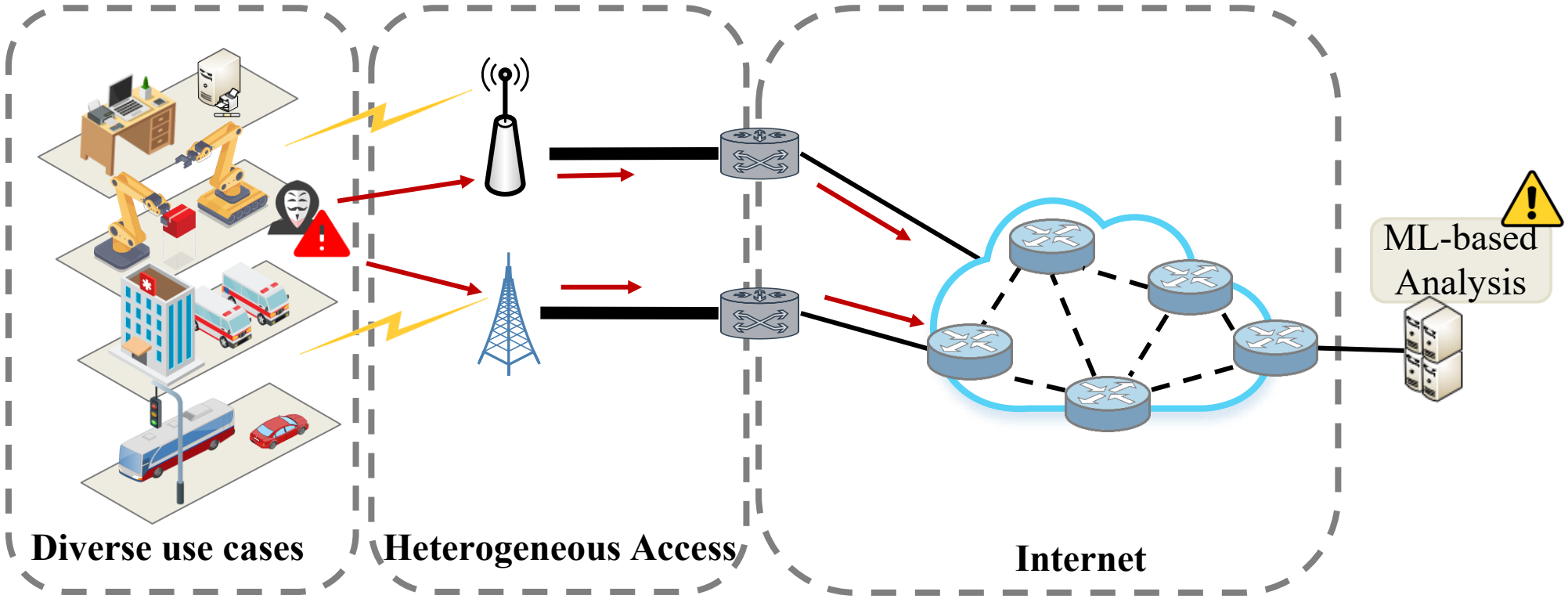
Dynamic Adaptation of In-Band Network Monitoring via Meta Learning

Mingyuan Zang*, Eder Ollora Zaballa‡, Lars Dittmann‡, Jie Wu*†

* China Telecom Cloud Computing Research Institute,

‡Technical University of Denmark, † Temple University

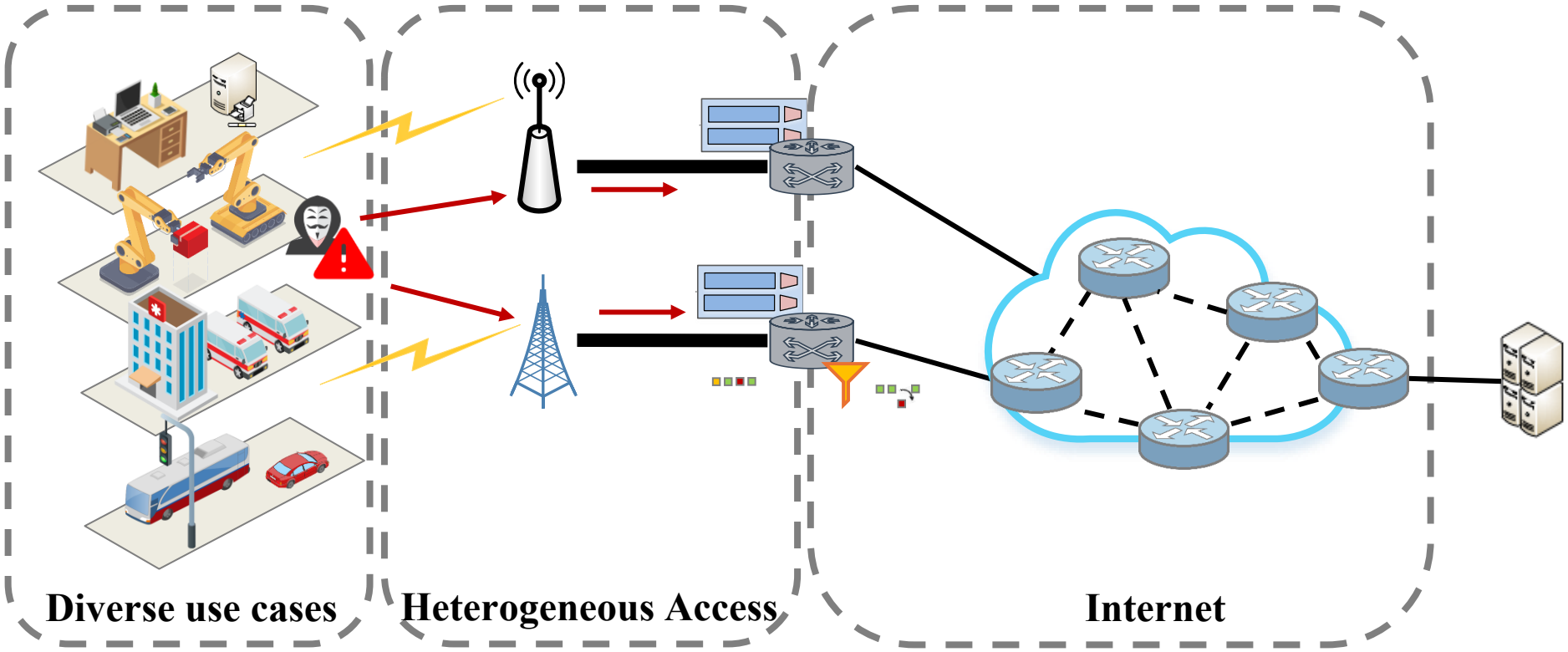
Internet of Things (IoT) Network



5G/6G's extremely low latency requirements + emerging attack variants in IoT

→ Fast spreading threats with changing patterns

Internet of Things (IoT) Network



Programmable data planes enable flexible in-network traffic processing

Dynamic Traffic Pattern

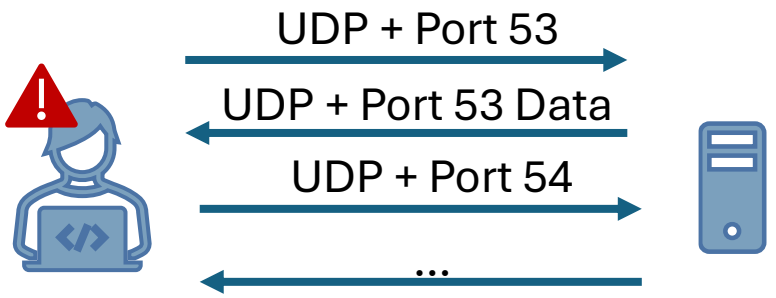
In-Band Feature Collection

Port Scan attack as an example:

- A UDP packet sent to all ports and response from the target port
- reflected by port, length distribution in packets

Collected Features:

- Number of UDP packets
- Number of unique port pairs
- Max/min packet length
- …

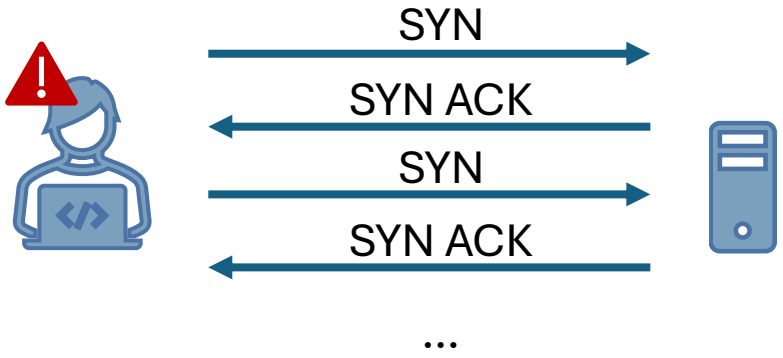


Dynamic Traffic Pattern

In-Band Feature Collection

DDoS LOIT as an example:

- Halfway handshakes triggered by numerous SYN
- Reflected by the number, rate and TCP flags of packets within a time window



Collected Features:

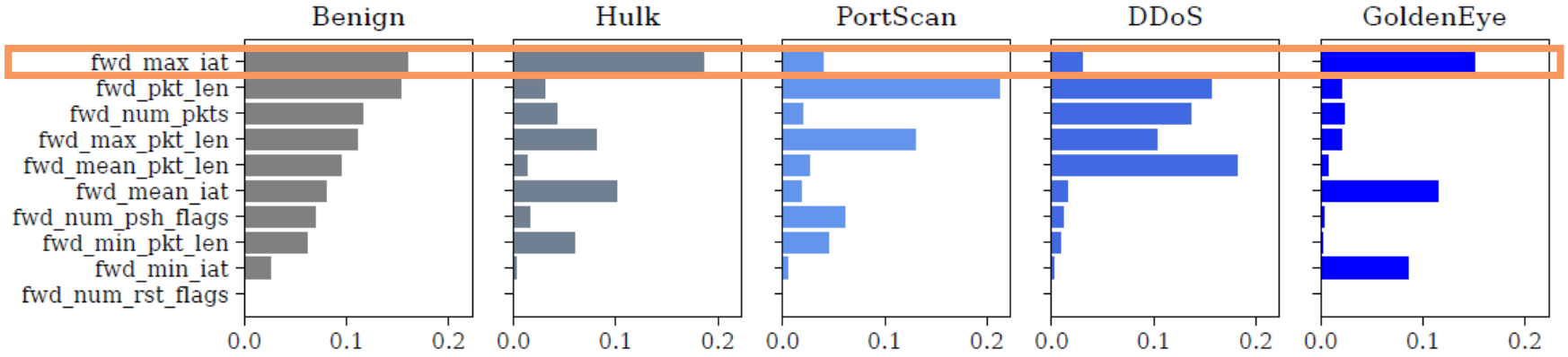
- Number of TCP packets
- Number of TCP SYN flags
- Inter-Arrival Time (IAT)
- …

Dynamic Feature Importance

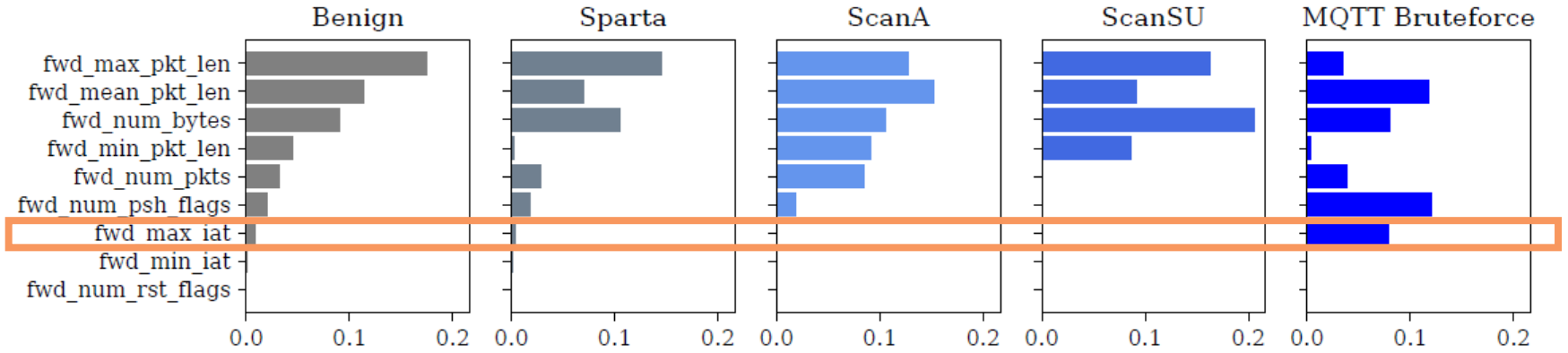
Distribution of Feature Importance

- Same group of features shows different levels of importance in attack identification
- e.g. Inter-arrival time

Dataset: CICIDS2017



Dataset: MQTT2020

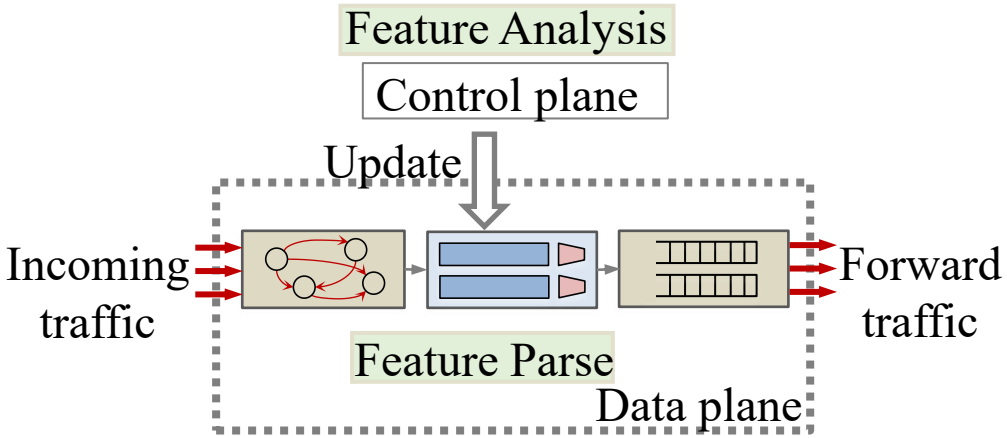


How to adaptively collect the new feature sets?

Programmable Data Plane

Programmable data plane with P4 language

- Protocol-Independent architecture
- Custom packet header & processing
- More flexibility on the data plane



Design Target

- Apply P4 for in-band feature collection
- ML algorithm for accurate & generalizable detection

How to adaptively learn from new traffic patterns?

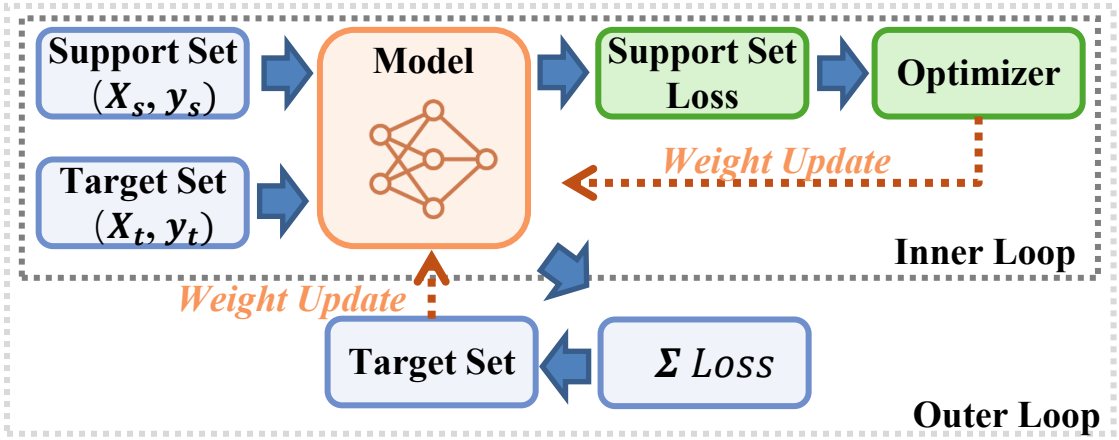
Proposed Design – Meta Learning

"Learning to learn"

Rapid adaptation to new scenarios with minimal data learned

Our work: Model-Agnostic Meta-Learning (MAML)

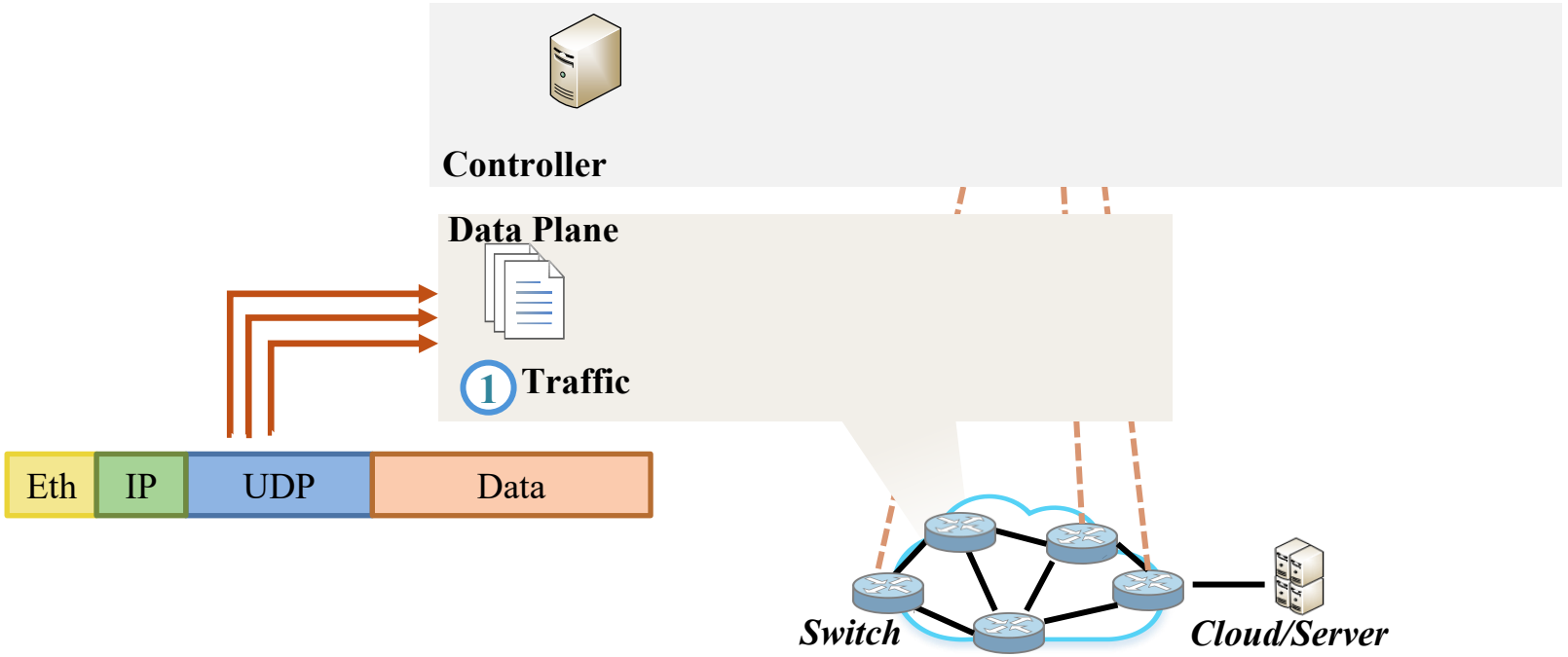
- Optimization-based meta-learning
- Inner loop (quick adaptation) + outer loop (stable convergence)
- Stronger generalization



Proposed Design – Framework

Step 1 Packet Parsing

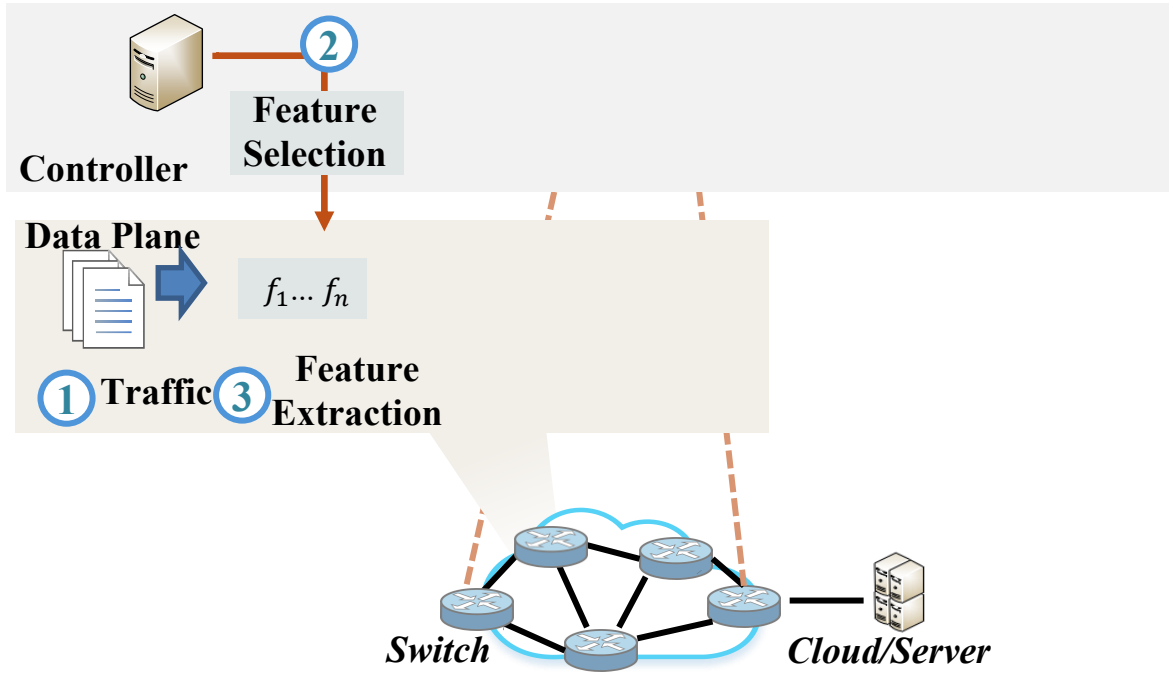
- Header fields are parsed and protocol information is extracted



Proposed Design – Framework

Step 2 & 3 Statistic Extraction

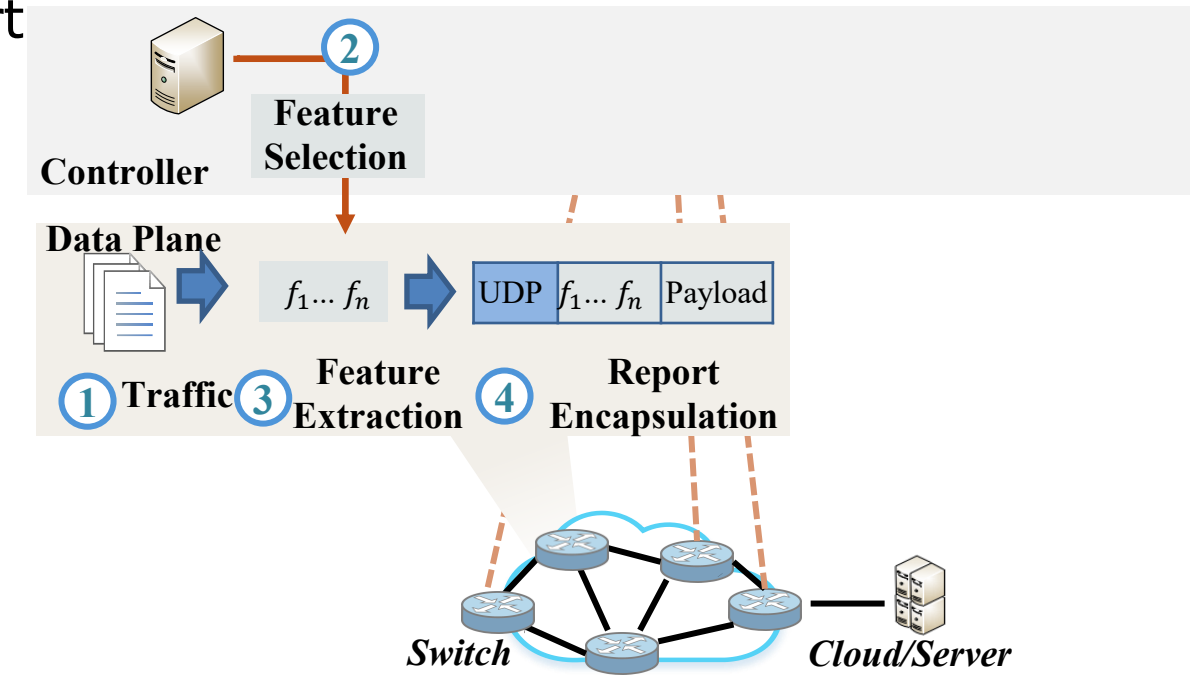
- Stateful features are computed by hashing and updating counter registers



Proposed Design – Framework

Step 4 Report Encapsulation

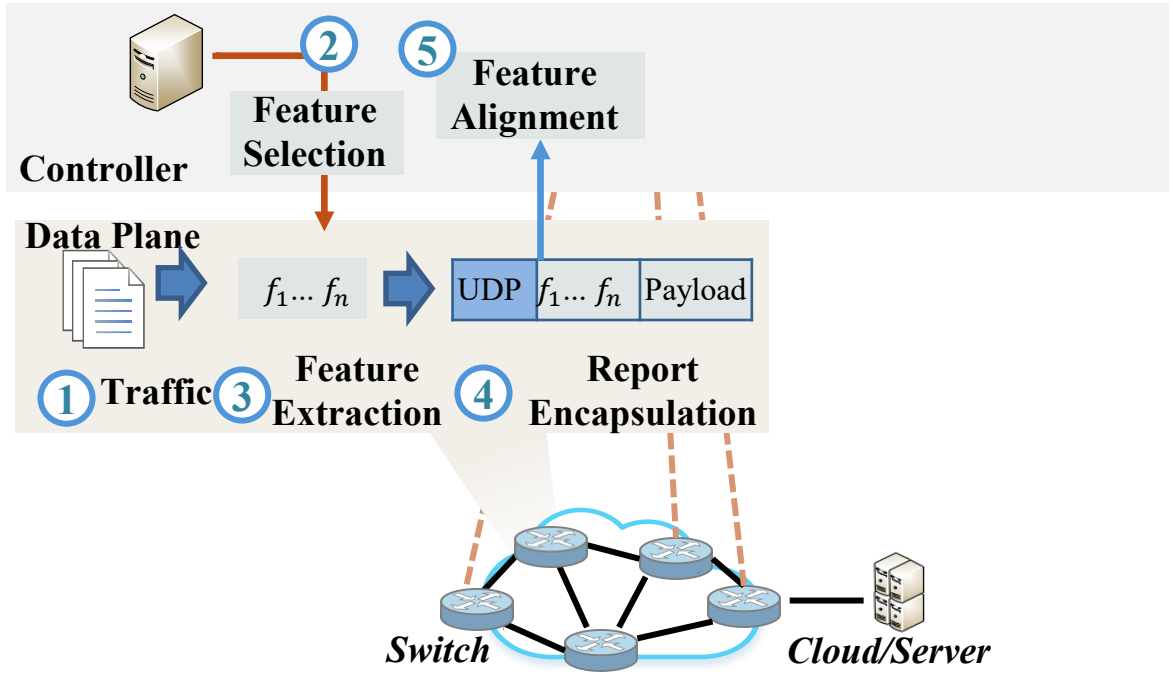
- When a packet arrives at the ingress port, a timestamp is recorded
- Once the timestamp exceeds time window T, collected features {f1,..., fn} are read
- Features are encapsulated in UDP report



Proposed Design – Framework

Step 5 Feature Alignment

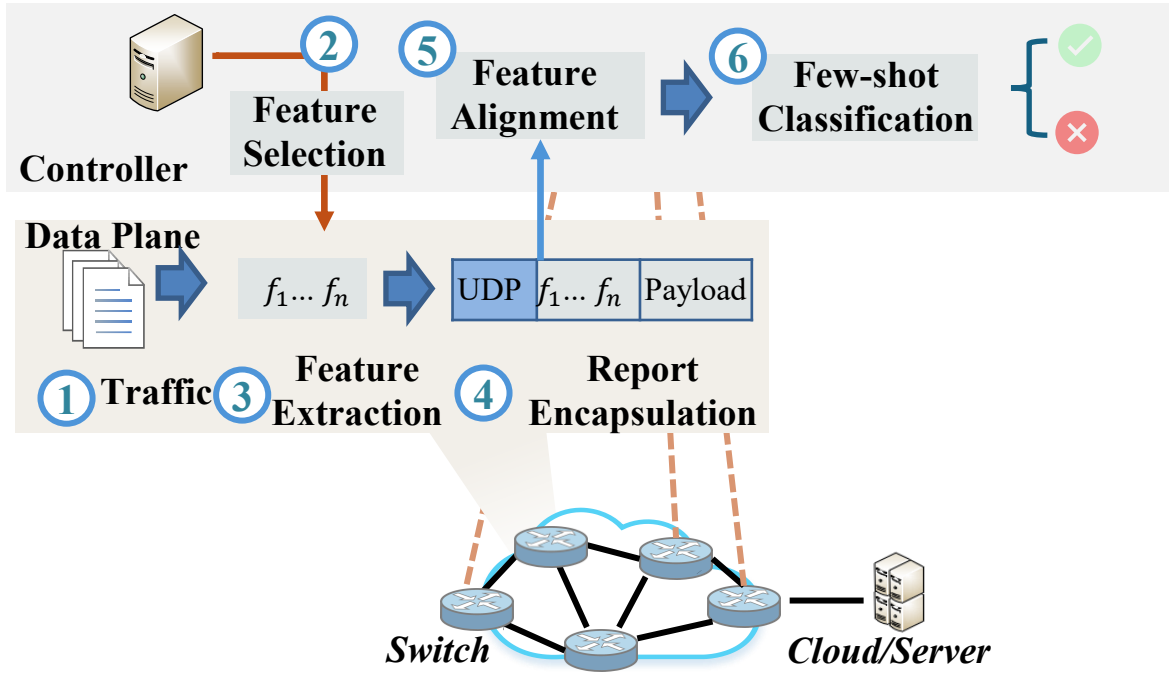
- Feature alignment is performed to adapt to new feature sets



Proposed Design – Framework

Step 6 Few-Shot Classification

- Features are parsed and labeled by a pre-trained meta-learning model
- The model is trained offline and then loaded for real-time inference on a small number of examples



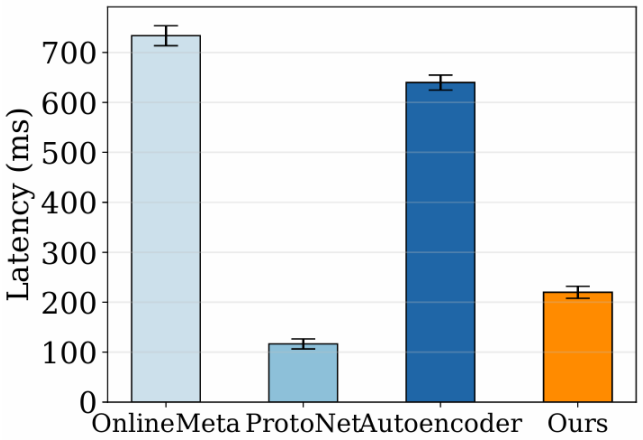
Evaluation Results

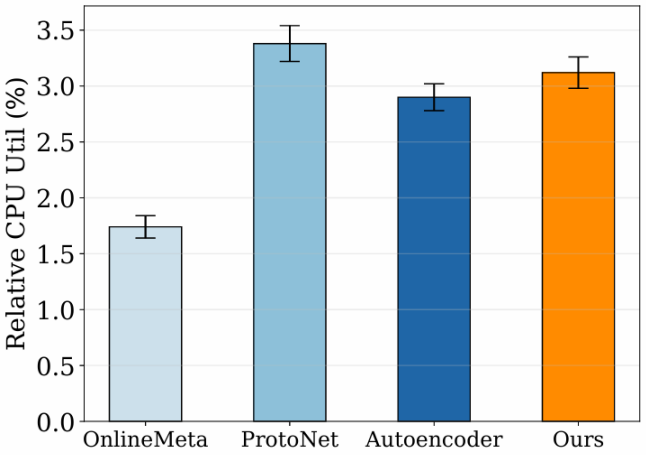
- **Public Dataset:** public dataset CICIDS 2017, IoT Sentinel, MQTT2020
- **ML Baseline:** Autoencoder, PotoNet, OnlineML
- **Prototype:** BMv2 programmable switch + topology emulated in Mininet

- **Performance**
  >70% accuracy, better trade-off between accuracy/inference time, 1.5× faster latency

TABLE III: Model Performance

	Accuracy	Precision	Recall	F1	Train Time (ms)	Infer. Time (ms)
AE	0.71	0.72	0.72	0.71	45	612
ProtoNet	0.43	0.33	0.50	0.30	25	189
OnlineML	0.43	0.45	0.49	0.34	47	1068
Ours	0.75	0.75	0.76	0.75	2357	792

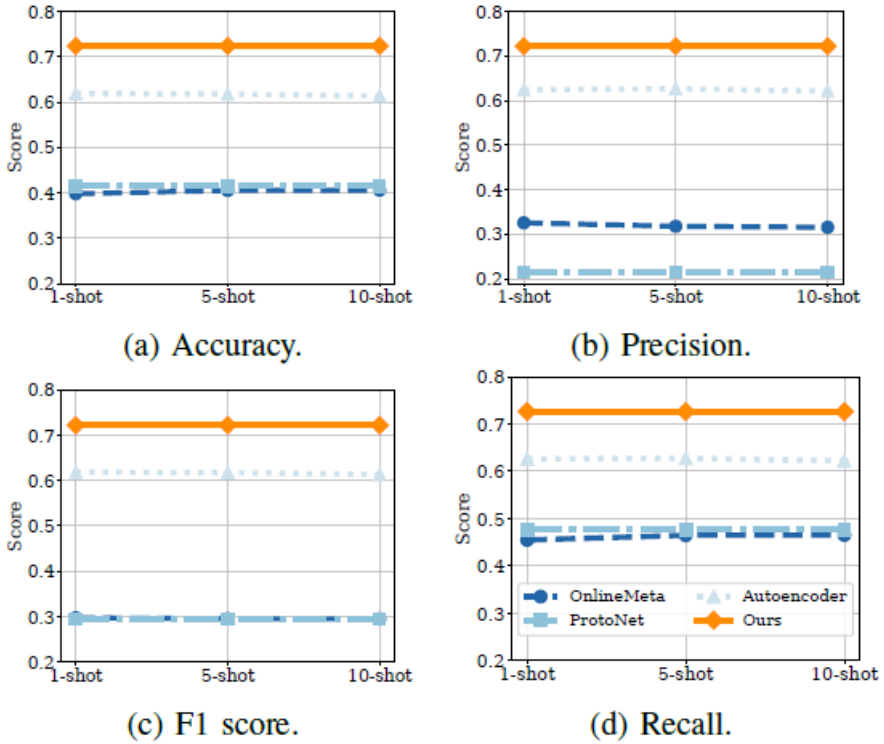




Evaluation Results

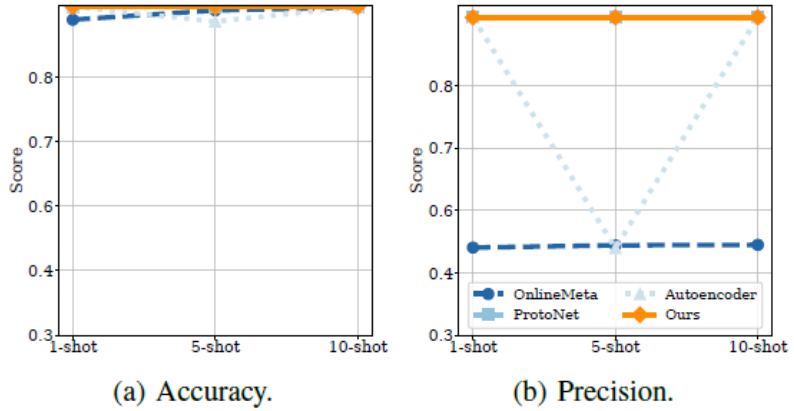
- **Performance**

  outperform baseline algorithms by 8%-17% higher accuracy across few-shot samples



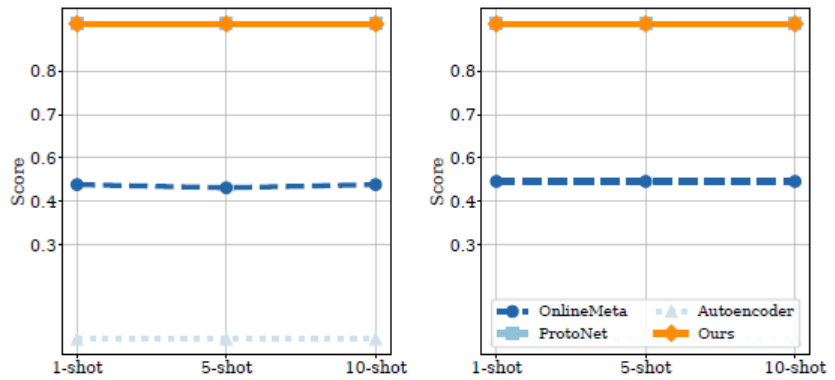
(a) Accuracy. (b) Precision. (c) F1 score. (d) Recall.

Dataset: CICIDS2017



(a) Accuracy. (b) Precision.

Dataset: IoTSentinel



(a) Accuracy. (b) Precision.

Dataset: MQTT2020

Conclusion

We present a framework to adaptively collect the new feature sets and promptly learn from them:

- Accurate ML-based traffic analysis with reconfigurable feature parsing
- Generalizable performance with reasonable learning time
- Lower manual recompilation overhead

Further work:

- Scalable deployment scheme on hardware

Thanks!
Questions?