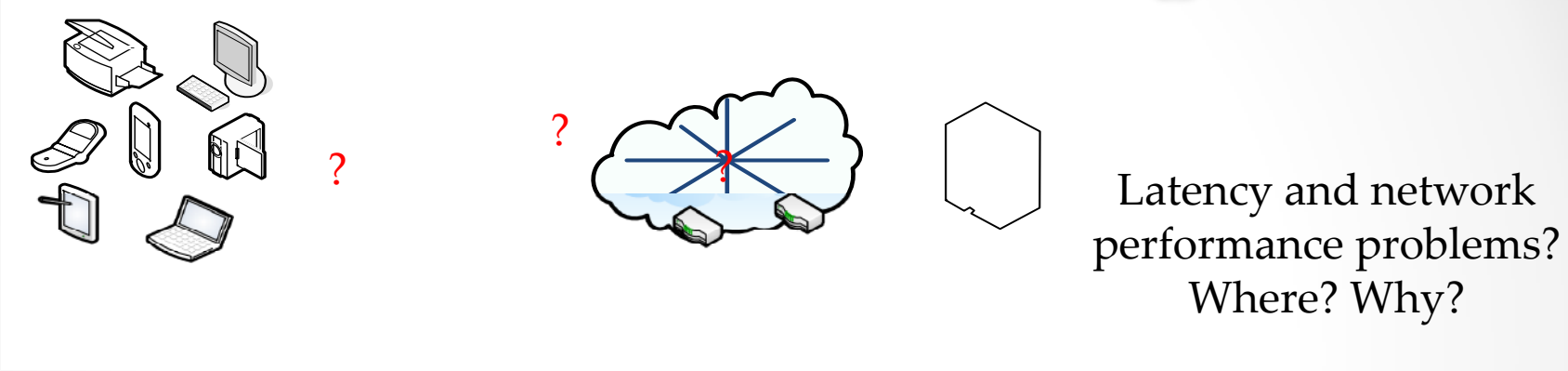
Estimating TCP Latency Approximately with Passive Measurements

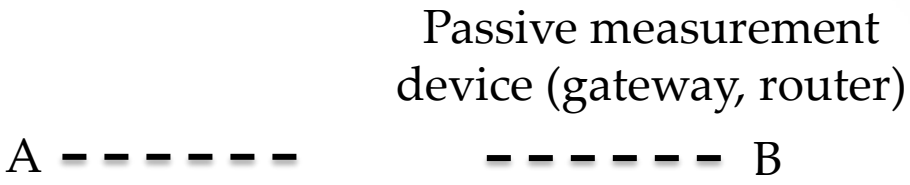
Sriharsha Gangam, Jaideep Chandrashekar, Ítalo Cunha, Jim Kurose

Motivation: Network Troubleshooting



- Home network or access link?
  - WiFi signal or access link?
- ISP, enterprise and content provider networks
  - Within the network or outside?

Passive Measurement in the Middle

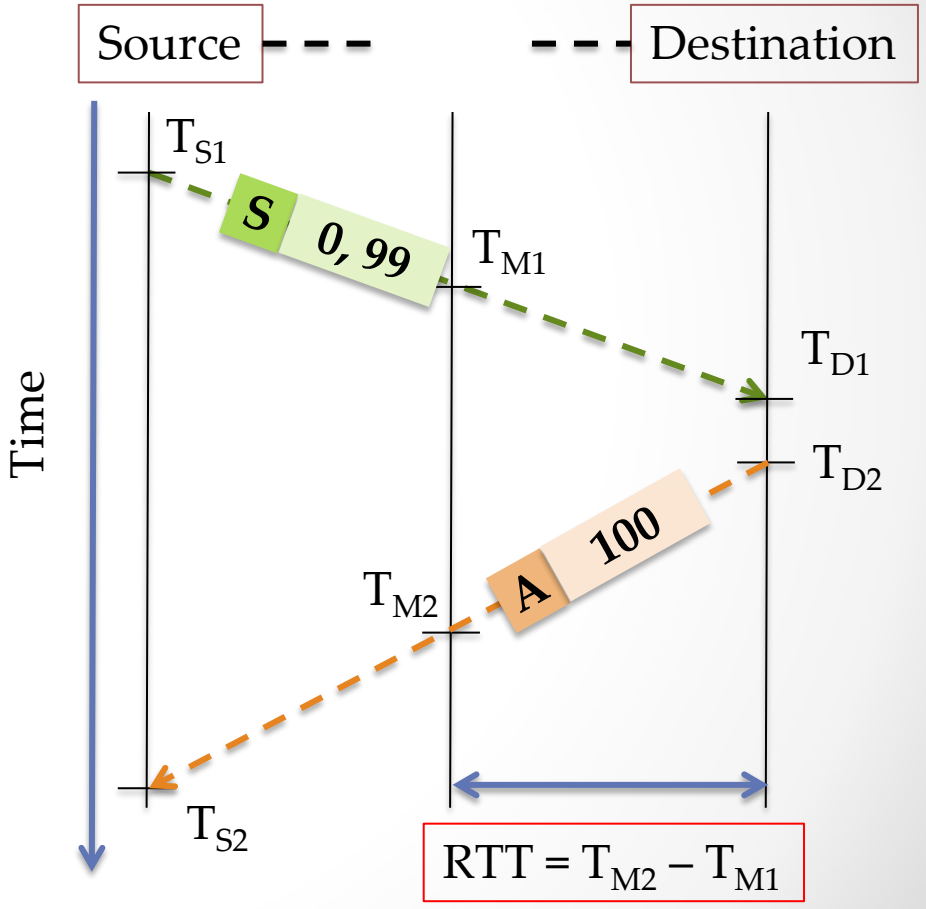


Challenge: Constrained Resources

- Decompose path latency of TCP flows
  - Latency: Important interactive web-applications
  - Other approaches: active probing, NetFlow
- Challenges
  - Constrained devices and high data rates

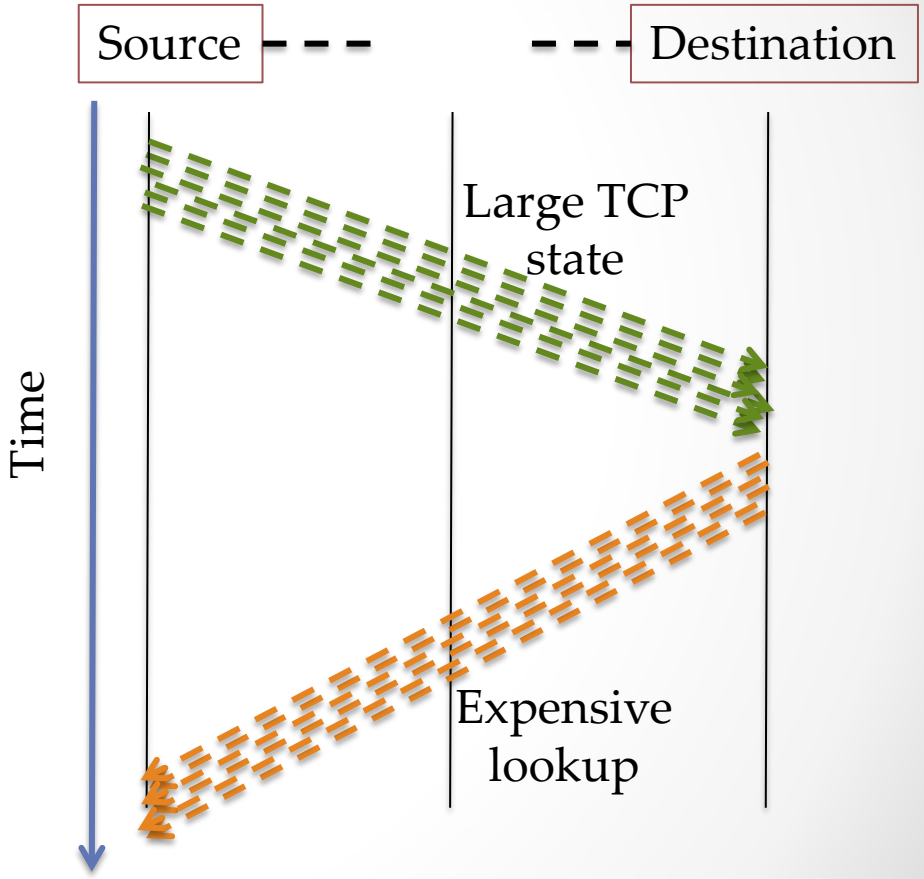
TCP Latency

- Existing methods: Emulate TCP state for each flow
- Accurate estimates but expensive

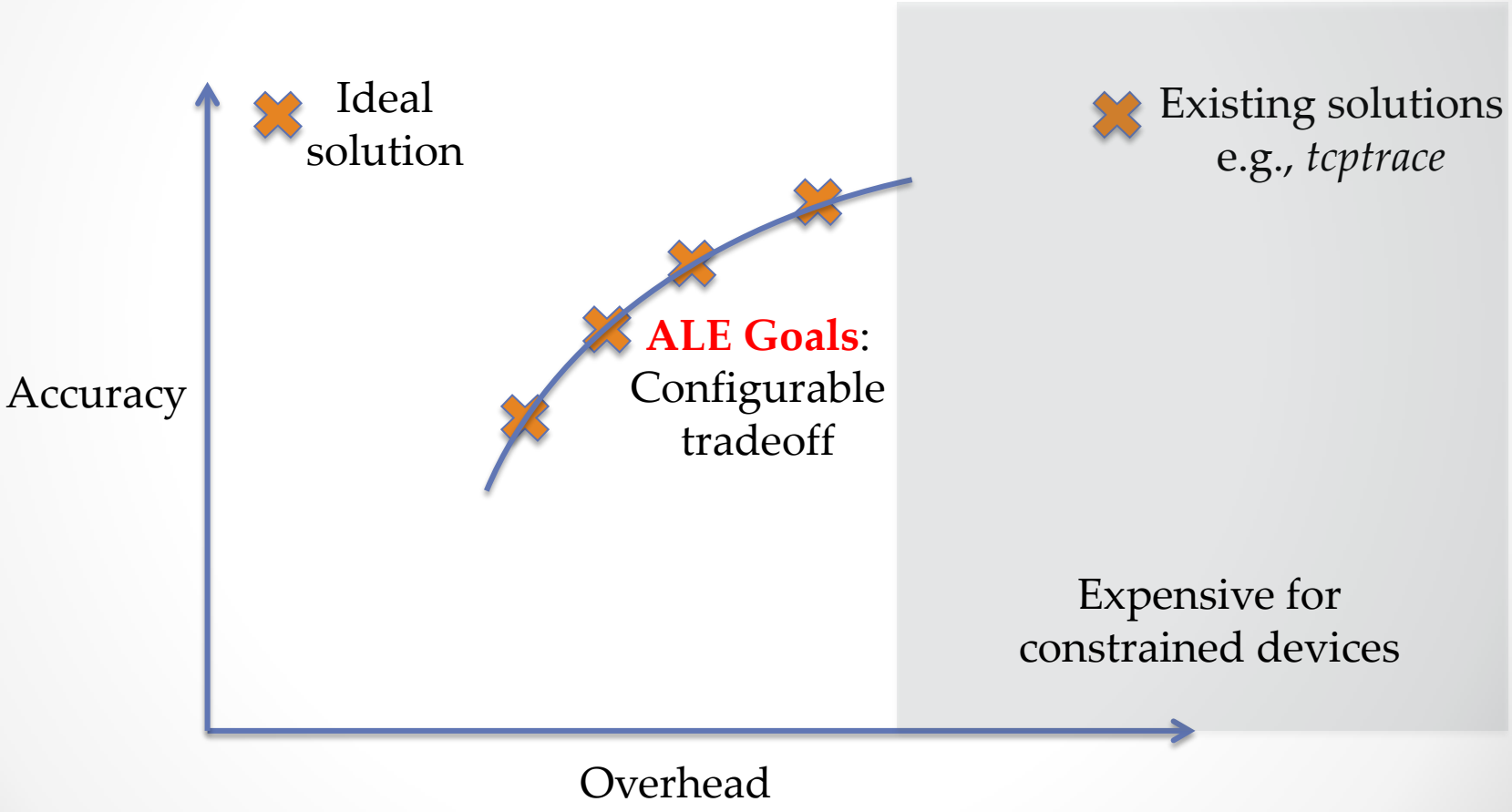


TCP Latency

- Scaling challenges
  - Large packet and flow arrivals
  - Constrained devices (limited memory/CPU)
- ALE – Approximate Latency Estimator

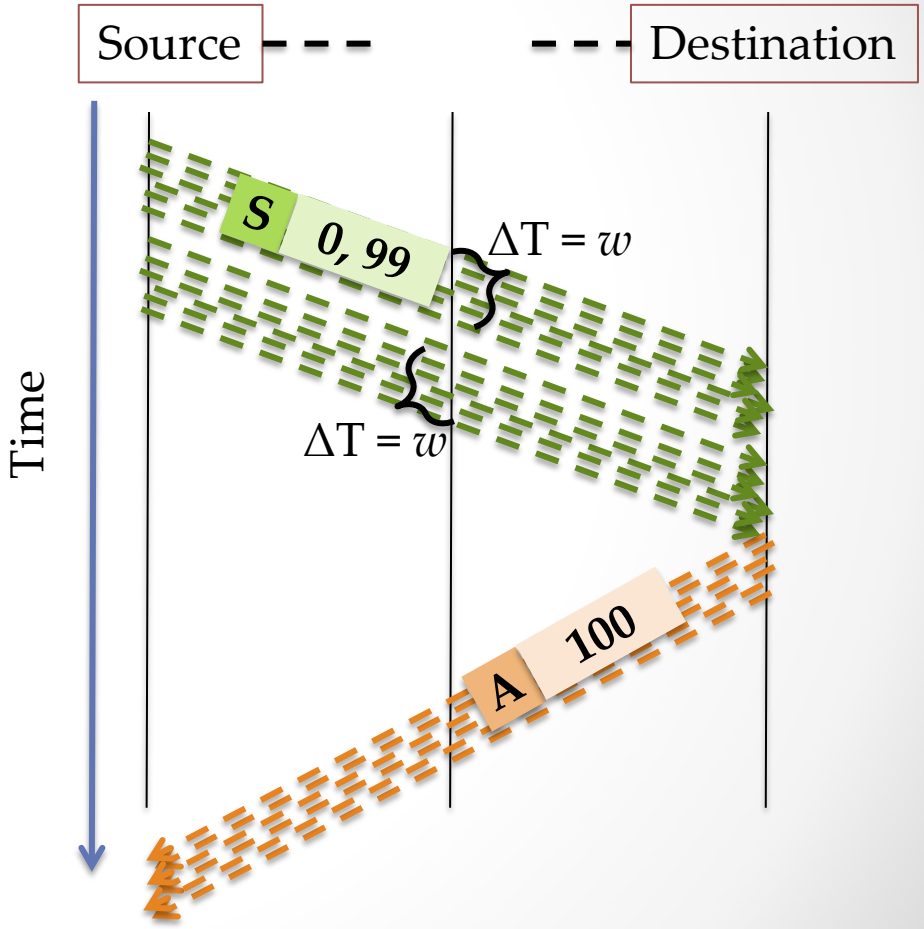


Design Space



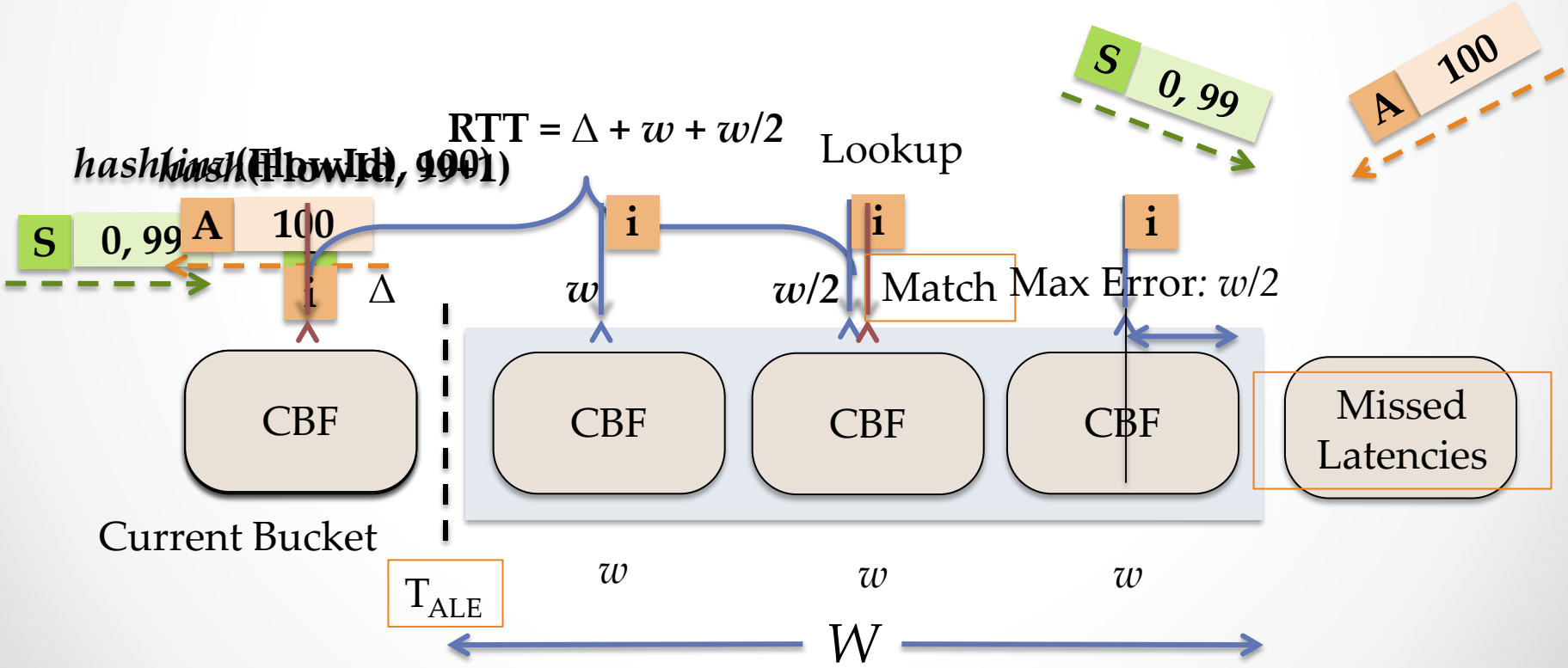
ALE Overview

- Eliminate packet timestamps
  - Group into time intervals with granularity (*w*)
- Store expected ACK
  - SEG: 0-99, ACK: 100
- Use probabilistic set membership data structure
  - Counting Bloom Filters

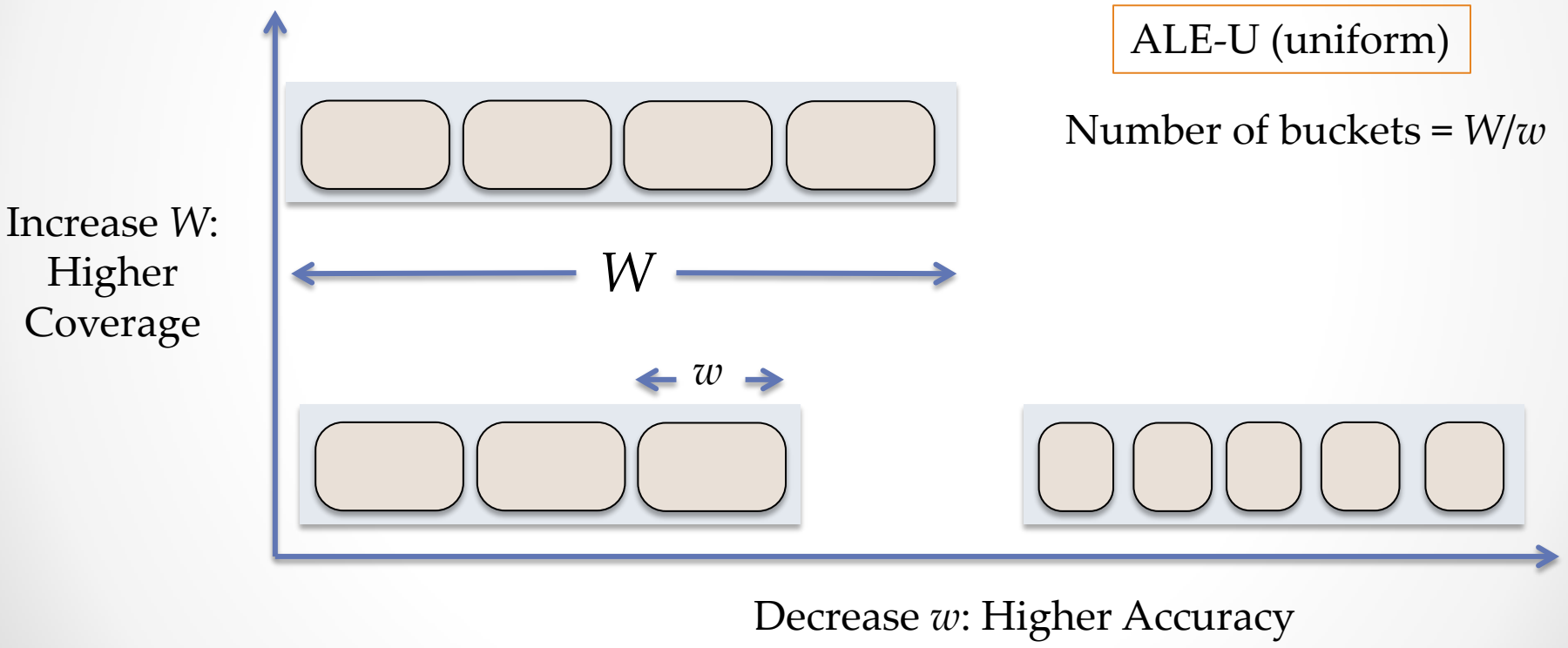


ALE Overview

- Sliding window of buckets (time intervals)
  - Buckets contain a counting bloom filter (CBF)

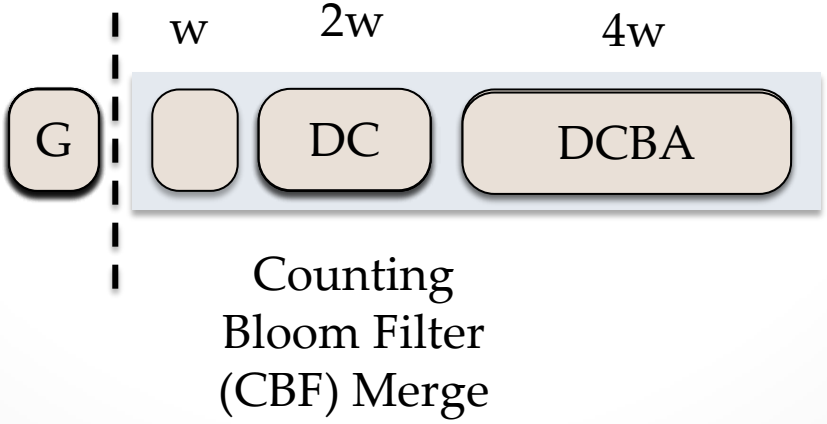


Controlling Error with ALE Parameters



ALE-Exponential (ALE-E)

- Process large and small latency flows simultaneously
- Absolute error is proportional to the latency
- Larger buckets shift slowly



Error Sources

- Bloom filters are probabilistic structures
  - False positives and negatives

Artifacts from TCP

- Retransmitted packets and Reordered packets
  - Excess (to *tcptrace*) erroneous RTT samples

- ACK numbers not on SEQ boundaries, Cumulative ACKs

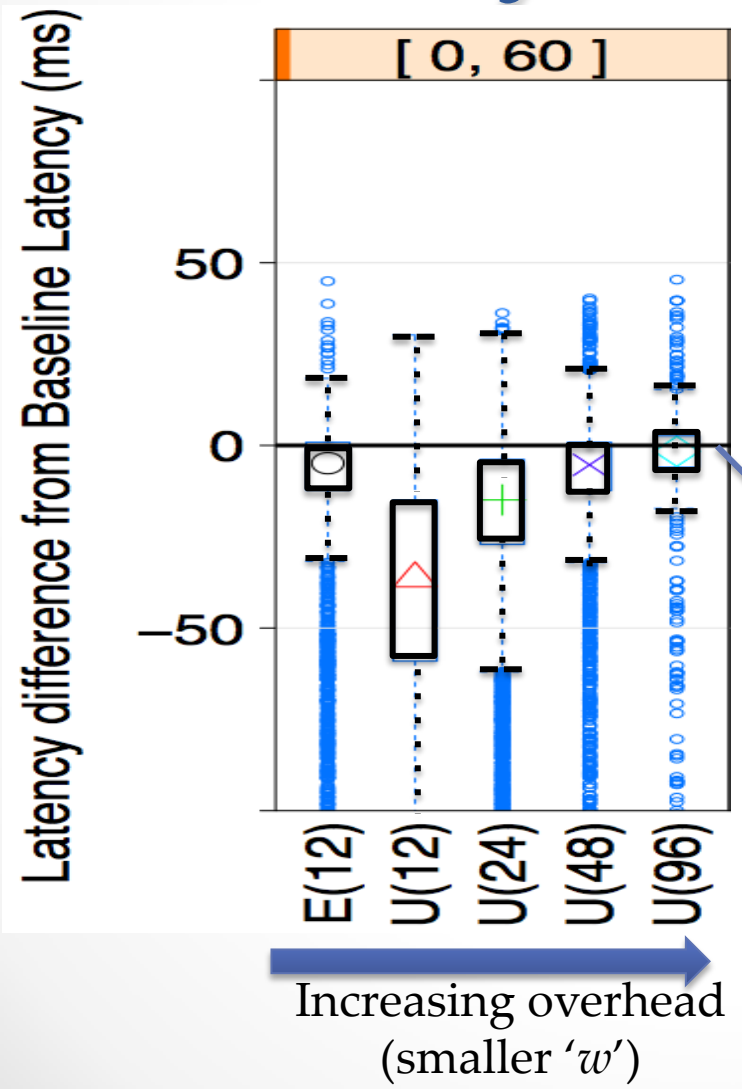
Evaluation Methodology

- 2 x Tier 1 backbone link traces (60s each) from CAIDA
  - Flows: 2,423,461, Packets: 54,089,453
  - Bi-directional Flows: 93,791, Packets: 5,060,357
- Ground truth/baseline comparison: *tcptrace*
  - Emulates TCP state machine
- Latencies ALE and *tcptrace*
  - Packet and flow level
- Overhead of ALE and *tcptrace*
  - Memory and compute

Latency Distribution

60 ms 120 ms 300 ms 500 ms

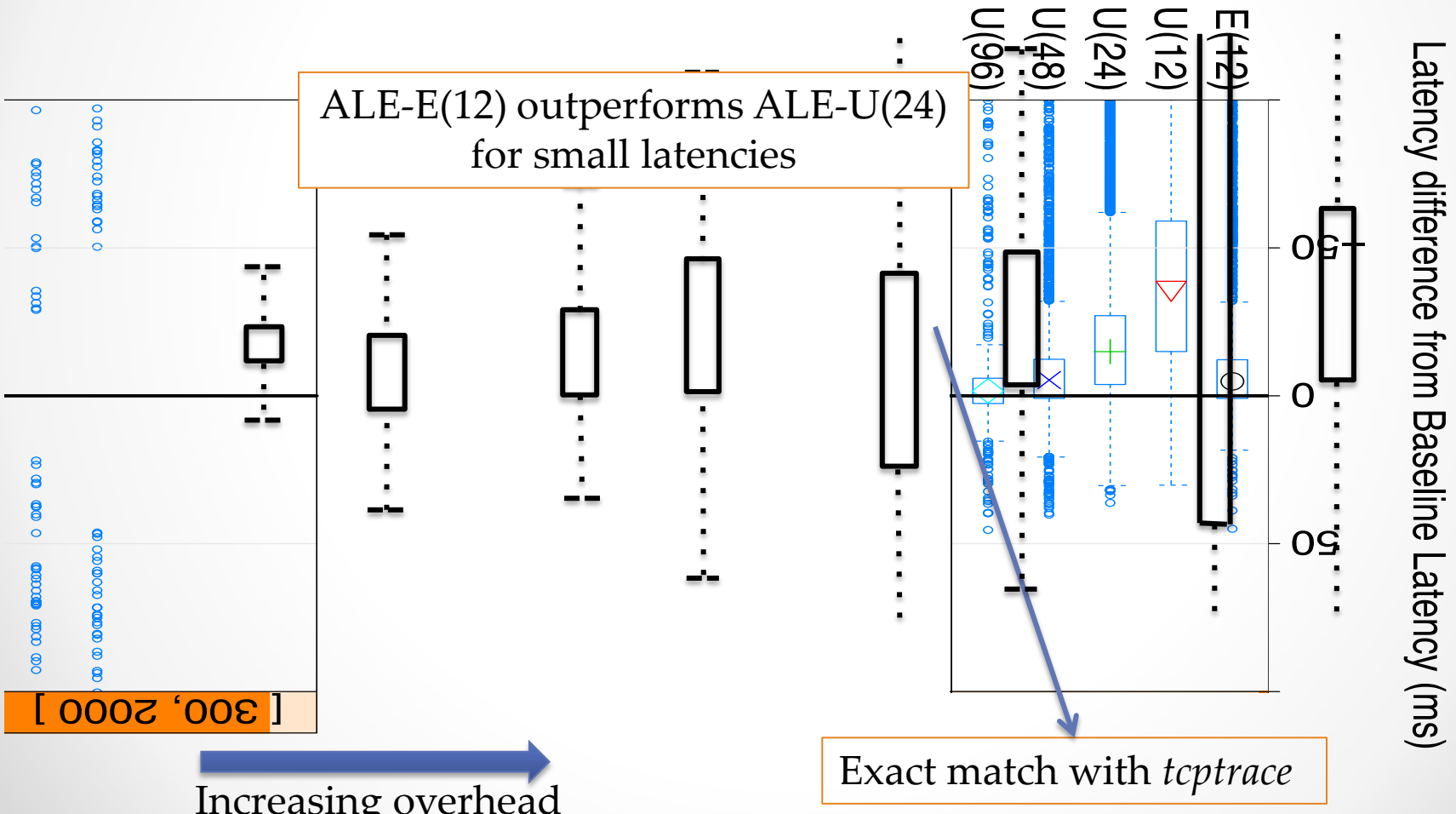
Accuracy: RTT Samples



- Range 0-60 ms
- Box plot of RTT differences
- More memory => Closer to *tcptrace*

Exact match with *tcptrace*

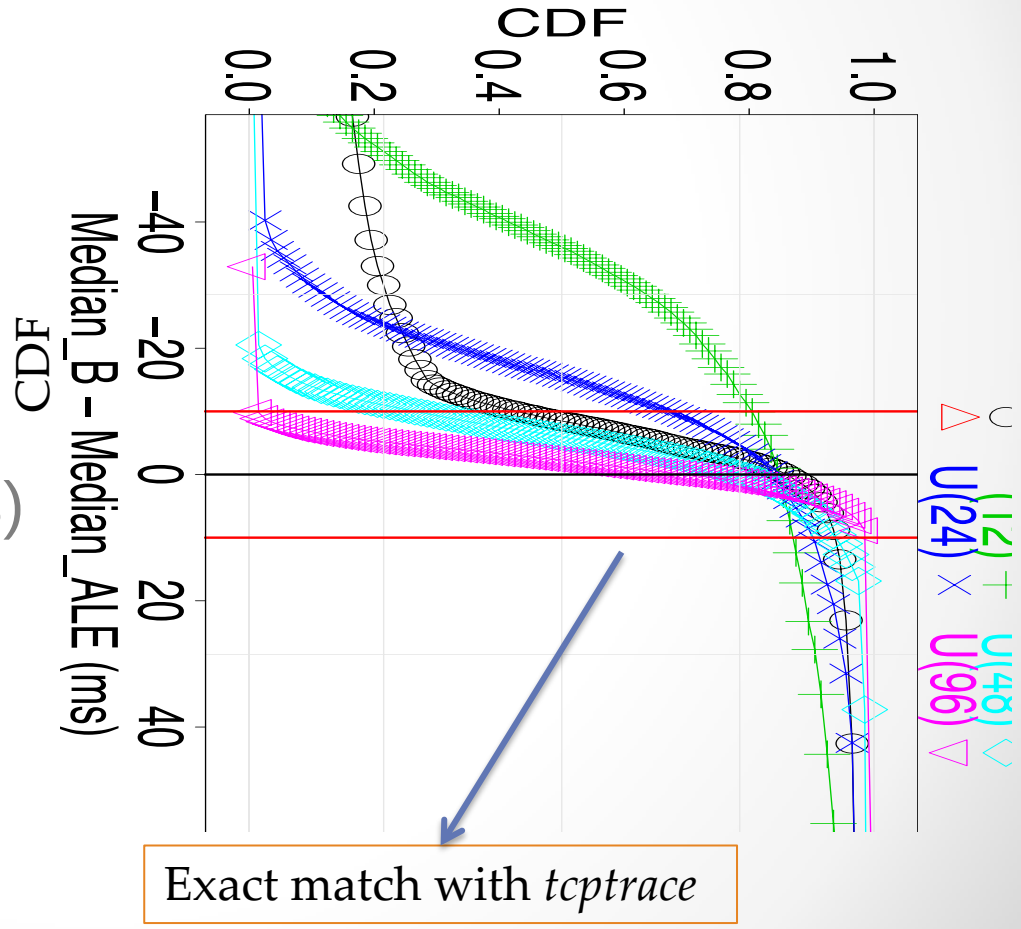
Accuracy: RTT Samples



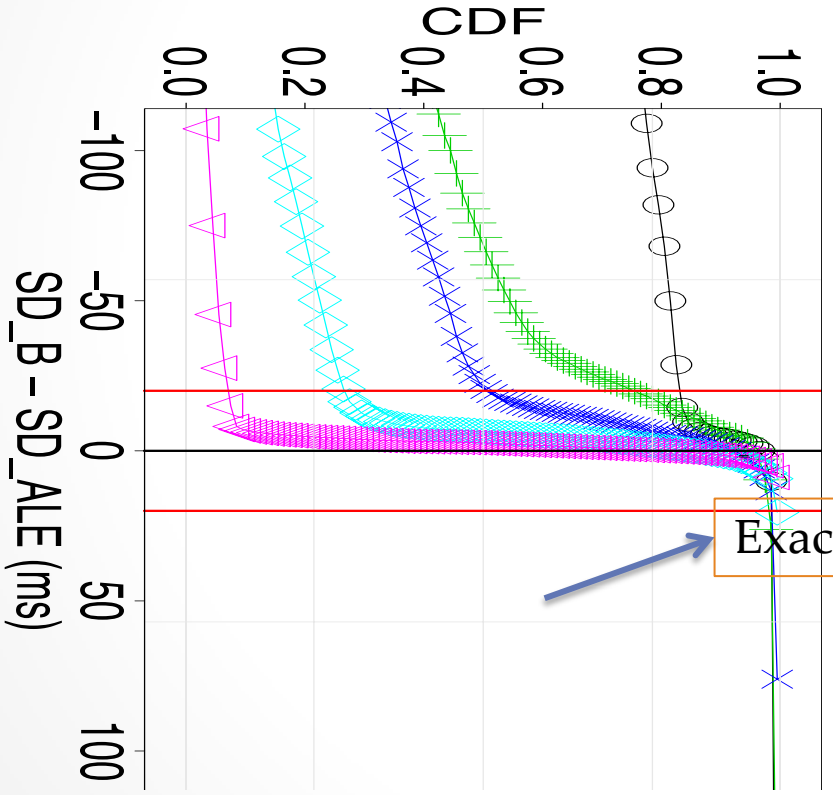
Accuracy: Flow Statistics

- Many applications use aggregated flow statics

- Small *w* (more buckets)
  - Median flow latencies approach *tcptrace*



Accuracy: Flow Statistics



Small w: Standard deviation of flow latencies approach *tcptrace*

Exact match with *tcptrace*

Compute and Memory Overhead

- Sample flows uniformly at random at rates 0.1, 0.2, . . . 0.7
  - 5 pcap sub-traces per rate
  - Higher sampling rate (data rate) ⇒ More state for *tcptrace*
- GNU Linux taskstats API
  - Compute time and memory usage

Compute Time

ALE scales with increasing data rates

Memory Overhead

TCPTRACE

- RSS memory: ≈64 MB (0.1) to ≈460 MB (0.7)
- VSS memory: ≈74 MB (0.1) to ≈468 MB (0.7)

ALE-U(96)

- RSS memory: 2.0 MB for all Sampling rates
- VSS memory: 9.8 MB for all sampling rates

ALE uses a fixed size data structure

Conclusions

- Current TCP measurements emulate state
  - Expensive: high data rates, constrained devices
- ALE provides low overhead data structure
  - Sliding window of CBF buckets
- Improvements over compute and memory with *tcptrace* sacrificing small accuracy
  - Tunable parameters
- Simple hash functions and counters
  - Hardware Implementation

Thank You

ALE Compute and Memory Bounds

- Insertion: *O(h)* time
  - CBF with *h* hash functions
- Matching ACK number: *O(hn)* time
  - *'n'* buckets
- Shifting buckets: *O(1)* time
  - Linked list of buckets
- ALE-E: *O(C)* time to merge CBFs
  - 'C' counters in the CBF
- Memory usage: $n \times C \times d$ bits
  - 'd' bit CBF counters

ALE Error Bounds

- ALE-U: Average case error $w/4$
- ALE-U: Worst case error $w/2$

- ALE-E: Average case error $(3w/16)2^i$
- ALE-E: Worst case error $(2^{i-1}w)$
  - ACK has a match in bucket i

ALE Parameters

- '*w*' on accuracy requirements
- 'W' estimate of the maximum latencies expected
- CBFs with $h = 4$ hash functions
- C on traffic rate, w, false positive rate and h

- E.g., For $w = 20$ ms, $h = 4$, and $m = R \times w$, the constraint for optimal h ($h = (C/m) \ln 2$) yields $C = 40396$ counters

Compute Time

10 min trace

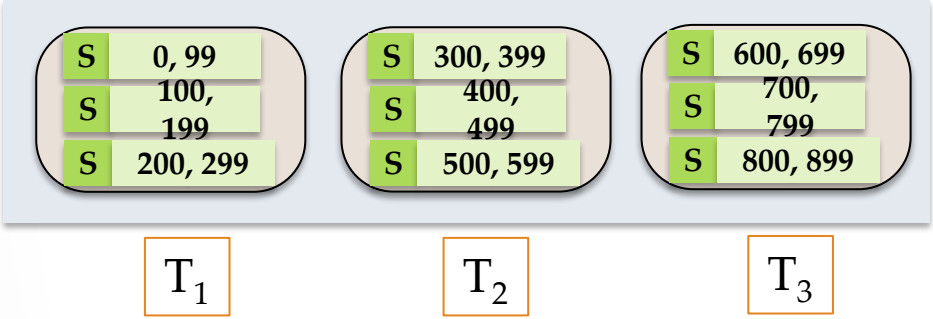
ALE scales with increasing data rates

Accuracy: RTT Samples

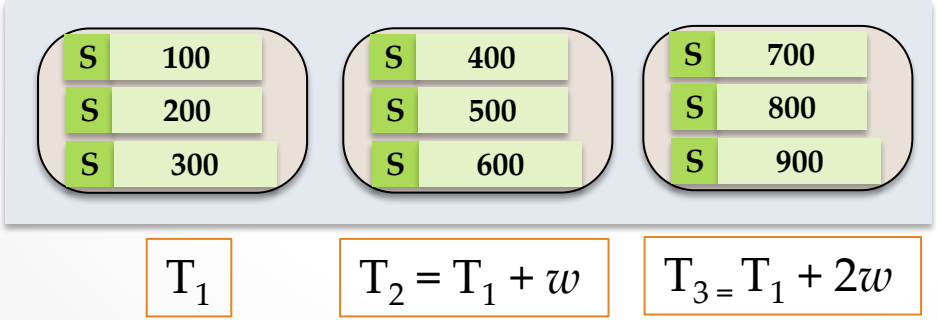
Memory Overhead

ALE Overview

- Eliminate packet timestamps
  - Time quantization: Use fixed number of intervals

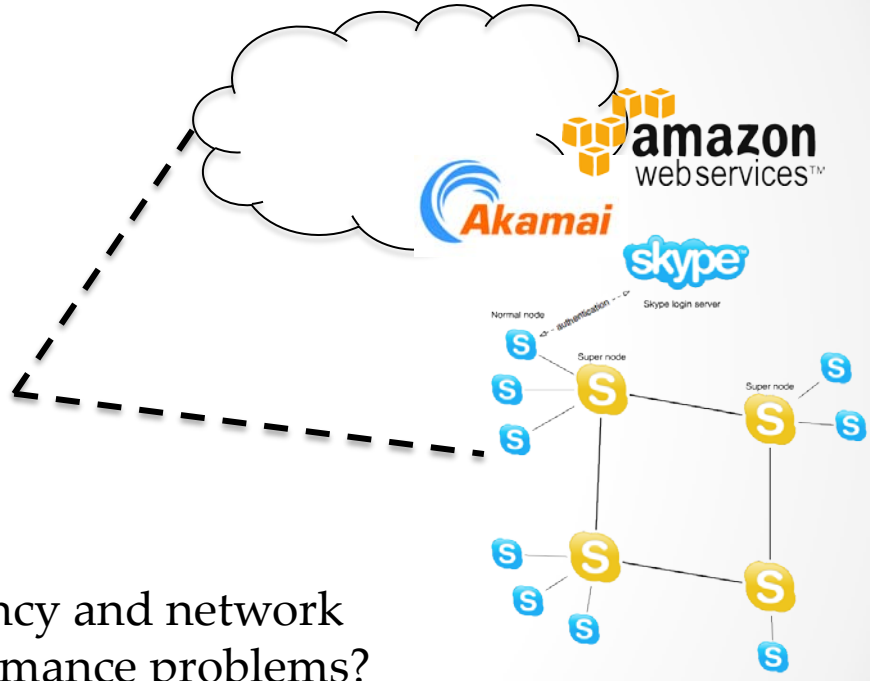


- Store expected acknowledgement number



Motivation: Network Troubleshooting

Latency Sensitive Applications



Latency and network performance problems? Where? Why?