# STAR: Self-Tuning Aggregation for Scalable Monitoring

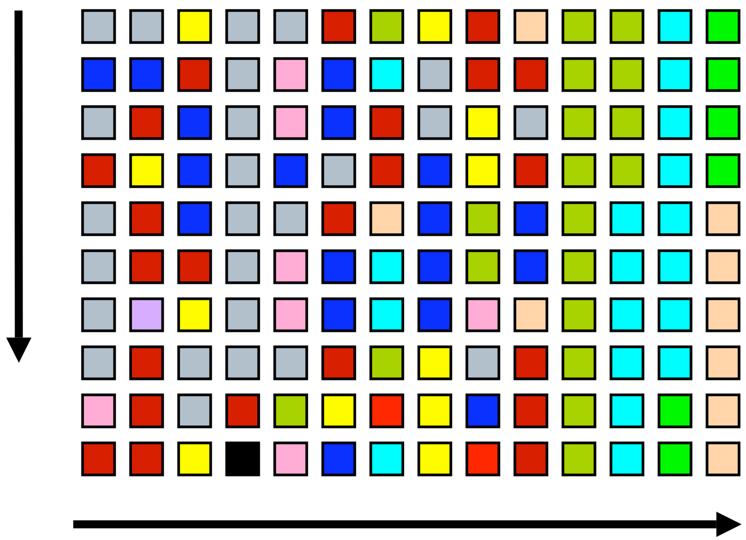
[On job market next year]

Navendu Jain, Dmitry Kit,
Prince Mahajan, Praveen Yalagandula†,
Mike Dahlin, and Yin Zhang

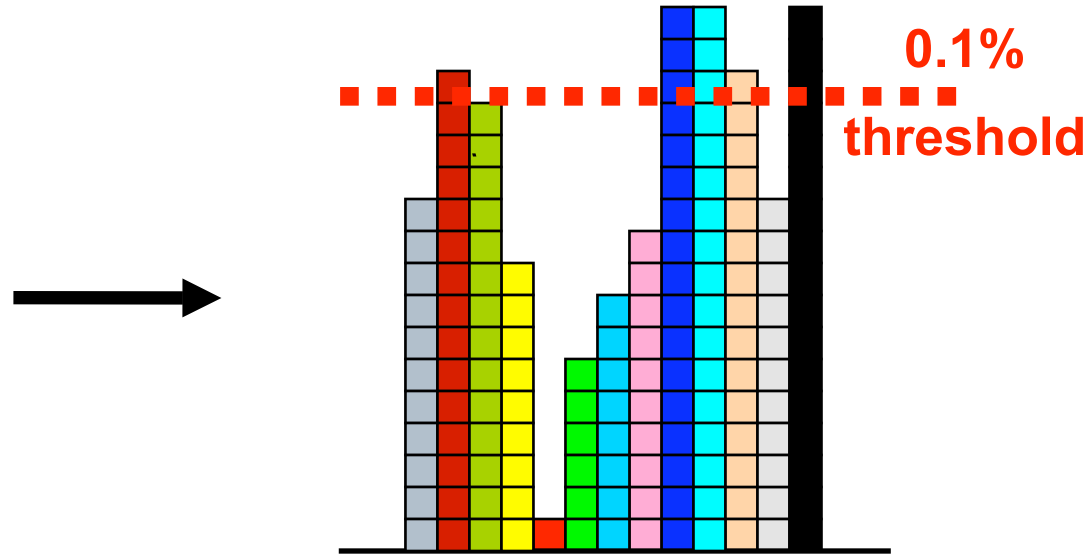
University of Texas at Austin
†HP Labs

# Motivating Application

- Network traffic monitoring: Detect Heavy Hitters



Traffic Stream

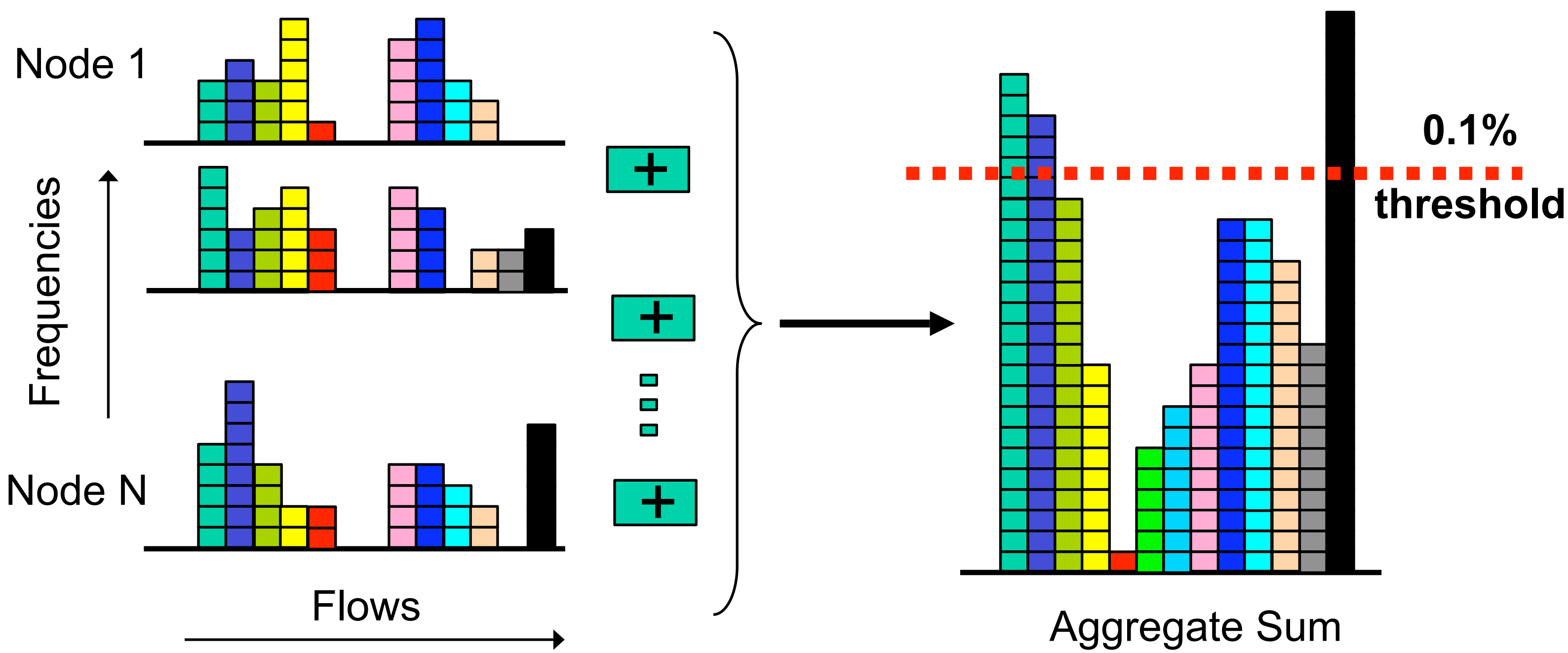



Frequency Counts

Identify flows that account for a significant fraction (say 0.1%) of the network traffic

# Global Heavy Hitters

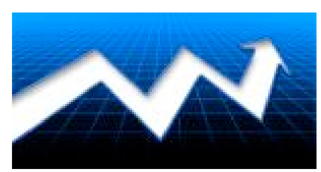
- Distributed Heavy Hitter detection
  - Monitor flows that account for a significant fraction of traffic across a collection of routers



# Broader Goal

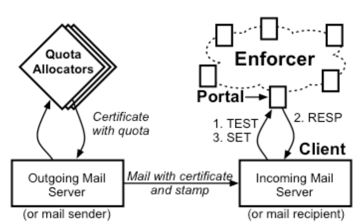
- Scalable Distributed Monitoring
  - Monitor, query, and react to changes in global state
    - Examples: Network monitoring, Grid monitoring, Job scheduling, Efficient Multicast, Distributed quota management, sensor monitoring and control, ...

Grids

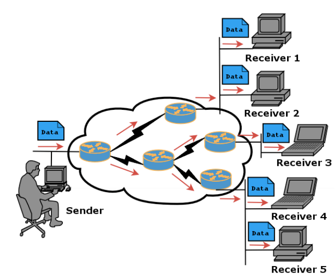


Financial apps

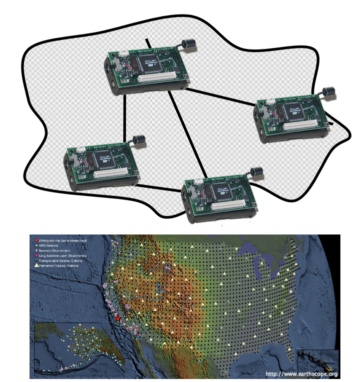

IP Traffic




Quota Management




Multicast



Sensor Networks

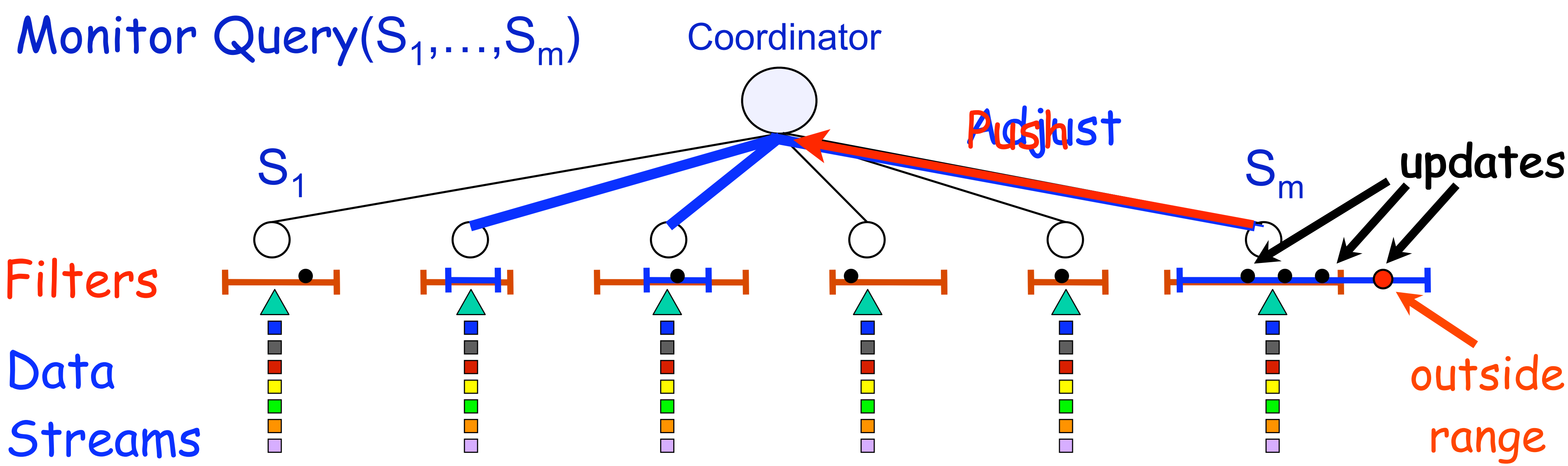
# System Model

__Key Challenges__

Large-scale: nodes, attributes (e.g., flows)

Robustness to dynamic workloads

Cost of adjustment



# Our Contribution: STAR

A scalable self-tuning algorithm to adaptively set the accuracy of aggregate query results

- Flexible precision-communication cost tradeoffs

Approach

- Aggregation Hierarchy
  - Split filters flexibly across leaves, internal nodes, root
- Workload-Aware Approach
  - Use variance, update rate to compute optimal filters
- Cost-Benefit Analysis
  - Throttle redistribution

# Talk Outline

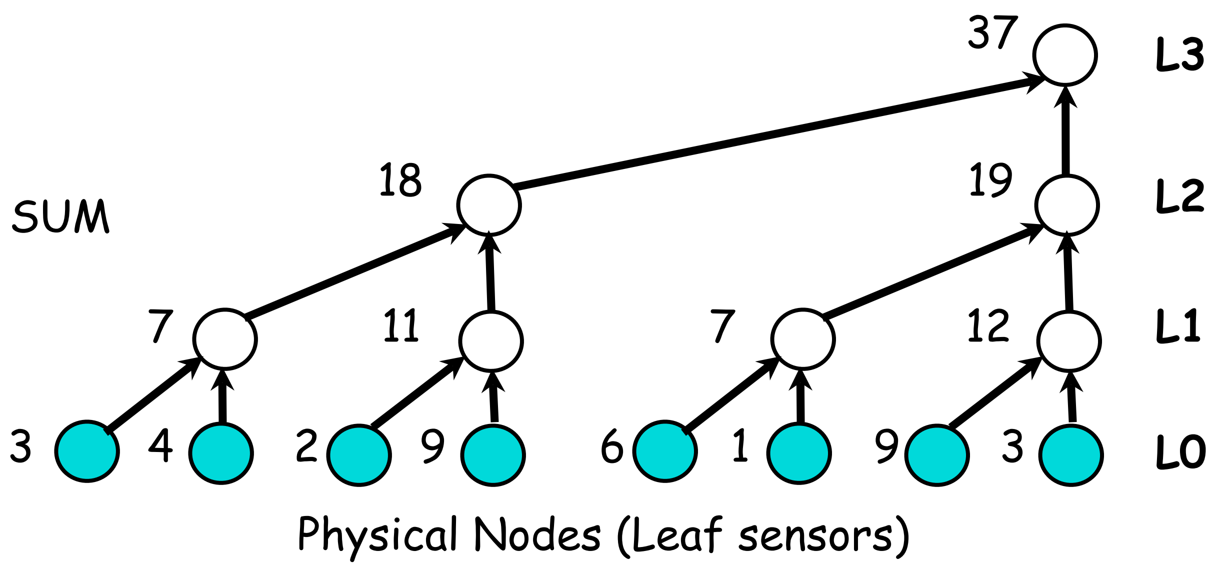
- Motivation
- STAR Design
  - Aggregation Hierarchy
  - Self-Tuning Filter Budgets
    - Estimate Optimal Budgets
    - Cost-Benefit Throttling
- Evaluation and Conclusions

# Background: Aggregation

PIER [Huebsch VLDB '03], SDIMS [Yalagandula SIGCOMM '04], Astrolabe [VanRenesse TOCS '03], TAG [Madden OSDI '02]

Fundamental abstraction for scalability

- Sum, count, avg, min, max, select, ...
- Summary view of global state
- Detailed view of nearby state and rare events

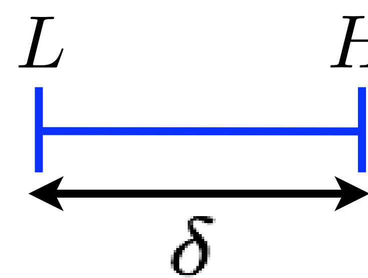


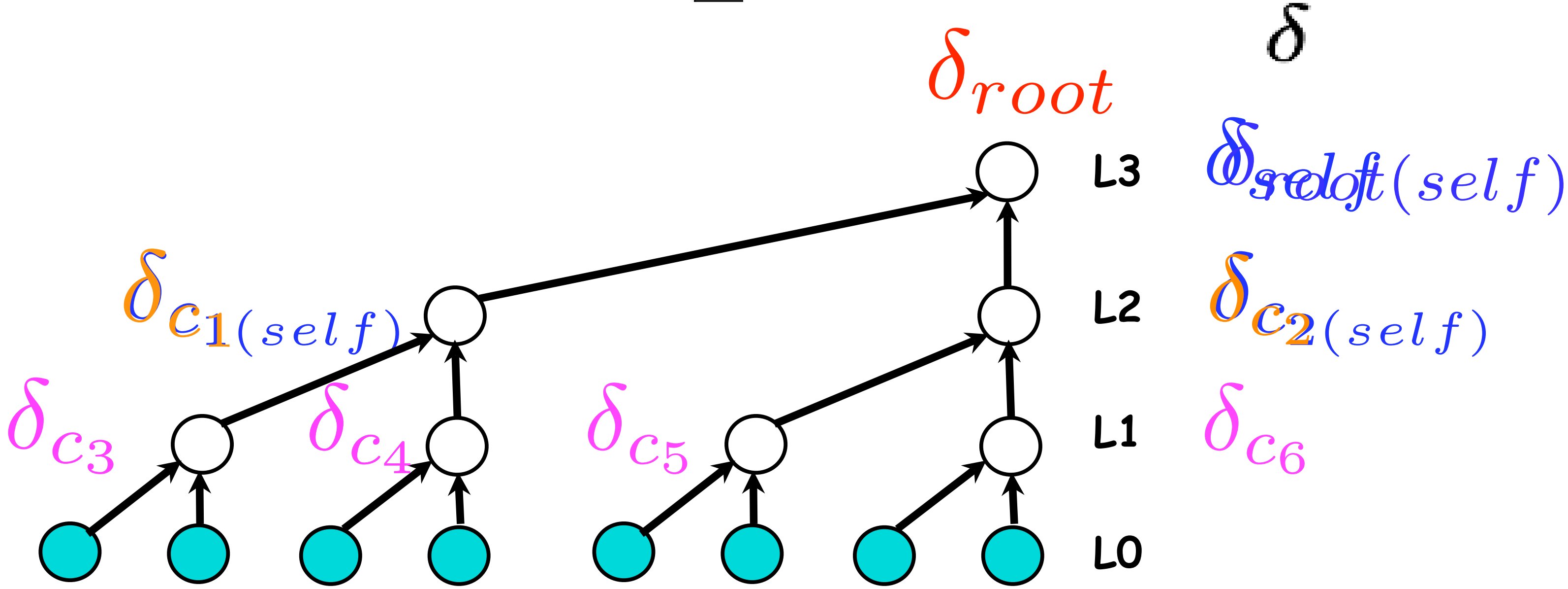
# Setting Filter Budgets

- Guarantees
  - Given an error budget $\delta$, report a range $[L, H]$ s.t.

(1) $$L \leq V_{actual} \leq H$$

(2) $$H - L \leq \delta$$



# Aggregation Hierarchy

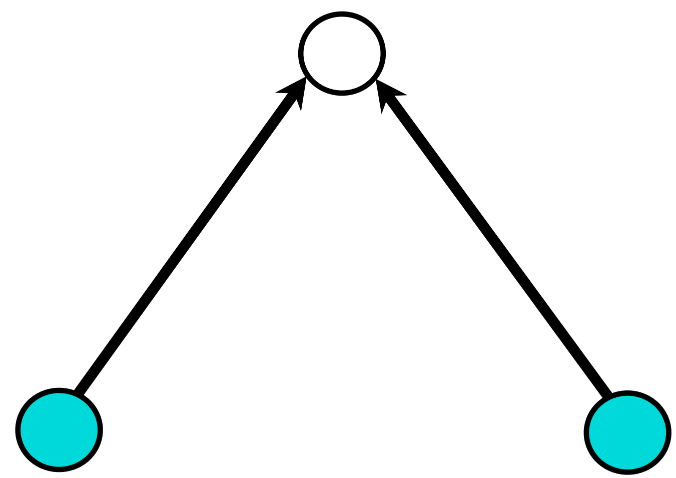
$$\delta_{root} = 5$$

[6,11] $\left[L - \frac{\delta^R_{self}}{2}, H + \frac{\delta^R_{self}}{2}\right]$

[4+3, 6+4] $\left[L, H\right] = \left[\sum_c L_c, \sum_c H_c\right]$

Node R

$\delta^R_{self} = 2$



Node A

$\delta^A_{self} = 2$

[4,6]

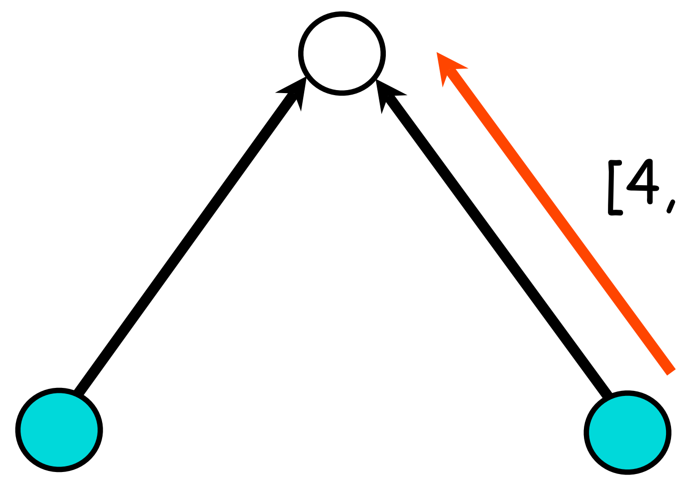
Node B

$\delta^B_{self} = 1$

[3,4]

# Aggregation Hierarchy

$$\delta_{root} = 5$$

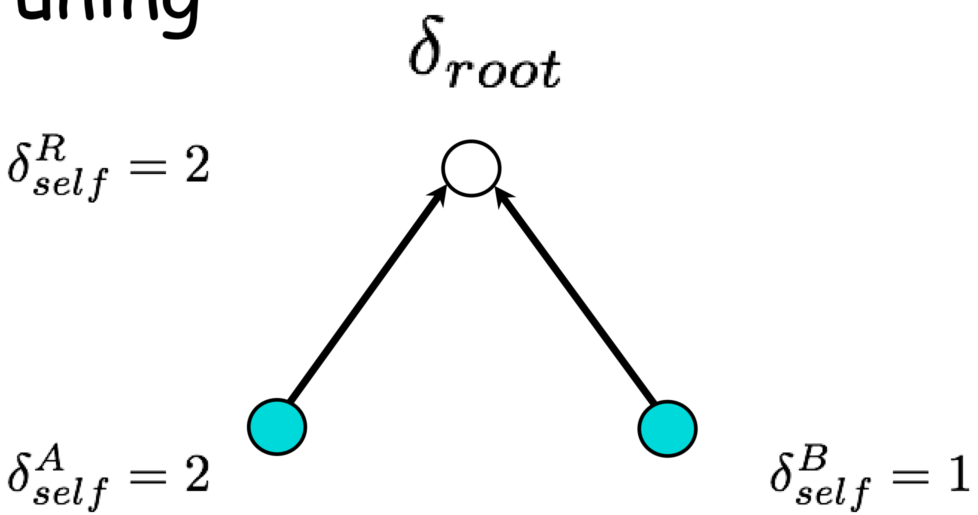


# Talk Outline

- [x] Motivation
- [ ] STAR Design
  - Aggregation Hierarchy
  - Self-Tuning Error Budgets
    - Estimate Optimal Budgets
    - Cost-Benefit Throttling
- [ ] Evaluation and Conclusions

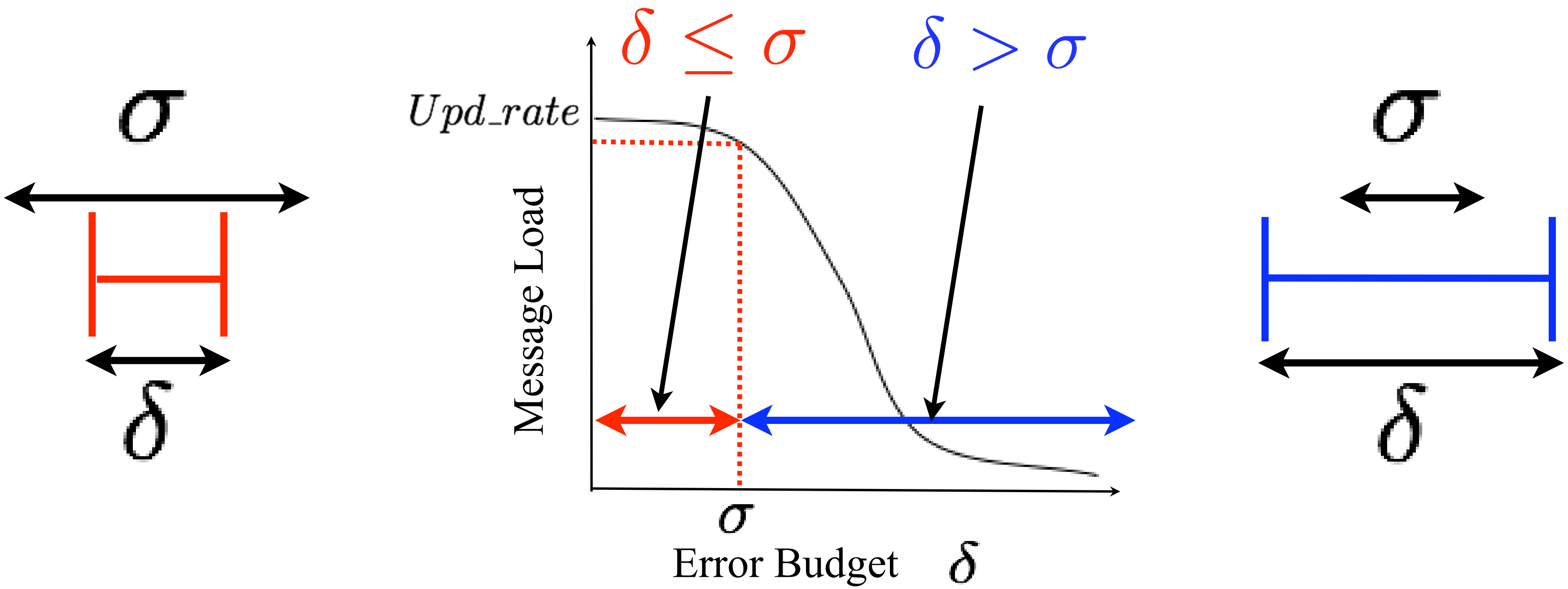
# How to Set Budgets?

Goal: Self-tuning




- Ideal distribution
  - Send budget to where filtering needed/effective
    - Large variance of inputs --> Require more budget to filter
    - Higher update rate of inputs --> Higher load to monitor

# Self-tuning Budgets: Single Node



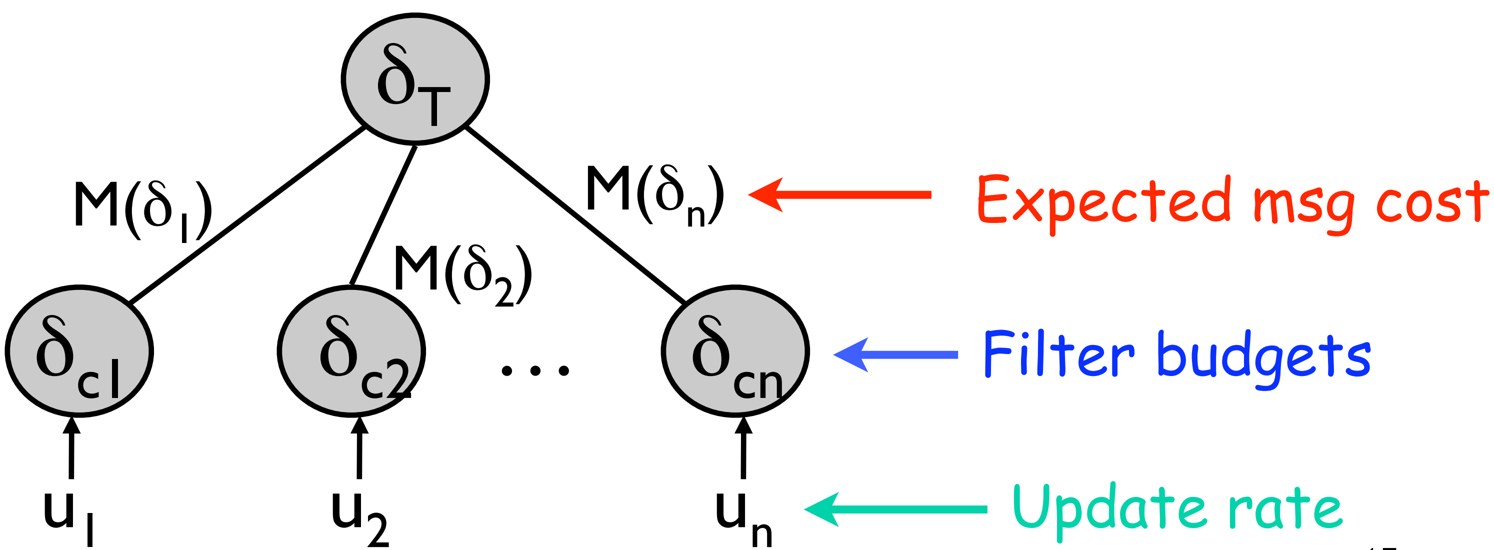

- Quantify filtering gain
  - Chebyshev's inequality
  - Expected message cost

$$M(\delta) = MIN\left(1, \frac{\sigma^2}{\delta^2}\right) * Upd\_rate$$

# Self-tuning Budgets: Hierarchy

- Single-level tree
  - Estimate optimal filter budget
    - Optimization problem: Min. msg cost under fixed budget
    - Solution:

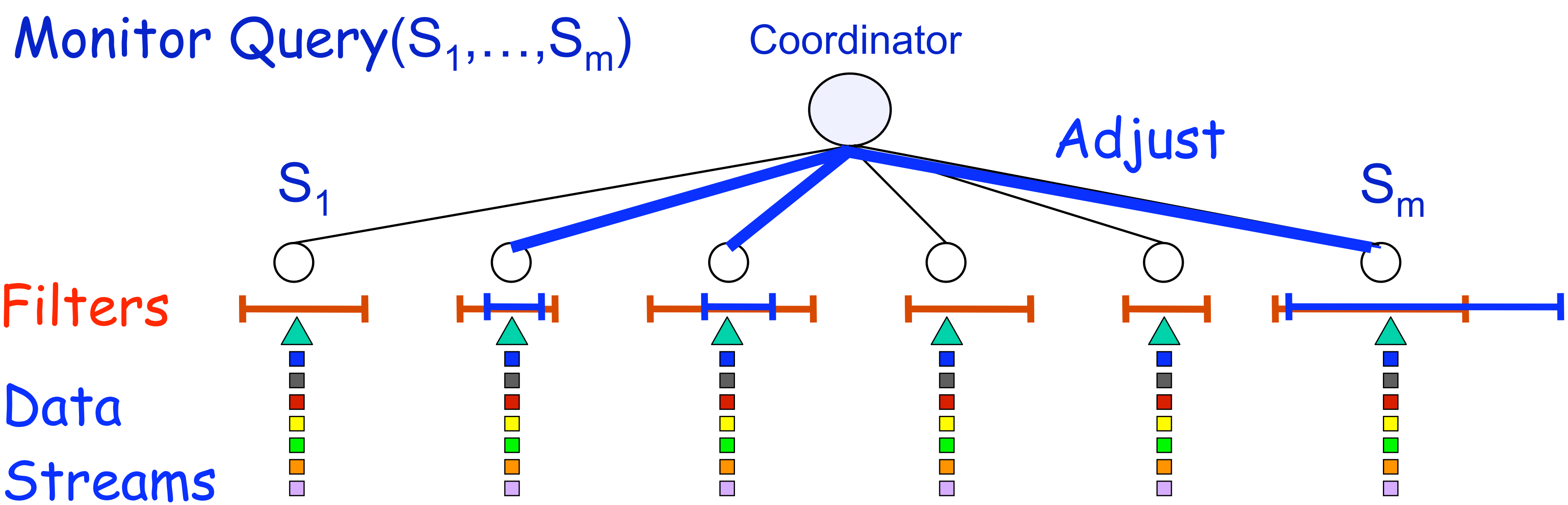
$$\delta_i^{opt} = \delta_T * \frac{\sqrt[3]{\sigma_i^2 * u_i}}{\sum_{i \in children} \sqrt[3]{\sigma_i^2 * u_i}}$$



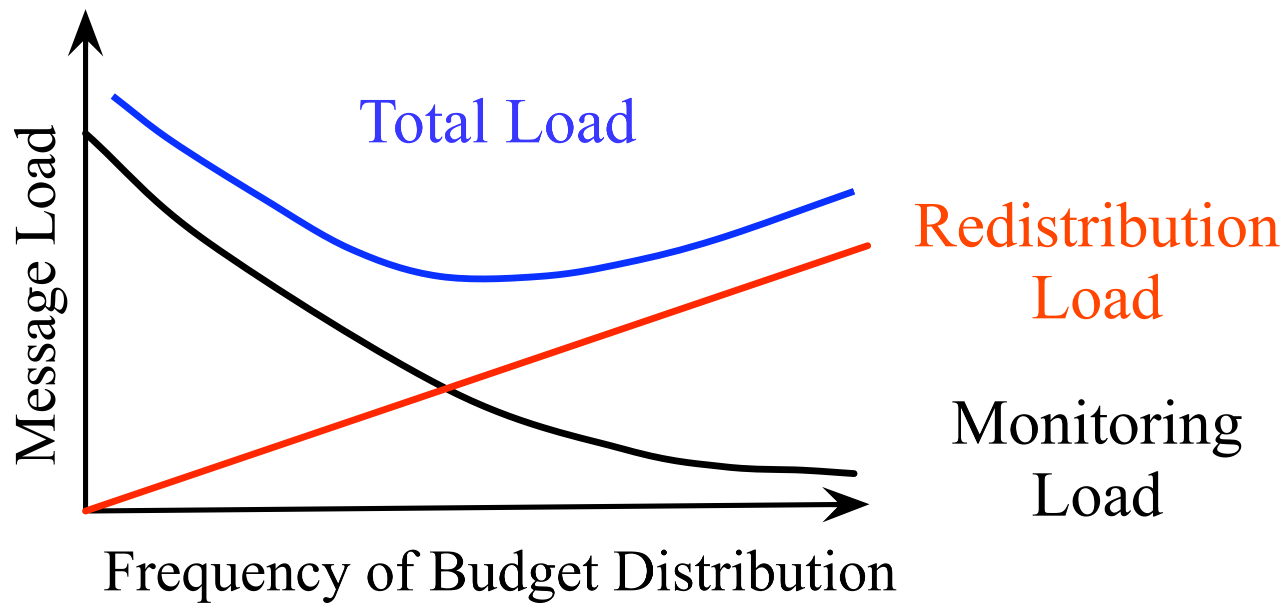
# Talk Outline

- Motivation
- STAR Design
  - Aggregation Hierarchy
  - Self-Tuning Filter Budgets
    - Estimate Optimal Budgets
    - Cost-Benefit Throttling
- Evaluation and Conclusions

# Redistribution Cost



# When to Redistribute Budgets?




- More frequent redistribution
  - More closely approx. ideal distribution (current load)
  - Heavier redistribution overhead

# Cost-Benefit Throttling

1. Load Imbalance

$M(\delta_{current}) - M(\delta_{ideal})$

2. Long-lasting Imbalance

$T_{current} - T_{time\_last\_redist}$

Charge: $(M(\delta_{current}) - M(\delta_{ideal})) * (T_{current} - T_{time\_last\_redist})$

Rebalance if Charge > Threshold

# Talk Outline

- ☑ Motivation
- ☑ STAR Design
  - Aggregation Hierarchy
  - Self-Tuning Filter Budgets
    - Estimate Optimal Budgets
    - Cost-Benefit Throttling
- ☐ Evaluation and Conclusions

# Experimental Evaluation

STAR prototype

- Built on top of SDIMS aggregation [Yalagandula '04]
- FreePastry as the underlying DHT [Rice Univ./MPI]
- Testbeds
  - CS Department, Emulab, and PlanetLab

Questions

- Does arithmetic approximation reduce load?
- Does self-tuning yield benefits and approximate ideal?

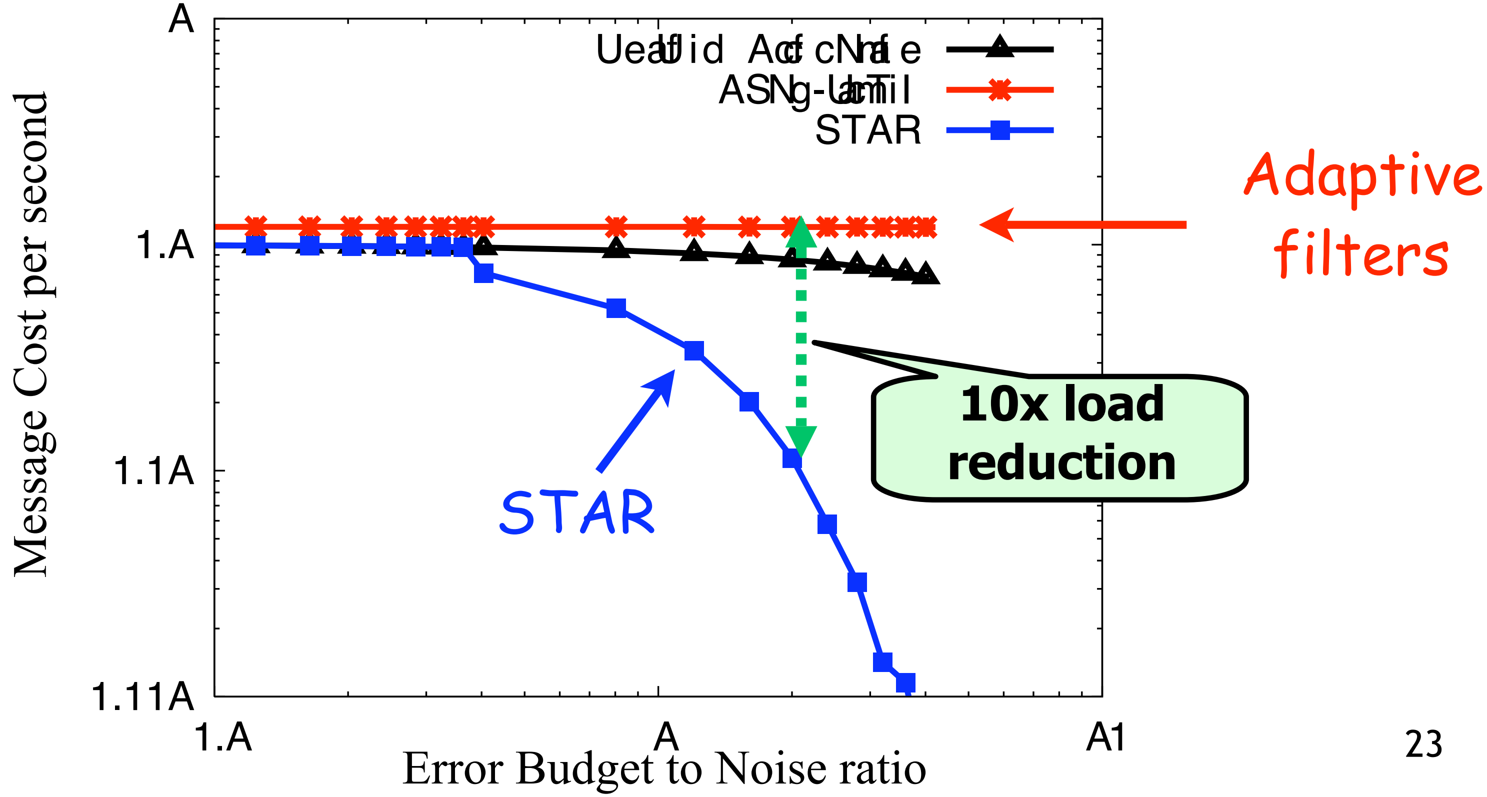
# Methodology

- Simulations
  - Quantify load reduction due to self-tuning budgets under varying workload distributions
- App:Distributed Heavy Hitter detection (DHH)
  - Find top-100 destination IPs receiving highest traffic
  - Abilene traces for 1 hour (3 routers); 120 nodes
    - Netflow data logged every 5 minutes

# Does Throttling Redistribution Benefit?

## 90/10 synthetic workload

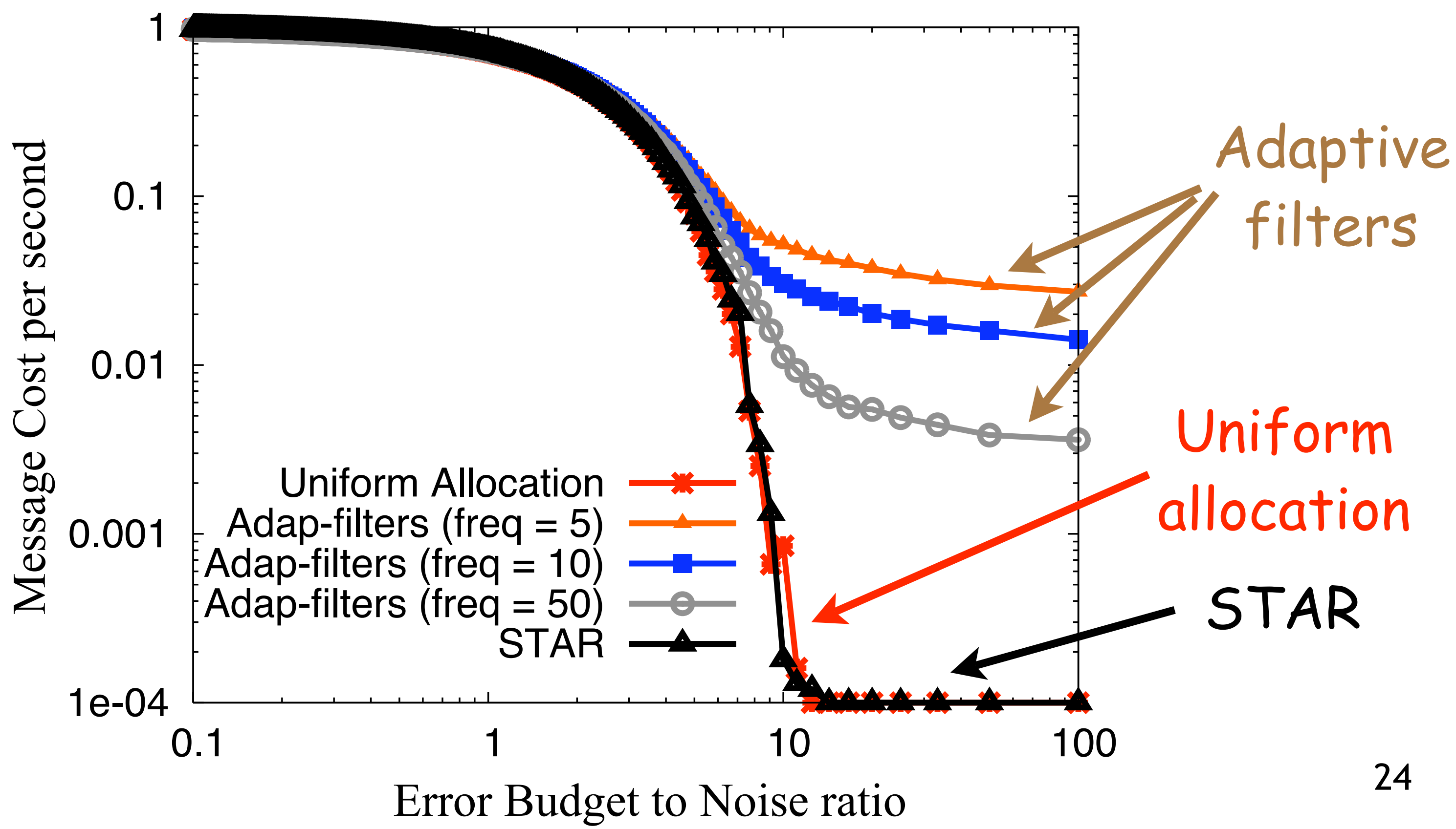
- Self-Tuning: Much better than uniform
- Throttling: Adaptive filters [Olsten '03] wastes messages on useless adjustments



# Does Self-Tuning Approximate Ideal?

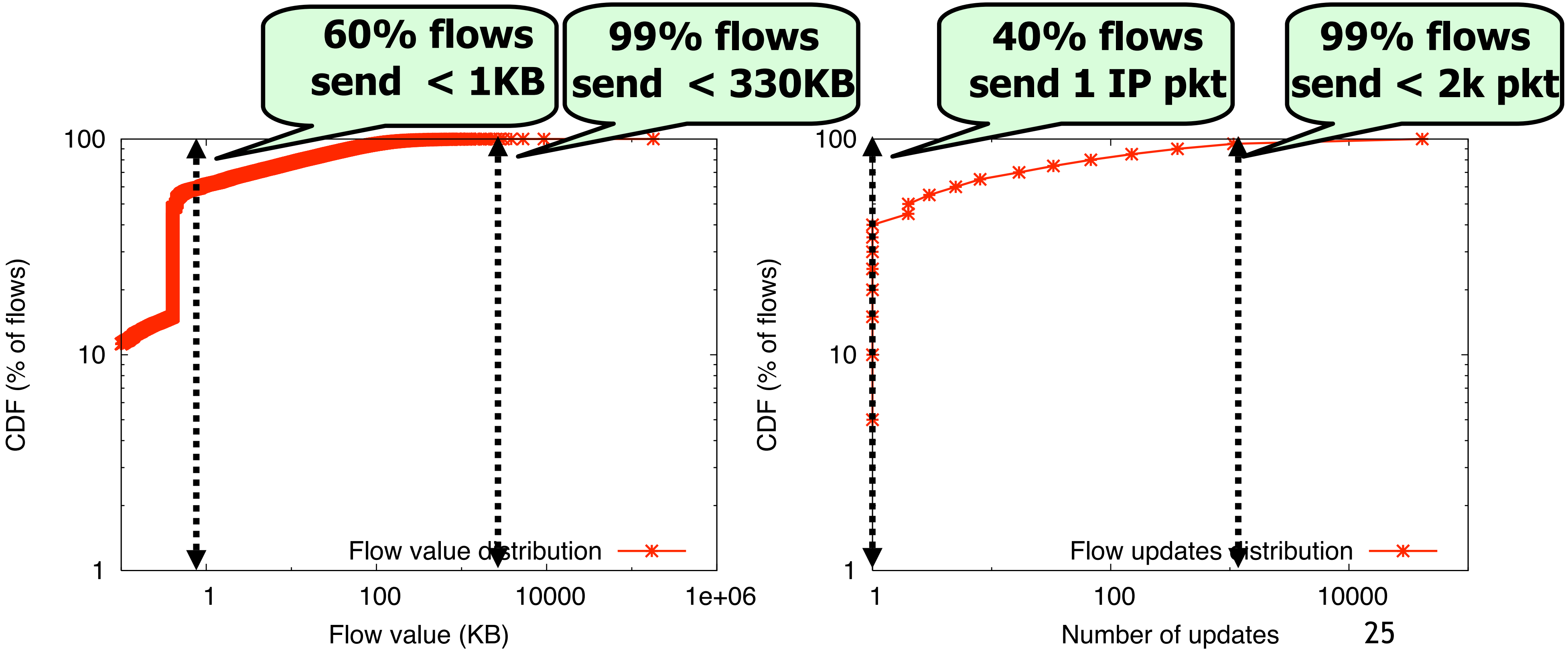
## Uniform noise workload

- Self-tuning approximates uniform allocation
- Avoid useless readjustments



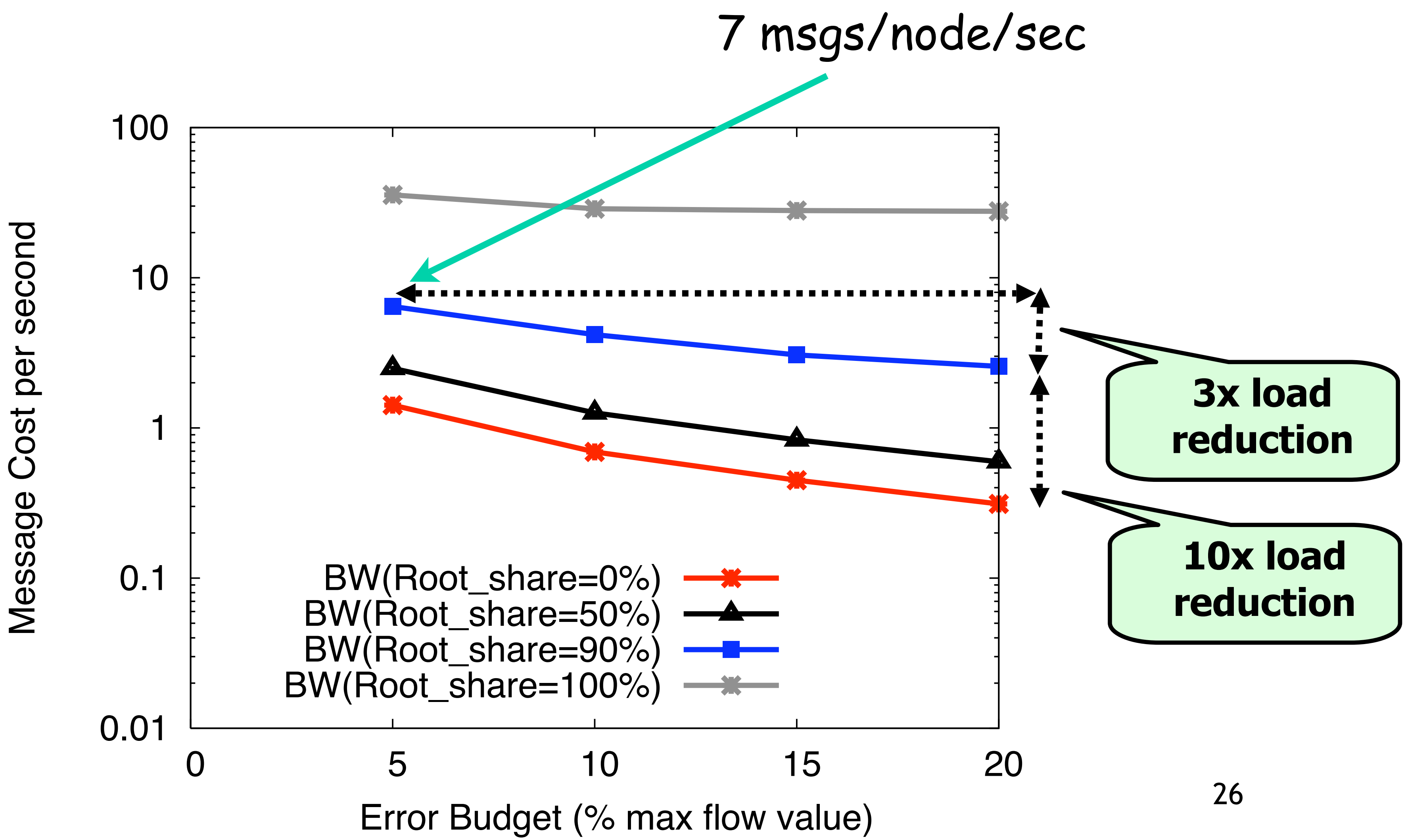
# Abilene Workload

- 80K flows send about 25 million updates in 1 hr
  - Centralized server needs to process 7K updates/sec
  - Heavy tailed distribution



# DHH: Does Self-Tuning Reduce Load?

- Self-tuning significantly reduces load



# STAR Summary

- Scalable self-tuning setting of filter budgets
  - Hierarchical Aggregation
    - Flexible divide budgets across leaves, internal nodes, root
  - Workload-Aware Approach
    - Use variance, update rate to estimate optimal budgets
  - Cost-Benefit Throttling
    - Send budgets where needed

# Thank you!

http://www.cs.utexas.edu/~nav/star

nav@cs.utexas.edu