



# An event-based IEEE 802.11ax WLANs Simulator

Sergio Barrachina-Muñoz, Francesc Wilhelmi

# **Introduction**

# Introduction to the simulator

- Open source, available at <https://github.com/wn-upf/Komondor>
- Based on COST
  - Implementation of IEEE 802.11ax novel functionalities
  - High-density scenarios
  - Prepared to handle intelligent agents
- Documentation:
  - Tutorial at [https://github.com/wn-upf/Komondor/blob/master/Documentation/Tutorial/LaTeX%20files/komondor\\_technical\\_report.pdf](https://github.com/wn-upf/Komondor/blob/master/Documentation/Tutorial/LaTeX%20files/komondor_technical_report.pdf)
  - Technical report at [https://github.com/wn-upf/Komondor/blob/master/Documentation/Technical%20Report/LaTeX%20files/komondor\\_technical\\_report.pdf](https://github.com/wn-upf/Komondor/blob/master/Documentation/Technical%20Report/LaTeX%20files/komondor_technical_report.pdf)

# COST

- CompC++ library that allows generating discrete event simulations
- Main website: <http://www.ita.cs.rpi.edu/cost.html>

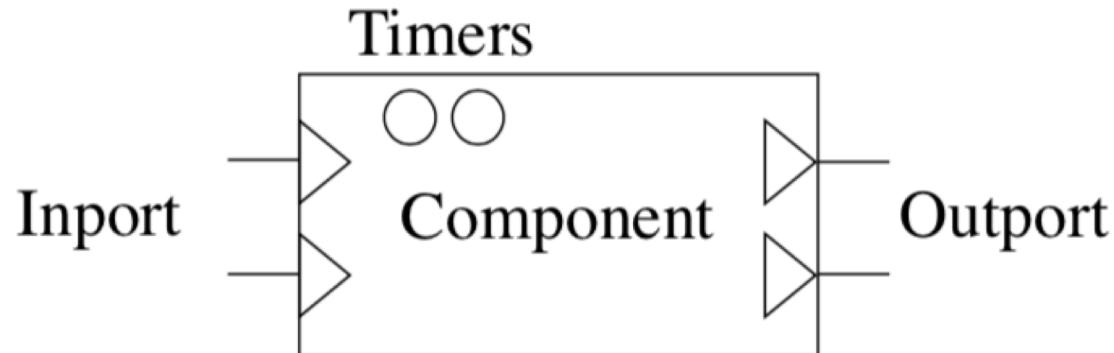
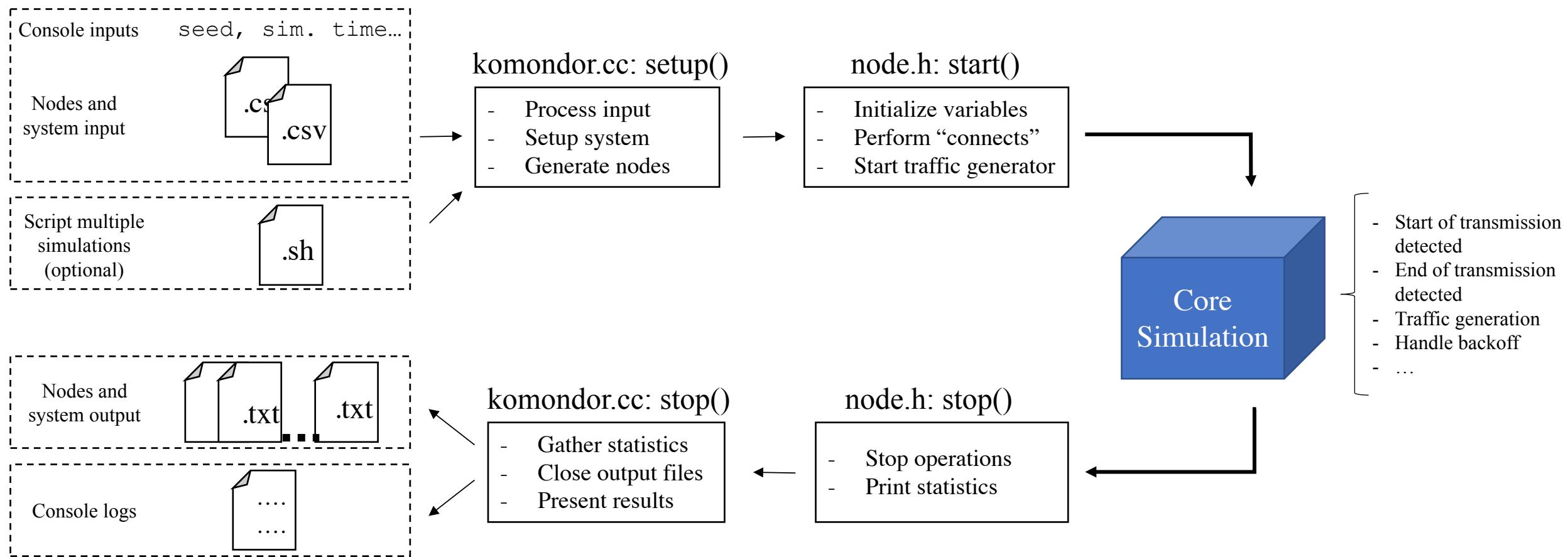


Fig. 1: Component in COST

- Nodes are represented by components
- Message exchange between nodes through *outport/inport* connections
- Timers for triggering events (e.g., end of transmission)

# Overview

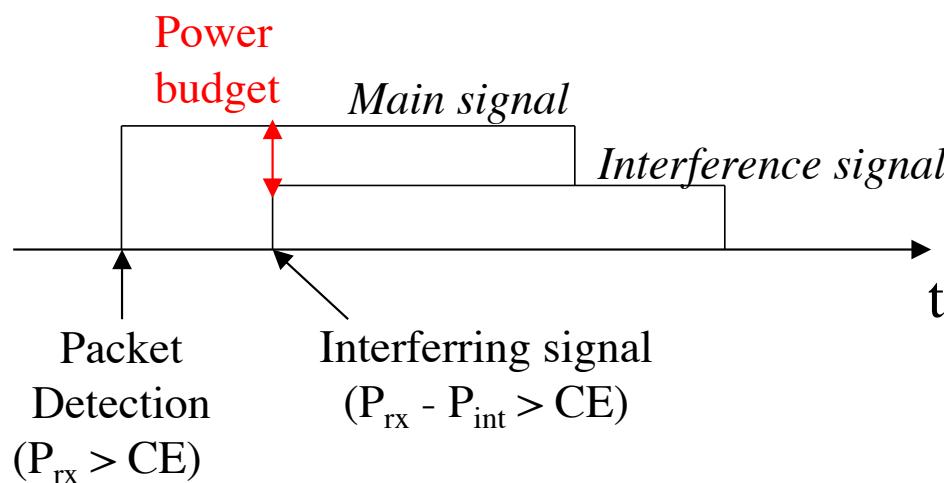


**Fig. 2: Komondor flowchart**

# **IEEE 802.11 Features**

# Channel Access

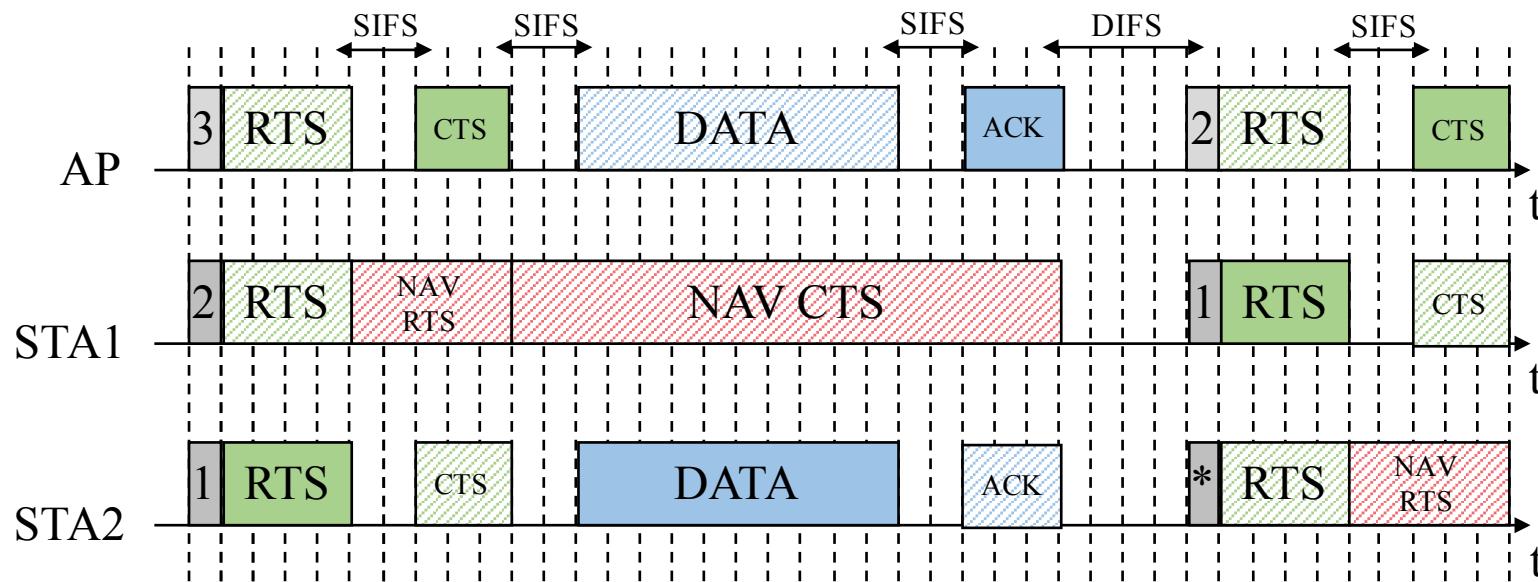
- Distributed Coordination Function (DCF): Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with Binary Exponential Backoff (BEB)
  - Slotted vs Continuous Backoff (BO)
  - Uniform vs Exponential distributions for BO calculation
- Capture Effect (CE): stronger-first approach



**Fig. 3:** Capture Effect approach

# RTS/CTS

- Ready-To-Send/Clear-To-Send (RTS/CTS) is mandatory
- Virtual carrier sensing is done

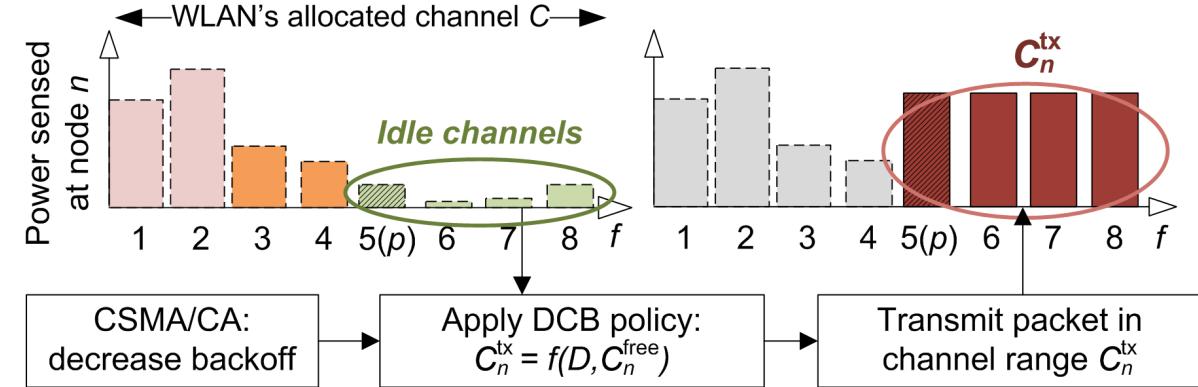


**Fig. 4:** RTS/CTS utilization example

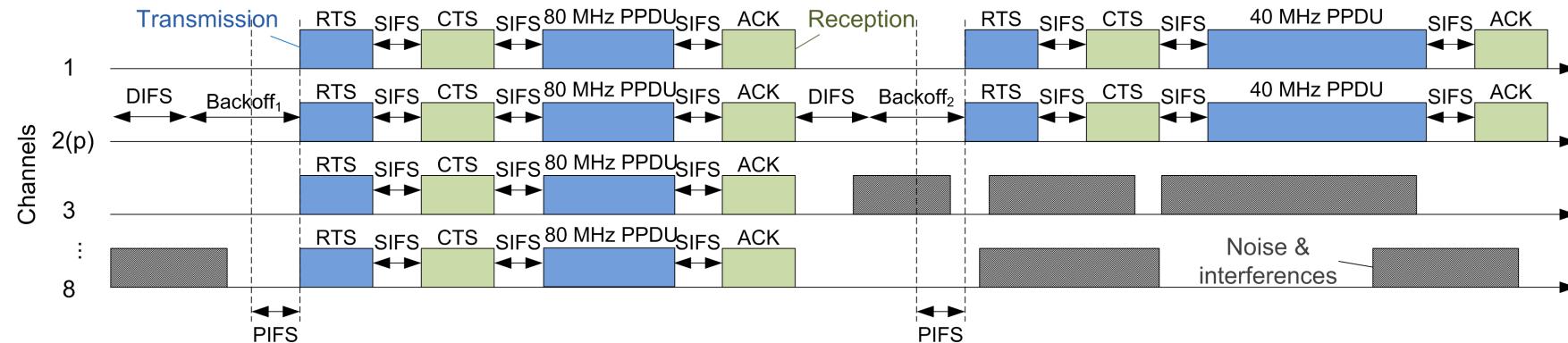
# Channel Bonding

Several implemented policies:

- Only Primary (OP)
- Static Channel Bonding (SCB):
- Always-max (AM)
- Probabilistic uniform (PU)



**Fig. 5:** Channel access when AM is applied



**Fig. 6:** CSMA/CA temporal evolution under the utilization of AM

# Packet Aggregation

- Concatenation of N MPDUs to be sent over the same packet transmission
- Acknowledged by a Block-ACK (BACK)

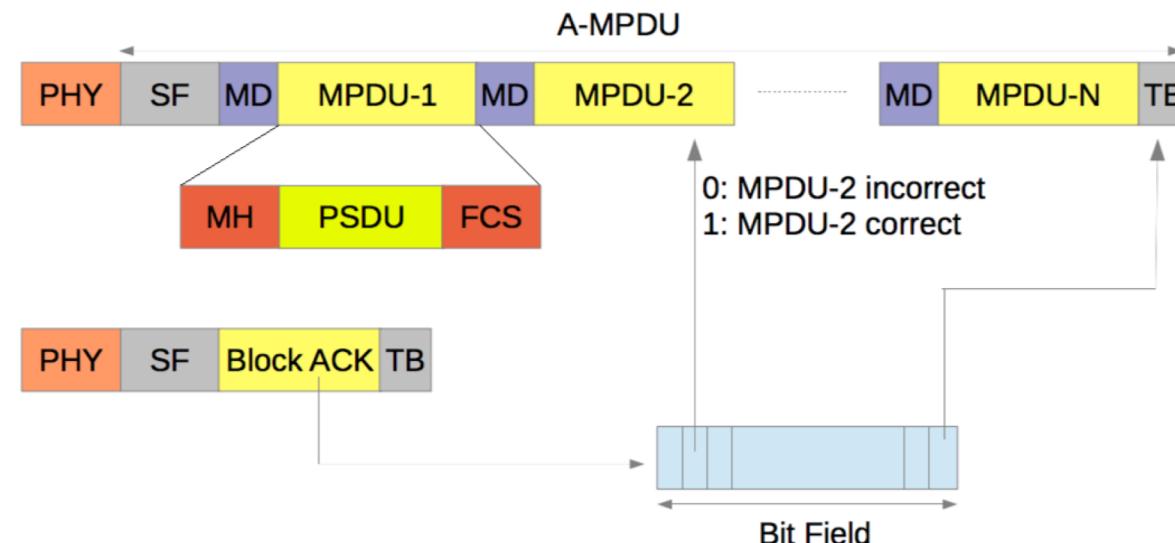


Fig. 7: Packet aggregation for a single transmission

# **Design Principles**

# Architecture based on states

- The IEEE 802.11ax-based Simulator considers several states to capture the behavior of nodes
  - Sensing
  - NAV
  - Transmitting an RTS or a CTS
  - Receiving an RTS or a CTS
  - Transmitting DATA
  - Receiving DATA
  - ...

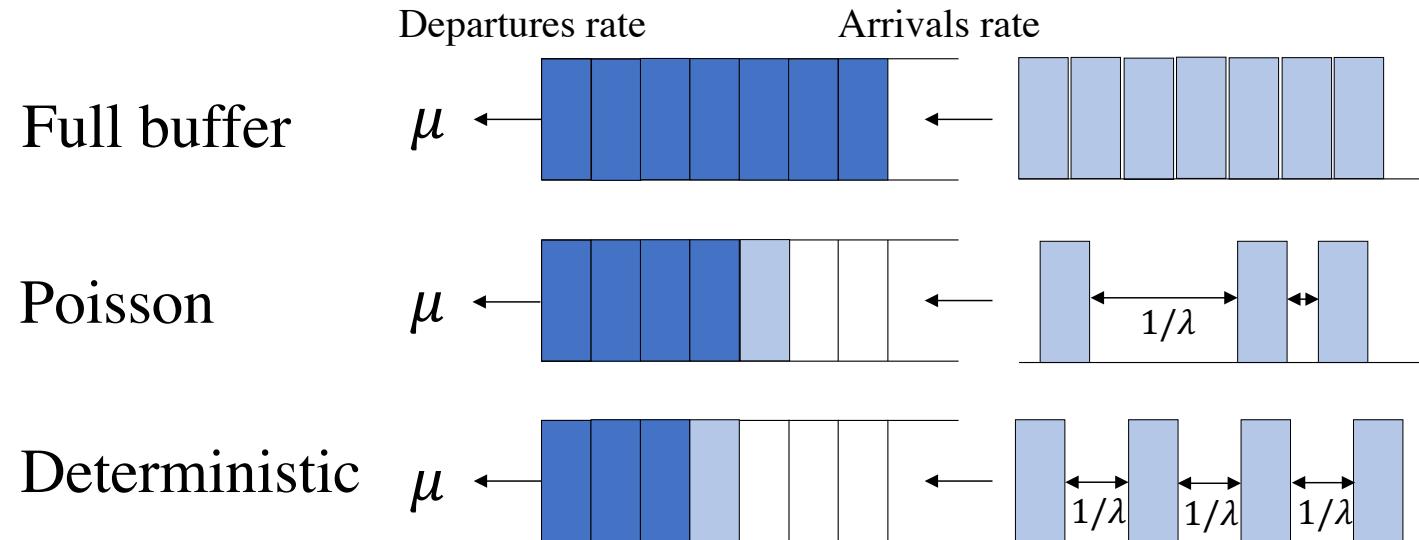
# Channel Modeling

- Several Path-loss models (easy to add new ones):
  - Free Space Path Loss (FSPL)
  - Okumura-Hata model
  - Generic indoor model
  - IEEE 802.11ax residential scenario model
- Different adjacent channel interference models:
  - No interference
  - Limited interference: only adjacent channels leak power
  - Total interference: all the channels leak power (20 dB per channel)

**Assumption:** the incoming power in a given receiver is assumed to be the same during the entire transmission

# Traffic Modeling

Three implemented models for traffic generation:



**Fig. 8:** Traffic generation models

# Link Modeling

- IEEE 802.11ax MCS are used

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	$\geq 52$	1024-QAM	5/6	135	271	567	1143

Tbl. 1: IEEE 802.11ax MCS table

**Assumption:** The MCS is computed at the beginning of the simulation, based on the power received from the transmitter

# Collisions Modeling

- Packet losses occur if:
  - The destination is already transmitting
  - The signal strength is not strong enough ( $P_{rx} < CCA$ )
  - The CE is not accomplished due to interference
  - The destination is already receiving a packet and the CE condition is still accomplished for the first transmission
  - The destination is in NAV
  - Backoff collisions

# **Validations**

# System parameters

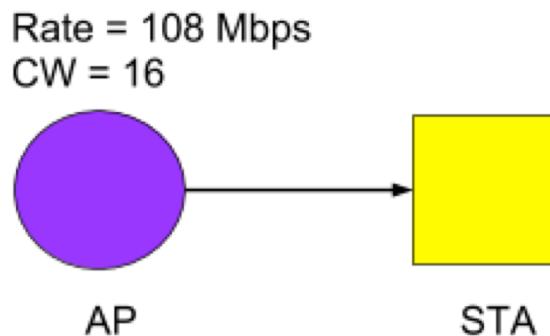
- IEEE 802.11ax parameters are considered
  - Channel access
  - Frames length
  - Channelization
  - ...

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	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
$n_{agg}$	Num. data packets aggregated	64
CE	Capture effect threshold	20 dB
$N$	Background noise level	-95 dBm
$T_{slot}$	Slot duration	9 $\mu$ s
SIFS	SIFS duration	16 $\mu$ s
DIFS	DIFS duration	34 $\mu$ s
PIFS	PIFS duration	25 $\mu$ s
$\eta$	Packet error rate	0.1
$f_c$	Central frequency	5 GHz
$T_{ofdm}$	OFDM symbol duration	16 $\mu$ s
$T_{phy}$	Legacy PHY header duration	20 $\mu$ s
$n_{ss}$	SU spatial streams	1
$T_{phy}^{HE}$	HE header duration	32 $\mu$ s
$L_{sfF}$	Length of MAC's service field	16 bits
$L_{del}$	Length of MAC's MPDU delimiter	32 bits
$L_{mac}$	Length of MAC header	272 bits
$L_{tail}$	Length of MAC's tail	6 bits

Tbl. 2: IEEE 802.11ax PHY & MAC parameters

# Basic Operation

## Scenario



**Fig. 9:** Scenario 1

	Position AP (m)	Position STA (m)	Channel	Transmit power (dBm)	CCA (dBm)
AP	[2,2,0]	-	1	20	-68
STA A	-	[0,0,0]	1	20	-68

**Tbl. 3:** Configuration Scenario 1

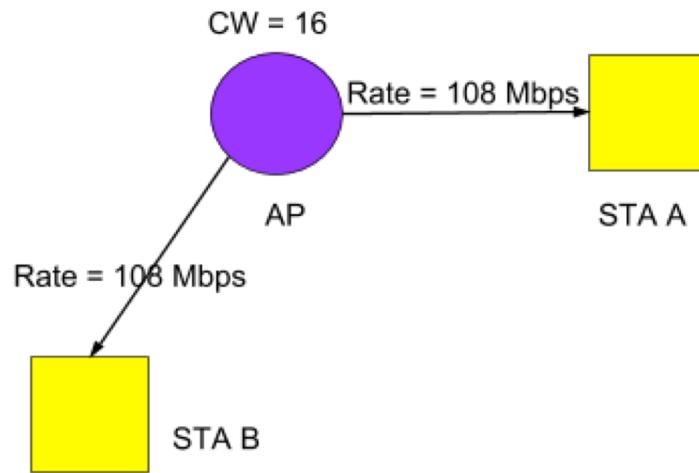
## Results

Packet aggregation (N = 64)		
Komondor	ns3	Bianchi
91.45 Mbps	90.71 Mbps	91.44 Mbps
Single packet (N = 1)		
Komondor	ns3	Bianchi
26.87 Mbps	25.28 Mbps	26.87 Mbps

**Tbl. 4:** Results Scenario 1

# Two STAs

## Scenario



**Fig. 9:** Scenario 2

	Position AP (m)	Position STA (m)	Channel	Transmit power (dBm)	CCA (dBm)
AP	[2,2,0]	-	1	20	-68
STA A	-	[0,0,0]	1	20	-68
STA B	-	[4,4,0]	1	20	-68

**Tbl. 5:** Configuration Scenario 2

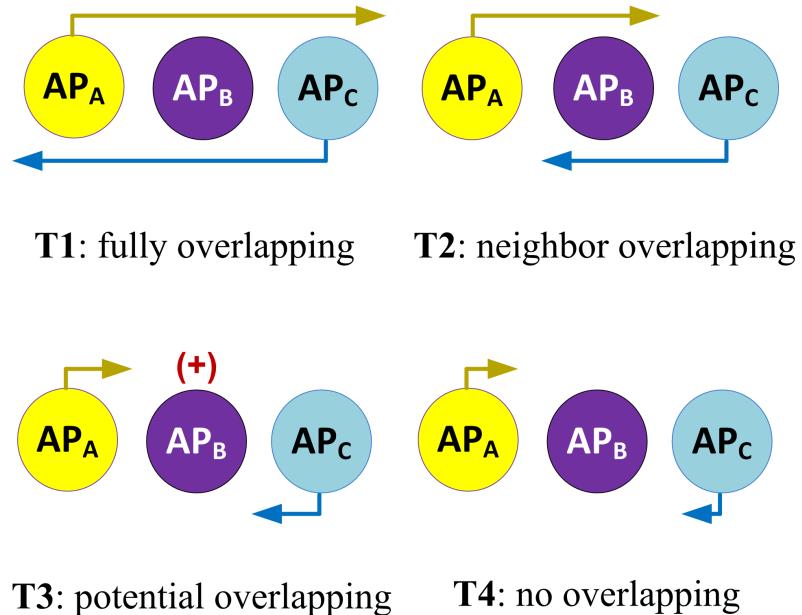
## Results

Packet aggregation (N = 64)		
Komondor	ns3	Bianchi
91.45 Mbps	90.71 Mbps	91.44 Mbps
Single packet (N = 1)		
Komondor	ns3	Bianchi
26.87 Mbps	25.28 Mbps	26.87 Mbps

**Tbl. 6:** Results Scenario 2

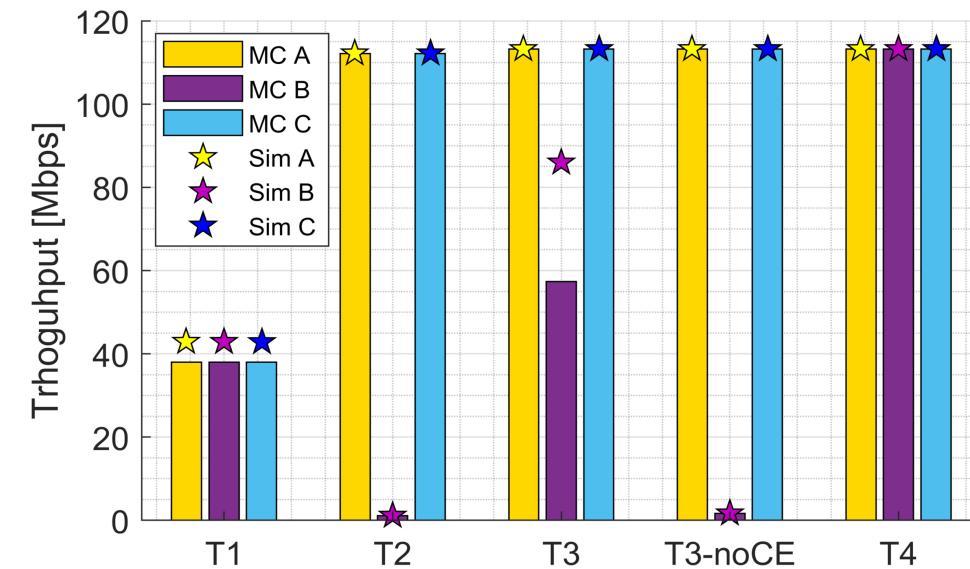
# Dynamic Channel Bonding

## Scenarios



**Fig. 10:** Scenarios for DCB validation

## Results



**Fig. 11:** Results DCB validation

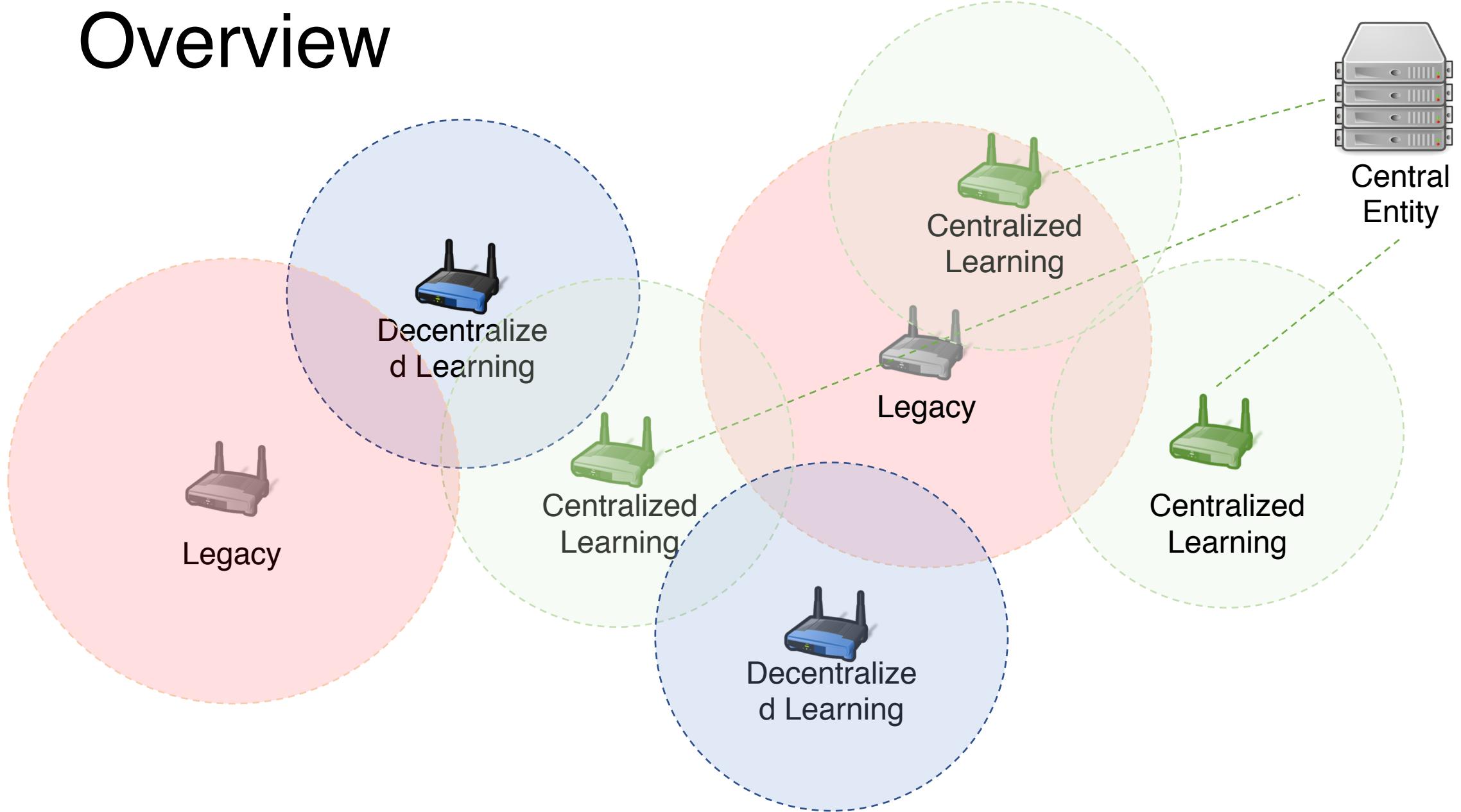
More information in Barrachina-Munoz, S., Wilhelmi, F., & Bellalta, B. (2018). [Performance Analysis of Dynamic Channel Bonding in Spatially Distributed High Density WLANs](#). *arXiv preprint arXiv:1801.00594*.

# **Intelligent Agents**

# Some remarks

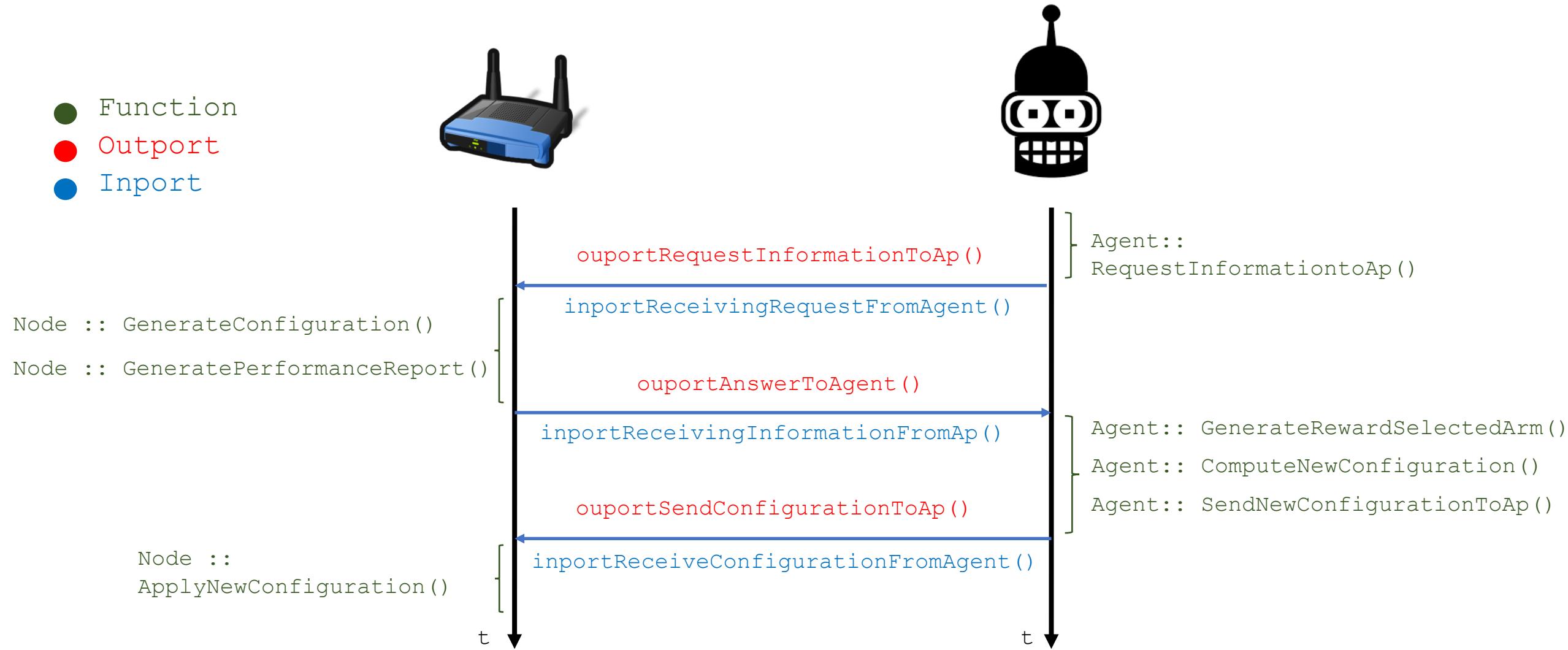
- Agents are introduced to modify the configuration of simulated WLANs in order to maximize a certain performance metric (local or globally)
- Each agent is associated to an AP, so that all the information is shared between these two type of entities (we assume that the agent is located in the AP)
- Communication between agents is not implemented yet, but future developments contemplate centralized approaches
  - Agents share information with a central entity, which takes decisions
  - Adaptive delay for such a communication

# Overview



# AP-Agent communication

- Function
- Outport
- Import



# Agent-oriented functionalities at the AP

1. Wait for agent requests → `inportReceivingRequestFromAgent()`  
`ouportRequestInformationToAp()`
2. Encapsulate important information regarding the current configuration (transmit power, CCA, MCS...) and the offered performance (packets sent, packets lost, collisions by hidden node...) → `GenerateConfiguration()`  
`GeneratePerformanceReport()`
3. Send information to the Agent → `ouportAnswerToAgent()`
4. Wait for the new configuration granted by the Agent →  
`inportReceiveConfigurationFromAgent()`
5. Apply changes and broadcast new configuration to STAs →  
`ApplyNewConfiguration()`

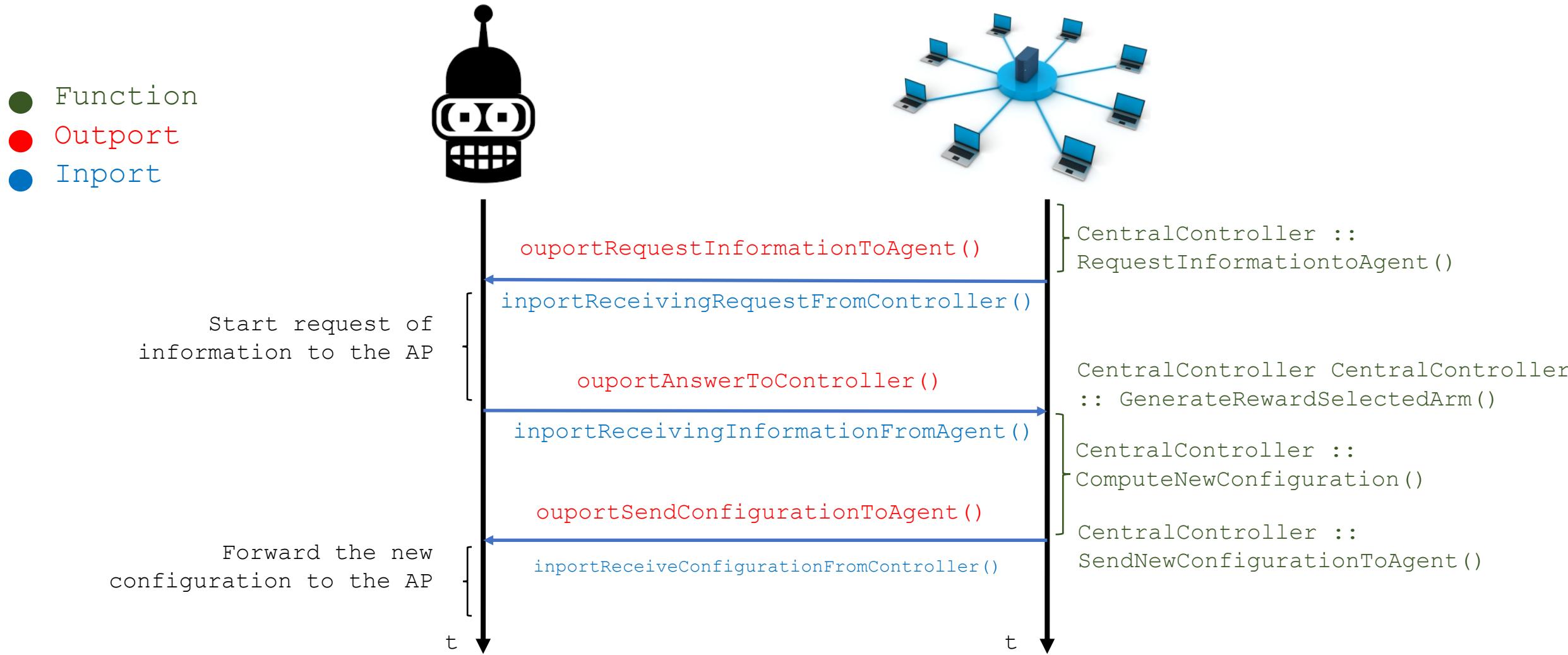
# Agent functionalities

1. Generate and send requests for APs periodically →  
`RequestInformationtoAp()`   `ouportRequestInformationToAp()`
2. Wait for the requested information →  
`inportReceivingInformationFromAp()`
3. Generate a reward according to the current configuration and performance in  
the AP → `GenerateRewardSelectedArm()`
4. Draw a new configuration according to updated rewards →  
`ComputeNewConfiguration()`
5. Send the new configuration to the AP → `SendNewConfigurationToAp()`  
`ouportSendConfigurationToAp()`

# Centralized Learning - Remarks

- In order to perform centralized learning, we have added a central entity that controls the operations done by agents
- The central entity works on the top of the Agent-AP operation
- It is important to remark that agents can be freely assigned to the central entity, so that we can support hybrid scenarios with
  1. Legacy APs
  2. Agent-embedded APs
  3. Centralized agent-embedded APs
- Each agent communicates to the centralized entity in a synchronized manner

# Agent-Central Entity (CC) communication



# Agent-embedded Architecture

