New Distribution Scheme

- ***Raw Sockets -> Framework***
  - Much more condensed and managed
  - Transparent
  - Scalable
- ***Flow Generation:***
  - NFStream - automatically parallel
- DATA: 2,830,743 samples (total amongst all CSVs)

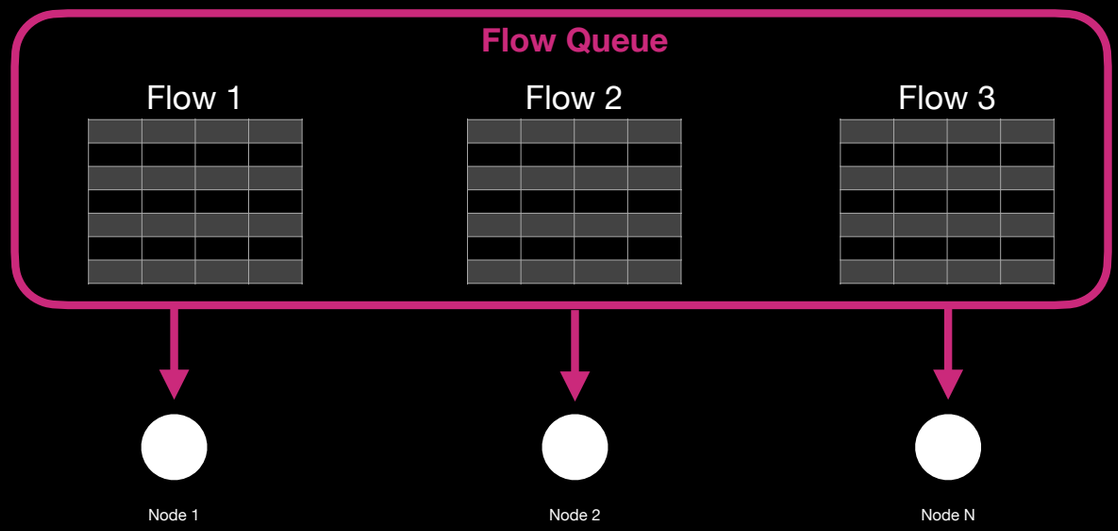
Flows are resource intensive - take time to generate. I got around this by using NFStream only to parse data from a temporary file written to by Scapy. All data written by Scapy is a very small amount of packets that NFStream can easily handle in parallel.

Data Delegation - Options

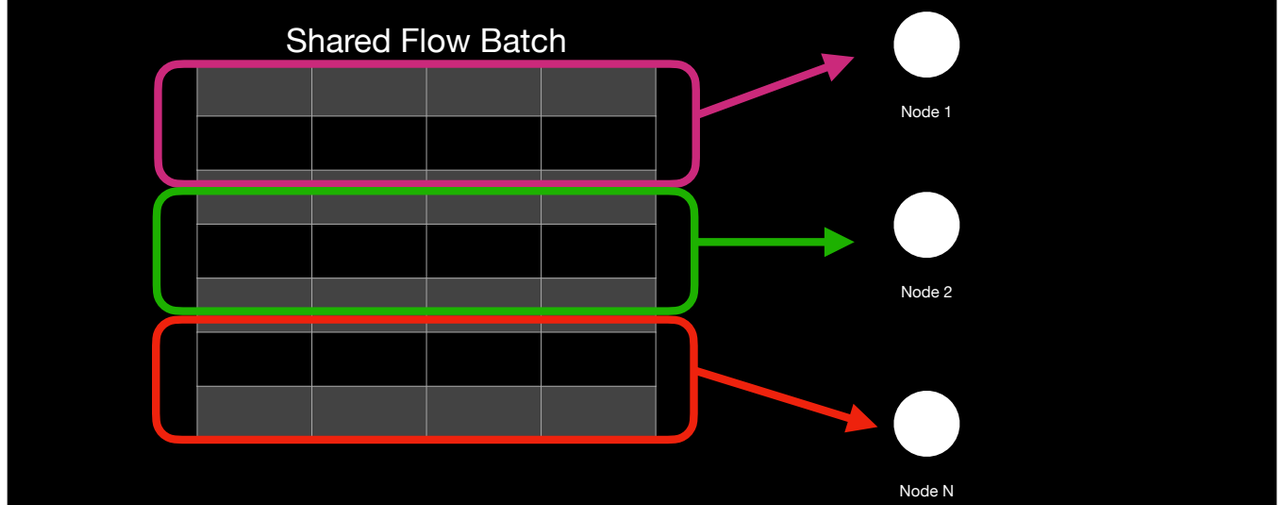
- **Option 1:**
  - Collect flows in increments and each node in the cluster gets handed a batch - our current approach.
- **Option 2:**
  - Collect flows into a larger batch, and have it distributed across the cluster with batch learning (available as an implicitly parallel operation in Ray).

For the head node on a cluster: > ray start --head —port=6379
For any nodes manually managed: > ray start —address='127.0.0.1:6379'
To run a script on the cluster (must be done on the cluster itself) either use:
ray.init(address="auto") or ray.init(address=…)
Then use the submission on the cluster as such:
ray submit <script work>
To resolve any dependencies on the cluster use:
ray rsync-up

Data Delegation - Option 1



Data Delegation - Option 2



From Zero (offline) Deployment Pipeline

- Step 1: Train all known nodes in a **federated** manner.
  - 1.5: Consider “best” model; e.g., accuracy, performance.
- Step 2: Begin IDS System
  - Consider data delegation strategy (collaboration).
- Step 3: Scaling
  - New nodes can join asynchronously and be served the model - already trained in step 1.
  - Now these nodes obtain their own flow batches and can predict using the model.
  - Many nodes can cause many aggregation requests (master node needs to be reasonably resource-capable).

Federated learning: Each node trains locally its own dataset, and contributes to global accuracy.
Dual data delegation strategy - either way each node predicts on it’s own “batch” of flows
- Question may not necessarily be which is better, or easiest, but rather most efficient.

Master node - responsible for a lot; training, and IDS management.
We can build our own router linux.

Ray - it is able to make the models served into a web service and also automatically select the resources to be used on each node. Need to explore its applicability on lower-end devices.

Additional thought - Distributed **model,** but how can it be made parallel?
Profile different models: Inference times, [CPU/Memory/Disk] footprints, training times (federated). ** IMPACT on real time performance.

Online machine learning (is there a possibility of concept drift - maybe with new attacks, but they are all generally the same pattern)? We can use evidence building to correlate with inference strength - confidences can be taken into account from the individual nodes **as the inferences are made and then push the result to the master when ready.**

Why is federated training a benefit? Consider full training on the complete dataset on a single node versus training all of them separately in federated mode - perhaps we can integrate it as a framework.