Deep Reinforcement Learning for Network Routing

Redha Abderrahmane ALLICHE

Supervisors:

- Ramon APARICIO-PARDO
- Lucile SASSATELLI

About me

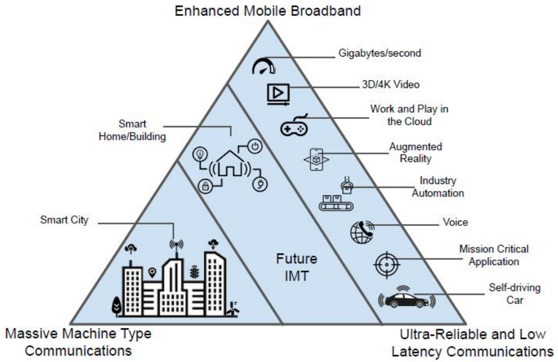
- Oct 2020: Starting the thesis
- Formation:
  - 2020: Master degree in Data Science at the University Paris-Saclay
  - 2018: Bachelor in Telecommunication in Algeria
- Work Experience:
  - 2020: Intern at CNAM : Machine learning projects
  - 2019: Intern at LATMOS/CNRS: DL applied to weather forecasting
  - 2018: Intern at Nokia: Developing tool to configure net-devices

Outline

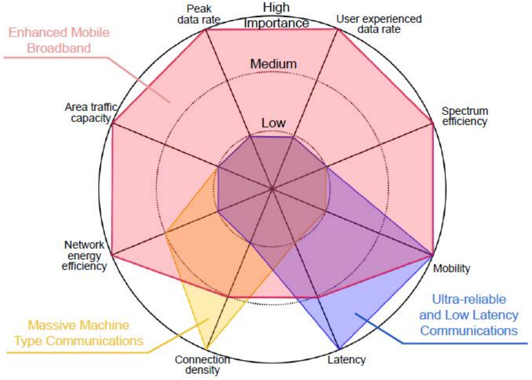
- Introduction and Context
- DRL applied to network routing
    a. Distributed packet routing case
    b. Routing and allocation of network slices case
- Conclusion

Introduction

- Context
  - 5G brings new challenges in terms of:
    - High mobility
    - Massive device connectivity
    - QoS requirements



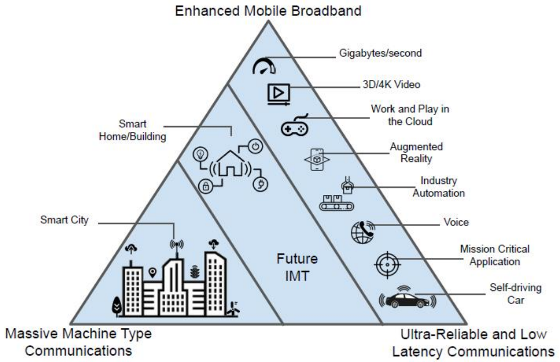
(a) 5G Use Cases



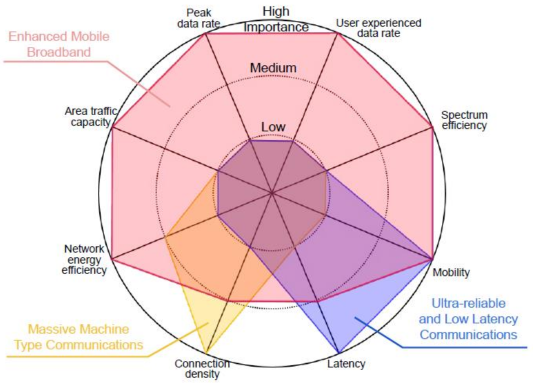
(b) 5G Needs classes

Introduction

- Context
  - These challenges are hard to handle using classical optimization algorithms, because of:
    - Number of variables
    - Dynamically changing environment
    - Variety of network demands



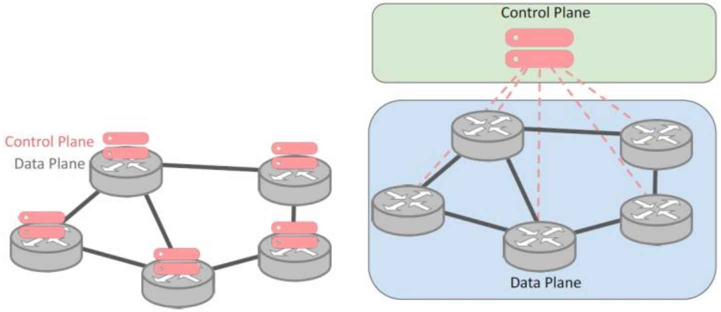
(a) 5G Use Cases



(b) 5G Needs classes

Introduction

- Context
  - Objective
    - Make networks smart and robust
  - SDN makes network management more flexible by decoupling the control plane from the data plane

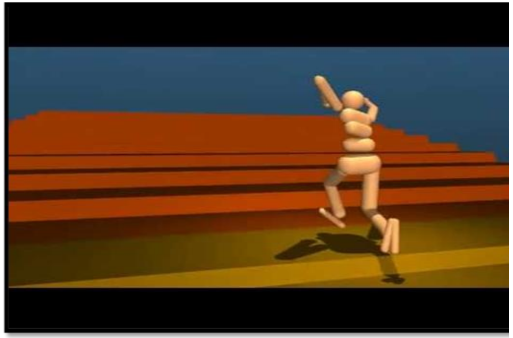


(a) Traditional Network   (b) SDN Network

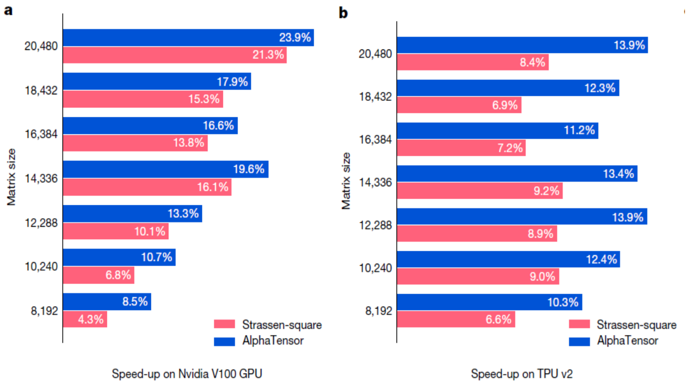
Introduction

- Context
  - Deep Reinforcement Learning (DRL) has achieved incredible performances in many domains (e.g. robotics and video games)

Mnih et al. (2015)



Heess et al. (2017)

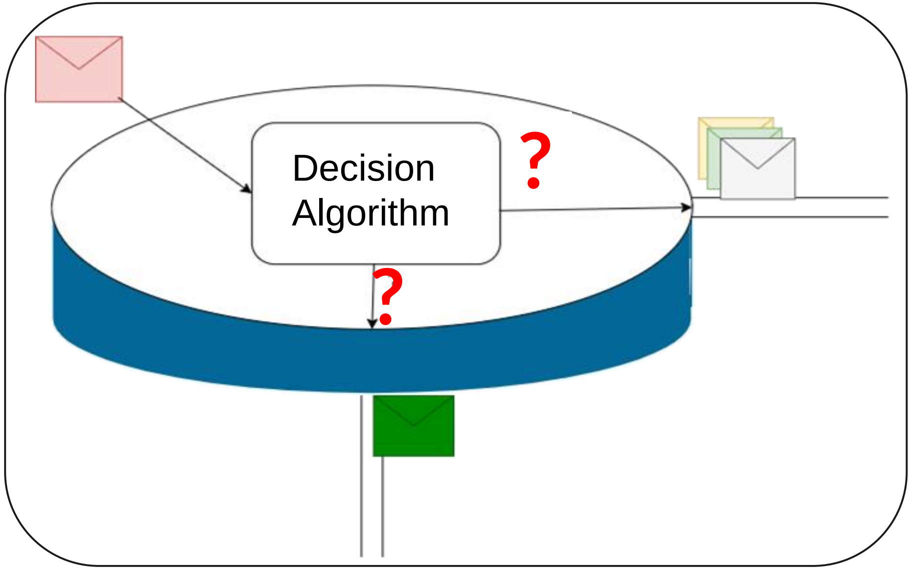


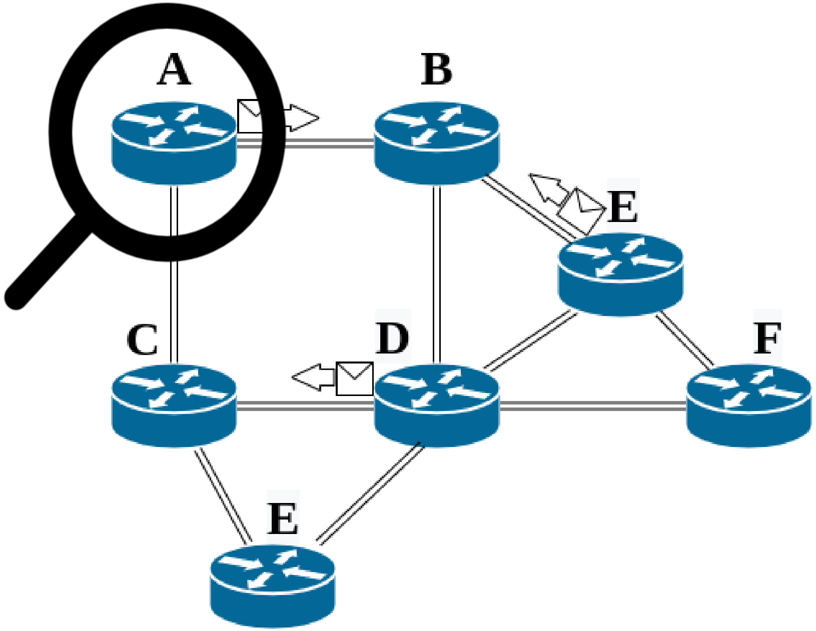
Fawzi et al. (2022)

Introduction

- Presented use cases
  a. Distributed packet routing case
     - Propose a Multi-agents DRL for distributed packet routing
     - PRISMA: Network simulator based on ns-3 to evaluate the solution
     - Evaluate the impact of information sharing overhead of the approach
  b. Routing and allocation of network slices case
     - Propose an optimization model for network slice reconfiguration
     - Design an RL model to find the best time to reconfigure the network slices

Distributed Routing





Distributed Routing

Challenging cases:

- Can only exploit local information
- No complete view of the topology
- Traffic features unknown, e.g., traffic matrix
- Optimize complex metrics related to QoE

⇒ Multi-agent Deep Reinforcement Learning (MADRL)

Examples:

- Wireless ad-hoc networks
- Multi-domain networks
- Cloud overlay routing

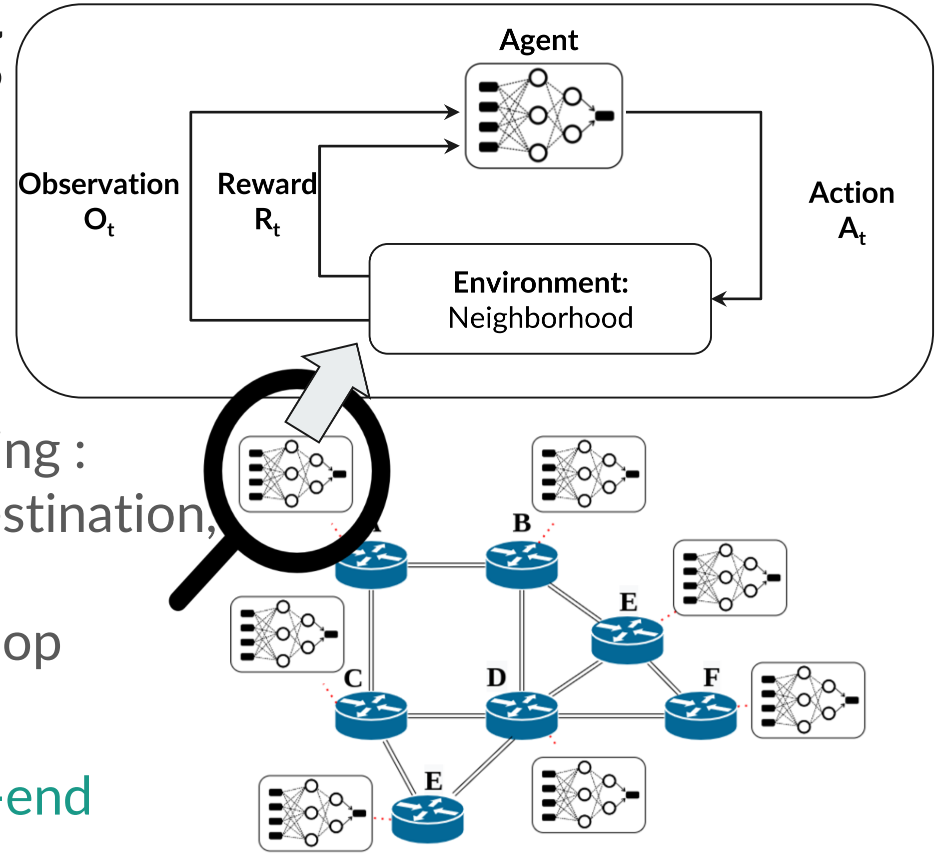
Classical algorithms:

- AODV
- BGP
- Cisco OMP

Distributed Routing

Multi-Agent Deep Reinforcement Learning Approach

- Definition
- Application to Distributed Routing :
  - **Observation (O_t)**: packet destination, link's buffer occupation
  - **Reward (R_t)**: delay to next hop
  - **Action (A_t)**: next hop index
- **Objective**: minimize the end-to-end delay for all the packets



Context

- Related work:
  - Deep Q Routing with Communication (DQRC) [X.You et al., IEEE Transactions on Systems, Man, and Cybernetics, 2020]
  - Relational Features for Routing Decision [V. Manfredi et al., WoWMoM, 2021]

Context

- Issues with the state of the art:
  1. Use non-realistic and non-standard ad-hoc Python based simulator
  2. Do not evaluate the impact of information sharing (overhead) between agents

Contributions

1) Provide realistic and standard simulator
   - ✓ PRISMA: A Packet Routing Simulator for Multi-Agent Reinforcement Learning; 4th International Workshop on Network Intelligence 2022 (NI 2022)
2) Evaluate the information sharing overhead of the MADRL approach
   - ✓ Impact Evaluation of Control Signalling onto Distributed Learning-based Packet Routing; In 34th Intl. Teletraffic Congress, ITC 2022.

Contrib 1 : PRISMA

- Existing Simulation Tools
  - NS3 : Networking Simulator [G. Riley, Modeling and Tools for Network Simulation, 2010]
  - NS3-gym : NS3 and OpenAI Gym [P. Gawłowicz et al., MSWiM, 2019]

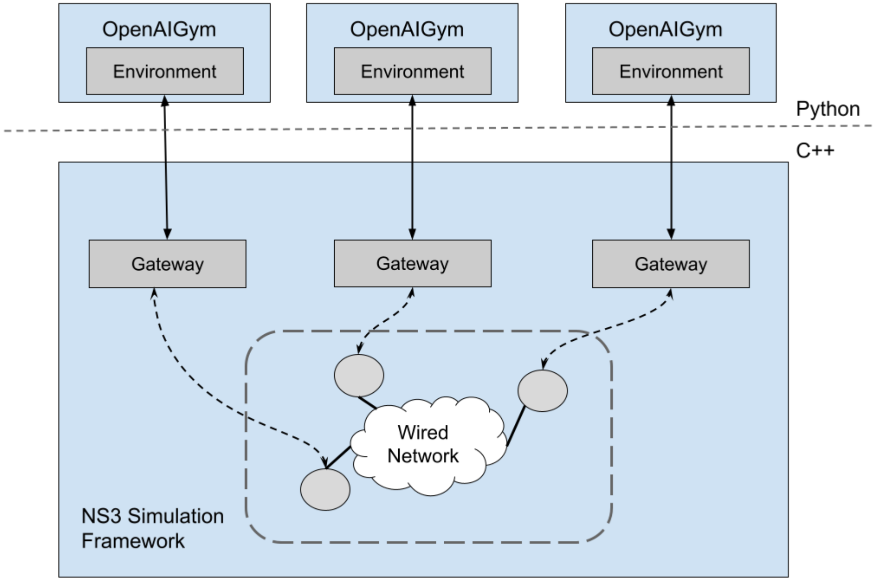
OpenAI

ns-3 NETWORK SIMULATOR

- **Not adapted for MADRL in Networking**

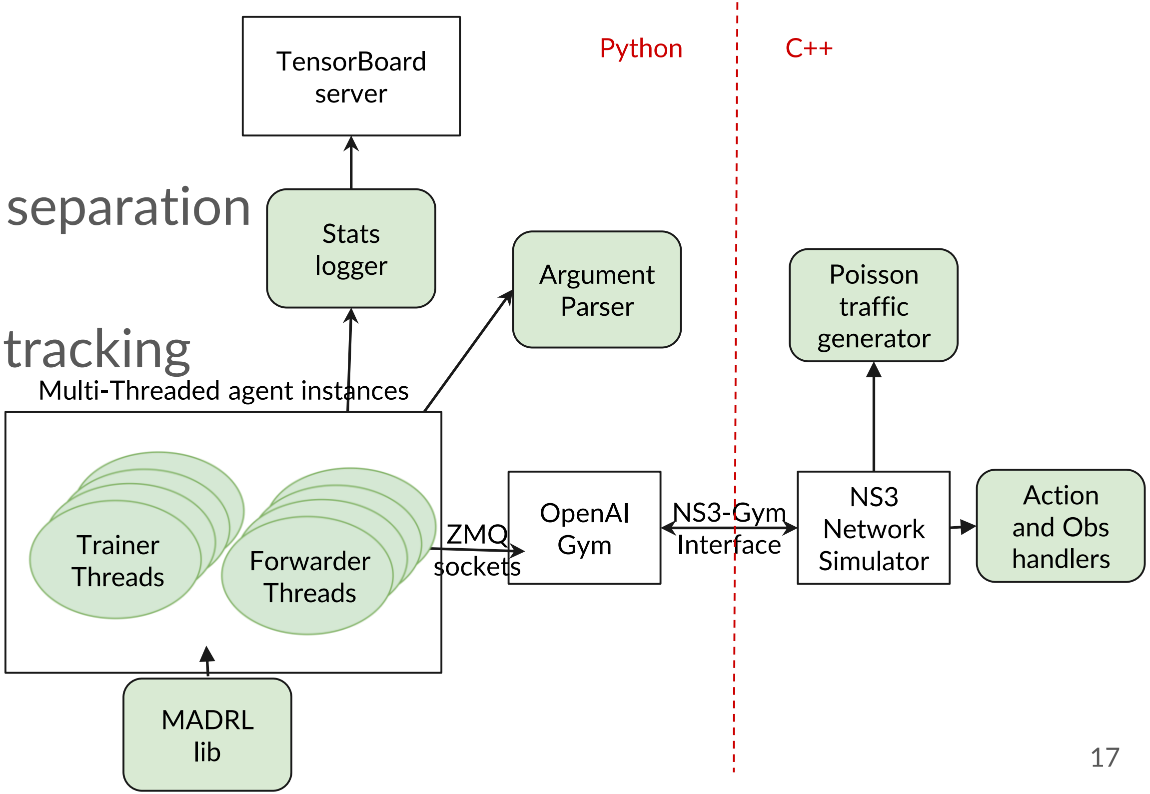
Contrib 1 : PRISMA

- Goal :
  - Extend NS3-gym to Multi-Agent Reinforcement Learning approach



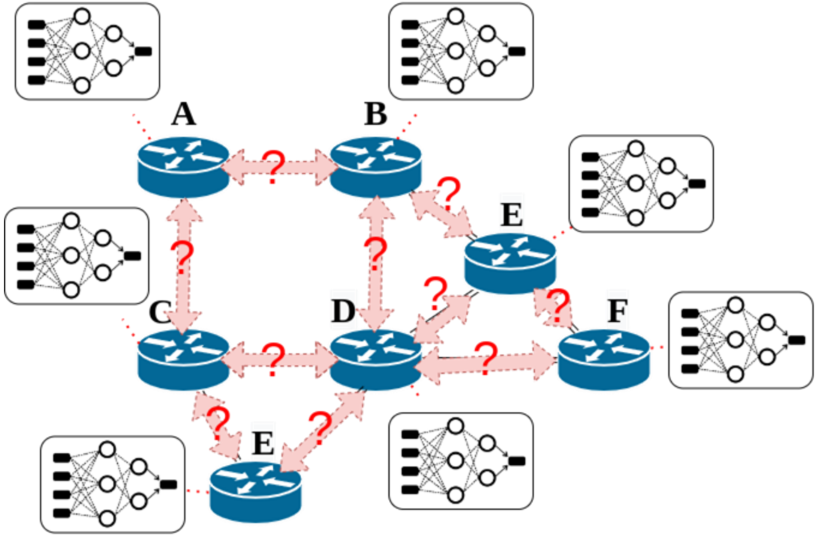
Contrib 1 : PRISMA

- **Features :**
  - Training-Forwarding separation
  - Realistic simulation
  - Real-time simulation tracking
  - Fast prototyping
  - Modularity



Contrib 2 : Impact of signalling

- Goal :
  - Evaluate the impact of implementing MADRL in a production network



Contrib 2 : Impact of signalling

- **Overview of the approach**

$$Y_n = r_{n'} + \gamma \cdot \min_{a_{n'} \in \mathcal{A}_{n'}} Q_{n'}(o_{n'}, a_{n'}; \theta_{n'}) \cdot (1 - f)$$

Where ***n*** is the actual node index and **n'** is the neighbor node index

Contrib 2 : Impact of signalling

- **Overview of the approach**

$$Y_n = \boxed{r_{n'}} + \gamma \cdot \boxed{\min_{a_{n'} \in \mathcal{A}_{n'}} Q_{n'}(o_{n'}, a_{n'}; \theta_{n'})} \cdot (1 - f)$$

Where ***n*** is the actual node index and **n'** is the neighbor node index

  - Value sharing
    - Sharing the target value and the reward

Contrib 2 : Impact of signalling

- **Overview of the approach**

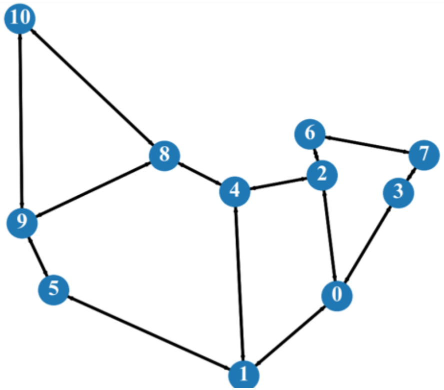
$$Y_n = \boxed{r_{n'}} + \gamma \cdot \min_{a_{n'} \in \mathcal{A}_{n'}} Q_{n'}(\boxed{o_{n'}}, a_{n'}; \boxed{\theta_{n'}}) \cdot (1 - f)$$

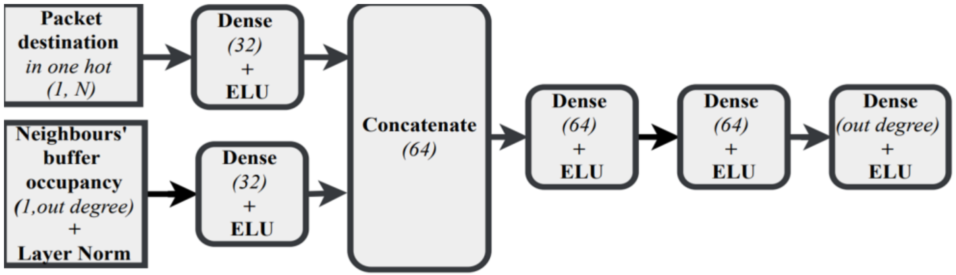
Where ***n*** is the actual node index and **n'** is the neighbor node index

  - Model sharing
    - Sharing the state and the reward
    - Sharing the parameters of Deep-Q network

Contrib 2 : Impact of signalling

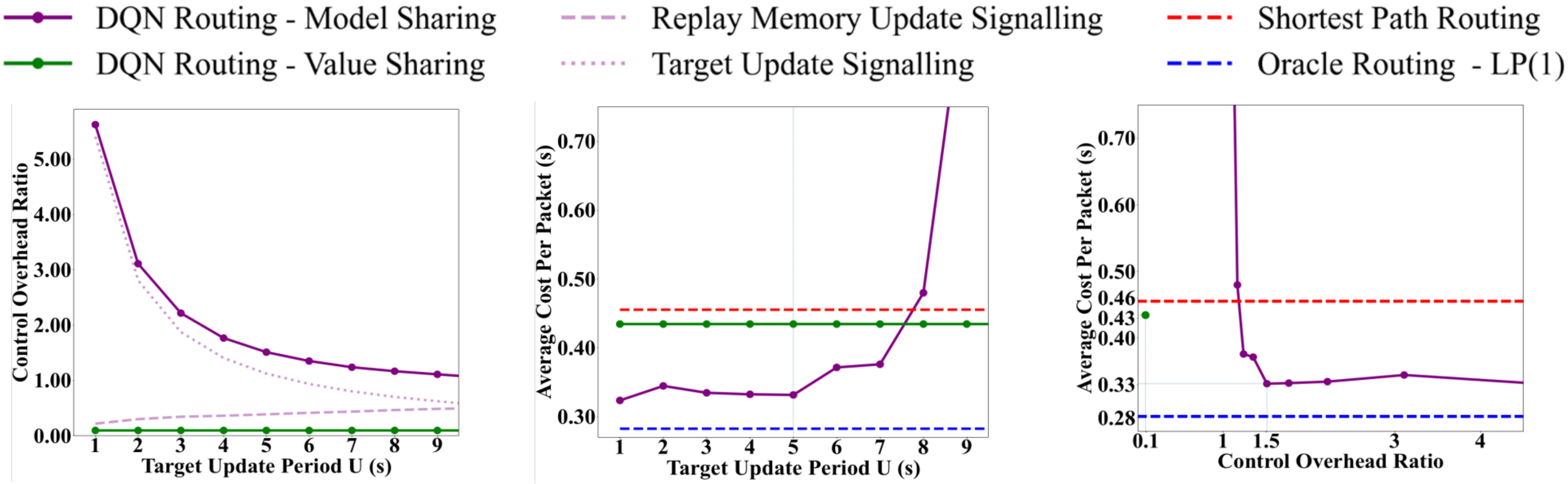
- **Experimental setting**
  - Topology : Abilene (11 nodes)
  - Traffic model : Poisson generator
  - Traffic matrix : Random uniform distribution
  - Model : Deep Q Network





Contrib 2 : Impact of signalling

- Results

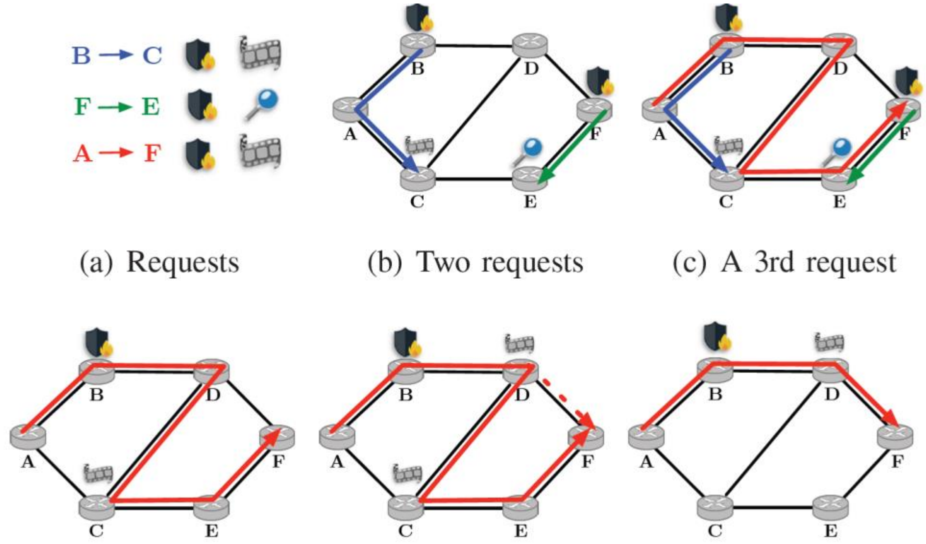


Reconfiguring Network Slices

- Network Slicing Principle
  - The idea is to divide the network infrastructure to multiple logical networks
  - A network slice needs to fulfill an end-to-end service demand:
    - A network path from the source to the destination providing the required bandwidth
    - A Set of network functions needed by the service
  - **Goal**: allocate slices in order to reduce the **resource utilization** and thus accept the **maximum number of requests**

Reconfiguring Network Slices

- Network Slice Reconfiguration Example



(a) Requests (b) Two requests (c) A 3rd request

(d) First two requests leave (e) Reconfiguration phase (f) Optimal routing for the third request

Reconfiguring Network Slices

- Network Slicing Problem
  - Find the best time to reconfigure the network slices
    - Reconfigure more often ⇒ high management cost
    - Reconfigure less often ⇒ suboptimal network usage

- **Proposition**: *Deep-Rec*, smart reconfiguration management agent that chooses when to initiate reconfiguration depending on the traffic dynamics and network congestion

Deep-REC

- **Action**: perform or not a reconfiguration
- **State**:
  a. Number of minutes since the last reconfiguration
  b. Number of slices added since the last reconfiguration
  c. Number of slices released since the last reconfiguration
  d. Current profit
  e. Current time t.

- **Reward**:
  - if no reconfiguration

    $$r = 0$$

  - if reconfiguration

    $$r = \Delta p_R - \Delta p_{NR} - \upsilon_R$$

    where,

    $$\Delta p_{NR} = \{\sum_{k=t}^{t+3} p_k | \text{no reconf at } t\}$$

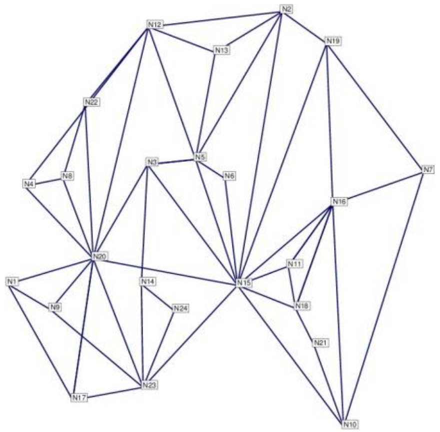
    $$\Delta p_R = \{\sum_{k=t}^{t+3} p_k | \text{reconf at } t\}$$

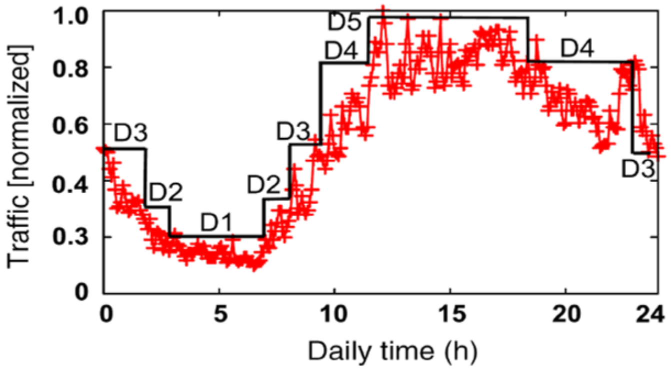
    υ_R : Artificial penalty

Experimental setting

- Network topology: `ta1` (24 nodes, 55 links, 6 datacenters)
- Frequency of action = 5 minutes
- Artificial cost per reconfiguration = cost of deploying a VNF for 15 minutes
- Slice services chain = 5
- 4 types of service considered :

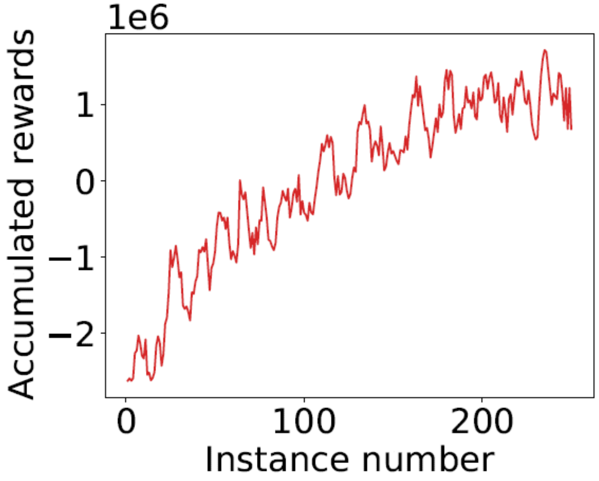
Slice Types	VNF chain	Latency	bw (Mbps)
Web Service	NAT-FW-TM-WOC-IDPS	10ms	100
Video Streaming	NAT-FW-TM-VOC-IDPS	5ms	256
VoIP	NAT-FW-TM-FW-NAT	3.5ms	64
Online Gaming	NAT-FW-VOC-WOC-IDPS	2.5ms	50



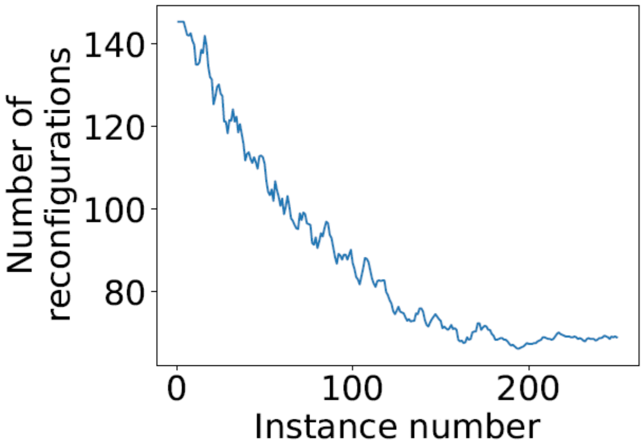


Results

- Learning curves

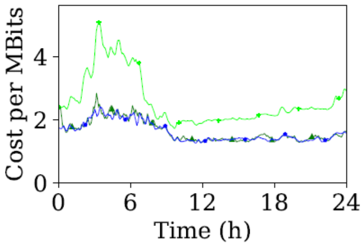


(a) Accumulated rewards

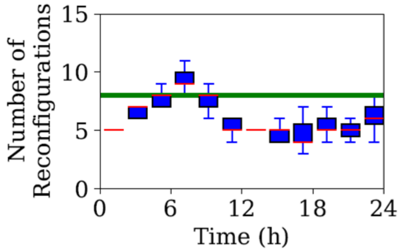


(b) Number of reconfigurations

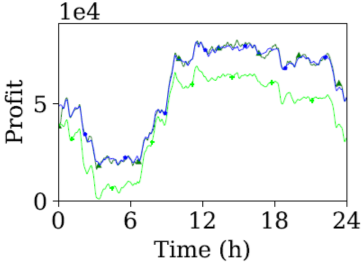
Results



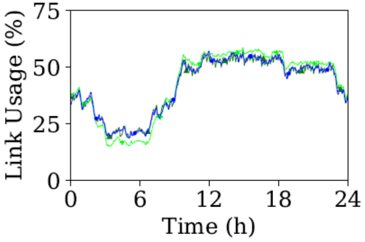
Cost per MB



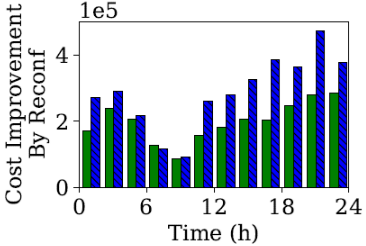
Reconfiguration distribution



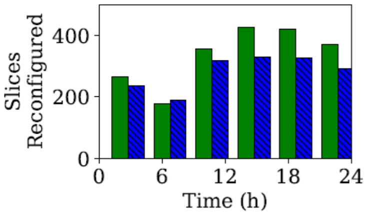
Profit



: Percentage of links capacity used



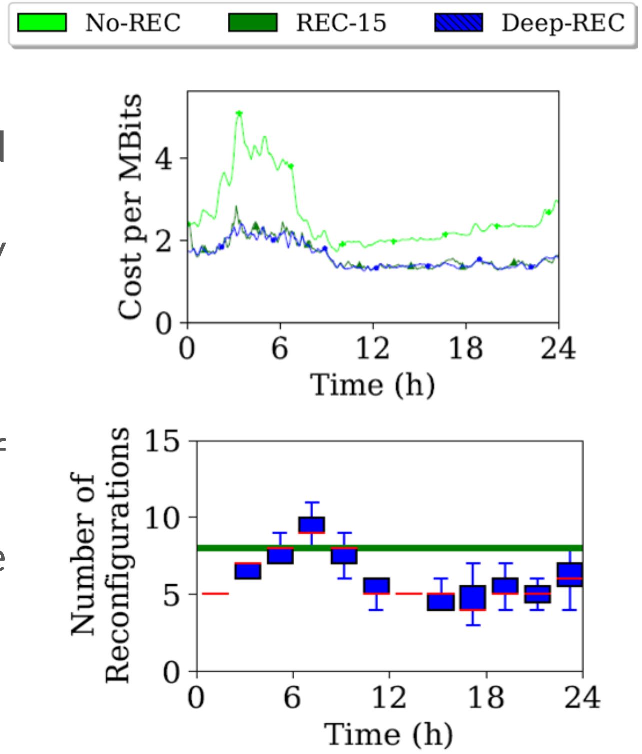
Cost gain per Reconf



Number of slices modified

Results

- Compared to *No-REC*, *REC-15* and *Deep-REC* improved the network cost and profit by 32% to 38% especially during the congestion period.
- *Deep-REC* significantly reduces the number of reconfiguration compared to *REC-15* by 20% while keeping the same cost.



Conclusion

- SDN allows better management of the network.
- Deep Reinforcement Learning is a promising solution to make network devices smart and reactive to user's demand.
- Presentation of two use cases :
  - Distributed packet routing case
    - We proposed a realistic network simulation to test MADRL approach
    - We proposed a realistic network simulation to test MADRL approach
  - Network slices reconfiguration case
    - We proposed *Deep-REC*, a DRL approach to find the best time to do reconfiguratio

Any questions ?