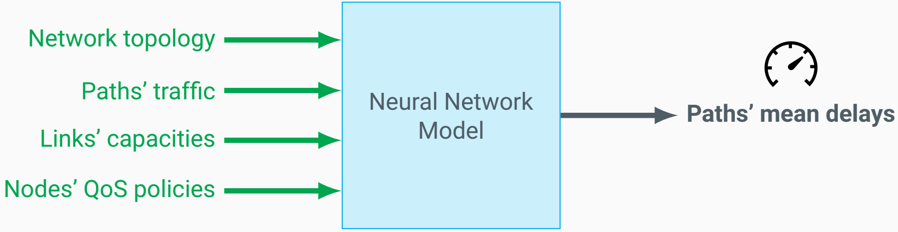
Graph Neural Networks for Physical Networks Modeling

ITU-ML5G-PS-014

Loïck Bonniot Christoph Neumann François Schnitzler François Taïani

Goal: predict delays for any network configuration



Applications:

- SDN optimization
- 5G networks
  “ML-based QoE optimization” [ITU-T, 2019]

Related work

“RouteNet” [Rusek et al., 2019] baseline provided

- Message-passing Graph Neural Network (GNN)
- Models hidden states of **paths** and **links**
- Resolves circular dependencies using **fixed-point** convergence

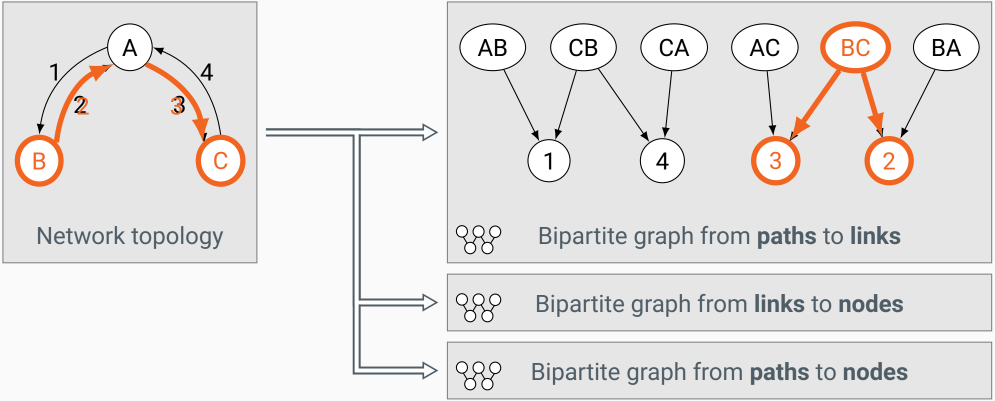
✗ Node states?

✗ QoS policies?

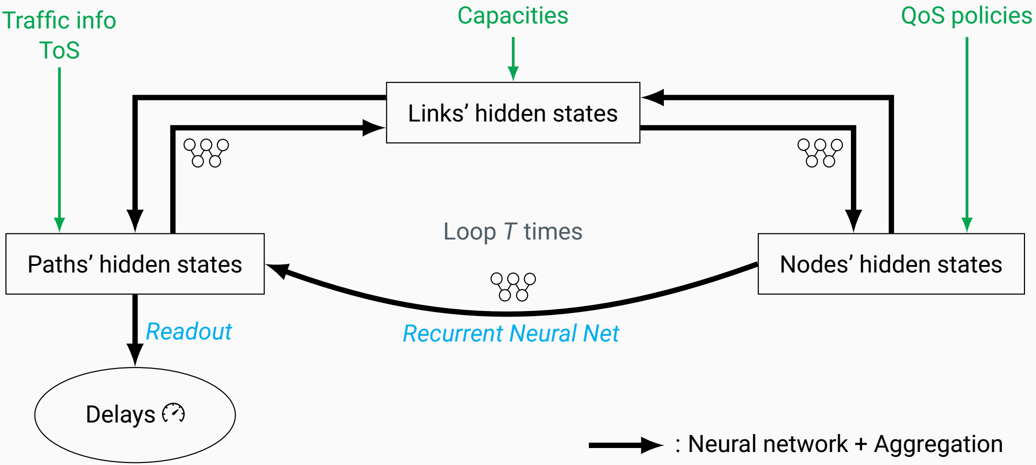
Our GNN Modelization

⟹ Add **nodes** hidden states to model QoS

⟹ Transform network topologies into **3 bipartite graphs** for GNN convolutions



GNN model inner workings



+ specific aggregation, loss function, learning rate scheduler, etc. $\Longrightarrow$ see our report

Demo!

`gnnet.interdigital.com`

Conclusion

interdigital

- Improved **graph modelization** and hidden-state **convergence**
- Custom strategy for **hidden state aggregation**
- More optimizations are detailed in our report
- Demo: `gnnet.interdigital.com`
- Source-code available upon request (contact `christoph.neumann(@)interdigital.com`)

RANK	TEAM	MAPE (%)
1	Steredeg	1.53
2	Salzburg Research	1.95
3	Gradient Ascent	5.42

interdigital

[ITU-T, 2019] ITU-T (2019).
***Machine learning in future networks including IMT-2020: use cases*.**
Supplement 55 to Y.3170 Series, October 2019.

[Rusek et al., 2019] Rusek, K., Suárez-Varela, J., Mestres, A., Barlet-Ros, P., and Cabellos-Aparicio, A. (2019).
Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN.
In *SOSR*.

[Scarselli et al., 2008] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008).
The graph neural network model.
IEEE Transactions on Neural Networks, 20(1):61–80.

Convolutions on bipartite graphs

interdigital

$$\mathbf{x}'_{i,t+1} = \gamma\left(\mathbf{x}_{i,t}, \square_{j\in\mathcal{N}(i)}\, \phi\left(\mathbf{x}_{j,t}\right)\right)$$

- $\mathbf{x}_{i,t}$ "destination" embeddings
- $\mathbf{x}_{j,t}$ "source" embeddings
- $\mathcal{N}(i)$ neighbors of i in graph = list of "sources" j leading to "destination" i
- γ and ϕ are trained perceptrons
- $\square$ is the **aggregation** function
  - Min, Max, Mean, etc. when neighbors are **unordered** (e.g. links to nodes)
  - RNN when neighbors are **ordered** (e.g. paths to nodes)

Training details

- Preprocessing: pandas dataframe, standardization
- Model rewritten completely in Pytorch + Pytorch-geometric
- Embeddings size: 400, leading to 11 465 185 parameters
- Adam Optimizer with CyclicLR Scheduler; no regularization
- $\approx$ 1 week of training on single Tesla M60 GPU