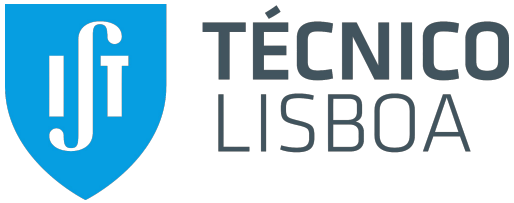
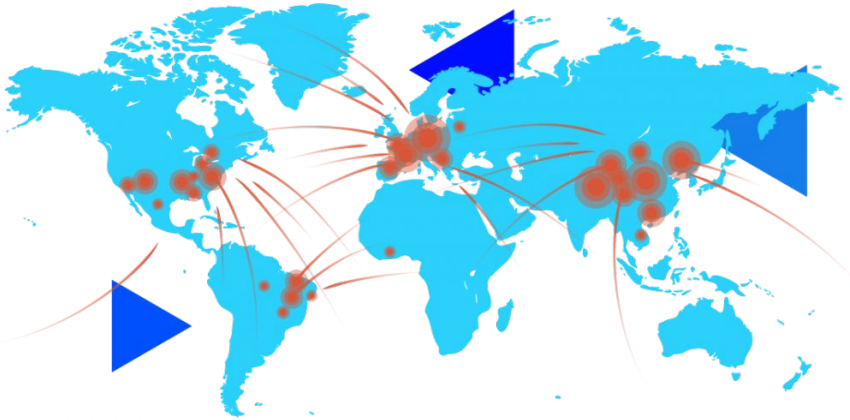
Towards in-network anomaly detection

João Romeiras Amado, Salvatore Signorello,
Miguel Correia, Fernando Ramos

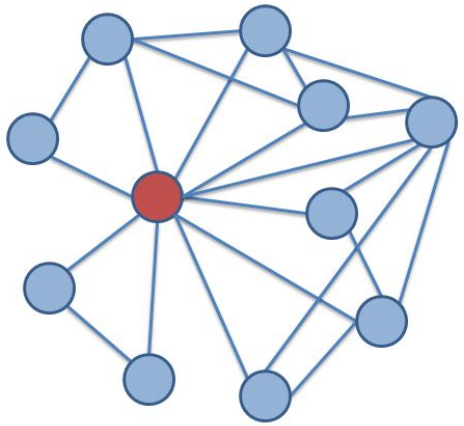


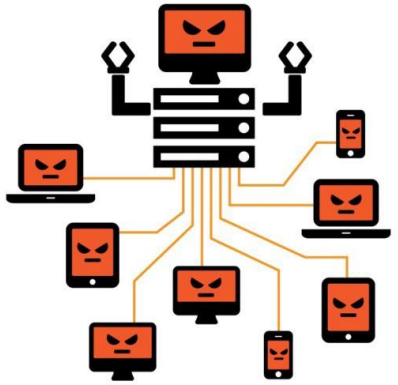
Problem



New network attacks keep evolving in scale and complexity

Leveraging **in-network detection** is key for achieving both higher **detection speed** and **packet processing rates**

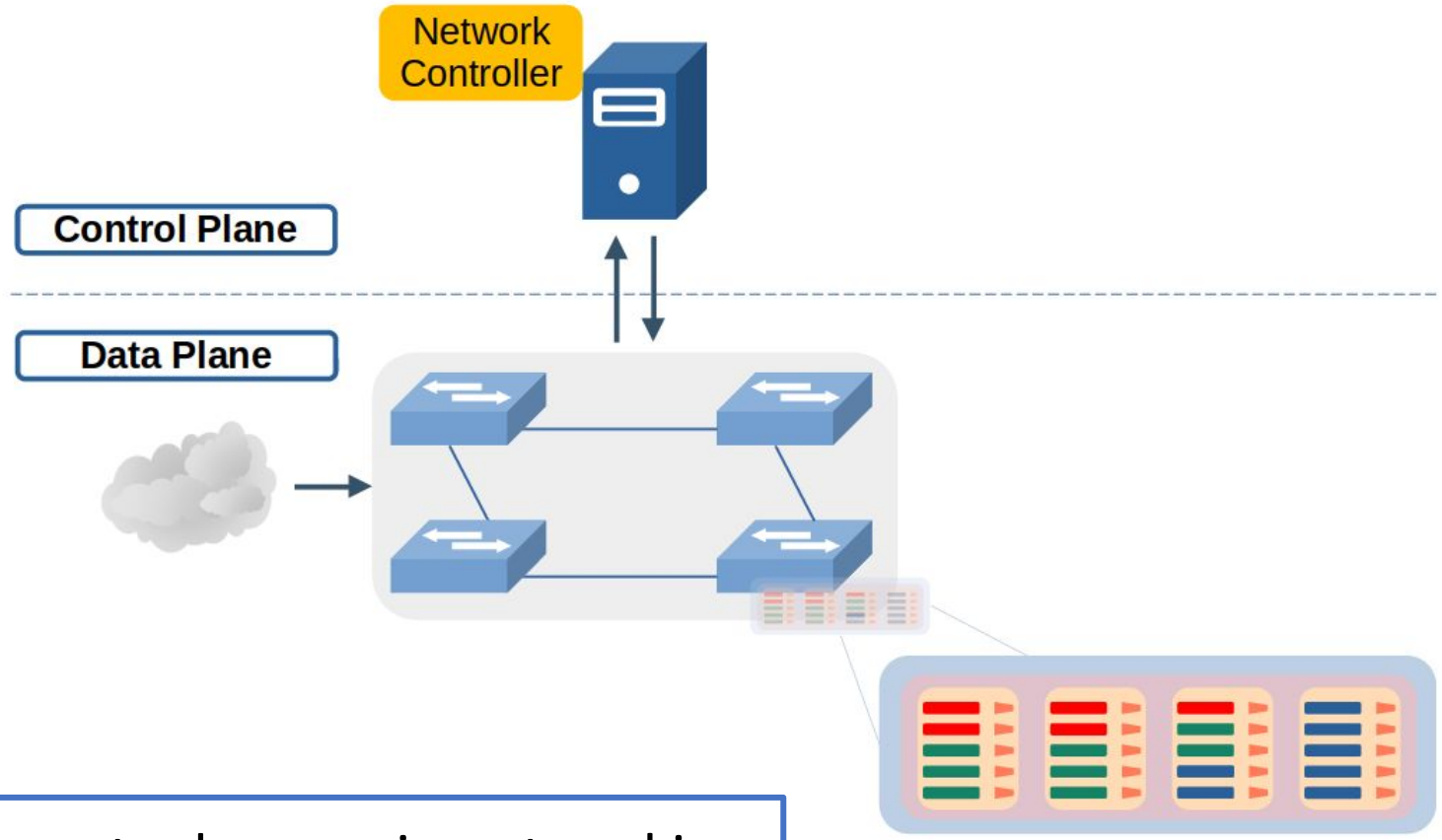




Anomaly-based Intrusion Detection

- Detect deviations from regular network traffic profiles
- Machine learning-based classification
- Higher performance tax
- Dependent on measurement quality

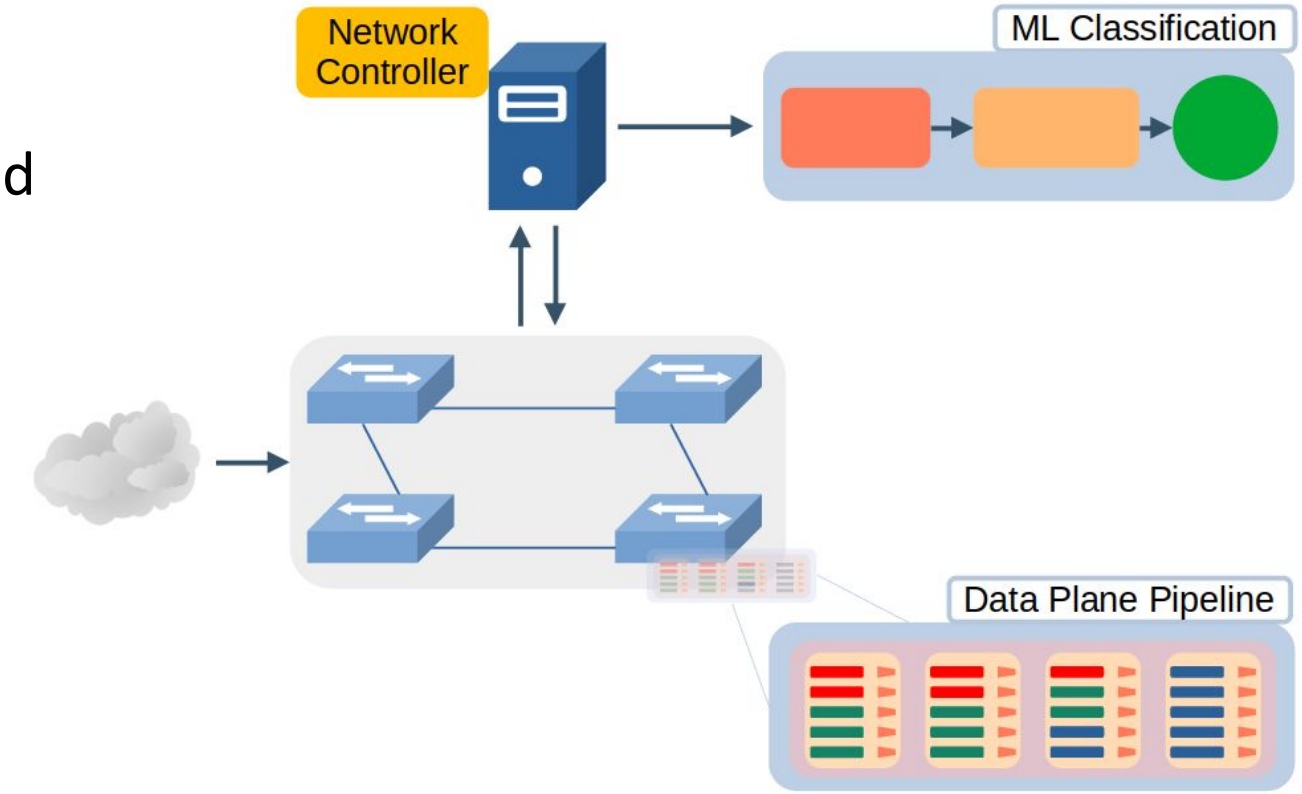
Enter the Programmable Data Plane



How to leverage recent advances in networking infrastructure to improve Intrusion Detection?

Contributions

- **Peregrine**, an in-network, ML-based anomaly detection framework
- Implementation targeting a programmable hardware switch
- Preliminary evaluation showcasing anomaly detection results



Design

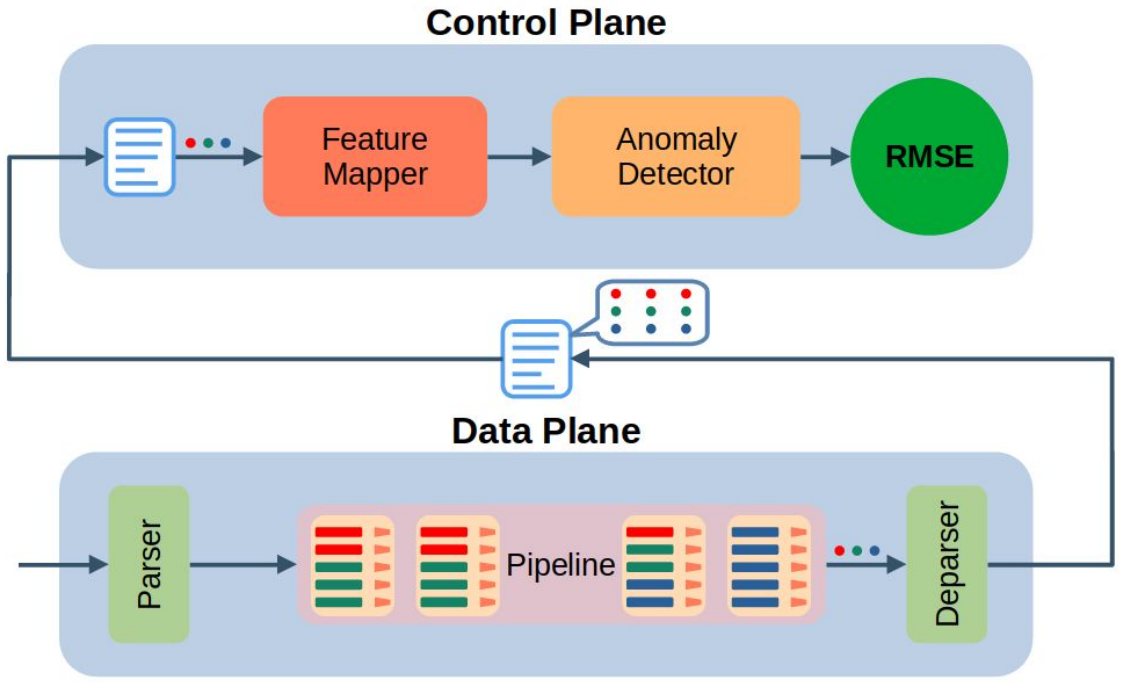
Peregrine

- Network Controller

- Feature Mapper
- Anomaly Detector

- Feature Extraction

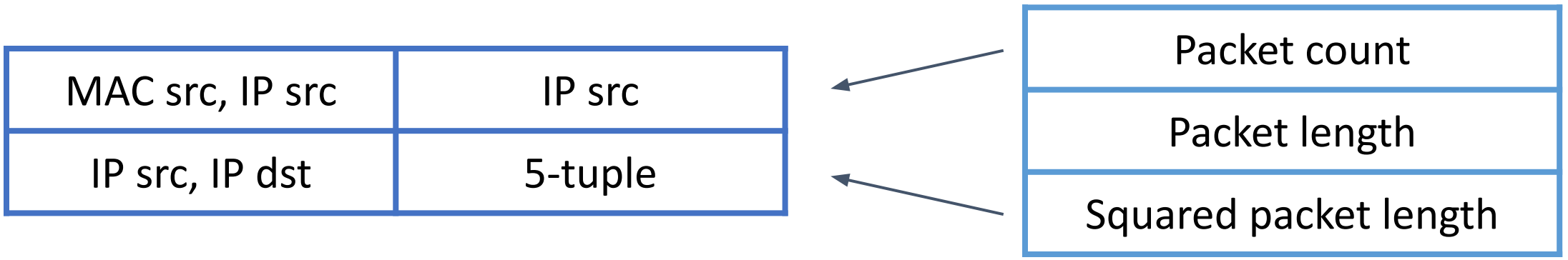
- Traffic Statistics Calculation



Peregrine - Data plane challenges

- Some **mathematical operations** are not directly supported by the target architecture
- The calculation process must be split across several **pipeline stages**
- Strict limitations regarding **per-packet memory access**

Feature Extraction



- Peregrine manages **four flow keys**
- On every arriving packet, **three basic counters** are updated for each key and stored in P4 registers

Traffic Statistics Calculation

1D statistics

- Calculated for all flow keys
- Depend on a single flow direction

2D statistics

- *[IP src, IP dst]* and *[5-tuple]* keys
- Depend on both flow directions

Type	Statistics	Notation	Calculation
1D	Weight	w	w
	Mean	μ	LS/w
	Std. Deviation	σ_{S_i}	$\sqrt{\lvert SS/w - (LS/w)^2 \rvert}$
2D	Magnitude	$\lVert S_i, S_j \rVert$	$\sqrt{\mu^2_{S_i} + \mu^2_{S_j}}$
	Radius	$R_{S_iS_j}$	$\sqrt{(\sigma^2_{S_i})^2 + (\sigma^2_{S_j})^2}$
	Approx. Covariance	$Cov_{S_iS_j}$	$SR_{ij}/(w_i + w_j)$
	Corr. Coefficient	$P_{S_iS_j}$	$Cov_{S_iS_j}/\sigma_{S_i}\sigma_{S_j}$

Decay Factor

Recent statistics should be given more weight than older values

- **Metric:** difference between the arrival times of packets with matching flow keys, over four time intervals

Time Intervals
100ms
1s
10s
60s

$$d_{\lambda}(t) = 2^{-\lambda t}$$

λ → Decay factor

t → Time elapsed since the last matching packet

- Only one time interval is checked per packet, with the selected value alternating for each pipeline execution

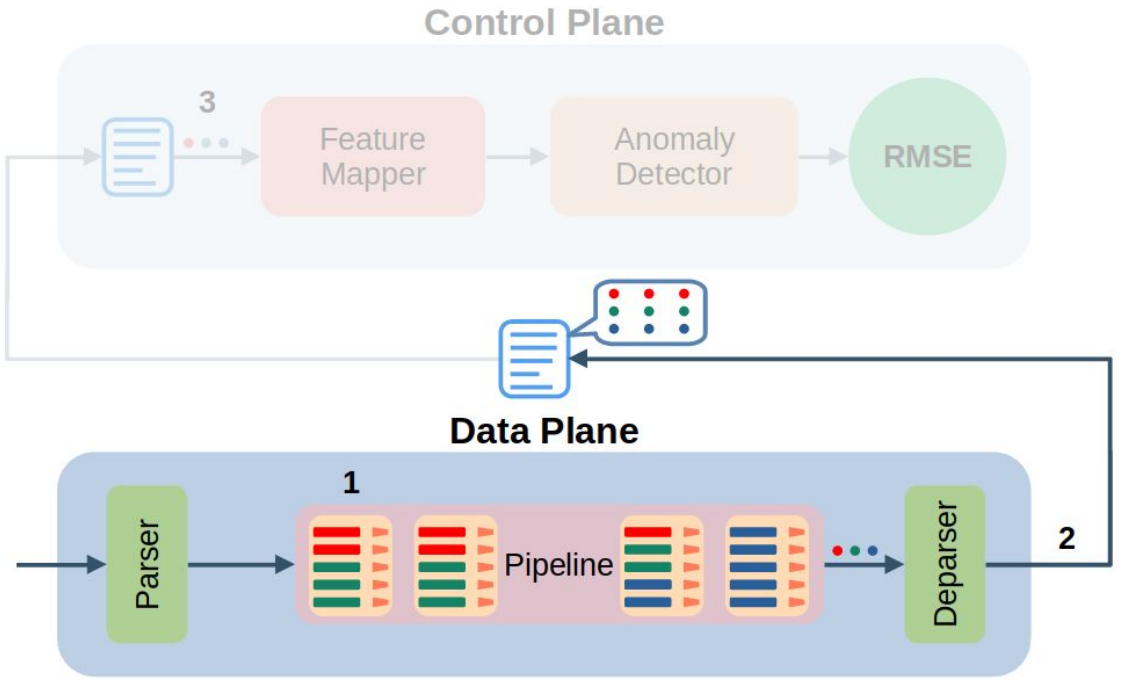
Target Architecture Constraints

For each register, only a single position may be accessed per packet

- 2D statistics require counters for both flow directions

Calculate 2D statistics every *x* packets

- Once per epoch
- Whenever the calculations occur, each register access can be changed between read and write as required



Target Architecture Constraints

Some mathematical operations are not feasible in the data plane

- Division
- Floating point operations

$$\sqrt{\left|SS/_w - \left(LS/_w\right)^2\right|}$$

Operations performed using approximations

- **Tofino math externs:** Square, Square root
- **Bit-shifting:** Multiplication, division

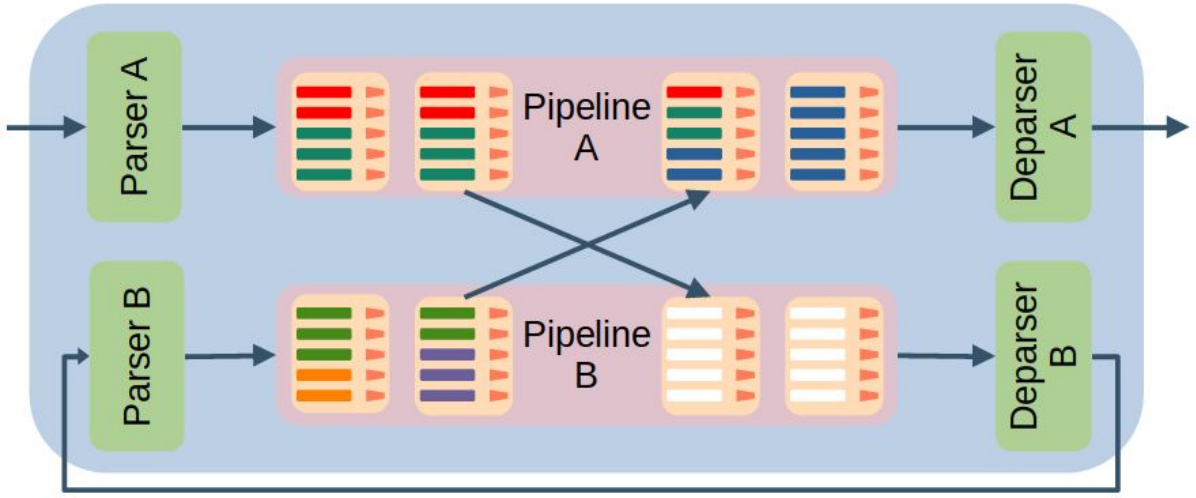
  (Operators rounded to the nearest lower power of two)

$$\sqrt{\left|SS/_w - \left(LS/_w\right)^2\right|}$$

Target Architecture Constraints

Fixed number of processing stages per pipeline

- A single pipeline is insufficient due to the complexity and number of operations performed

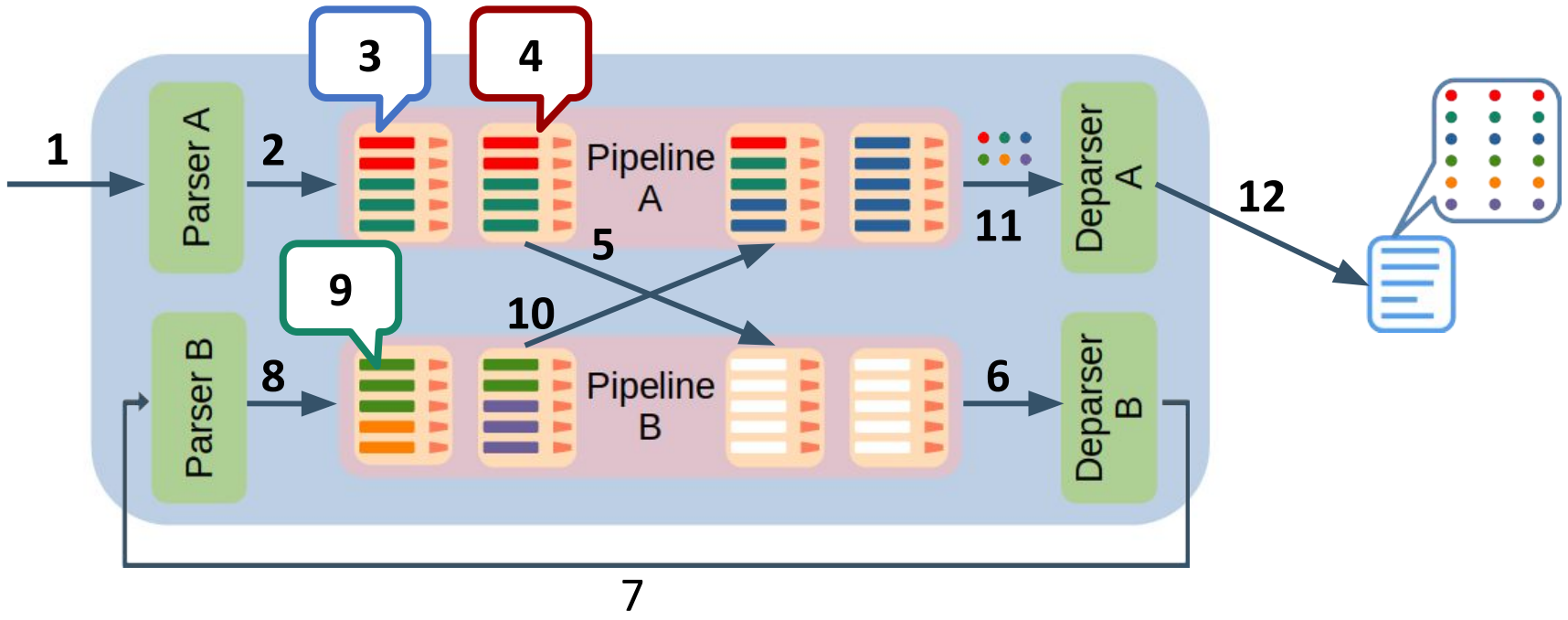


Packet Recirculation

- Packet sent to another pipeline
- Gain access to additional stages for further processing
- Once per epoch

Data Plane Workflow

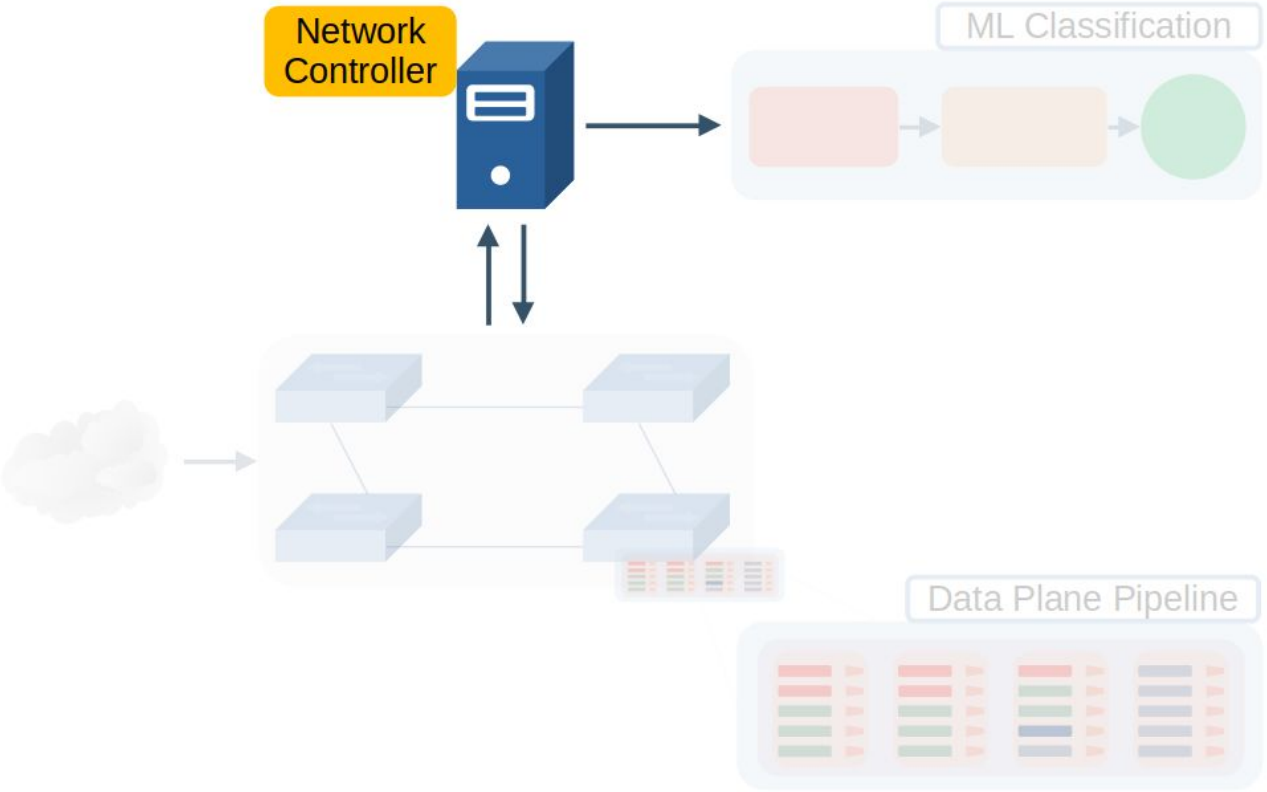
Magnitude	$\|S_i, S_j\|$	$\sqrt{\mu^2_{S_i} + \mu^2_{S_j}}$



1. Packet arrival
2. Parsing
3. S_i: mean calculation
4. S_j: Read previous mean
5. Send packet to pipeline B egress
6. Deparsing
7. Recirculation
8. Parsing
9. Magnitude calculation
10. Send packet to pipeline A egress
11. Deparsing
12. Send packet to control plane

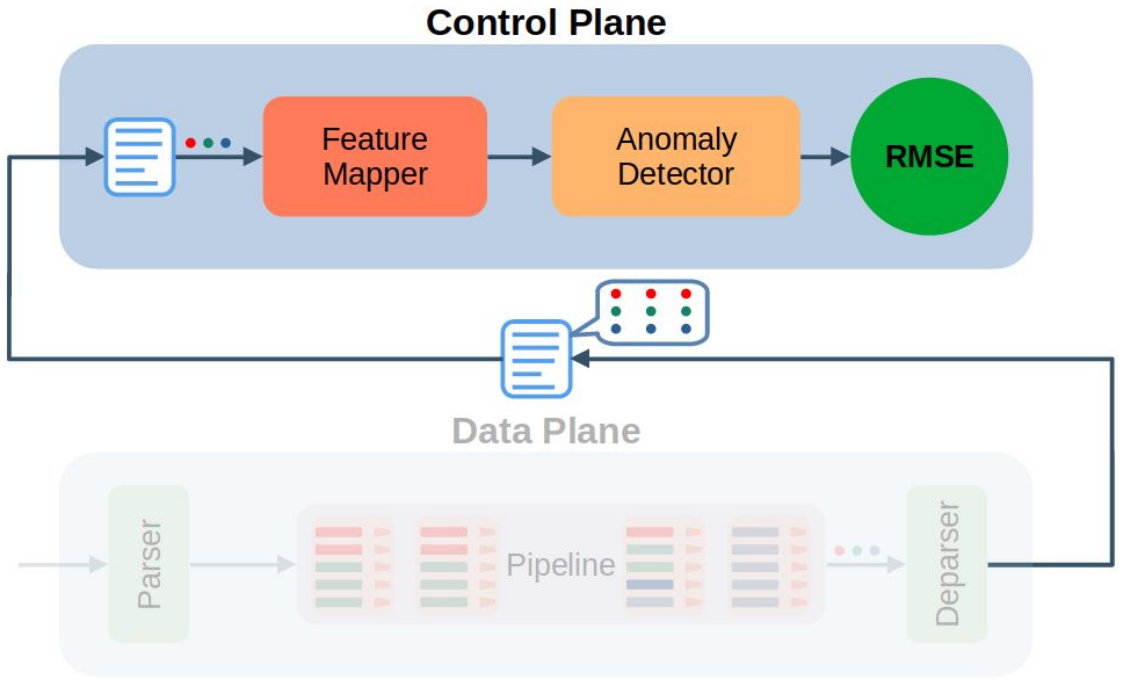
Control Plane - Controller

- Manages the configuration of the switch's data plane
- Receives packets containing the calculated statistics and feeds them to the ML pipeline



Control Plane - ML pipeline

- ML pipeline leveraged from the Kitsune IDS[1]
- Neural network of **autoencoders** trained with benign traffic
- Outputs an **RMSE score** for each processed flow



[1] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. NDSS'18

Evaluation

Questions

- **Q1.** How do Peregrine's measurements perform in terms of anomaly detection when compared with Kitsune?
- **Q2.** What is the runtime performance achieved by Peregrine on a programmable switch?
- **Q3.** What is Peregrine's resource usage?

Experimental Setup

- Network traces containing labeled attacks from two datasets:
  - Kitsune's evaluation dataset
  - CIC-IDS-2017

- The preliminary evaluation was performed entirely on the control plane, simulating the statistics' calculation process of the Tofino

- For each trace, the first 1M packets (benign) are used to train the neural network, with the remainder used to evaluate the model

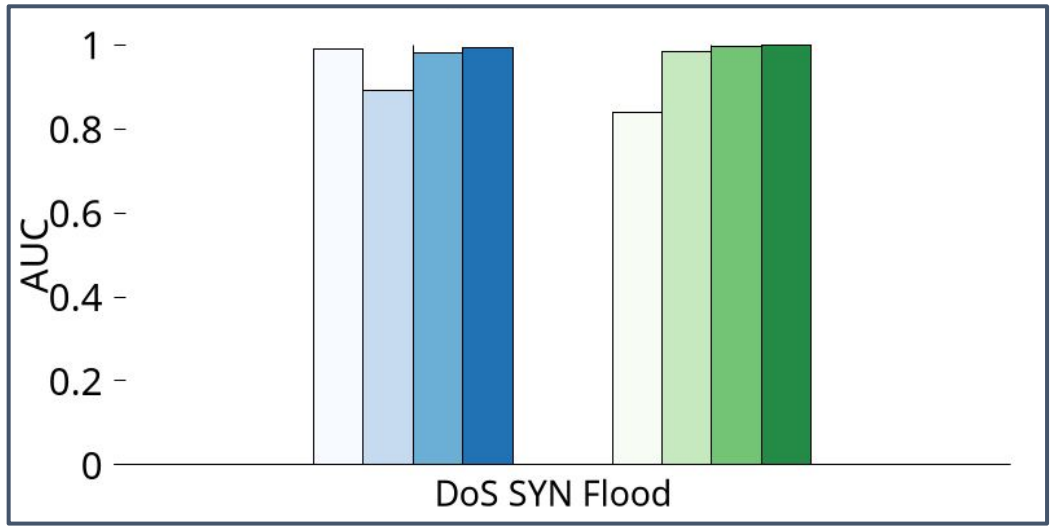
Q1. Anomaly Detection vs. Kitsune

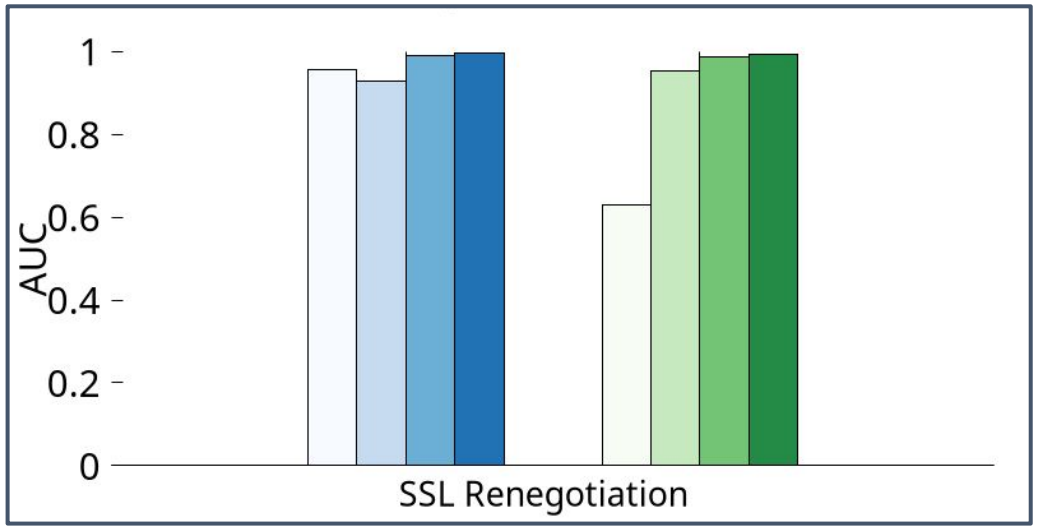
Kitsune 1/1
Kitsune 1/8
Kitsune 1/64
Kitsune 1/1024

Peregrine 1/1
Peregrine 1/8
Peregrine 1/64
Peregrine 1/1024

Area under the Curve (AUC)

Higher results are better





Peregrine's results with 1/1024 sampling closely match Kitsune's original results.

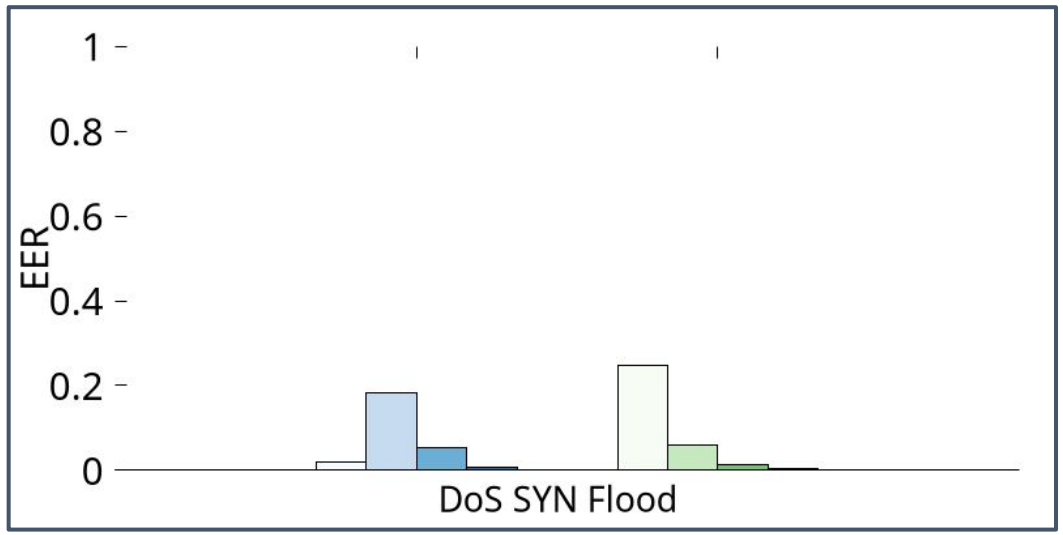
Q1. Anomaly Detection vs. Kitsune

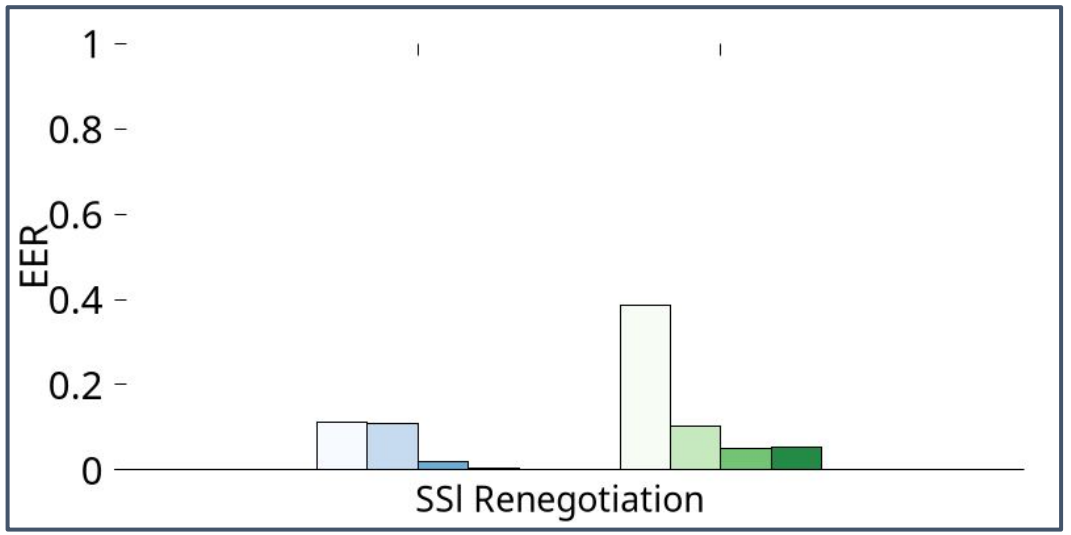
Kitsune 1/1
Kitsune 1/8
Kitsune 1/64
Kitsune 1/1024

Peregrine 1/1
Peregrine 1/8
Peregrine 1/64
Peregrine 1/1024

Equal Error Rate (EER)

Lower results are better





Peregrine's results with 1/1024 sampling closely match Kitsune's original results.

Q2. Runtime performance

- Kitsune’s experimental results achieved a maximum processing rate of 35k packets-per-second

- Peregrine’s data plane implementation has been successfully compiled on the Tofino switch architecture, **ensuring line-rate performance** at Tbps speeds

Q3. Resource usage

Peregrine prototype for the **Tofino Native Architecture**

	Pipeline 0	Pipeline 1
Stages	100%	91.7%
Meter ALU	72.9%	6.9%
Hash Dist Units	43.1%	0%
VLIWs	26.6%	56.0%
SRAM	37.2%	6.4%
TCAM	6.9%	9.7%

Current Status

- **Peregrine** leverages programmable networking hardware to move part of the intrusion detection to the data plane
- Feature extraction and calculation of measurements entirely on the data plane
- Successful compilation on the Tofino switch, ensuring **line-rate performance**
- Preliminary evaluation shows a **detection performance comparable to Kitsune**, while scaling to around 5 orders of magnitude higher in packet processing speed

Next Steps & Future Work

- Performance evaluation on the Tofino switch
- Integration of machine learning components in the data plane

Thank You