Computer Network

Traffic engineering framework with machine learning in SDN

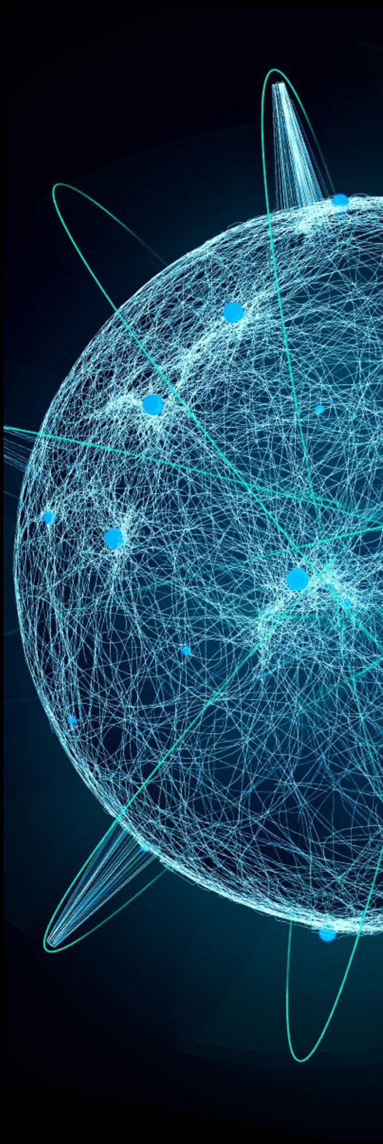
Yeli Zhu, Ruiming Lu, Mengfan Jiang, Chengwei Li, Lixi Zhao

CREATING THE NEXT

Georgia Institute of Technology

Outline

- **Introduction**
- **Background**
  - Architecture
  - Topology
- **Algorithms**
  - Heuristic Algorithm
  - Machine Learning Layer
- **Experiments**
- **Conclusion**



Introduction

Software Defined Network

- **Network Paradigm** enables flexible network resource allocations for traffic engineering
- **Aims** to gain better network capacity and improved delay and loss performance.
- **Separate Control plane and Data Plane**
  - Controller: Complete knowledge of current network state
  - makes all control decisions and manages the overall network behavior

Introduction

Intuition

(1) Routing problem : shortest path first algorithm

Pros : fast, allow large scale problem

cons : inefficient usage of network resource

(2)NP-complete algorithms

Pros: consider the current flow state within the whole network;

Cons.: takes a long time

(3)Machine learning routing

Pros: consider the current flow state within the whole network, fast)

Background: Approach

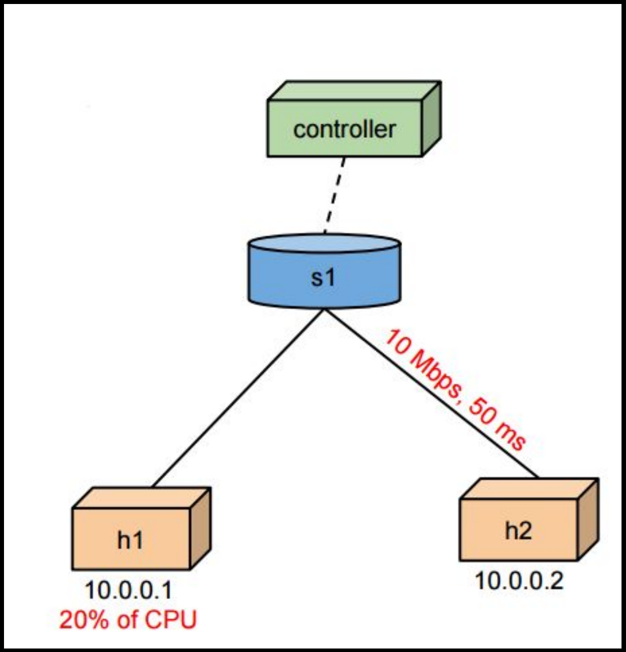
- **NP-Hard Problem**
  - Branch-and-Bound, Local Search, Approximation...
  - Inefficient or Inaccurate
- **Intuition:**
  - Only low delay is essential, not low computation
  - "Preprocessing" makes it faster
- **Machine Learning Model**
  - "Remember" input-output pairs
  - Fast predictor to achieve near-optimal solutions
  - Artificial Neural Networks (ANN)

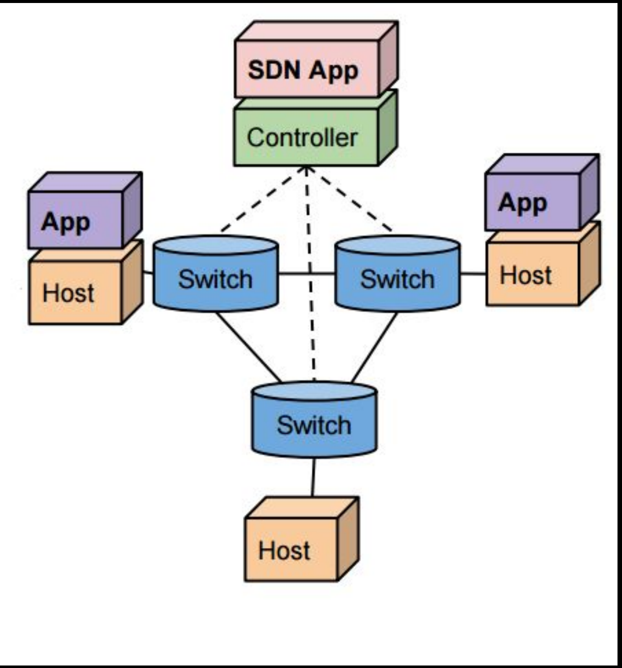
Background: Our Implementation

- **Mininet**
- **Link Layer Forwarding (L2)**
- **Pyretic Controller**
- **Generate Random Traffic : "iperf"**
- **Measure Traffic : "bwm-ng"**
- **Heuristic : Backtracking**
- **Machine Learning : Neural Network (Pybrain)**
- *** Smaller Graphs (Computation Constraint)**

Background: Mininet

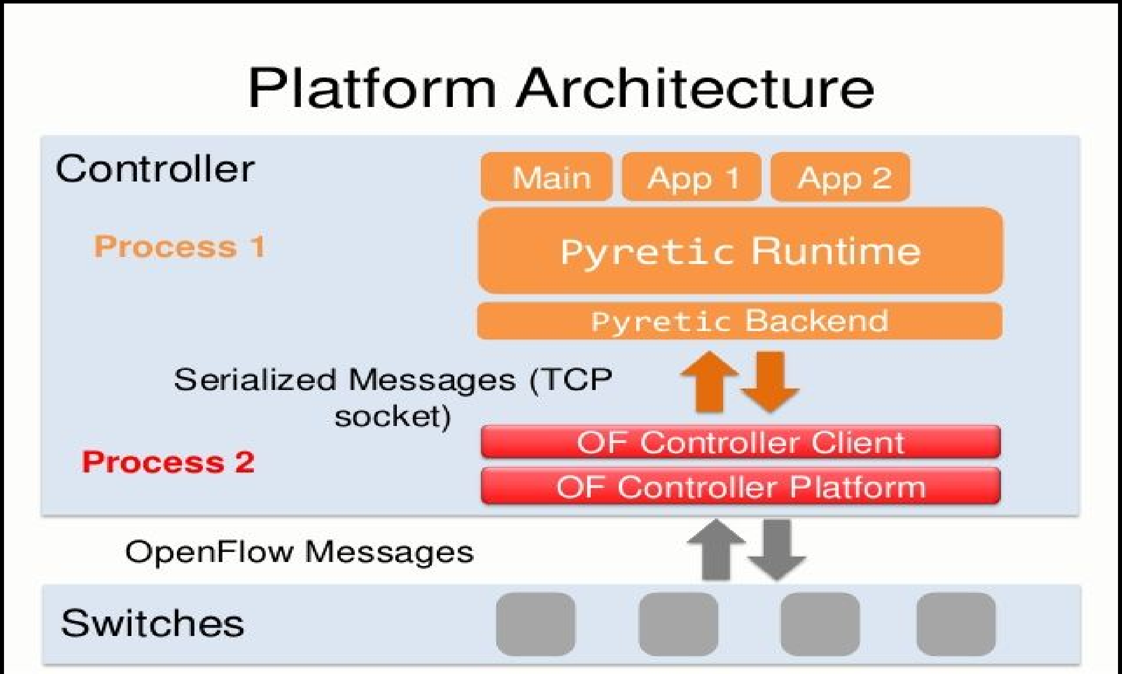
- **Easy to use**
  - Provide Python API
  - Create switch: topo.addSwitch('s1')
  - Execute command: host1.cmd('ping 10.0.0.2')
  - Interactive: CLI
- **More realistic simulation**
  - Real network interfaces
  - Runs unmodified code
- **Customizable**
  - Define your own controller
  - Remote Controller (on another process)
  - Pyretic



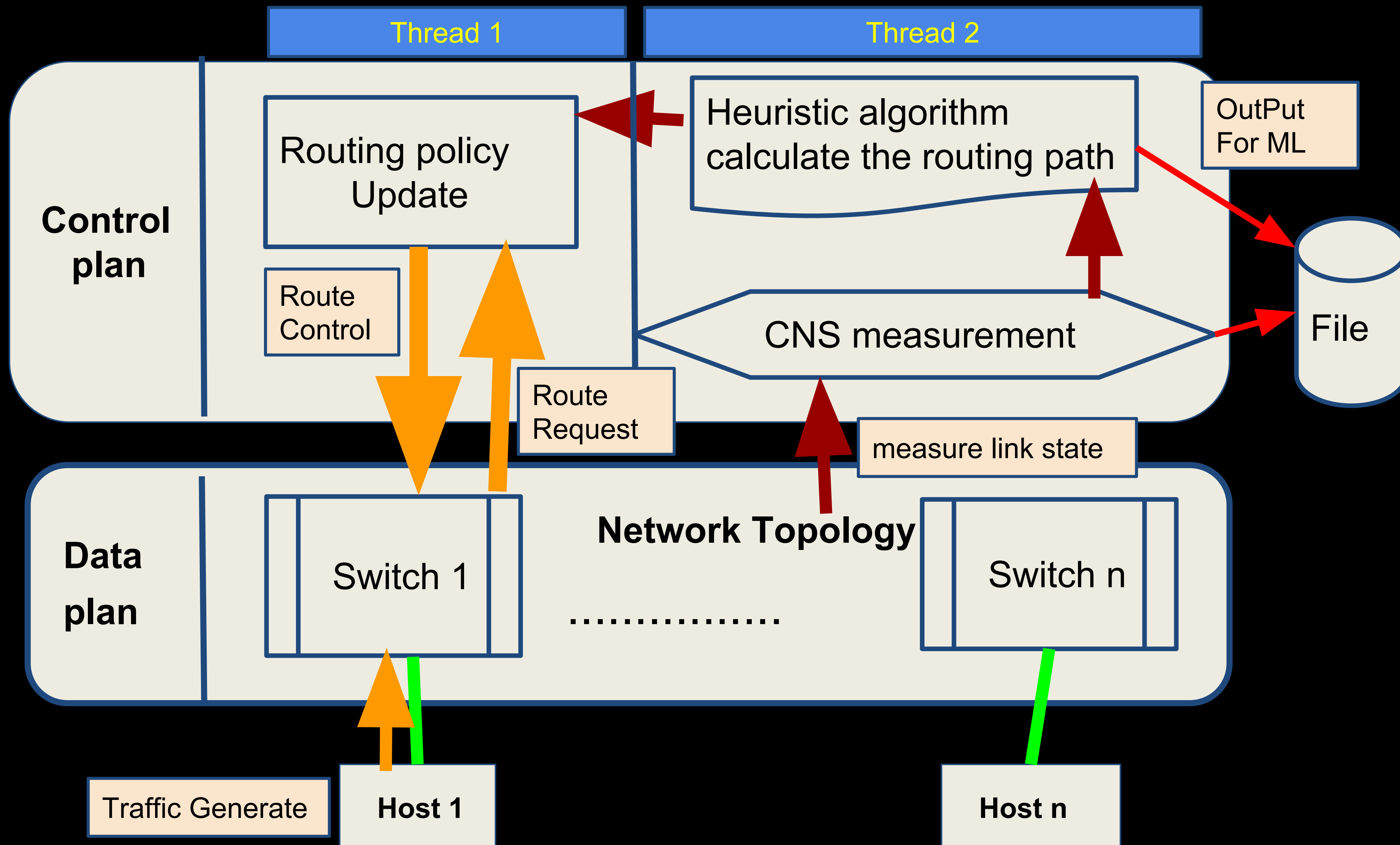


Background : Pyretic Controller

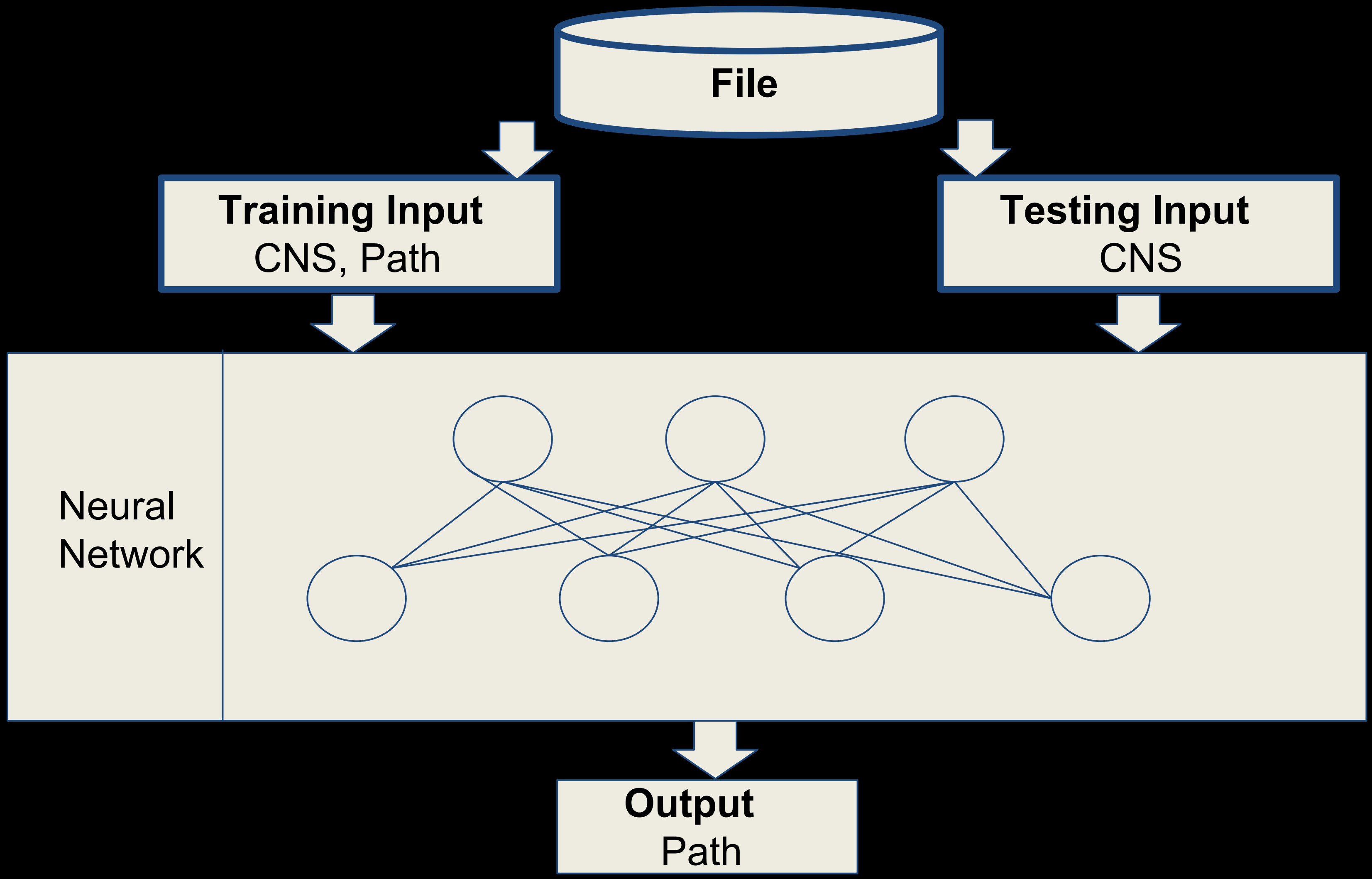
- **Dynamic policy**
- **Simple language to define policies**
- e.g.
  - path = [2, 1, 6, 4, 5]
  - match(srcip='10.0.0.1', dstip='10.0.0.5', switch=1) >> fwd(6)



Architecture



Neural Network Architecture

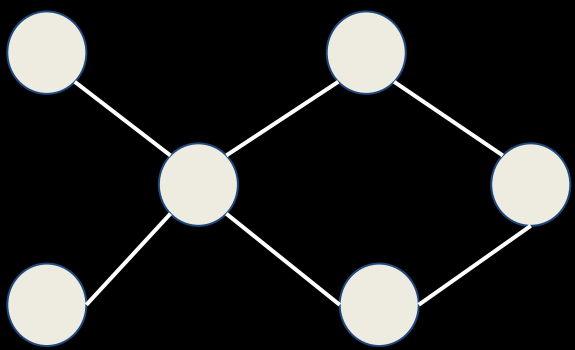


Experiments

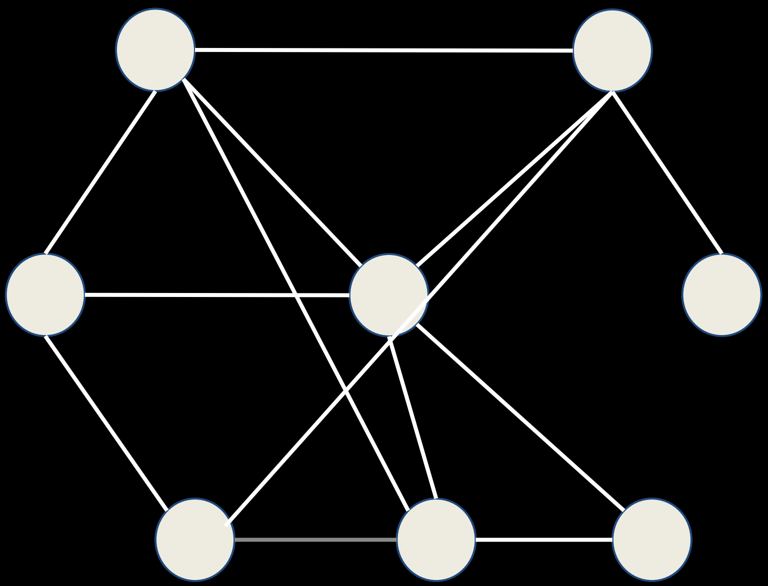
- **Collect Data**
  - Simulate Network Traffic
  - Measure Traffic
  - Update Policy by a Branch-and-Bound Algorithm
  - Collect CNS (Traffic)
- **Training**
  - Train ANN by CNS Data
  - Training Error
- **Comparison**
  - Test Error
  - Computation Speed (Branch-and-Bound v.s. ANN)

Network Topology

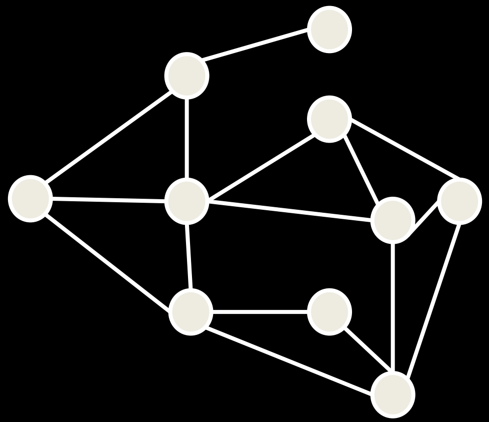
6 Nodes

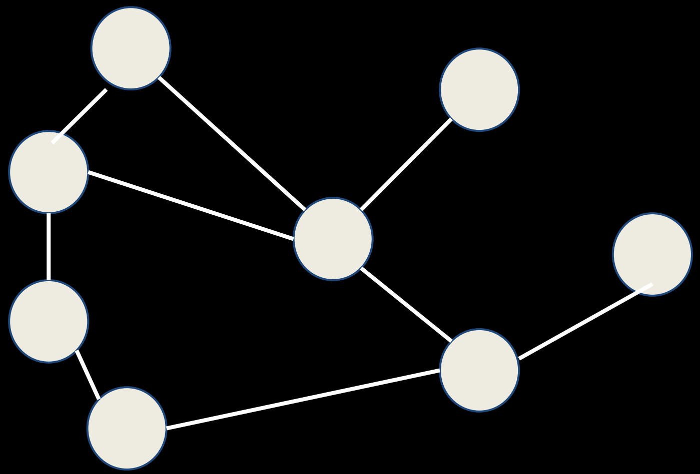


8 Nodes



- **Small Node Number**
  - Computation capability constraint
- **Contains Cycle**
  - Route decision making based on CNS





Heuristic Algorithm VS. Neural Network Algorithm

$$G = (V, E)$$

$$CNS = (t_{uv}, d_{uv})u, v \in V$$

$$\text{Input: } (s, d, D_{max})s, d \in V$$

$$\text{Compute: } p = [s, u_1, u_2, \ldots, u_r, d]$$

$$s.t. \sum_{p} d < D_{max}, t_{s,u_1} + t_{u_1,u_2} + \ldots + t_{u_r,d} \text{ is minimized}$$

Heuristic Algorithm

Input: CNS

Output: global optimal path

Time complexity: O(n!)

Neural Network Model

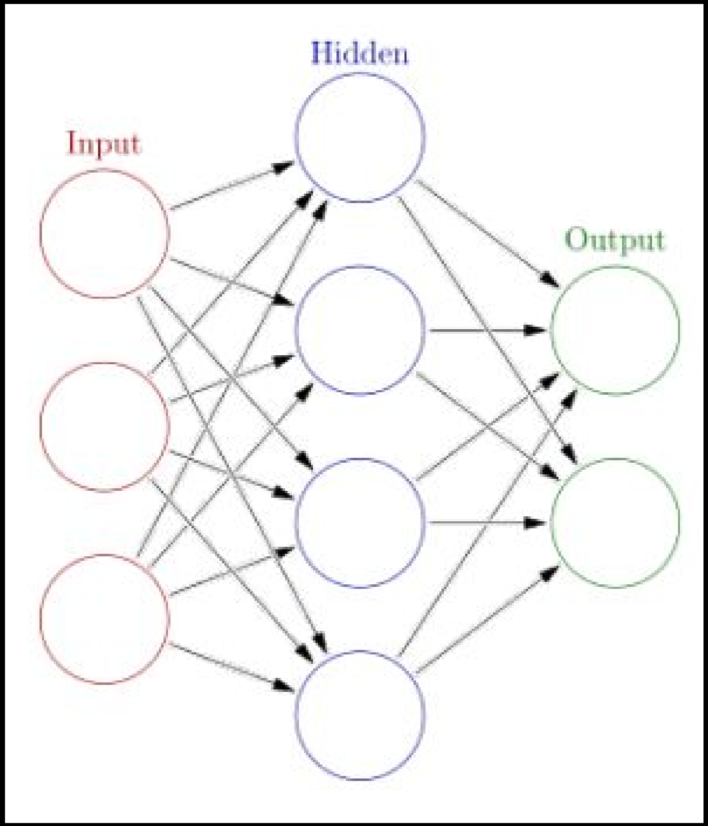
Training input: CNS, optimal path

Testing input: CNS

Testing output: path

Testing time complexity: O(1)

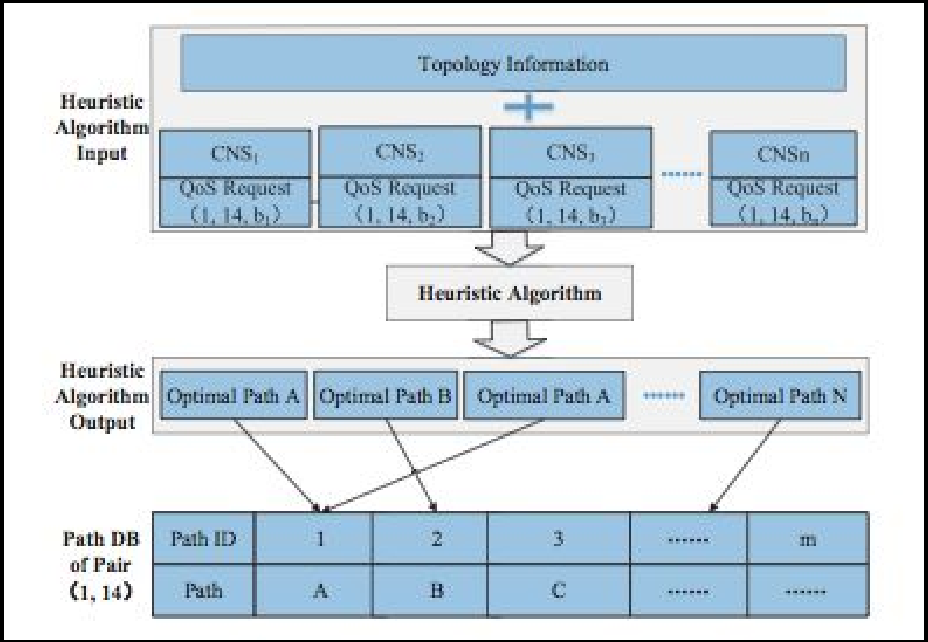
Artificial Neural Network



Artificial Neural Networks are a family of models inspired by biological neural networks and are used to estimate or approximate functions that depend on a large number of input, which suits our purpose well.

After training, we can feed in the current network state and out model will output heuristic-like result in much shorter time compared to heuristic-like algorithms.

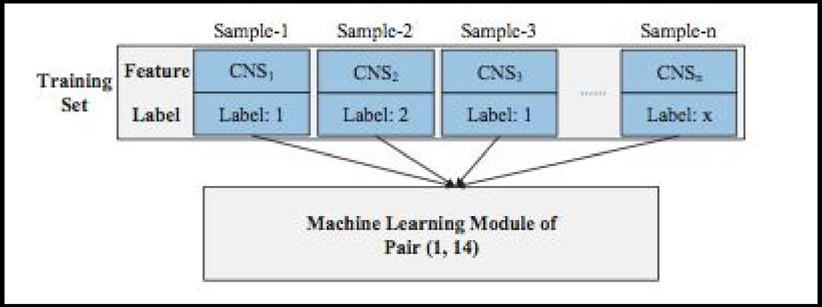
Path Database Construction



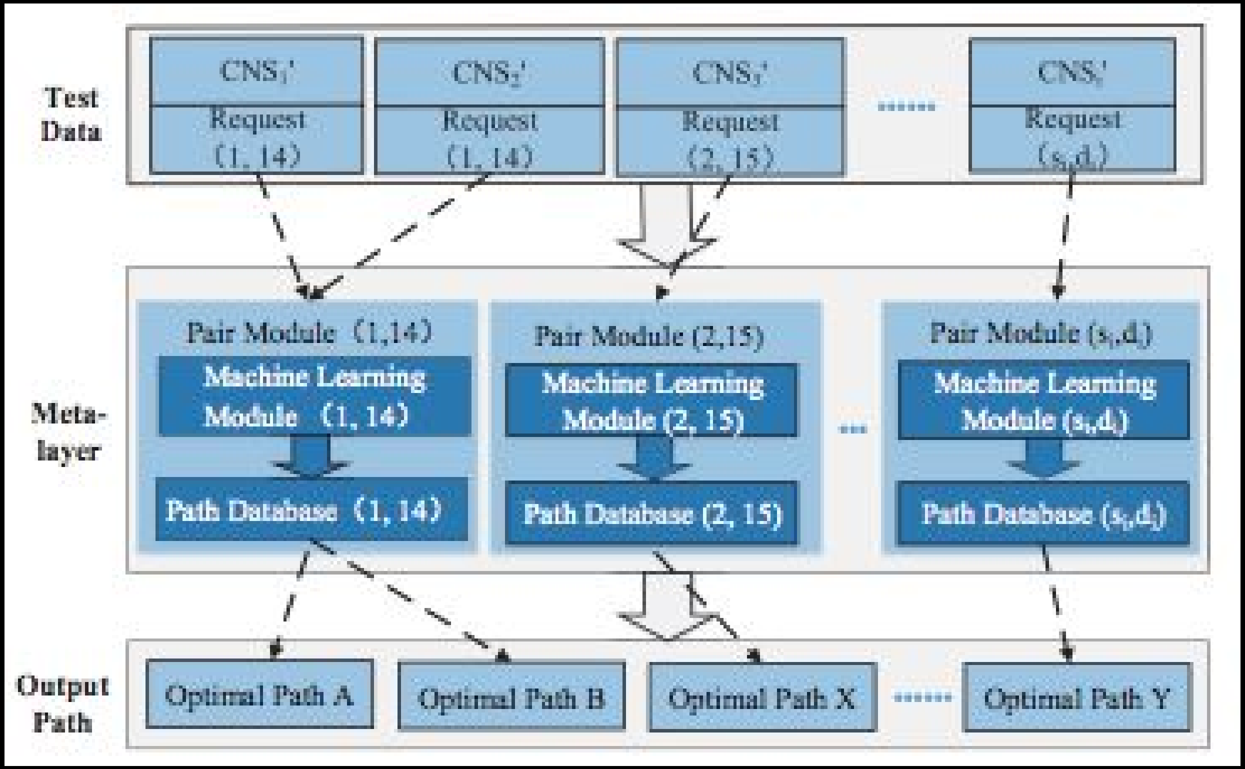
To facilitate the model training, we build the path database for each pair module.

It contains all the unique optimal paths generated by heuristic algorithm, which will indexed by path ID.

During the model training, we feed in the current network state and path ID



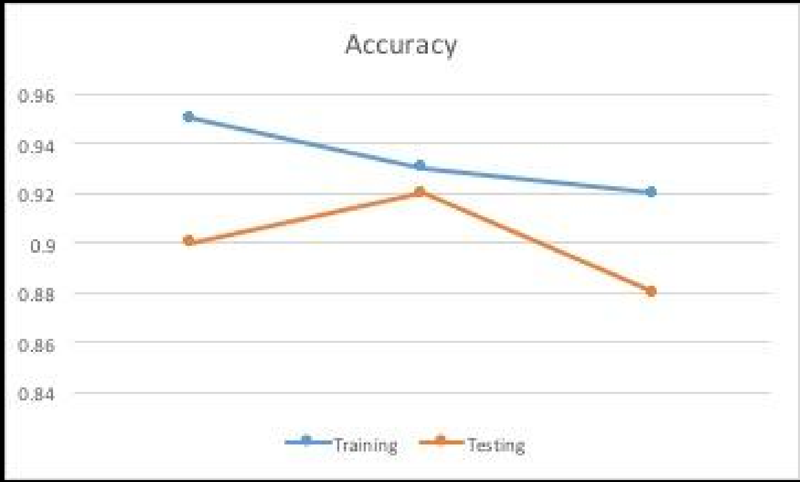
Dynamic routing decision



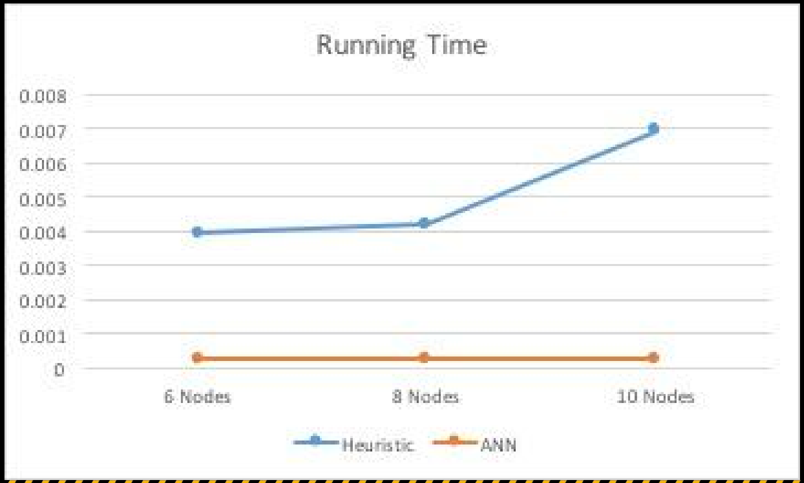
Request → **Corresponding Module** → **Path ID** → **Path**

Experiments (preliminary results)

Machine learning Accuracy



Average Running Time comparison



Conclusion

- The experiments shows that our machine learning algorithm can give heuristic-like result in almost real time.
- Due to computational constraints, we only simulated topologies with 6, 8 and 10 nodes. However, we can still see that ML is able to respond in almost real time, while the heuristic algorithm run time increases significantly as the topologies become larger.
- One the other sides, for the machine learning algorithm we need to measure the traffic and record it to get the ANN model which will cost space.
- Our experiment is based on a stable network topology, but in reality, it need to design much more complicated model to deal with.

References

[1] Li Yanjun, Li Xiaobo, ”Traffic engineering framework with machine learning based meta-layer in software-defined networks” Network Infrastructure and Digital Content (IC-NIDC), 2014 4th IEEE International Conference, pp. 121-125, Sep 2014.

[2] Lan F.Akyildiz, Ahyoung Lee, ”A roadmap for traffic engineering in SDNOpenFlow Networks” Computer Networks, pp. 121-125, Sep 2014.

[3] A Sridharan, R Guerin and C. Diot, ”Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks” IEEE/ACM Transactions on Networking (TON) 13, vol. 2, pp.234-247 2005

[4] T. V. Lakshman, ”Traffic engineering in software defined networks” INFOCOM, 2013 Proceedings IEEE

[5] Kwang Mong Sim and Hong Sun Weng,”Ant colony optimization for routing and load-balancing: survey and new directions” Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 33, no. 5, pp.560 -572 2003

[6] Kodialam, Mudi, and T. V. Lakshman. ”Minimum interference routing with applications to MPLS traffic engineering.” INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 2. IEEE, 2000. APA

[7] Fortz, Bernard, and Mikkel Thorup. ”Internet traffic engineering by optimizing OSPF weights.” INFOCOM 2000. Nineteenth annual joint conference of the IEEE computer and communications societies. Proceedings. IEEE. Vol. 2. IEEE, 2000.

[8] Kodialam, Mudi, and T. V. Lakshman. ”Minimum interference routing with applications to MPLS traffic engineering.” INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 2. IEEE, 2000.

Thanks