Maximizing Airtime Efficiency for Reliable Broadcast Streams in WMNs with Multi-Armed Bandits

Giovanni Perin*, David Nophut†, Leonardo Badia* and Frank H.P. Fitzek†
*Department of Information Engineering, Università degli Studi di Padova, Via Gradenigo 6/b, 35131 Padova, Italy
†Chair of Communication Networks, Technische Universität Dresden, Helmholtzstraße 18, 01069 Dresden, Germany
Email: {peringiova, badia}@dei.unipd.it, {david.nophut, frank.fitzek}@tu-dresden.de

***Abstract*—Wireless broadcast routing is a complex problem, shown in the literature to be NP-complete. Current protocols implement either heuristics to find solutions that are not guaranteed to be optimal or classic flooding. However, many future use cases, like automotive applications, industrial robotics, and multimedia broadcast, will require efficient yet reliable methods. In this work, we use contextual multi-armed bandits together with opportunistic routing (OR) and network coding (NC) to find approximately optimal solutions to the problem of broadcast routing in a distributed fashion. Each router independently learns its own transmission credit, i.e., the number of packets to forward for each innovative packet received, so that the airtime cost, subject to latency constraints, is minimized. Results show that the proposed solutions, particularly the deep learning based one, vastly improve the overall reliability, while performing close to MORE multicast in terms of airtime and to B.A.T.M.A.N. in latency, both being the best candidates in the respective discipline among the tested ones.**

***Index Terms*—Reinforcement learning, multi-armed bandits, wireless mesh networks, routing, broadcast.**

I. Introduction

Access to a shared medium puts a limit to data transmission and can therefore decrease the effectiveness of routing algorithms designed to improve the information flow or the latency. We consider *broadcast routing*, defined as the challenge of reaching all the nodes of a network as end destinations of a stream, not to be confused with link-local broadcasting, which can be actually employed for opportunistic routing (we will exploit it in our proposal). While unicast routing is well researched, and elaborate schemes have been proposed for it [1], even providing optimal solutions in airtime [2], practical and capacity-achieving algorithms for broadcast routing have not been found. As a consequence, state-of-the-art implementations use approximated heuristics [3]–[5] or even approaches like flooding, resulting in excessive transmissions [6].

We remark that broadcast routing is of major interest in scenarios such as wirelessly connected control systems, media broadcast with mobile devices, and especially upcoming V2X use cases, e.g., platooning cars being reliably informed about movement commands or status updates. For the latter scenario, geographical routing was standardized by ETSI [7]. However, the wireless channel gain only partially relates to physical distance. Furthermore, even knowing geographical distances does not solve the problem of minimal-cost routing. This makes our proposal both timely in scope and original, since we evolve over the state of the art thanks to the application of recent trends in reinforcement learning (RL) [8], [9].

Specifically, we develop a multi-agent RL framework based on contextual multi-armed bandits (cMABs) [10] to solve the problem of efficient broadcast routing in terms of airtime, with constraints on latency in the form of hard timeout thresholds. We propose two solutions, one based on classic Q-tables and another implemented thanks to Bayesian neural networks (BNNs), and we compare them with standard reference cases, i.e., MORE multicast [2] and B.A.T.M.A.N. [6] in terms of reliability, airtime, and latency. Notably, our proposals are realized by making all the nodes of the network as independent learners, thereby allowing for distributed implementations.

As a result, we can make the following contributions. First of all, we discuss how we can implement our techniques based on cMABs in a distributed fashion; since this requires to exchange information among the node/learners, we also explore techniques to keep the control information exchange limited. Still, this offers a significant advantage towards scalability with respect to fully centralized strategies. Moreover, besides exchanging some information among neighbors, we propose a time-varying reward function, which forgets outdated situations, to cope with the problem of non-stationarity typical of multi agent environments [11]. Second, and most importantly, we are able to show the superiority of our approach compared to the benchmark cases, which is especially true from two different perspectives: not only do we improve the objective function of our broadcast routing, i.e., the airtime cost, but we also show that the fraction of nodes meeting the delivery deadline is significantly improved. Thus, we improve the overall performance while at the same time allowing for timely information provision to a vaster set of nodes, and we realize this without resorting to a centralized approach.

978-1-7281-9656-5/20/$31.00

The rest of the paper is organized as follows. In Sec. II the benchmark protocols are briefly presented as well as relevant work found in the literature about routing employing RL. Sec. III is dedicated to the definition of the optimization problem, the details of the framework, and the tools employed to solve it. In Sec. IV the simulation experiments are presented, and results are shown and commented. Finally, in Sec. V relevant findings are summarized, and possible future work is suggested.

II. Related Work

An optimal solution concerning airtime cost exists for opportunistic unicast routing, including also the recoding ability of network coding [2]. Specifically, random linear network coding (RLNC) [12] can significantly reduce overhead transmissions in wireless mesh networks (WMNs). Using this technique, actually, a relay does not need to discriminate which packet to forward, and the risk of sending redundant information is significantly decreased. This protocol, named MORE, computes the optimal transmission credit, i.e., the number of packets to forward for each innovative packet received from upstream, based on distance metrics such as the expected transmissions (ETX) or the expected anypath transmissions (EAX) [13]. However, because a scalar network order through distance metrics like these is missing in a broadcast scenario, every node can be assumed to be closer to the source. Therefore, the question of setting a transmission credit in the network to minimize the airtime cost remains. Moreover, research shows that broadcast routing is harder to solve. In [14], it is shown that the problem of finding the minimum broadcast latency is NP-hard, while [4] proves that finding minimum-cost multicast trees is NP-complete, even without exploiting the broadcast advantage of the wireless medium, which adds another layer of complexity. In order to approximate optimality, heuristics are used [4], [5], sometimes by step-wise optimization, e.g. in the OLSR protocol, where a 2-hop Dijkstra-like optimization is performed [3]. Another popular implementation of a wireless mesh protocol, B.A.T.M.A.N., uses a simple flooding approach, where every node repeats each message three times [6]. The multicast extension of the cited MORE protocol [2] computes the optimal unicast transmission credit for each node belonging to the multicast group, and then chooses the maximum value for transmitting. The procedure is repeated every time a destination received the message, excluding that node from the new multicast group. However, this is not necessarily optimal, as can be easily shown.

The tradition of applying RL to routing problems is rooted in Q-routing [15]. This work deals with the problem of path selection, minimizing the end-to-end latency. The method proves to be beneficial in high load conditions, where the shortest path may not be the quickest one. However, the approach of Q-routing and the related following works were only based on Q-tables, thus not being scalable. In more recent years, neural networks (NNs) started to be profitably employed in routing problems applied to software defined networks (SDNs). These architectures, however, have a centralized network controller, and therefore single agent RL is applied. Examples can be found in *QAR* [16] and *DROM* [17], where authors also investigate throughput maximization. A distributed approach using deep Q-networks with LSTMs in WMNs is adopted instead in [18], outperforming Q-routing. Multicast routing, however, is still widely unexplored. A survey on this can be found in [19], with works dealing mostly with multiple path selection and learning algorithms aiming at improving existing heuristics. Very few works, however, combine RL with the possibilities of OR and RLNC. A distributed and asynchronous algorithm for unicast communications, *d-AdaptOR,* is proposed in [20], where RL is exploited in the absence of reliable estimation of the channel quality. The objective is the minimization of the expected number of transmissions for unicast streams, and the method outperforms ExOR [1]. The broadcast problem of live video streaming is tackled instead in [21], where packets are forwarded according to the learned delay-based metric.

Differently from the literature, we do not deal with path selection minimizing the end-to-end latency nor maximizing throughput. The proposed algorithms are dedicated instead to optimize the number of packets to forward using RLNC, to minimize the number of transmissions.[1] We add latency constraints typical of many applications, increasing the complexity of the problem. Unlike previous work, which concentrated mostly on unicast communications, we consider broadcast routing, which makes our proposal original.

III. Methods

A. Problem statement

The optimization objective of this work is the airtime cost minimization, pursued by finding an approximately optimal transmission credit c_i^* for each relay node i, given the context of the transmission. We consider a simple scenario with single rate links and fixed packet error rates per link. Therefore, the airtime cost for delivering a message throughout the whole mesh is defined as the sum of the total number of transmissions performed by each node. The optimization problem is thus defined as follows:

$$\min \sum_{i=1}^{N} n_{tx}^i \tag{1}$$

where n_{tx}^i represents the total number of transmissions executed by node i. Furthermore, as broadcast applications often involve time deadlines for the execution, hard latency constraints are added to the optimization. We demand that all the nodes of the mesh receive the message within

[1] Actually, this is inversely proportional to throughput in case of no spatial reuse.

a deadline, that is, the delay experienced by the last node able to decode the message is below a threshold τ. Hence,

$$\max_i (d_i) < \tau \tag{2}$$

with d_i being the latency associated with node i for the entire message. A feasible transmission credits combination $[c_0, \ldots, c_n]$ must guarantee that the information flow is 1 at all the destinations. However, this requirement is implicitly satisfied in the formulated problem thanks to constraint (2), as it is asked that all the nodes receive the entire message in a limited amount of time.

B. Contextual Multi-Armed Bandits

A cMAB is a simpler form of RL whose goal is to select the best action $a \in \mathcal{A}$, with $\mathcal{A}$ being the set of actions, according to a policy π, and given a context $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of system states [10]. Every agent keeps a state-action value function $Q^\pi(s,a)$, which denotes the value of taking action a in state s, following a policy π. This function, also called Q-function, or Q-value is equal to the expected reward given to the bandit while in state s and after taking action a. It is computed updating progressively its value with the new observation in the following way:

$$Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha[r - Q^\pi(s,a)] \tag{3}$$

where r is the reward given to the bandit from the environment, i.e., the system, and α is the learning rate. Intuitively, the best choice would be to select, at any instant t, always the action a_t leading to the best Q-value, that is $a_t = \arg\max_a Q^\pi(s_t,a)$. Such a policy, which only exploits the Q-value, is called greedy, but it has the problem that the state-action pair space would not be searched enough to find out which is the actual best action, given the context. Therefore, some exploration must be introduced, at least at the beginning of the learning process.

In this work, two cMAB based solutions are compared, the first using a classic discrete tabular approach and the other one approximating the Q-function with an NN. The exploration-exploitation trade-off policy for the Q-table is realized using an upper confidence bound (UCB1) decision rule for every context, expressed as

$$a_t = \arg\max_a \left[Q^\pi(s_t,a) + c\sqrt{\frac{\ln t}{N_t(s_t,a)}} \right] \tag{4}$$

with $N_t(s_t,a)$ denoting the number of times that action a was chosen while in state s_t, and $c > 0$ controlling the amount of exploration. The algorithm based on deep learning earns instead the needed randomized behavior thanks to Thompson sampling, realizing Bayesian NNs with concrete dropout, a recent technique allowing to tune progressively the dropout parameter [8], [9].

C. Distributed algorithms

Each node of the network has its own cMAB and is regarded as an agent. Nodes are hence trained as independent learners in a multi-agent scenario. To improve cooperation, some information is exchanged between neighbors. This additional overhead constitutes the context given as input to the bandits, and it comprises: i) source ID, ii) destination IDs, iii) delivery probabilities to the neighbors, and iv) most recent available c_is picked from the neighbors. We highlight that the required information is at local level and can be retrieved just by piggybacking it over data packets, and by estimating periodically the channel quality. The topology of the mesh at global level is instead learned by the bandits as a consequence of experience.

1) Tabular bandits: every router i keeps its own table Q_i, initialized to an all-zero matrix of size $|\mathcal{S}| \times |\mathcal{A}|$, where $|\mathcal{S}|$ and $|\mathcal{A}|$ denote the cardinalities of the sets of states and actions respectively. When dealing with large networks, it is infeasible to store all the cited data in a table. Therefore, the state space dimensionality of this algorithm is reduced, taking into account only the source ID, hence having a cardinality equal to the number of possible sources. The training phase is performed online: at the beginning of a batch transmission, the router gets the source ID as a context s_t and sample an action a_t according to (4). This sampled real value corresponds to the transmission credit c_i that will be played by the router during the whole batch transmission. Each node should also keep track of the number of transmissions performed and broadcast the final value when it stops forwarding packets. This way, the total number of transmissions accomplished by the network can be retrieved, thus obtaining the reward. At the end of each batch, every agent updates $Q_i^\pi(s_t,a_t)$ according to (3).

2) BNNs: the major advantage of using an NN in RL is that the dimension of the input space can be increased at low memory and complexity costs. Therefore, each BNN_i belonging to router i is fed with the whole information aforementioned. Thus, this method would support also general multicast communications, as long as destination IDs are specified. The learning algorithm is basically the same as the tabular bandit, except that action sampling is fulfilled with the forward pass of the BNN, taking the transmission credit corresponding to the maximum Q-value predicted. Moreover, because training an NN is expensive, it is not done for every batch. A replay memory buffer is kept at each node, and the BNN is trained with an exponentially decreasing rate, but with more samples each time, until a certain maximum retraining period T and dataset size M. Parameters regarding the size of the dataset of concrete dropout are tuned accordingly.

D. Reward shaping

Because in RL the expected reward is maximized, the reward function has its maximum when the message is successfully broadcast within the latency constraint and

with the smallest airtime cost experienced. A penalty is given, instead, when the constraint is violated, and intermediate values encode conditions for which the timeout threshold is satisfied, but the airtime is suboptimal. Generally, multi-agent games are played in highly non-stationary conditions, because each agent modifies the environment for other players with continually changing policies [11]. To overcome this problem, other than the aforementioned information exchanged among neighbors, the reward is made a function of the past but only most recent observations. This way, the effects of outdated policies are progressively forgotten. The proposed procedure is also useful as the channel conditions, or even the topology of the network, can change during learning, for instance as a result of a movement of some nodes. Therefore, each router, other than the replay memory buffer, keeps track of the maximum and minimum number of transmissions that satisfied the delay constraint for each context, refreshing these values when they get outdated. If a timeout is not met, the network airtime cost experienced is firstly linearly mapped to $[0, 1]$ thanks to these two stored values:

$$r_l(s_i, a_i) = \frac{\sum_j n^j_{tx} - \min_{h^\pi_i} N_{tx}}{\max_{h^\pi_i} N_{tx} - \min_{h^\pi_i} N_{tx}} \tag{5}$$

with h^π_i denoting the past history of node i under policy π. Then, a non-linear weighted gamma transformation is applied:

$$r_{NN_i}(r_l) = \begin{cases} w(1 - r_l^\gamma) + (1 - w) & \text{if } \max_j(d_j) < \tau \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $w \in [0, 1]$, which is the gap width, balances the trade-off between the avoidance of the timeout and the numerical differentiation among similar solutions, and γ controls the non-linearity of the function, encoding quality and speed of convergence.

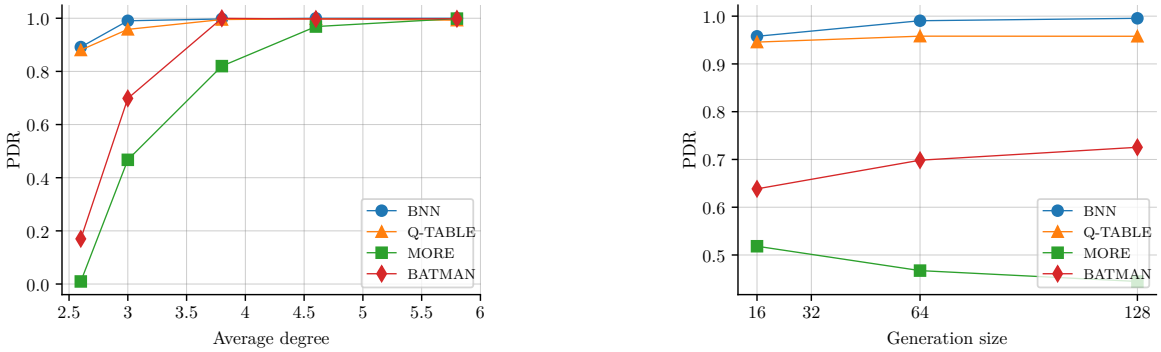
IV. Results and Discussion

A. Simulation settings

In this work, the two learning algorithms are compared with the multicast versions of MORE and B.A.T.M.A.N. To provide a fair comparison between routing protocols, each one uses RLNC with a finite generation size to examine the practicability of the approaches. A transmission batch is only successful if all nodes in the network can decode. Rules of transmissions are the same for every protocol, except for the policy selecting the transmission credit c_i. 1) Every time a node receives an innovative packet, i.e., linearly independent with those already in the buffer, it increases its credit counter of c_i. 2) A node transmits in the current slot only if its credit counter is positive, decreasing it by 1. 3) When a node can decode, an ACK is broadcast and assumed to be received with no loss. 4) If a node received ACKs from all of its neighbors, it stops transmitting. The transmission credit c_i is chosen by the bandits as a consequence of learning among a discrete set of 50 values in $[0, \max_j(1/p_{ij})]$, where p_{ij} is the delivery link probability from i to j. MORE selects c_i values according to the rules defined in [2], requiring global knowledge on the topology of the mesh, whereas B.A.T.M.A.N. chooses a fixed $c_i = 3$ for all the routers. We highlight that an enhanced version of this protocol is considered, with encoded packets and feedback.

The behavior of 15 meshes with 10 nodes having routing functions and created with a random geometric graph generator with distance correlated error probabilities on links is evaluated. Experiments are performed with 5 different average degrees and 3 coding generation sizes, and last 2000 broadcast epochs comprising 32 batch transmissions. The last 500 epochs are used for evaluation: the exploration policy is switched off, and actions are selected greedily.

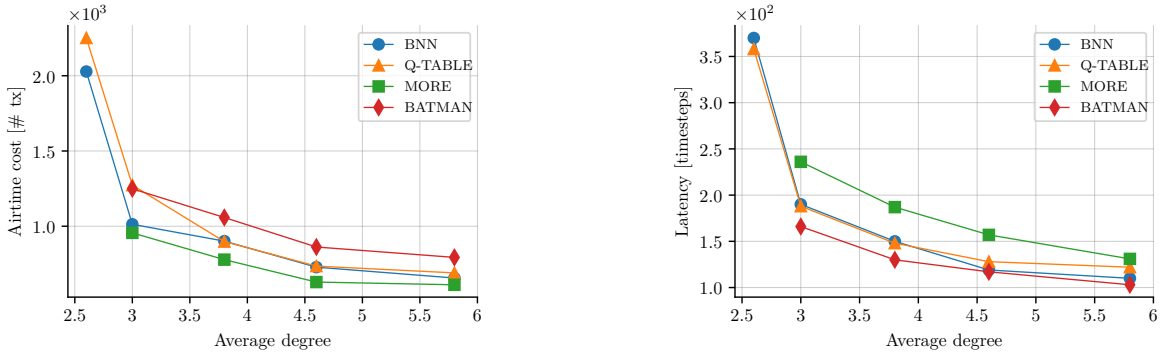
The simulator is entirely written in Python, using PyTorch to develop the BNNs. They have a simple feed-forward structure, with three inner layers with size 128, the first one being composed of 4 separate sets of 32 units, one for each input piece of information. The output layer is trained to predict the Q-values for each action with the MSE criterion. Simulation parameters are listed in Tab. I, unless differently stated in what follows.



(a) Generation size = 64

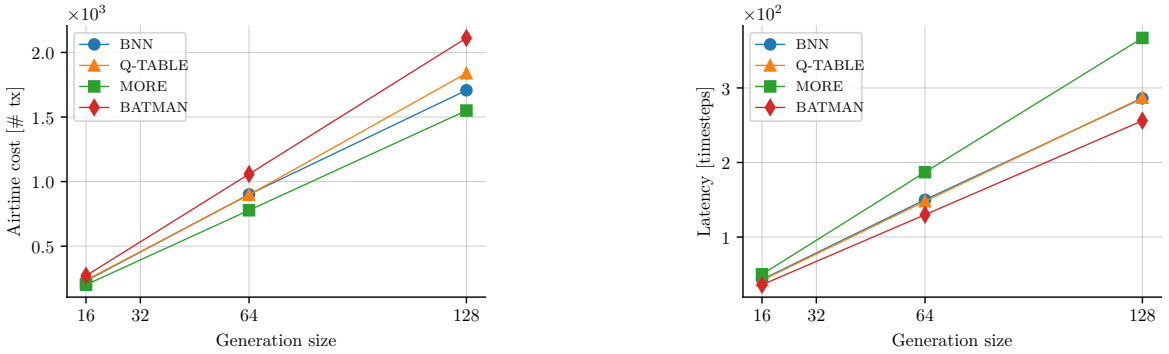
(b) $\bar{d} = 3$

Figure 1: Packet delivery ratio (PDR) w.r.t. the average degree and the generation size.



(a) Airtime (b) Latency

Figure 2: Median of airtime and latency w.r.t. the average degree (Generation size = 64).



(a) Airtime (b) Latency

Figure 3: Median of airtime and latency w.r.t. the generation size ($\bar{d} = 3.8$).

Parameter	Value
c (tabular exploration coefficient)	$\sqrt{2}$
α (tabular learning rate)	$\frac{1}{N_t}$
lr (BNN learning rate)	0.001
$\lvert\mathcal{A}\rvert$ (# of actions)	50
minibatch size (for training BNNs)	64
w (reward function weight)	0.9
γ (reward function exponent)	1.2
C (BNN max buffer capacity)	8000
K (exponential retraining coefficient)	1.01
N (# of nodes in the mesh)	10
G (coding generation size)	$[16, 64, 128]$
τ (latency threshold in timesteps)	$[2000, 1500, 1000, 700, 400]$
$\bar{d}$ (average degree)	$[2.6, 3.0, 3.8, 4.6, 5.8]$
batch size (# of subtransmissions)	32

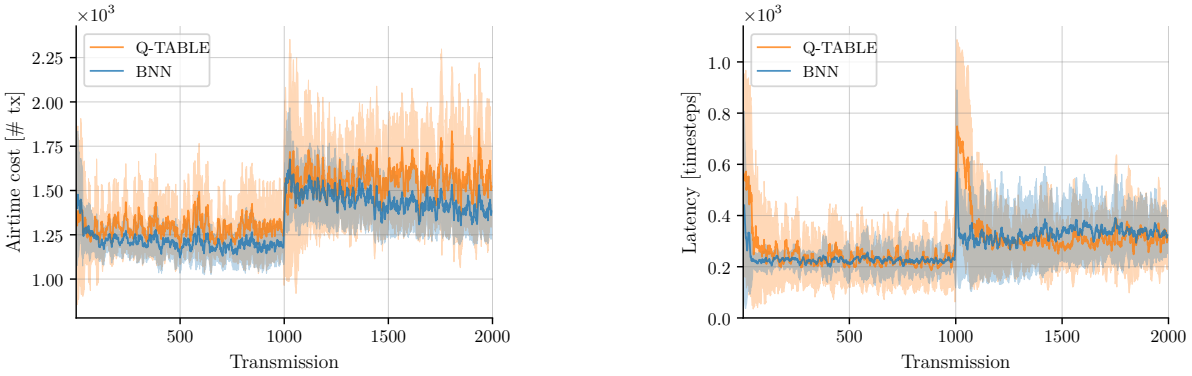
Table I: Summary of the parameters used for the simulations.

B. Reliability

The packet delivery ratio (PDR) is shown in Fig. 1. Even when the network is poorly connected, RL can guarantee that the full message is retrieved in time by all the components consistently above the 90% of the cases, and above 99% as soon as $\bar{d} \geq 3$. MORE multicast becomes more reliable as the connectivity increases, because minor contributions increase the robustness against dead ends. These may occur quite often in poorly connected networks, because the generation size is not infinite and MORE only contemplates average results, without considering deviations. B.A.T.M.A.N., instead, suffers from the fact that 3 forwarded packets are too few in regions with low connectivity, unless link qualities are sufficiently high. In case the connectivity is high enough, messages are correctly received with high probability, but the medium is often filled with useless overhead and energy is wasted. It is also notable that the learning algorithms are slightly more robust with higher generation sizes (Fig. 1b), because data collected for training is more reliable and stabler. On the other hand, MORE decreases its robustness because the latency threshold is fixed, and sending more data requires more time, making MORE's minimal solutions sometimes infeasible.

C. Airtime and latency

As explained in Sec. III, delay constraints are also considered, conflicting with the general purpose of minimizing the airtime cost. They may force the bandits to choose higher transmission credits than the optimal ones. Nonetheless, results evaluating airtime and latency, shown in Figs. 2 and 3, prove that the proposed algorithms can reach good performance in both metrics. In these plots, only assessable points are depicted. Hence, the benchmark protocols performance relative to $\bar{d} = 2.6$ is not considered, as the number of available points is statistically misleading



(a) Airtime

(b) Latency

Figure 4: Learning history of the bandits. The solid line is obtained with a moving average filter with a window of 10 full transmissions and the background shaded error is the standard deviation, computed on the same window.

(less than 20%, see Fig. 1). From Fig. 2, we find that MORE multicast is the best evaluated protocol in airtime. However, we remark that this comes at the cost of being slower and that global knowledge of the topology is required. B.A.T.M.A.N. shows the opposite behavior as it is consistently the least efficient protocol in airtime and the best in latency. The learning algorithms, instead, are often close enough to MORE in airtime and to B.A.T.M.A.N. in latency. Moreover, the BNN based algorithm performs better or equal than the Q-table one in both metrics. For $\bar{d} = 3.8$, for instance, it is 15% worse than MORE multicast considering the median airtime, but it has a 20% higher PDR and is 25% faster. Fig. 3 shows that there exists a linear dependence between the generation size and both airtime and latency, with a protocol-dependent slope. Again, the BNN based algorithm is the closest to MORE in airtime, whereas, in latency, both the learning methods outperform it, being close to B.A.T.M.A.N.

D. Convergence and adaptability

In Fig. 4, the learning history of the bandits is shown, considering a mesh with $\bar{d} = 3.8$. In the experiment performed, 3 random links are deleted at mid simulation, the exploration-exploitation policy is reset, and the replay buffer is emptied. In particular, parameters of concrete dropout are reinitialized, whereas, for what concerns the UCB1 policy, $N_t(s, a)$ is reset to zero for all the state-action pairs, and $c = 0.4$. The ability of the bandits to learn and recover a good solution can be thus evaluated by looking at the performance before and after mid simulation. As can be seen, BNNs can reach a better and stabler solution in terms of airtime, and in a lower amount of time. Contrarily, the tabular bandits minimize the cost only marginally, worrying mostly about the latency threshold. Nonetheless, this is violated in 1.1% of the cases before the links deletion and in 2.2% after by the Q-tables, whereas never before and in 2% of the cases after for the BNN bandits. Notably, the standard deviation of the results, i.e., the stability of the solutions, is greatly affected in latency for the BNNs after the links removal. However, this is because exploration was reset, and, actually, the airtime cost starts again to decrease to find a new minimum. This behavior proves the superiority of the exploration technique implemented for the BNNs. Note, moreover, that this plot also shows the convergence of the algorithm to (at least) a local optimum. Actually, the reward, as explained in Sec. III-D, is the direct output of a map that takes as input the global airtime cost of the network, subject to latency constraints. As can be seen in Fig. 4, both airtime and latency converge to a stable solution even after heavy environment modifications, this being possible if and only if the average reward also converged to a stable value.

V. Concluding remarks

In this work, we presented two multi agent RL frameworks for broadcast routing in WMNs, based on cMABs, to minimize the airtime cost with latency constraints. Specifically, we evolve over the optimal solution for unicast routing to propose a learning framework for the broadcast case. We propose an effective way to compute the reward, so that the agents can learn in a time-varying environment, adjusting their policies. Notably, we explore information exchange among nodes, in such a way to induce cooperation in a fully decentralized environment.

Results show that the BNNs based algorithm performs close to MORE multicast in airtime, while improving in latency. Actually, a better compromise between these two contrasting objectives is reached with the proposed framework, concerning standard reference cases such as MORE and B.A.T.M.A.N. Notably, reliability is significantly improved in terms of PDR with respect to both benchmark protocols, especially in poorly connected networks. Moreover, the framework shows good adaptability to channel variations, as it can recover a good solution after a disruptive event such as link removals.

Although the results look promising, there is certainly room for improvements. Possible future work may involve the transition from a discrete to a continuous control

framework, for instance employing a multi agent actor-critic approach [22]. The Q-value could be learned via federated learning [23] on a higher hierarchy architecture, while the private policy is trained locally at each router. The addition of more and diverse input information could finally be investigated.

References

[1] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM computer communication review*, vol. 35, no. 4. ACM, 2005, pp. 133–144.

[2] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 169–180, Aug. 2007. [Online]. Available: http://doi.acm.org/10.1145/1282427.1282400

[3] T. Clausen and P. Jacquet, " Optimized Link State Routing Protocol (OLSR)," Internet Requests for Comments, IETF, RFC 3626, October 2003. [Online]. Available: https://tools.ietf.org/html/rfc3626

[4] P. Ruiz and A. Skarmeta, "Approximating optimal multicast trees in wireless multihop networks." *Proceedings - IEEE Symposium on Computers and Communications*, pp. 686–691, 01 2005.

[5] C. T. Chou, A. Misra, and J. Qadir, "Low-latency broadcast in multirate wireless mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2081–2091, Nov 2006.

[6] Open Mesh. (2019) Broadcasts in B.A.T.M.A.N. advanced. [Online]. Available: https://www.open-mesh.org/projects/batman-adv/wiki/Broadcast

[7] ETSI, "Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications," *Intelligent Transport Systems (ITS)*, 2011.

[8] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[9] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems*, 2017, pp. 3581–3590.

[10] L. Zhou, "A survey on contextual multi-armed bandits," *arXiv preprint arXiv:1508.03326*, 2015.

[11] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *arXiv preprint arXiv:1906.04737*, 2019.

[12] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*, vol. 41, no. 1, 2003, pp. 11–20.

[13] Z. Zhong and S. Nelakuditi, "On the efficacy of opportunistic routing," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2007, pp. 441–450.

[14] R. Gandhi, A. Mishra, and S. Parthasarathy, "Minimizing broadcast latency and redundancy in ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 840–851, Aug 2008.

[15] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, 1994, pp. 671–678.

[16] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 2016, pp. 25–33.

[17] C. Yu, J. Lan, Z. Guo, and Y. Hu, "Drom: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64 533–64 539, 2018.

[18] X. You, X. Li, Y. Xu, H. Feng, and J. Zhao, "Toward packet routing with fully-distributed multi-agent deep reinforcement learning," *arXiv preprint arXiv:1905.03494*, 2019.

[19] O. Ashour, M. St-Hilaire, T. Kunz, and M. Wang, "A survey of applying reinforcement learning techniques to multicast routing," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 1145–1151.

[20] A. A. Bhorkar, M. Naghshvar, T. Javidi, and B. D. Rao, "Adaptive opportunistic routing for wireless ad hoc networks," *IEEE/ACM Transactions On Networking*, vol. 20, no. 1, pp. 243–256, 2011.

[21] K. Tang, C. Li, H. Xiong, J. Zou, and P. Frossard, "Reinforcement learning-based opportunistic routing for live video streaming over multi-hop wireless networks," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2017, pp. 1–6.

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[23] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.