ITU KALEIDOSCOPE
SANTA FE 2018

Machine learning for a 5G future

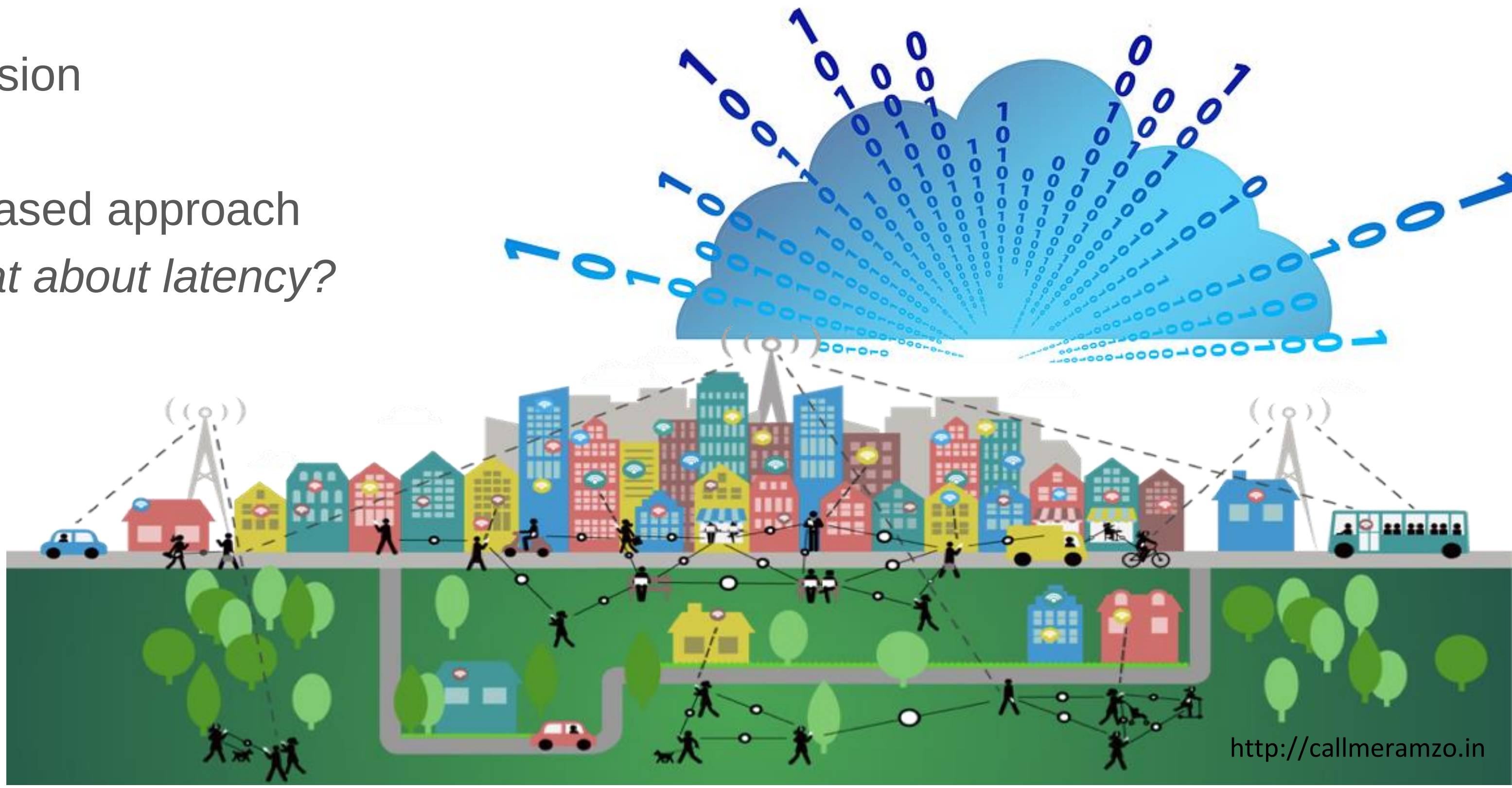
A Deep Reinforcement Learning Approach for Data Migration in Multi-access Edge Computing

Dario Bruneo
University of Messina, Italy
dbruneo@unime.it

26-28 November
Santa Fe, Argentina

Smart services in Smart environments

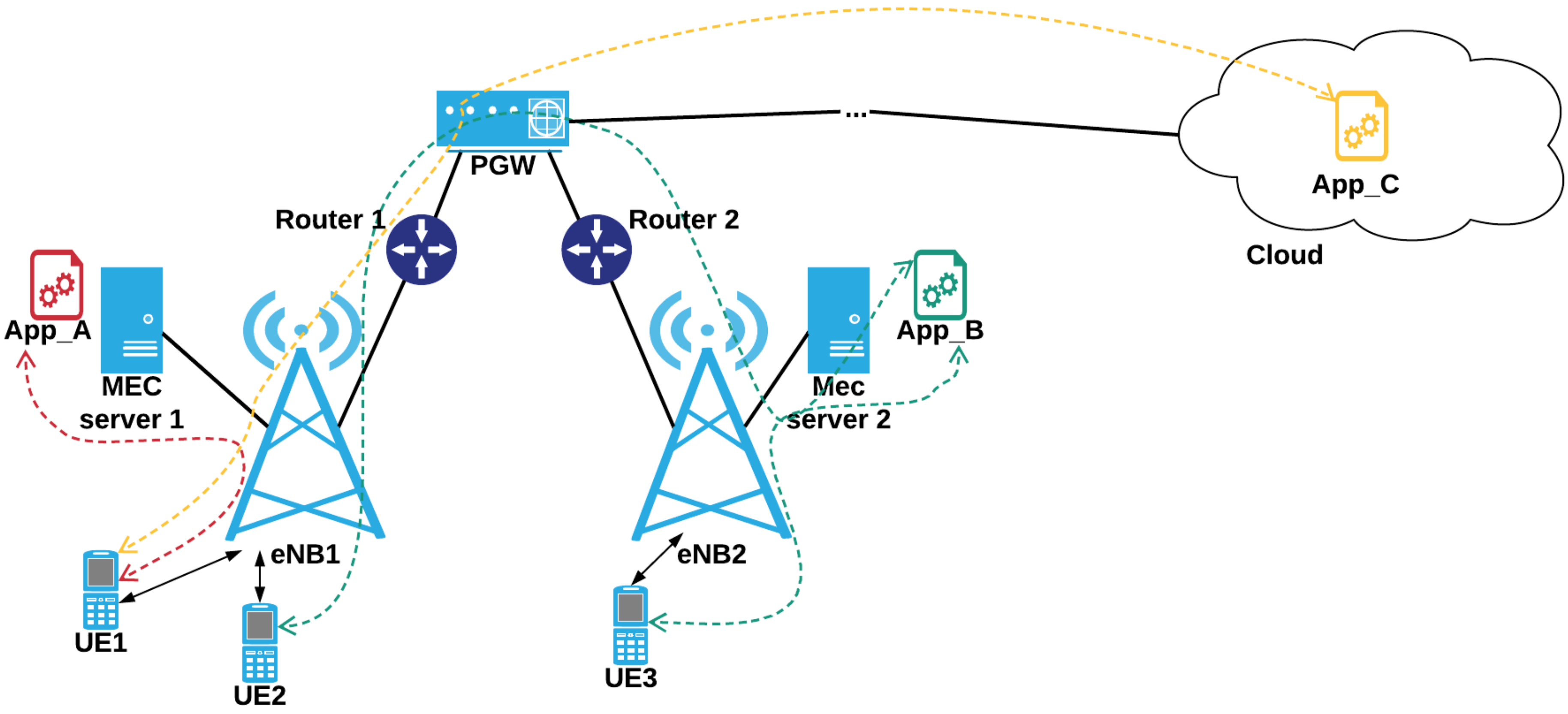
- IoT diffusion
- Cloud-based approach
  - *What about latency?*



Multi-access Edge Computing (MEC)

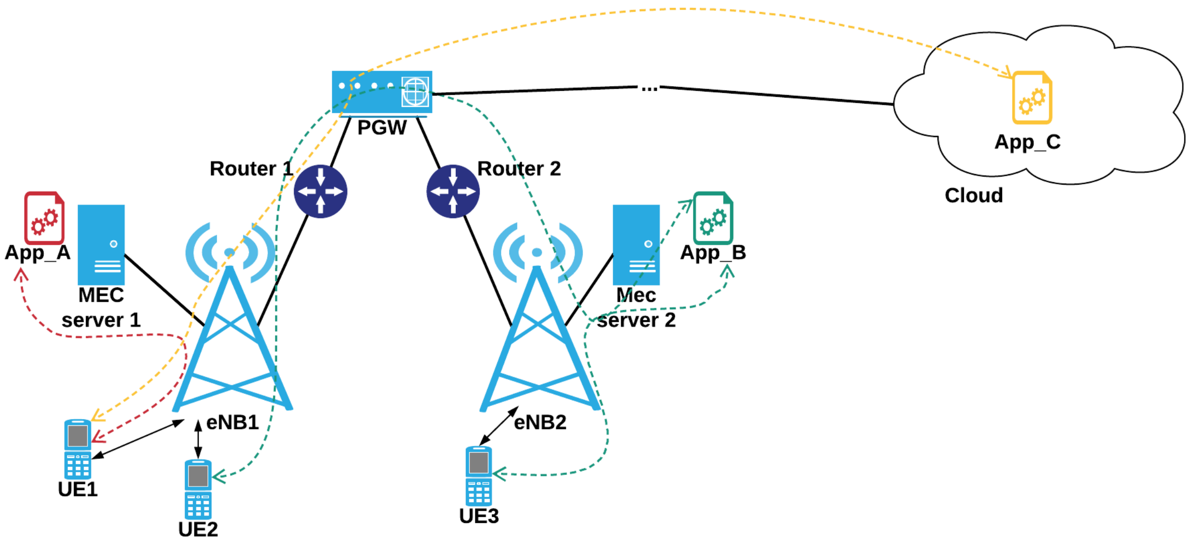
- ETSI standard
- → placing nodes with computation capabilities, *MEC servers*, **close** to the elements of the network edge
- MEC Vs. Fog Computing
  - explicit interaction with network elements
  - (network) information gathering

A 5G MEC-enabled LTE scenario



Challenges

- Resource allocation
- Application migration (*App Containerization*)
- Proactive Vs. Reactive approaches

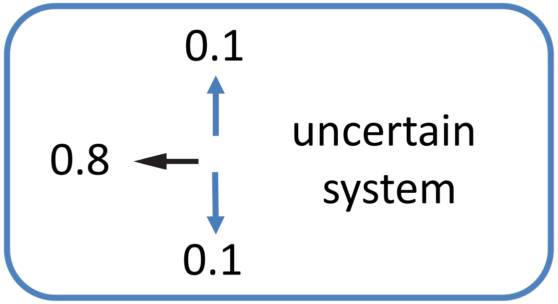


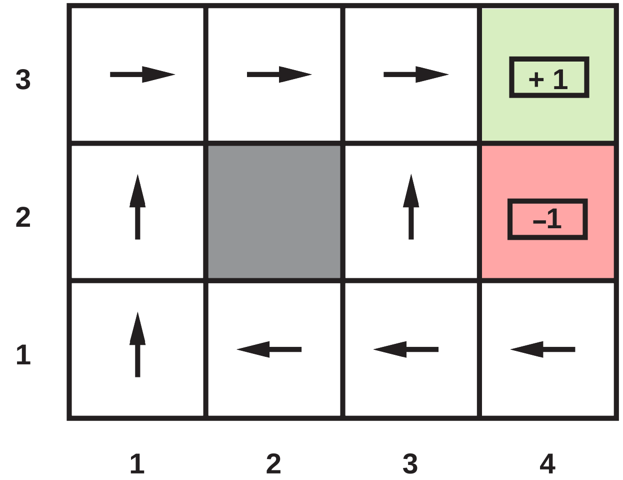
AI-based techniques → Machine Learning

Reinforcement Learning (RL)

- Learning through a trial and error process
- Best choice to solve decision making problems
- Markov Decision Process (MDP) formalism
- Q-Learning (model free approach)

$$Q(s,a) = Q(s,a) + \alpha(R(s + \gamma max_{a'}(Q(s',a') - Q(s,a))$$

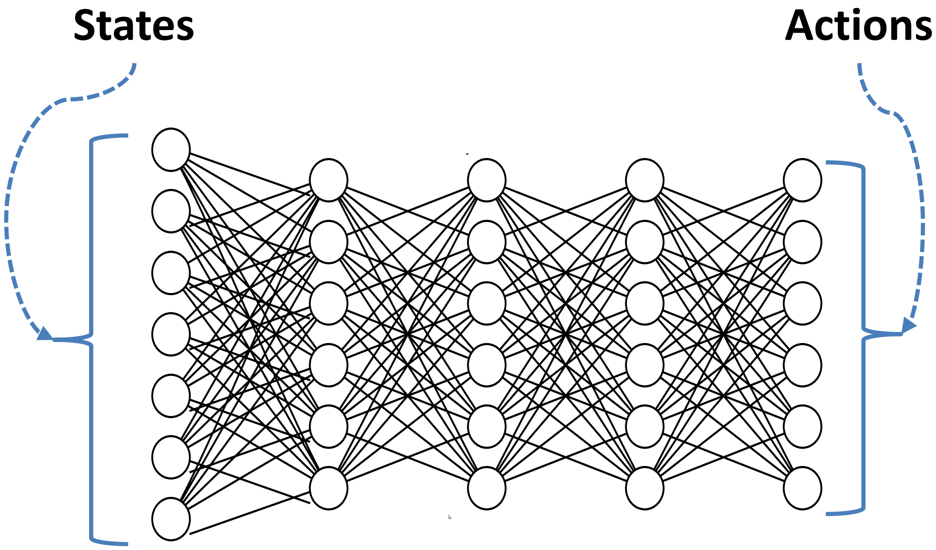




Russel-Norvig Arificial Intelligence: A modern approach – Prentice Hall

Deep RL

- Q-learning does not converge when the number of states is too large (e.g., 10^{20})
- Deep RL introduced by DeepMind
- Using a Deep Neural Network (DNN) to predict the Q-values for a given state



Applying Deep RL to MEC LTE scenarios

State → user position and app distribution

$$
\begin{aligned}
s =< (&UE_{eNB1}, UE_{eNB2}, UE_{eNB3},\\
&eNB^1_{app1}, eNB^1_{app2}, eNB^1_{app3},\\
&eNB^2_{app1}, eNB^2_{app2}, eNB^2_{app3},\\
&eNB^3_{app1}, eNB^3_{app2}, eNB^3_{app3},\\
&Mec^1_{app1}, Mec^1_{app2}, Mec^1_{app3},\\
&Mec^2_{app1}, Mec^2_{app2}, Mec^2_{app3},\\
&Mec^3_{app1}, Mec^3_{app2}, Mec^3_{app3}) >
\end{aligned}
$$

Actions → app migration

$$Actions = [a_1, a_2, a_3, \ldots, a_Z]$$

$$Z = \|N\| \cdot \|M\|$$

N → set of MEC/Cloud servers

M → set of Applications

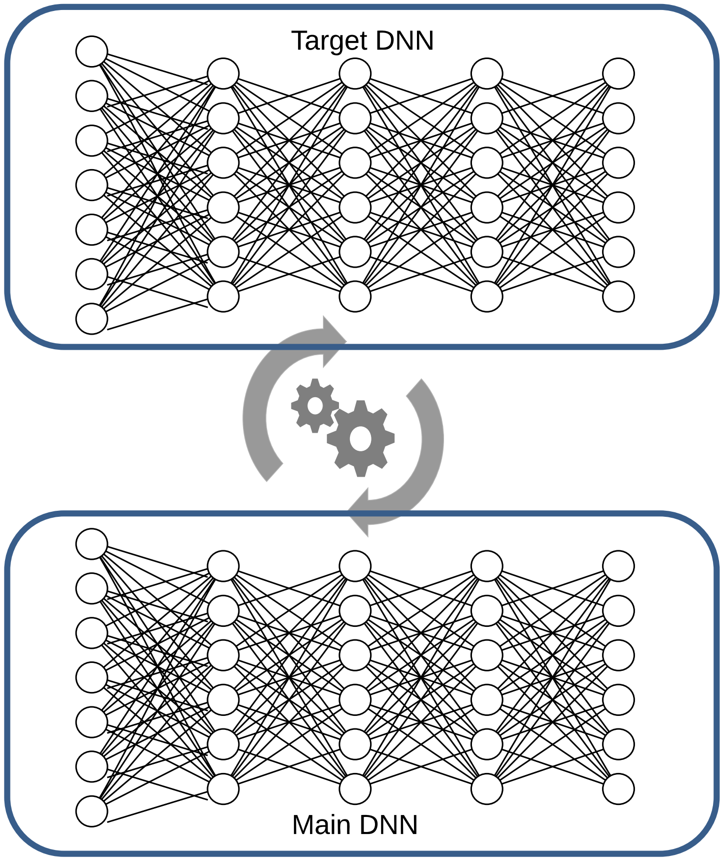
Reward → combination of network performance indexes

$$D^{app_i} = \sum \frac{Received_{THR}}{Sent_{THR} \cdot packetSize}$$

↓

Percentage of received data

The proposed algorithm



Algorithm 1: *Deep RL*

Init

initialize experience replay memory E to {}
random initialize main DNN network weights θ
set target DNN network weights $\widehat{\theta}$ equal to θ
set discount factor γ
set batch size
set update step U
set waiting time t
set exploration rate ϵ
set decay rate d

action selection

for episode = 1 to *end*:
  observe current state s_j
  $p = random([0,1])$
  if $\epsilon > p$:
    action = $random([1,\mathcal{Z}])$
  else:
    action = $argmax(Q(s_j,\theta))$
  end if

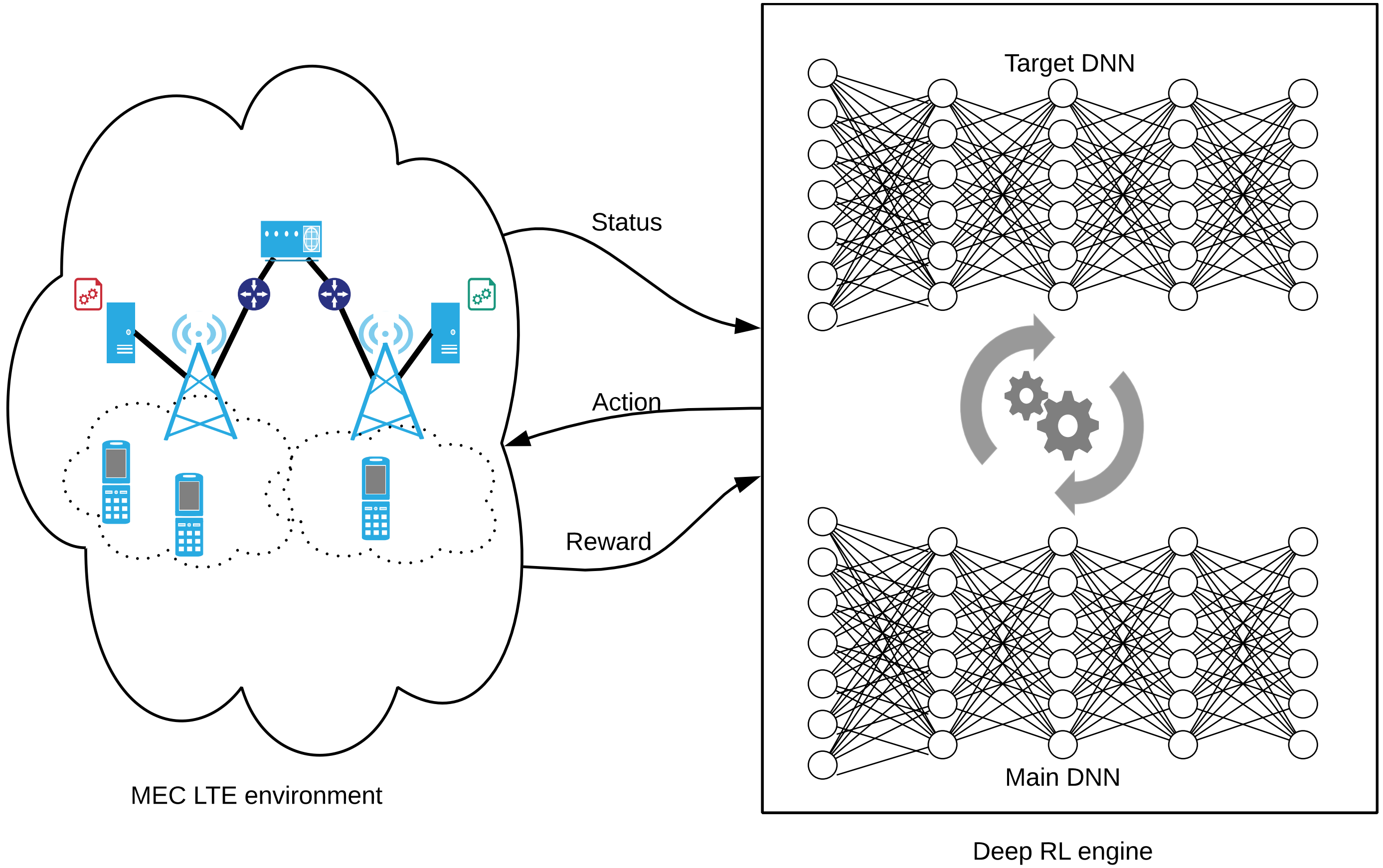
action execution

  execute the action
  wait(x seconds)
  observe the new state s_{j+1}
  observe the reward r

DNN training

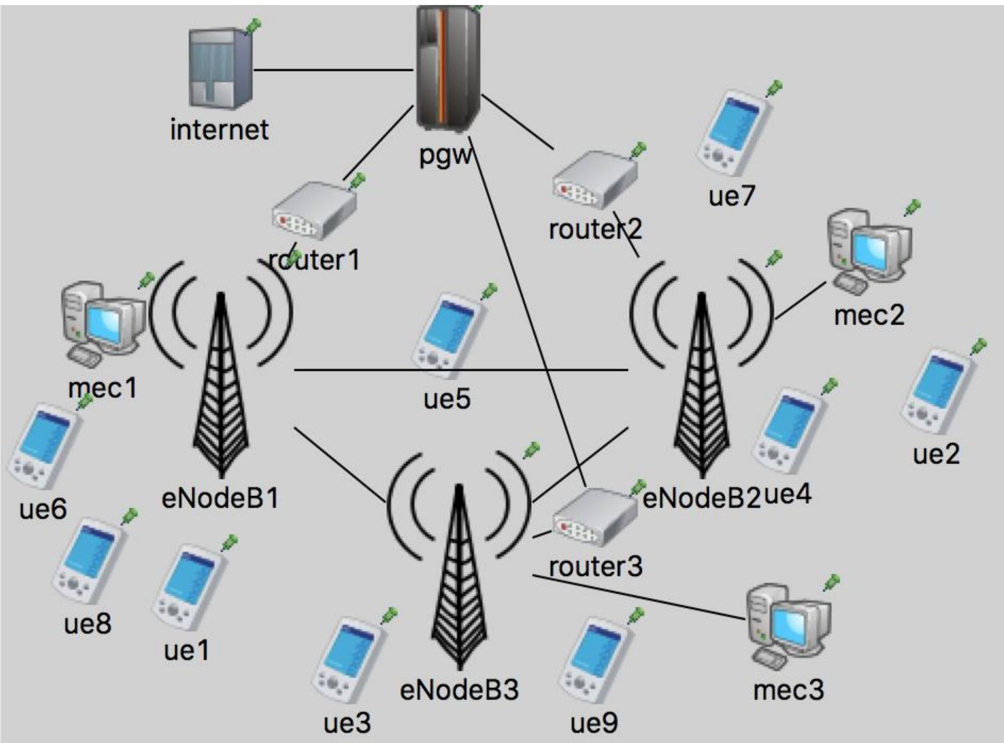
  store the t-uple $(s_j,action,s_{j+1},r)$ in E
  sample a *batch* from E
  $y = Q(s_j,\theta)$
  $y_{target} = \widehat{Q}(s_{j+1},\widehat{\theta})$
  $y_{action} = r + \gamma \cdot max(y_{target})$
  execute one training step on main DNN network
  every U steps set $\widehat{\theta} = \theta$
end for

Creating a Deep RL environment for MEC



MEC-LTE environment

SIMULte



- OMNeT++
  - iNet
  - **SimuLTE** → MEC extension

Configuration Parameters	
Number of users	9
User mobility	*RandomWayPointMobility*
User speed	1.5 mps
Number of applications	3
Application type	*UDP ConstantBitRate*
Packet size	1500*B*
Simulation Time	420 *seconds*

Deep RL engine

- Keras on top of TensorFlow
  - build complex neural network topologies with just a few lines of code
  - keeping the power of the neural network engine that runs underneath

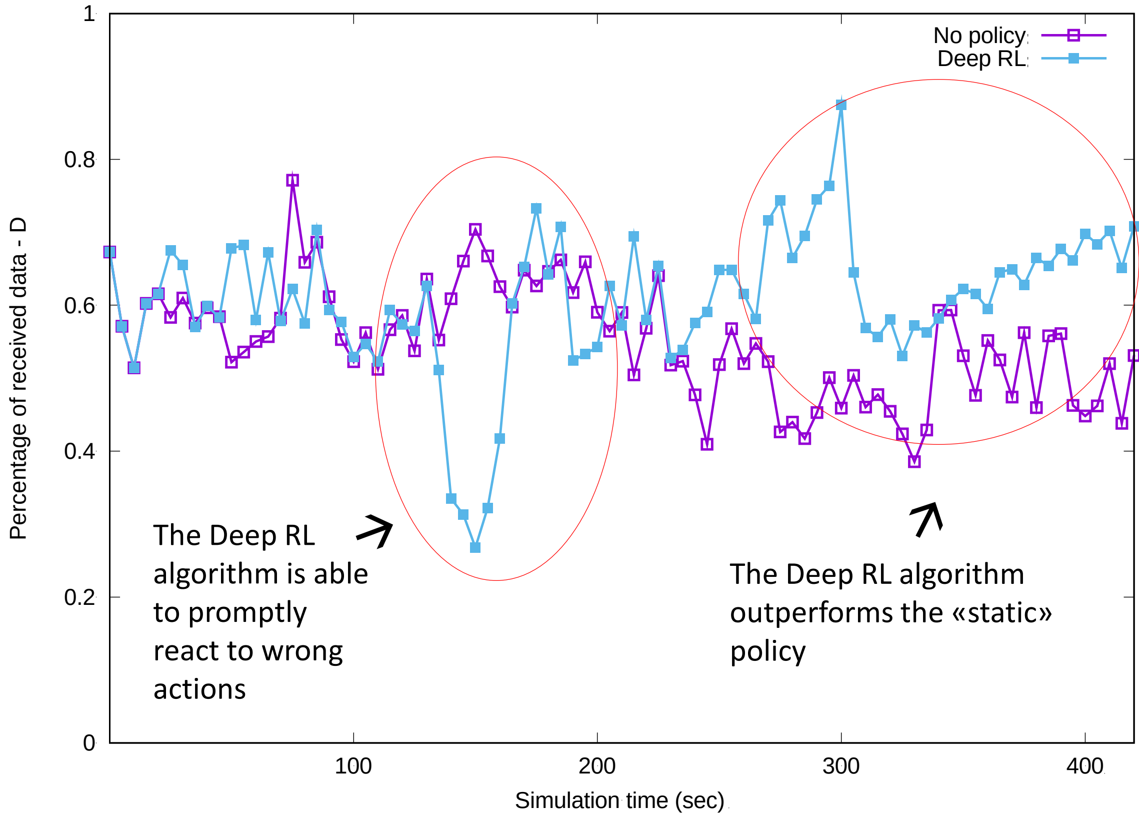
OMNeT++ (C++) and Keras (Python) have been integrated by implementing a mechanism to let them communicate using text files

DNN parameters	
Number of hidden layers	3
Number of neurons	15
Input dimension	21
Output dimension	9
Learning rate	0.001
Activation function	*ReLU*
Update step	50
Batch size	32
Experience replay dimension	2000

Experimental results

- 3 eNBs
- 9 UE
- 3 MEC Servers
- 3 Applications (CBR)
- Random walk (walking speed)
- Training for 25,000 simulation seconds

Comparison with a «static» policy where no App migration is performed



Conclusions and Future Work

- We presented a machine learning approach to address the problem related to the network environment dynamics in a 5G MEC-enabled LTE scenario
- We designed a Deep RL algorithm and tested it in a real scenario demonstrating the feasibility of the technique
- Future works will be devoted to:
  - better integration between OMNeT++/SimuLTE and Keras/TensorFlow
  - analysis of more complex scenarios
  - comparison with other solutions

ITU KALEIDOSCOPE
SANTA FE 2018

Thank you