Data-Driven Processing In Sensor Networks

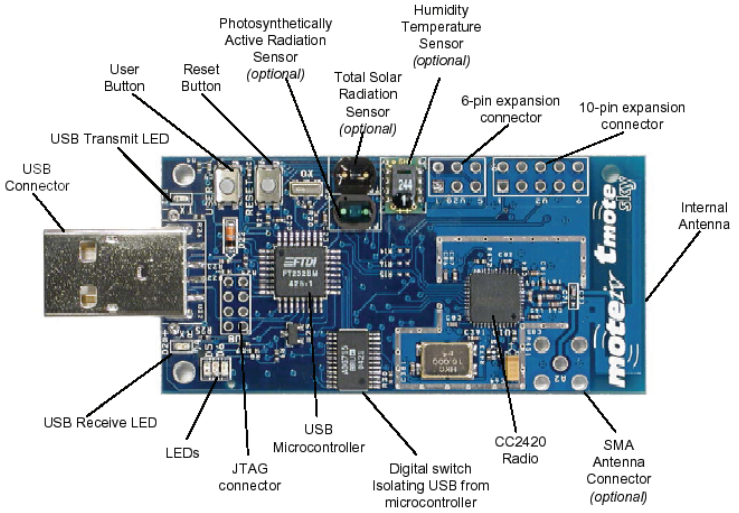
Jun Yang

Duke University

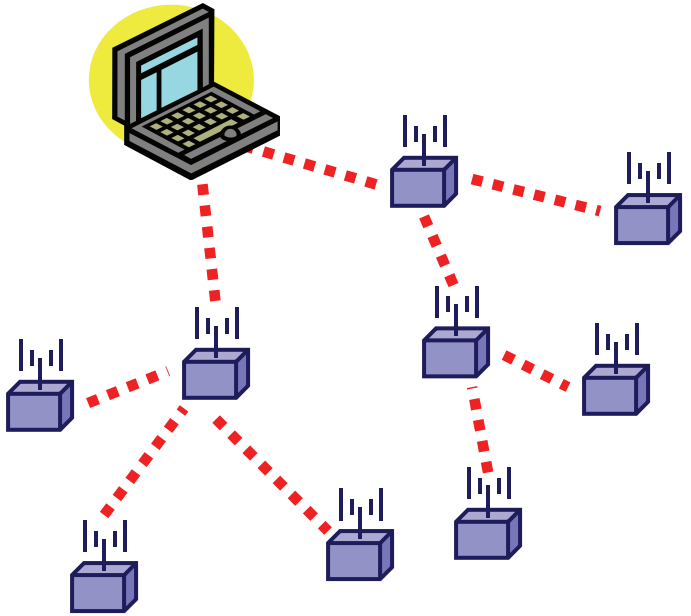
January 9, 2009

One possible type of sensor network

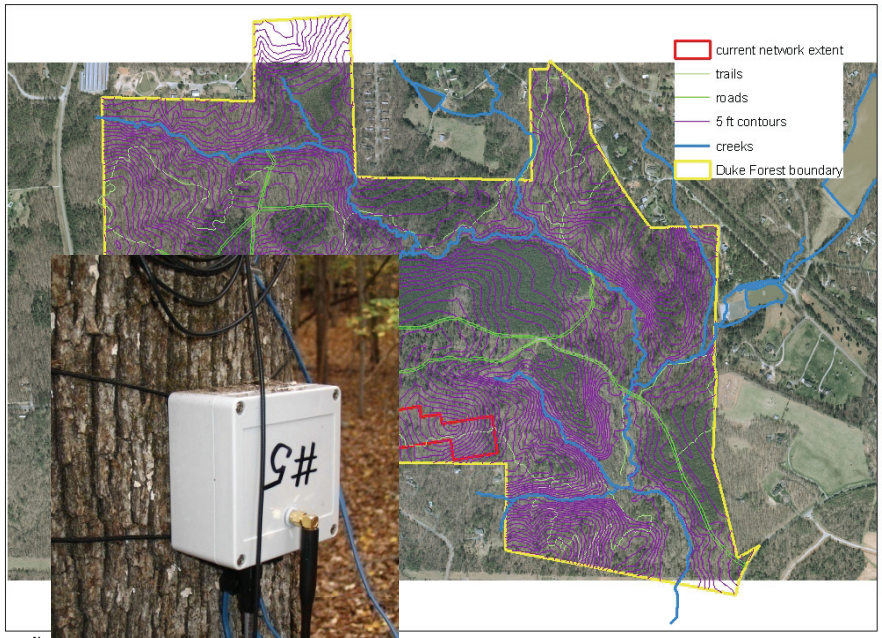
- Small, untethered nodes with severe resource constraints
  - Sensors, e.g., light, moisture, …
  - Tiny CPU and memory
  - Battery power
  - Limited-range radio communication
    - Often dominates energy consumption



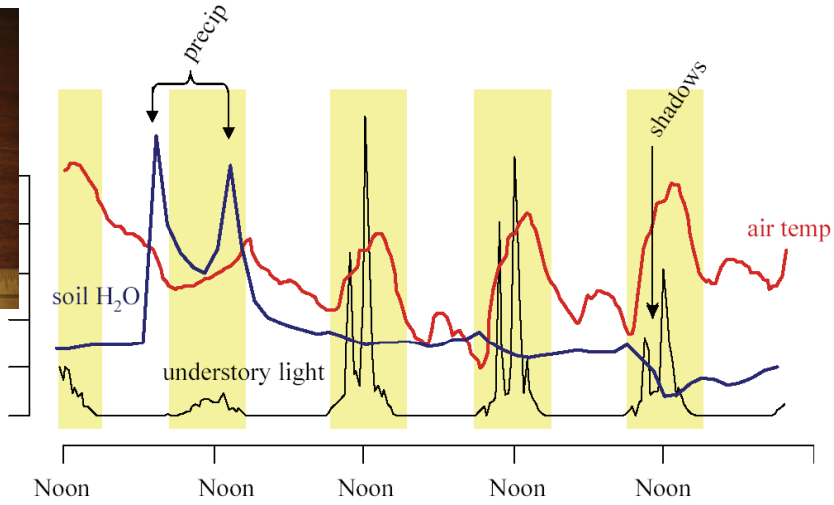
- Nodes form a multi-hop network rooted at a base station
  - Base station has plentiful resources and is typically tethered or at least solar-powered



Duke Forest deployment

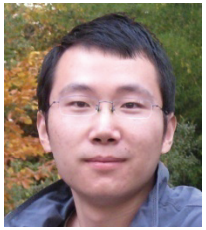


- Use wireless sensor networks to study how environment affects tree growth in Duke forest
  - Collaboration with Jim Clark (ecology) et al. since 2006



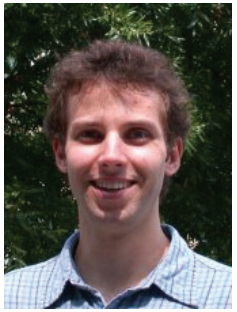
Acknowledgement






Adam Silberstein, Rebecca Braynard, Yi Zhang, Pankaj Agarwal, Carla Ellis, Kamesh Munagala (computer science)

Greg Filpus (undergrad)

Jim Clark, David Bell (ecology)



Alan Gelfand, Gavino Puggioni, Kristian Lum (statistics)

Paul Flikkema (EE, NAU)

National Science Foundation

So, what do ecologists want?

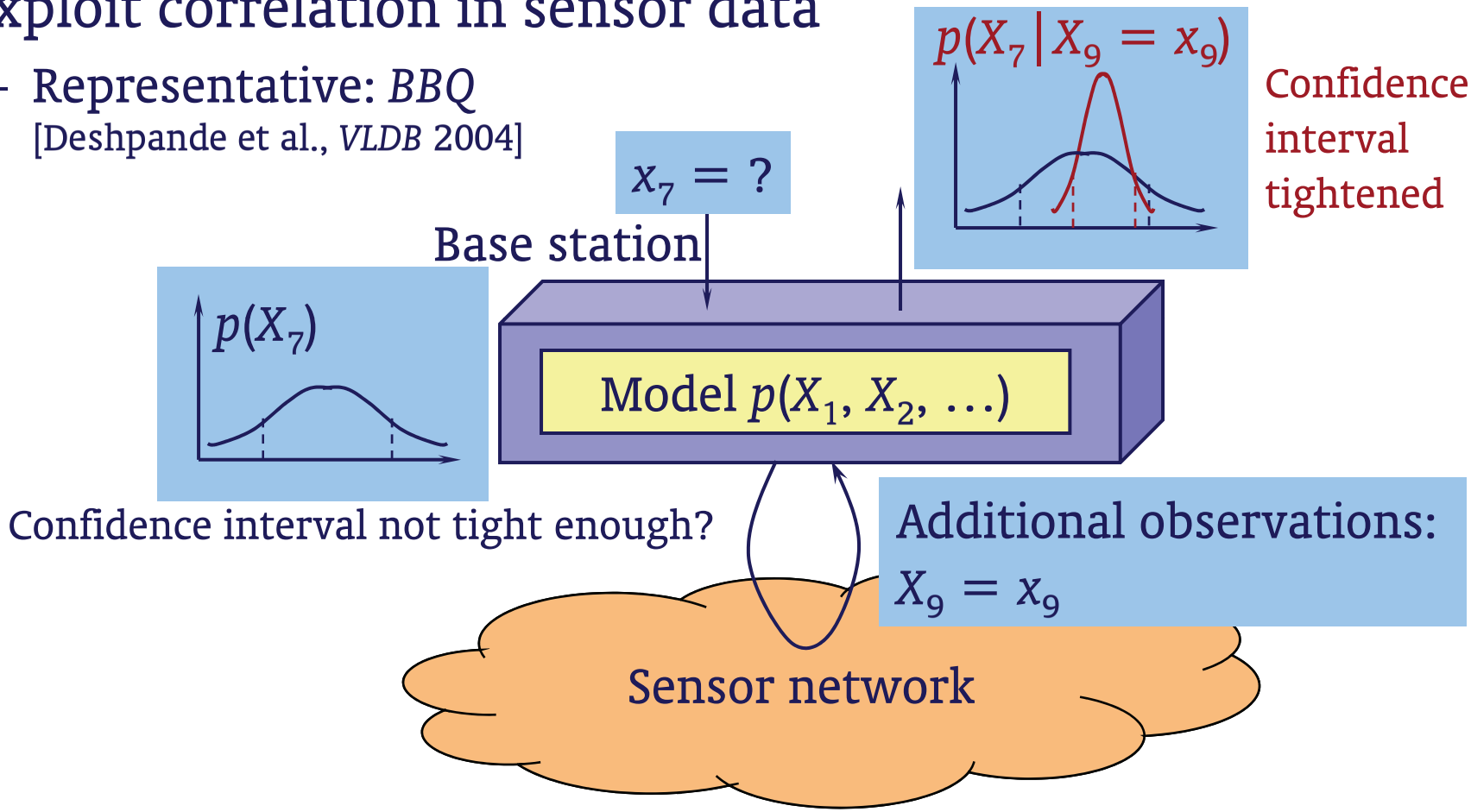
- Collect all data (to within some precision)
  - Probably the most boring database query
- Fit stochastic models using data collected
  - Cannot be expressed as database queries

☞ *Very different from how I, a database researcher, would think about "querying data"*

  - E.g., SQL, selection, join, aggregation…

Model-driven data collection: pull

- Exploit correlation in sensor data
  - Representative: *BBQ*
    [Deshpande et al., *VLDB* 2004]



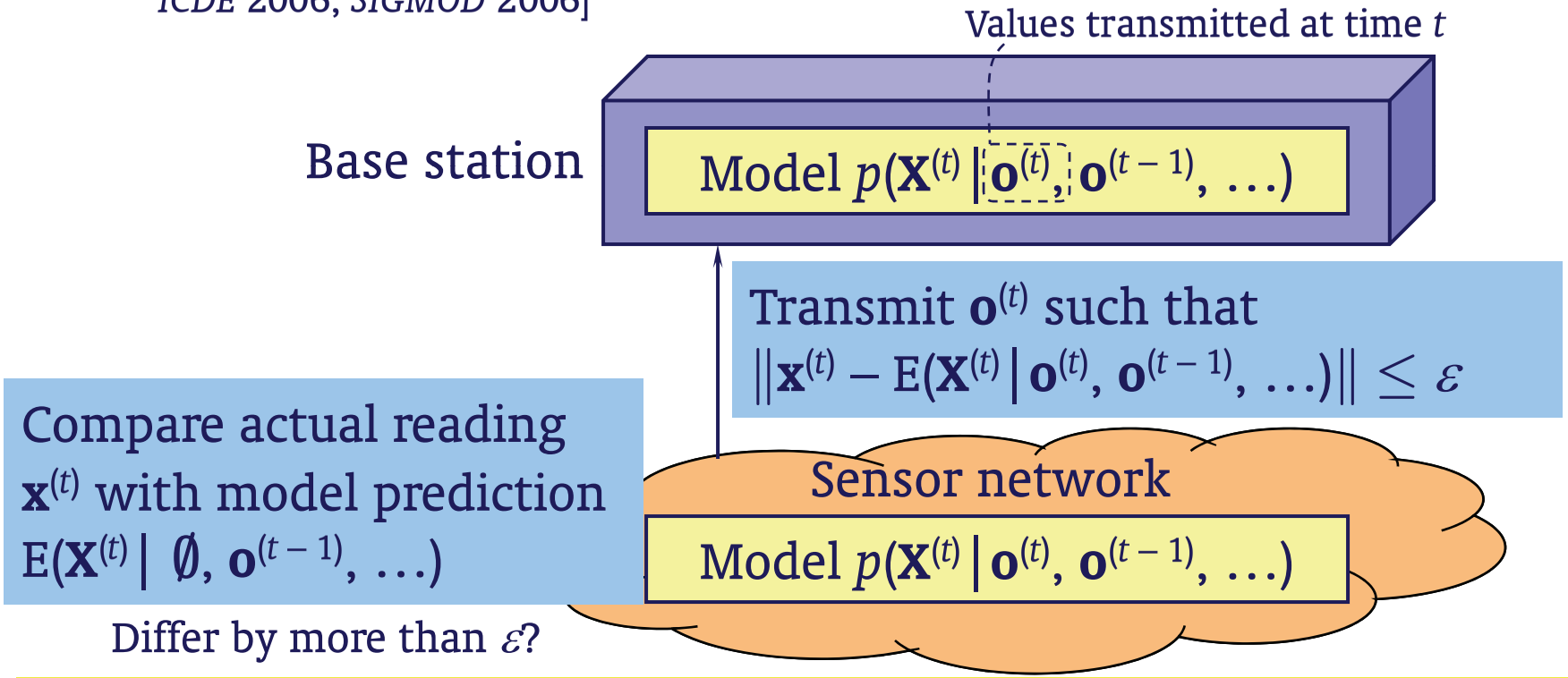
Answer correctness depends on model correctness
Risk missing the unexpected

Data-driven philosophy

- Models don't substitute for actual readings
  - Particularly when we are still *learning* about the physical process being monitored
  - Correctness of data collection should not rely on correctness of models
- Models can still be used to *optimize* collection

Data-driven: push

- Exploit correlation in data + put smarts in network
  - Representatives: *Ken* [Chu et al., *ICDE* 2006], *Conch* [Silberstein et al., *ICDE* 2006, *SIGMOD* 2006]



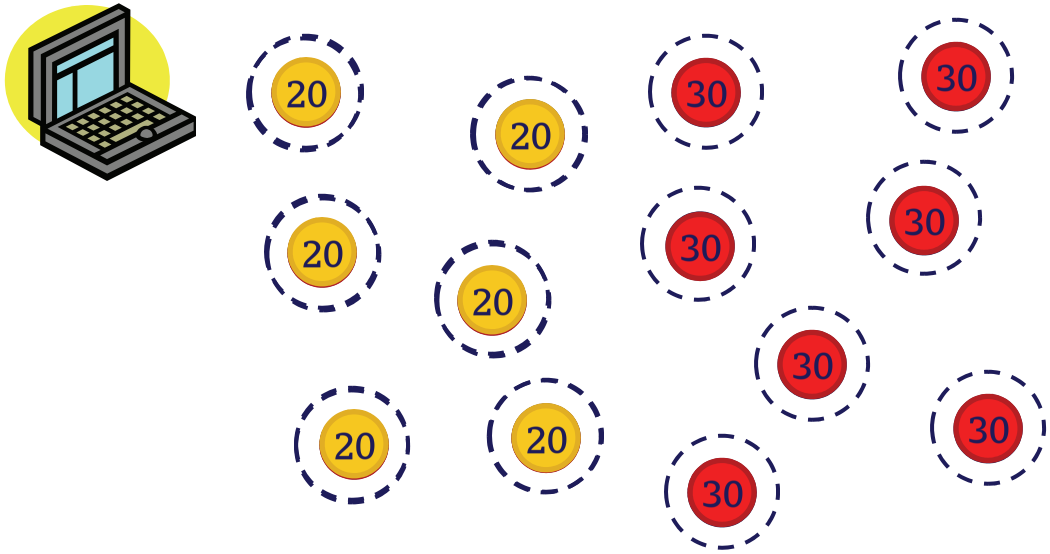
Regardless of model quality, base station knows $x^{(t)}$ to within ε
Better model ⇒ more "suppression" ⇒ fewer transmissions

Problem I:

The Quest for Suppression Schemes

Simple temporal suppression

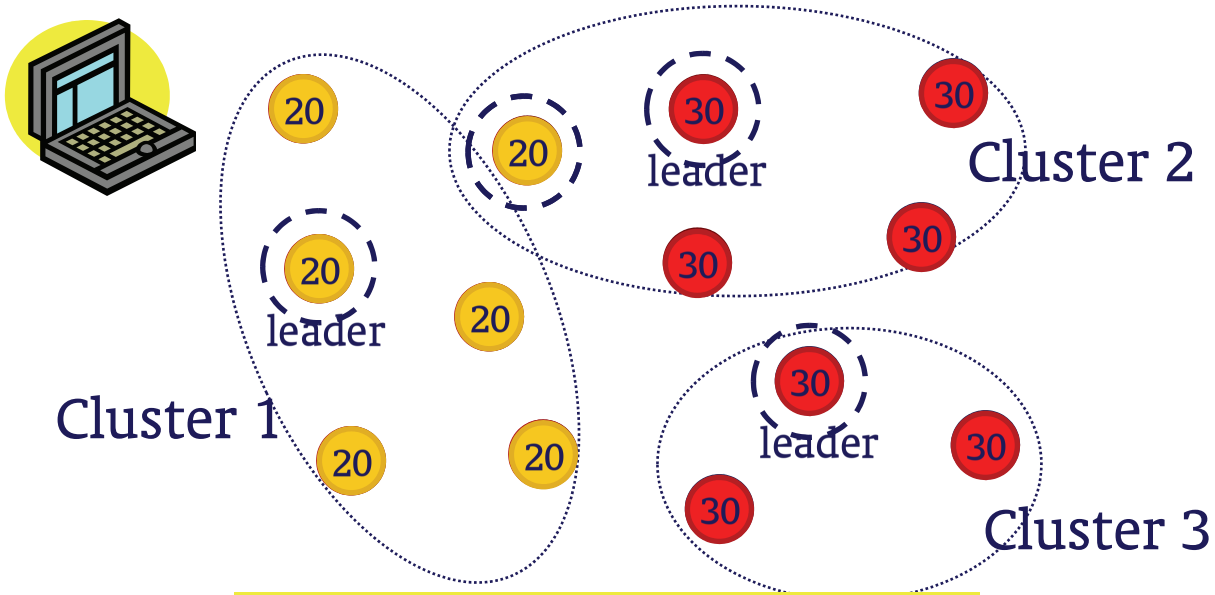
- Suppress transmission if
  $|\text{current reading} - \text{last transmitted reading}| \leq \varepsilon$
  - Model: $X^{(t)} = X^{(t-1)}$
- Effective when readings change slowly
- What about large-scale changes?



Phenomenon is simple to describe, but all nodes transmit!

Simple spatial suppression

- "Leader" nodes report for cluster
- Others suppress if
  $|\text{my reading} - \text{leader's reading}| \leq \varepsilon$
  - Model: $X_{\text{me}} = X_{\text{leader}}$
- Effective when nearby readings are similar



Leaders always transmit!

Combining spatial and temporal

Spatiotemporal suppression condition = ?

- Temporal **AND** spatial?
  - I.e., suppress if both suppression conditions are met
  - Results in less suppression than either!
- Temporal **OR** spatial?
  - I.e., suppress if either suppression condition is met
  - Base station cannot decide whether to set suppressed value to the previous value (temporal) or to the nearby value (spatial)!

Conch = constraint chaining

[Silberstein et al., *SIGMOD* 2006]

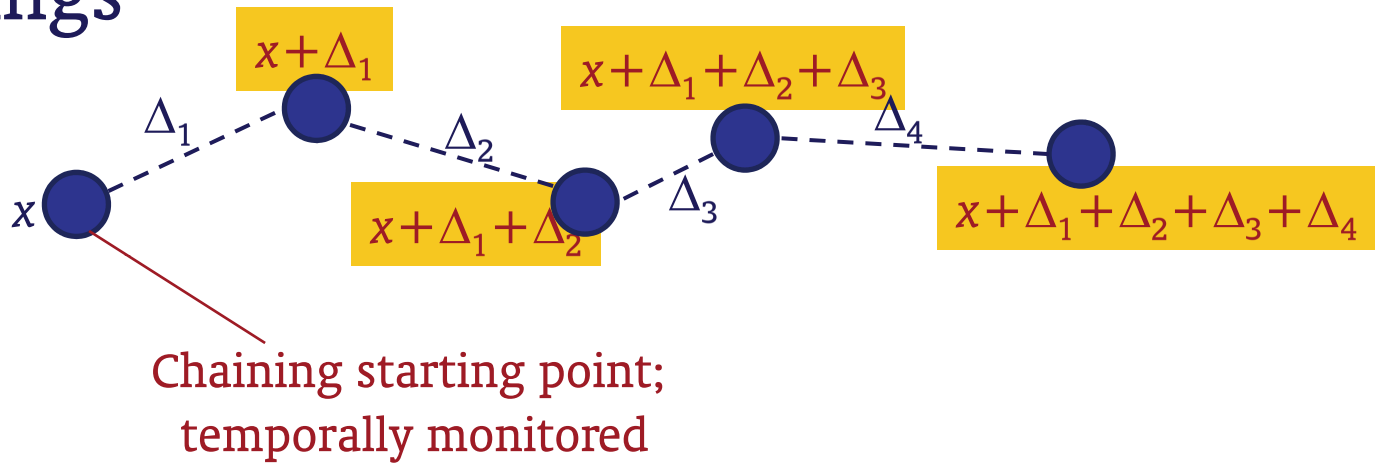
Temporally monitor spatial constraints (edges)

- x_i and x_j change in similar ways ⇒ temporally monitor edge difference $(x_i - x_j)$
  - "Difference" can be generalized

- One node is reporter and the other updater
  - Reporter tracks $(x_i - x_j)$ and transmits it to base station if its value changes
  - Updater transmits its value updates to reporter

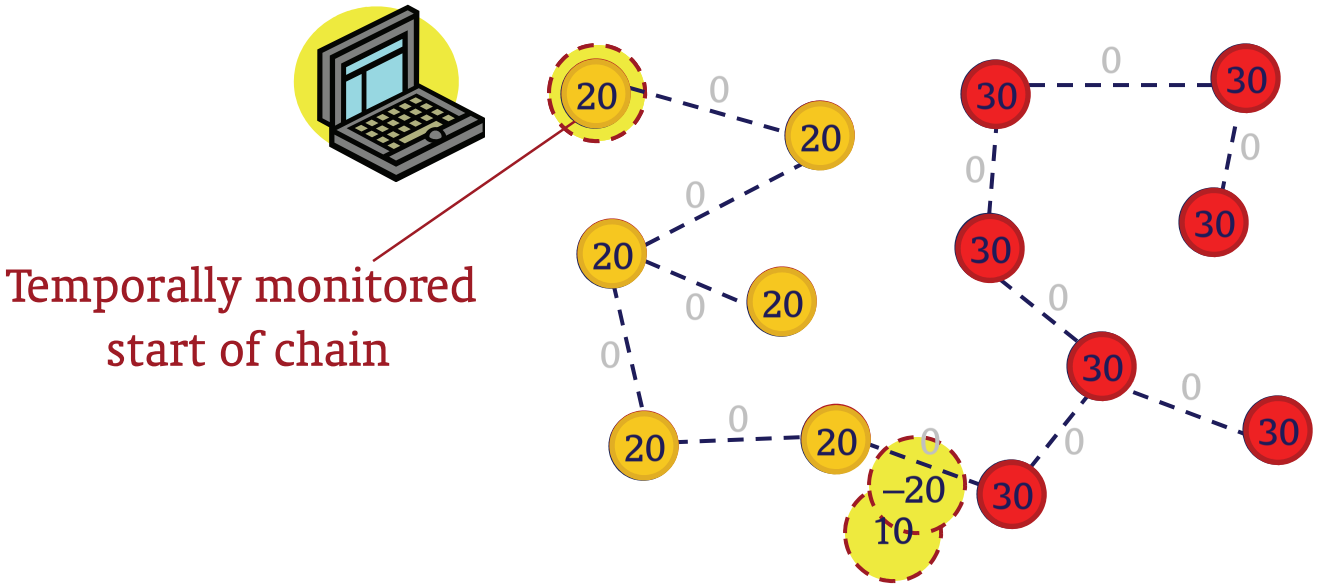
Recovering readings in *Conch*

- Base station "chains" monitored edges to recover readings



- Discretize values to avoid error stacking
  - $[k\varepsilon, k\varepsilon+\varepsilon) \rightarrow k$
  - Monitor discretized values exactly
    - Discretization is the only source of error
    - No error introduced by suppression

Conch example



Only "border" edges transmit to base station

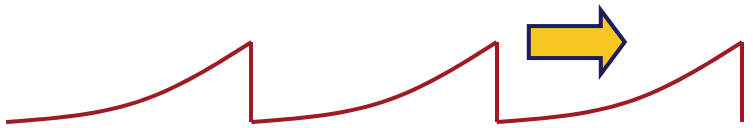
Combines advantages of both temporal and spatial suppression

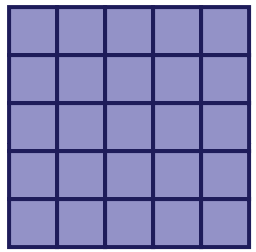
Choosing what to monitor

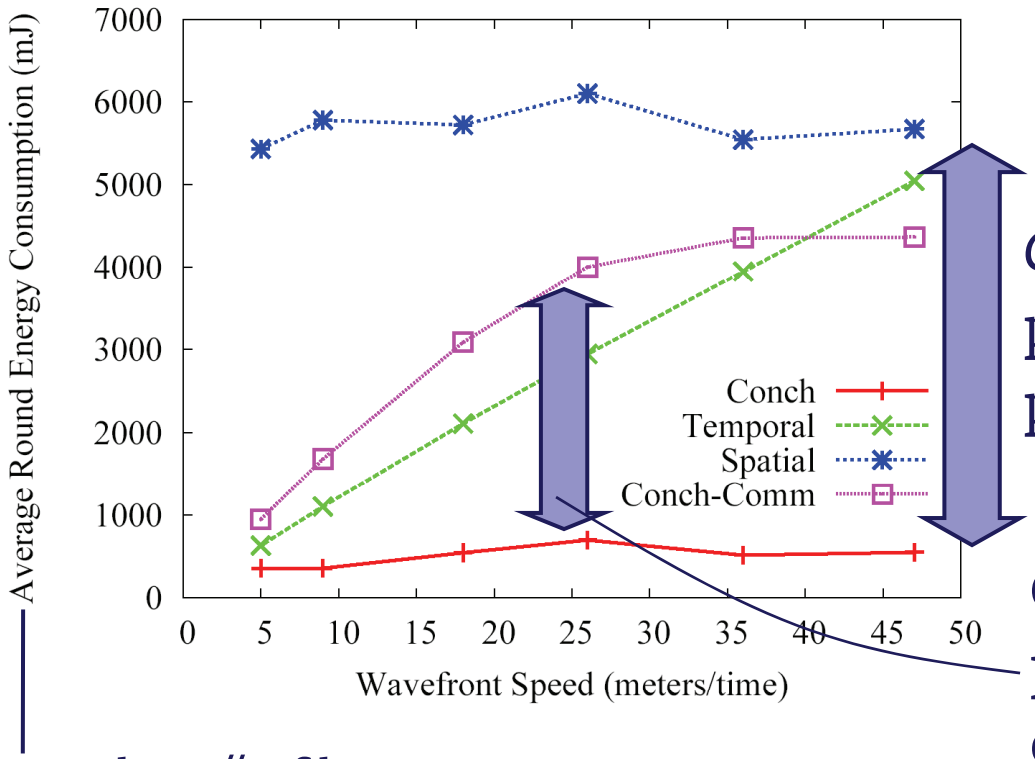
- A spanning forest is necessary and sufficient
  - Choose edges between correlated nodes
  - Do not connect erratic nodes
    - Just temporally monitor as singleton trees
- Observe
  - In pilot phase, use any spanning forest to collect data
    - Even a poor spanning forest correctly collects all data
- Optimize
  - Use collected data to assign monitoring costs
    - # of rounds in which monitored value changes
  - Build a min-cost spanning forest (e.g., *Prim's*)
- Re-optimize as needed

Wavefront experiment

- Simulate periodic vertical wavefronts moving across field
- Sensors at random grid points







Conch beats both pure temporal and pure spatial

Communication tree is poor for monitoring; optimization makes a huge difference

Based on # of bytes sent/received on Mica2 nodes

Summary of key ideas

- Observe and optimize
- *Cascaded suppression*: push/suppress not only between nodes and base station, but also among nodes themselves
- Strive for simplicity
  - Monitor locally—with cheap two-node spatial models
  - Infer globally—through chaining
- Vision for ideal suppression
  - Number of reports $\propto$ description complexity of phenomenon

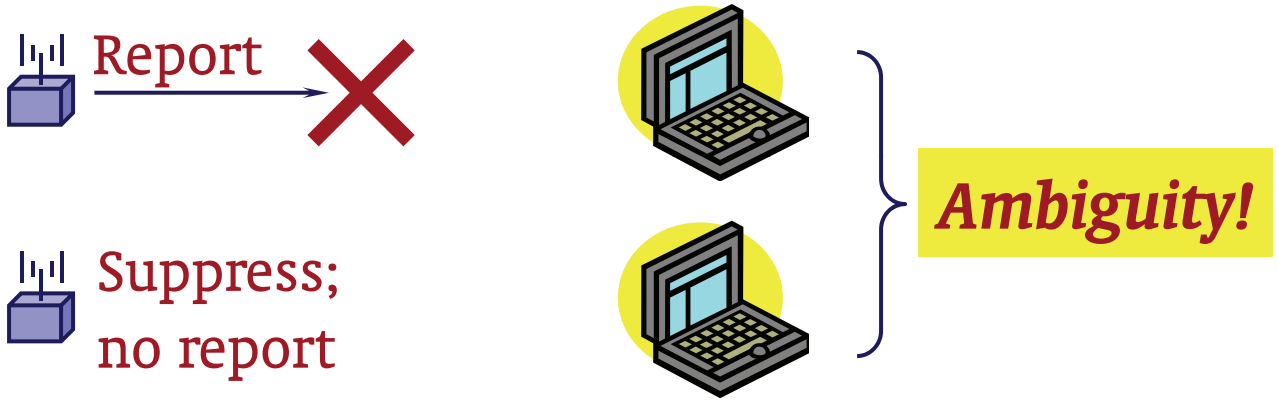
What's the catch?

Problem II:

Handling Uncertainty in Suppression

Failure and suppression

- Message failure common in sensor networks
  - Interference, obstacles, congestion, etc.



- Is a non-report due to suppression or failure?
  - Without additional information/assumption, base station has to treat every non-report as plain "missing"—no accuracy bounds!

A few previous approaches

- Avoid missing data: ACK/Retransmit
  - Often supported by the communication layer
  - Still no guaranteed delivery → does not help with resolving ambiguity
- Deal with missing data
  - Interpolation
    - Point estimates are often wrong or misleading
    - Uncertainty is lost—important in subsequent analysis/action
  - Use a model to predict missing data
    - Can provide distributions instead of point estimates
    - But we have to trust the model!

BayBase: basic Bayesian approach

- Model $p(\mathbf{X} \mid \Theta)$ with parameters Θ
  - Do not assume Θ is known
  - Any prior knowledge can be captured by $p(\Theta)$
- $\mathbf{x}_{obs}$: data received by base station
- Calculate posterior $p(\mathbf{X}_{mis}, \Theta \mid \mathbf{x}_{obs})$
  - Joint distribution instead of point estimates
  - Quantifies uncertainty in model; model can be improved

⇐ ***data-driven philosophy***

☞ Problem: non-reports are treated as generically missing

- But most of them are "engineered"
- Non-report ≠ no information!

How do we incorporate knowledge of suppression scheme?

BaySail

Bayesian Analysis of Suppression and Failure

[Silberstein et al., *VLDB* 2007]

- Bayesian, data-driven
- Add back some redundancy
- Infer with redundancy + knowledge of suppression scheme

Redundancy strikes back

At app level, piggyback redundancy on each report

- ***Counter***: number of reports to base station thus far

A good CS trick, but…

- ***Timestamps + Direction Bits***:
  last r timesteps when node reported +
  bits indicating whether each report is caused by
  (actual – predicted $> \varepsilon$) or
  (predicted – actual $> \varepsilon$)

Why?!

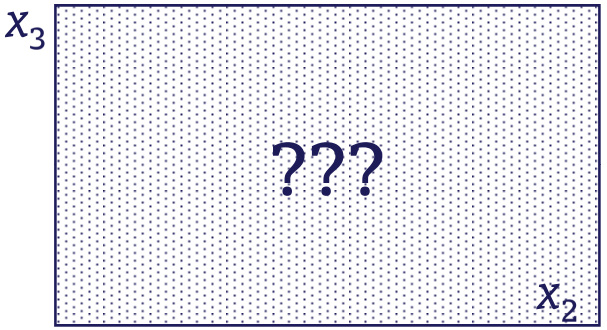
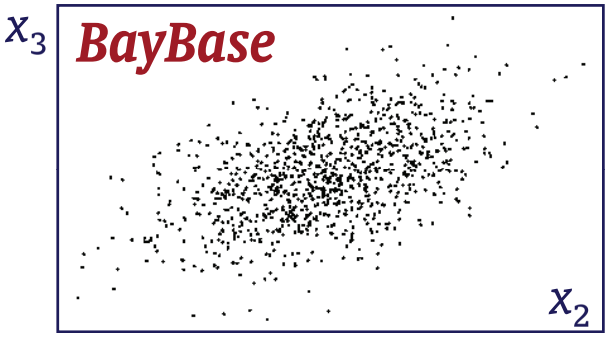
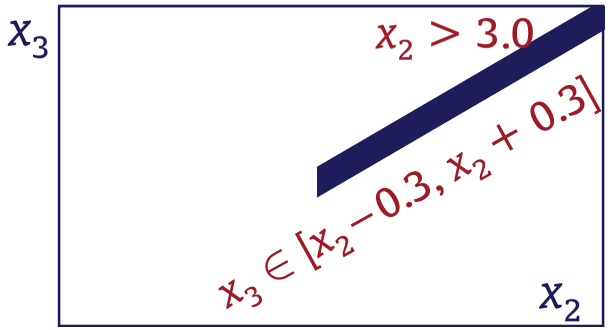
Suppression-aware inference

- Redundancy + knowledge of suppression scheme
  $\Rightarrow$ hard constraints on $\mathbf{X}_{mis}$

  - Temporal suppression: $\varepsilon = 0.3$, prediction = last reported
  - Actual: $(x_1,x_2,x_3,x_4) = (2.5/\text{sent}, 3.5/\text{sent}, 3.7/\text{suppressed}, 2.7/\text{sent})$
  - Base station receives: (2.5, nothing, nothing, 2.7)
  - With ***Timestamps+Direction Bits*** $(r=1)$
    - (2.5, failed & under-predicted, suppressed, 2.7 & over-predicted)
    - $x_2 - 2.5 > 0.3$; $-0.3 \le x_3 - x_2 \le 0.3$; $x_2 - 2.7 > 0.3$
  - With ***Counter***
    - One suppression and one failure in x_2 and x_3; not sure which
    - Hairy constraints!

- Posterior: $p(\mathbf{X}_{mis}, \Theta \mid \mathbf{x}_{obs})$, with $\mathbf{X}_{mis}$ subject to constraints

Benefit of modeling/redundancy

	Just data	Bayesian, model-based AR(1) with uncertain parameter
No knowledge of suppression		
Knowledge of suppression & ***Timestamps + Direction Bits***		

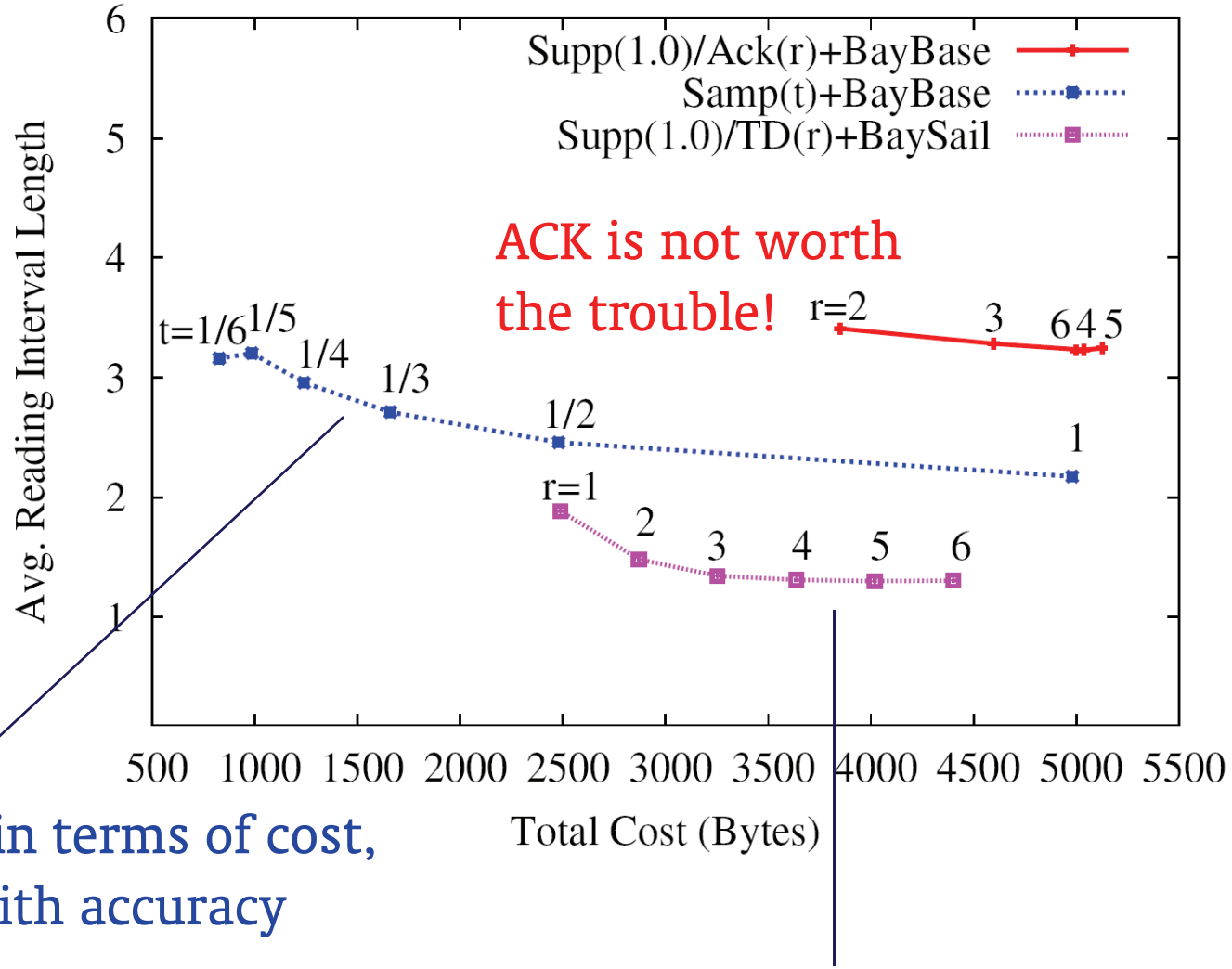
Inference

- Arbitrary distributions & constraints are difficult
  - Monte Carlo methods generally needed
  - Various optimizations apply under different conditions

- A simplified soil moisture model: $y_{s,t} = c_t + \phi\, y_{s,t-1} + e_{s,t}$
  - c_t is a series of known precipitation amounts
  - $\mathrm{Cov}(Y_{s,t}, Y_{s',t'}) = \sigma^2 \left(\phi^{|t-t'|}/(1-\phi^2)\right) \exp(-\tau \|s - s'\|)$
  - $\phi \in (0, 1)$ controls how fast moisture escapes soil
  - τ controls the strength of the spatial correlation over distance
- Given $\mathbf{y}_{\text{obs}}$, find $p(\mathbf{Y}_{\text{mis}}, \phi, \sigma^2, \tau \mid \mathbf{y}_{\text{obs}})$ subject to constraints
- Gibbs sampling
  - Markovian ⇒ okay to sample each cluster of missing values in turn
  - Gaussian + linear constraints ⇒ efficient sampling method
    - Timestamps + direction bits give us linear constraints!

Energy cost vs. inference quality

- Cost: bytes tx'd (incl. message overhead)
- Quality: size of 80% high-density region
- 30% message failure
- ≈ 60% suppression



Sampling is okay in terms of cost, but has trouble with accuracy

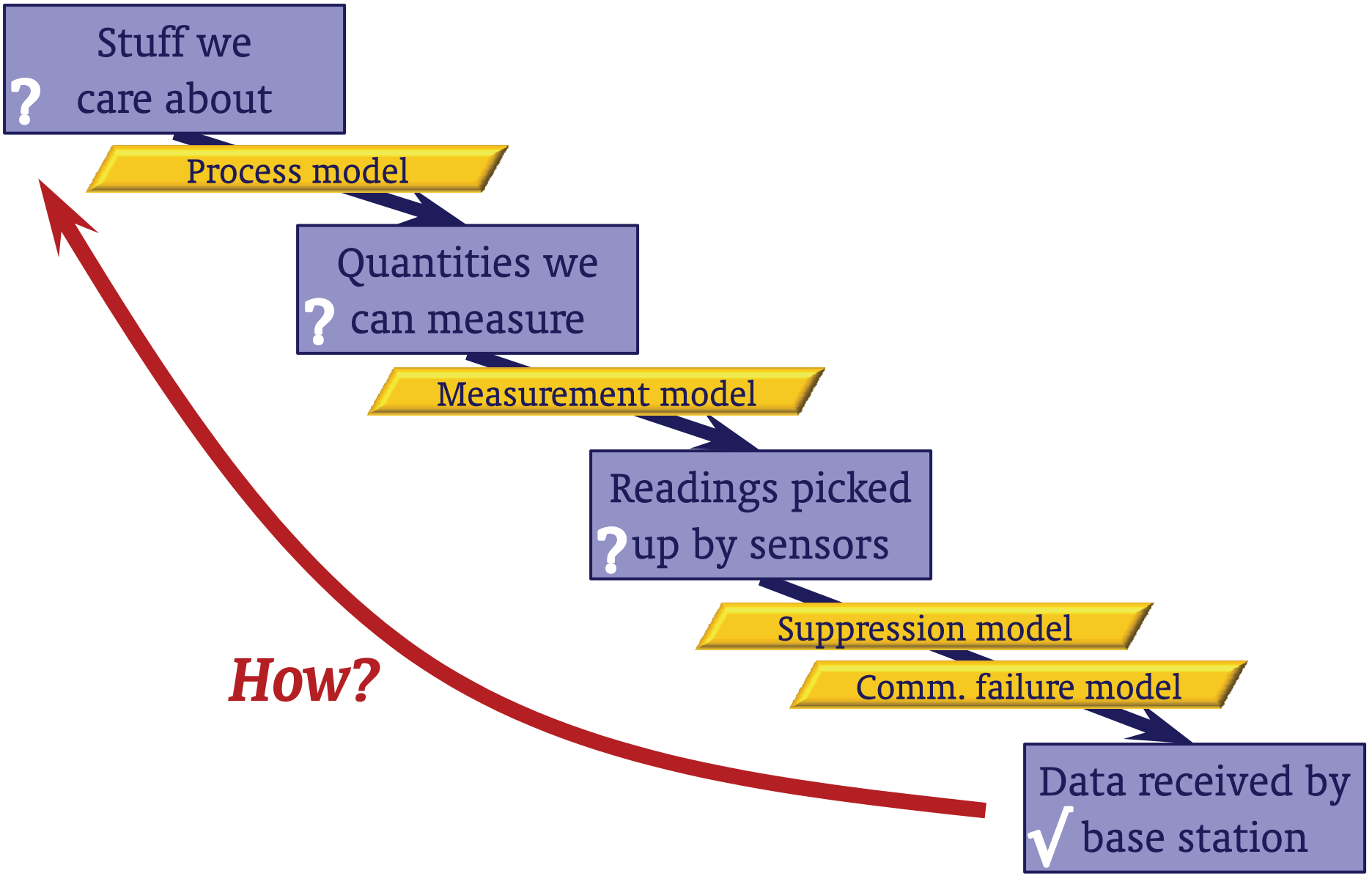
Suppression-aware inference with app-level redundancy is our best hope to get higher accuracy

Beyond *BaySail*

[Zhang et al., work in progress]

- Handling cascaded suppression
  - Receiver node needs to act on potentially incorrect knowledge
  - Resolve uncertainty and make corrections in network?
  - No! Send enough info to base station for final interpretation
- Many interesting future directions
  - Dynamic, local adjustments to ε and degree of redundancy
  - Ties to compression and information theory
  - Alternative/complementary to redundancy—model communication failures
    - Can do without redundancy
    - Can learn parameters of the failure model
    - But must trust its form

Model, model, everywhere



Concluding remarks

All models are wrong, but some models are useful

— George Box

- Data-driven approach
  - Use model to optimize, not to substitute for real data
    → suppression
  - Quantify uncertainty in models; use data to learn/refine
    → Bayesian

☞ ***Conch***: suppression by chaining simple spatiotemporal models

☞ ***BaySail***: suppression-aware inference with app-level redundancy to cope with failure

Food for thought for DB folks

Or, my personal wish list for an UDB/PDB:

- A data model that can represent arbitrary distributions
- Better support for provenance and metadata
  - E.g., what suppression scheme used during collection
  - Publishing received/interpolated data is not enough!
- Constraints also as first-class citizens
- Not only end-user SQL but also inference
- *Model/data independence*: same data can be interpreted by different models

Thanks!

http://www.cs.duke.edu/~junyang/