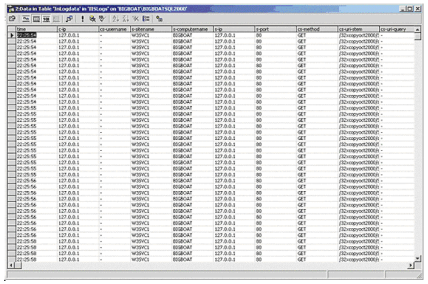
Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area

Ariel Rabkin
Princeton University
asrabkin@cs.princeton.edu

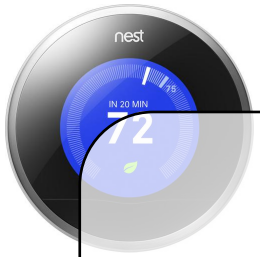
Work done with Matvey Arye, Siddhartha Sen,
Vivek S. Pai, and Michael J. Freedman

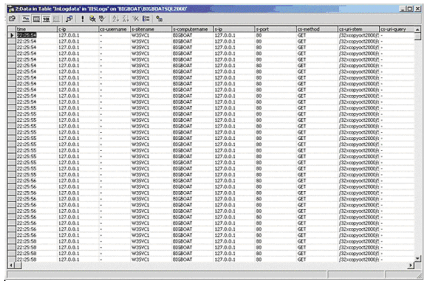
Today's Analytics Architectures



- Backhaul is inefficient and inflexible

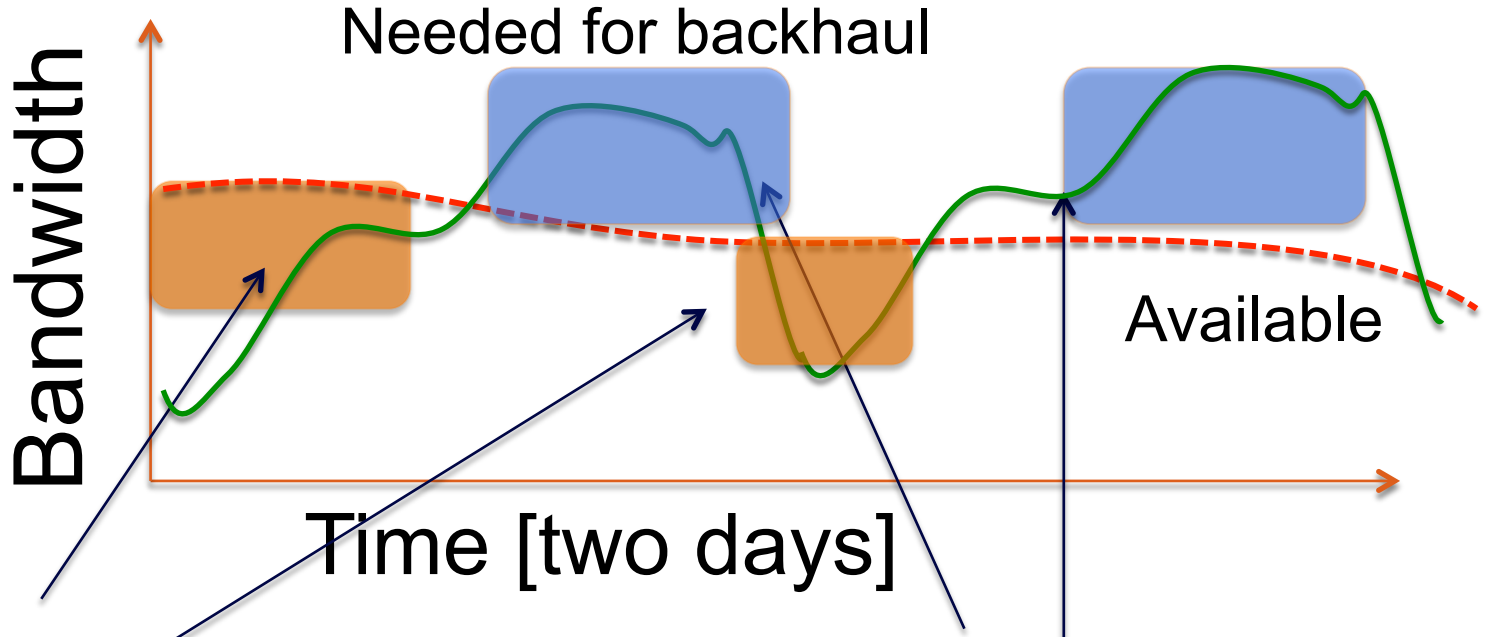
Tomorrow's Architecture: JetStream





- Backhaul is inefficient and inflexible
- Goal: optimize use of WAN links by exposing them to streaming system.

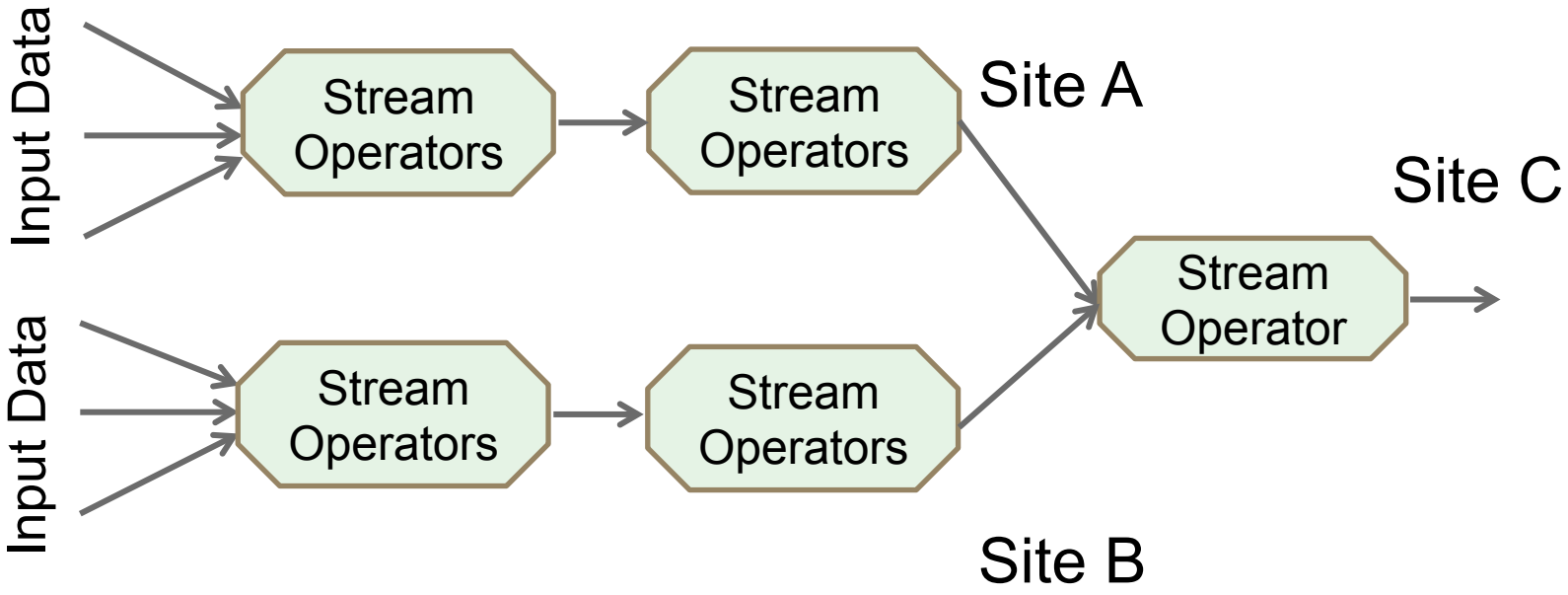
Backhaul is Intrinsically Inefficient



Buyer's remorse: wasted bandwidth

Analyst's remorse: system overload or missing data

Stream Processing Basics



Some Operators in JetStream:

Filtering (count > 100)
Sampling (drop 90% of data)
Image Compression
Quantiles (95th percentile)
Query stored data

The JetStream System

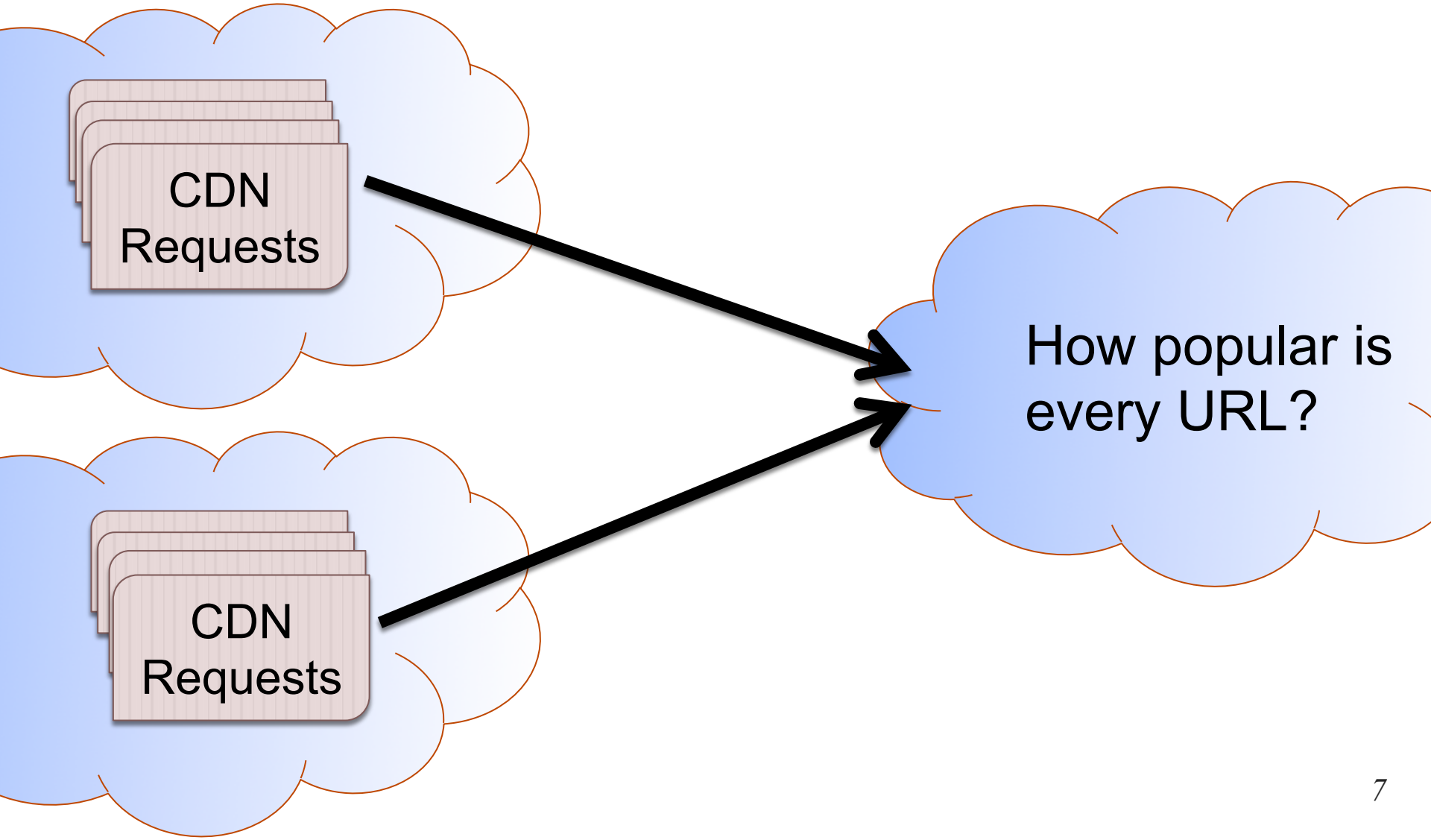
What: Streaming with aggregation and degradation as first-class primitives

Where: Storage and processing at edge

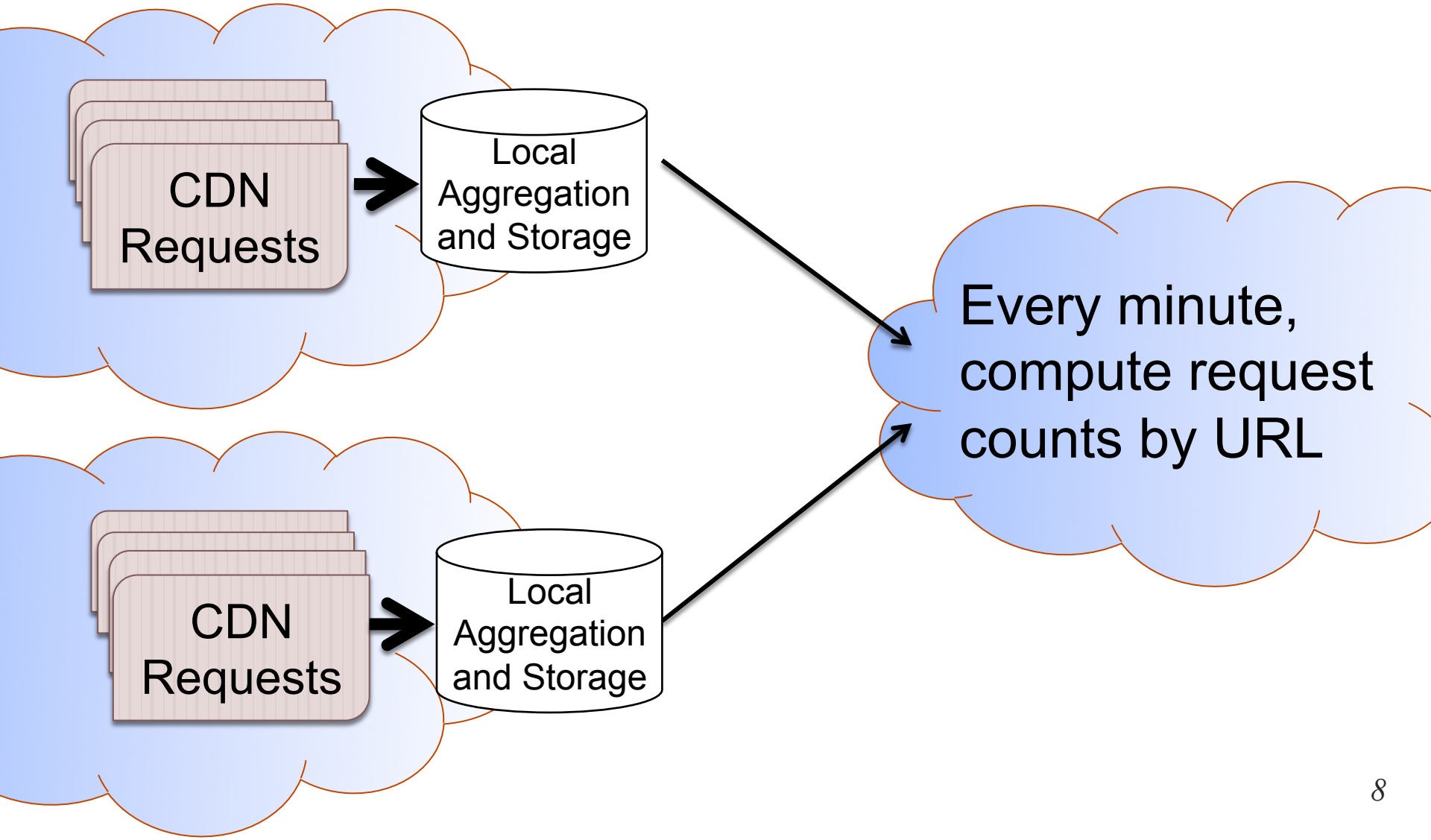
Why: Maximize goodput using aggregation and degradation

How: Data cubes and feedback control

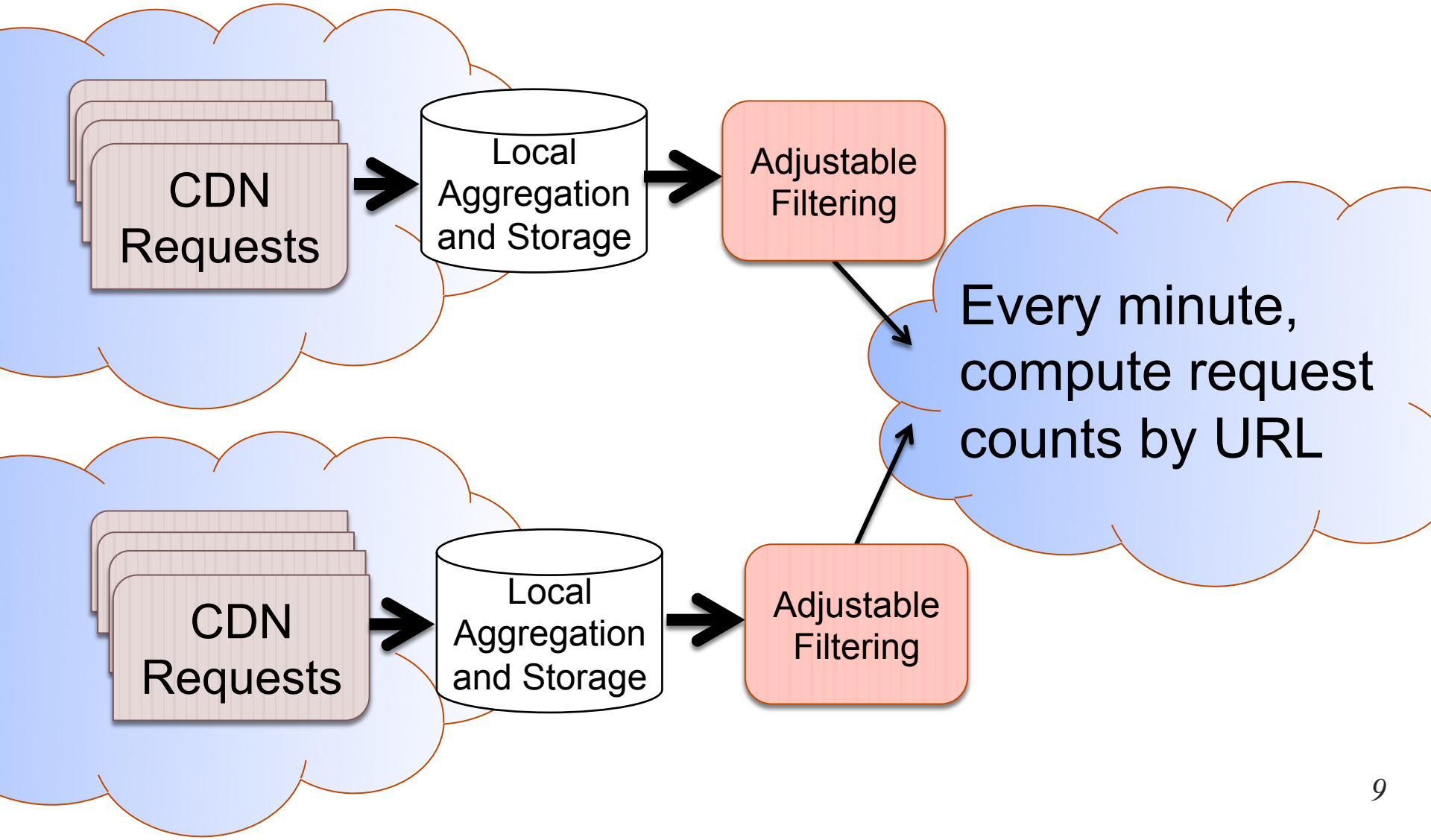
An Example Query



Mechanism 1: Storage with Aggregation

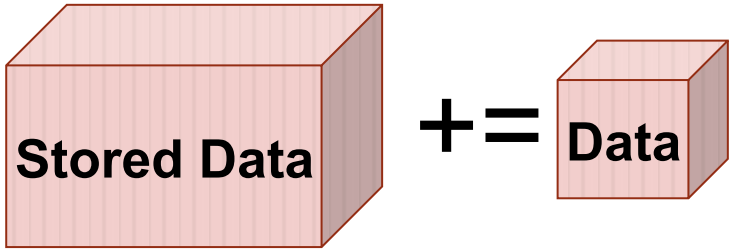


Mechanism 2: Adaptive Degradation

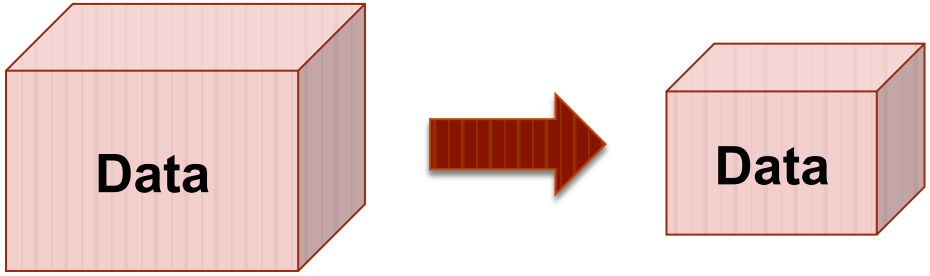


Requirements for Storage Abstraction

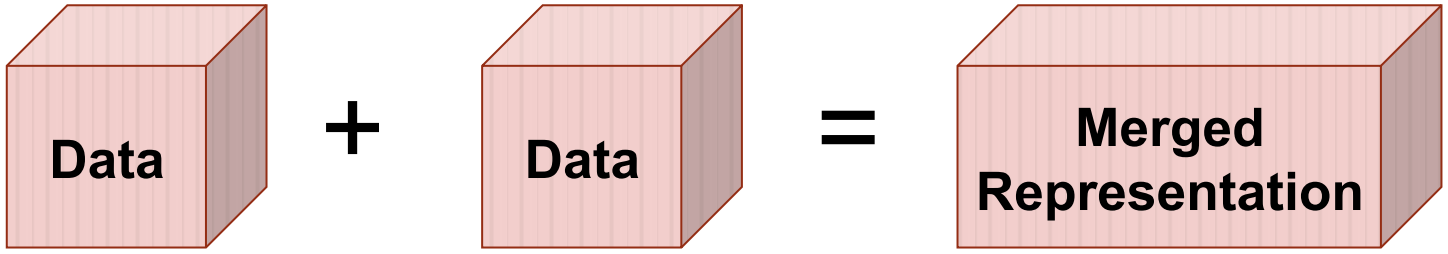
- **Update-able** (locally and incrementally)



- Data size is **reducible** (with predictable accuracy cost)



- **Merge-able** (without accuracy penalty)



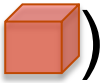
The Data Cube Model

Cube: A multidimensional array, indexed by a set of *dimensions*, whose cells hold *aggregates.*

Counts by URL	12:00	12:01	12:02
www.mysite.com/a	3	5	0
www.mysite.com/b	0	2	0
www.yoursite.com	5	4	…
www.her-site.com	8	12	…

Aggregation used for:

- Updates
- Roll-ups
- Merging cubes
- Summarizing cubes

Cubes have aggregation function: Agg( , )→ 

Cubes can be “Rolled Up”

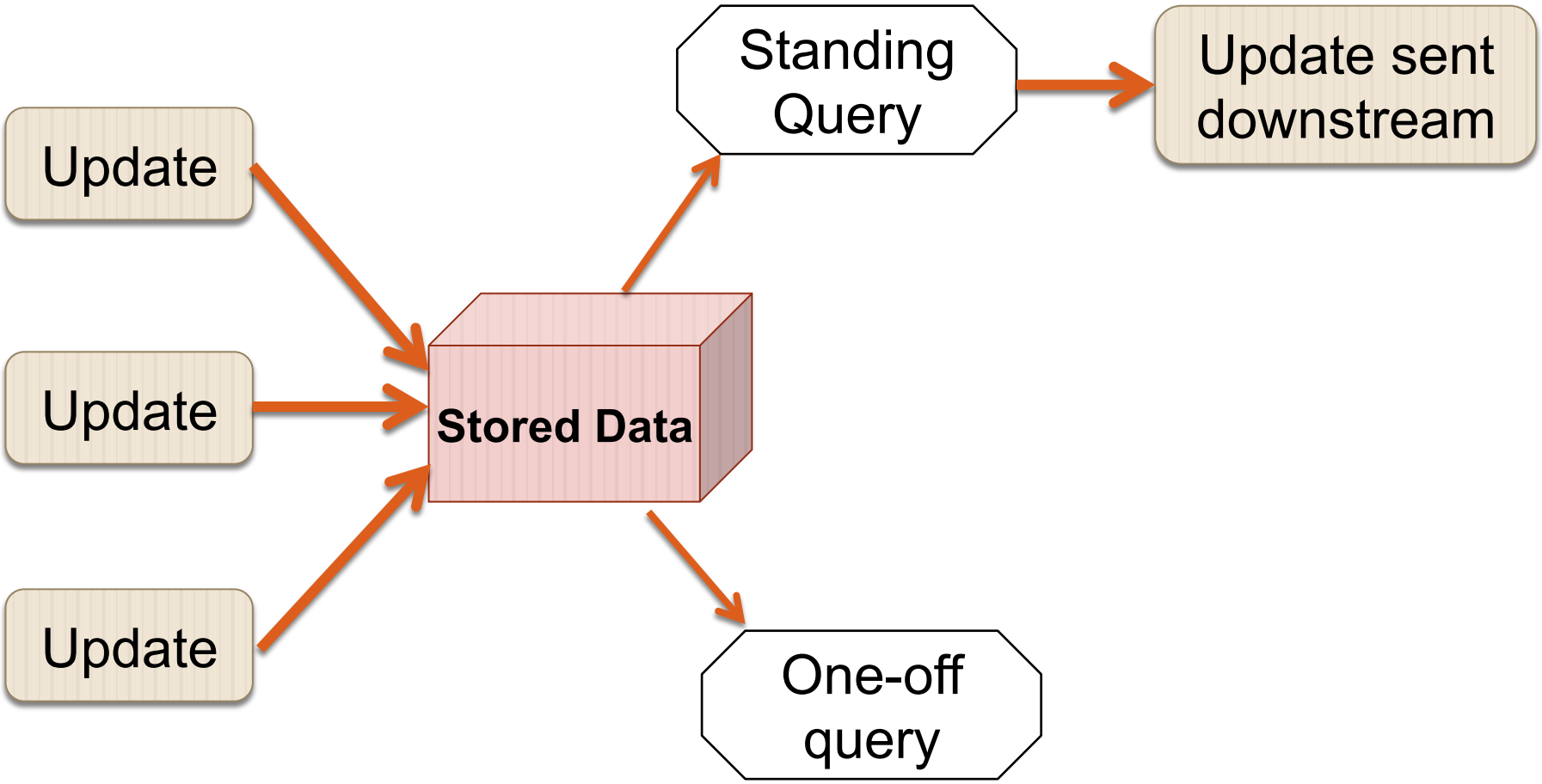
Cube: A multidimensional array, indexed by a set of *dimensions*, whose cells hold *aggregates.*

Counts by URL	12:00	12:01	12:02
www.mysite.com/a	3	5	0
www.mysite.com/b	0	2	0
www.yoursite.com	5	4	…
www.her-site.com	8	12	…

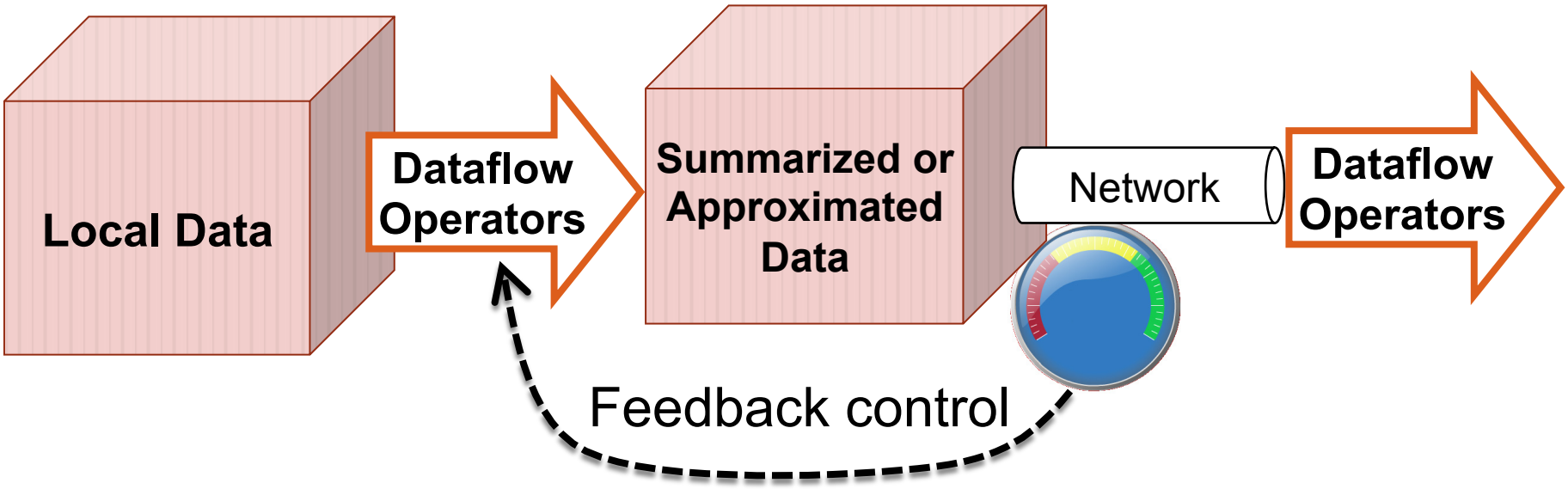
Counts by URL	*
www.mysite.com/a	8
www.mysite.com/b	2
www.yoursite.com	9
www.her-site.com	20

Counts by URL	12:00	12:01	12:02
*	16	23	…

Cubes Unify Storage and Aggregation

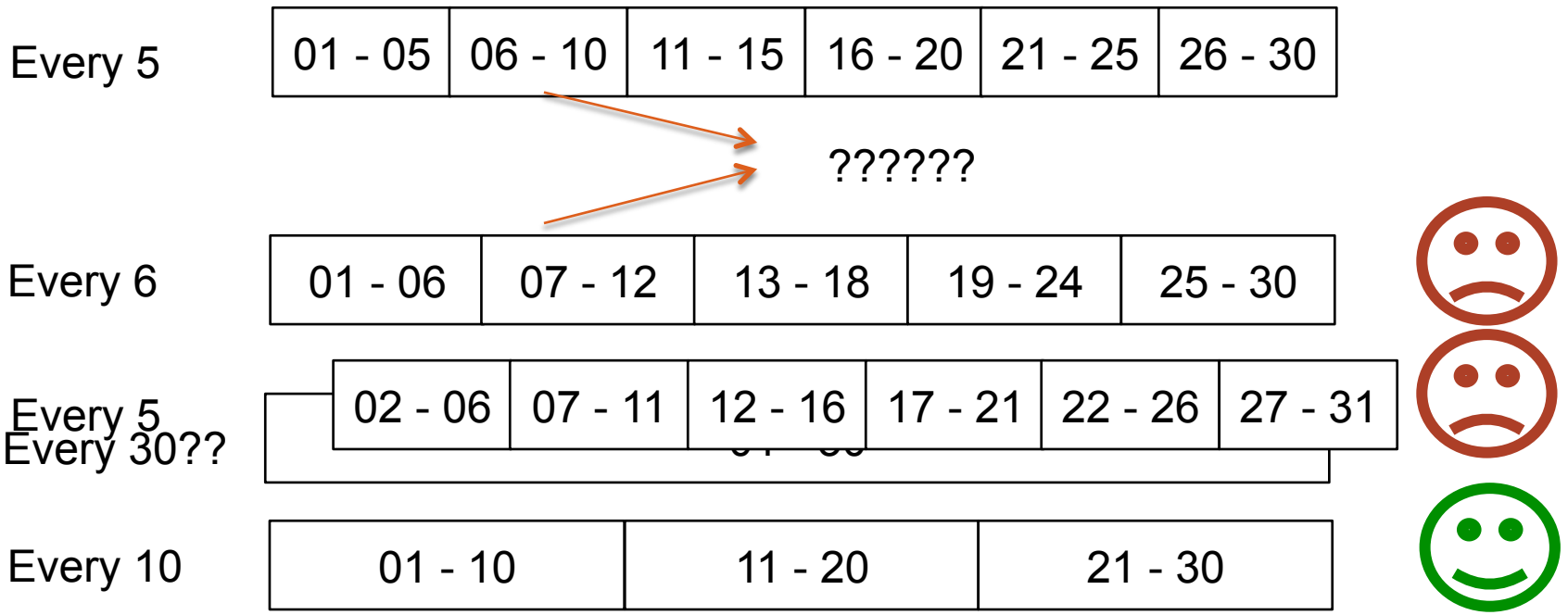


Degradation: The Big Picture



- Level of degradation auto-tuned to match bandwidth.
- Challenge: Supporting mergeability and flexible policies

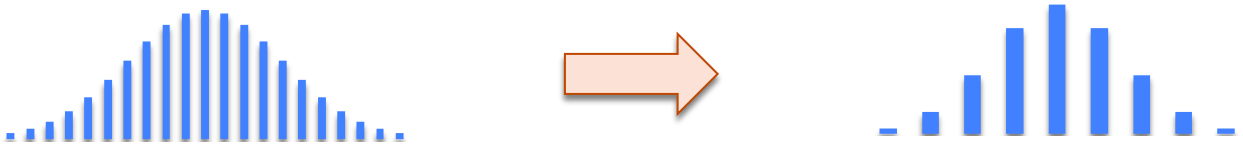
Mergeability Imposes Constraints



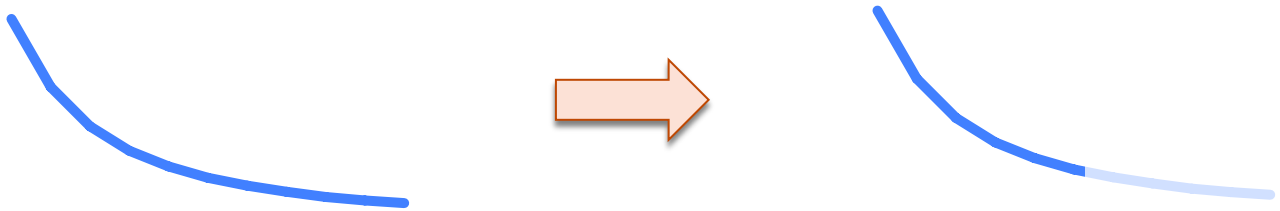
- Insight: Degradation may be discontinuous

There Are Many Ways to Degrade Data

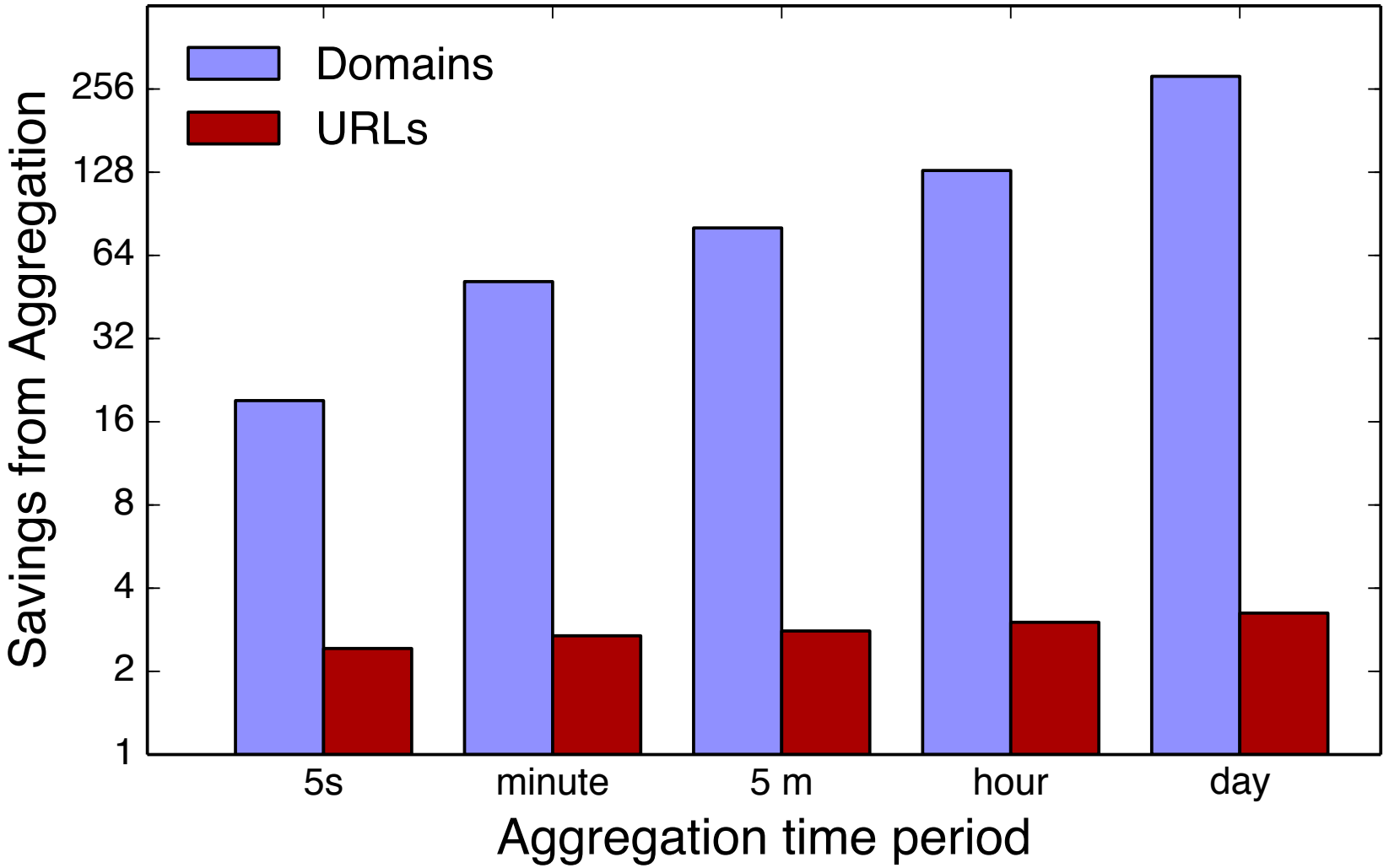
- Can *coarsen* a dimension



- Can drop low-rank values



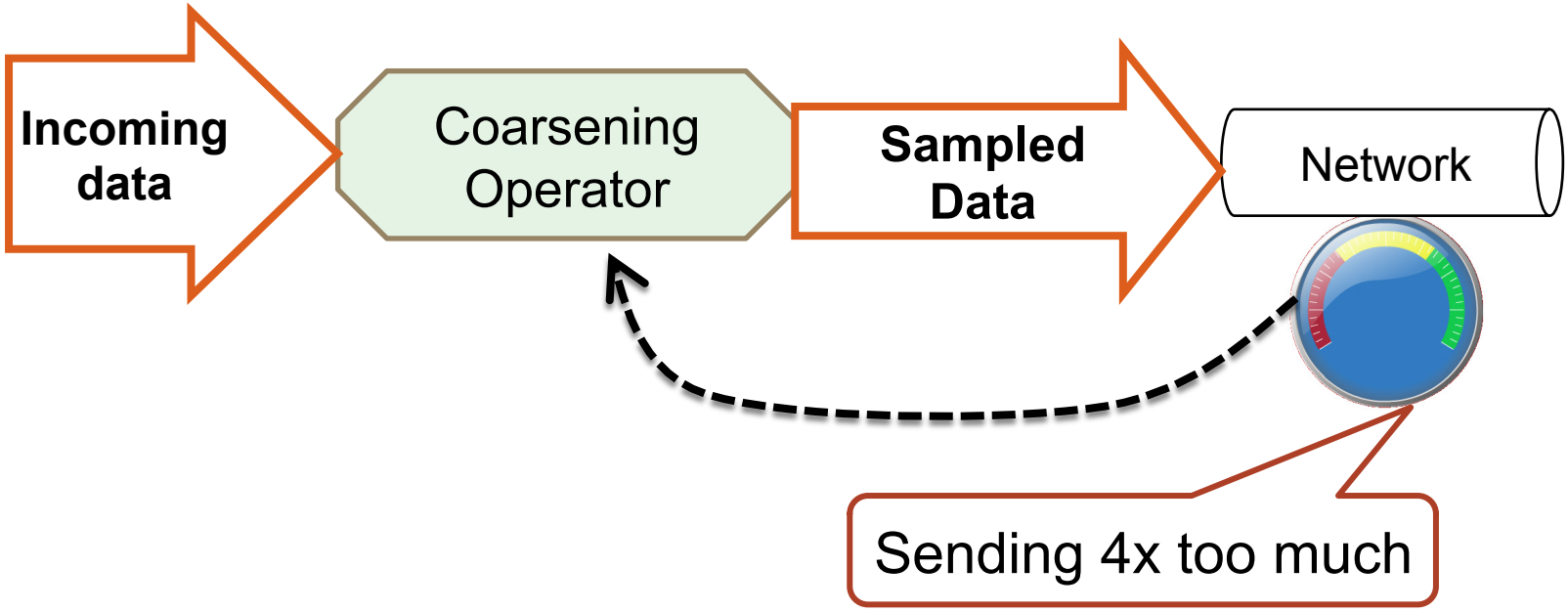
Coarsening Does Not Always Help



Degradations Have Trade-offs

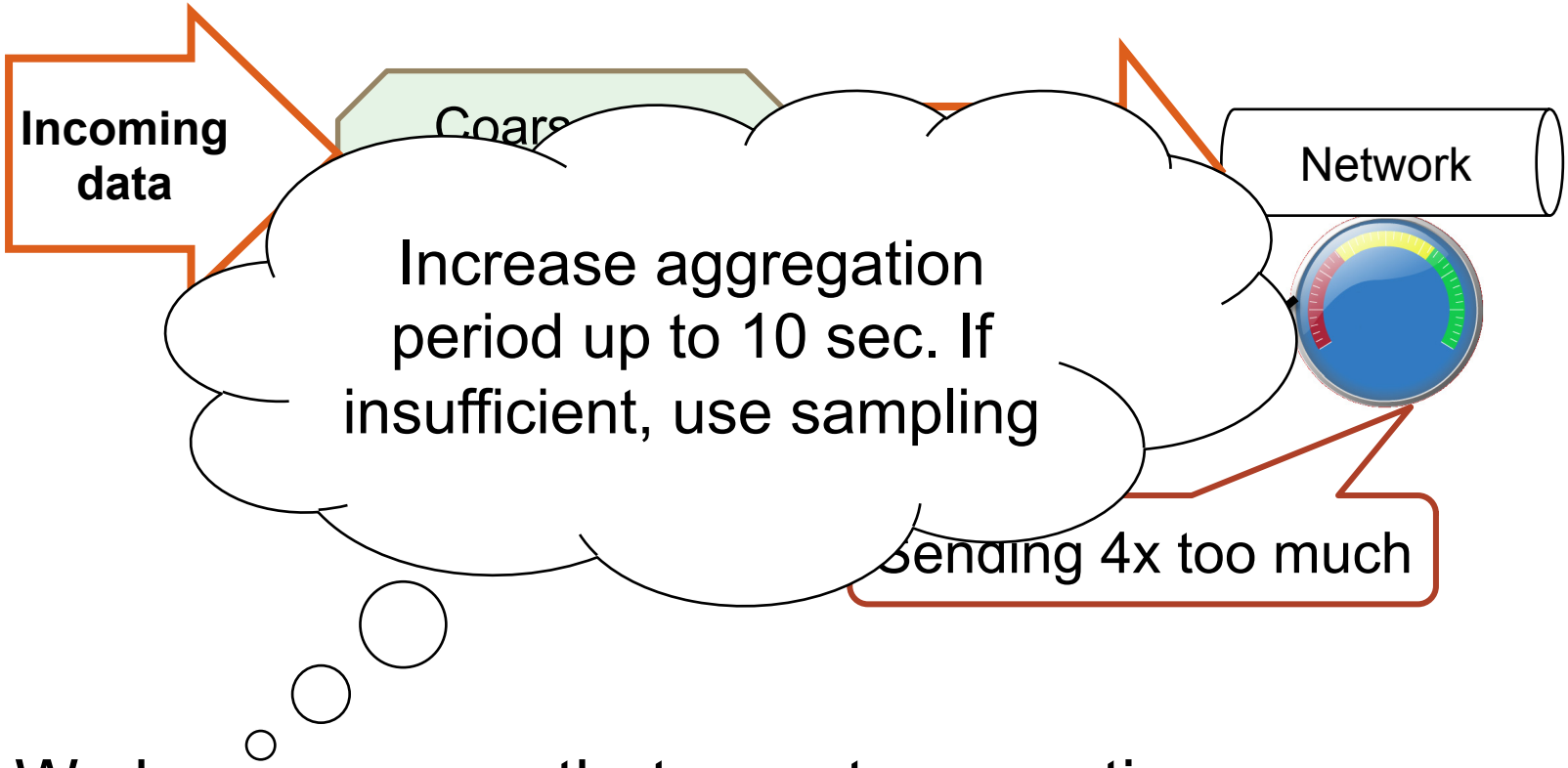
Name	Fixed BW Savings	Fixed Accuracy cost	Parameter
Dim. Coarsening	**Usually no**	**Yes**	**Dimension Scale**
Drop values (locally)	**Yes**	**No**	**Cut-off**
Drop values (globally)	**No, multi-round protocol**	**Yes**	**Cut-off**
Audiovisual downsampling	**Yes**	**Yes**	**Sample rate**
Histogram Coarsening	**Yes**	**Yes**	**Number of Buckets**

A Simple Idea that Does Not Work



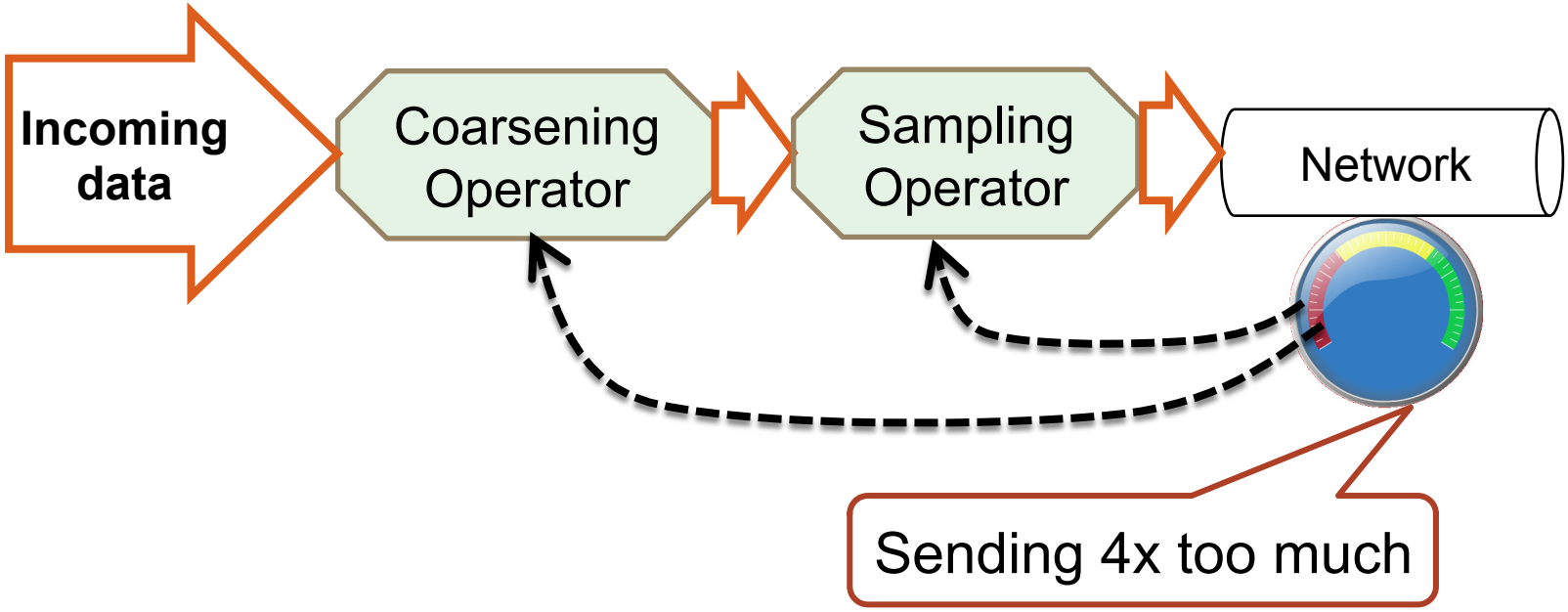
- We have sensors that report congestion….
- Have operators read sensor and adjust themselves?

A Simple Idea that Does Not Work



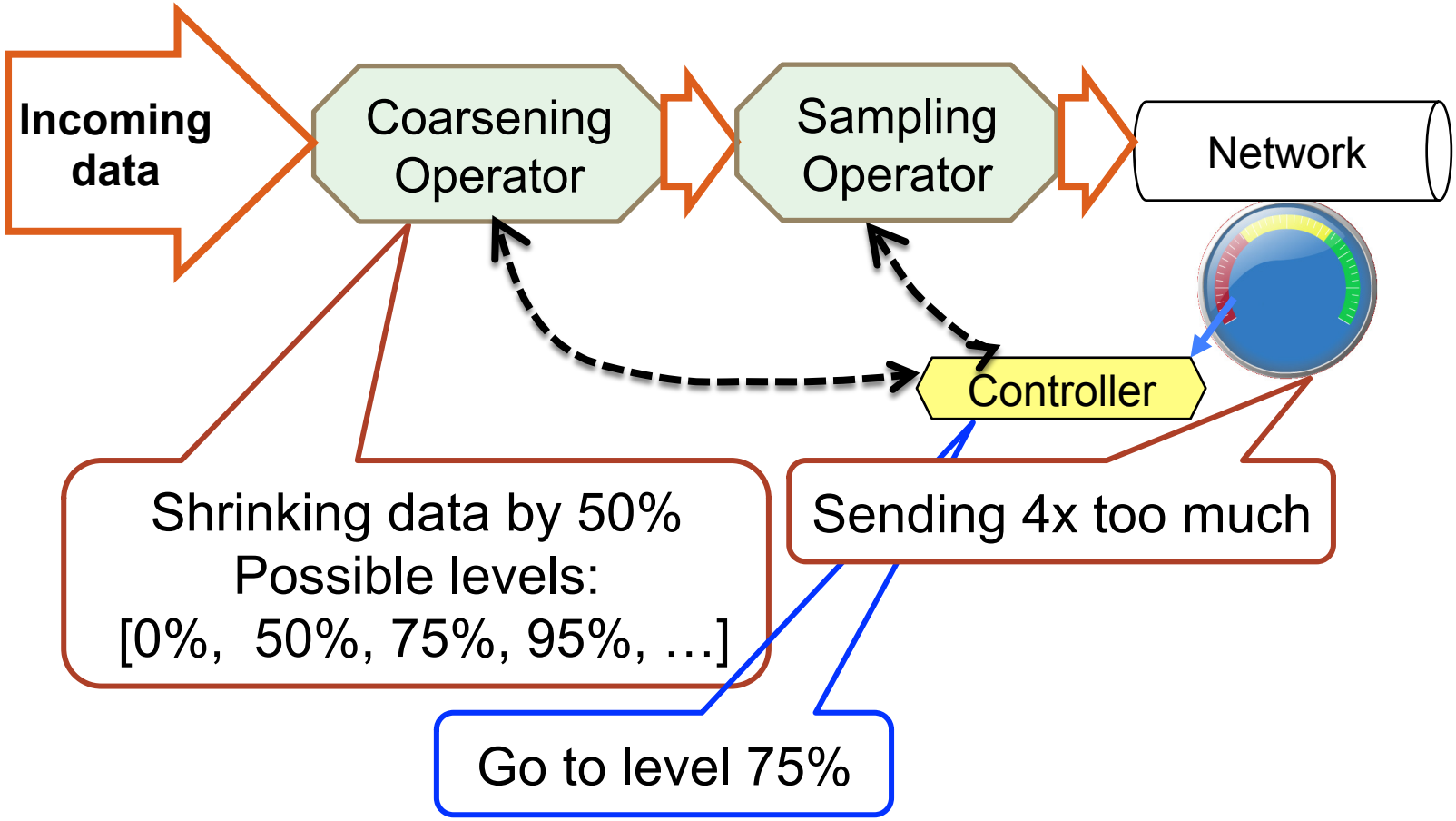
- We have sensors that report congestion….
- Have operators read sensor and adjust themselves?

Challenge: Composite Policies



- Chaos if two operators are simultaneously responding to the same sensor

Interfacing with Operators



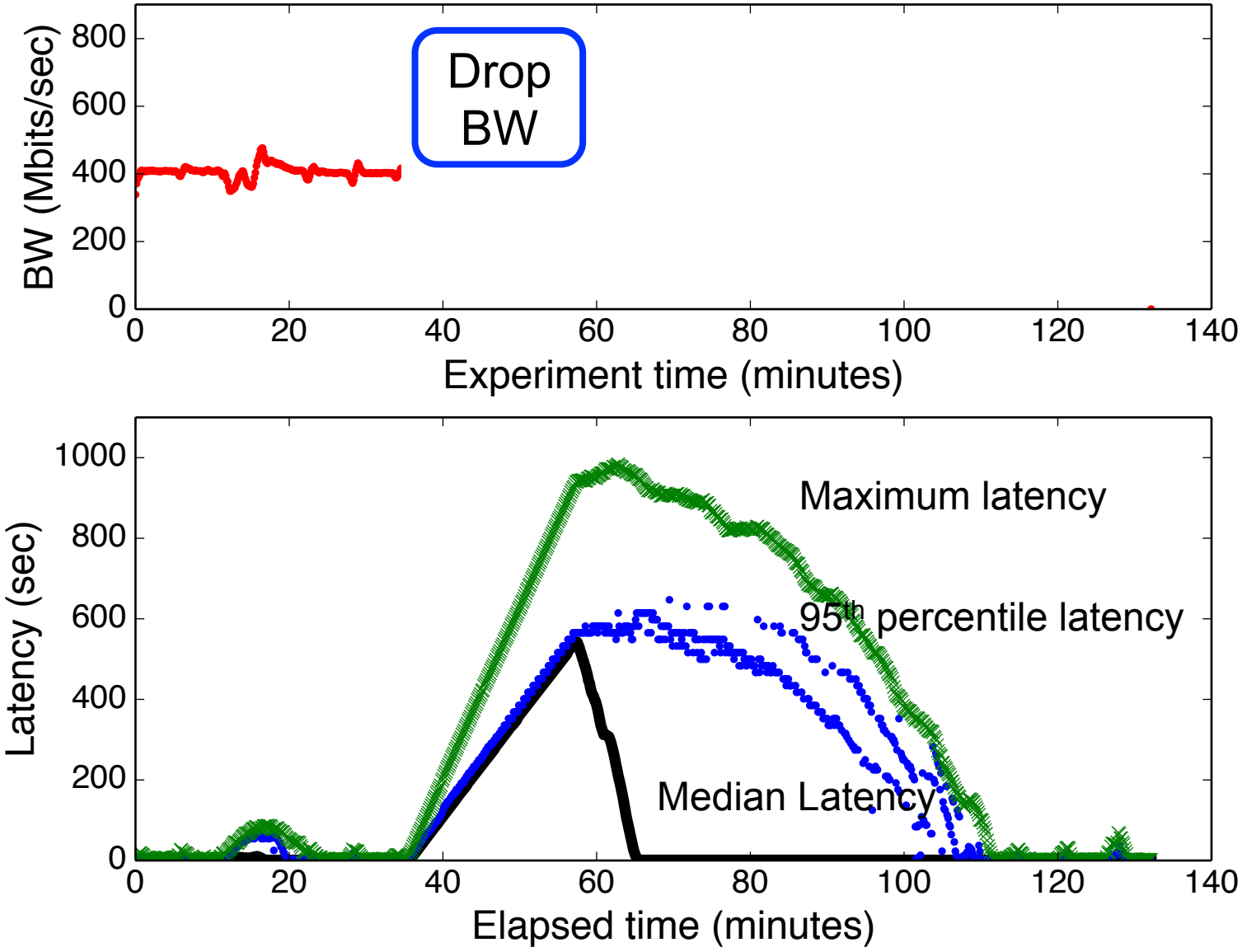
Experimental Setup

80 nodes on VICCI testbed at three sites
(Seattle, Atlanta, and Germany)

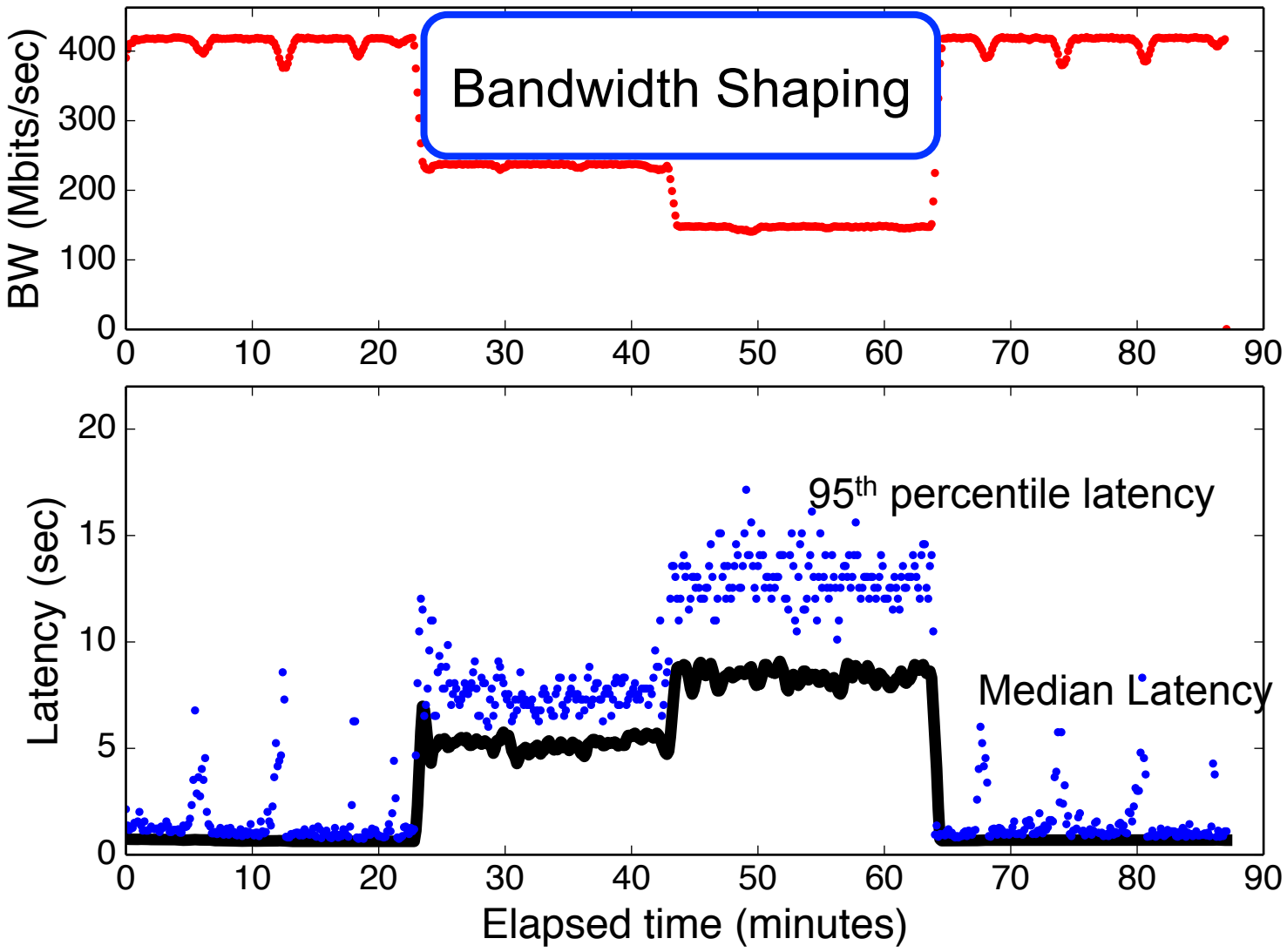
Princeton

Policy: Drop data if insufficient BW

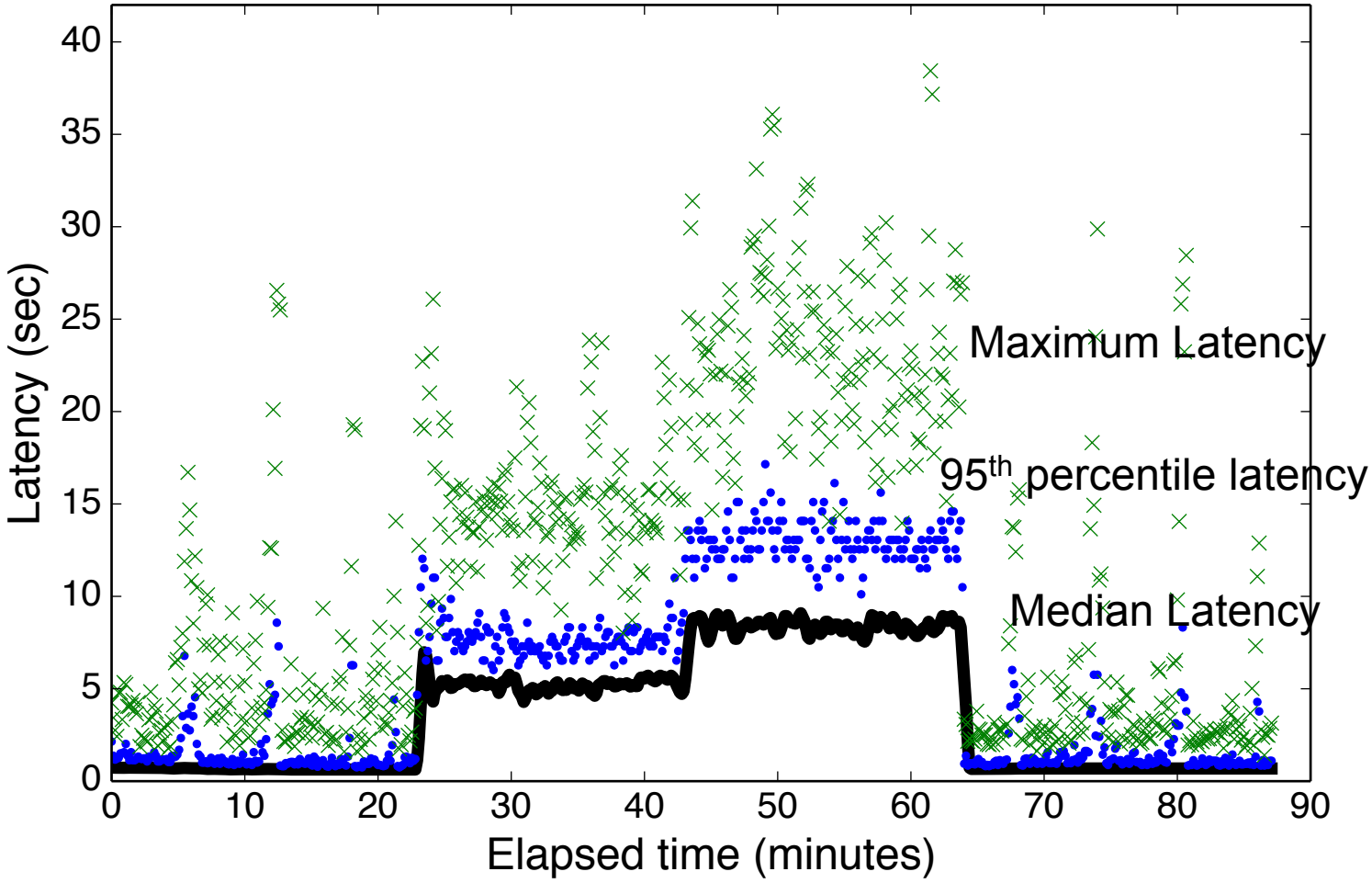
Without Degradation



Degradation Keeps Latency Bounded



Showing maximum latencies



Programming Ease

Scenario	Lines of code
Slow requests	5
Requests by URL	5
Bandwidth by node	15
Bad referrers	16
Latency and size quantiles	25
Success by domain	30
Top 10 domains by period	40
Big Requests	97

Conclusions and Future Work

- Useful to embed aggregation and degradation abstractions in streaming systems.
- Aggregation can be unified with storage.
- System must accommodate degradation semantics.
- Open questions:
  - How to guide users to the right degradation policy?
  - How to embed abstractions in higher-level language?