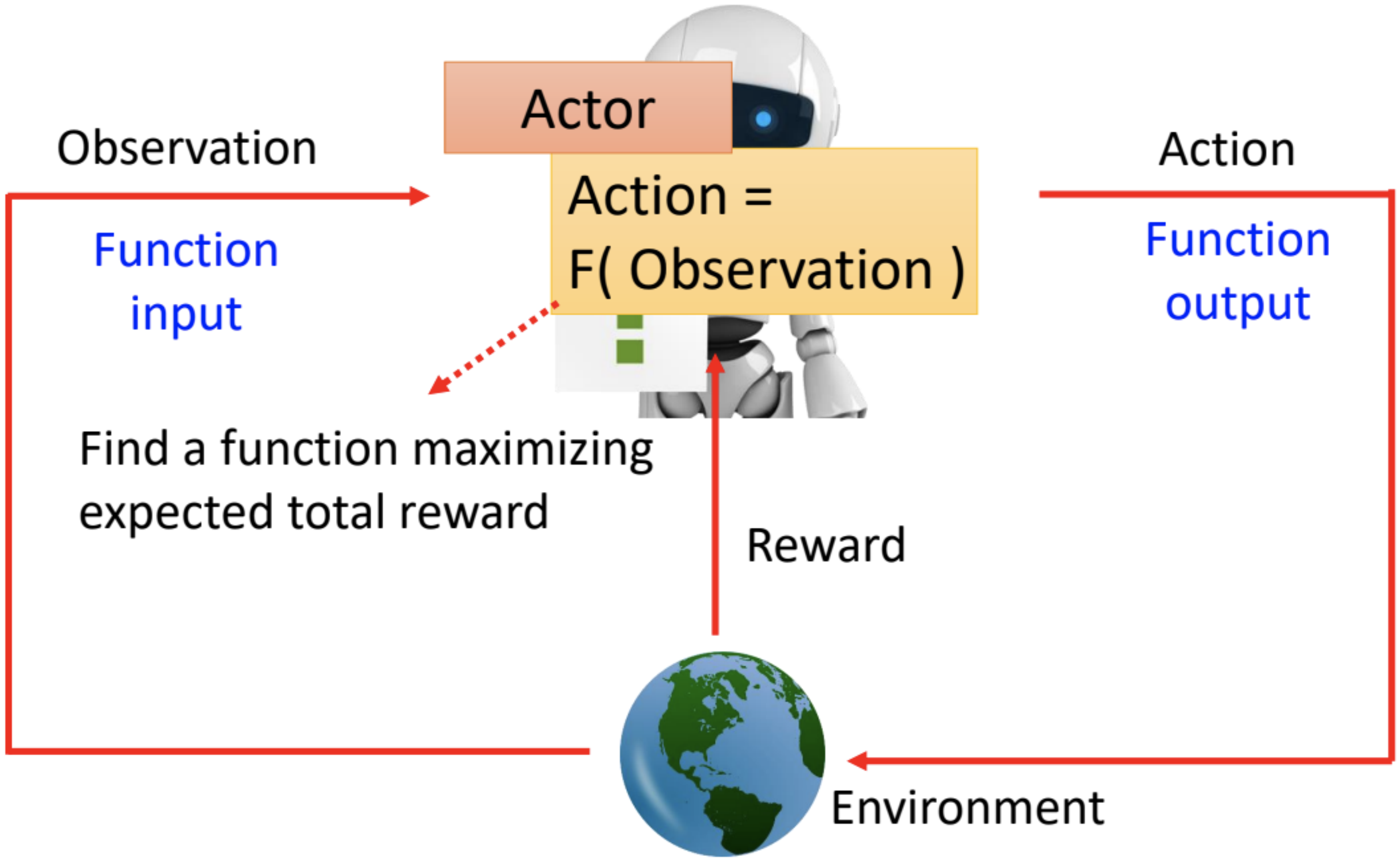
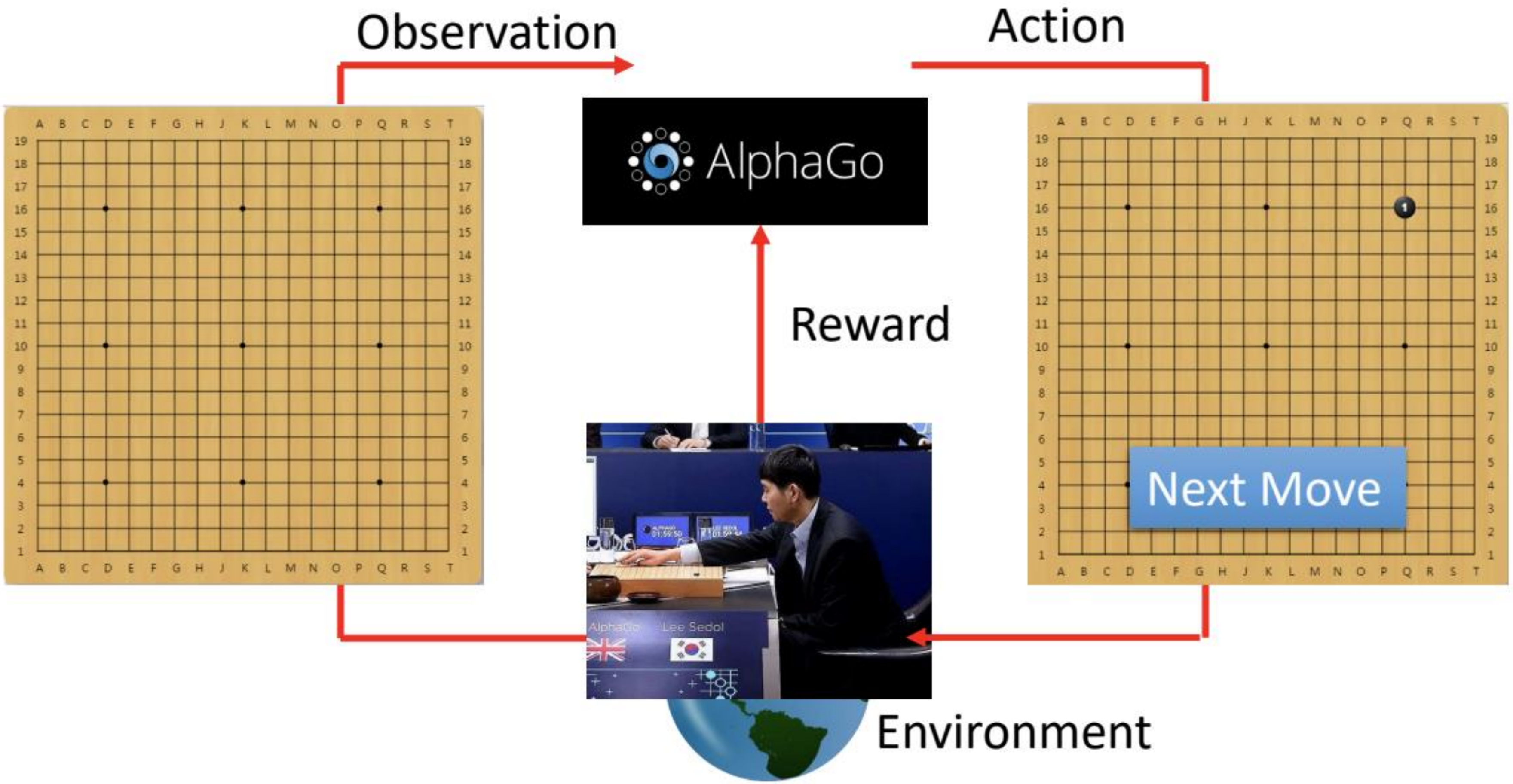
AI無線通訊系統實驗
Reinforcement Learning for Network Resource Allocation

助教: 李育亭、吳渟禎
教材編寫: 蕭安紘、沈立翔 博士

Reinforcement Learning, RL

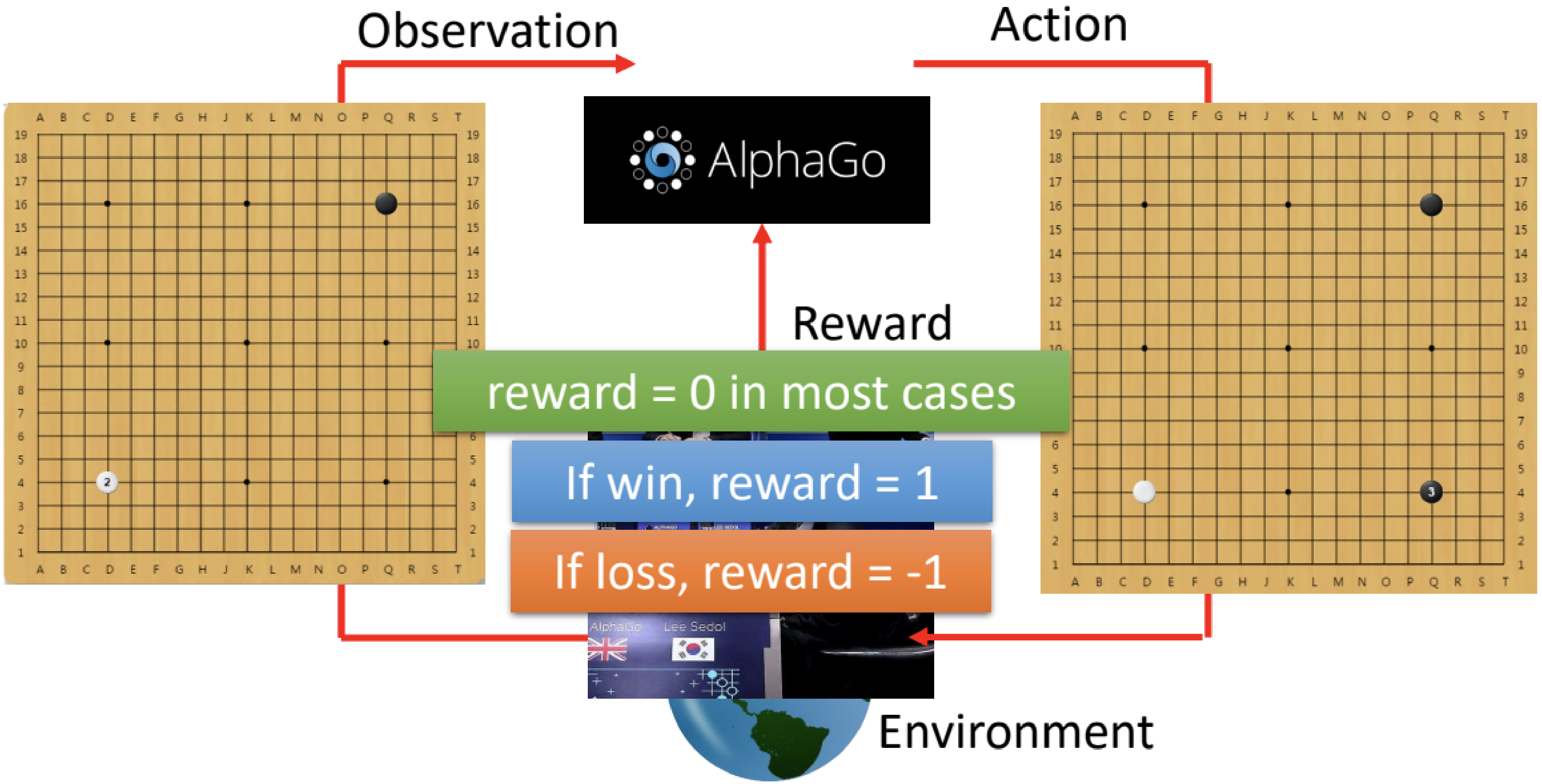


AlphaGo

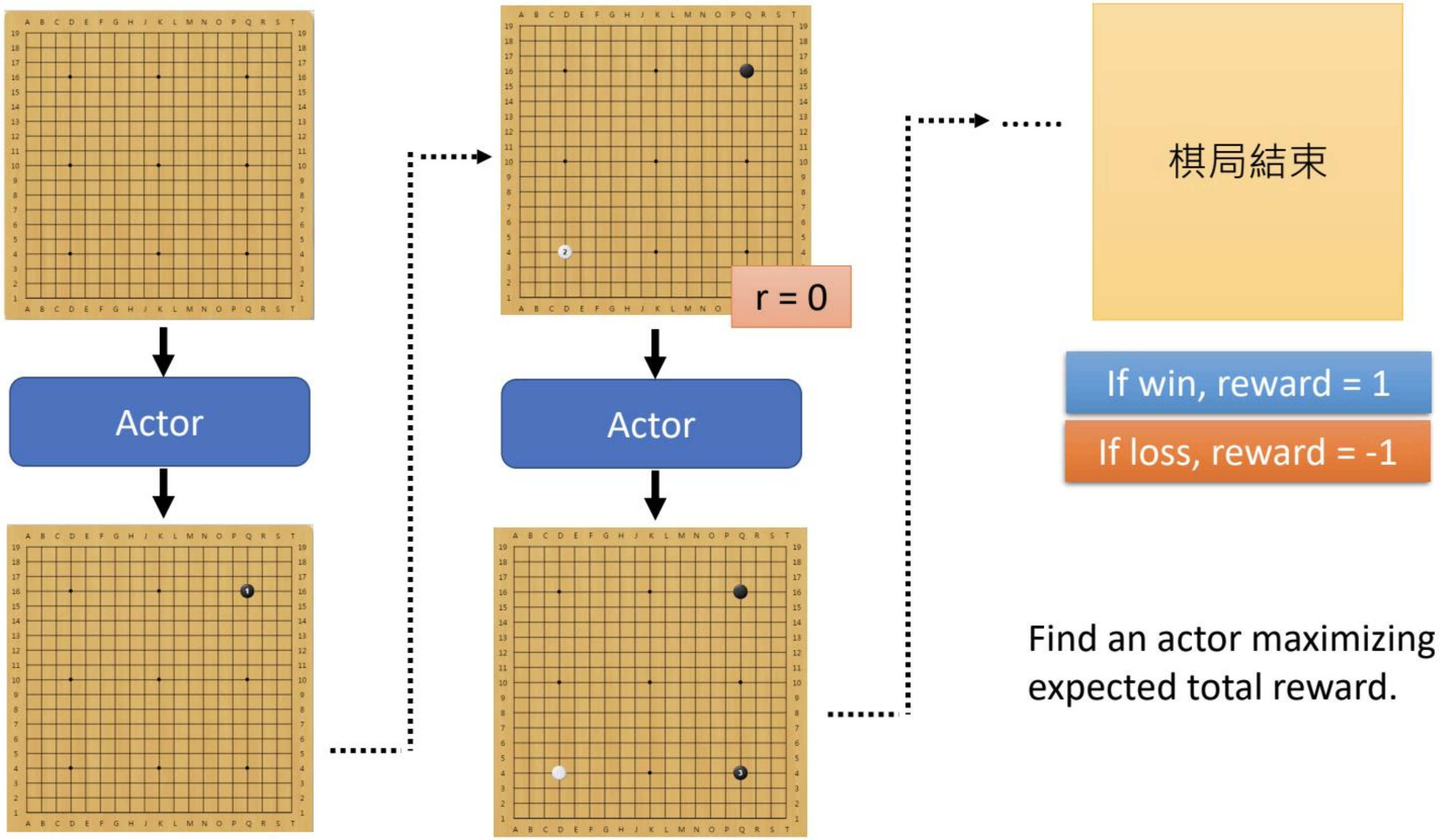

Observation
Action
AlphaGo
Reward
Next Move
Environment
AlphaGo
Lee Sedol

AlphaGo

Find an actor maximizing expected total reward.



AlphaGo



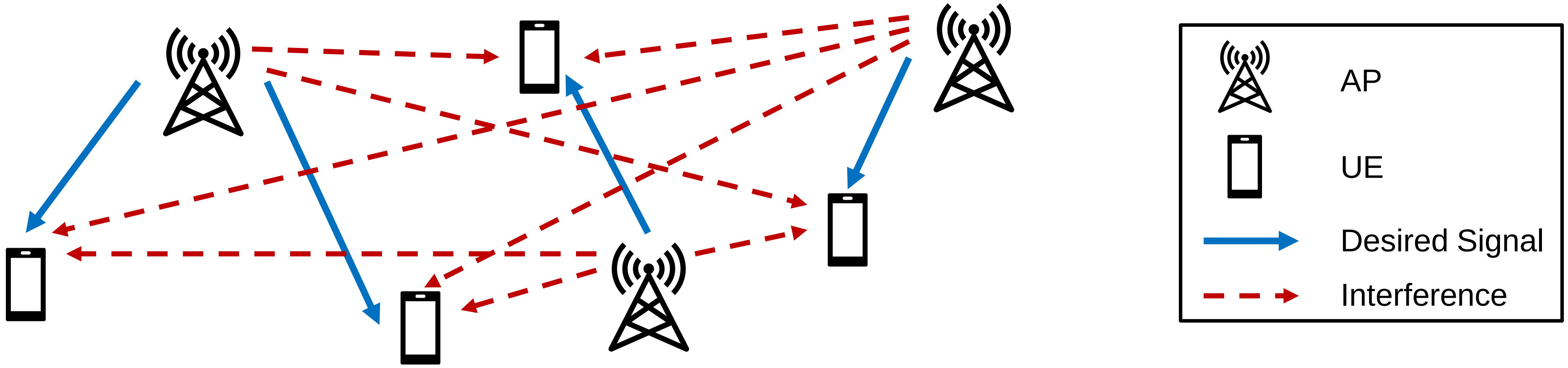
Find an actor maximizing expected total reward.

Resource Allocation

- In a real mobile communication, the system always simultaneously performs several tasks under different demanding services from devices
  - However, each device has different channel conditions that we have to take into account for providing optimum resource allocation
- There exist a great number of resources provided from the central system
  - Spatial resources
  - Temporal resource
  - Frequency bandwidth
  - Transmit power

System Model

- Consider a downlink (DL) scenario with N access points (APs) and K user equipments (UEs).



- Channel Gain as the channel gain set from N APs to K UEs

$$\boldsymbol{H} = \{h_{n,k} | \forall 1 \leq n \leq N, 1 \leq k \leq K\}$$

  - $h_{n,k}$: the channel gain from the n-th AP to the k-th UE

System Model

- The downlink transmit power of AP is denoted as

$$\boldsymbol{P} = \{P_n|\forall n\}$$

- The association indicator set for AP-UE connection

$$\boldsymbol{\rho} = \left\{\rho_{n,k} \in \{0,1\}|\forall n, k\right\}$$

- We can obtain the signal-to-noise-ratio (SINR) for the *n*-th AP to the *k*-th UE as

$$\gamma_{n,k} = \frac{P_n G_{n,k} h_{n,k}}{\sum_{j \neq n}^{N} P_j G_{j,k} h_{j,k} + \sigma^2}$$

  - $G_{n,k}$: the beamforming gain obtained in the beam training process
  - σ^2: background noise

- Then, throughput of the overall system can be expressed as

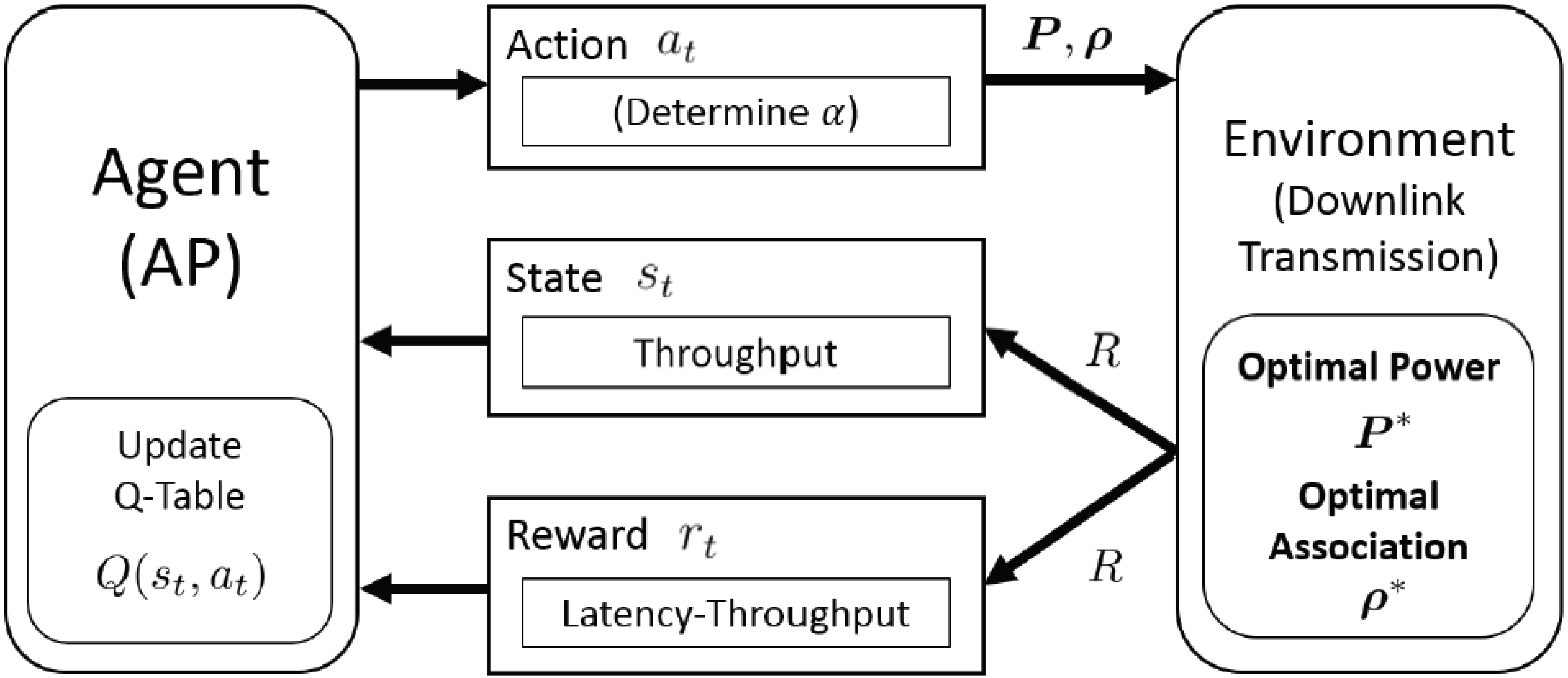
$$R = \sum_{n=1}^{N} \sum_{k=1}^{K} \rho_{n,k} \log_2\left(1 + \gamma_{n,k}\right)$$

Problem Formulation

- We can then formulate the throughput maximization problem that determines the allocation of power and user association, which is given by

$$
\begin{aligned}
\max_{\boldsymbol{P},\boldsymbol{\rho}} \quad & R \\
\text{s.t.} \quad & 0 < P_n \leq P_{max}, && \forall n && \text{maximum allowable power constraint} \\
& \rho_{n,k} \in \{0,1\}, && \forall n,k, && \text{association indicator constraint} \\
& \sum_{n=1}^{N} \rho_{n,k} \leq 1, && \forall k. && \text{one UE can only be served by one AP at most}
\end{aligned}
$$

Q-learning based Resource Allocation



Q-learning based Resource Allocation

- The action is referred as power allocation and association assignment, which is expressed as selection of the maximum value in the Q-table or random action as

$$a_t = \begin{cases} \underset{a_t'}{\operatorname{argmax}}\, Q(s_t, a_t'), \text{if rand}() > \epsilon, \\ \text{random action, otherwise.} \end{cases}$$

  - We exploit ϵ-greedy policy, where $\epsilon \in [0,1]$.
  - t: the current training or testing time step

- Since the state is a discrete variable, we quantize the sum rate objective as discrete states as

$$s_t = \left\lceil \frac{R \cdot s_n}{R_{max}} \right\rceil$$

  - R_{max}: the maximum throughput among off-line collected training dataset
  - s_n: pre-defined constant indicating the total number of given states

Q-learning based Resource Allocation

- Rewards

$$r_t = R = \sum_{n=1}^{N} \sum_{k=1}^{K} \rho_{n,k} \log_2\left(1 + \gamma_{n,k}\right)$$

- According to the Bellman equation, the Q-table is updated by

$$Q_t(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \eta \cdot \left[r_t + \delta \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right]$$

  - $\eta \in [0,1]$: learning rate
  - $\delta \in [0,1]$: discount factor

- In order to address constraint, we often define penalty term to punish the violations.

$$0 < P_n \leq P_{max}, \qquad \forall n$$

Q-learning based Resource Allocation

Algorithm 8: Q-Learning for Resource Allocation

Initialization: Time step $t = 0$, learning rate η, discount factor δ, exploitation probability ϵ

Offline: Collect training dataset of power and association, find R_{max}, and update the Q-table based on **Q-learning**

Online: Based on trained Q-table, perform online testing based on **Q-learning**

Q-learning:

while Not converged as $t < T$ **do**

  $t = t + 1$

  Randomly generate a number $\kappa \in [0, 1]$

  if $t = 1$ or $\kappa > \epsilon$ **then**

    Select random action a_t

  else

    Select optimal action based on $a_t = \underset{a'_t}{\operatorname{argmax}}\, Q(s_t, a'_t)$

  end if

  Obtain power $\boldsymbol{P}$ and association $\boldsymbol{\rho}$ decision based on a_t

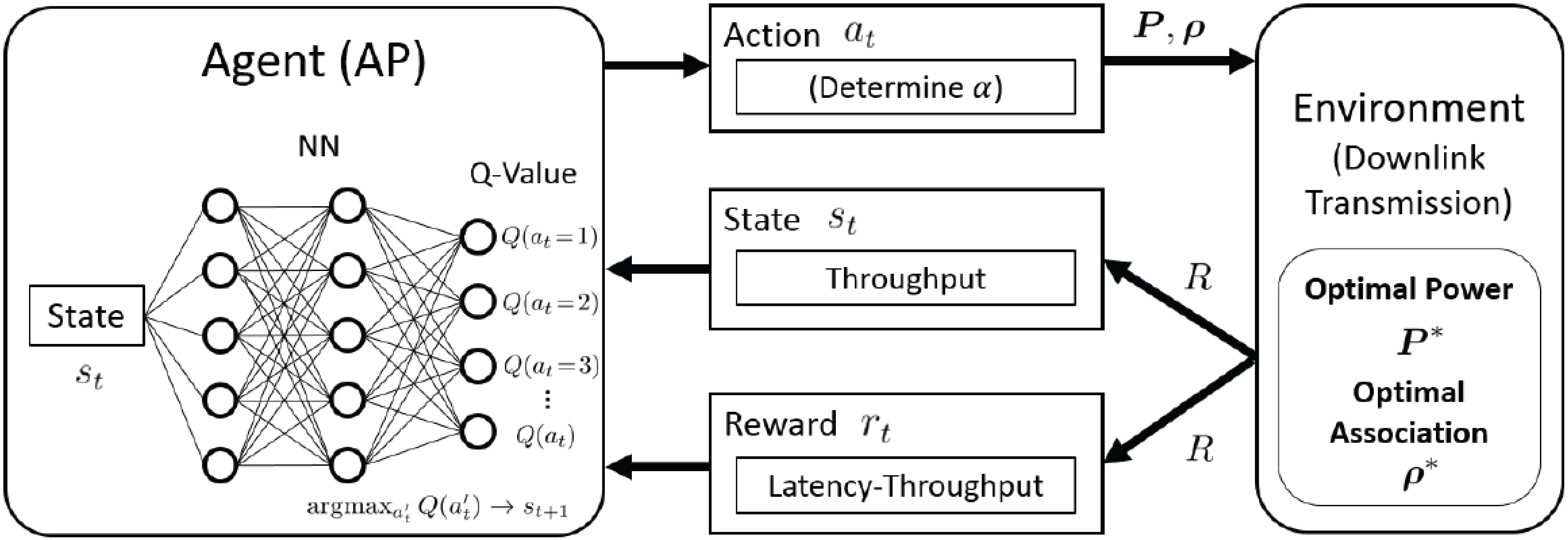
  Obtain reward: $r_t = R$

  Update Q-table: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \cdot \left[r_t + \delta \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$

  Update next state: $s_{t+1} = \lceil \frac{r_t \cdot s_n}{R_{max}} \rceil$

end while

Deep Q-learning



Deep Q-learning