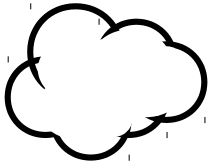
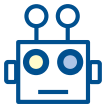
Machine Learning for Networking

# Reinforcement Learning

## Session 6 – Multi-armed Bandits II

Boris Bellalta: boris.bellalta@upf.edu

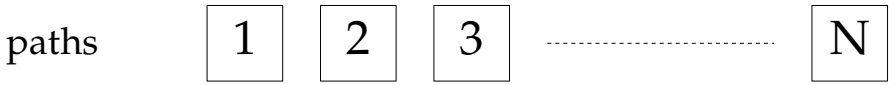


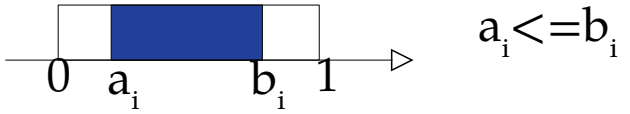


# Contents

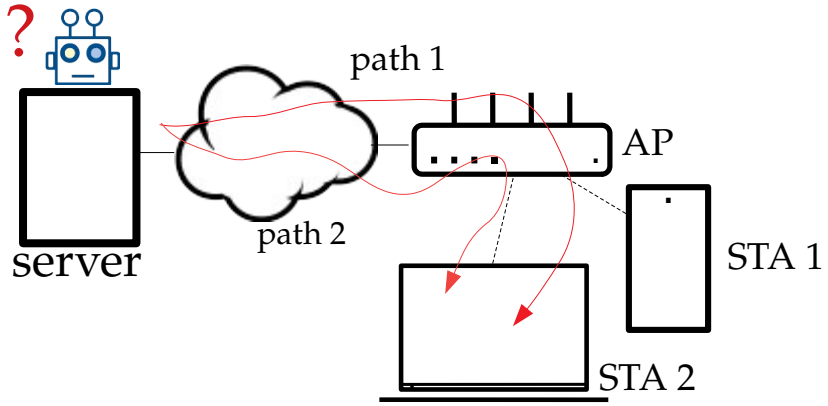
- The adaptive routing problem in multi-path transport protocols
- Non-stationary environments
- Multi-armed Bandits (next session)
  - Exp3 → MAB for non-stationary cases

# Multi-path Transport Protocols




$\rho(i)$ = occupancy path i [0,1]




**Problem**:
- Find the best strategy to send the data between paths.
- Data: a flow of B Mbps
- Strategy: $B_1$, $B_2$ ,... to be send by each path

The server, from the transport-layer ACKs, is able to know the 'RTT' of each path.
Important: Only if we send data we obtain the information

# Multi-path protocols

*This can be a topic for another TFG!*

- On-going efforts by Internet community
  - Multi-path TCP, Multi-path QUIC
- Guillem's TFG, 2017 [1]

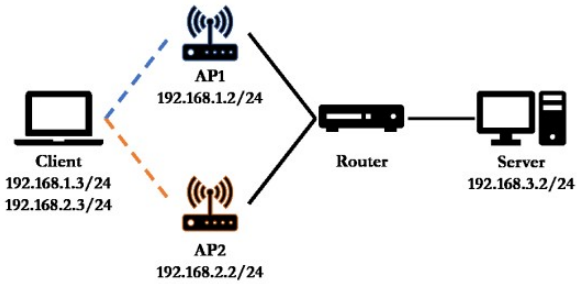



**Figure 1. Testbed**

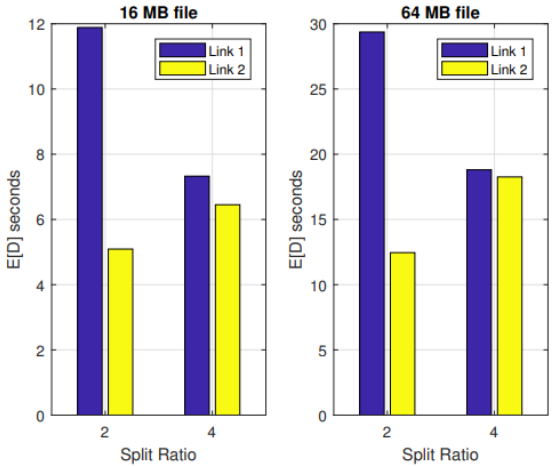
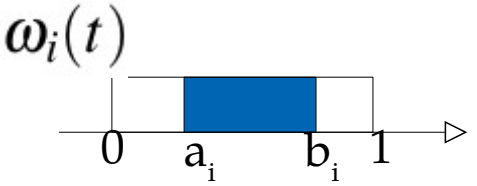



**Figure 5. File Transfer Time for different file sizes and background traffic loads (Third Test)**

[1] Cañizares, Guillem, and Boris Bellalta. "Improving User's Experience through Simultaneous Multi-WLAN Connections." In Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks, pp. 245-249. 2018.

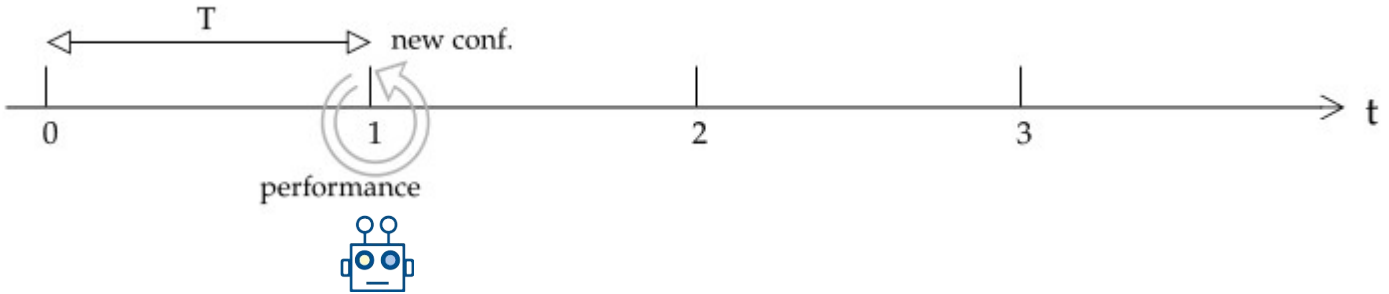
# MABs to learn the best sending strategy

- Let us consider 2 paths
- The RTT of each path is modeled as a M/M/1/K queue



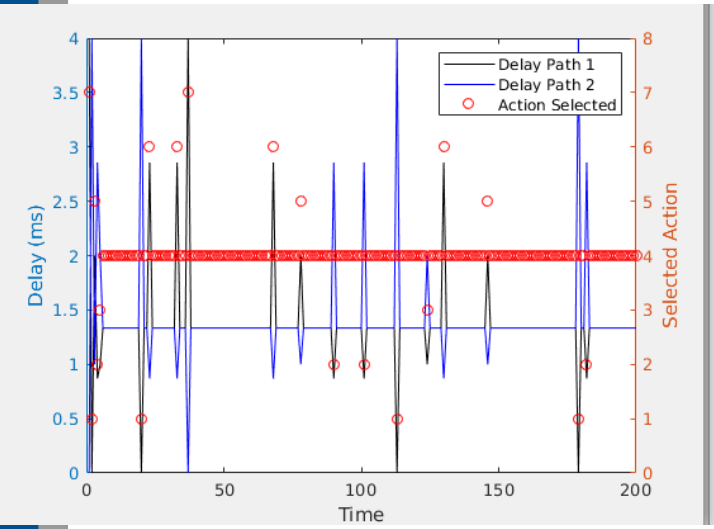

```
ES = L/C;
lambda=(B/L);
a = lambda*ES+w;

if a==1
    EN = K/2;
    Pb = 1/(K+1);
else
    Pb = (1-a)*a^K/(1-a^(K+1));
    EN = a/(1-a) - ((K+1)*a^(K+1))/(1-a^(K+1));
end
ED = EN/(lambda*(1-Pb));
ELosses = B*Pb;
```

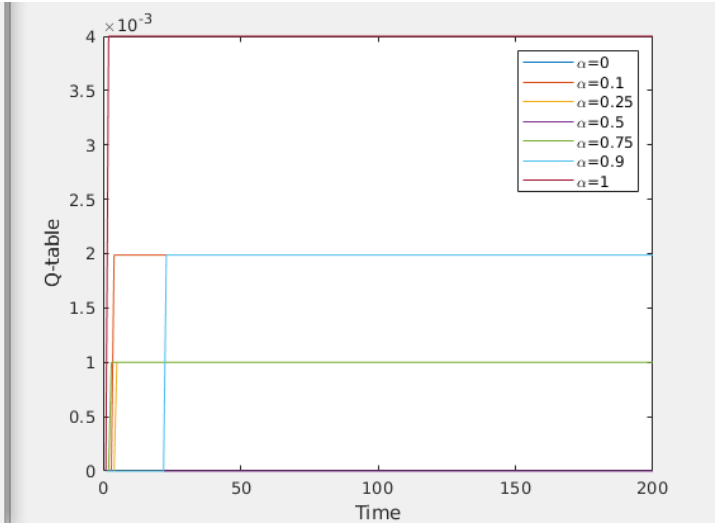
- $E[S_i]=E[L]/C_i$
- $\lambda_i=\alpha_i B/E[L]$
- $\alpha_1+\alpha_2=1$

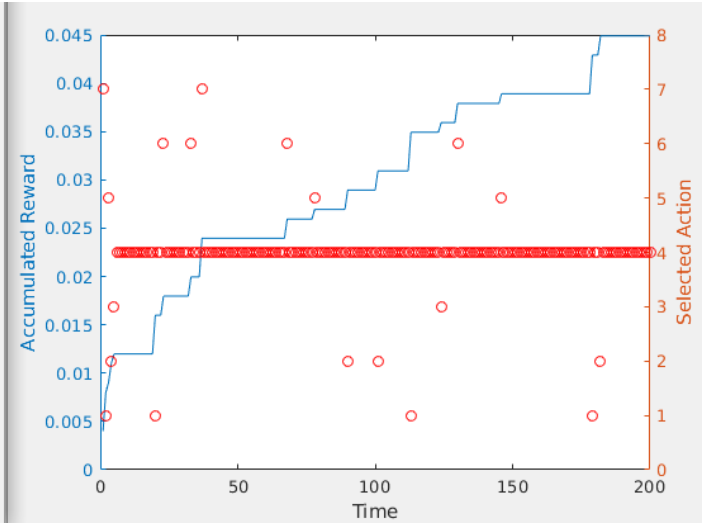


# Example 1 (Epsilon-greedy)

- C1=C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- Deterministic channels: a1=b1=a2=b2=0; | ε=0.1; Delta=0;



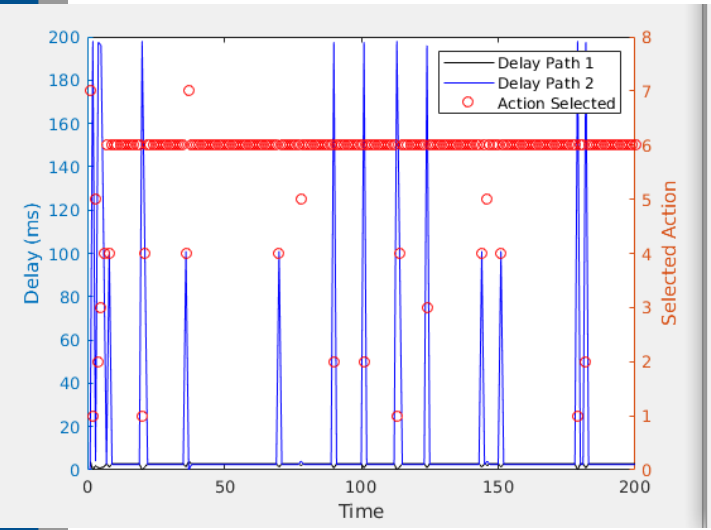



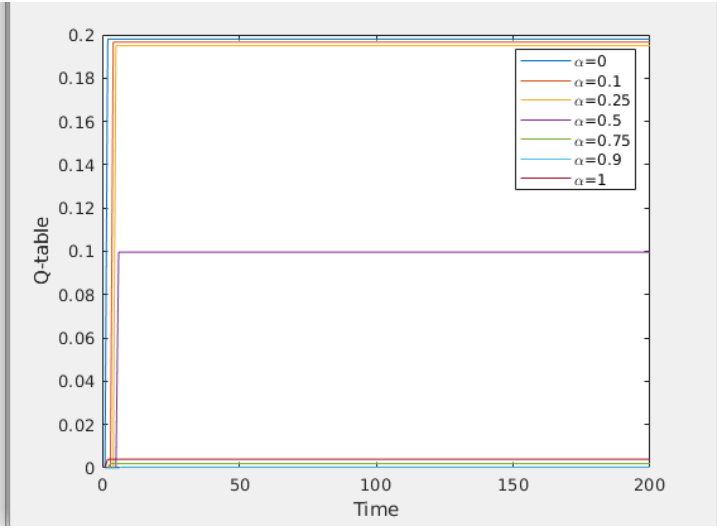



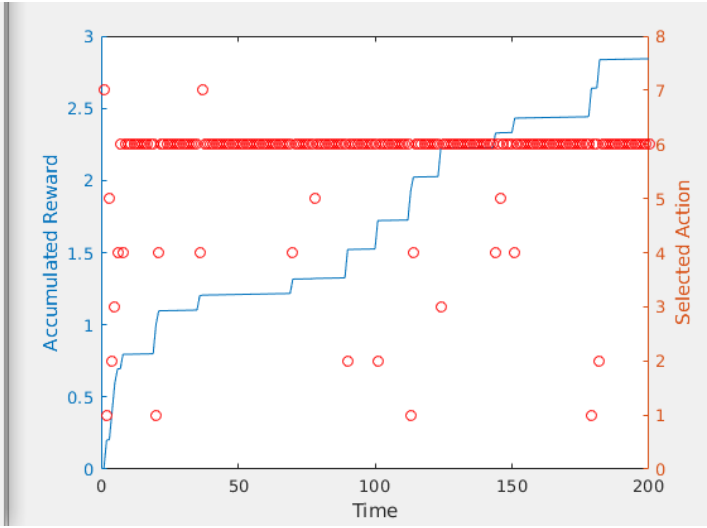

```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    1.3962e-003     9.7782e-006
```

# Example 2 (Epsilon-greedy)

- C1=10 Mbps; C2=4 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- Deterministic channels: a1=b1=a2=b2=0; | ε=0.1; Delta=0;

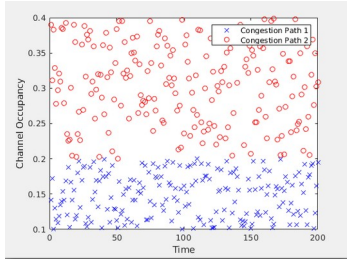


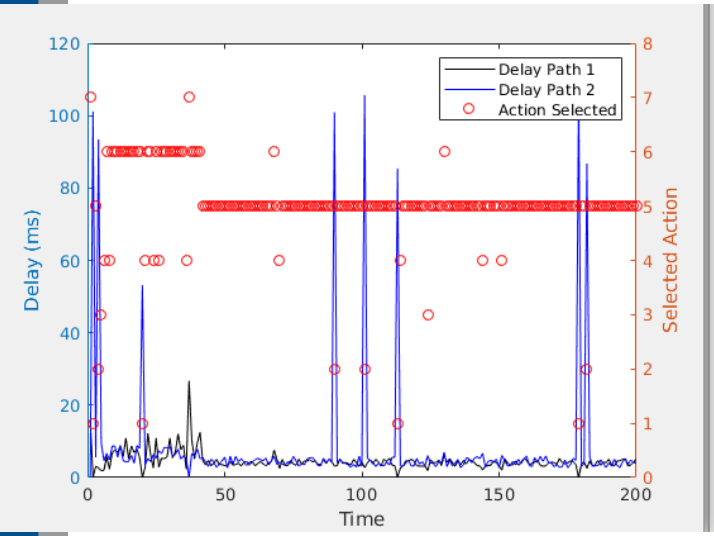



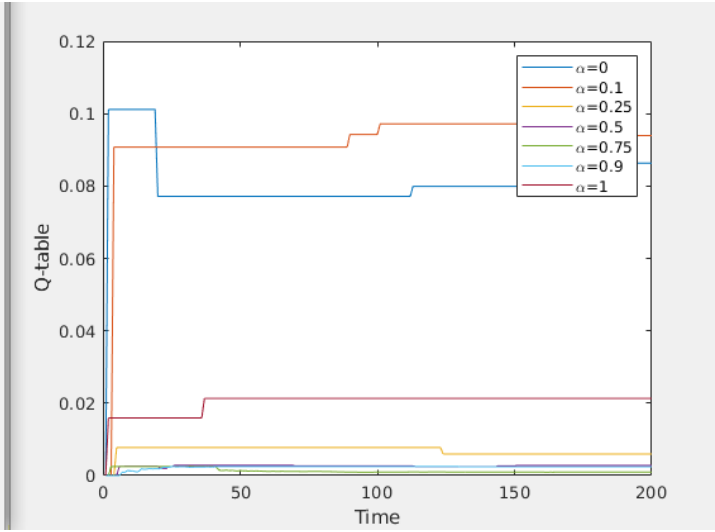

```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    16.1825e-003   165.5842e+003
```

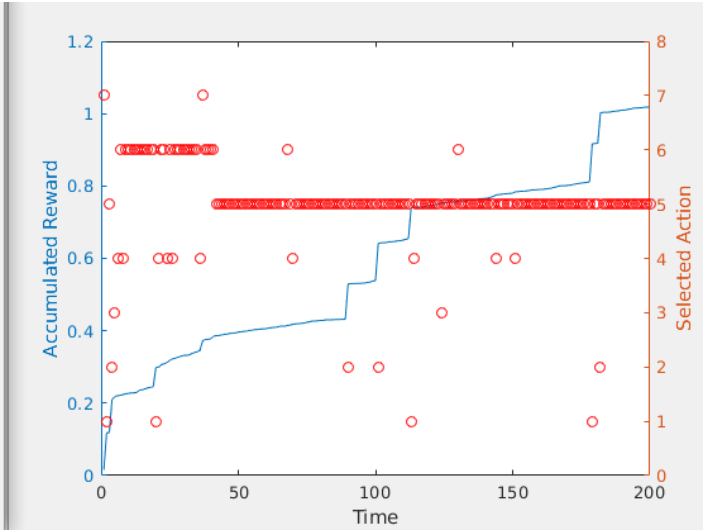
# Example 3 (Epsilon-greedy)




- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- Deterministic channels: a1=0, b1=0.2, a2=0.2, b2=0.4; | ε=0.1; Delta=0;



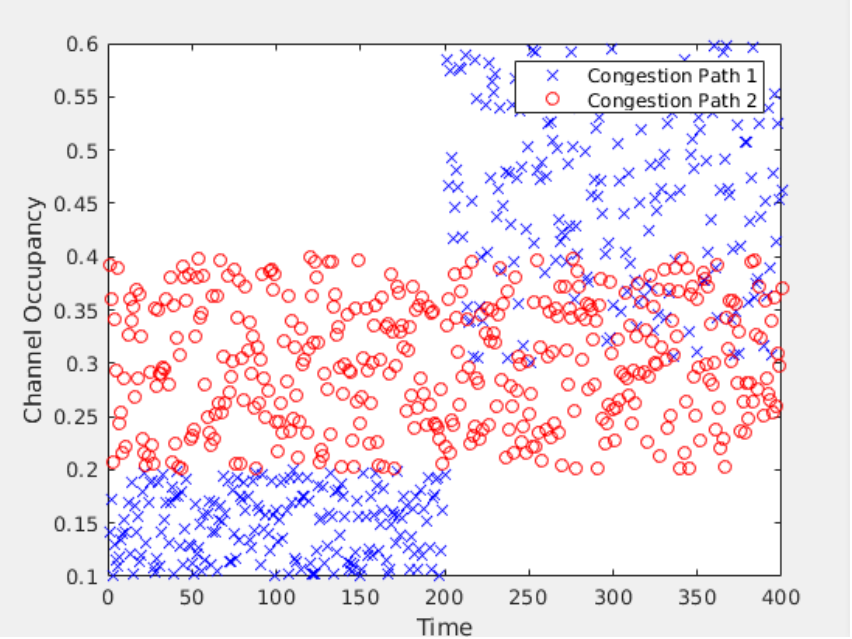




```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
     8.0897e-003    19.3161e+003
```

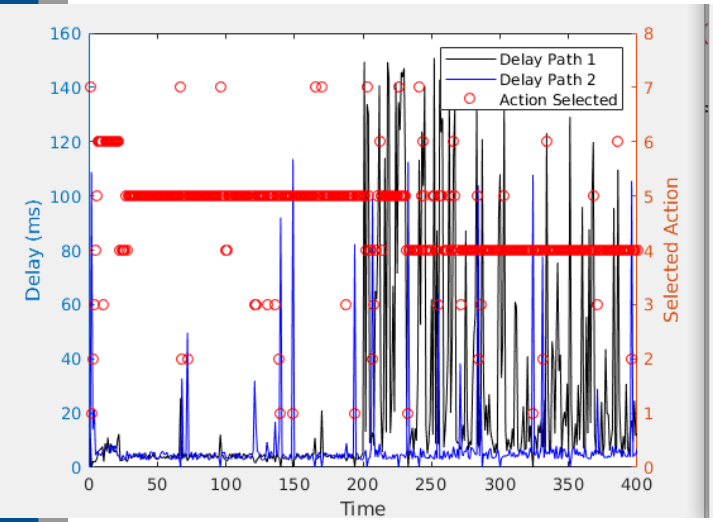
# Example 4 (Epsilon-greedy + channel changes T=200)

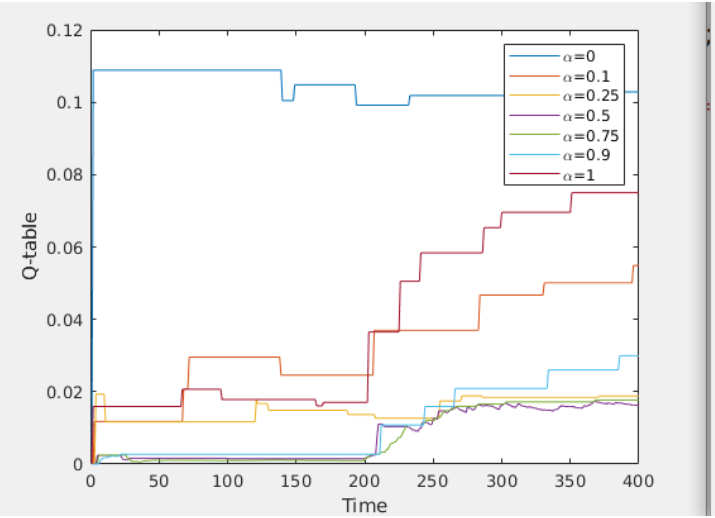
- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.2, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.2, a2=0.3, b2=0.6; | ε=0.1; Delta=0;

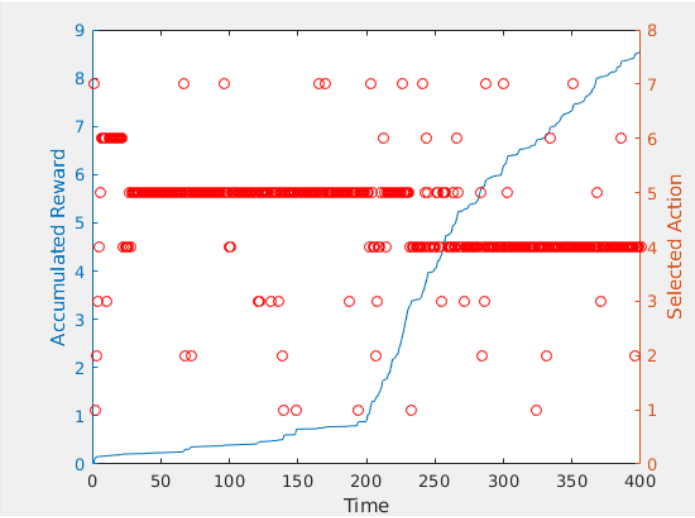


# Example 5 (Epsilon-greedy + channel changes T=200)

- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.2, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.2, a2=0.3, b2=0.6; | ε=0.1; Delta=0;



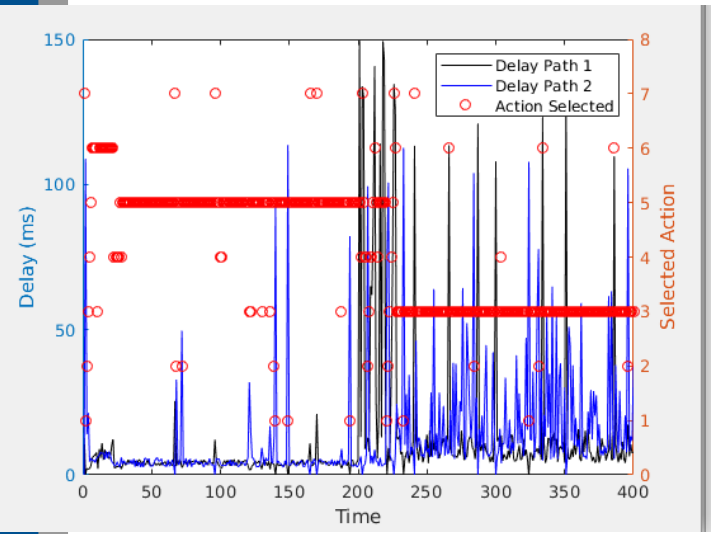



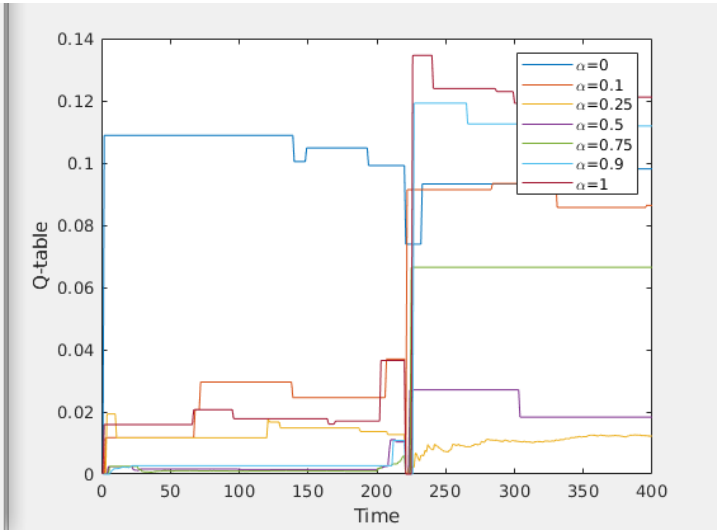



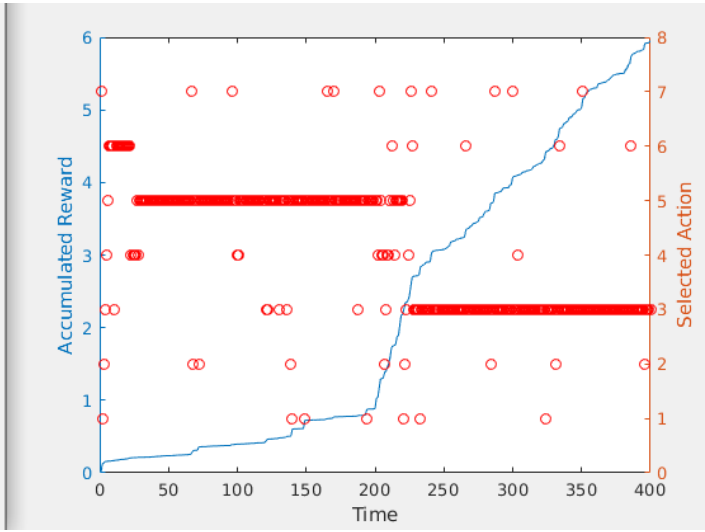

```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    21.9910e-003    92.3643e+003
```

# Example 6 (Epsilon-greedy + channel changes + RESET)

- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.2, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.2, a2=0.3, b2=0.6; | ε=0.1; Delta=0;



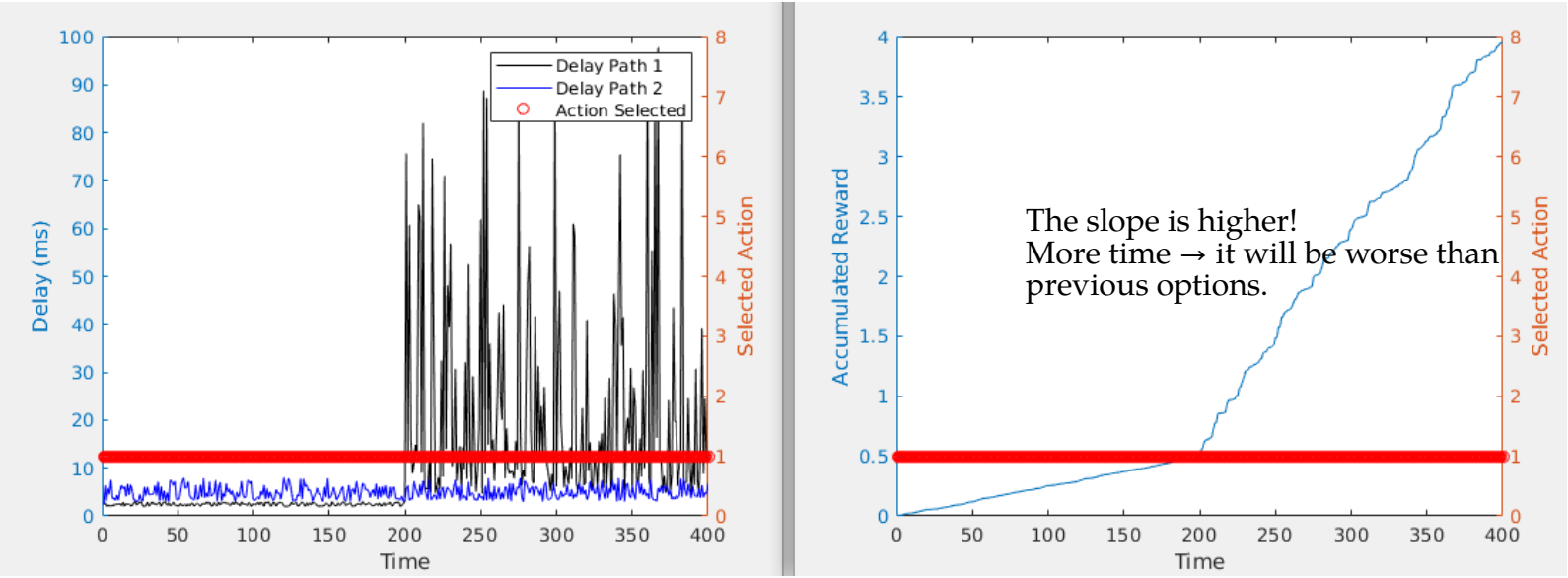




```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    13.3379e-003    70.5026e+003
```

# Example 7 (Static + channel changes)

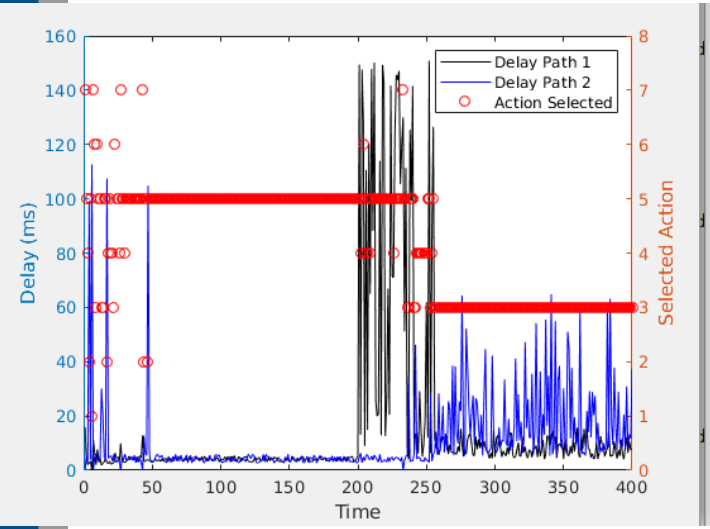
- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0.5]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.2, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.2, a2=0.3, b2=0.6; | ε=0.1; Delta=0;

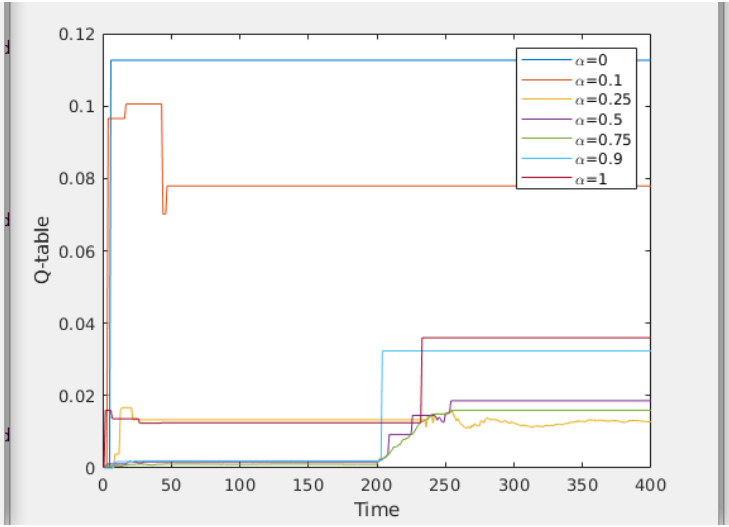


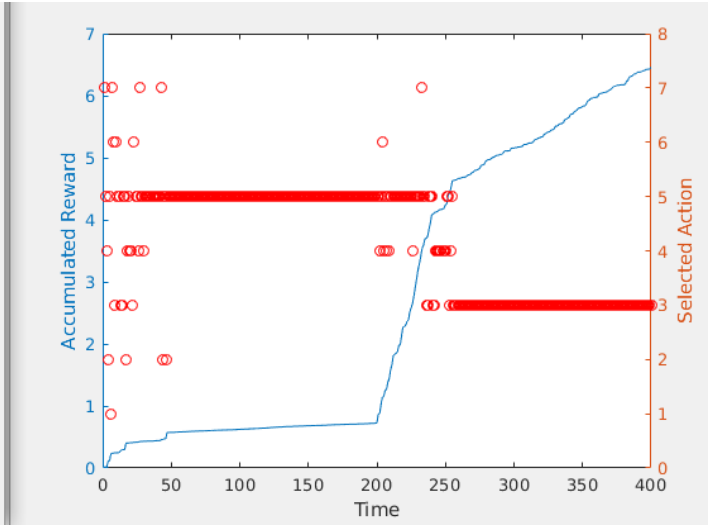

```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    12.3151e-003     1.1211e+003
```

# Example 8 (… + Decreasing Epsilon + No RESET)

- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.1, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.1, a2=0.3, b2=0.6; | ε=1; Delta=0.02;



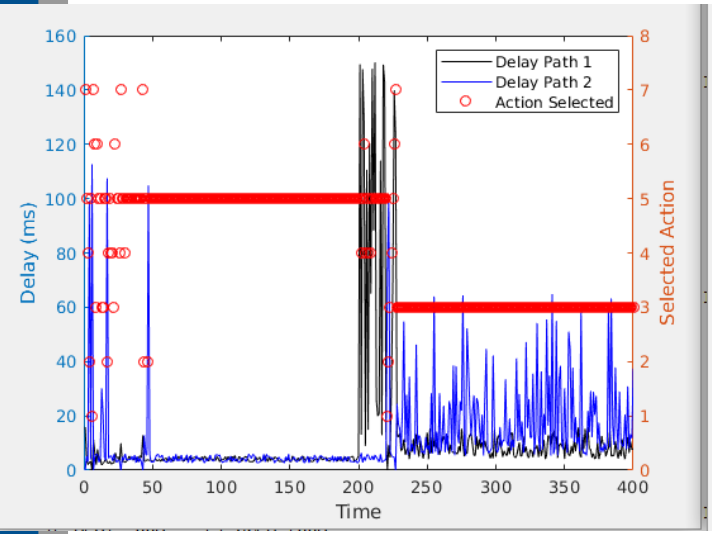



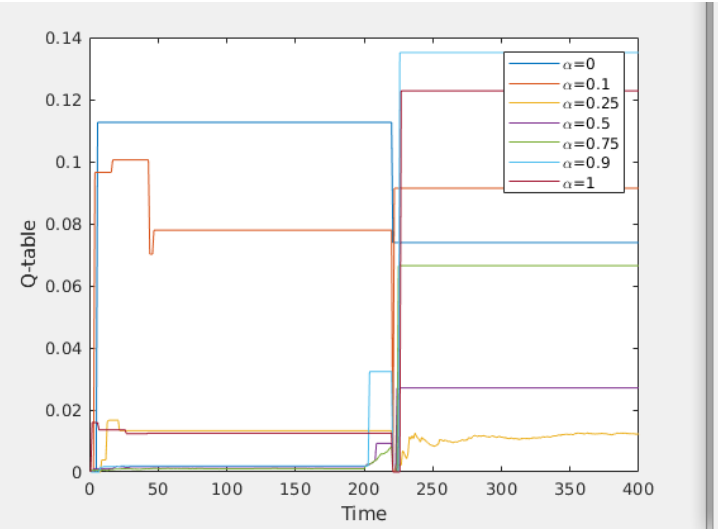



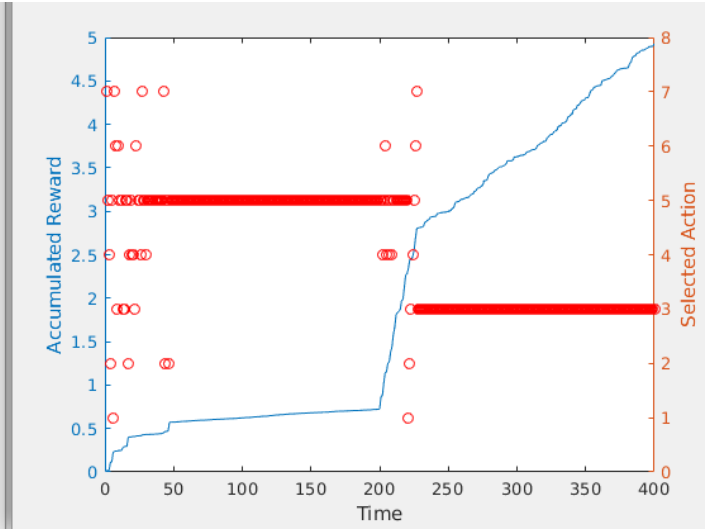

```
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    15.2581e-003    51.9652e+003
```

# Example 9 (… + Decreasing Epsilon + RESET)

- C1=10 Mbps; C2=10 Mbps; B=8 Mbps; | alfa1=[0 0.1 0.25 0.5 0.75 0.9 1]]; alfa2 = 1-alfa1
- T=[0,200]: a1=0, b1=0.2, a2=0.2, b2=0.4; | T=[200,400]: a1=0, b1=0.2, a2=0.3, b2=0.6; | ε=1; Delta=0.02;




```
Final Performance
Max delay (av) between path 1 and path 2 | Total Losses (sum of path 1 and path 2)
    12.0901e-003    38.6449e+003
```

# The code (Epsilon Greedy)

```
function ExampleLecture7(seed)

rng(seed);

C1 = 10E6;
C2 = 10E6;
L=8000;
KS=100;
B=8E6;

% Sim parameters
TimeHorizon=200; % Increase to 400 for the non-stationary case
TimeChange=200;

N=2; % number of paths

a1=zeros(1,TimeHorizon);
a1(1:TimeChange)=0.1.*ones(1,TimeChange);
a1(TimeChange+1:TimeHorizon)=0.3.*ones(1,TimeHorizon-TimeChange);

b1=zeros(1,TimeHorizon);
b1(1:TimeChange)=0.2.*ones(1,TimeChange);
b1(TimeChange+1:TimeHorizon)=0.6.*ones(1,TimeHorizon-TimeChange);

a2=0.2;
b2=0.4;

w(1,:)=a1 + (b1-a1).*rand(1,TimeHorizon);
w(2,:)=a2 + (b2-a2).*rand(1,TimeHorizon);

% Weights - Actions
%alfa = [0:0.05:1];
%alfa = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
alfa = [0 0.1 0.25 0.5 0.75 0.9 1];
%alfa = [0.5];


K = length(alfa);
```

```
% Gamma
gamma = 0.8;

%----------------- Epsilon greedy -----------------

epsilon = 0.1;
Delta = 0.0;

%epsilon = 1;
%Delta = 0.01;

Q_EG_disp = zeros(K,TimeHorizon);
Q_EG = zeros(K,1);

R_EG = zeros(1,TimeHorizon);
alfa_EG = zeros(1,TimeHorizon);
alfa_selected_EG = zeros(1,K);

% First Iteration

n_sel = randi(K,1); % Action Selected, random
alfa_EG(1) = n_sel;
alfa_selected_EG(n_sel) = 1;

% First Path
[ED1(1),ELosses1(1)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,1));
% Second Path
[ED2(1),ELosses2(1)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(1,1));

% Reward (taking only the delay into account)
R_EG(1) = alfa(n_sel)*ED1(1)+(1-alfa(n_sel))*ED2(1); % Mean
%R_EG(1) = max([ED1(1) ED2(1)]); % Max

% Update Q-table
Q_EG(n_sel)=R_EG(1);
Q_EG_Disp(:,1) = Q_EG;

alfa_selected_EG(n_sel) = 1;
```

# The code (Epsilon Greedy)

```
for t=2:TimeHorizon

    if(rand < epsilon) % explore
        n_sel=randi(K,1);
    else % exploit
        [~,n_sel]=min(Q_EG(:)); % Maximize the reward --> minimize the delay!!!
    end

    alfa_EG(t) = n_sel;
    alfa_selected_EG(n_sel)=alfa_selected_EG(n_sel)+1;

    % First Path
    [ED1(t),ELosses1(t)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,t));
    % Second Path
    [ED2(t),ELosses2(t)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(2,t));

    % ----------Rewards--------------------
    %R_EG(t) = alfa(n_sel)*ED1(t)+(1-alfa(n_sel))*ED2(t); % Average Delay
    R_EG(t) = abs(ED1(t)-ED2(t)); % Diference between delays
    %if(t<10)
    %    R_EG(t) = max([ED1(1:t) ED2(1:t)]); % Max delay
    %else
    %    R_EG(t) = max([ED1(t-9:t) ED2(t-9:t)]); % Max delay
    %end

    % --------- Q table ---------------------
    %Q_EG(n_sel)=(Q_EG(n_sel)*(alfa_selected_EG(n_sel)-1) + R_EG(t))/alfa_selected_EG(n_sel); % Average
    Q_EG(n_sel)=gamma*R_EG(t)+(1-gamma)*Q_EG(n_sel); % Gamma method

    Q_EG_disp(:,t)=Q_EG(:); % For display purposes

    % Update Learning
    epsilon = epsilon - Delta;

    % ------ Reset ---------

    %if(t==220)
    %   alfa_selected_EG = zeros(1,K);
    %   Q_EG = zeros(K,1);
    %   epsilon = 1;
    %   Delta = 0.01;
    %end

end
```

# Activity

- Download Example7.zip code
- Execute: example7(2), and try other seeds
- Play with the value of C1 and C2, and values of a1, b1, a2, b2. Are the obtained results consistent?
  - a) B=8 Mbps; C1=8 Mbps; C2=10 Mbps; a1=0.5; b1=0.5; a2=0.1; b2=0.9
  - b) B=8 Mbps; C1=8 Mbps; C2=10 Mbps; a1=0.1; b1=0.2; a2=0.1; b2=0.9
  - c) B=8 Mbps; C1=8 Mbps; C2=10 Mbps; a1=0.1; b1=0.2; a2=0.1; b2=0.6
- Change the reward from the difference, to the mean and maximum delay.
  - Discuss how sensitive is the algorithm to the reward definition. Are the obtained performance results better or worse? Try for a) and b)
- Update the Q-table following the 'gamma method'. Try for gamma=0.1, gamma=0.5 and gamma =0.8. Discus the differences.
  - Whis is the best gamma value? What does it mean?
- Play with different epsilon and Delta values.
  - Epsilon = 0.9; Delta = 1/200;

# Activity

- Homework:
  - Repeat the examples in this slides
  - Implement TS and UCB to solve this problem
    - If you do it, send me an e-mail with the results, and we can discuss about the obtained results

# Reading

- Wilhelmi, Francesc, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. "**Collaborative spatial reuse in wireless networks via selfish multi-armed bandits.**" Ad Hoc Networks 88 (2019): 129-141. https://arxiv.org/pdf/1710.11403.pdf