


AWStream: Adaptive Wide-Area Streaming Analytics

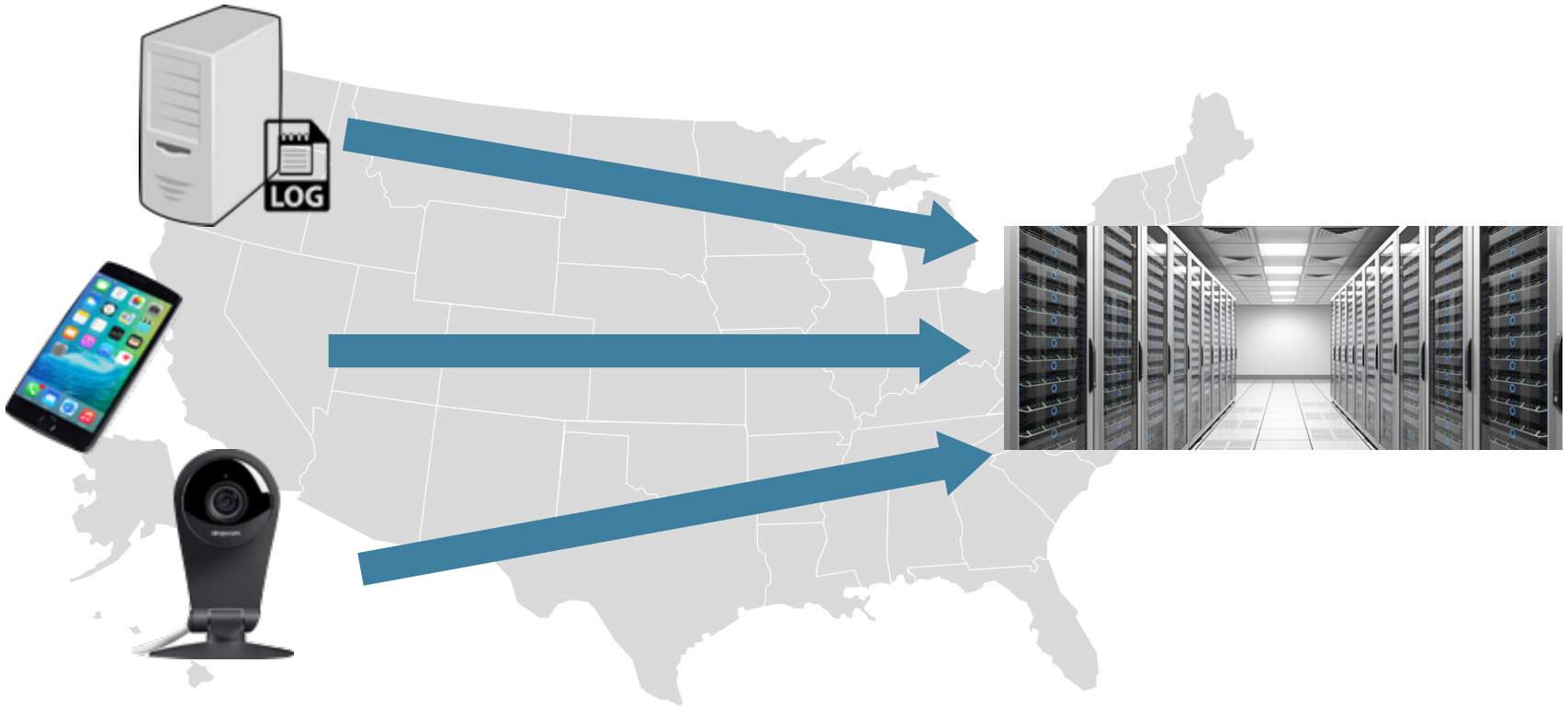
Ben Zhang, Xin Jin, Sylvia Ratnasamy

John Wawrzynek, Edward A. Lee

Presented by Radhika Mittal (not a co-author)

PowerPoint File is available at:
https://awstream.github.io/talk/talk.pptx

Wide-Area Streaming Analytics



Demand

Huge data generated at the edge

Resource

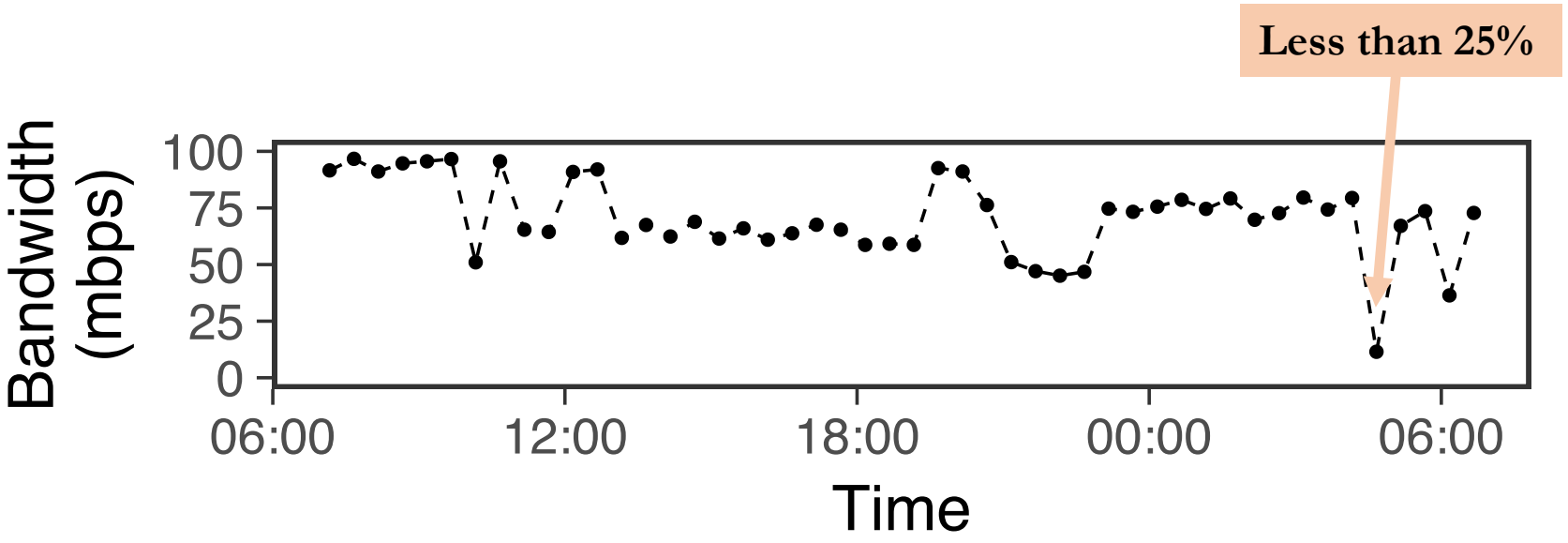
Scarce and varying WAN bandwidth

Demand: Huge Data at the Edge

- Machine logs, 25 TB daily at Facebook (2009)
- Electrical grid monitoring, 1.4 million data points per second [Andersen and Culler, FAST '16]
- Video surveillance, 3 mbps per camera [Amerasinghe, 2009]

- . . . Dropcam, a WiFi video-streaming camera and associated cloud backend service for storing and watching the resulting video. Dropcam has the fewest clients (2,940) . . .. Yet, each client uses roughly 2.8 GB a week and uploads nearly 19 times more than they download, implying that Dropcam users do not often watch what they record. [Biswas, SIGCOMM '15]

Resource: Scarce and Varying WAN Bandwidth



Bandwidth variations throughout the day between Amazon EC2 sites. Similar scarcity and variation for wireless networks, broadband access networks and cellular networks.

What happens when bandwidth becomes insufficient?

- TCP ensures data delivery, but hurts latency
- UDP sends fast, suffering uncontrolled packet loss
- Manual policies (developer heuristics) are sub-optimal
  - JetStream [Rabkin et al., NSDI 14] uses manual policy
  - "if bandwidth is insufficient, switch to sending images at 75% fidelity, then 50% if still not enough"
- Application-specific optimizations don't generalize
  - (more on the next slide)

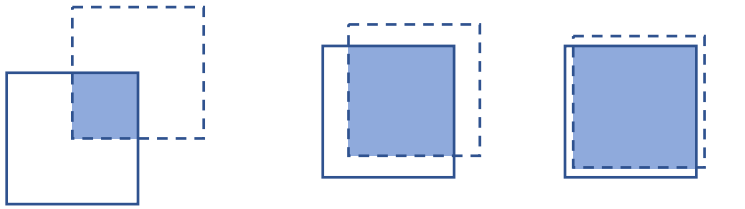
Application-specific optimizations don't generalize

For a surveillance application that detects pedestrians on a busy street

t=0s, small target in far-field views

t=1s, small difference

Positive if intersection over union (IOU) larger than 0.5.



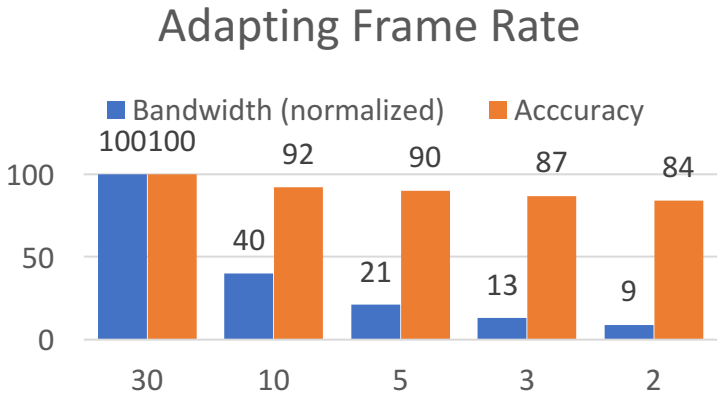
IOU=0.2 IOU=0.5 IOU=0.8

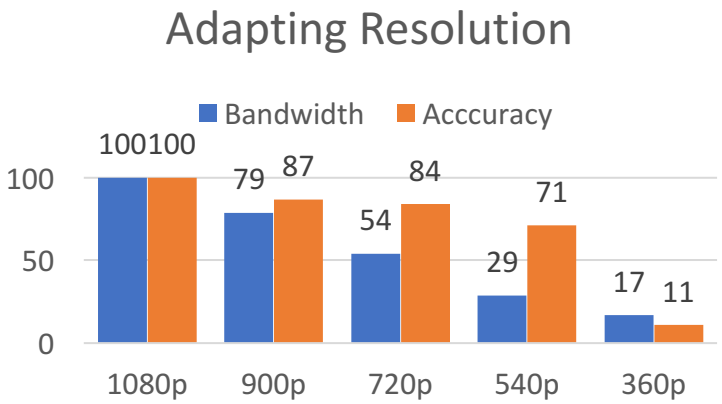
Application-specific optimizations don't generalize

For a surveillance application that detects pedestrians on a busy street

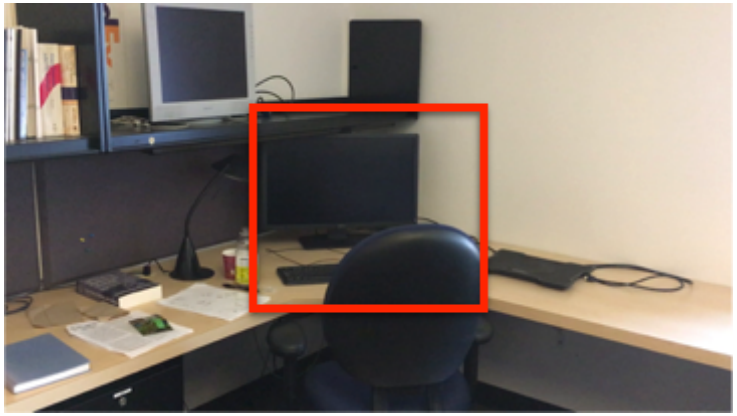
t=0s, small target in far-field views

t=1s, small difference

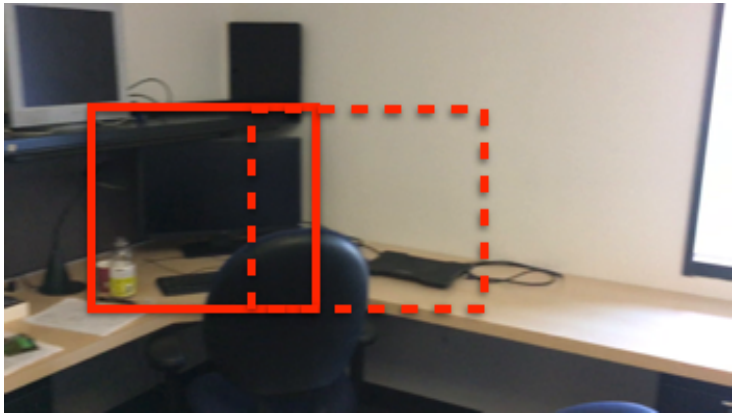




Application-specific optimizations don't generalize

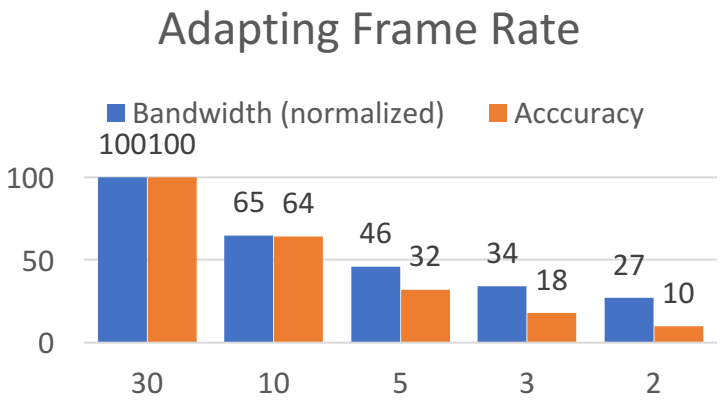


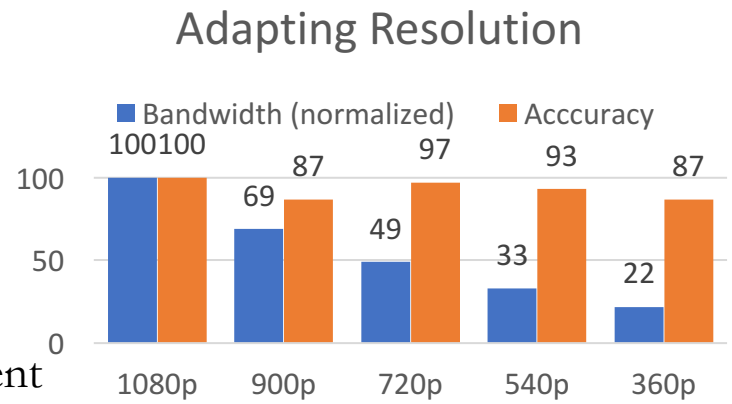
t=0s, nearby and large targets



t=1s, large difference due to camera movement

For an application that detects objects on a mobile phone

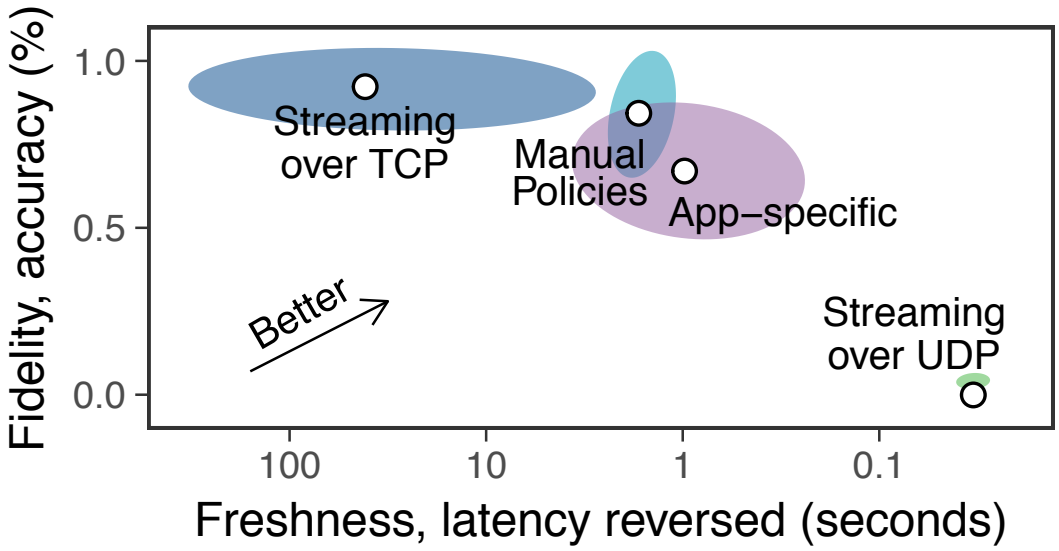




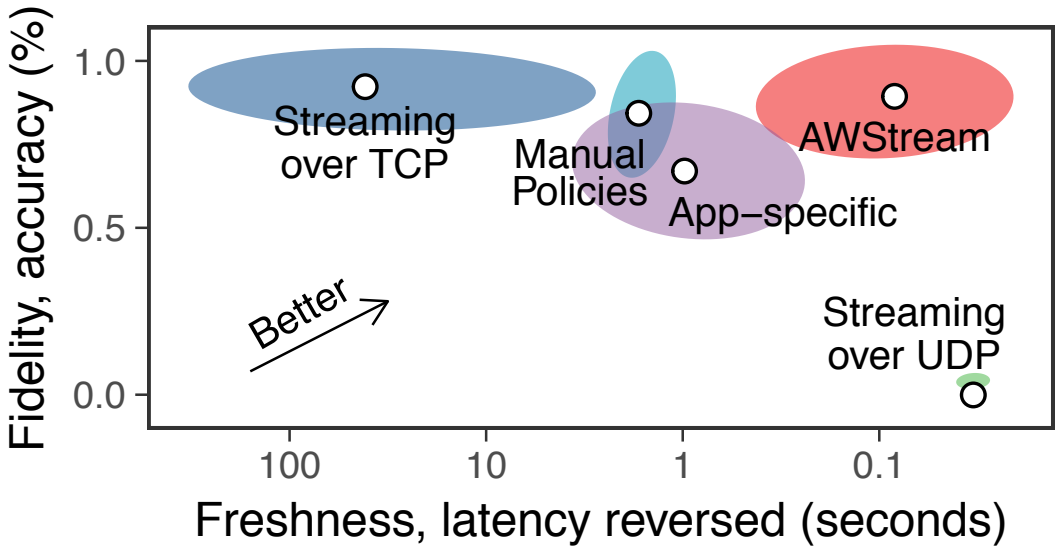
What happens when bandwidth becomes insufficient?

- TCP ensures data delivery, but hurts latency
- UDP sends fast, suffering uncontrolled packet loss
- Manual policies (developer heuristics) are sub-optimal
  - JetStream [Rabkin et al., NSDI 14] uses manual policy
  - "if bandwidth is insufficient, switch to sending images at 75% fidelity, then 50% if still not enough"
- Application-specific optimizations don't generalize
  - Adaptation often requires expertise and manual work to explore multidimensional adaptation for each application

Fidelity vs. Freshness



Fidelity vs. Freshness

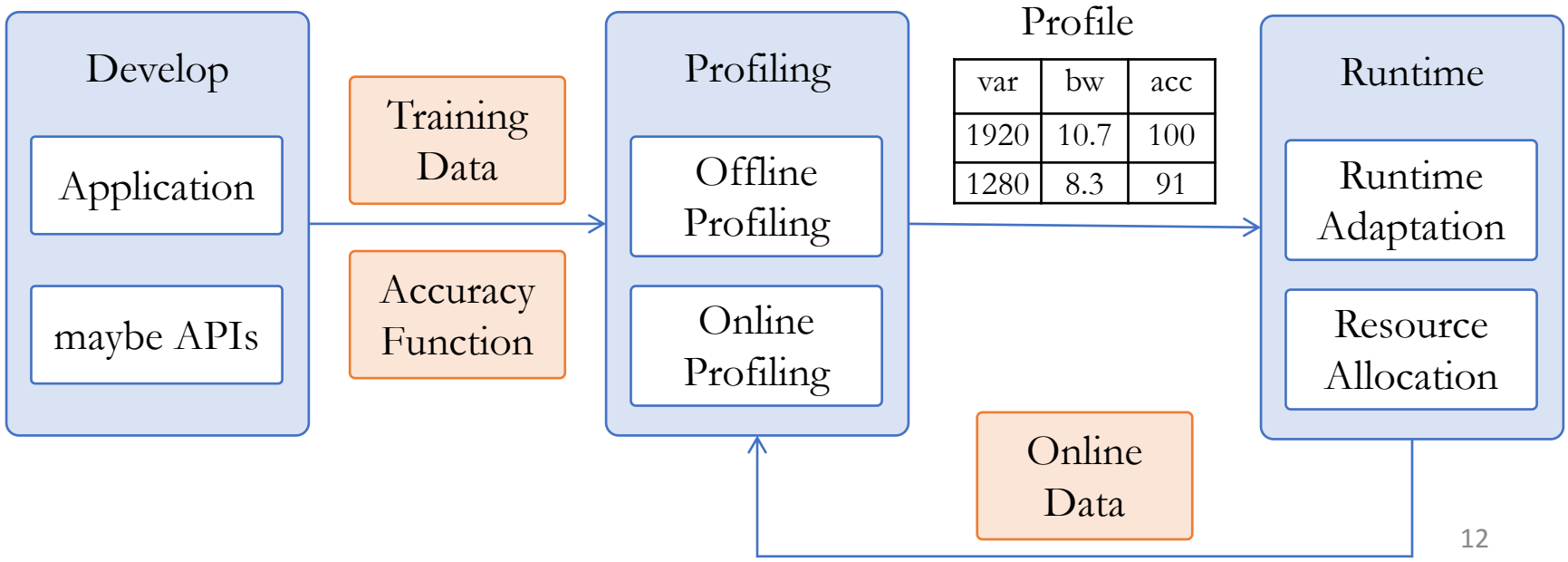


- Applications must be adaptive.
- Adaptation policies must be,
  - precise
  - automatically generated
  - for each application

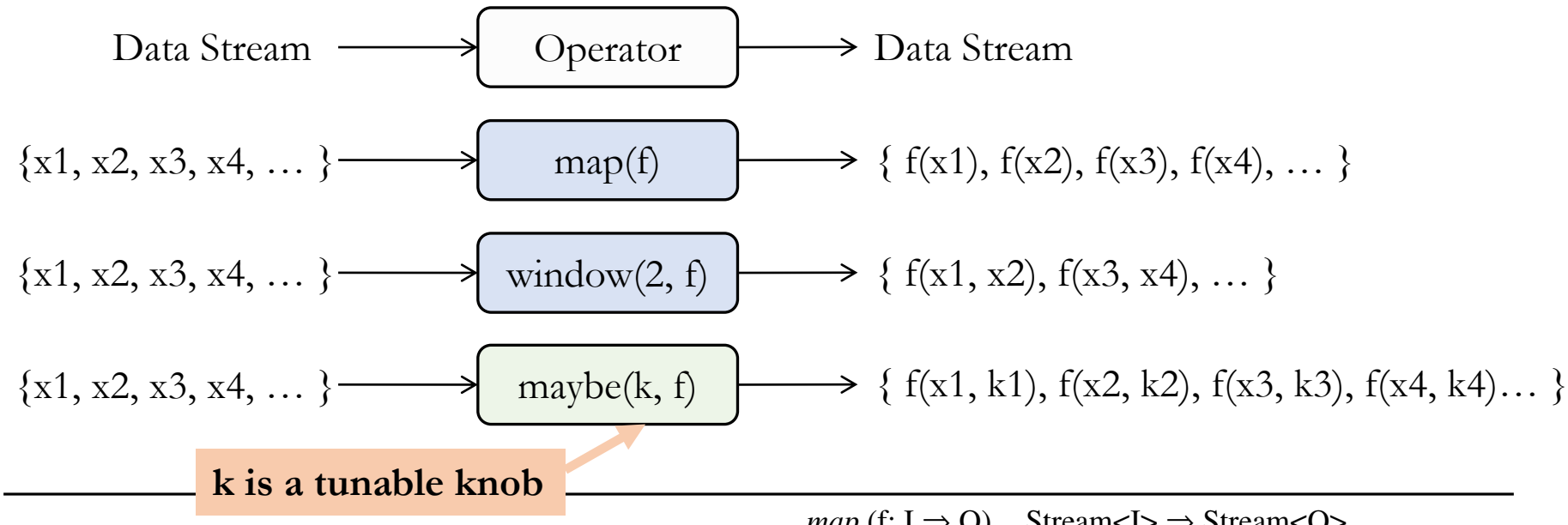
Pareto-optimal Profile: maximizing application accuracy while satisfying bandwidth requirement (avoid congestion)

AWStream Overview

- Systematic and quantitative adaptation
  - New programming abstractions to express adaptation
  - Automatic data-driven profiling
  - Runtime adaptation balancing freshness and fidelity



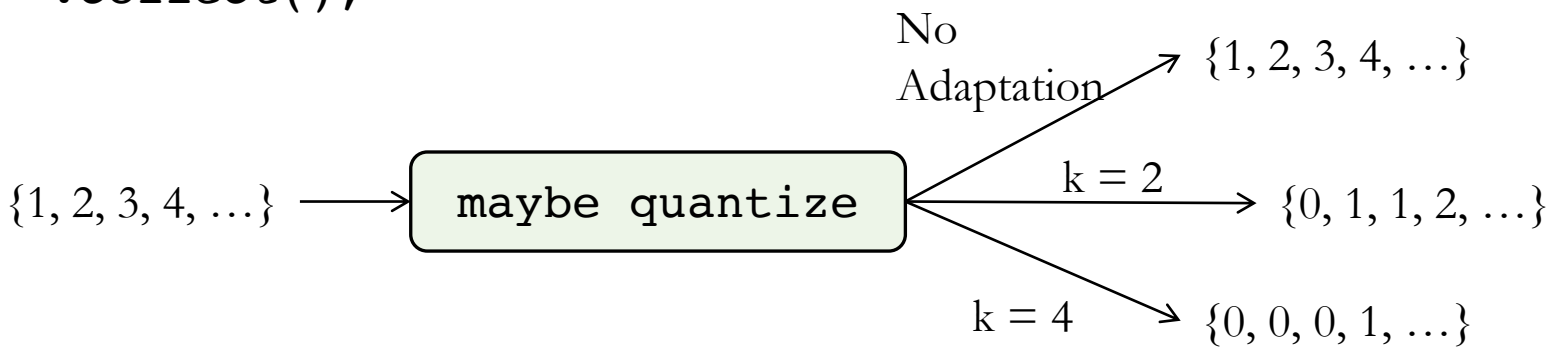
(1) Streaming Operators and APIs



Normal Operators	*map* (f: I ⇒ O)	Stream<I> ⇒ Stream<O>
	skip (i: Integer)	Stream<I> ⇒ Stream<I>
	sliding_window (count: Integer, f: Vec<I> ⇒ O)	Stream<I> ⇒ Stream<O>
	…	…
Degradation Operators	*maybe* (knobs: Vec<T>, f: (T, I) ⇒ I)	Stream<I> ⇒ Stream<I>

(1) maybe APIs in use

```
let quantized = vec![1, 2, 3, 4].into_stream()
    .maybe(vec![2, 4], |k, val| val / k)
    .collect();
```




```
let app = Camera::new((1920, 1080), 30)
    .maybe_downsample(vec![(1600, 900), (1280, 720)])
    .maybe_skip(vec![2, 5])
    .map(|frame| pedestrian_detect(frame))
    .compose();
```

(2) Data-driven profiling

```
let app = Camera::new((1920, 1080), 30)
    .maybe_downsample(vec![(1600, 900), (1280, 720)])
    .maybe_skip(vec![2, 5])
    .map(|frame| pedestrian_detect(frame))
    .compose();
```

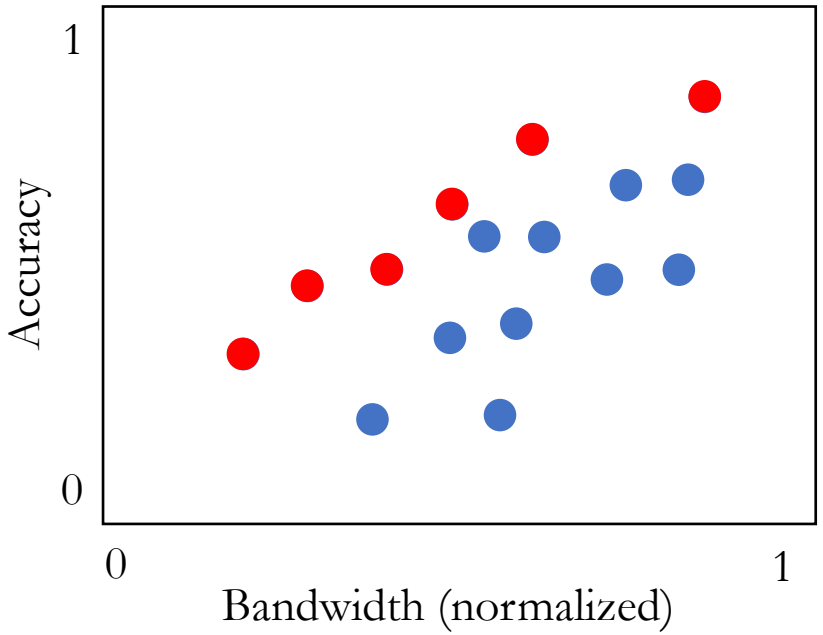
Training Data

Accuracy Function

downsample	skip	bandwidth	accuracy
(1920, 1080)	0	10.7	1.0
(1600, 900)	0	8.3	0.88
(1280, 720)	0	6.3	0.87
(1920, 1080)	2	9.3	0.90
…	…	….	…

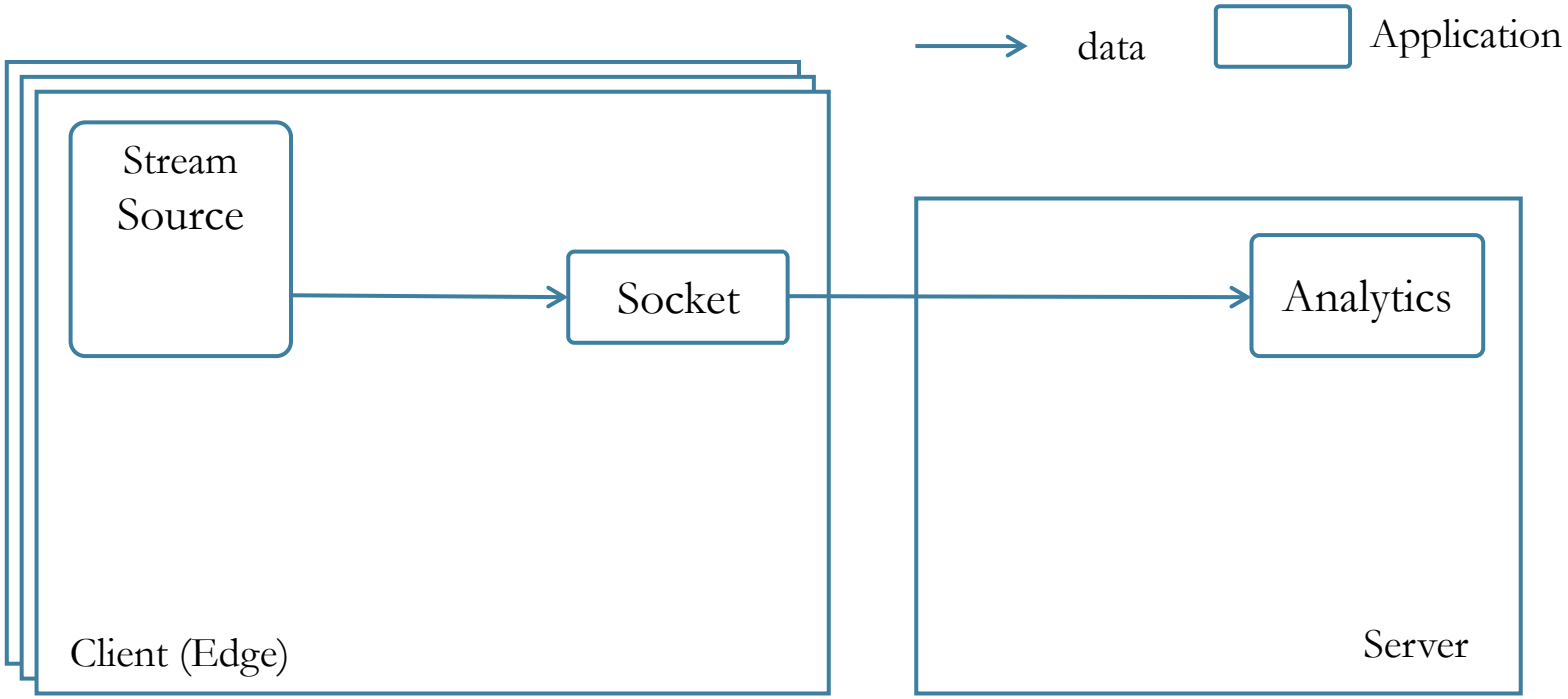
(2) Profile: Pareto-optimal Strategy

configuration	bandwidth	accuracy
c1	10.7	1.0
c2	8.3	0.88
c3	6.3	0.87
c4	9.3	0.90
…	….	…

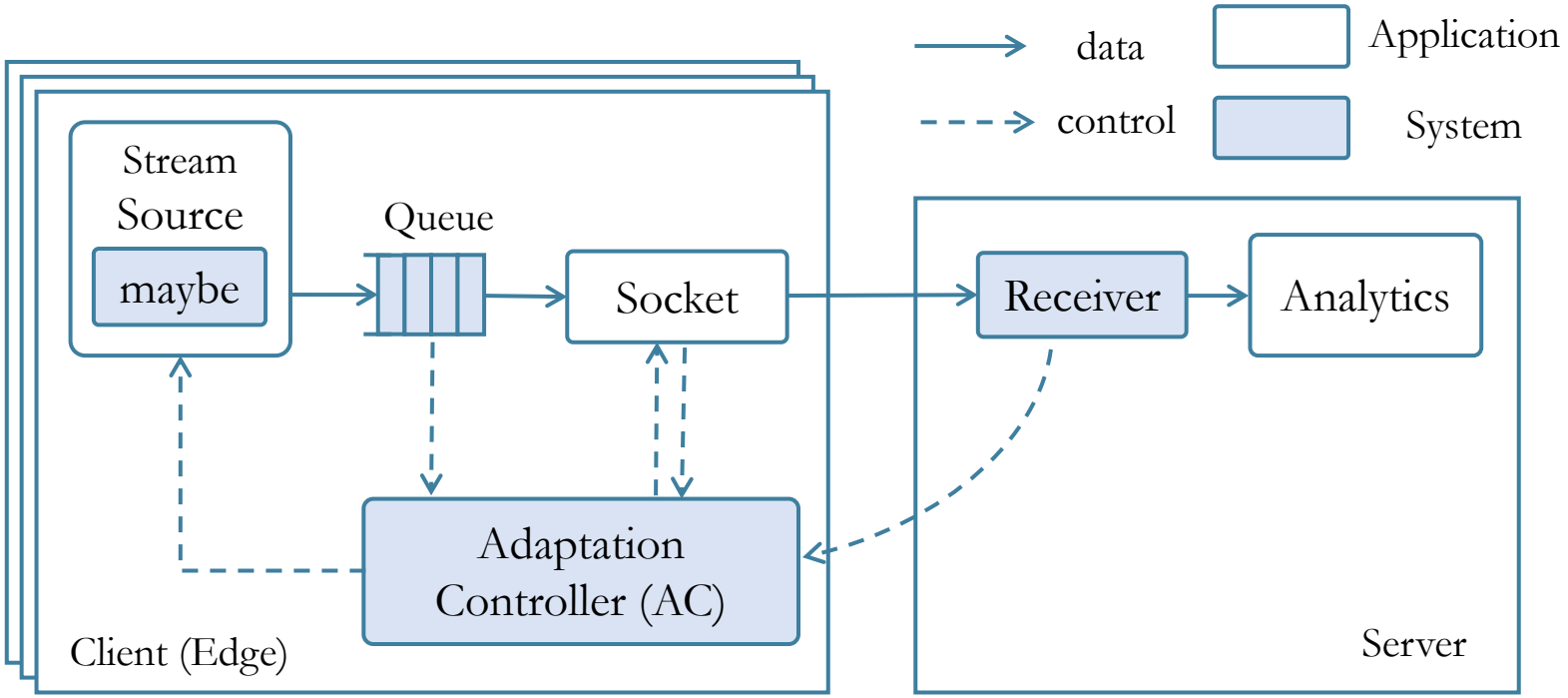


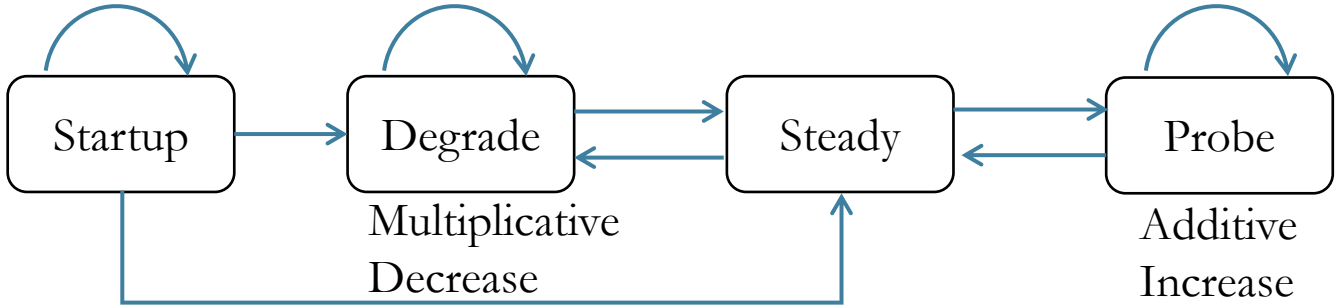
$$\mathbb{P} = \{c \in \mathbb{C} : \underbrace{\{c' \in \mathbb{C} : B(c') < B(c), A(c') > A(c)\}}_{\text{the set of better configuration c'}} = \varnothing\}$$

(3) Runtime Adaptation

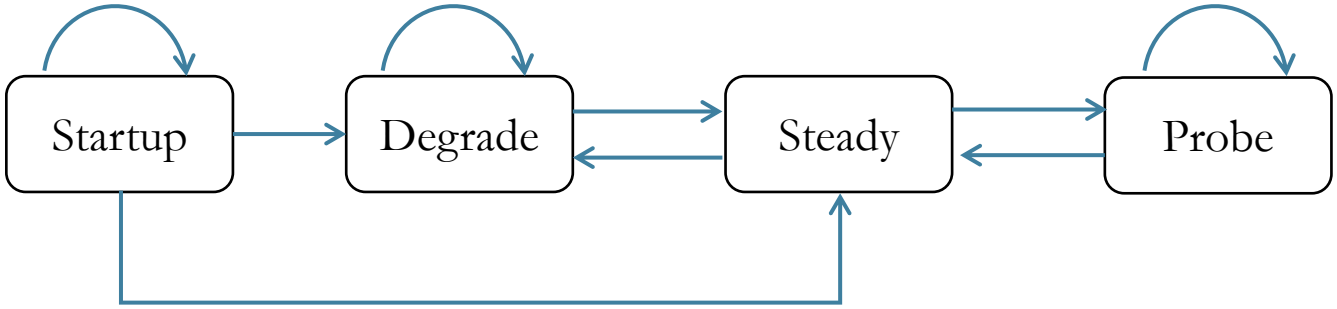


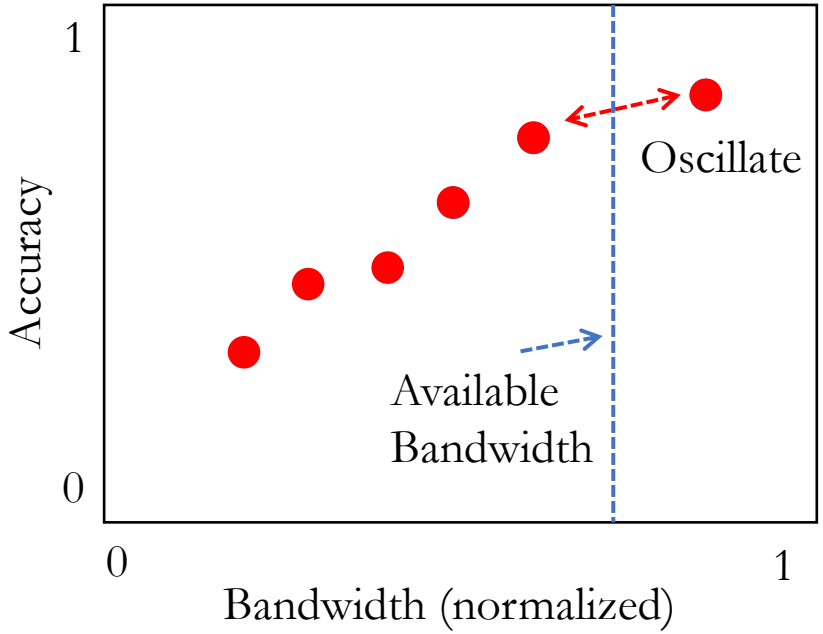
(3) Runtime Adaptation





(3) Probing at Runtime



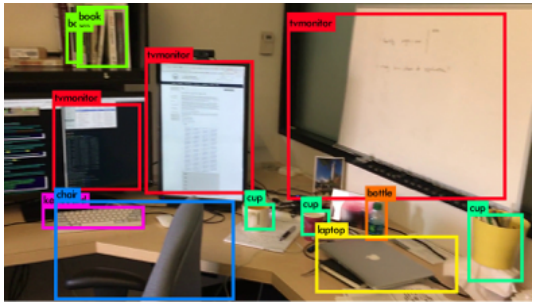


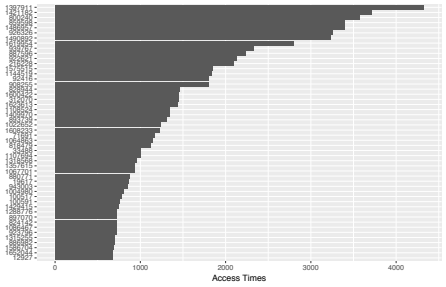
- These configurations are discrete
- Without probing, applications can jump to the next configuration that demands too much bandwidth. They end up oscillate between configurations.
- Probing (with dummy traffic) stabilizes adaptation.

Applications

Quantization is a parameter exposed from the video encoder (H.264)

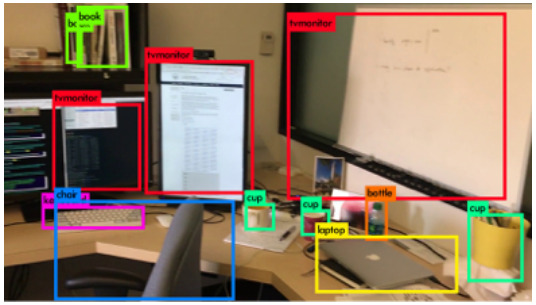
Application	Knobs	Accuracy	Dataset
Augmented Reality	Resolution Frame rate Quantization	F1 Score [Rijsbergen, 1979]	iPhone video clips Training: office, 24s Testing: home, 240s
Pedestrian Detection	Resolution Frame rate Quantization	F1 Score	MOT 16 [Milan et al., 2016] Training: MOT 16-04 Testing: MOT 16-03
Log Analysis (Top-K, K=50)	Head(N) Threshold(T)	Kendall's Tau [Abdi, 2007]	SEC.gov logs Training: 4 days Testing: 16 days

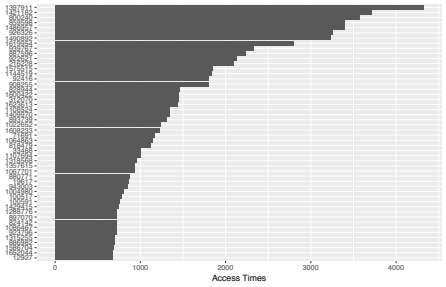




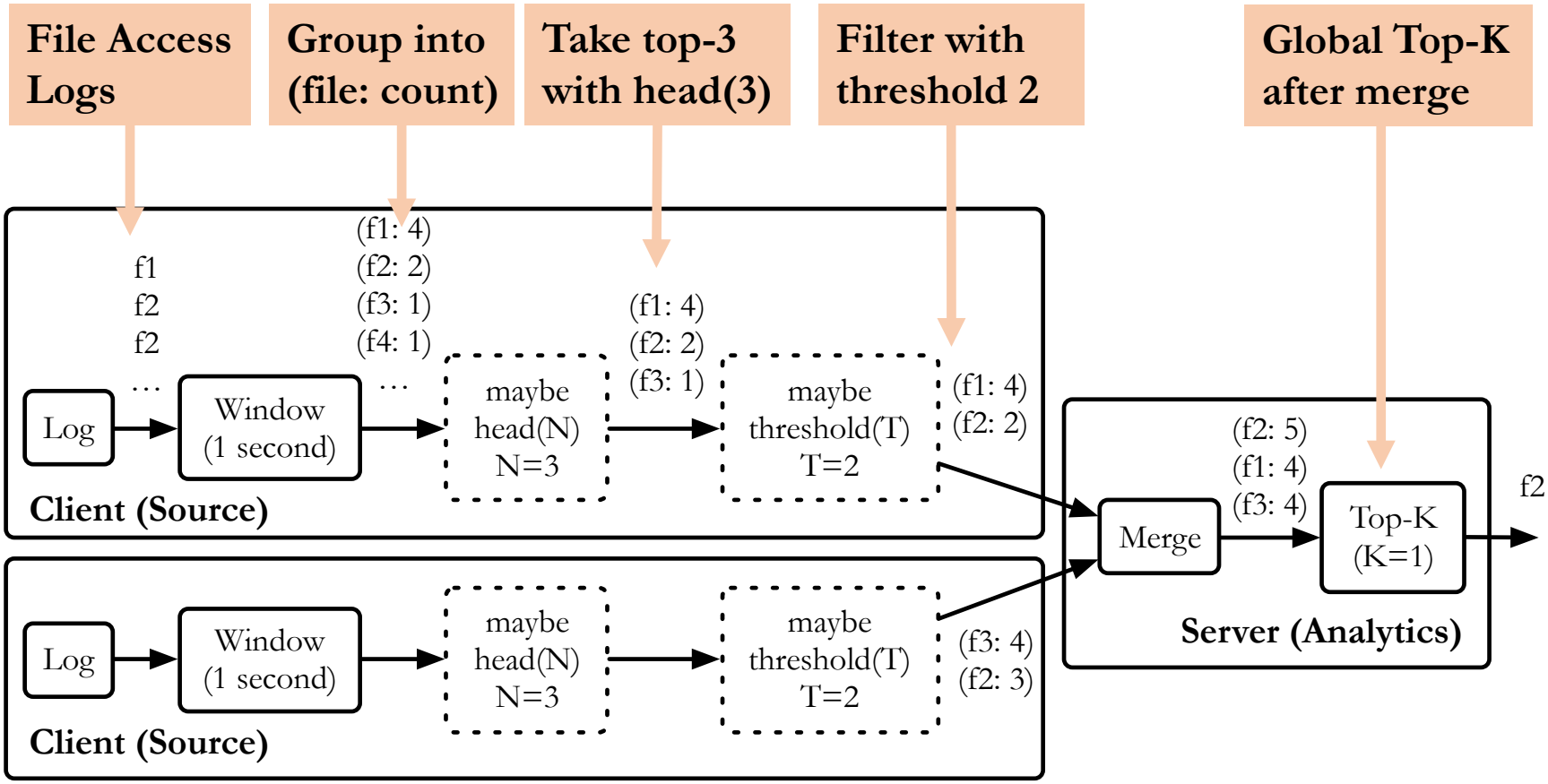
Applications

Application	Knobs	Accuracy	Dataset
Augmented Reality	Resolution Frame rate Quantization	F1 Score [Rijsbergen, 1979]	iPhone video clips Training: office, 24s Testing: home, 240s
Pedestrian Detection	Resolution Frame rate Quantization	F1 Score	MOT 16 [Milan et al., 2016] Training: MOT 16-04 Testing: MOT 16-03
Log Analysis (Top-K, K=50)	Head(N) Threshold(T)	Kendall's Tau [Abdi, 2007]	SEC.gov logs Training: 4 days Testing: 16 days

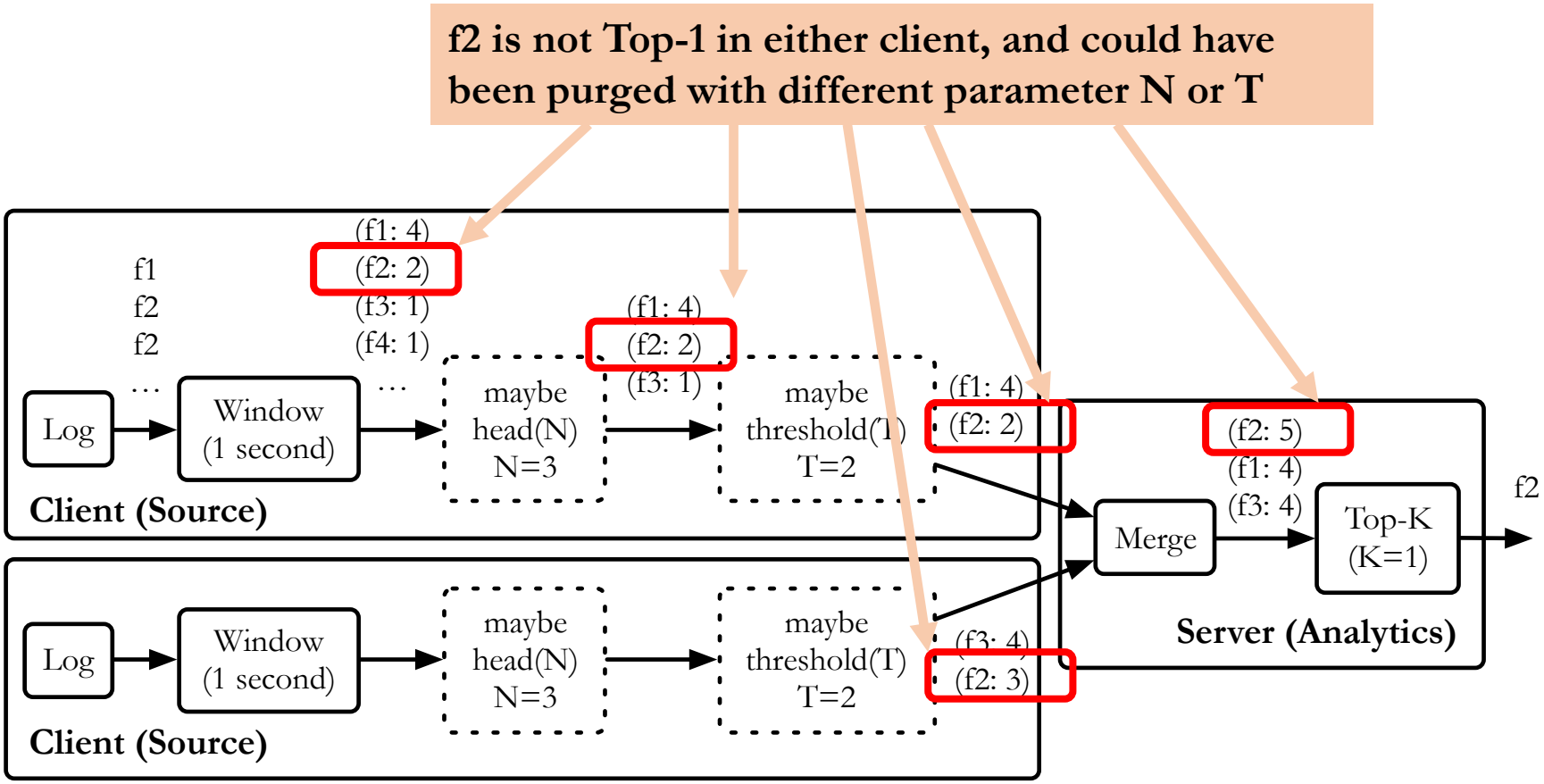




Adaptation in Top-K



Adaptation in Top-K

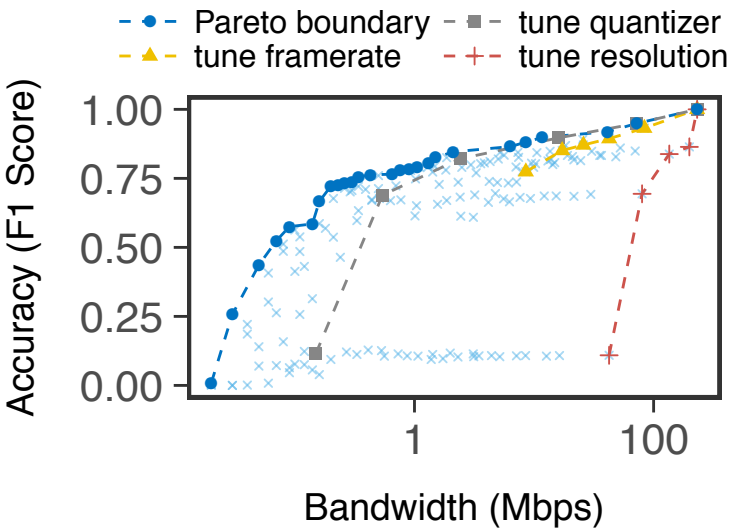


Evaluations

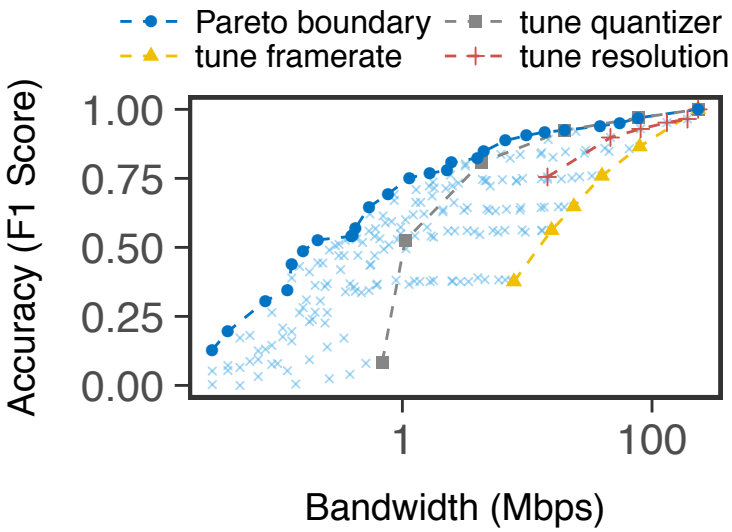
- Can AWStream generate accurate profiles across multiple dimensions?
- Can AWStream profile efficiently and support online profiling?
- Can AWStream runtime improve data freshness and fidelity when facing insufficient bandwidth?
- Can we use the profiles to guide bandwidth allocations among multiple applications?

Profiles across multiple dimensions

Pedestrian Detection

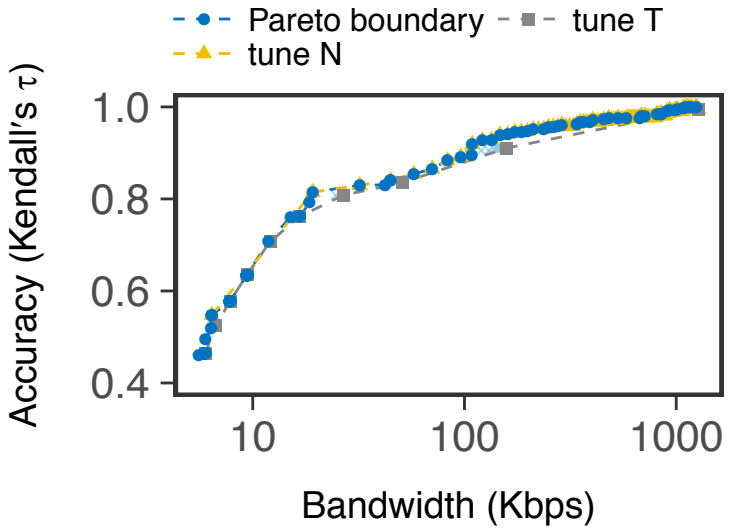


Augmented Reality



- Optimal strategy needs multiple dimensions
- For the same application, different dimensions have different impact.
- For different applications, the same dimension has different impact.

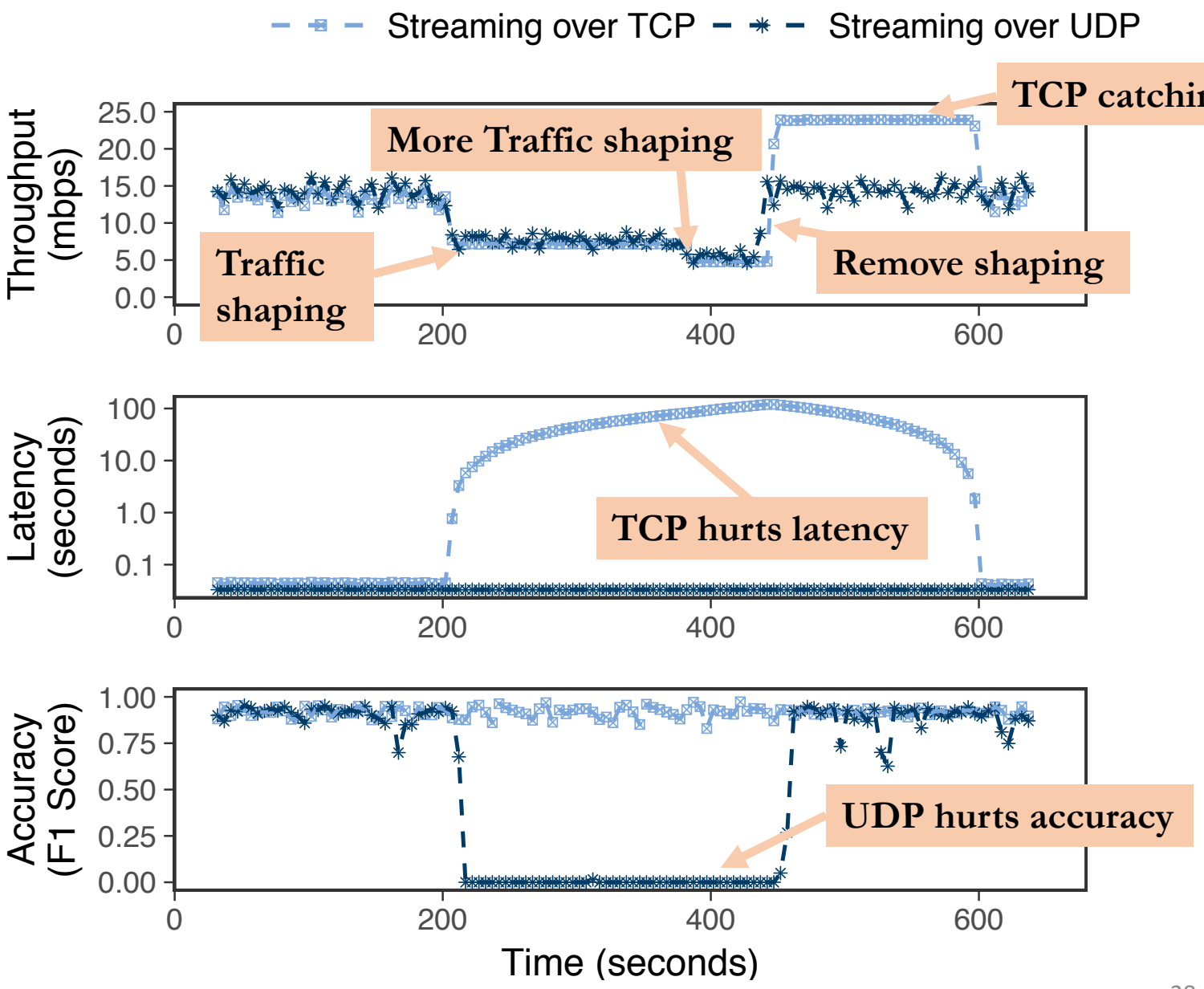
Profiles are precise

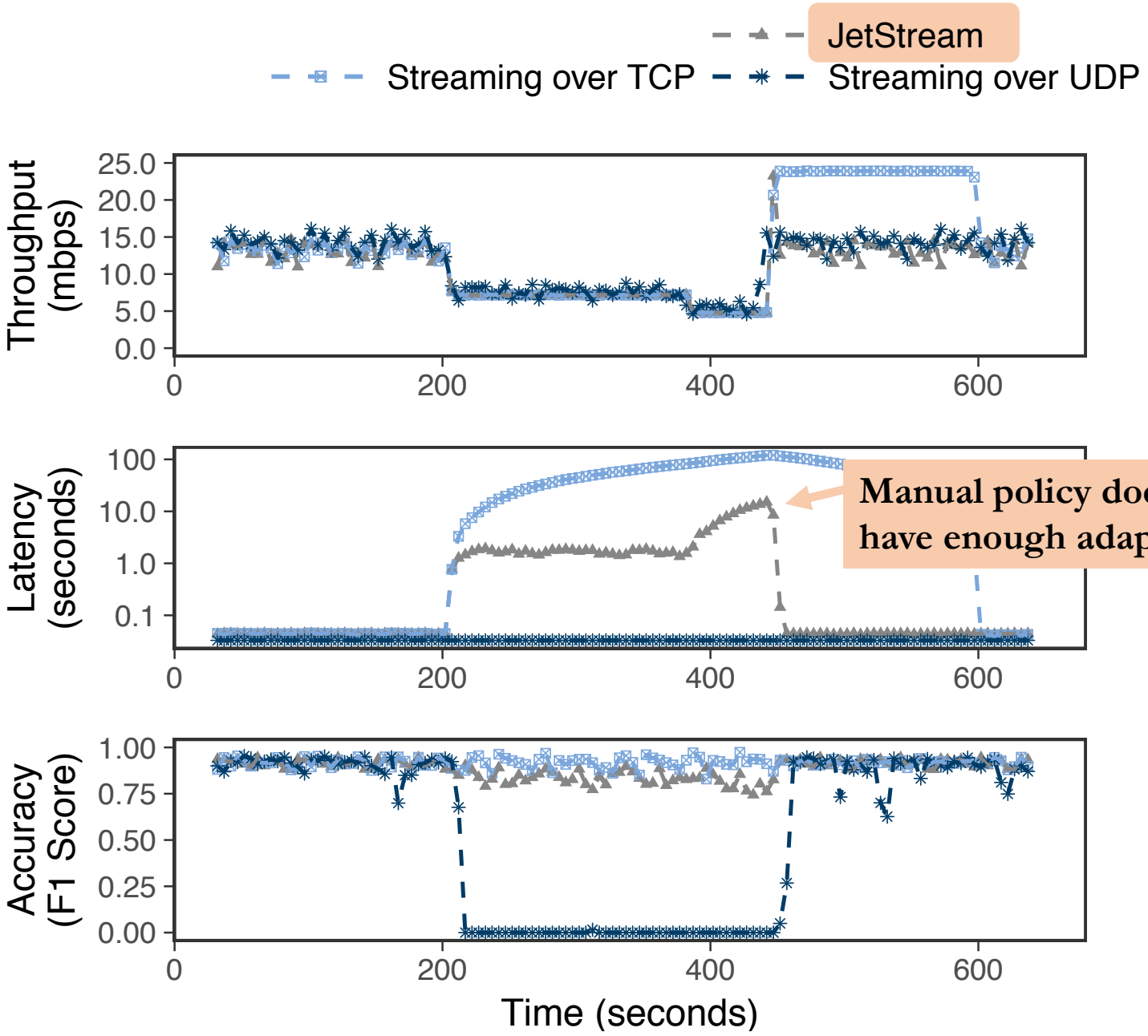


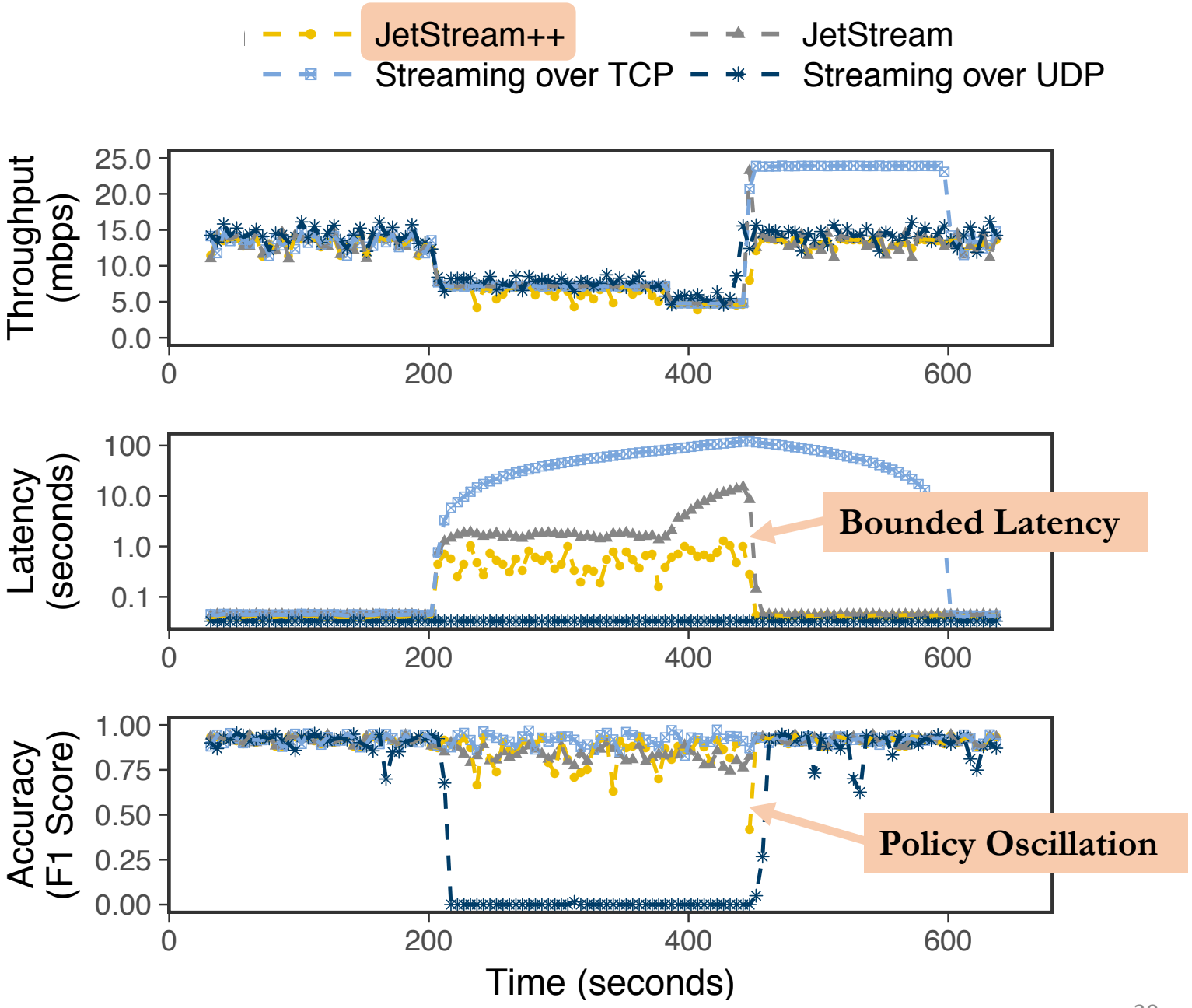
- The effect of each dimension is not significantly different.
- The profile offers quantified effects of adaptation options.

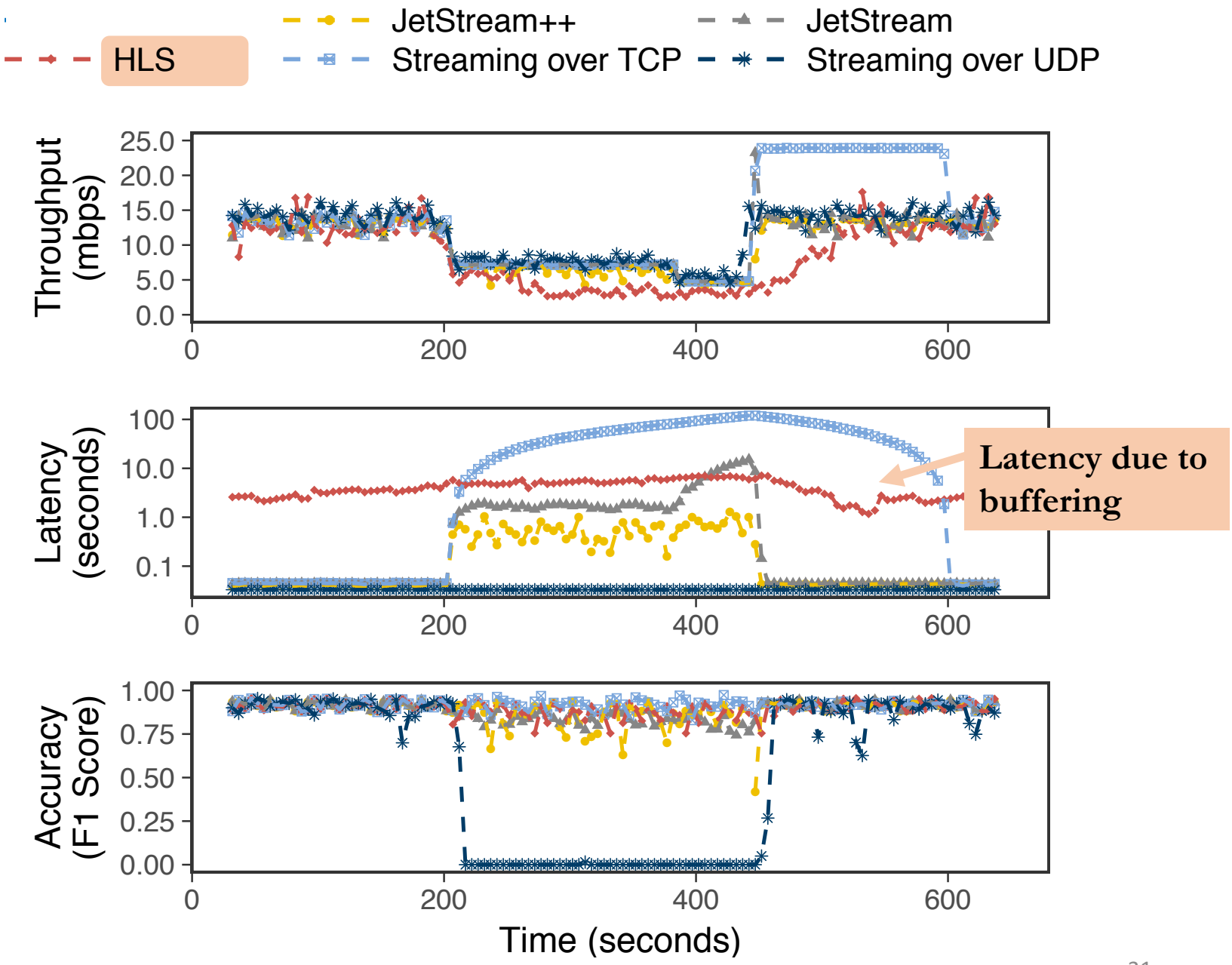
Runtime Performance Baselines

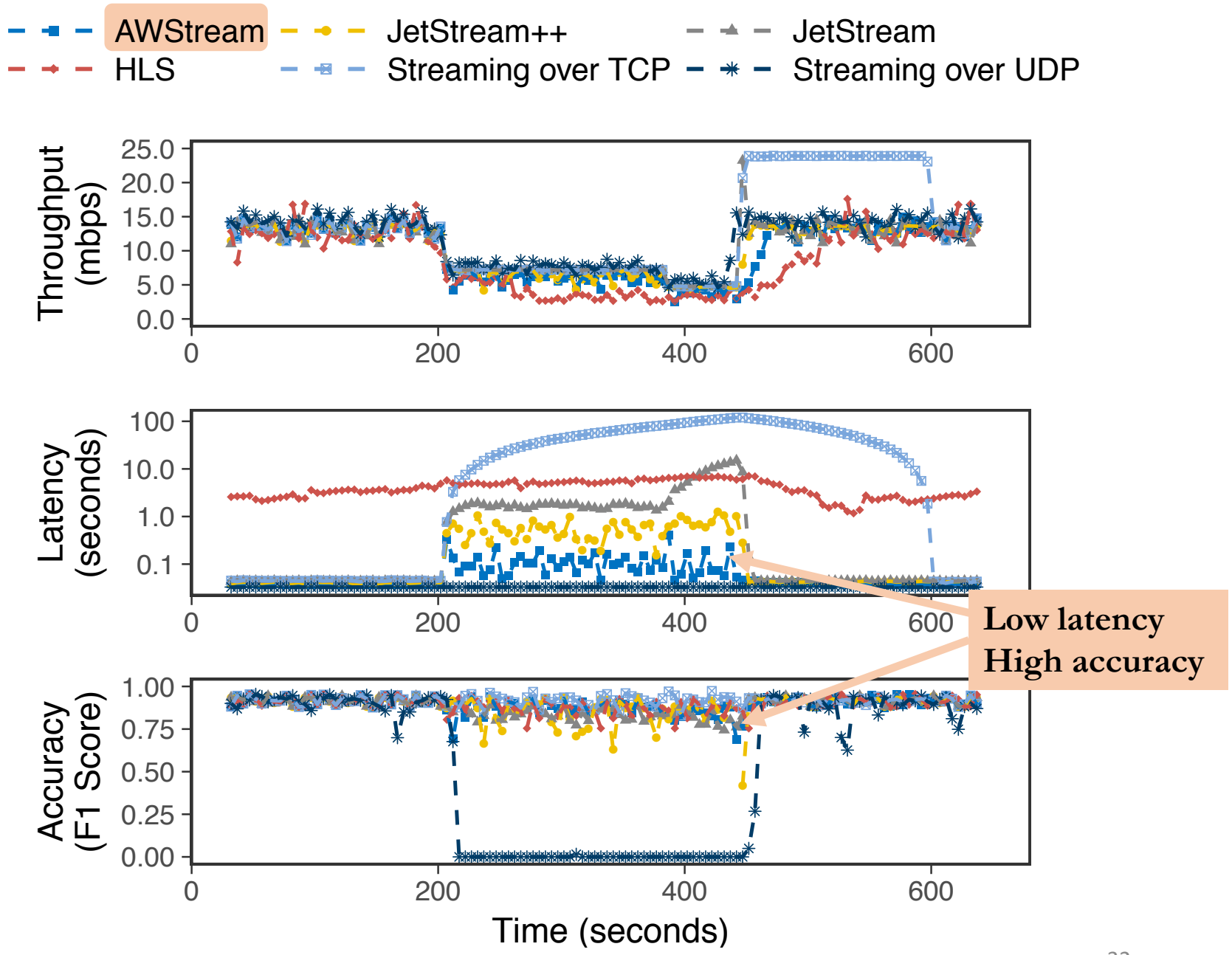
Baseline	Description
Streaming over TCP	A non-adaptive approach
Streaming over UDP	A non-adaptive approach, representing RTP/UDP/RTSP video streaming
JetStream [Rabkin et al., 2014]	Manual Policy: "if bandwidth is insufficient, switch to sending images at 75% fidelity, then 50% if there still isn't enough bandwidth. Beyond that point, reduce the frame rate, but keep the image fidelity."
JetStream++	Uses adaptation policy generated by AWStream. JetStream runtime does not probe (hence may oscillate between policies).
HLS [Pantos and May, 2016]	HTTP Live Streaming represents popular adaptive video streaming techniques; used for Periscope video stream [Wang et al., 2016].


Streaming over TCP
Streaming over UDP
Throughput (mbps)
25.0
20.0
15.0
10.0
5.0
0.0
0
200
400
600
More Traffic shaping
TCP catching up
Traffic shaping
Remove shaping
Latency (seconds)
100
10.0
1.0
0.1
TCP hurts latency
Accuracy (F1 Score)
1.00
0.75
0.50
0.25
0.00
UDP hurts accuracy
Time (seconds)

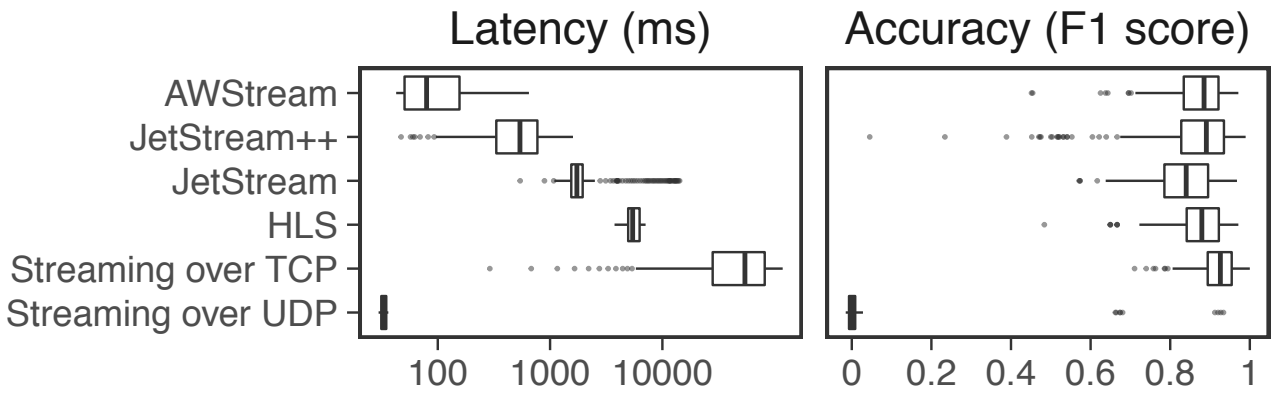

JetStream
Streaming over TCP
Streaming over UDP
Throughput (mbps)
25.0
20.0
15.0
10.0
5.0
0.0
0
200
400
600
Latency (seconds)
100
10.0
1.0
0.1
Manual policy does not have enough adaptation
Accuracy (F1 Score)
1.00
0.75
0.50
0.25
0.00
Time (seconds)

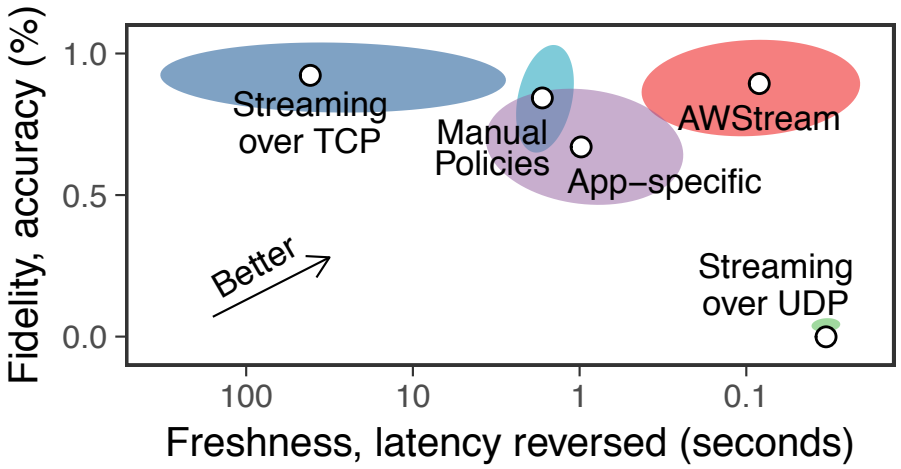

JetStream++
JetStream
Streaming over TCP
Streaming over UDP
Throughput (mbps)
25.0
20.0
15.0
10.0
5.0
0.0
0
200
400
600
Latency (seconds)
100
10.0
1.0
0.1
Bounded Latency
Accuracy (F1 Score)
1.00
0.75
0.50
0.25
0.00
Policy Oscillation
Time (seconds)


HLS
JetStream++
Streaming over TCP
JetStream
Streaming over UDP
Throughput (mbps)
25.0
20.0
15.0
10.0
5.0
0.0
0
200
400
600
Latency (seconds)
100
10.0
1.0
0.1
Latency due to buffering
Accuracy (F1 Score)
1.00
0.75
0.50
0.25
0.00
Time (seconds)


AWStream
JetStream++
JetStream
HLS
Streaming over TCP
Streaming over UDP
Throughput (mbps)
25.0
20.0
15.0
10.0
5.0
0.0
0
200
400
600
Latency (seconds)
100
10.0
1.0
0.1
Accuracy (F1 Score)
1.00
0.75
0.50
0.25
0.00
Time (seconds)
Low latency
High accuracy

Runtime Performance Summary





Conclusion

- The emerging wide-area streaming analytics
  - They are becoming pervasive with more IoT applications
  - They must address scarce and varying WAN bandwidth
- We present AWStream.
  - A systematic and quantitative approach towards adaptation
  - Novel APIs, automatic profiling, and runtime adaptation

- For more questions,
  - Contact: Ben Zhang, benzh@cs.berkeley.edu
  - Slides: https://awstream.github.io/talk/talk.pdf
  - Repository: https://github.com/awstream