1:

            if max(agent[j].pckt\_ratio) >= 1:

                opp = 1

            else:

                opp = 0

            next\_state = np.concatenate([ state[j][4:], [agent\_action[j], cap, ack[j], opp] ])

env = ENVIRONMENT(state\_size=4\*2)

environment.py:

                if opponent\_id > j:

                    opponent\_id -= 1

                agent[j].pckt\_ratio[opponent\_id] = opponent\_packets/(agent[j].real\_packets + 0.001)

            elif action[j] == 1 and cap == 0 and agent[j].real\_packets == 0: ##########

                agent\_reward[j] = 1

                reward[j] = 0

                ack[j] = 1#0

reward[j] = 1#2

# =============================================================================

#       my scheme

# =============================================================================

        for j in range(n):

            maxim = max(agent[j].pckt\_ratio)

            if action[j] == 1 and maxim > 1:

                reward[j] = 0

        #reward = [r for x in range(n)]

وقتی یه ایجنت در حال ارساله و از بقیه خبر نداره و در همین حال به ما پکت اضافه میشه، اون به ارسال خودش ادامه میده، همین باعث میشه شبکه بعد از یه مدت این رو یاد بگیره که "وقتی پکت های ما تعدادش بیشتر از بقیست، اگر بفرستم تصادف میشه! و ریورد صفر میگیرم، پس بفرستم نفرستم فرقی نداره"

1(4):

if action[j] == 1 and maxim > 1:

                if cap == 0:

                    reward[j] = 0

                else:

                    reward[j] = -1

4:

NN = 5

5:

    agent.append(Agent(60, test, NN, 600))

    for j in range(NN-1):

        agent.append(Agent(60, test, NN, 300))

6:

            if max(agent[j].pckt\_ratio) >= 1:

                opp = 1

            elif max(agent[j].pckt\_ratio) == 0 and agent[j].real\_packets == 0:

                opp = -1

            else:

                opp = 0

7:

elif max(agent[j].pckt\_ratio) == 1:

                opp = 2

8:

NN = 5

9:

for j in range(n):

            maxim = max(agent[j].pckt\_ratio)

            if action[j] == 1 and maxim > 1:

                if cap == 0:

                    reward[j] = 0

                else:

                    reward[j] = -1

env = ENVIRONMENT(state\_size=4\*1)

1’:

env = ENVIRONMENT(state\_size=4\*20)

1’’:

env = ENVIRONMENT(state\_size=4\*100)

2:

            for x in range(number\_of\_DRLs-1):

                if agent[j].pckt\_ratio[x] >= 1:

                    opponent\_buffer[x] = 1

                else:

                    opponent\_buffer[x] = 0

            next\_state = np.concatenate([ state[j][5:], [agent\_action[j], cap, ack[j]], opponent\_buffer ])

env = ENVIRONMENT(state\_size=5\*2)

3:

opp = [max(x, 2) for x in agent[j].pckt\_ratio]

            next\_state = np.concatenate([ state[j][5:], [agent\_action[j], cap, ack[j]], opp ])

env = ENVIRONMENT(state\_size=5\*2)