Rate Adaptation by Reinforcement Learning for Wi-Fi Industrial Networks

Giovanni Peserico*, Tommaso Fedullo†, Alberto Morato‡, Stefano Vitturi§, Federico Tramarin¶

*Autec s.r.l.
and *Dept. of Information Engineering*
University of Padova, Italy
giovanni.peserico@phd.unipd.it

†*Dept. of Management and Engineering*
University of Padova, Italy
tommaso.fedullo@phd.unipd.it

‡*CMZ Sistemi Elettronici s.r.l.*
and *Dept. of Information Engineering*
University of Padova, Italy
alberto.morato.3@phd.unipd.it

§*National Research Council of Italy*
CNR–IEIIT
stefano.vitturi@ieiit.cnr.it

¶ *Dept. of Engineering "Enzo Ferrari"*
University of Modena and Reggio Emilia, Italy
federico.tramarin@unimore.it

***Abstract*—Wireless technologies play a key role in the Industrial Internet of Things (IIoT) scenario, for the development of increasingly flexible and interconnected factory systems. Wi-Fi remains particularly attracting due to its pervasiveness and high achievable data rates. Furthermore, its Rate Adaptation (RA) capabilities make it suitable to the harsh industrial environments, provided that specifically designed RA algorithms are deployed. To this aim, this paper proposes to exploit Reinforcement Learning (RL) techniques to design an industry-specific RA algorithm. The RL is spreading in many fields since it allows to design intelligent systems by means of a stochastic discrete–time system based approach. In this work we propose to enhance the Robust Rate Adaptation Algorithm (RRAA) by means of a RL approach. The preliminary assessment of the designed RA algorithm is carried out through meaningful OMNeT++ simulations, that allow to recognize the beneficial impact of the introduction of RL with respect to several industry-specific performance indicators.**

***Index Terms*—Factory Automation, Wi–Fi, Rate Adaptation, Reinforcement Learning, SARSA**

I. Introduction

In the industrial and factory automation scenarios, the rising interest for the Industry 4.0 and the Industrial Internet of Things (IIoT) has fostered the attention towards the deployment of flexible high-performance wireless communication systems at all the levels of the automation pyramid [1]. The IEEE 802.11 standard, which defines the pervasive Wi-Fi networks, has already proven its suitability for industrial automation applications [2], [3], although its actual adaptability to real–time and deterministic communications is still challenging. A fundamental feature of Wi-Fi systems is represented by the Rate Adaptation (RA) capability, which allows a node to adapt the transmission speed to the underlying channel status. Such a feature, actually, revealed beneficial in harsh industrial environments to increase the timeliness and reliability of Wi-Fi networks [4]–[6]. In the aforementioned context, this paper aims at proposing a preliminary design of new RA algorithm for Wi-Fi industrial networks, exploiting the features of Reinforcement Learning (RL). Indeed, following the RL concept, the goal is to make an agent learn how to effectively perform RA under some industrial typical constraints. To the best of the authors' knowledge, currently the usage of RL within the RA context has been poorly addressed in literature, and mainly applied to different application scenarios. This paper considers the Robust Rate Adaptation Algorithm (RRAA) as a starting point for the introduction of RL techniques. Then, the proposed RA algorithm is preliminary assessed by means of a simulation campaign that addresses some meaningful performance indicators.

II. Background

A. Robust Rate Adaptation Algorithm

The behavior of RRAA [7] is based on the concepts of *Short Term Window (STW)* and of loss ratio R_{loss}. Particularly, a specific rate r_i is initially chosen and kept constant for a fixed number of frames, to obtain a STW^i, whose length depends on the rate r_i. At the end of a STW, two pre–defined thresholds for rate r_i, namely Maximum Tolerable Loss threshold (P^i_{mtl}) and Opportunistic Rate Increase threshold (P^i_{ori}), are compared with the loss ratio to define the subsequent rate. The loss rate for a specific STW is simply calculated by Eq. (1).

$$R_{loss} = \frac{\#LostFrames}{\#TransmittedFrames} \tag{1}$$

The RRAA algorithm initially starts selecting the highest rate and defining the corresponding STW size. The packet loss is then compared with both P^i_{ori} and P^i_{mtl}. The rate is decreased to the immediately lower rate r_{i-1} if $R_{loss} > P^i_{mtl}$ to decrease the loss probability. Instead, it is increased to the following higher rate r_{i+1} if $R_{loss} < P^i_{ori}$ since it is supposed that in good conditions the throughput can be increased. Finally, if $P^i_{mtl} \leq R_{loss} \leq P^i_{ori}$, the current rate r_i is maintained.

B. Reinforcement Learning

RL is based on the definition of *states*, that should be able to effectively describe each possible system condition, *actions*, representing all the possible transitions among states,

and *rewards*, that need to be designed accurately to properly evaluate the goodness of an action. An interesting review on RL can be found in [8]. Fig. 1 represents the relationships between these elements, for a simple RL model: specifically, an *agent* is trained observing the reaction of the *environment*. Given the initial state S_t and the chosen action A_t, the training consists in a reward R_{t+1} and a new state S_{t+1}.

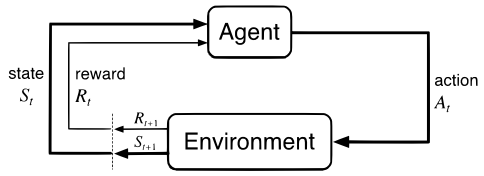

Figure 1. RL basic elements.

Within RL, a probability P is associated with the execution of a specific action starting from a specific state. In this way, we can define the tuples $\{S, A, P, R\}$, whose elements are states, actions, probabilities and rewards, that finally allows to define a Markov Decision Process (MDP). The best policy π, consisting in an ordered sequence of pairs ($state_i$, $bestActionForState_i$), can be found exploiting several different algorithms. The evaluation of a specific policy π is carried out either with a value function $V^{\pi}(s) = E_{\pi}\{R_t|S_t = s\}$, or with an action–value function $Q^{\pi}(s, a) = E_{\pi}\{R_t|S_t = s, A_t = a\}$, representing the expected cumulative reward associated with policy π. Either model–based or model–free techniques can be used to this extent. In the latter case, the evaluation of the value function is made by means of experimental *events*, observed as the result of an agent's action. For example, Monte Carlo or Time Difference methods can be used if, respectively, the evaluation is made once the reward has been produced or by means of a prediction. The latter ones often are called *bootstrapping algorithms*. Clearly this breadth of possibilities represents, on the one side, a significant asset of RL while, on the other one, it increases the difficulty to choose and define the best model.

III. Rate Adaptation Learning by Reinforcement

A. MDP model design

In this Section we will provide a description of the proposed RL-based rate adaptation. It draws inspiration by RRAA, and defines an *action* in such a way that it becomes better as the related packet loss (Eq. (1)) decreases. In addition, the transmission speed is taken into account, such as the reward derived from a trade–off between the minimization of the packet loss and the maximization of the rate. Differently from RRAA, this proposal aims at identifying the "best" rate under any circumstance, removing the constraint of rate changes of only one step, i.e. from r_i to r_{i-1} or from r_i to r_{i+1}, thus allowing to move through all the different rates. The algorithm starts from the the highest rate and with initial loss rate R_{loss} equals to zero. For simplicity and without loss of generality, in the following the basic IEEE 802.11g amendment is considered. Indeed this choice does not affect the consistency of the design, since the proposed algorithm is general and not specifically tied to the Wi-Fi version at hand. IEEE 802.11g includes 8 different rates, from 6 Mbps up to 54 Mbps. Table I presents the association between each rate and the specific action, defined in order to allow moving from each state to that specific rate.

Table I
Actions Table

Rate	6Mbps	9Mbps	12Mbps	18Mbps
Action	$A_t = 0$	$A_t = 1$	$A_t = 2$	$A_t = 3$
Rate	24Mbps	36Mbps	48Mbps	54Mbps
Action	$A_t = 4$	$A_t = 5$	$A_t = 6$	$A_t = 7$

For each rate r_i, a set of ten R_{loss} intervals is defined. The states are hence designed to fully characterize the system status: 80 states are identified, defined by the transmitting rate r_i and R_{loss}. The following Table II provides their description. Furthermore, given the action A_t that allows to select the specific r_i, each state S_t is uniquely identified from R_{loss} and r_i by the following Eq. (2).

$$S_t = \lfloor (R_{loss} \cdot 100)/10 \rfloor + 10 * A_t \tag{2}$$

Table II
States table

r_i \ R_{loss}	$< 10\%$	$> 10\%$ $< 20\%$	$\cdots$	$> 80\%$ $< 90\%$	$> 90\%$
6Mbps	$S_t = 0$	$S_t = 1$	$\ldots$	$S_t = 8$	$S_t = 9$
9Mbps	$S_t = 10$	$S_t = 11$	$\ldots$	$S_t = 18$	$S_t = 19$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
54Mbps	$S_t = 70$	$S_t = 71$	$\ldots$	$S_t = 78$	$S_t = 79$

The assignment of rewards is a critical point of RL. In the proposed algorithm, rewards are calculated by Eq. (3), where the more R_{loss} is reduced and $Rate$ is increased, the higher the reward. The factor β is introduced as a weighting benchmark. Different values of β were tested, and $\beta = 0.45$ was chosen, since it has proved to be the best trade-off between the loss rate minimization and the throughput maximization.

$$\begin{aligned} R_{t+1} = \quad & \beta \cdot (-R_{loss_{t+1}} + R_{loss_t}) \\ & + (1 - \beta) \cdot \frac{Rate_{t+1} - Rate_t}{Rate_{t+1} + Rate_t} - 1; \end{aligned} \tag{3}$$

An additional special management of the boundary rates is foreseen, choosing the minimum rate when the loss does not decrease and it is bigger than 50%, and rewarding the maximum rate when the loss does not increase and it is smaller than 50%.

B. RL Algorithm

Each episode E is defined as a sequence of windows $\{w_0, w_1, \ldots w_n\}$ where n is a fixed and pre–determined value. In our case, it is not possible to guarantee an exploration of all

the defined states within each episode, as the loss rate relies on many uncontrollable factors, such as the noise. Furthermore, every episode should be different from the others. For these reasons a bootstrapping algorithm is chosen, to make the agent learn from experience. In particular, since the policy is updated by means of an online procedure, then the State Action Reward State Action (SARSA) algorithm [9] is chosen to this extent. This technique is realized performing the evaluation of the action–value function expressed by Eq. (4), whose value is then stored in the Q State Table (QST).

$$QST[S_t][A_t] = QST[S_t][A_t] + \\ + \alpha * (R + \gamma * QST[S_{t+1}][A_{t+1}] - QST[S_t][A_t]) \quad (4)$$

Indeed, SARSA foresees, at each iteration, to update the Q value proportionally to the observed reward and to the difference between the next (i.e. calculated for the subsequent state and action) and the current Q values. α and γ need to be properly tuned by a trial and error mechanism, good values are $\alpha = 0.1$ and $\gamma = 0.7$. Moreover, in order to make the agent learn by reinforcement, a good exploration of all the pairs (S_t, A_t) has to be guaranteed. To this aim, the choice of the subsequent action is performed, with probability ϵ, in a random way, while in any other case the greedy one is chosen, namely ϵ–Greedy algorithm. In practice, A_{t+1} is selected randomly the 10% of times to perform *exploration*, while the best action is *exploited* the other 90% of times. The RL algorithm is presented in Algorithm 1, where the boundary situations are not represented for simplicity.

Algorithm 1: RL algorithm

```
cR, pR ← maxRate //current and previous Rate
cS ← 70 //current State
while True do
    starts counter
    sends frame using cR
    while not(counter == 0) do
        R_loss ← update(R_loss)
    R ← Equation 3
    A_{t-1} ← A_t
    A_{t+1} ← ε–Greedy(S_{t+1})
    QST[S_t][A_t] ← Equation 4
    A_t, S_t ← A_{t+1}, S_{t+1}
```

After a training of 200 episodes with $n = 1000$, where each window w_i is a sequence of 40 consecutive messages, the "best" policy for a specific situation or location is identified. An attractive computational advantage consists in the possibility to store the best policies in memory, to be properly used when needed.

IV. Simulation Outcomes

OMNeT++ is used to simulate a simple Wi–Fi-based WLAN composed by two devices. In the experiments, the two devices move within the environment with different speeds, continuously varying their relative packet loss. A further attenuation is generate introducing a simulated obstacle between the elements of the network. The IEEE 802.11 Nist error model, already implemented in OMNeT++, is used to introduce a stochastic error model, allowing to explore every possible state. Finally, the Two Ray Interference model [10] is used for the path loss. The simulations carried out lasted 100 s. In this scenario, a node sends packets to the other with a period of $T = 1$ ms. A first result, reported in Fig. 2, is relevant to the behavior of RRAA. It appears evident that the RA activity strongly depends on the loss rate evaluation (Eq. (1)). Indeed, when R_{loss} is low (that is, when the throughput is high), the algorithm choose higher rates, and vice–versa.

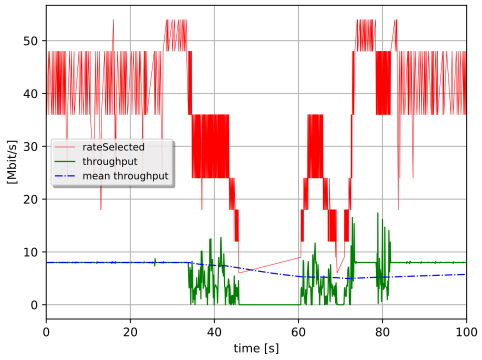

Figure 2. Throughput and rates selection for RRAA.

The RL algorithm is tested at first when exploration is still being performed, choosing the 10% of times a random rate, namely "RL with training". Secondly, the previous training activity is capitalized obtaining a best policy, whose behaviour is presented in Fig. 3. Here, it is possible to appreciate the correct behaviour of the proposed RL RA algorithm in relation with the throughput. In particular, only when the throughput is low the algorithm chooses low rates.

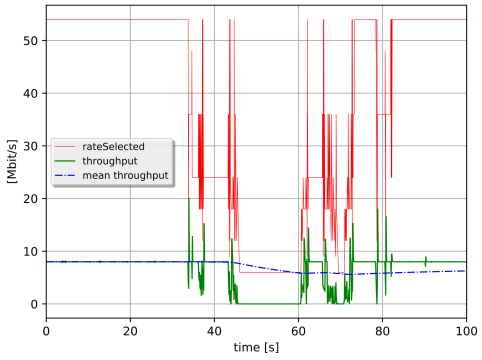

Figure 3. Throughput and rate selection for policy–based RL.

A comparison of the aforementioned three situations is carried out in Fig. 4, by means of the mean throughput.
We can notice that, when the loss is high (and the throughput is consequently low) the rate selection activity is unstable

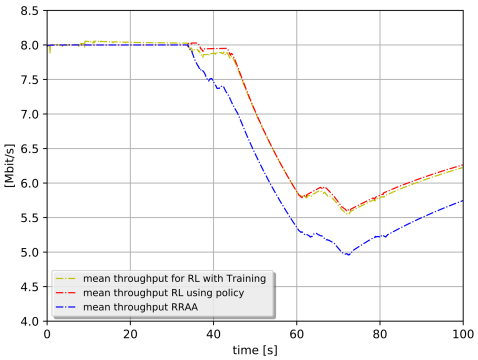

Figure 4. Throughput comparison.

(Fig. 3), since this entails the selection of very different subsequent rates. This behaviour is owed to the particular definition of the rewards and the percentage of the exploration activity done during the training phase. Indeed, the latter extreme situation of Fig. 4 has been presented to underline that the choice of selecting the rate at each step from the whole set of rates may provide benefits in terms of throughput. Clearly, a real implementation will need a more accurate rewards tuning. Nevertheless, the aim of this preliminary analysis is to prove the effectiveness of exploiting RL for RA. From an industrial networks perspective, a further performance evaluation needs to be carried out considering the transmission delay experienced by the packets in the network. To this aim, Table III reports the outcomes of this analysis, in terms of both the mean and standard deviation values. In addition, the Cumulative Distribution Function (CDF) of the delay is presented in Fig. 5.

Table III
SIMULATION RESULTS (100.000 PACKETS SENT)

Algorithm	Received packets	R_{loss} [%]	Delay [ms] Mean	Delay [ms] Std. Dev.
RRAA	71084	28.016	35.255	140.970
RL with training	76191	23.809	22.742	129.773
RL best policy	77401	22.599	16.122	123.917

Both Table III and Fig. 5 demonstrate that the RL techniques performs better than the RRAA algorithm. Firstly, RL techniques have an higher number of successful transmissions stemming from the lower packet loss. Secondly, the introduction of RL permits to decrease the End–to–End delay in terms of mean and standard deviation. This reveals that RL–based RA algorithms are able to converge to a better trade–off between rate maximization and loss minimization. As a final observation, the RL with training policy, characterized by a randomised rate selection in the 10% of times, performs slightly worse than the RL using the best policy.

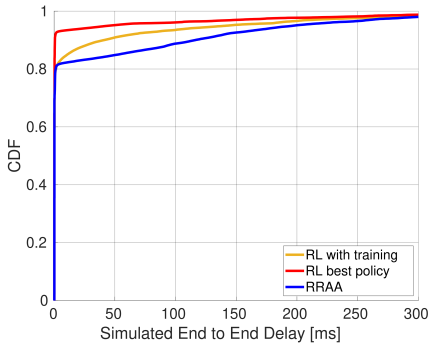

Figure 5. End to End Delay CDF.

V. CONCLUSIONS AND FUTURE RESEARCH ACTIVITIES

Reinforcement Learning has proven to be promising for improving the RA capabilities of Wi-Fi networks. The preliminary simulation assessment presented in this paper showed that RL introduces appreciable improvements over RRAA, and a further algorithm tuning, with smoothed rewards might allow even better results. Future directions of this research will be focused on a more comprehensive implementation and assessment of the presented RL–based RA algorithm, as well as on the application of RL to other industrial specific RA algorithms. For example, the usage of the SNR information within an RL–based policy may considerably improve the performance of the algorithm. Moreover, the implementation of RL techniques for RA on larger networks, comprising more nodes, and on an experimental testbed is foreseen, allowing to consider other issues, such as required computational efforts.

REFERENCES

[1] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation ethernet, iiot, and 5g," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 944–961, 2019.

[2] F. Tramarin, A. K. Mok, and S. Han, "Real-time and reliable industrial control over wireless lans: Algorithms, protocols, and future directions," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1027–1052, 2019.

[3] A. Aijaz, "High-performance industrial wireless: Achieving reliable and deterministic connectivity over ieee 802.11 wlans," *IEEE Open Journal of the Industrial Electronics Society*, vol. 1, pp. 28–37, 2020.

[4] F. Tramarin, S. Vitturi, and M. Luvisotto, "A dynamic rate selection algorithm for ieee 802.11 industrial wireless lan," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 846–855, 2017.

[5] M. A. Gawas and R. Tambi, "Data rate adaptation algorithms survey for ieee 802.11 networks," *International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017.

[6] M. Yofune, M. Suto, Y. Amezawa, and S. Sato, "Rate-adaptation scheme for multiband wlan system," in *2018 Advances in Wireless and Optical Communications (RTUWO)*, 2018, pp. 72–77.

[7] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 146–157.

[8] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an introduction*, 2nd ed. The MIT press, 2018.

[10] C. Sommer and F. Dressler, "Using the right two-ray model? a measurement–based evaluation of phy models in vanets," in *MobiCom*, 2011.