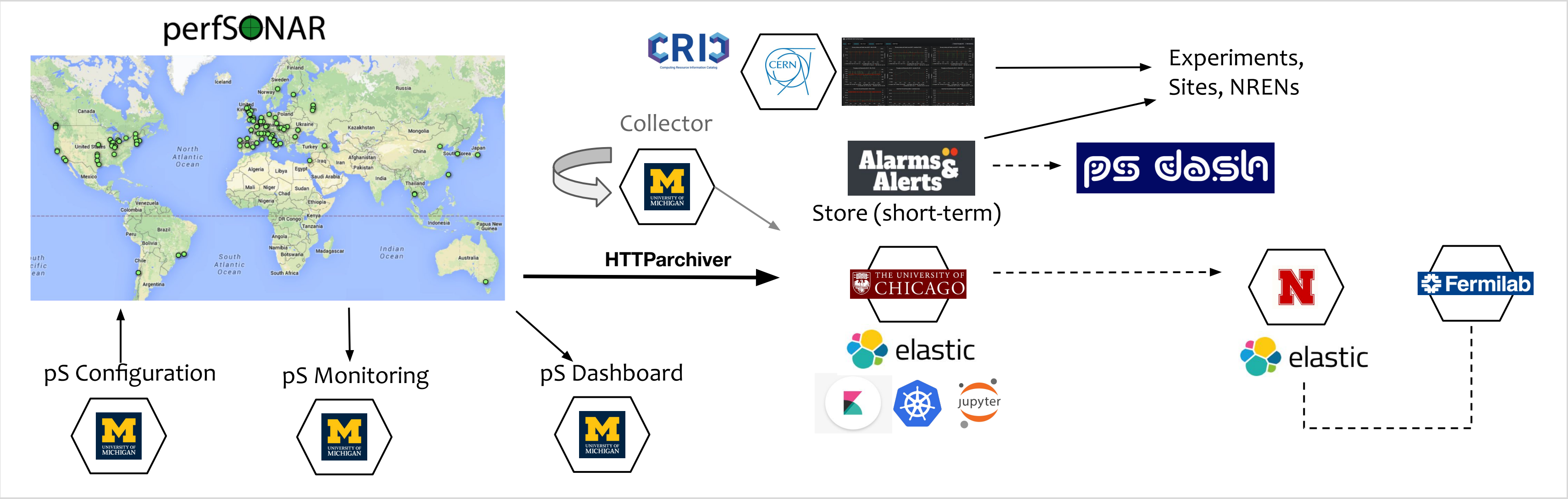
# Enhancing Network Analytics through Machine Learning

**CHEP’24**, 19–25 Oct 2024
Krakow, Poland

**Petya Vasileva** / U Michigan, **Marian Babik** / CERN
**Shawn McKee** / U Michigan, **Ilija Vukotic** / U Chicago

# The perfSonar platform

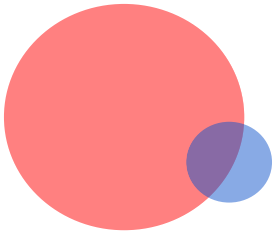
perfSONAR continuously **measures** network performance metrics like **bandwidth, latency, packet loss, across various network paths** that are crucial for **OSG** and **WLCG** operations



The goal is to

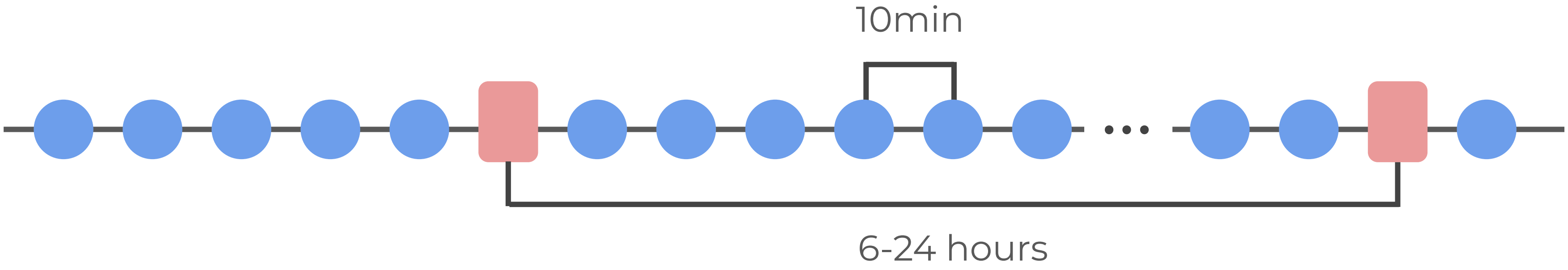
**proactively discover network issues**

# How to correlate network metrics?



many paths, few bandwidth tests

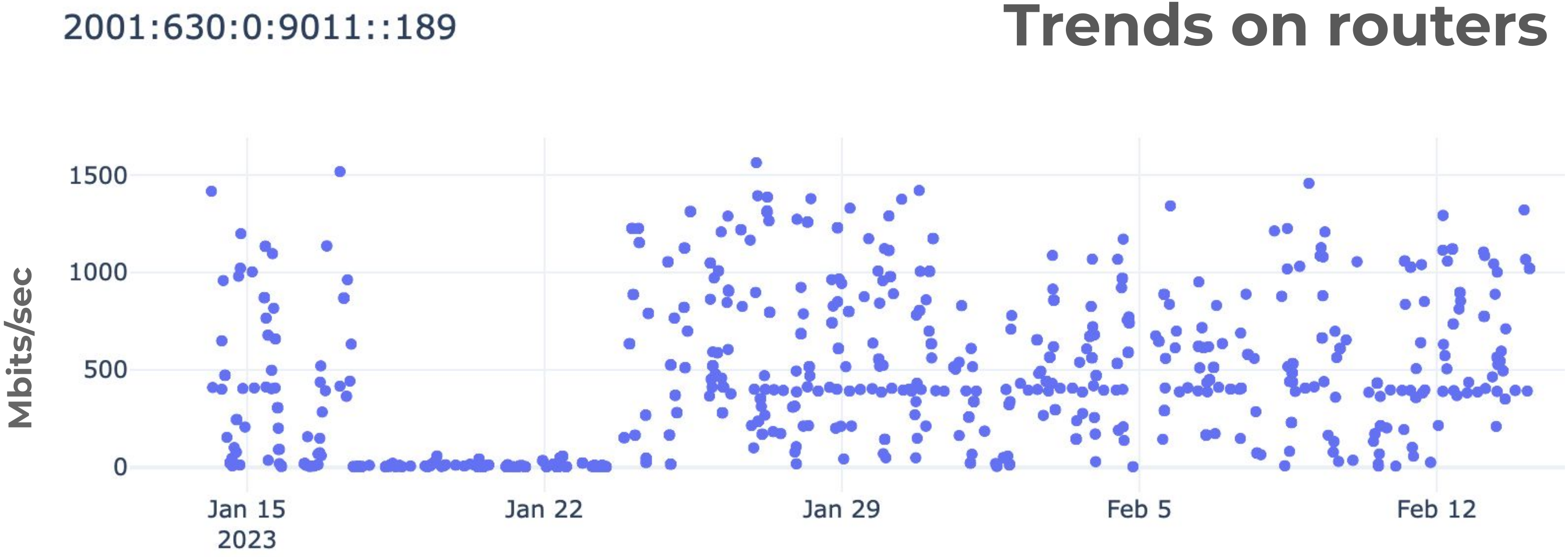
Tests' rate of execution varies by type




traceroute test every 10 min

throughput test every 6-24 hours

# Trends on routers



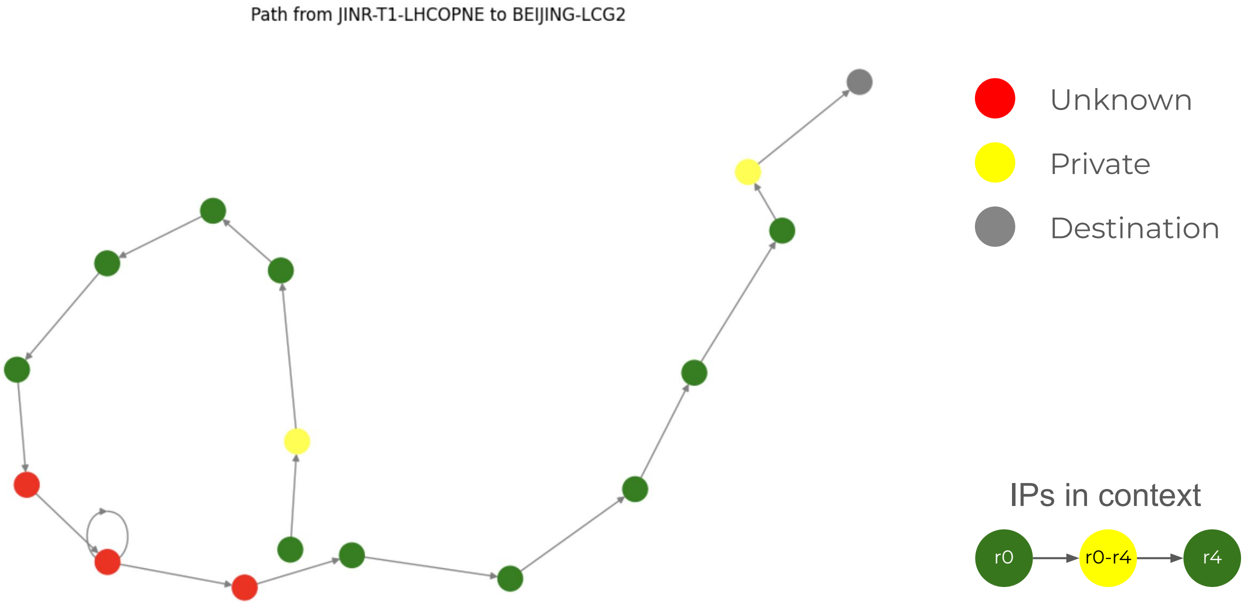

Each **point** represents the throughput values collected when the node was on the path

# Simplified example of traceroute data

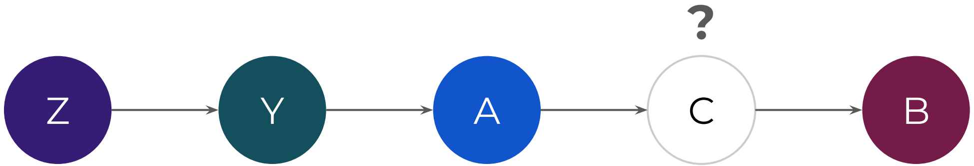
| timestamp | src_host | dest_host | hops | asns |
|---|---|---|---|---|
| Oct 22, 2024 @ 17:54:47.000 | perfsonar-bandwidth.esc.qmul.ac.uk | btw-lat.t1.grid.kiae.ru | [2a01:56c1:10:1000::1, 2a01:56c0:a020:1::1, 2001:630:0:9001:10::29, 2001:630:0:10::251, 2001:630:0:10::1cd, 2001:630:0:10::1c9, … | [198864, 198864, 786, 786, 786, 786, 20965, 20965, 20965, 20965, 20965, 59624, 59624] |
| Oct 22, 2024 @ 17:54:47.000 | ccperfsonar1.in2p3.fr | perfsonar-grid.uaic.ro | [193.48.99.100, 192.70.69.153, 193.51.187.137, 193.55.204.196, 62.40.124.61, 62.40.98.77, 62.40.98.182, 62.40.98.159, 62.40.98.53, … | [2200, 2200, 2200, 2200, 20965, 20965, 20965, 20965, 20965, 20965, 20965, 2614, 2614, 2614, 2614, 2614] |
| Oct 22, 2024 @ 17:54:47.000 | ps02-b.farm.particle.cz | clrperf-owamp.in2p3.fr | [147.231.25.253, 195.113.179.105, 62.40.126.254, 62.40.126.254, 62.40.98.52, 62.40.98.158, 62.40.98.183, 62.40.98.239, 62.40.126.26, … | [2852, 2852, 20965, 20965, 20965, 20965, 20965, 20965, 20965, 20965] |
| Oct 22, 2024 @ 17:54:47.000 | perfsonar-bandwidth.esc.qmul.ac.uk | hepsonar1.ph.liv.ac.uk | [194.36.11.1, 138.37.0.79, 146.97.143.217, 146.97.35.233, 146.97.33.22, 146.97.33.42, 146.97.35.46, 146.97.78.33, 146.97.78.38, … | [198864, 198864, 786, 786, 786, 786, 786, 786, 786, 786, 786, 786] |
| Oct 22, 2024 @ 17:54:47.000 | t2ps-bandwidth2.physics.ox.ac.uk | perfsonar02.hep.wisc.edu | [163.1.5.254, 172.24.73.38, 172.31.4.242, 193.63.109.41, 193.63.108.69, 146.97.37.193, 146.97.33.1, 62.40.124.197, 62.40.125.18, … | [786, 0, 0, 786, 786, 786, 786, 20965, 20965, 0, 0, 0, 0, 59, 59] |

# Challenges

- 50% of the paths are incomplete
- some addresses are private



To build **reliable topology** models for identifying weak points on the network, we need to **reconstruct the paths**




## What is the most probable C, given it's between A and B?

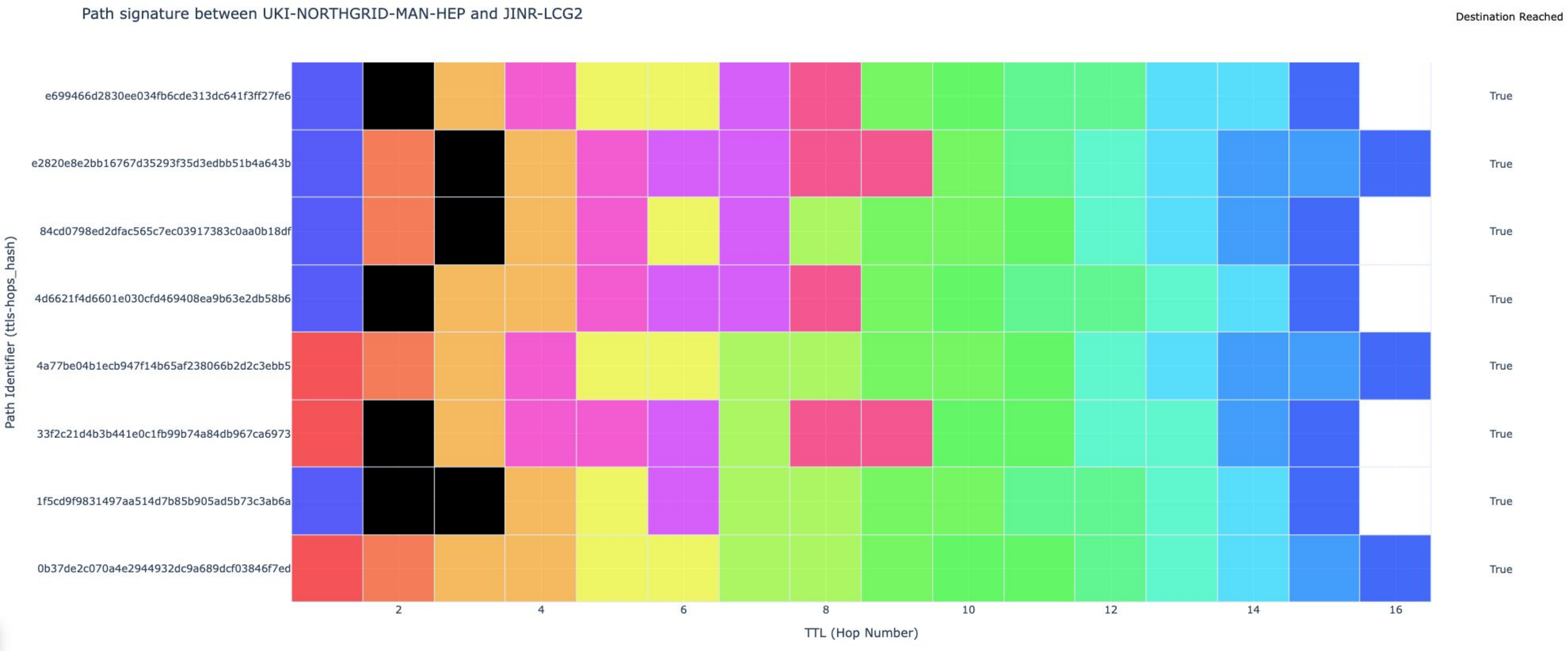
```
Possible intermediates:
r237 P(AtoC)= 0.008  and  P(CtoB)= 0.038
r265 P(AtoC)= 0.056  and  P(CtoB)= 0.009
r536 P(AtoC)= 0.176  and  P(CtoB)= 0.551
r792 P(AtoC)= 0.072  and  P(CtoB)= 0.008
r838 P(AtoC)= 0.008  and  P(CtoB)= 0.01
The most probable intermediate router between r792 and r237 is r536
with a probability of 0.097
```

There are multiple possibilities for C. What is the correct node that lies between A and B depends more on the surrounding nodes rather than on highest probability value

# Site to site path signature

Unknown IP

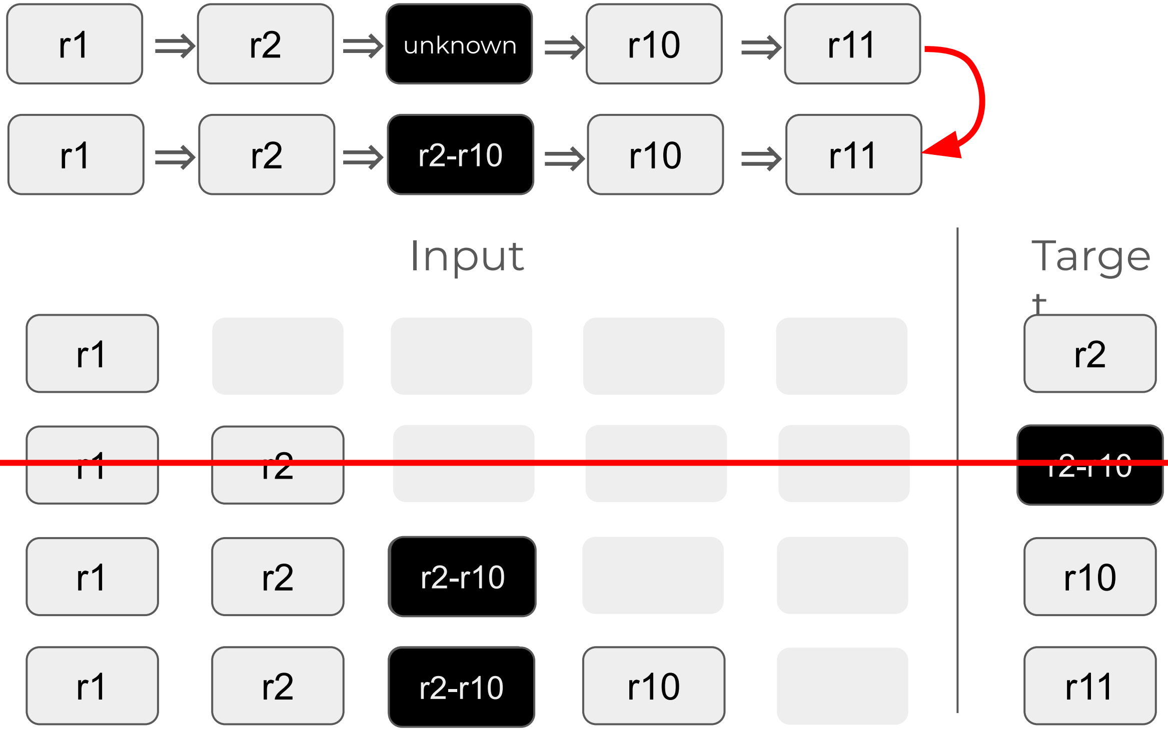
Each color is a different IP



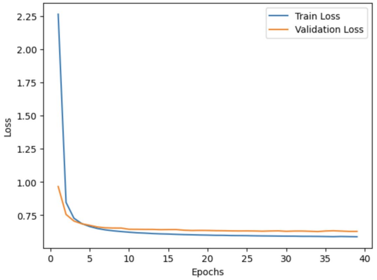
If our eyes can **intuitively spot** these **gaps**, could we **teach a model** to do the same - only **faster, at scale, and with consistent accuracy**?

How about a Transformer model?

# Dataset v0



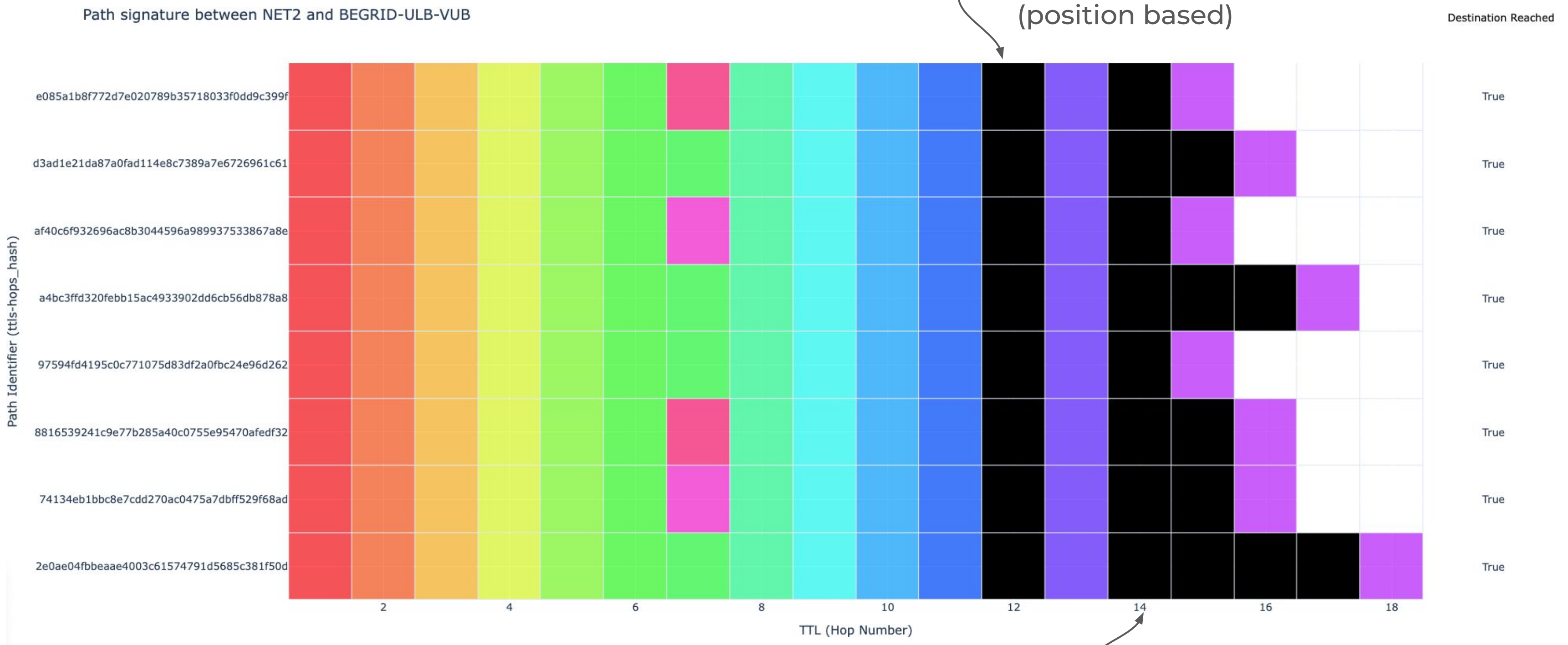

Skip unknown nodes as targets




Model performance v0

# Corner cases

Contextual node (position based)



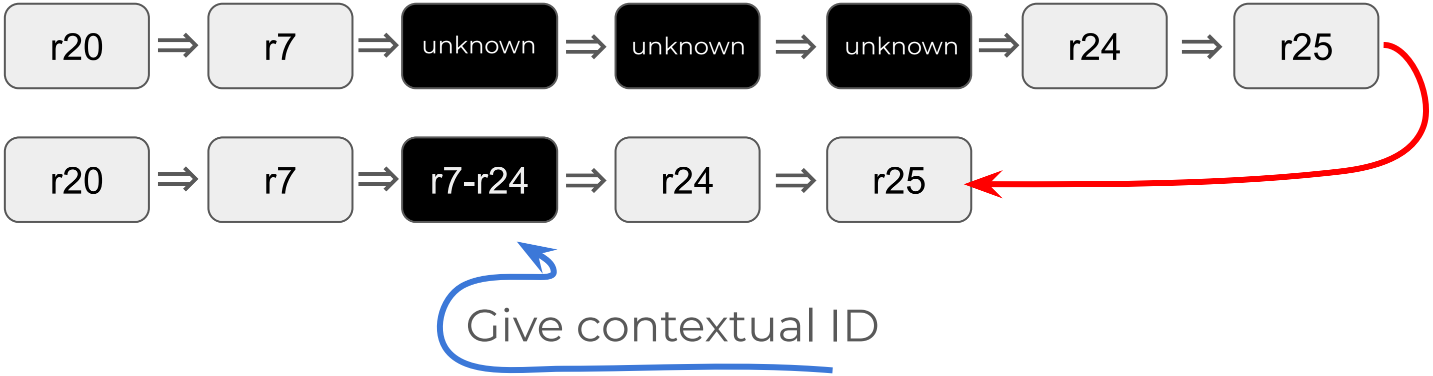

Unknown IP

Each color is a different IP

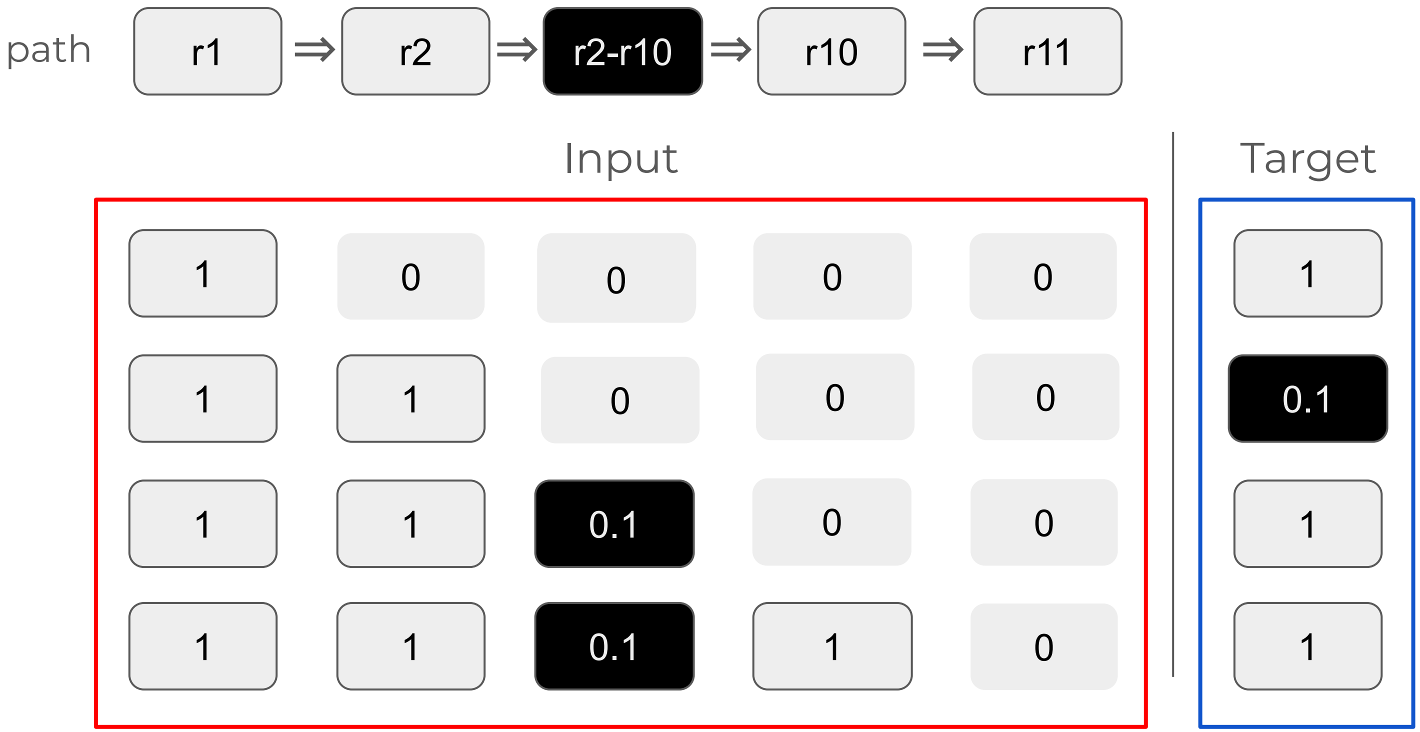
Contextual node or retrying until it gets to the final node?

# Dataset v1

## Collapsed unknowns



# Attention mask and loss confidence



# Attention mask

Helps the model focus more on known and reliable nodes when processing sequences, while still considering the presence of uncertain or unknown nodes

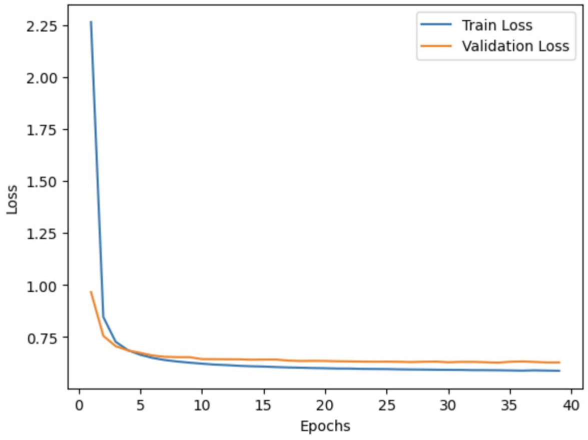
```
# Training phase
for input_batch, mask_batch, target_batch, confidence_batch in train_loader:
    optimizer.zero_grad()

    input_batch = input_batch.to(device)
    target_batch = target_batch.to(device)
    confidence_batch = confidence_batch.to(device)

    # Forward pass
    outputs = model(input_batch, src_mask=confidence_batch)

    # Compute loss
    loss = custom_loss_function(outputs, target_batch, confidence_batch, input_batch)
    loss.backward()
    optimizer.step()

    total_train_loss += loss.item()
```

# Custom loss

Tokens with higher confidence contribute more to the overall loss

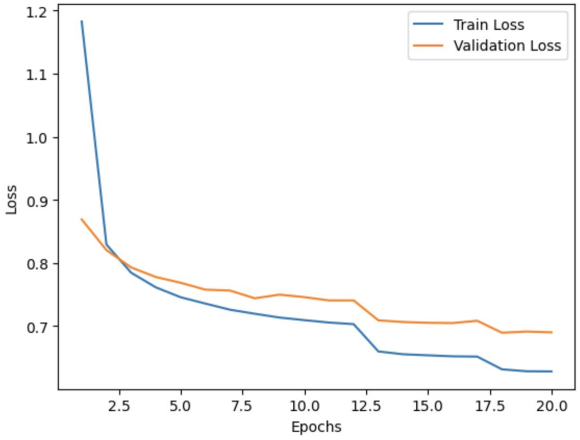
```
def custom_loss_function(outputs, target_batch, confidence_batch, input_batch):
    input_lengths = (input_batch != 0).sum(dim=1)  # Shape: (batch_size,)

    # Get the logits at the last non-padded positions
    last_outputs = outputs[range(outputs.size(0)), input_lengths - 1]  # Shape: (batch_size, vocab_size)

    # Get the confidence scores for the last non-padded tokens
    confidence = confidence_batch[range(confidence_batch.size(0)), input_lengths - 1]  # Shape: (batch_size,)

    loss_fn = nn.CrossEntropyLoss(reduction='none')  # We need element-wise loss
    loss = loss_fn(last_outputs, target_batch)  # Shape: (batch_size,)

    # Apply confidence scores: scale the loss by the confidence score for each sequence
    weighted_loss = loss * confidence  # Shape: (batch_size,)

    return weighted_loss.mean()
```

# Transformer model
work in progress



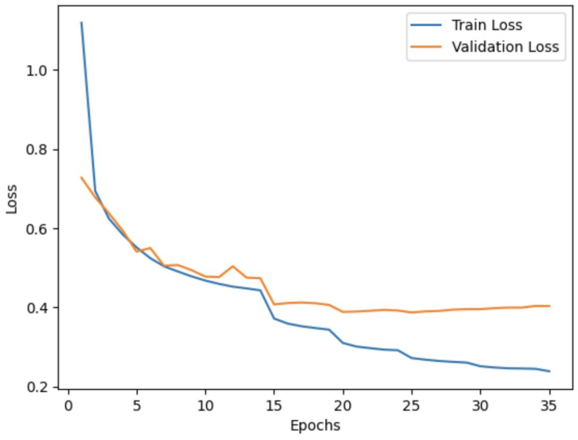

V0

Only certain nodes allowed as targets
Custom loss (with confidence score)




V1

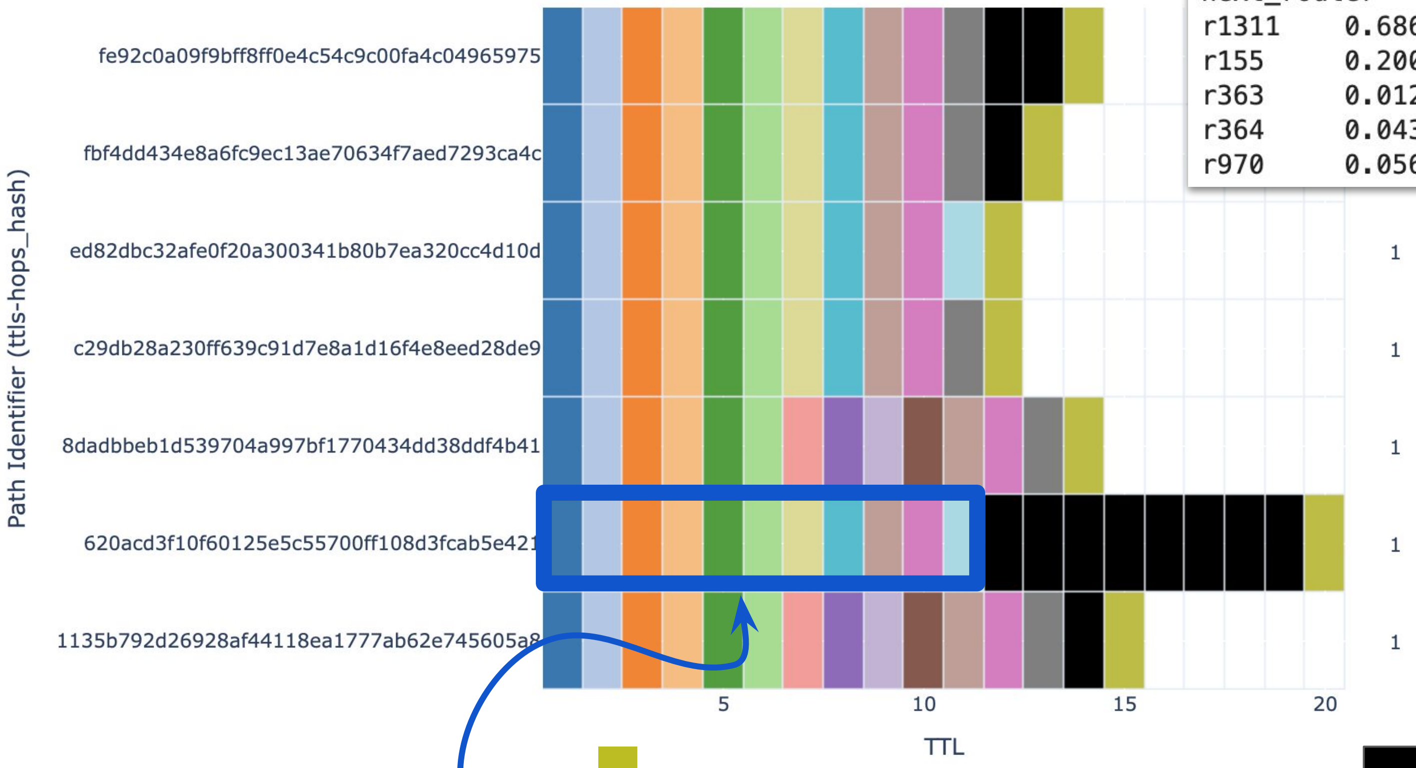
Dynamic LR
Collapsed nodes
Custom loss (using confidence scores)




V2

Dynamic LR
Collapsed nodes
Custom loss (using confidence score)
Confidence mask

# Path signature between FR-ALPAMED-CPPM and JINR-LCG2




All possible transitions from r363:

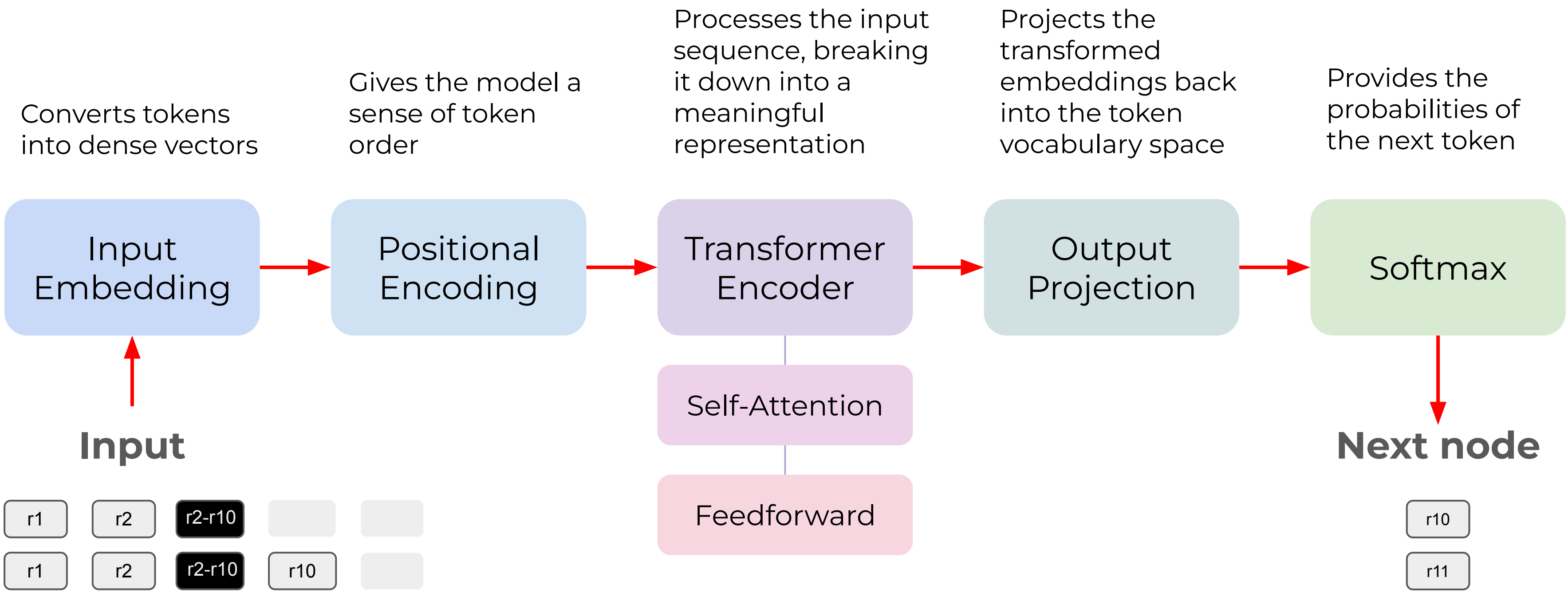
| next_router | |
|---|---|
| r1311 | 0.686825 |
| r155 | 0.200864 |
| r363 | 0.012959 |
| r364 | 0.043197 |
| r970 | 0.056156 |

**Predicted next node: r155**

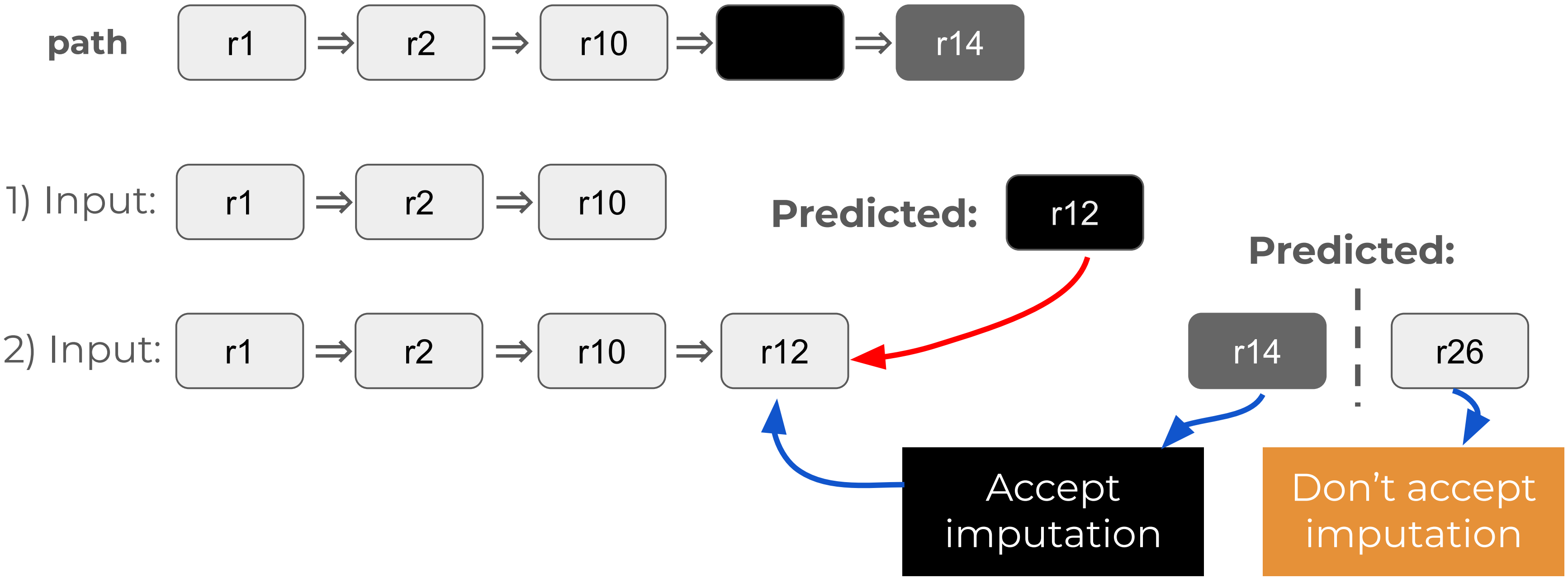
Unknown IP

Each color is a different IP

# Transformer architecture



# Node imputation through self-validation



# Next steps

- Use AutoML tools to finetune the model
- Introduce time dimension
- Implement other models to compare the results
- Deploy the best model and define incremental learning

Once the **topology** is **fixed**, we can proceed by building more **complex models** that incorporate other metrics such as **loss, bandwidth** or **file transfer** statistics

# Acknowledgements

We would like to thank the **WLCG**, **HEPiX**, **perfSONAR** and **OSG** organizations for their work on the topics presented.

In addition we want to explicitly acknowledge the support of the **National Science Foundation** which supported this work via:

- OSG: NSF MPS-1148698
- IRIS-HEP: NSF OAC-1836650

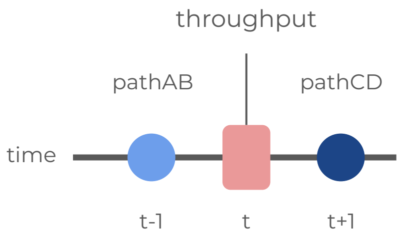
# Thank you!

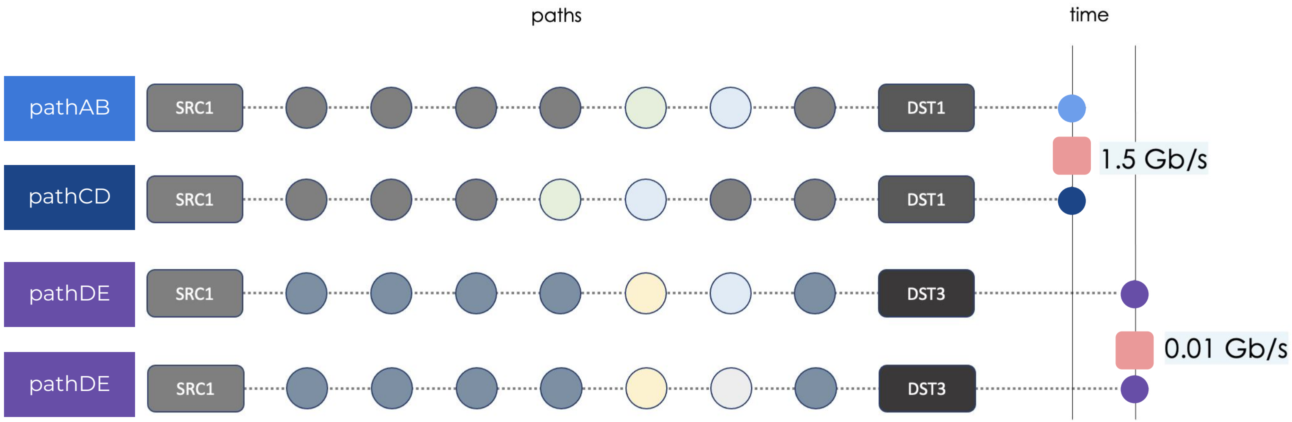
# Any questions?

Contact us: @ net-discuss@umich.edu
Contact me: @ petyav@umich.edu

# Backup slides

# Correlate network tests



# Additional information

Number of paths: 28’444
Vocabulary size: 2’193
Input data shape: (274’997, 30)

Training on 2x NVIDIA GeForce GTX 1080 Ti