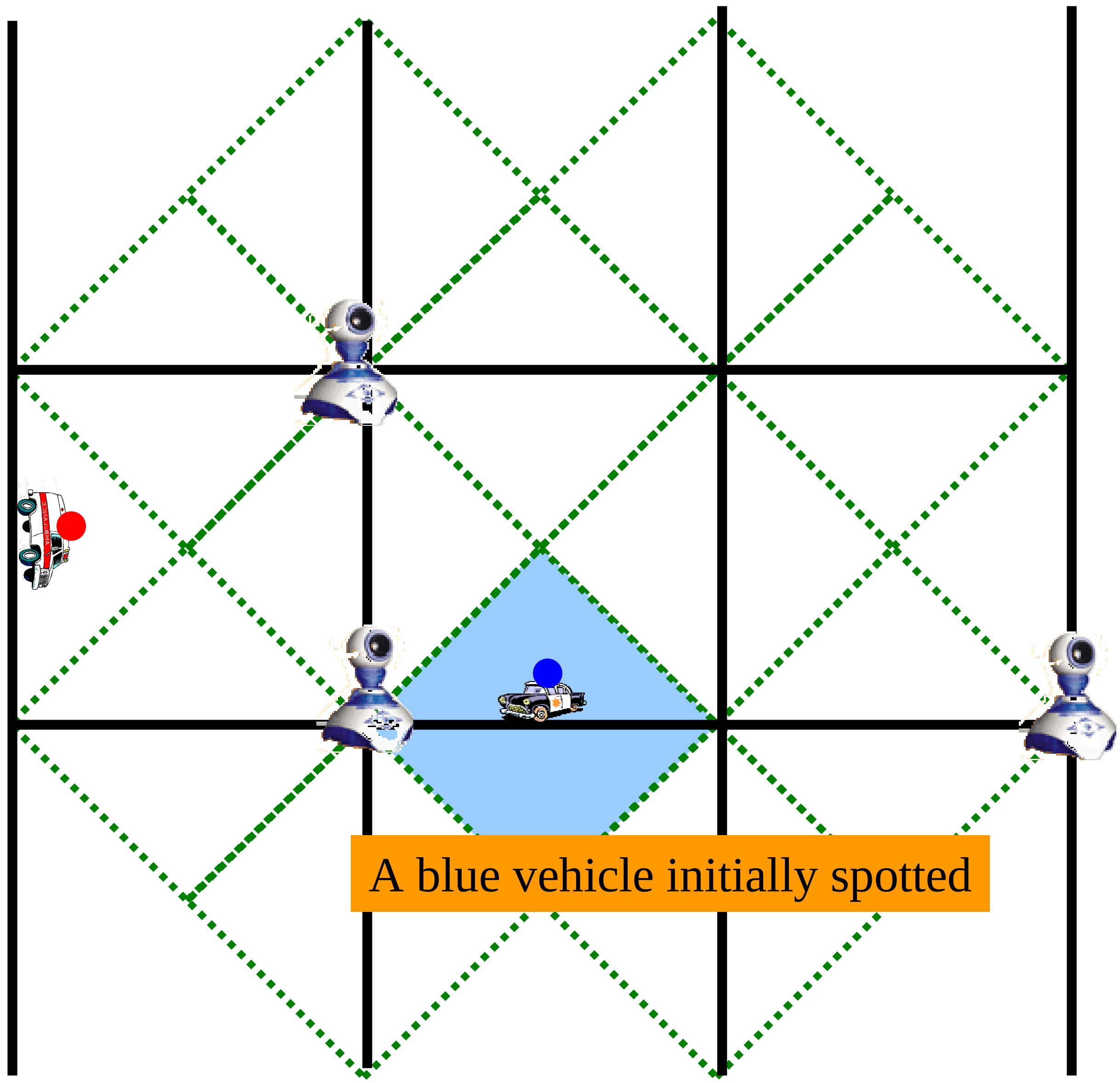
Video: car driving

- As the vehicle drives through the testbed, the infrared break beams can detect its speed, length and direction.
- If the vehicle is very large, a magnetic signature can be acquired. If the vehicle is speeding or moving in the wrong direction, an image can be acquired.
- Users can query the system independently, viewing output such as occurrences of vehicles.

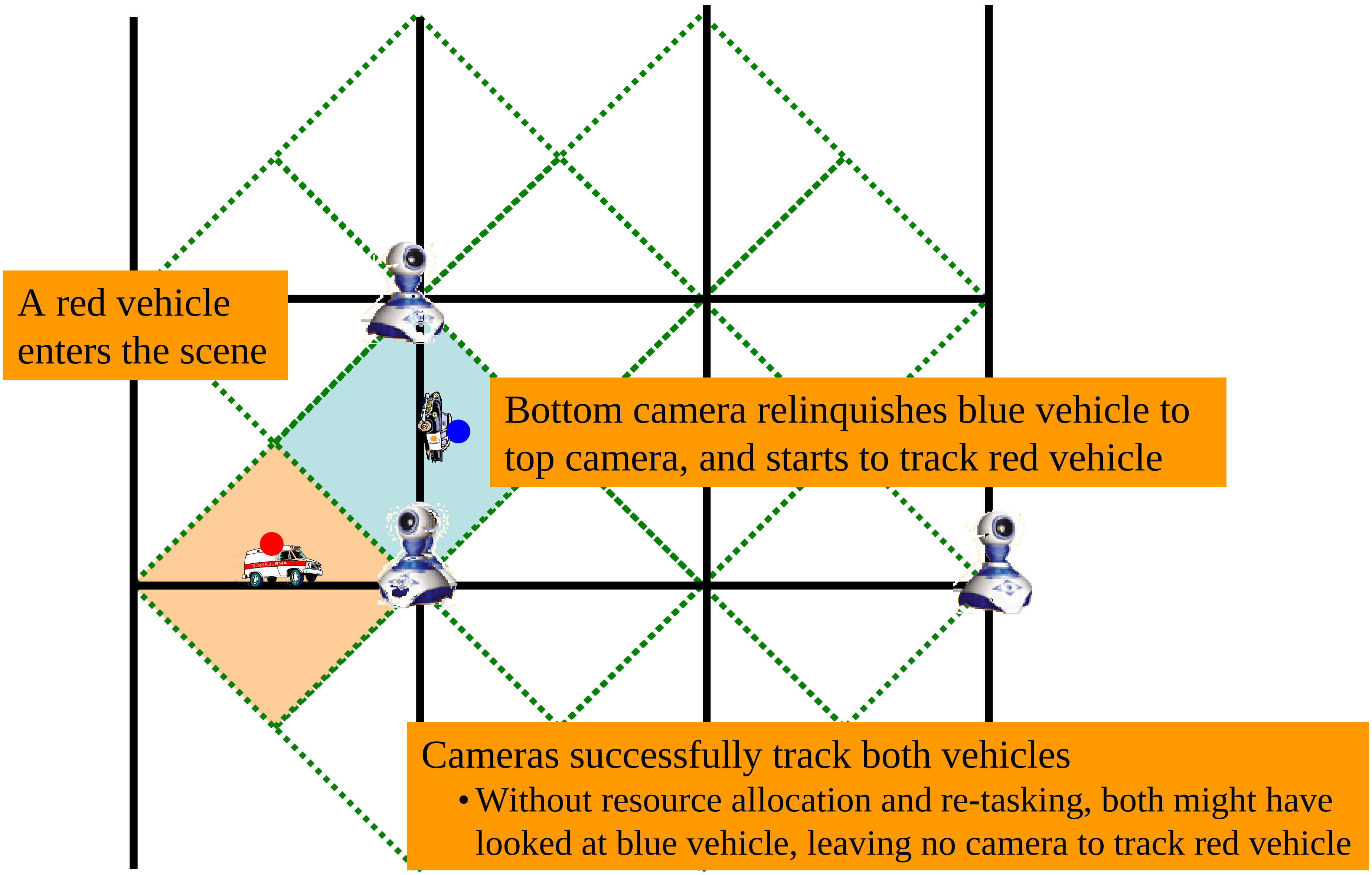


Capture an image of a speeding vehicle

Juggling Multiple Tasks



Juggling Multiple Tasks

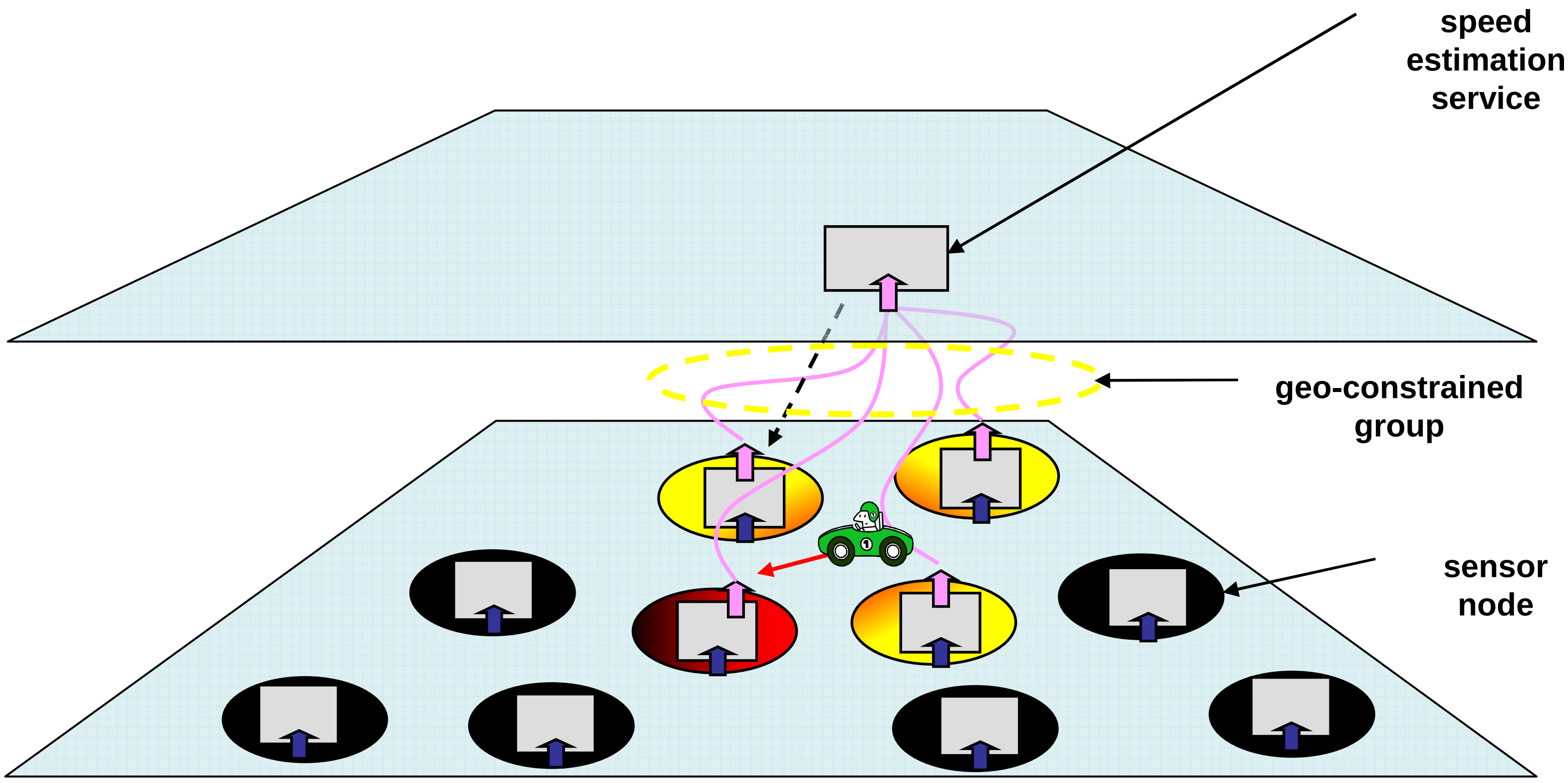


Sensor Tasking

- Unique to sensor net
  - Concurrent events, uncoordinated tasks, limited resources
- Tasking and re-tasking based on
  - Conflict resolution
  - Load balancing
  - Failure recovery
  - Cascading tasks
  - …
- Need to be aware of the physical phenomena being monitored as well as system configuration

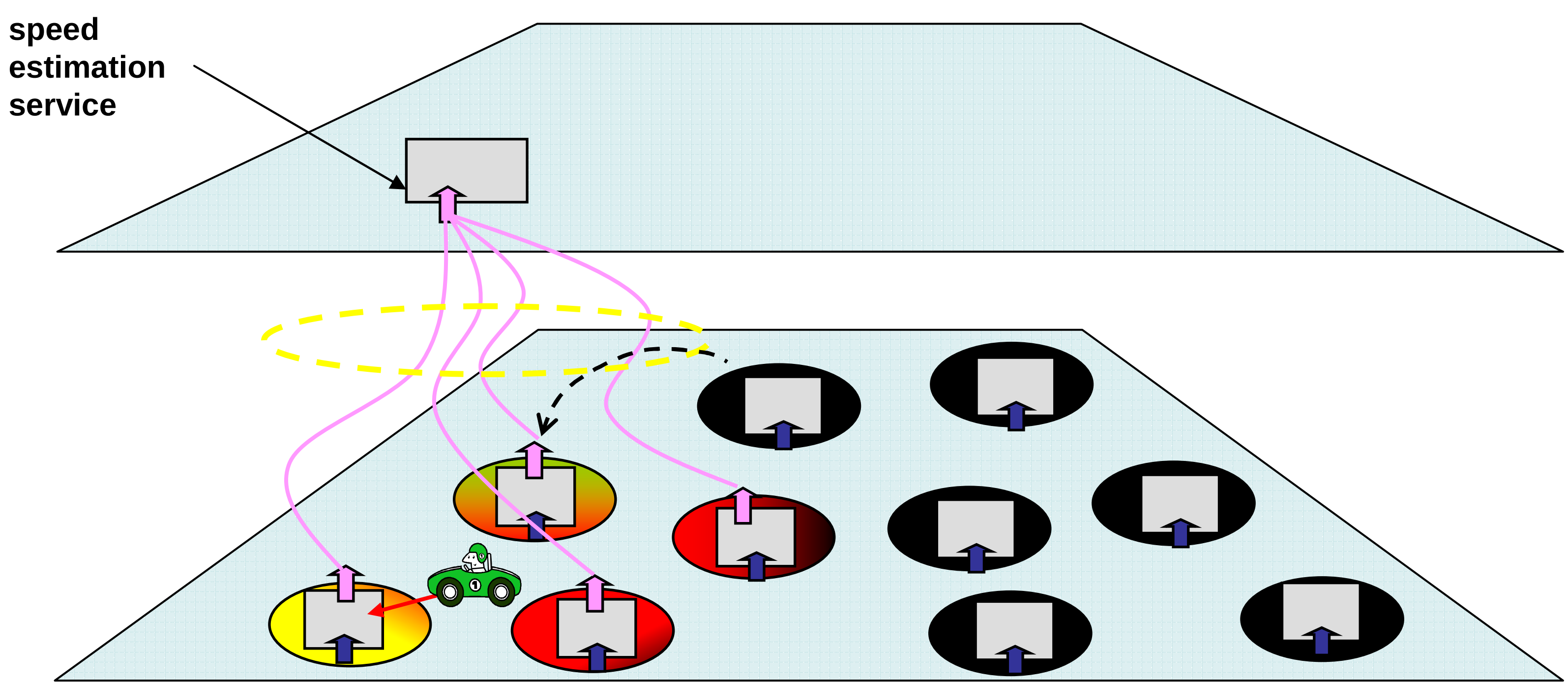
Manage both system and event uncertainty

A tracking example



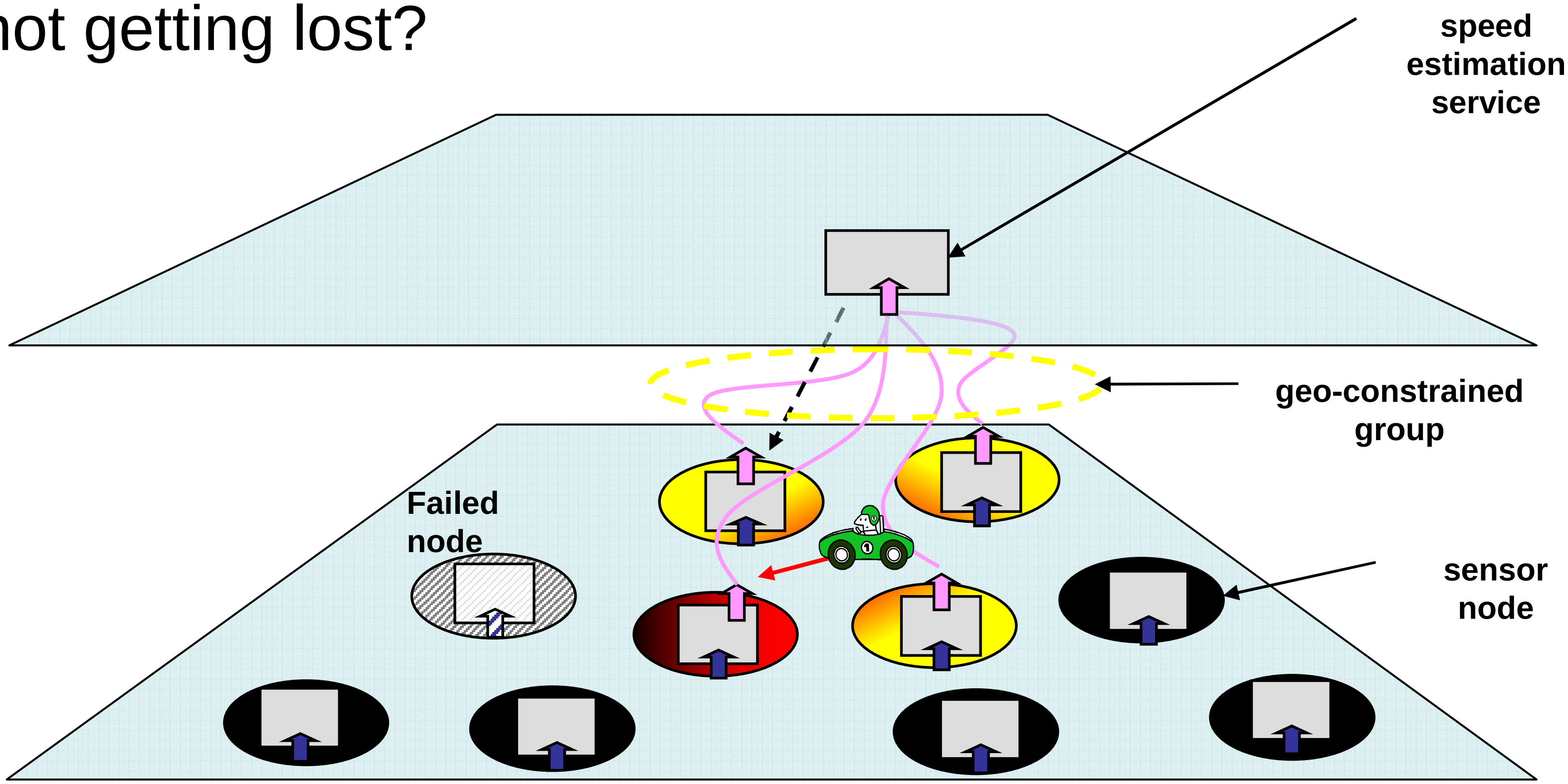
Migration …

As the event moves, one needs to move the code and/or state



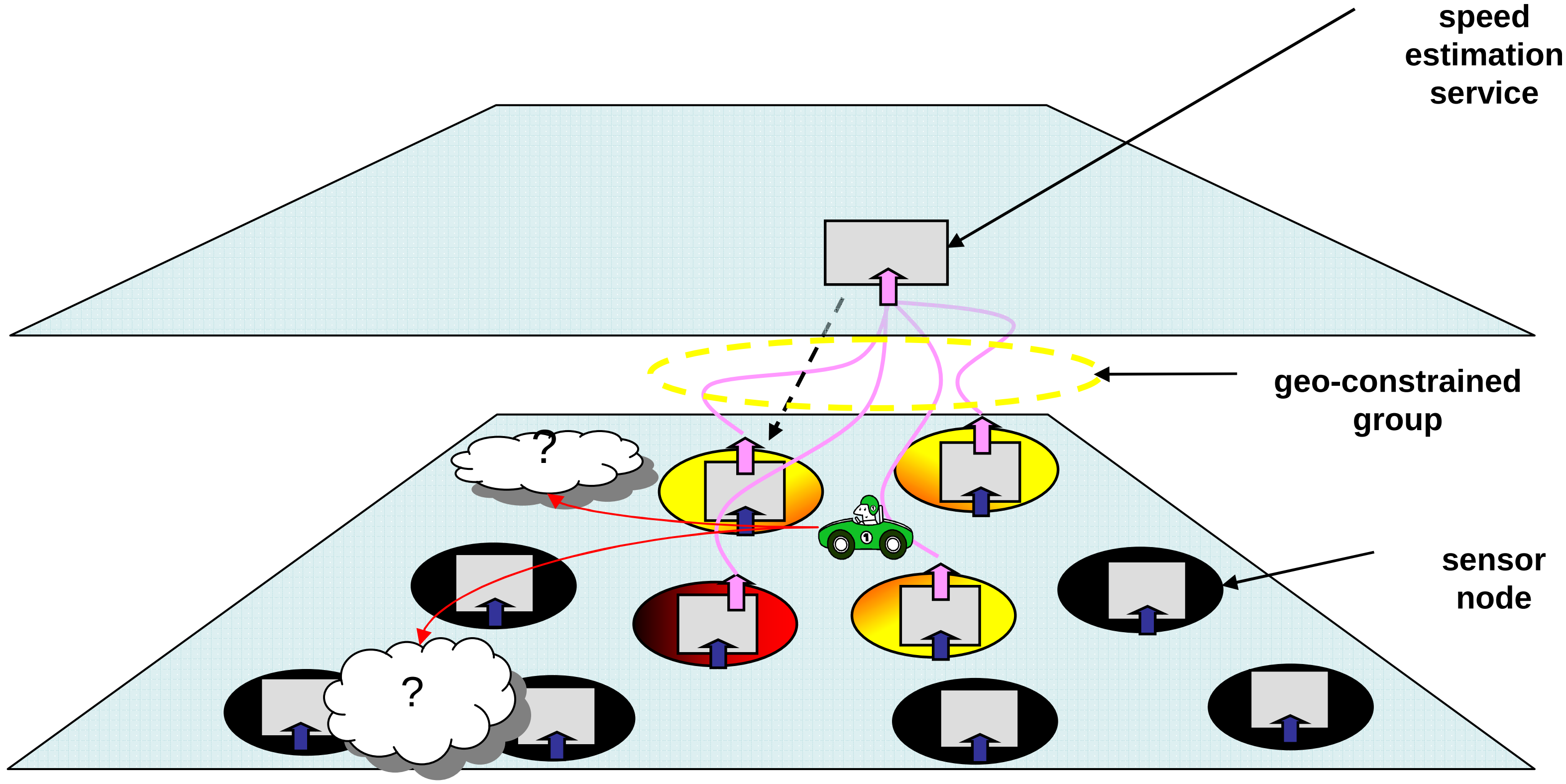
What if nodes fail?

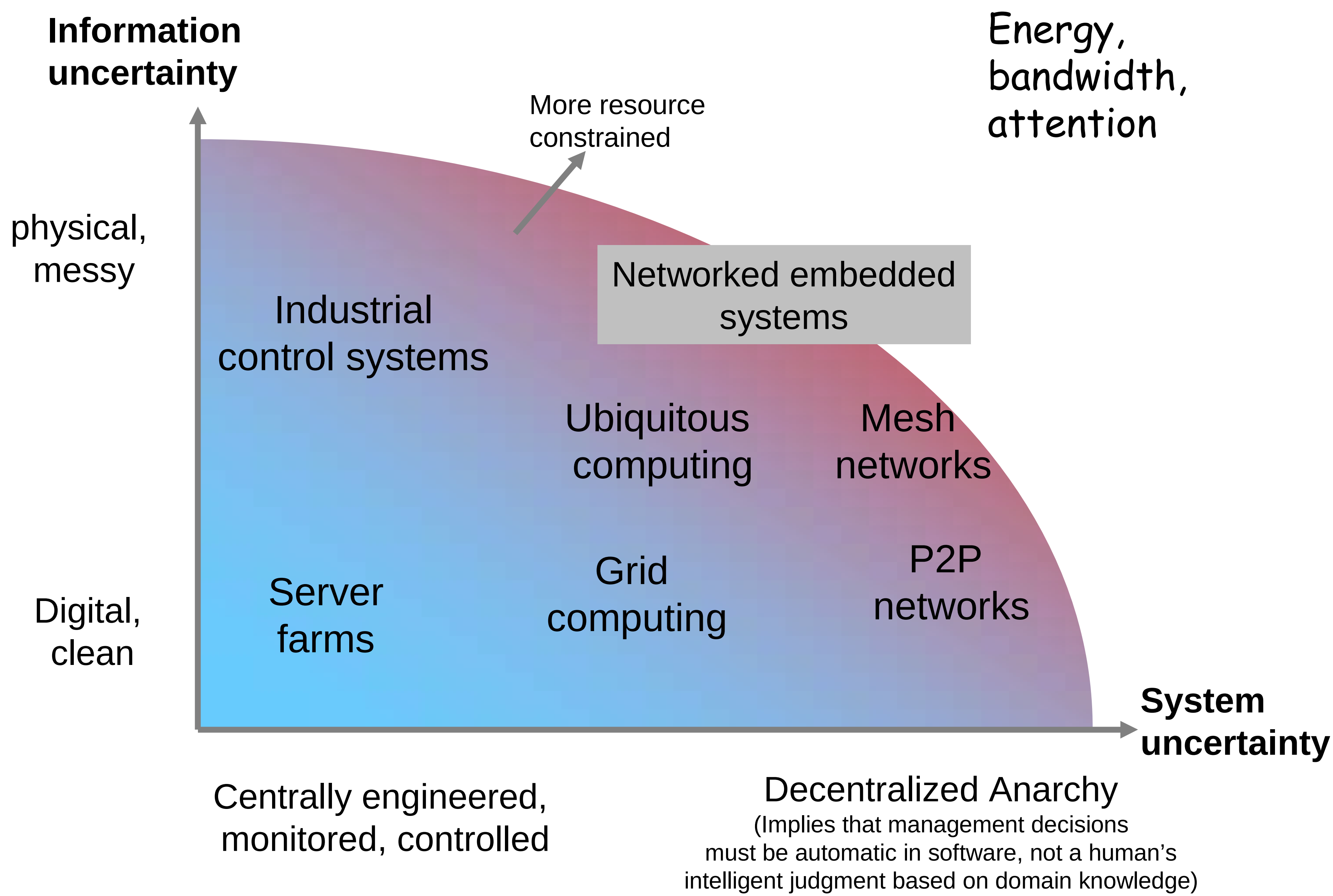
- Contingency plan?
- How many copies do I send around, to maximize odds of not getting lost?



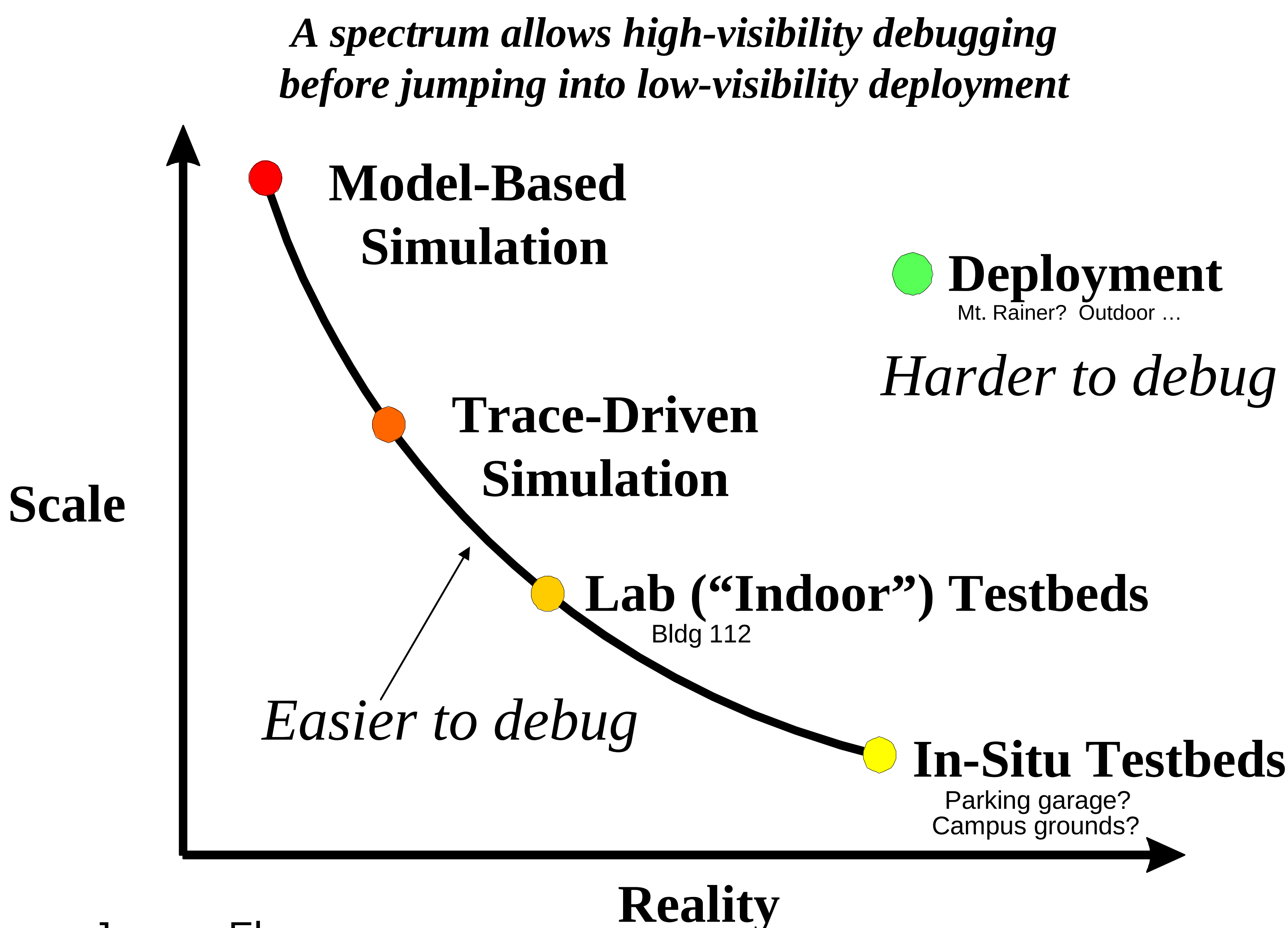
… and world knowledge is incomplete?

- Follow the clouds
- Probabilistic tasking?
- Proactive or reactive?



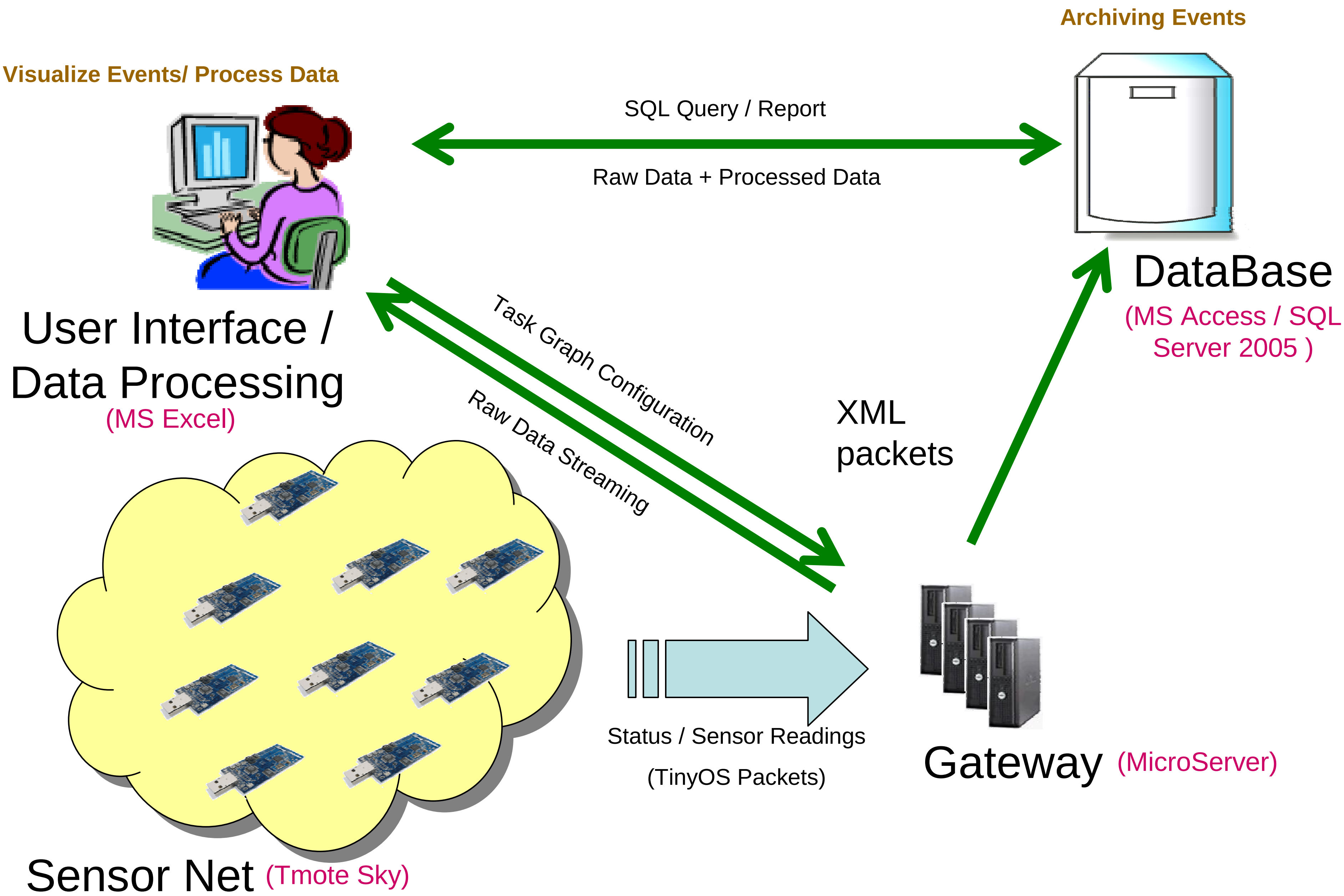

Information uncertainty
Energy, bandwidth, attention
More resource constrained
physical, messy
Networked embedded systems
Industrial control systems
Ubiquitous computing
Mesh networks
Digital, clean
Server farms
Grid computing
P2P networks
System uncertainty
Centrally engineered, monitored, controlled
Decentralized Anarchy
(Implies that management decisions must be automatic in software, not a human's intelligent judgment based on domain knowledge)

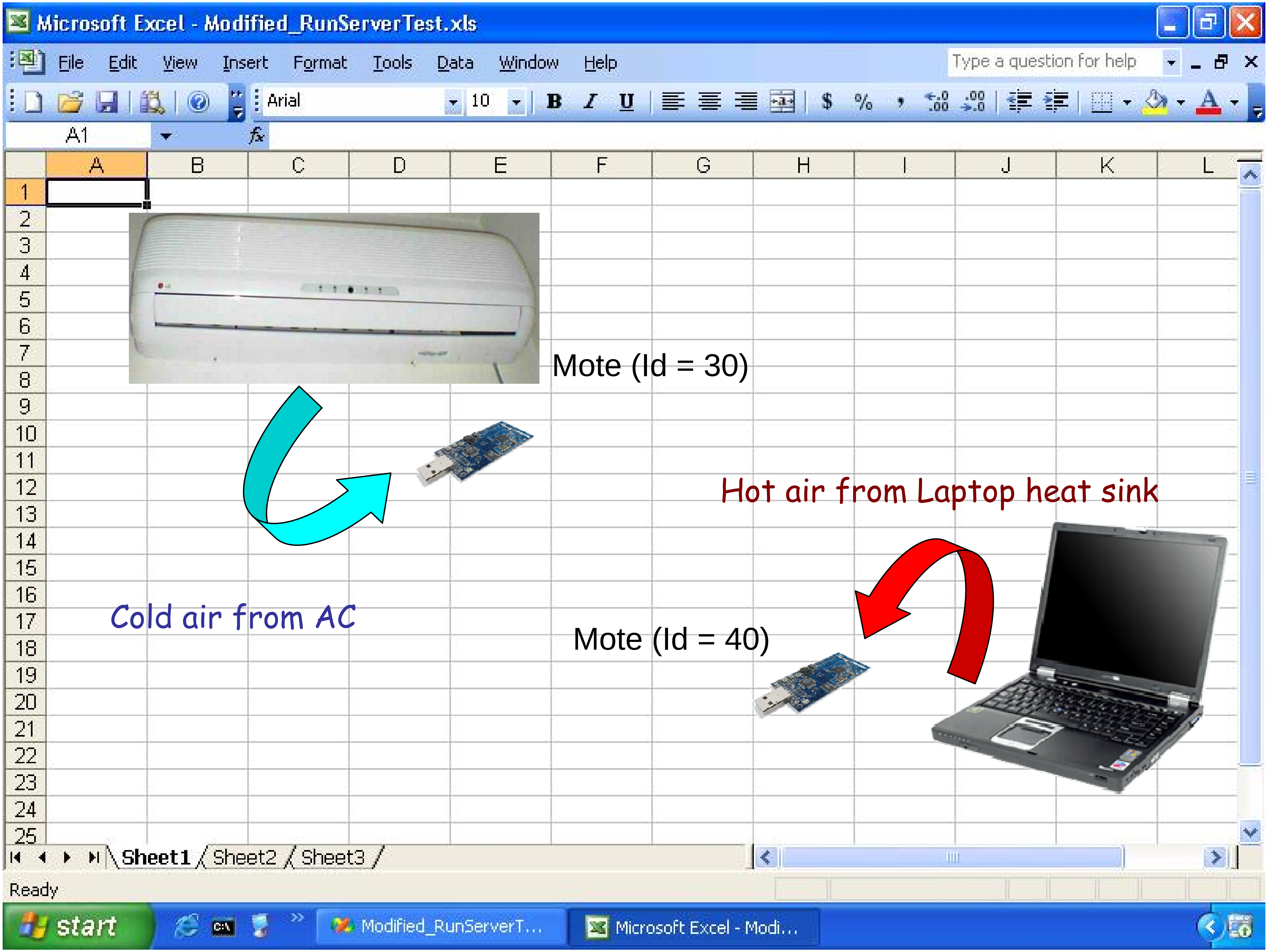
Visibility to systems



Source: Jeremy Elson

Data Collection




Microsoft Excel - Modified_RunServerTest.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Mote (Id = 30)
Hot air from Laptop heat sink
Cold air from AC
Mote (Id = 40)
Sheet1 Sheet2 Sheet3
Ready
start
Modified_RunServerT...
Microsoft Excel - Modi...

Microsoft Excel - Modified_RunServerTest.xls

= -39.60 + 0.01 * Raw Data

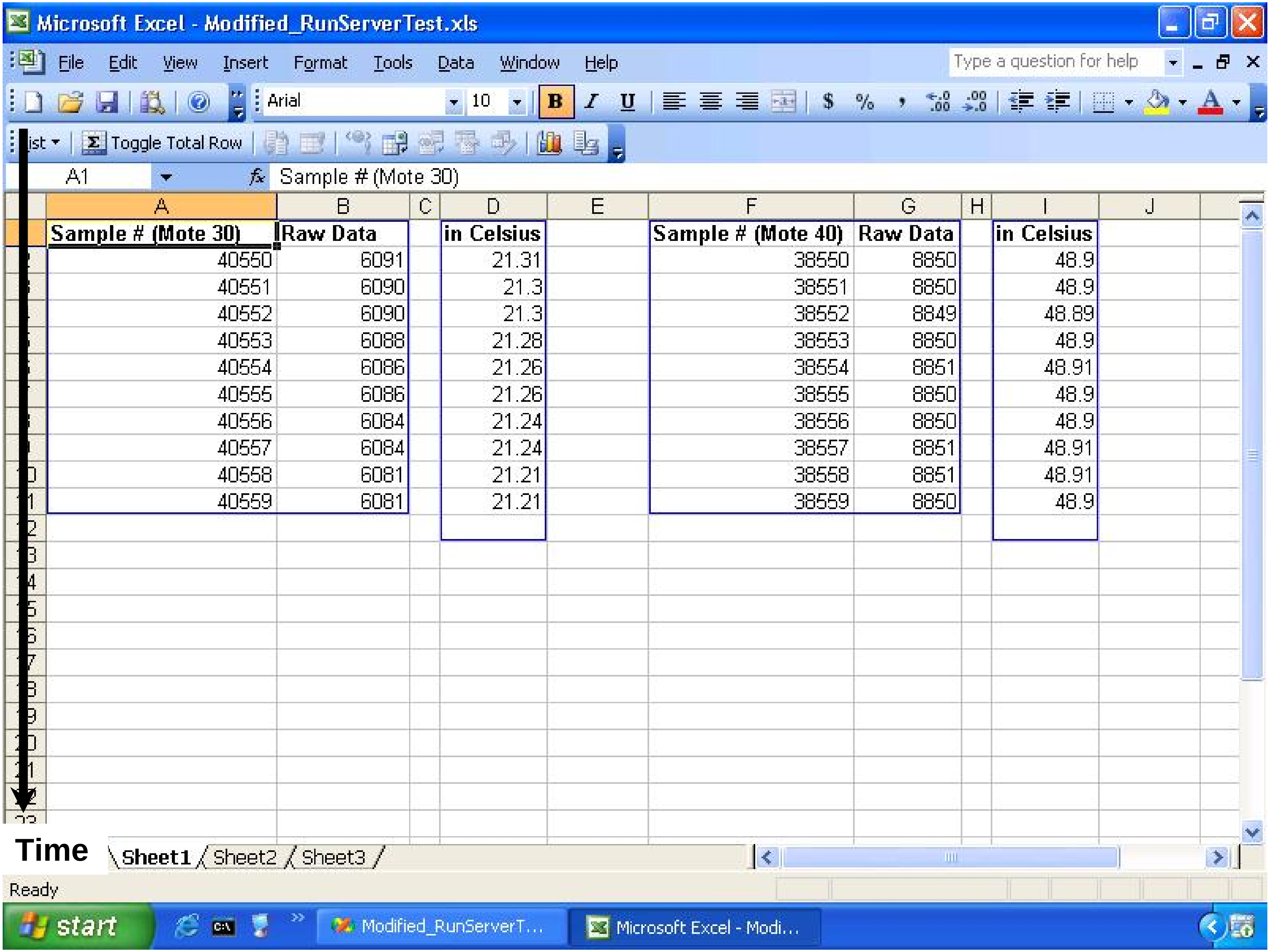
Sample # (Mote 30)	Raw Data	in Celsius
40550	6091	21.31
40551	6090	21.3
40552	6090	21.3
40553	6088	21.28
40554	6086	21.26
40555	6086	21.26
40556	6084	21.24
40557	6084	21.24
40558	6081	21.21
40559	6081	21.21


```
<OscopeMsg xmlns:xsd=
http://www.w3.org/2001/XMLSchema
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
  <sourceMoteID>30</sourceMoteID>
  <lastSampleNumber>40550
                </lastSampleNumber>
  <channel>1</channel>
  <data1>6091</data1>
  <data2>6090</data2>
  <data3>6090</data3>
  <data4>6088</data4>
  <data5>6086</data5>
  <data6>6086</data6>
  <data7>6084</data7>
  <data8>6084</data8>
  <data9>6081</data9>
  <data10>6081</data10>
</OscopeMsg>
```


Time

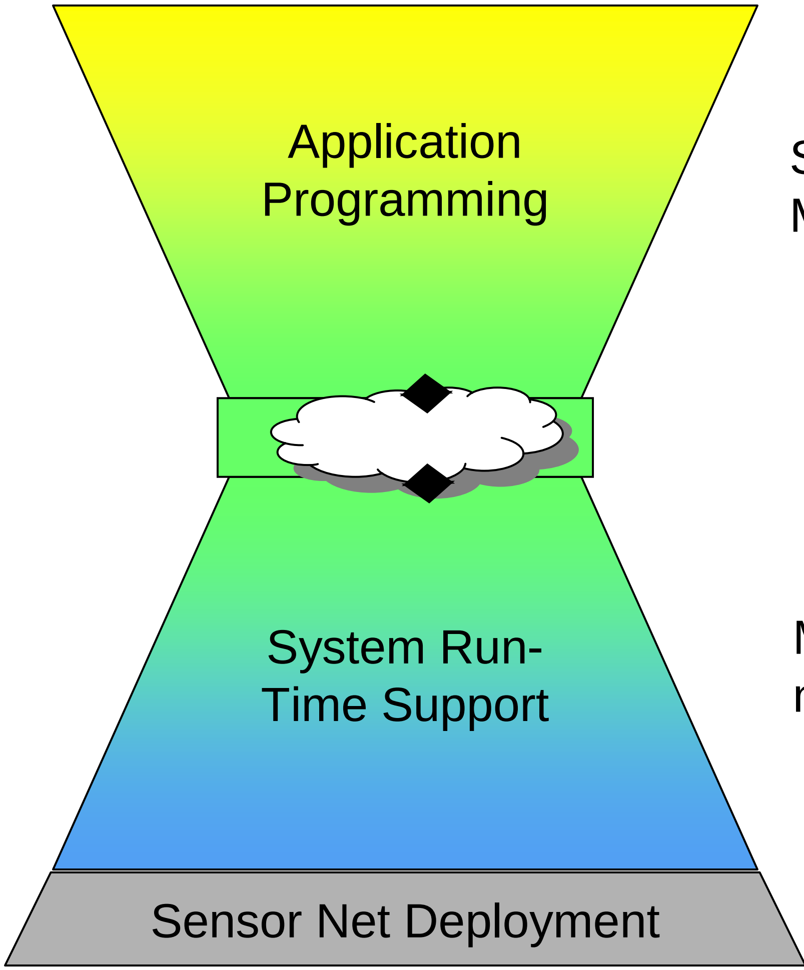
Sheet1 Sheet2 Sheet3

Ready



Sample # (Mote 30)	Raw Data	in Celsius	Sample # (Mote 40)	Raw Data	in Celsius
40550	6091	21.31	38550	8850	48.9
40551	6090	21.3	38551	8850	48.9
40552	6090	21.3	38552	8849	48.89
40553	6088	21.28	38553	8850	48.9
40554	6086	21.26	38554	8851	48.91
40555	6086	21.26	38555	8850	48.9
40556	6084	21.24	38556	8850	48.9
40557	6084	21.24	38557	8851	48.91
40558	6081	21.21	38558	8851	48.91
40559	6081	21.21	38559	8850	48.9

Narrow waist should allow applications to be specified independent of system configurations



Spreadsheet, logic/functional, visual, MATLAB, SQL

Mote programs, microserver programs, network protocols, …