Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning

Fabien Geyer, Georg Carle

Monday 20th August, 2018

ACM SIGCOMM Workshop Big-DAMA’18, Budapest, Hungary

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich
Motivation

Distributed protocols

Today’s distributed network protocols

- Manually developed, engineered and optimized
- Sometimes hard to configure to achieve good performance
- Not always adapted to evolving networks and requirements (e.g. mobile networks, sensor networks, …)

Main research questions

- Can we automate distributed network protocol design using high-level goals and data?
- If yes, can properties such as resilience to faults be included (e.g. packet loss)?

Contribution

- Method for generating protocols using Graph Neural Networks
- Today’s focus: routing protocols
Motivation

Why now?

Two recent trends in networking for enabling such data-driven protocols

- More advanced in-network processing resources and capabilities (e.g. SDN, P4, DPDK, ...) + flexibility
- Data-driven networks and data-driven protocols → See this year’s SIGCOMM workshops
Motivation

Why now?

Two recent trends in networking for enabling such data-driven protocols

- More advanced in-network processing resources and capabilities (e.g., SDN, P4, DPDK, ...) + flexibility
- Data-driven networks and data-driven protocols → See this year’s SIGCOMM workshops

A more general problem in Artificial Intelligence

- Research question: autonomous agents communicating and collaborating to reach a common goal
- Human-level performance in multiplayer games:
  - OpenAI: 5vs5 Dota 2 (August 2018) → https://blog.openai.com/openai-five/

Figure 1: Overview of DeepMind’s Quake 3 challenge (source: https://arxiv.org/abs/1807.01281)
Outline

Introduction

Machine learning

Numerical evaluation

Conclusion
Introduction

Definition

Distributed network protocols

- Distributed nodes need to solve a common high-level goal
- Nodes need to share some information to achieve the goal
- Examples: routing, congestion control, load balancing, content distribution, ... 

Target protocol behavior for this talk: simplified version of OSPF (Open Shortest Path First)
Introduction

Main assumptions

Protocol properties and requirements

- Routing follows a predetermined path-finding scheme (e.g. shortest path)
- Protocol needs to support routers entering and leaving the network
- Protocol needs to be resilient to packet loss
- Should work on any topology

Assumptions

- Routers start with no information about the network topology
- Routers have only their own local view of the network and need to exchange information
Introduction
General idea

- Represent the network as a graph
  - Nodes ↔ Routers (+ some extra nodes)
  - Edges ↔ Physical links
  - Data exchange between nodes ↔ Communication between routers
- Use a neural network architecture able to process graphs
- Train on dataset emulating the network protocol’s goal
Graph Neural Networks

Main concept

**Graph Neural Networks** [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$.

---

**Figure 5**: Example graph

---
Graph Neural Networks

Main concept

**Graph Neural Networks** [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$

### Principle
- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$

![Figure 5: Hidden representations](image)
Graph Neural Networks

Main concept

**Graph Neural Networks** [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$

**Principle**

- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors

![Relationship between hidden representations](image.png)

**Figure 5:** Relationship between hidden representations
**Graph Neural Networks**

**Main concept**

*Graph Neural Networks* [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$

**Principle**

- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $h_v = f (\{h_u \mid u \in Nbr(v)\})$

![Figure 5: Relationship between hidden representations](image-url)
Graph Neural Networks

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$.

Principle

- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$
- ... computed according to the vector of its neighbors
- ... and are fixed points: $h_v = f \left( \{h_u \mid u \in Nbr(v)\} \right)$

Implementation

- The vectors are initialized with the nodes’ input features

Figure 5: Hidden representations initialization
Graph Neural Networks

Main concept

**Graph Neural Networks** [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$

**Principle**
- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $h_v = f \left( \{ h_u \mid u \in Nbr(v) \} \right)$

**Implementation**
- The vectors are initialized with the nodes’ input features
- They are iteratively propagated between neighbors

---

Figure 5: Hidden representations propagation
Graph Neural Networks

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $o_v$.

Principle
- Each node has a hidden representation vectors $h_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $h_v = f \left( \{h_u \mid u \in Nbr(v)\} \right)$

Implementation
- The vectors are initialized with the nodes’ input features
- They are iteratively propagated between neighbors
- ...until a fixed point is found or for a fixed number of iterations

Figure 5: Hidden representations propagation
Graph Neural Networks

Main concept

**Graph Neural Networks** [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes \( o_v \)

**Principle**
- Each node has a hidden representation vectors \( h_v \in \mathbb{R}^k \)
- ...computed according to the vector of its neighbors
- ...and are fixed points: \( h_v = f \left( \{ h_u \mid u \in Nbr(v) \} \right) \)

**Implementation**
- The vectors are initialized with the nodes’ input features
- They are iteratively propagated between neighbors
- ...until a fixed point is found or for a fixed number of iterations
- Those vectors are then used for the final prediction: \( o_v = g(h_v) \)

![Figure 5: Hidden representations fixed point](image-url)
Graph Neural Networks

Details

Implementation

- $f$ and $g$ are neural networks which need to be trained
- $f$ and $g$ implemented as standard feed-forward neural networks in [Scarselli et al., 2009]
- $f$ also implemented using a Gated Recurrent Unit in [Li et al., 2016]
- GNN extended with edge attention to learn which edges are important [Veličković et al., 2018]

Main advantage of GNNs

- Not restricted to a specific graph (i.e. network topology) type such as size, shape, etc.
Generation of distributed protocols

Basic idea

1. Nodes in the computer network periodically broadcast their hidden representation vector
2. Periodically process locally the received hidden representations using the $f$ function previously trained
3. Go to step 1

**Figure 7:** Graph analyzed by the GNN

**Figure 8:** Network topology
Generation of distributed protocols

**Goal**: Given a destination, routers need to know the next hop, i.e. which output interface to use

**Transformation from topology to graph**

- Each router is a node with a router identifier as input feature
- Each interface is a node with a binary output feature: given a destination (i.e. router identifier) use interface or not
- Edges correspond to physical links

![Network topology](image1)

*Figure 9: Network topology*

![Graph encoding of topology](image2)

*Figure 10: Graph encoding of topology*

![Output features based on queried destination](image3)

*Figure 11: Output features based on queried destination*
Generation of distributed protocols

Extension of Graph Neural Networks

- Exchange data about the network topology $\rightarrow h_n^{(t)}$
- Store a local view of the topology (i.e. the hidden representation vector $h_n^{(t)}$)
- Query the local view for routing information (i.e. next hop $q$) $\rightarrow h_n^{(T)} \odot q = o_n$

Figure 12: Graph Query Neural Network
Numerical evaluation

Description

Dataset

- Dataset based on Topology Zoo [Knight et al., 2011][1]
  - Max number of nodes: 20
  - Max hop count: 10
- Randomly add or remove one router in the topologies
- Randomly generate router identifiers
- Total number of generated data points: 40,000

Use-cases

- Shortest-path routing
- Max-min routing

---

Numerical evaluation

Accuracy of predicted routes

In average, 98% accuracy for shortest-path, 95% for max-min

Cumulative distribution

Routing type

Max-min

Shortest path

Average accuracy after fixed number of iterations

Cold start

Warm start
Numerical evaluation

Convergence time

![Convergence time graph]

- **Routing type**: Max-min, Shortest path
- **Start**: Cold start, Warm start
- **Number of iterations of the algorithm**: 0, 5, 10, 15, 20
- **Average accuracy**: 0.00, 0.25, 0.50, 0.75, 1.00
Numerical evaluation

Resilience to packet loss

F. Geyer, G. Carle — Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning
Numerical evaluation

Visualization of the protocol evolution

Figure 13: Evaluated topology. Node 6 is first offline and booted at iteration 20.
Conclusion

Summary

- **Generation of distributed routing protocol using Graph Neural Networks**
- Representation of network topologies as graphs
- Evaluations show that specific protocol properties can be explicitly trained

Key lesson

- Graph Neural Networks are well suited for reasoning about computer networks
- Also been applied to predict bandwidth [Geyer, 2017] and latency of protocols

Future work

- Comparison with manually-engineered protocols
- Generation of other distributed protocols
Performance Evaluation of Network Topologies using Graph-Based Deep Learning.

The Internet Topology Zoo.

Gated Graph Sequence Neural Networks.

The Graph Neural Network Model.

Graph Attention Networks.
In International Conference on Learning Representations.
Backup

Convergence time

Comparison with theoretical protocol based on graph diameter

Routing type
- Max-min
- Shortest path

Theoretical protocol
- Both-ways
- One-way

Cumulative distribution

Convergence difference vs. theoretical protocol