Kom8ndor: An IEEE 802.11bn-Oriented Simulator for Wi-Fi 8 and Beyond

Francesc Wilhelmi
Engineering Department
Universitat Pompeu Fabra
Barcelona, Spain
francisco.wilhelmi@upf.edu

Sergio Barrachina-Muñoz
Services As NetworkS
Centre Tecnològic de Telecomunicacions de Catalunya
Barcelona, Spain
sergio.barrachina@cttc.cat

Boris Bellalta
Engineering Department
Universitat Pompeu Fabra
Barcelona, Spain
boris.bellalta@upf.edu

***Abstract*—The upcoming IEEE 802.11bn amendment marks a paradigm shift in Wi-Fi, which will pose ambitious performance targets under the paradigm of Ultra-High Reliability (UHR). To understand the implications of such a new technology and to support early research and protocol design for Wi-Fi 8, we present Kom8ndor. This discrete-event network simulator extends the open-source Komondor platform (a simulator validated against ns-3 and other analytical tools) with 802.11bn features. Among the newly added functionalities, we highlight Multi-Access Point Coordination (MAPC)—including Coordinated Time-Division Multiple Access (Co-TDMA), Coordinated Spatial Reuse (Co-SR), and Coordinated Beamforming (Co-BF)—, Non-Primary Channel Access (NPCA), and Dynamic Subband Operation (DSO). Beyond Wi-Fi 8 implementations, Kom8ndor introduces novel functionalities (e.g., a machine learning wrapper for building AI-based protocols) and a modular design to boost the prototyping and research of future Wi-Fi technologies. Kom8ndor is open-source (GNU GPLv3) and available at https://github.com/wn-upf/Komondor.**

***Index Terms*—IEEE 802.11bn, Wi-Fi 8, Network simulator, Ultra-High Reliability**

I. Introduction

With the upcoming IEEE 802.11bn-2028 (Wi-Fi 8), Wireless Local Area Networks (WLANs) are set to go beyond supporting mere Internet access, and step toward communications-critical applications like industrial automation [1]. To meet Ultra-High Reliability (UHR) goals, 802.11bn will introduce new functionalities, such as Non-Primary Channel Access (NPCA), Dynamic Subband Operation (DSO), or Multi-Access Point Coordination (MAPC). Given the unprecedented enhancements and architectural changes that 802.11bn proposes over 802.11, the need for open-source simulation tools becomes critical, as they enable a thorough evaluation of both standard functionalities and novel extensions (e.g., beyond 802.11bn MAPC) in various scenarios and use cases.

ns-3 [2] remains the most popular choice, offering high-fidelity, full-stack simulation with a mature set of Wi-Fi 7 features (e.g., Multi-Link Operation (MLO), Multi-User (MU)-Resource Unit (RU) Orthogonal Frequency-Division Multiple Access (OFDMA)), while OMNeT++/INET [3] provides high modularity and realistic emulation, and MATLAB's WLAN Toolbox [4] excels at high-fidelity link-level and Physical (PHY) modeling (including some Wi-Fi 8 PHY models). All three, however, follow a slow but necessary development pace that limits rapid prototyping of fast-evolving standards, pushing early Wi-Fi research efforts toward narrow-scope (sometimes closed-source) tools [5]–[8]. Regarding 802.11bn, very few works provide simulation results given the technology's early status (currently Draft 2.0), and existing examples are either not standard-compliant and tightly coupled to specific proposals [9] or not open-sourced [10]. To the best of our knowledge, no publicly available simulator offers Wi-Fi 8 and beyond functionalities within an extensible framework (see Table I).

TABLE I: High-level comparison of Wi-Fi simulation tools.

Tool	Open-source	Main strength	Wi-Fi 8	Future Wi-Fi
ns-3	✓	Full-stack, wide adoption.	✗	✗
MATLAB's WLAN Toolbox	✗	High-fidelity PHY modeling.	✓[a]	✗
OMNeT++/ INET	✓	High modularity, realistic emulation.	✗	✗
Kom8ndor	✓	Fast event processing, rapid prototyping, AI/ML integration.	✓[b]	✓

[a]Includes Wi-Fi 8 PHY models (e.g., new MCSs), [b]Built explicitly to evaluate MAPC and other Wi-Fi 8 features.

For this reason, we introduce Kom8ndor, a C/C++ 802.11bn-oriented event-driven network simulator that substantially extends the original Komondor simulator [11] (originally meant for Wi-Fi 6 simulations [12], [13]) to introduce Wi-Fi 8 and beyond features, including a new MAPC framework alongside DSO, NPCA, and new channel access methods. Komondor serves as a well-suited simulation platform for Wi-Fi research, offering a balance between implementation speed and simulation fidelity that more comprehensive frameworks like ns-3 are less suited to provide for rapidly evolving standards, while retaining a lightweight, high-throughput Medium Access Control (MAC)/PHY abstraction well suited to rapid

This work was supported by the following projects: TRUE Wi-Fi PID2024-155470NB-I00 (MICIU/AEI/10,13039/501100011033/FEDER,UE), MLDR (Chist-ERA WAI 2022) PCI2023-145958-2 (MCIU/AEI/10.13039), ICREA Academia 2024 (00077 AGAUR), and MdM CEX2021-001195-M (MICIU/AEI/10.13039/501100011033).

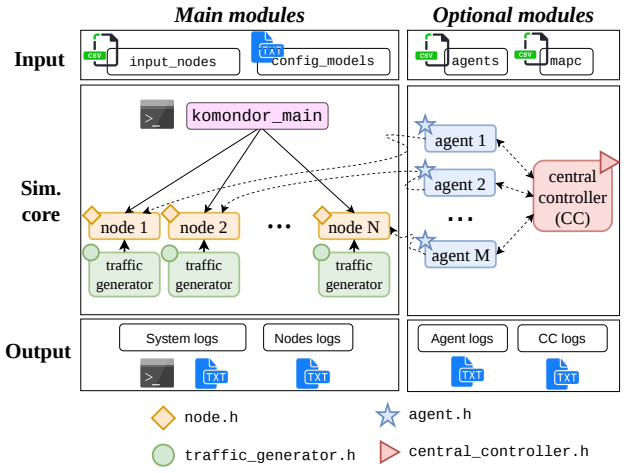


Fig. 1: Kom8ndor's main components and simulation flow.

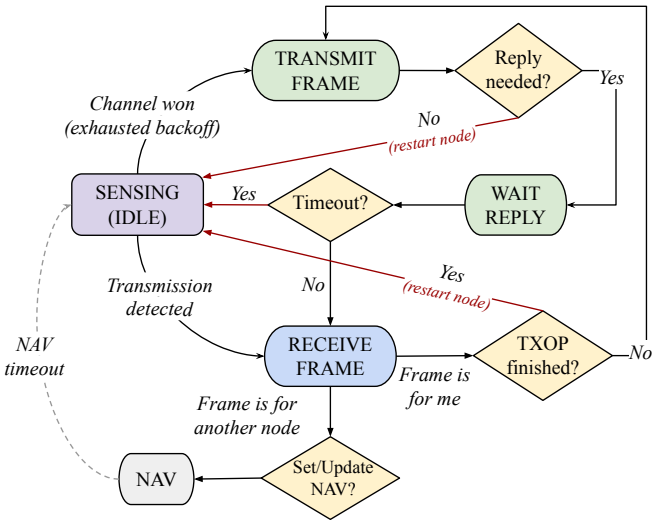


Fig. 2: Kom8ndor's generic FSM.

prototyping. Moreover, Kom8ndor extends the Machine Learning (ML) modules from Komondor by introducing a Python wrapper, which is key to accelerating the experimentation of AI-native solutions for future Wi-Fi.

II. Architecture and Design

Komondor was initially designed for cost-effective simulation of high-density Wi-Fi 6 scenarios, featuring novel features at that time, such as Overlapping Basic Service Set/Packet Detect (OBSS/PD)-based Spatial Reuse (SR) [12] or Dynamic Channel Bonding (DCB) [13]. Kom8ndor extends the original architecture to introduce Wi-Fi 8 and beyond features, higher modularity for forward extensions and integration of third-party ML tools, higher efficiency, and better usability.

A. Main simulation components

Kom8ndor uses the C/C++ COST library [14], which allows the creation of COST components and handles discrete-event interactions among them. The main file (komondor_main) inherits a *CostSimEng* singleton (a special type of component defined by COST) that is in charge of instantiating all the simulation components, creating the connections between them via *inport* and *outport* methods, and running the simulation. *TypeII* components (native to COST) are used to represent different types of entities in a Wi-Fi network, including:

- node: Characterizes the logic of either Access Point (AP) or Station (STA) devices, including traffic management, MAC, and PHY operations.
- traffic_generator: Generates traffic based on a selected traffic model (e.g., Poisson traffic, burst traffic, full-buffer). Every transmitting node is connected to a dedicated traffic generator.
- agent: Provides in-simulation ML operations to optimize the performance of the network online. Agents are connected to APs to exchange information and commands.
- central_controller: Orchestrates enrolled agents (e.g., centralized decision-making), thus unlocking centralized ML and non-ML solutions and algorithms.

The different components are flexibly instantiated based on a set of input files, comprising:

- input_nodes: defines per-node capabilities (e.g., position, channel, transmission power).
- config_models: defines environment models (e.g., path loss, co-channel interference model) applied to the simulated scenario.
- agents (optional): defines ML-related implementations, ranging from Reinforcement Learning (RL)-based dynamic parameter configuration to custom models interacting with Kom8ndor through a Python server.
- mapc (optional): defines coordination groups, MAPC schemes, and configurations to be used by the instantiated Wi-Fi nodes.

Kom8ndor outputs include simulation summaries (e.g., per-Basic Service Set (BSS) performance, per-node statistics, collision counts), and detailed logs for nodes, agents, and the central controller. The overall system is illustrated in Fig. 1.

B. Finite State Machine

Kom8ndor simulations are based on discrete events, following a flexible Finite State Machine (FSM), where node states represent frame transmissions (e.g., TRANSMIT DATA), frame receptions (e.g., RECEIVE DATA), and other operations (e.g., WAIT DATA, NAV), and transitions between states follow operations such as channel contention, backoff counters, or inter-frame spacing (see Fig. 2). First, when a traffic_generator pushes data to a node, the FSM transitions through frame preparation, medium sensing, and frame transmission, scheduling events for the exact microsecond each action is expected to be performed. Accordingly, if a node has a frame to transmit, it undergoes a backoff procedure, which can run as long as the medium is sensed as idle (the

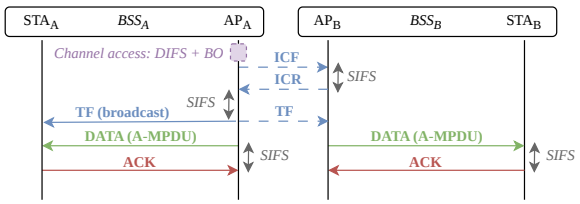


Fig. 3: Co-SR/Co-BF frame exchange sequences.

node is in SENSING state). Upon channel access, the node transitions to the TRANSMIT FRAME state, which triggers *transmission detected* events in other nodes that can eventually activate the Network Allocation Vector (NAV) procedure and make them transition to NAV state (depending on the spatial geometry of the scenario). Depending on the need for a reply or not (e.g., acknowledgment (ACK), Clear-to-Send (CTS)), the transmitting node transitions through WAIT REPLY and then RECEIVE FRAME, or goes back to SENSING upon finishing the transmission, respectively.

This FSM design provides the necessary flexibility to realize various transmission modes, ranging from simple data and ACK exchanges to more complex MAPC frame sequences. This flexibility enables seamless extensibility, allowing new transmission approaches and protocols to be easily developed. Figure 3 shows an example of a frame sequence for Coordinated Spatial Reuse (Co-SR)/Coordinated Beamforming (Co-BF).

To support MAPC features, Kom8ndor includes new states and signaling that involve multiple nodes, including Initial Control Frame (ICF)/Initial Control Response (ICR) exchange (to agree on a coordinated transmission), the transmission of a Trigger Frame (TF) (used by Co-SR and Co-BF to synchronize simultaneous data transmissions across multiple APs), or Multi-User Request-to-Send (MU-RTS) (used by Coordinated Time Division Multiple Access (Co-TDMA) to schedule time slots among coordinated APs). Depending on the selected MAPC scheme, the FSM is adapted accordingly, and new transient states are introduced to manage the chronological flow of the coordination protocols. For a detailed overview of the MAPC schemes, we refer the reader to the tutorial in [15].

III. New Features: From 802.11bn to AI/ML

A. Multi Access Point Coordination

MAPC is a new feature in 802.11bn that allows a *coordinating AP* (the AP that wins the channel access) to share its Transmit Opportunitiess (TXOPs) with other *coordinated APs* and, in some cases, perform simultaneous transmissions. Current MAPC schemes comprise Co-TDMA (schedules multiple transmissions within the same TXOP), Co-SR (enables simultaneous transmissions with limited transmit power), Co-BF (enables simultaneous transmissions using beamforming and nulling), Coordinated R-TWT (Co-RTWT) (coordinates Restricted Target Wake Time (R-TWT) schedules), and Coordinated Channel Recommendation (Co-CR) (coordinates the channels used for device-to-device transmissions).

Kom8ndor introduces a MAPC framework that builds upon the baseline established by 802.11bn while enabling extensions to anticipate beyond Wi-Fi 8 capabilities. For instance, while 802.11bn restricts MAPC agreements to a maximum of two APs, Kom8ndor facilitates the efficient evaluation of scalability beyond this limit. Kom8ndor includes the MAPC schemes that operate on a per-TXOP basis, i.e., Co-TDMA, Co-SR, and Co-BF.

As described in Section II, Kom8ndor initializes the simulation using input files. For MAPC, agreements and parameters are provided in a new configuration file (e.g., mapc.csv). This file is used to pre-establish the logical coordination groups and per-scheme parameters, which depend on the scheme itself (e.g., time-split approach in Co-TDMA). During the simulation, Kom8ndor performs the new handshakes needed to coordinate shared TXOPs, for which new control frames were defined (ICF, ICR, TF, MU-Request-to-Send (RTS)). Accordingly, the FSM is updated to support new states such as TRANSMIT ICF, WAIT ICR, or WAIT MU-RTS. These states ensure that participating nodes remain synchronized and that non-participating STAs correctly update their NAV to defer access until the end of the coordinated TXOP.

1) Co-TDMA Operation: The coordinating AP splits the TXOP into orthogonal time slots, assigned to different participating APs. After the ICF-ICR exchange, a fair TXOP split is decided among coordinated APs, followed by the sequential data transmissions. By default, Kom8ndor splits the resources equally among coordinated APs. In particular, the per-AP data duration T_n for each participating AP $n \in \mathcal{N}$ is capped equally. While not currently included, other policies, such as Quality of Service (QoS)-based scheduling or maximum-delay minimization, can be implemented.

2) Co-SR Operation: In Co-SR, coordinated APs agree on the transmit power to be used during a simultaneous Aggregate MAC Protocol Data Unit (A-MPDU) downlink transmission. In the default implementation, the power to be used by the coordinated AP is indicated by the user (indicated in the mapc input file), while the coordinating AP employs its predefined transmit power (indicated in the nodes input file). However, other solutions can be implemented on top, including a smart power assignment based on the maximum allowable interference at each communication end.

3) Co-BF Operation: In Co-BF, two APs transmit simultaneously thanks to coordinated inter-BSS interference suppression, where each AP steers a beam toward its own associated STA and places nulls in the directions of the recipient STAs of the other BSS. Kom8ndor's implementation of Co-BF models each AP as a horizontal Uniform Linear Array (ULA) of N isotropic elements[1] and computes beamforming weights via Zero Forcing (ZF) precoding, which enforces unity gain toward the desired STA while placing exact nulls at all K peer STAs, solving the resulting $(K+1) \times (K+1)$ Gram system via Gaussian elimination with partial pivoting. The resulting

[1]The ULA parameters (N, d, and nominal azimuth) are optionally informed in the input_nodes file.

beam gain then scales the received power relative to the default omnidirectional model. Beamforming gains are applied to data frames only, while all control frames (ICF/ICR, TF, RTS/CTS, ACK) are transmitted under the omnidirectional model.

B. Flexible Spectrum Access: DSO, NPCA, and More

`Kom8ndor` includes two new spectrum access features: DSO and NPCA. In addition, Komondor's Dynamic Channel Bonding (DCB) baseline is extended [13].

1) Dynamic Subband Operation (DSO): DSO tries to fill the increasing capability gap between AP and STA devices, where APs might use wide channels (e.g., 160 MHz or 320 MHz), while STAs might have more moderate capabilities (e.g., 20 MHz or 40 MHz). With DSO, narrow-band STAs can be allocated to secondary subchannels during the same TXOP in a wide-band transmission, hence increasing efficiency. For example, without DSO, a 160 MHz AP serving four 40 MHz STAs would use only 40 MHz per TXOP, wasting 120 MHz each time. With DSO, in contrast, the four STAs could be scheduled in a single TXOP. `Kom8ndor` introduces DSO by simulating the mandatory ICF/ICR exchange before the data transmission begins. After that, all the scheduled STAs transmit on their assigned portion of the bandwidth. Currently, the set of scheduled STAs in a DSO transmission is decided using round-robin, allocating in order the selected STAs based on their declared bandwidth (specified as `min_ch`/`max_ch` in the `input_nodes` file).

2) Non-Primary Channel Access (NPCA): NPCA aims to overcome the contention aspects in an Overlapping Basic Service Set (OBSS) by allowing an AP to temporarily translate its operation to a different channel. When an NPCA node detects an inter-BSS transmission, it switches to a pre-configured NPCA primary channel that is within the total BSS bandwidth. There, the node can run a new backoff (as if it was the primary) and eventually initiate a transmission. At the end of the TXOP, the node must return to the primary channel. `Kom8ndor` implements NPCA by capturing its sequence of events. First, the detection of an OBSS transmission triggers the NPCA channel switching (a fixed configurable radio switching delay can be added), where a new backoff is run. Then, a transmission can occur on the NPCA primary channel, which must be preceded by an ICF/ICR exchange. Regardless of whether a transmission is performed or not on the NPCA primary channel, an NPCA trigger determines the time at which the node returns to its primary channel. The NPCA primary channel can be flexibly defined in `Kom8ndor`, but it defaults to the upper portion of the BSS bandwidth.

3) Preamble Puncturing: Based on the DCB implementation in Komondor, `Kom8ndor` adds *preamble puncturing*, which was introduced in IEEE 802.11ax. In preamble puncturing, the TXOP remains anchored to the primary subband, but spectrum is aggregated more flexibly. In particular, any individual busy secondary 20 MHz channels are excluded from the set of channels bonded for a data transmission (achieved by marking the busy basic channels in a puncturing bitmap), making the most of the available bandwidth.

C. Advanced Channel Access Methods

Channel access has traditionally been an object of discussion in Wi-Fi. The standard Distributed Coordination Function (DCF) relies on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with Binary Exponential Backoff (BEB), whereby nodes initiate transmissions upon successfully decrementing a random backoff while the channel is idle; despite its simplicity and long adoption, DCF inefficiencies are well known, especially as the number of coexisting nodes increases. Beyond baseline DCF, `Kom8ndor` supports Enhanced Distributed Channel Access (EDCA) [16], today's baseline mechanism, which extends DCF with QoS differentiation across four Access Categories (ACs)—Voice (VO), Video (VI), Best Effort (BE), Background (BK)—and is fully configurable per node and AC to set the Arbitration Interframe Space (AIFS), Contention Window (CW) bounds, and TXOP limit. It also supports several determinism-oriented alternatives: *deterministic backoff* [17], which fixes $BO = b + \text{IPT}$ (a base value plus the number of backoff interruptions suffered) and reverts to random backoff only after collisions; It's Your Turn (IYT) [18], a distributed round-robin scheme where a logical token is passed to the next node in a shared transmission-order list after each success; and simpler heuristic baselines such as Enhanced Collision Avoidance (ECA) [19] (repeats the same backoff after a success), Repeat Backoff, and Synchronized Backoff (fixed, non-adaptive backoff), useful as analytical references for studying the limits of channel access.

D. Machine Learning Tools

Since its first release in 2018, Komondor has included Artificial Intelligence (AI) modules for studying next-generation WLANs. Specifically, `agent` and `central_controller` components were introduced to realize RL algorithms to dynamically optimize operations such as channel allocation, SR, or DCB. Now, `Kom8ndor` has improved this design and has added, on top, a seamless integration with Python, which is key to leveraging third-party ML packages such as PyTorch [20].

1) Built-in Learning Algorithms: `Kom8ndor` includes built-in learning functions that allow the dynamic configuration of Wi-Fi PHY and MAC parameters during the simulation time. In particular, a single dispatch point handles all the agent decisions. Currently, a Multi-Armed Bandit (MAB) module is implemented to enable online decision-making, in which the agent maintains empirical reward estimates for each arm (e.g., a channel configuration) and selects among them via a configurable exploration strategy (e.g., ε-greedy). Other implementations are easily pluggable, since algorithmic differences between RL paradigms (e.g., model-based vs model-free) are hidden from the rest of the simulator. With this architecture, different learning paradigms can be flexibly instantiated: *i) Decentralized* (agents perform decision-making based on local information only [21]), *ii) Coordinated* (multiple agents exchange information to perform decision-making cooperatively [22]), and *iii) Centralized* (a central controller gathers global state information from agents and performs centralized decision-making).

TABLE II: Kom8ndor simulation flags.

	Flag	Arg	Description
Sim. params.	--nodes (-n)	file	Path to nodes file
	--time (-t)	float	Simulation duration in seconds
	--seed (-s)	int	Random seed
	--code (-c)	str	Simulation code/ID
	--out (-o)	file	Path to output file
Advanced	--mapc (-m)	file	Path to MAPC file
	--agents (-a)	file	Path to agents file
Logging	--logs-sys (-L)	0/1	Print system events to terminal
	--logs-node (-l)	0/1	Print per-node events to terminal
	--save-node (-S)	0/1	Write per-node logs to file
	--save-agent (-A)	0/1	Write per-agent logs to file

2) External Python Integration: To run complex ML paradigms such as Deep Neural Networks (DNNs) and leverage existing libraries such as PyTorch, TensorFlow, or scikit-learn, Kom8ndor introduces a new agent mode that offloads decisions to an external Python process. With this, agents can send a feature vector at each step or iteration to inform about the played action index, the current configuration, or the observed reward (performance). While the current implementation remains simple, it can be easily extended to include richer state representations (e.g., channel measurements, queue status information). The communication between the simulator and the Python process uses a Portable Operating System Interface (POSIX) Unix-domain socket. In particular, in each iteration, Kom8ndor sends a 32-bit integer header specifying the number of features, followed by a vector of 32-bit floats with the values of those features. The Python server returns a vector of 32-bit floats encoding the ML model output (e.g., next action).

IV. Tutorial & Performance Showcase

Kom8ndor is a command-line simulator that runs from the terminal or through multi-scenario shell scripts. Running Kom8ndor requires a C++ compiler (g++ with C++98 support) and a Linux or macOS environment (or WSL on Windows) to run the COST code-generation step. A *Makefile* is included in Kom8ndor to automatically build the source code (including the necessary COST libraries) and generate the binary komondor_main that runs the simulation. The *Makefile* also tracks all **.cc* and **.h* files as dependencies, so any change to any header (including the ones in Code/methods/) triggers a full rebuild. To build the code, one must run

Build the code

```
cd Code/main && make
```

Simulations are launched by invoking komondor_main with the necessary flags to specify the scenario and running features (--nodes, --mapc, --agents) together with simulation parameters (--time, --seed, --code). Log verbosity is controlled via --logs-sys and --logs-node, and output results are written to a generic .txt file (--out) and individual per-node and per-agent files (--save-node, --save-agent).[2] The complete list of Kom8ndor flags is provided in Table II.

A. Running MAPC Schemes

We consider an example of a simulation involving MAPC schemes, where two overlapping BSSs (A and B) share the same 40 MHz channel. We run four different cases: *i)* DCF, *ii)* Co-TDMA (an equal time split between APs is considered), *iii)* Co-SR (the coordinating AP uses full power while the coordinated AP power is limited to 15 dBm), and *iv)* Co-BF (an 8-element ULA is considered for beamforming).[3] The following commands are used to run these scenarios:

Running MAPC examples

```
# DCF only
./komondor_main --n ../input/examples/mapc_
example/input_nodes_mapc_example.csv
--t 100 --s 1

# Scenario with MAPC (Co-TDMA example)
./komondor_main --n ../input/examples/mapc_
example/input_nodes_mapc_example.csv --m
../input/examples/mapc_example/mapc_cotdma_
example.csv --t 100 --s 1
```

The resulting throughput per BSS and scheme, which were extracted from Kom8ndor logs, are plotted in Fig. 4. In this simple scenario, Co-TDMA leads to slightly lower throughput than DCF because of the overheads associated with the coordination. Co-SR increases the throughput further thanks to the geometric position of nodes in this deployment, which favors space reutilization. Finally, Co-BF shows the greatest improvement due to the Signal-to-Interference-plus-Noise Ratio (SINR) gains that result from coordinated nulling.

B. Running NPCA and DSO

We consider a single deployment with two BSSs to show DSO and NPCA operation. In particular, BSS A uses a 40 MHz channel and has two associated 20-MHz STAs with DSO enabled. BSS B, instead, uses 80 MHz, with the first 40 MHz portion overlapping with BSS A's bandwidth. To run these scenarios, the following commands are used:

[2]Saving logs entails high simulation times, particularly when simulating large deployments for a long time.

[3]The different input files used across this section can be found in folder https://github.com/wn-upf/Komondor/tree/master/Code/input/examples.

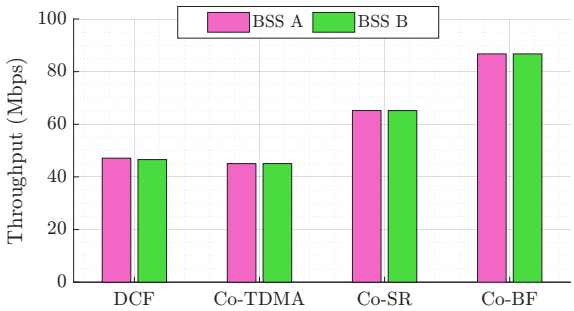


Fig. 4: Throughput achieved per STA in the MAPC example.

Running DSO and NPCA

```
# DCF
./komondor_main --n ../input/examples/
dso_npca_example/input_nodes_dcf.csv
--t 100 --s 1

# DSO and NPCA
./komondor_main --n ../input/examples/
dso_npca_example/input_nodes_dso_npca.csv
--t 100 --s 1
```

The resulting throughput is shown in Fig. 5: The two BSSs get the same aggregate throughput (equally split between the two STAs in BSS A). With DSO (BSS A) and NPCA (BSS B), significant gains are observed, which result from the higher flexibility of the two BSSs in accessing the medium, where STA $A1$ and $A2$ can be served simultaneously by AP A on channels #0 and #1, respectively, and BSS B can access an unused portion of the spectrum (channels #2 and #3).

C. Running ML Agents with Python

When running the newly added external ML models (indicated as `learning mechanism=7` in the `agents` input), the agent delegates arm selection (or other tasks) to an external Python server. The following example (`ml_server_random.py`) simply selects random actions:

Running ML with Python

```
# Terminal 1: start the server first
python3 Code/learning_modules/python_servers/
ml_server_random.py /tmp/komondor_ml.sock

# Terminal 2: run the simulation (Code/main)
./komondor_main --n ../input/examples/
external_model_example/input_nodes.csv
--a ../input/examples/external_model_example/
agents_external.csv --t 100 --s 1
```

Beyond MAB algorithms, the external server enables countless Python implementations (`ml_server_pytorch.py` uses a pre-trained DNN for inference). The same interface naturally supports algorithms like Deep Q-Network (DQN), turning `Kom8ndor` into an RL gym like [23].

V. Conclusions and Future Work

`Kom8ndor` is a live project. It is the basis for simulation-driven research on Wi-Fi 8 and beyond in the Wireless Networking research group at Universitat Pompeu Fabra (UPF). In this paper, we present the first release of Wi-Fi 8, outlining its main features so far: MAPC, NPCA, DSO, and the integration of ML tools. However, this does not stop here, and more work is in the pipeline. First, relevant features from previous generations of Wi-Fi, e.g., MLO, OFDMA, or R-TWT, will be included to study their interplay with Wi-Fi 8 functionalities. In addition, beyond Wi-Fi 8 MAPC solutions will be progressively included and evaluated, such as a dynamic transmit power computation for Co-SR, QoS-based Co-TDMA scheduling, or smart MAPC scheme selection.

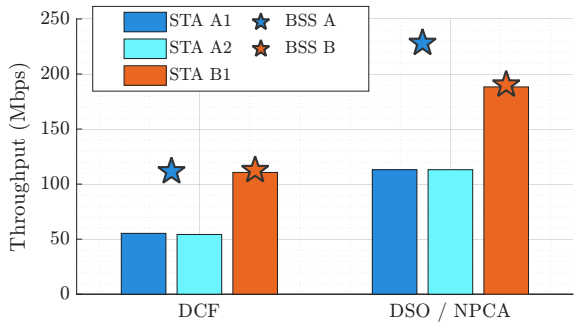


Fig. 5: Throughput achieved per STA/BSS in the DSO & NPCA example.

References

[1] G. Geraci *et al.*, "Wi-Fi: 25 Years and Counting," *Proc. IEEE*, 2026.
[2] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.
[3] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *SIMUtools*, 2008, pp. 1–10.
[4] MathWorks, "WLAN Toolbox," R2026a. [Online]. Available: https://www.mathworks.com/products/wlan.html
[5] M. Carrascosa-Zamacois *et al.*, "Performance evaluation of MLO for XR streaming: Can Wi-Fi 7 meet the expectations?" in *CAMAD*. IEEE, 2024, pp. 1–6.
[6] M. Alsakati *et al.*, "Performance of 802.11 be Wi-Fi 7 with multi-link operation on AR applications," in *WCNC*. IEEE, 2023, pp. 1–6.
[7] A. Jeknić and E. Kočan, "Development steps that brought to Wi-Fi 7," *ETF Journal of Electrical Engineering*, vol. 29, no. 1, pp. 65–79, 2023.
[8] C. Chen *et al.*, "Overview and performance evaluation of Wi-Fi 7," *IEEE Commun. Standards Mag.*, vol. 6, no. 2, pp. 12–18, 2022.
[9] D. Nunez *et al.*, "Enabling reliable latency in Wi-Fi 8 through multi-AP joint scheduling," *OJCOMS*, 2025.
[10] X. Liu *et al.*, "Wi-Fi 8: Embracing the millimeter-wave era," *IEEE Commun. Mag.*, vol. 63, no. 3, pp. 69–75, 2024.
[11] S. Barrachina-Munoz *et al.*, "Komondor: A wireless network simulator for next-generation high-density WLANs," in *Wireless Days*. IEEE, 2019, pp. 1–8.
[12] F. Wilhelmi Roca *et al.*, "Spatial Reuse in IEEE 802.11 ax WLANs," *Comp. Comm. 2021, 170: 65-83*, 2021.
[13] S. Barrachina-Munoz, F. Wilhelmi, and B. Bellalta, "Dynamic channel bonding in spatially distributed high-density WLANs," *IEEE Trans. Mobile Comp.*, vol. 19, no. 4, pp. 821–835, 2019.
[14] G. Chen and B. K. Szymanski, "Cost: A component-oriented discrete event simulator," in *Proc. of the Winter Simulation Conf.*, vol. 1. IEEE, 2002, pp. 776–782.
[15] F. Wilhelmi *et al.*, "A Tutorial on IEEE 802.11bn Multi-AP Coordination for Wi-Fi 8: From Standardization to Performance Evaluation," *arXiv preprint arXiv:2606.13759*, 2026.
[16] B. Bellalta *et al.*, "Modeling the IEEE 802.11 e EDCA for MAC parameter optimization," in *Het-Nets*, 2006.
[17] I. Tinnirello *et al.*, "A Deterministic Backoff Approach for Wi-Fi and NR-U Coexistence in Shared Bands," *IEEE Trans. Mobile Comp.*, 2025.
[18] F. Wilhelmi, L. Galati-Giordano, and G. Fontanesi, "It's Your Turn: A Novel Channel Contention Mechanism for Improving Wi-Fi's Reliability," in *WCNC*. IEEE, 2025, pp. 1–6.
[19] L. Sanabria-Russo *et al.*, "A high efficiency MAC protocol for WLANs: Providing fairness in dense scenarios," *IEEE/ACM Trans. on Networking*, vol. 25, no. 1, pp. 492–505, 2016.
[20] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.
[21] F. Wilhelmi *et al.*, "Decentralized Spatial Reuse Optimization in Wi-Fi: An Internal Regret Minimization Approach," in *EuCNC/6G Summit*. IEEE, 2026, pp. 254–259.
[22] F. Wilhelmi *et al.*, "Coordinated multi-armed bandits for improved spatial reuse in Wi-Fi," in *ICMLCN*. IEEE, 2025, pp. 1–6.
[23] P. Gawłowicz and A. Zubow, "Ns-3 meets openai gym: The playground for machine learning in networking research," in *MSWiM*, 2019, pp. 113–120.