Komondor: An Event-Based Wireless Network Simulator for Next-Generation IEEE 802.11ax WLANs

Sergio Barrachina and Francesc Wilhelmi

December 20, 2017

Abstract

Komondor is a wireless network simulator that includes novel mechanisms for next-generation WLANs, such as Dynamic Channel Bonding or enhanced Spatial Reuse. One of Komondor's main purposes is to emulate the behavior of IEEE 802.11ax-2019 networks, which main challenge is spectral efficiency in dense deployments. Furthermore, due to the growing popularity of autonomous systems and the tendency of WLANs to use learning, Komondor is intended to include intelligent agents that make decisions that allow enhancing the network performance. In this document we provide an overview of the Komondor simulator, making insight on its main features, its operational mode and its development stages. Komondor has been developed to be an open source tool that contributes to the ongoing research in wireless networks. For that, all the contents are published at the following public repository: `https://github.com/wn-upf/Komondor`. Any interested researcher is invited to collaborate.

Contents

List of Figures

List of Tables

1 Introduction

Komondor [1] is an event-based simulator based in COST [2], a CompC++ library that allows generating discrete event simulations.[1] Komondor is mostly intended to reproduce the novel techniques included in the IEEE 802.11ax-2019 amendment [3], which is called to become a benchmark in next-generation wireless networks. Furthermore, due to the increasing popularity of learning-based approaches in WLANs, our simulator is being built to allow the inclusion of intelligent agents that make decisions in simulation time. The decision of building a new wireless networks simulator is based on *i*) the lack of 11ax-oriented simulators that include novel techniques, *ii*) the need of generating a tool able to simulate intelligent agents behavior, *iii*) the difficulty in extending other existing solutions (e.g., ns-3) towards our abovementioned main goals.

1.1 Next-Generation WLANs

The increasing popularity of IEEE 802.11 WLANs has led to new strict requirements in terms of data rate and users capacity. Such situation has brought the wireless communications community to introduce novel approaches. In particular, the 11ax amendment is being developed to improve spectrum efficiency in high density scenarios. To accomplish that, it introduces the concept of High-Efficiency (HE) WLANs, which incorporates new mechanisms such as OFDMA, Dynamic Channel Bonding, Beamforming and Multi-User Multiple-Input Multiple-Output (MU-MIMO) [4]. Such advanced mechanisms drastically change the current operation of WLANs and have not been previously implemented with detail in other network simulators.

In addition to the novel HE techniques, wireless networks are evolving towards autonomous management, which in many cases is achieved through Artificial Intelligence (AI). Its utilization is expected to be key in next-generation complex systems, since it allows solving (or at least approximating) computational-intensive problems. In particular, online learning has been previously applied in well-known problems such as Transmit Power Control (TPC), Carrier Sense Threshold (CST) adjustment and channel allocation [5, 6, 7, 8]. Since most of the literature that applies learning into wireless networks is theoretical in nature, there is a strong need of tools that allow implementing learning algorithms in a realistic simulation environment.

1.2 Komondor Main Features

Komondor aims to thoroughly simulate the operation of wireless networks. Henceforth, it reproduces actual transmissions on a per-packet basis. For that, nodes properties (e.g., location, transmit power, CCA threshold) are taken into account during data exchange procedures. In overview mode, Komondor provides the following functionalities:

- Flexible input files processing with error control: network capabilities can be introduced into the simulator in a very flexible manner. Moreover, an input checker is provided in order to identify the most prominent errors in the input provided by the user.
- IEEE 802.11ax WLANs features:
  - Distributed Coordination Function (DCF): the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) captures the basic Wi-Fi operation for accessing the channel. Moreover, Contention Window (CW) adaptation is considered.
  - Channel Bonding (CB): several channel ranges can be selected during transmissions in order to maximize the spectrum efficiency.
  - Packet aggregation: several MPDUs can be aggregated into the same PPDU in order to reduce the generated communication overheads.
  - Dynamic Modulation Coding Scheme (MCS): the MCS is negotiated between any transmitter-receiver pair according to the Signal-to-Interference-and-Noise Ratio (SINR).
  - Ready-to-Send/Clear-to-Send (RTS/CTS) and Network Allocation Vector (NAV): nodes exchange packets before transmitting in order to allocate the channel and prevent collisions.
- Statistics gathering and output generation

[1] COST main website: http://www.ita.cs.rpi.edu/cost.html

Future development stages are considered to include other features such as OFDMA, MU-MIMO transmissions, beamforming, or dynamic transmit power and CST adjustment.

1.3 COST

In order to provide a deeper understanding of Komondor, it is important to briefly understand the COST library, which allows building discrete interactions between components (e.g., wireless nodes). Such interaction is achieved through synchronous and asynchronous events. While the former are messages explicitly exchanged between components through input/output ports, the later are based on timers. In practice, components perform a set of operations until a significant event occurs. For instance, a node that is decreasing its backoff (i.e., current operation) may freeze it when an overlapping node occupies the channel (i.e., an event). Moreover, the node may start a transmission when the backoff timer is over (i.e., a trigger). Figure 1 shows the schematic of a COST component, which is characterized by its inports and outports, and a set of timers.

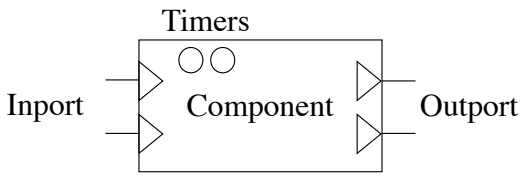


Figure 1: COST component. While inports and outports allow to directly communicate with other components, timers trigger events specific to the component.

1.4 Contributions

In this document we describe the main features of the Komondor simulator and system model considerations, as well as some basic guidelines to run it. The main contributions done in this document are listed below:

- We provide an overview of the first version of the Komondor simulator.
- We describe the implementation done for each of the included functionalities, as well as the main considerations done.
- We provide a set of validations that ensure the proper operation of the presented simulator.
- We provide a tutorial to allow an easy installation and execution of the simulator. Moreover, some implementation details are granted in case the reader is interested in extending the simulator.

Note, as well, that detailed technical information regarding code development is out of the scope of this document.

1.5 Document Structure

This documents is structured as follows: Section 2 describes the system model. Then, Section 3 defines the implementation of MAC functionalities considered so far, which are validated in Section 4. A tutorial and some development notes are provided in Section 5. Finally, some remarks are given in Section 6.

2 System Model

One of the main tasks regarding the implementation of Komondor relies on defining the system model, which lays the foundations of the simulator. In this Section we describe the models defined for simulating the different communication aspects that a wireless device is intended to implement.

2.1 Channel Modeling

Channel modeling refers to medium in which wireless devices must interact each other. In order to be flexible and allowing to simulate different scenarios and casuistic, we define a set of path-loss and co-channel interference models, which are described next.

2.1.1 Path-loss Models

In order to provide the most representative wireless environments, we implemented the following set of path-loss models, which can be further extended by any developer:

- Free Space Path Loss (FSPL): a free-space model is considered, which captures direct line-of-sight and ignores shadowing effects. The experienced loss of power during a transmission that assumes this model is given by:

$$\text{FSPL} = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}(\frac{4\pi}{c}) - G_t - G_r,$$

  where d is the distance between the transmitter and the receiver, f is the frequency used in GHz, c is the light speed in m/s, and G_t and G_r are the gains in dB at the transmitter and the receiver, respectively.

- Okumura-Hata model [9]: this well-known model was conceived for predicting the path-loss of cellular transmissions in outside urban and rural environments. Since our main concern is related to dense urban areas, let the loss, L_U, be given by:

$$\begin{aligned} L_U =& 69.55 + 22.16\log_{10}(f) - 13.82\log_{10}(h_B) + (44.9 - 6.55\log_{10}(h_B))\log_{10}(d) - \\ & 3.2\log_{10}(11.7554h_M)^2 - 4.97, \end{aligned}$$

  where f is the center frequency in MHz, h_B is the height of the transmitter antenna (fixed to 10 m), d is the distance between the transmitter and the receiver in meters, and h_M is the height of the receiver's antenna (fixed to 10 m).

- Indoor model: this model represents a simple indoor scenario, which is useful to simulate typical scenarios such as flats, schools or restaurants. According to this model, the loss, L_{indoor}, experienced during a transmission is:

$$L_{indoor} = 5 + 10\alpha\log_{10}(d) + h_s + (\frac{d}{f_w})h_o,$$

  where α is a constant that depends on the propagation model (in our case, it is set to 4.4), d is the distance in meters between the transmitter and the receiver, h_s is the shadowing factor, f_w is the frequency of walls (set to one wall each 5 meters), and h_o is the obstacles factor.

- Indoor model with random channel effects: we consider the same model as before, but now we introduce random variables to determine the shadowing and obstacles effects in the power losses.

- Residential path-loss model IEEE 802.11ax: such model is included in the 11ax amendment, and captures the path-loss effects of a typical apartments building. The loss, $L_{indoor-ax}$, experienced during a transmission is given by:

$$\begin{aligned} L_{indoor-ax} =& 40.05 + 20log10(\frac{f_c}{2.4}) + 20log10(min(d,5)) + (d > 5)35log10(\frac{d}{5}) + \\ & 18.3F^{\frac{F+2}{F+1} - 0.46} + 5W, \end{aligned}$$

  where f_c is the frequency in GHz, d is the distance between the transmitter and the receiver, F is the number of floors traversed, and W is number of walls traversed in the x-direction plus the number of walls traversed in the y-direction

2.1.2 Co-channel Interference Models

In addition to the introduced path-loss types, it is important to take into account the co-channel interference, which conditions the interaction between devices. The fact of using non-overlapping channels entails more aggressive interactions given that devices in other channels also contribute to the sensed interference in a given receiver. An example of channels overlapping is shown in Figure 2.

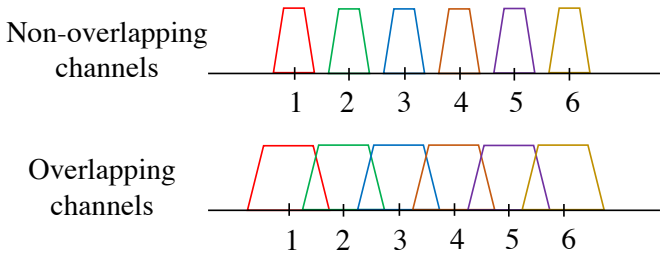


Figure 2: Channel models with and without adjacent interference

In particular, Komondor includes the following co-channel interference models:

- No interference: no power is leaked to adjacent channels.
- Total interference: power from other channels is leaked, so that a 20 dBr decrease is noticed for each channel distance. For instance, the power that channel 1 leaks into channel 3 is the actual power in channel 1 minus 40 dBr.
- Limited interference: in this case, only immediate adjacent channels leak power to the target one, which is done as for the total interference model.

It is important to remark that the incoming power in a given receiver is assumed to be the same during the entire transmission. This relaxation allows us to easily determine whenever the channel of interest is busy or not. A direct implication of it affects to path-loss models used, as well as some of them assume random variations of the medium, preventing to obtain the same result with different power received calculations (so far, power received is added and subtracted when the node accesses and leaves the channel, respectively). Thus, for each node, Komondor stores the incoming power of a given transmission when it begins and subtracts the power when it is over.

2.2 Traffic Modeling

Traffic modeling refers to the capacity of generating data in higher transmission layers. So far, Komondor only considers downlink traffic, so that data transmissions are initiated by APs. Regarding traffic generation, we have considered three different models:

- Full buffer: transmitters are in a permanent saturation regime, so that they always have packets to be sent.
- Poisson: packets are generated according to a Poisson distribution process, so that the time between packets Δ_p is determined by the packet generation rate λ, and is given by $\Delta_\mathrm{p} = e^{\frac{1}{\lambda}}$
- Deterministic: packets are generated at fixed time intervals, Δ_d, given by the packet generation rate, $\Delta_\mathrm{d} = 1/\lambda$.

Figure 3 illustrates the aforementioned traffic models.

2.3 Link Modeling

In order to determine the data rate to be used during a given transmission, the MCS table defined in the IEEE 802.11ax is considered. Komondor assumes that the MCS used between a pair of devices is determined by the SINR at the receiver, so that the maximum allowable MCS is used.

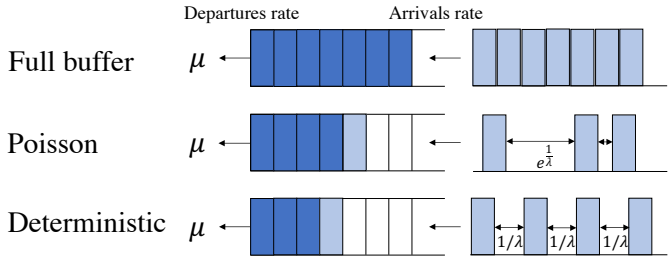


Figure 3: Traffic models used in Komondor

MCS index	SINR interval (dBm)	Modulation type	Coding rate	Data rate (Mbps)			
				20 MHz	40 MHz	80 MHz	160 MHz
0	[-82, -79)	BPSK	1/2	4	8	17	34
1	[-79, -77)	QPSK	1/2	16	33	68	136
2	[-77, -74)	QPSK	3/4	24	49	102	204
3	[-74, -70)	16-QAM	1/2	33	65	136	272
4	[-70, -66)	16-QAM	3/4	49	98	204	408
5	[-66, -65)	64-QAM	2/3	65	130	272	544
6	[-65, -64)	64-QAM	3/4	73	146	306	613
7	[-64, -59)	64-QAM	5/6	81	163	340	681
8	[-59, -57)	256-QAM	3/4	98	195	408	817
9	[-57, -54)	256-QAM	5/6	108	217	453	907
10	[-54, -52)	1024-QAM	3/4	122	244	510	1021
11	≥ 52	1024-QAM	5/6	135	271	567	1143

Table 1: Data rates granted per MCS in IEEE 802.11ax. Guard Intervals (GI) of 1600 ns are only considered.

The required SINR that corresponds to each MCS is defined in Table 1, as well as the granted data rate for each channel width.

So far, link adaptation is not considered[2]. Instead, the MCS to be used between each transmitter-receiver pair is negotiated at the beginning of the first transmission, which remains static throughout all the simulation. In practice, the highest possible modulation is computed for each number of channels used according to the power received (the transmit power decreases as the number of channels used increases). With that, we aim to simulate the Receiver Driver Protocol (RDP), in which few symbols are transmitted at the lowest bit-rate for all the subcarriers. Thus, according to the SINR perceived at the receiver, the MCS is chosen and communicated to the transmitter.

Finally, regarding the transmission time for sending a packet, it is computed as a function of the data rate and the size of the packet to be transmitted. Packet lengths are defined according to the IEEE 802.11ax specification, which is further described in Section 4.1.

2.4 Collisions Modeling

Collisions are critical to be implemented because them determine the actual performance in a given networks, thus allowing to capture situations that may occur in real environments. In particular, packet losses mostly occur because of a coincidence in the backoff computation, hidden-node effects or link asymmetries. An example of a hidden-node collision is shown in Figure 4, in which nodes A and C transmit simultaneously to B because they do not sense the other's transmission. The resulting interference may lead into a collision and provoke that none of the packets can be properly decoded.

In Komondor, a packet loss is considered when the ACK timeout expires at a given transmitter. However, it is critical to identify what caused such loss, which may allow to further analyze the problem, or even to solve it. In particular, Komondor is able to categorize packet losses as follows:

- PACKET_LOST_DESTINATION_TX: packet is discarded because the destination was already transmitting when the packet transmission was attempted. In this case we know that

[2]Future work contemplates the inclusion of Minstrel as a rate adaptation scheme.

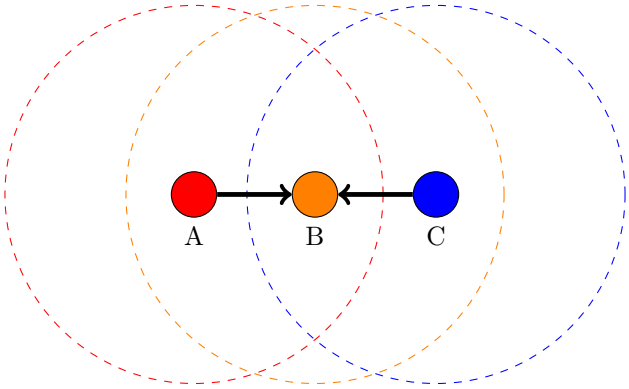


Figure 4: Hidden-node cause of collision

the transmitter could not listen to the receiver at the moment of starting a transmission.

- PACKET_LOST_LOW_SIGNAL: the packet cannot be decoded because the signal strength is not enough (i.e., it is less than capture effect).
- PACKET_LOST_INTERFERENCE: packet is lost due to interference sensed at the receiver, which does not accomplish the capture effect condition.
- PACKET_LOST_PURE_COLLISION: packet is lost because two nodes transmit to the same destination, so that both signal strengths are enough to be decoded at the receiver (the situation shown in Figure 4).
- PACKET_LOST_LOW_SIGNAL_AND_RX: the destination is already receiving data when a new data transmission starts. However, the newest signal strength is not enough to be decoded in normal conditions. In such situation, two problems are reported.
- PACKET_LOST_RX_IN_NAV: packet is lost because the target node is in a NAV period (it previously decoded an RTS/CTS sequence).
- PACKET_LOST_BO_COLLISION: the packet is lost because a backoff collision occurred, i.e., two or more interfering devices ended their backoff simultaneously.

3 MAC Features

In this Section we provide an overview of the main MAC layer features included in Komondor, so that their practical implementation can be further understood.

3.1 Channel Access

When an AP has a packet to be sent, it implements CSMA/CA to access the medium, which makes use of the DCF. With that, transmissions are carried out if the target channel has been empty for a given Backoff (BO) time, which is computed according to a dynamic CW. A channel is considered to be empty if the interference in it is lower than a Clear Channel Assessment (CCA) threshold. Furthermore, a packet transmission is successful whenever the receiver SINR is equal or higher than a Capture Effect (CE) threshold, which allows defining the frequency at which packet losses occur, regardless on the Modulation Coding Scheme used.

3.1.1 Distributed Coordination Function

As previously mentioned, the CSMA/CA operation is based on the DCF, which orchestrates channel access in a distributed manner. Roughly, transmitters (e.g., devices that have data to be transmitted) choose a random BO value, which is decremented only if the channel is sensed as idle due to the CCA condition. Otherwise, the BO is freeze and the transmission is contended. Figure 5 shows an example of the DCF operation in which three nodes listen to each other in the same wireless scenario. As it is shown, STA2 wins the channel access because its initial BO timer is

the lowest one. During the packet transmission, STA1 listens the channel busy and does not stops its BO operation. Once data transmission is finished and the channel has been idle for a DIFS interval, the BO procedure is activated in all the devices that have a packet to be transmitted and sense the channel idle.

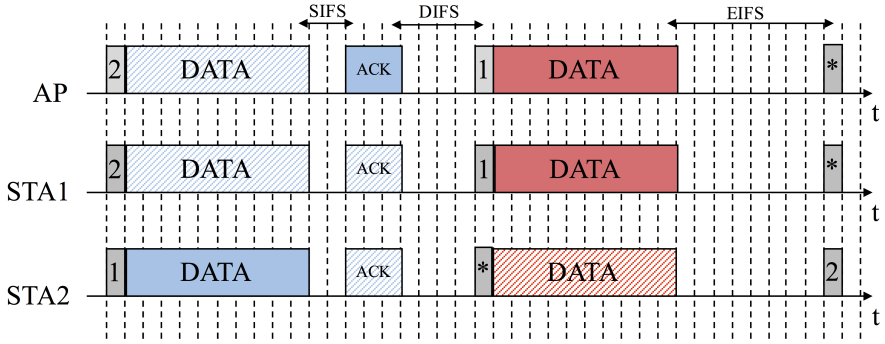


Figure 5: Example of the DCF procedure

The process of generating a BO before a packet transmission is determined by the Congestion Window (CW), which has been considered to be done in two ways. Komondor implements two kinds of BO PDFs to be used in different situations:

- Uniform: the generated BO is a number between 0 and CW − 1, and all the values have the same probability.
- Exponential: instead of using an uniform distribution, we use an exponential one, so that the generated BO is given by a Gaussian distribution with mean $\frac{\text{CW}-1}{2}$.

An important consideration with respect to discrete BO is that devices are synchronized, which allows reproducing collisions by BO that depend on the congestion window and the number of coexisting nodes.

3.1.2 Contention Window Adaptation

To complement the DCF operation, dynamic CW adaptation is considered on a per-packet basis. Optionally, one can activate CW adaptation so that the CW increases or decreases according to successful/failed transmissions. Given a minimum and a maximum boundaries for CW (CW_{min} and CW_{max}, respectively), the reset operation is performed when a successful transmission is carried out. In such situation, the CW is set to CW_{min}. Otherwise, when packet losses occur, the CW is increased without exceeding CW_{max}. To do so, a counter (namely CW_{count}) is maintained and increased one unit each time a packet loss occurs. Then, the CW is computed as $CW = CW_{min} \times 2^{CW_{count}}$.

3.1.3 Capture Effect

In order to perform transmissions through the radio link, a receiver must perceive that the desired signal strength is bigger than a CE threshold, so that the received data can be distinguished among noise and interference present in the channel. Despite there are two types of interference patterns (*stronger-first* and *stronger-last*), Komondor only considers data transmissions in which the target packet arrives first, just as shown in Figure 6. Otherwise, when a more data packet arrive in the middle of a data transmission, all the packets are discarded if the newest signal is stronger than the first one.

3.2 RTS/CTS and NAV Allocation

The RTS/CTS mechanism is implemented in order to minimize the collisions by hidden node. Through RTS/CTS, transmitting nodes attempt to allocate the channel during the duration of their transmissions. For that, they send control packets and wait for confirmation about the clearness of the channel from the receivers' point of view. Through such packet exchange, overlapping nodes

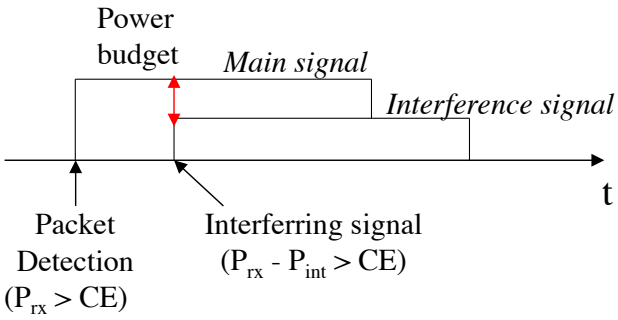


Figure 6: Stronger-First Capture Effect example

must set a virtual carrier sensing during the transmission duration, which allows reducing the collisions by hidden node. The RTS/CTS operation is exemplified in Figure 7.

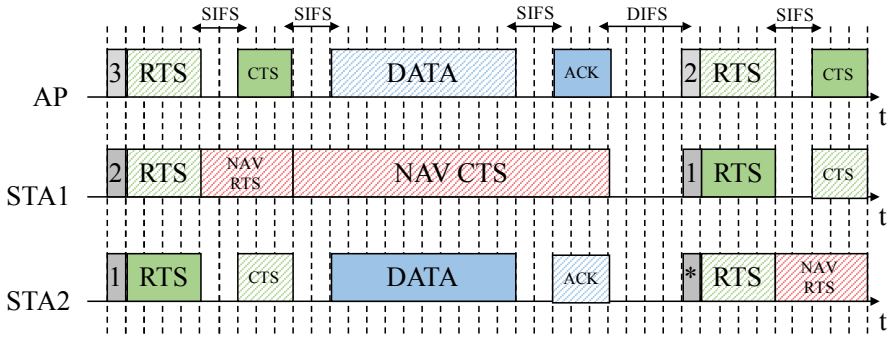


Figure 7: Example of RTS/CTS implementation. The transmitter (STA2) sends an RTS packet before starting a transmission. The receiver (AP) answers with a CTS as it senses the channel free. The other coexisting devices (STA1) that listen either the RTS and/or the CTS, set their NAV accordingly.

3.3 Channel Bonding

TO BE COMPLETED BY @SERGIO

Channel Bonding (CB) is one of the most promising techniques to enhance spectral efficiency in IEEE 802.11ax WLANs, since it aims to make the most of medium by transmitting data over several channels. For that, different CB policies are implemented in Komondor, which can be interchangeably applied by the simulated WLANs. To perform CB, one may explicitly define the available range of channels for each WLAN. Then, the following policies can be applied:

- Only Primary (OP): the legacy operation is performed, so that only the primary channel is attempted to be accessed.
- Static Channel Bonding (SCB): carrier sensing is performed at the primary channel. However, when attempting to transmit, all the channels within the CB range must be clear. Otherwise, a new backoff is computed.
  - Aggressive SCB:
  - Power of 2 SCB
- Dynamic Channel Bonding:
  - Aggressive DCB
  - Power of 2 DCB[3]: transmit in the larger channel range that is allowed by the *log2* structure shown in Figure **??**.

[3]In the Matlab code, this model corresponds to the configuration *onlymax* = *true* and *selfloop* = *false*

- Policy-dependent:
  - Sergio's PhD :D

3.4 Packet Aggregation

Packet aggregation aims to reduce transmission overheads such as headers, SIFS and DIFS intervals or backoff periods. For that, it concatenates N MPDUs to be sent over the same packet transmission, so that it can be acknowledged through a block ACK. Komondor allows to define the number of aggregated packets, which value remains static during the entire simulation.

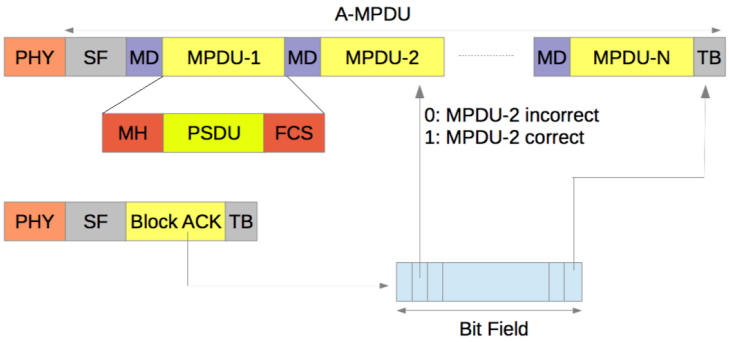


Figure 8: Example of packet aggregation in which N MPDUs are concatenated to be sent during the same packet transmission.

4 Komondor Main Features Validation through CTMN

In this Section we aim to validate the core operation of Komondor by using a set of key scenarios. Moreover, we use the Continuous Time Markov Networks (CTMNs) model [10] for analytical comparison.

4.1 IEEE 802.11ax Parameters

Before showing the set of Komondor validations, we introduce in Table 2 the IEEE 802.11ax parameters that we consider for simulations, and which are recommended to be kept in order to emulate 11ax's behavior.

4.2 Basic Operation Validation

In order to validate the basic operation of Komondor (e.g., DCF, RTS/CTS), we propose the following set of non-fully-overlapping scenarios, which are shown in Figure 9a:

- Topology 1 (*T1*): all the nodes are able to listen each other, so that they share the channel on equal terms.
- Topology 2 (*T2*): either AP_A or AP_C generate enough interference to contend AP_B's transmission.
- Topology 3 (*T3*): simultaneous transmission of both AP_A and AP_C generates enough interference to make AP_B sense the channel busy.
- Topology 4 (*T4*): none of the nodes' transmission is sensed by the others.

The average throughput experienced by each WLAN in each of the regions is shown in Figure 9b. Regarding topology *T1*, when APs are close enough to be inside the carrier sense range of each other in a fully-overlapping manner, the medium access is shared fairly because of the CSMA/CA mechanism. For that reason, the throughput is decreased to approximately 1/3 with respect to topology *T4*. Regarding the difference between Komondor and CTMNs results, note that successful

Parameter	Description	Value
CW_{min}	Min. contention window	16
m	Backoff stage	5
CCA	CCA threshold	-82 dBm
P_{tx}	Transmission power	15 dBm
G_{tx}	Transmitting gain	0 dB
G_{rx}	Reception gain	0 dB
L_{data}	Length of a data packet at NET layer	12000 bits
L_{BACK}	Length of a block ACK	240 bits
L_{RTS}	Length of an RTS packet	160 bits
L_{CTS}	Length of a CTS packet	112 bits
n_{agg}	Number of data packets aggregated	64
CE	Capture effect threshold	20 dB
N	Background noise level	-95 dBm
T_{slot}	Slot duration	9 μs
SIFS	SIFS duration	16 μs
DIFS	DIFS duration	34 μs
PIFS	PIFS duration	25 μs
η	Constant packet error rate	0.1
f_c	Central frequency	5 GHz
T_{ofdm}	Duration of OFDM symbol	16 μs
T_{phy}	Duration of legacy PHY header	20 μs
n_{ss}	Single user spatial streams	1
T_{phy}^{HE}	Duration of HE header	$(16 + n_{SUSS} * 16)$ μs
L_{sfF}	Length of the MAC's service field	16 bits
L_{del}	Length of the MAC's MPDU delimiter	32 bits
L_{mac}	Length of the MAC header	272 bits
L_{tail}	Length of the MAC's tail	6 bits

Table 2: IEEE 802.11ax parameters

simultaneous transmissions (or slotted backoff collisions) are allowed in Komondor, as well as the capture effect is accomplished. Henceforth, the throughput is slightly higher in Komondor.

The neighbor overlapping case in topology *T2*, where A and C can transmit at the same time whenever B is not active, but B can only do so when both A and C are not active, is a clear case of exposed-node starvation. Namely, B has very few transmission opportunities as A and C are transmitting almost permanently and B must continuously freeze its backoff consequently.

Regarding *T3*, the sum of the interference that B perceives when A and C are transmitting at the same time prevents it to decrease the backoff. However, B is able to decrement the backoff any time A or C are not transmitting. As it is shown, the CTMN estimates that B is transmitting only the 50.1% of the time, which contrasts with the results granted by Komondor (75%). However, according to the capture effect ($SINR_B > CE$)), packets are going to be lost or not, as displayed for node B in *T3-noCE* and *T3*, respectively.

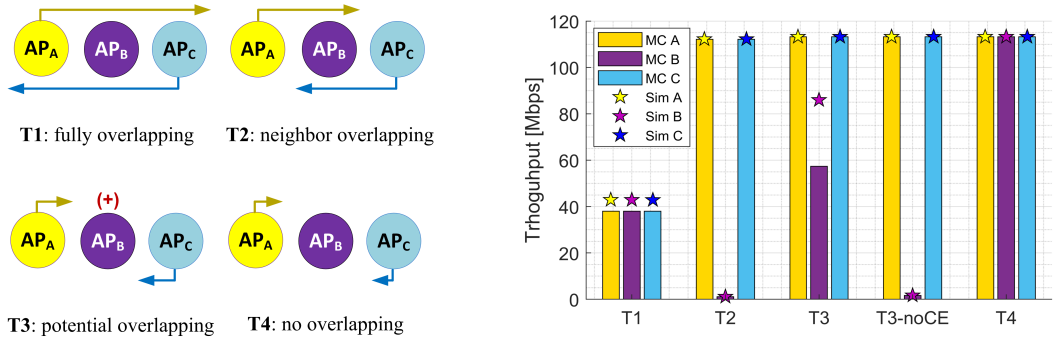
Finally, in topology *T4*, due to the fact that WLANs are isolated, the number of successful parallel transmissions is maximum.

4.3 Channel Bonding Validation

Now, in order to validate the most important technique included in the first version of the Komondor simulator (i.e., Channel Bonding), we introduce the scenarios shown in Figure 10, which consider two overlapping WLANs.

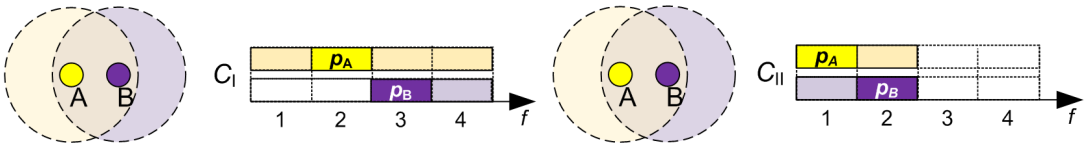
In Table 3 we show the results of applying the different DCB policies introduced in Section 3.3. We display the average throughput experienced by WLANs A and B (Γ_A and Γ_B, respectively), and by the whole network (Γ).

Let us first consider *Scenario I*. As expected, due to the fact that both WLANs overlap in channels 3 and 4 when transmitting in their whole allocated channels, the SCB policy reaches just three feasible states in which WLANs cannot transmit at the same time. In the case of OP



(a) Topologies (b) Average throughput

Figure 9: Scenarios and results for Komondor validations. Yellow and blue arrows indicate the carrier sense range of WLANs A and C, respectively (CST is equal for all the APs). The carrier sense range of WLAN B is not displayed. *T3-noCE* refers to topology *T3* when the STA in B does not accomplish the CE condition whenever A, B and C are active. MC and Sim refer to the values obtained through CTMN model and Komondor, respectively.



(a) *Scenario I* (b) *Scenario II*

Figure 10: WLANs A and B are inside the carrier sense range of each other with potentially overlapping channels 1 and 2.

both WLANs are forced to pick just their primary channel for transmitting and, therefore, they can transmit simultaneously. Regarding the AM policy, which is usually used as de-facto when applying DCB, it allows simultaneous transmissions provided that WLAN B started transmitting when the channel was idle. Notice that with AM, any time the backoff of an AP in WLAN X expires and the channel is idle, a given AP would pick the the widest available channel. The last policy studied is PU, which is characterized by providing further exploration regarding the possible channel range combinations. In scenario I, whenever the channel is idle and the backoff of either A or B expires, each of the possible available channel ranges may be chosen with same probability. For instance, A may choose channel ranges 2, 1-2 or 1-4 with probability 1/3. Similarly, B may choose transmitting over channels 3 or 3-4.

Intuitively, one could think that, as it occurs in *Scenario I*, picking always the widest channel found free by means of AM, i.e., maximizing the throughput of the immediate transmission (or short-term throughput), may be the best strategy for optimizing the long-term throughput as well. However, the *Scenario II* depicted in Figure 10b, is a counterexample that illustrates such lack of applicable intuition. Firstly, with OP, due to the fact that WLANs are only allowed to transmit in their primary channel, simultaneous transmissions can be carried out. On the other hand, with SCB, WLANs can only transmit in their complete allocated channel, thus allowing a single transmission at a time. Notice that in this case AM generates the same transition probabilities (and respective average throughput) than SCB because whenever a WLAN has the possibility to transmit. Finally, PU picks uniformly at random any of the possible channel combinations that A and B may choose when terminating their backoff in case that the channel is idle.

Concerning the throughput differences in the values obtained by CTMNs and Komondor, we note that the main disparities correspond to the AM and SCB policies. It is important to remark that while CTMN does not consider neither backoff collisions nor NAV periods, Komondor actually does so in a realistic way. Therefore, in Komondor, whenever there is a slotted backoff collision, the RTS packets can be decoded by the STAs in both WLANs and the average throughputs are increased consequently. Regarding the NAV periods, an interesting phenomena occurs in *Scenario I* when implementing SCB, AM or PU. While the RTS packets sent by B cannot be decoded by A because its primary channel is always outside the transmission channel range of B, the opposite

	Scenario I				*Scenario II*			
$\mathcal{D}$	$\lvert\mathcal{S}\rvert$	Γ_A	Γ_B	Γ	$\lvert\mathcal{S}\rvert$	Γ_A	Γ_B	Γ
OP	4	113.23 113.23	113.23 113.23	226.47 226.46	4	113.23 113.23	113.23 113.23	226.47 226.46
SCB	3	143.46 131.98	143.46 148.85	286.92 280.83	3	109.19 108.72	109.19 108.84	218.38 217.56
AM	5	220.12 217.60	212.21 214.81	432.34 432.41	3	109.19 108.72	109.19 108.84	218.38 217.56
PU	10	149.14 149.20	148.38 148.42	297.52 297.63	6	113.19 113.20	113.19 113.18	226.38 226.38

Table 3: DCB policy effect on the average throughput [Mbps] in *Scenario I* and *Scenario II*. The values obtained through Komondor are displayed in blue, while the other correspond to the CTMN model.

occurs when A sends them. Due to the fact that the RTS is duplicated in each of the basic channels used for transmitting, whenever A transmits in its whole allocated channel, B is able to decode the RTS and enters in NAV consequently.

5 Tutorial and Development Notes

In this Section we provide a brief tutorial to encourage researchers and other practitioners to use the Komondor simulator for their own experiments, and even to participate in the project.

5.1 Brief Tutorial

Komondor is composed by several modules that allow performing simulations with a high degree of freedom. Here we provide some details on the most important modules, as well as on their practical execution. Figure 11 summarizes the main operations carried out by Komondor.

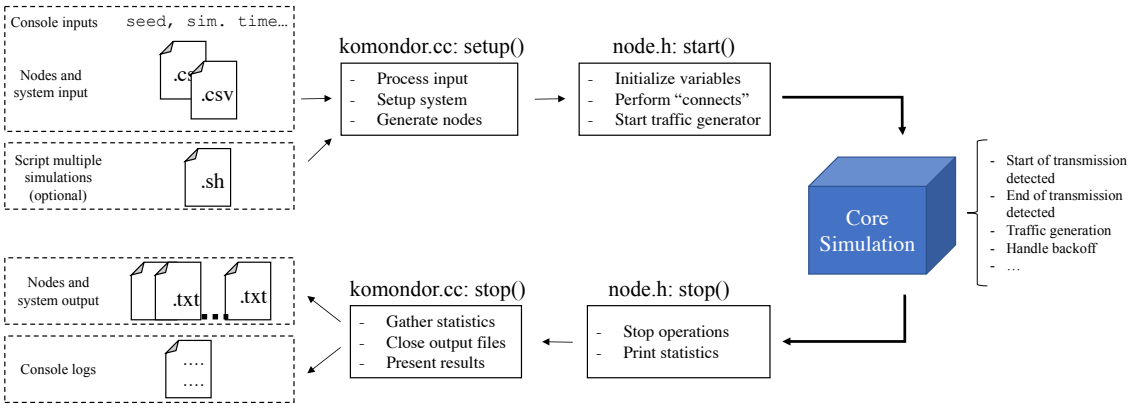


Figure 11: Komondor flowchart

As shown, Komondor receives a set of inputs (nodes information, simulation time, etc.) and initializes the main module, which is in charge of generating the network and gathering useful information regarding the simulation. During the core simulation, nodes interact each other by sending packets, so that DCF operation is implemented for accessing to the channel. Finally, when the simulation time is up, a set of outputs are generated in order to shed some light on the network performance.

5.1.1 Files Organization

To properly understand the Komondor's operation, it is important to know how the project is organized, which allows obtaining a broader vision of the different modules that constitute it. The code is organized as follows:

- COST libraries: contains the necessary COST libraries that constitute the Komondor's primitive operation.
- Code: contains the two core files (*komondor.cc* and *node.h*), which are in charge of orchestrating all the simulation. In addition, here we find the inputs and the file that compiles the libraries for executing the code (*build_local*).
- Methods: by following clean architecture guidelines, independent methods used by both *komondor.cc* and *node.h* files are contained in the methods folder. Several libraries are provided according to the nature of their functions. For instance, *backoff_methods.h* contains methods to handle the backoff operation in DCF.
- Structures: the Komondor simulator considers four main objects to carry out its operation. The first one is *wlan.h*, which defines the main characteristics of a WLAN (WLAN id, list of associated STAs, etc.). Furthermore, the *notification.h* object allows to exchange packets between devices. Finally, *logger.h* and *logical_nack.h* are used for auxiliary purposes, which are displaying logs and notifying packet losses causes, respectively.
- List of macros: all the static parameters are contained in the *list_of_macros.h* file.

5.1.2 Compilation and Execution

To compile and execute Komondor, the following instructions must be followed[4]:

1. Set the .csv input files (further defined in Section 5.1.3)
2. Access to the *KomondorSimulator* directory
3. Execute ".build_local". This file contains the instructions for compiling the Komondor code. It has been updated to enable debugging with Valgrind[5].
4. Execute *./KomondorSimulator arg_1 arg_2 ... arg_n*, where *arg_i* is the i_{th} input argument:
   - *arg_1* (INPUT_FILE_SYSTEM_CONFIGURATION): file containing system information (e.g., number of channels available, traffic model, etc.). The file must be a .csv with semicolons as separators.
   - *arg_2* (INPUT_FILE_NODES): file containing nodes information (e.g., position, channels allowed, etc.).The file must be a .csv with semicolons as separators.
   - *arg_3* (OUTPUT_FILE_LOGS): path to the output file to which write results at the end of the execution (if the file does not exist, the system will create it).
   - *arg_4* (FLAG_SAVE_SYSTEM_LOGS): flag to indicate whether to save the system logs into a file (1) or not (0).[6]
   - *arg_5* (FLAG_SAVE_NODE_LOGS): flag to indicate whether to save the nodes logs into separate files (1) or not (0). If this flag is activated, one file per node will be created.
   - *arg_6* (FLAG_PRINT_SYSTEM_LOGS): flag to indicate whether to print the system logs (1) or not (0).
   - *arg_7* (FLAG_PRINT_NODE_LOGS): flag to indicate whether to print the nodes logs (1) or not (0).
   - *arg_8* (SIM_TIME): simulation time in seconds.
   - *arg_8* (SEED): random seed for the experiments.
5. Collect the results either in the output files or in the console.

In case that the user does not have permissions to execute some of the files, grant them by introducing the following command in the target folder: *$ chmod -R 777 **.

[4] A GNU-based OS is assumed to be used for simulations, including basic compilation programs such as *gcc*.

[5] Valgrind is a programming tool for memory debugging, memory leak detection, and profiling. Valgrind main website: http://valgrind.org/

[6] Major increases in the execution time may occur if nodes logging is activated. E.g., for a simulation of 4 nodes, simulating 1000 seconds takes 1.127 s and 15.672 s when not logging and when doing so, respectively.

5.1.3 Input files

To define the simulation environment, the Komondor simulator relies in the following two types of input files:

- **System input:** defines global input parameters such as packets length or propagation models. System input parameters are defined in Table 4.
- **Nodes input:** defines specific nodes' characteristics such as type, location, or implementing features (e.g., CB policy). There are two ways of generating nodes, which is indicated in the file name. In case of including the keyword *nodes*, all the devices (both APs and STAs) must be introduced and described. Otherwise, if including the keyword *aps*, only the APs are defined, so that a set of STAs is randomly generated under certain introduced parameters (e.g., minimum/maximum number of STAs, maximum distance between APs and STA).[7] As a final remark, in order to ensure a proper execution, it is mandatory to introduce an input file with a list of nodes ordered by *node_id* and starting with *node_id* = 0 (it is a requirement for the array responsible of storing the power perceived by each node). Table 5 describes the Nodes input parameters for both *nodes* and *aps* files.

Parameter	Type	Description
num_channels	int	Maximum number of frequency channels in the system
basic_channel_bandwidth	int	Bandwidth for each channel [MHz]
pdf_backoff	int	PDF to compute the backoff ()
pdf_tx_time	int	PDF to compute the tx time ()
packet_length	int	Length of data packets [bits]
ack_length	int	Length of ACK packets [bits]
num_packets_aggregated	int	Number of packets aggregated per transmission
path_loss_model	int	Path-loss model (0: FSPL, 1: Hata, 2: Indoor 1, 3: Indoor 2, 4: TGax scenario 1)
capture_effect	int	Capture Effect Threshold [dB]
noise_level	int	Floor noise level [dBm]
adjacent_channel_model	int	Co-channel interference model (0: without adjacent interference, 1: contiguous adjacent interference, 2: complete adjacent interference)
collisions_model	int	Collisions model (reserved)
SIFS	int	SIFS period [μs]
constant_PER	int	Defines a constant Packet Error Rate
traffic_model	int	Traffic model (0: full buffer, 1: Poisson distr., 2: deterministic distr.)
backoff_type	int	Type of backoff (discrete: 0, continuous: 1)
rts_length	int	Length of RTS packets [bits]
cts_length	int	Length of CTS packets [bits]
cw_adaptation	bool	For activating CW adaptation
pifs_activated	bool	For activating PIFS

Table 4: System input parameters description

5.1.4 Input scripts

In order to facilitate researchers' work, we provide a set of scripts that allow performing several simulations at once, which is useful to avoid processing different output files. Such sample scripts can be found in the "scripts multiple executions" folder, which perform the following operations:

- *multiple_inputs_script.sh*: processes all the input files contained in ./input/script_input_files and generates a simulation for each one.

[7] The usage of APs input files is discouraged to the lack of maintenance.

Parameter	Type	Nodes or APs	Description
node_code	String	nodes	Code assigned to the node
node_type	int	nodes	Type of node (0: AP, 1: STA)
wlan_code	String	both	Code assigned to the WLAN
destination_id	int	nodes	To specify the ID of the destination (packets would be only sent to that devices). Setting it to -1 indicates random destination.
min_sta_number	int	aps	Minimum number of associated STAs
max_sta_number	int	aps	Maximum number of associated STAs
max_distance_sta	int	aps	Maximum distance of associated STAs
x	int	both	X location [m]
y	int	both	Y location [m]
z	int	both	Z location [m]
primary_channel	int	both	Primary channel
min_channel_allowed	int	both	Left channel in range
max_channel_allowed	int	both	Right channel in range
cw	int	both	Fixed CW
cw_stage	int	both	Initial CW stage (for CW adaptation)
tpc_min	int	both	Minimum transmit power allowed [dBm]
tpc_default	int	both	Default transmit power allowed [dBm]
tpc_max	int	both	Maximum transmit power allowed [dBm]
cca_min	double	both	Minimum CCA allowed [dBm]
cca_default	double	both	Default CCA allowed [dBm]
cca_max	double	both	Maximum CCA allowed [dBm]
tx_antenna_gain	int	both	Gain of the tx antenna [dB]
rx_antenna_gain	int	both	Gain of the rx antenna [dB]
channel_bonding_model	int	both	Channel bonding model (0: only primary, 1: SCB, 2: SCB log2, 3: always max, 4: always max log2, 5: always max log2 MCS, 6: uniform probability log2)
modulation_default	int	both	Modulation set by default (0 to use dynamic MCS)
central_freq	int	both	Frequency band used (2,4 or 5 GHz)
lambda	float	both	Packets transmission rate [packets/s]
ieee_protocol	int	both	IEEE protocol used

Table 5: Nodes input parameters description

- *multiple_inputs_script_several_seeds.sh*: in addition to process multiple inputs, generates different seeds for each simulation.

Similar procedures can be implemented to extend the current provided functionalities, such as reading multiple system inputs or generating specific output reports.

5.1.5 Output files

A lot of effort has been put on the output generation, since it is a sensitive module that allows understanding and validating the results provided by the simulator. Henceforth, we provide different kinds of outputs, which refer to console and file output logs. Note, as well, that generating output files considerably increases the computation time.

Regarding console output logs, them can be activated through *arg_6* and *arg_7* during the execution, which refer to system and nodes logs, respectively (see Section 5.1.2). Additionally, those logs can be copied into files, which are saved into the *output* folder, only if *arg_4* and *arg_5* are set to 1. While the path of the system's output file must be specified (*arg_3*), nodes' files are automatically created.

Finally, a set of statistics are show per node and for the entire simulation. Such statistics include throughput experienced, collisions, nodes sent, RTS/CTS sent, etc. An example of nodes and system statistics is shown in Figures 12 and 13

```
------- AP_A (N0) ------
  - Throughput = 102.120960 Mbps
  - RTS/CTS sent = 14750 - RTS/CTS lost = 0  (0.00 % lost)
    · RTS lost due to slotted BO = 0 (0.000000 %)
  - Data packets sent = 14750 - Data packets lost = 1453  (9.850847 % lost)
  - num_tx_init_tried = 14750 - num_tx_init_not_possible = 0 (0.000000 % failed)
    · Time EFFECTIVELY transmitting in N channels:
    · - 1: 86.849472 s (86.85 %)
    · Time EFFECTIVELY transmitting in each channel:
    · - 0 = 86.87 s (86.87 %)
    · - 1 = 0.00 s (0.00 %)
    · Number of tx trials per number of channels:
    · - 1: 14750 (100.00 %)
    · - 2: 0 (0.00 %)
  - num_tx_init_not_possible = 0


------- AP_B (N2) ------
  - Throughput = 101.798400 Mbps
  - RTS/CTS sent = 14751 - RTS/CTS lost = 0  (0.00 % lost)
    · RTS lost due to slotted BO = 0 (0.000000 %)
  - Data packets sent = 14750 - Data packets lost = 1495  (10.135593 % lost)
  - num_tx_init_tried = 14751 - num_tx_init_not_possible = 0 (0.000000 % failed)
    · Time EFFECTIVELY transmitting in N channels:
    · - 1: 86.581660 s (86.58 %)
    · Time EFFECTIVELY transmitting in each channel:
    · - 0 = 0.00 s (0.00 %)
    · - 1 = 86.61 s (86.61 %)
    · Number of tx trials per number of channels:
    · - 1: 14751 (100.00 %)
    · - 2: 0 (0.00 %)
  - num_tx_init_not_possible = 0
```

Figure 12: Example of nodes statistics in Komondor

```
 General Statistics:
  - Total number of packets sent = 58999
  - Total throughput = 408.27 Mbps
  - Average number of packets sent per WLAN = 14749
  - Average throughput per WLAN = 102.07 Mbps



- Average throughput per WLAN = 102.07 Mbps
- Proportional Fairness = 32.04
- Jain's Fairness = 1.00
- Average number of data packets successfully sent per WLAN = 14749.75
- Average number of RTS packets lost due to slotted BO = 0.00 (0.00 % loss)

------- FOR COMPARING TO BIANCCI -------
- Prob. collision by slotted BO = 0.000000
- Aggregate throughput = 408.268800 Mbps
- Aggregate number of transmission not possible = 0
---------------------------------------


    · - 1: 14750 (100.00 %)
    · - 2: 0 (0.00 %)
    · - 1: 14751 (100.00 %)
    · - 2: 0 (0.00 %)
    · - 1: 14750 (100.00 %)
    · - 2: 0 (0.00 %)
    · - 1: 14749 (100.00 %)
    · - 2: 0 (0.00 %) SIMULATION 'test' FINISHED
----------------------------------------
# ------------------------------------------------------------------------
# CostSimEng with SimpleQueue, stopped at 100.000000
# 590007 events processed in 3.920 seconds, event processing rate: 150523
administrador@ws119785:~/workspace/Komondor/Code/komondor_main$ S
```

Figure 13: Example of system statistics in Komondor

5.1.6 Events Categorization

In order to make output results more understandable, logs are categorized according to the event that generates it. With that, further filtering processes can be carried out by developers. Table 6 describes the codes used for each type of event.

Method	Type	Sub-type	Event description
Setup()	A	-	-
Start()	B	B00	Start()
		B01	Start() end
		B02	Node's info (one line)
Stop()	C	C00	Stop()
		C01	Stop() end
		C02	Time transmitting in number of channels
		C03	Time transmitting in each channel
		C04	Packets sent
		C05	Throughput
inportSomeNodeStartTX()	D	D00	inportSomeNodeStartTX()
		D01	inportSomeNodeStartTX() end
		D02	Node N has started a TX in channels: c_left - c_right
		D03	Pre update channel state
		D04	Distance to transmitting node
		D05	Power received from transmitting node
		D06	Post update channel state
		D07	I am (or not) the TX destination
		D08	Current SINR
		D09	Capacitiy
		D10	Primary channel affected (or not)
		D11	Power sensed in primary channel
		D12	CCA exceeded (or not)
		D13	Backoof active (or not)
inportSomeNodeFinishTX()	E	E00	inportSomeNodeFinishTX()
		E01	inportSomeNodeFinishTX() end
		E02	N%d has finished a TX in channel range: %d - %d
		E03	Initial power of transmitter
		E04	Pre update channel state
		E05	Post update channel state
		E06	Primary channel affected (or not)
		E07	Power sensed in primary channel
		E08	CCA exceeded (or not)
		E09	I am transmitting (or not)
endBackoff()	F	F00	endBackoff()
		F01	endBackoff() end
		F02	Channels for transmitting
		F03	Transmission is possible (or not)
		F04	Selected transmission range
		F05	New backoff generated
myTXFinished()	G	G00	myTXFinished()
		G01	myTXFinished() end
		G02	New backoff generated

Table 6: Node's event logs encoding

5.2 Code development

Here we provide some clarifications regarding code implementation, wit the aim to facilitate the Komondor's usage and manipulation to developers that may be interested.

5.2.1 Main considerations

Some technical information regarding code development is worth to be mentioned to properly understand how to use and modify the Komondor simulator. So far, the main considerations to be taken into account are:

- **Power and CCA**: power variables are stored in pW (pico watts) in order to be able to operate power magnitudes without loosing resolution[8]. However, values are presented to the user in dBm. W (-30) - mW (0) - uW (+30) - nW (+60) - pW (+90)
  $P_{\text{pw}} = 10^{\frac{P_{\text{dBm}}+90}{10}}$

5.2.2 Miscellany

- Transmitting capability: we have added a flag to each node that determines if it is able to transmit (1) or not (0), so that we can decide if the node is only listening or both transmitting and listening.
- Progress bar: the Komondor simulation progress bar is displayed through a *printf()* command called by any node with *node_ id* set to 0. If no node has *node_ id* set to 0, the progress bar is not displayed.

[8]For instance., the sum of two signals of power values -85 dBm (3.162 pW) and -90 dBm (1 pW), respectively, is -83.803 dBm (4.162 pW).

6 Conclusions

In this document we provided an overview of the first version of the Komondor simulator, which aims to reproduce the basic operation of IEEE 802.11ax WLANs in addition to allow the utilization of intelligent systems. We provided the system model considered when building the simulator, as well as the main MAC features implemented. Additionally, due to the open source nature of this project, we provided basic information of interest for developers that are expected to use or even modify this networks simulator.

Regarding validations, we provided a set of meaningful test scenarios to prove the proper behavior of the simulator. As shown, tests were satisfactory as the throughput computed with Komondor and the CTMN model are pretty similar, and the differences were properly justified.

This project is expected to move forward for including of novel mechanisms such as OFDMA, MU-MIMO, TPC or CST adjustment. In addition, intelligent agents are expected to be included for making operations such as Dynamic CB (DCB).

References

[1] Sergio Barrachina-Muñoz and Francesc Wilhelmi. Komondor: An IEEE 802.11ax simulator. https://github.com/wn-upf/Komondor, 2017.

[2] G. Chen and B. K. Szymanski. Reusing simulation components: cost: a component-oriented discrete event simulator. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, pages 776–782. Winter Simulation Conference, 2002.

[3] IEEE p802.11ax/d2.0, November 2017.

[4] Boris Bellalta. Ieee 802.11 ax: High-efficiency wlans. *IEEE Wireless Communications*, 23(1):38–46, 2016.

[5] Francesc Wilhelmi, Boris Bellalta, Cristina Cano, and Anders Jonsson. Implications of decentralized q-learning resource allocation in wireless networks. *arXiv preprint arXiv:1705.10508*, 2017.

[6] Francesc Wilhelmi, Boris Bellalta, Jonsson Anders Neu Gergely Cano, Cristina, and Sergio Barrachina. Collaborative spatial reuse in wireless networks via selfish bandits. *arXiv preprint arXiv:1705.10508*, 2017.

[7] Setareh Maghsudi and Sławomir Stańczak. Joint channel selection and power control in infrastructureless wireless networks: A multiplayer multiarmed bandit framework. *IEEE Transactions on Vehicular Technology*, 64(10):4565–4578, 2015.

[8] Setareh Maghsudi and Sławomir Stańczak. Channel selection for network-assisted d2d communication via no-regret bandit learning with calibrated forecastingtre. *IEEE Transactions on Wireless Communications*, 14(3):1309–1322, 2015.

[9] Masaharu Hata. Empirical formula for propagation loss in land mobile radio services. *IEEE transactions on Vehicular Technology*, 29(3):317–325, 1980.

[10] Boris Bellalta, Alessandro Zocca, Cristina Cano, Alessandro Checco, Jaume Barcelo, and Alexey Vinel. Throughput analysis in csma/ca networks using continuous time markov networks: a tutorial. In *Wireless Networking for Moving Objects*, pages 115–133. Springer, 2014.