

COMP4336/9336 Mobile Data Networking: 2025 Term 3

Individual Term Project/Assignment

Weighting: 25%

Due: 5pm Fri 14th Nov. 2025 (Week 9)

Version 1.0: Released on 13 October 2025

Project Conditions

- This is an **individual** assignment. The work you submit must be entirely your own work. Submission of work even partly written by any other person or AI is not permitted.
- Do not provide or show your assignment work to any other person, other than the teaching staff of COMP4336/9336.
- Do not publish your assignment solution (data/code) via the Internet. For example, do not place your assignment in a public GitHub repository. Sharing, publishing, or distributing your assignment work even after the completion of COMP4336/9336 is **not** permitted.

Violation of any of the above conditions constitutes academic misconduct. This may result in an investigation, with possible penalties up to and including a mark of **ZERO** in COMP4336/9336, and **exclusion from future studies at UNSW**.

Project Topic: WiFi Protocol Implementation (Version 1.0)

Objective

This project is designed to strengthen your understanding of WiFi protocol mechanisms by engaging with both the physical (PHY) and MAC layers. You will implement simplified simulations for each layer: **for the PHY layer, you may use either Python or MATLAB**, while **for the MAC layer, only MATLAB is allowed**, with provided frameworks and parameters guiding your work.

In addition to coding, you will produce a report that explains your implementation of both PHY and MAC layer, validates your MAC simulation outputs against reference logs, draws timing diagrams to illustrate DCF operation, and analyzes performance metrics under different scenarios. Through this combination of simulation and analysis, you will gain hands-on experience with protocol implementation, structured evaluation, and critical reflection on WiFi design trade-offs.

Project Scope

This project requires students to demonstrate understanding of the WiFi concepts covered in this course, at both the PHY and MAC layers. Students are also expected to learn concepts such as *Additive White Gaussian Noise (AWGN)*, *channel gain*, *equalization*, and *Maximum Likelihood (ML) decoding*.

PHY Layer

The physical layer knowledge required for this project includes:

- **OFDM:** The concept that many narrow subcarriers are used for transmitting multiple symbols at the same time.
- **Modulation and Demodulation:** Schemes such as BPSK, QPSK, 8-PSK, and QAM, with demodulation based on Maximum Likelihood (ML) or minimum distance decision rules.
- **Propagation and Noise Models:** Friis free-space path loss, two-ray ground reflection, and Additive White Gaussian Noise (AWGN) modeling. How to simulate the effect of noise on the received symbol, leading to demodulation errors (symbol error or bit errors).
- **Performance Metrics:** Key measures including Signal-to-Noise ratio (SNR) and Bit Error Rate (BER).
- **Analysis and Visualization:** BER vs. SNR comparisons under different modulation schemes, constellation diagrams, and related physical layer parameter studies.

MAC Layer: Distributed Coordination Function (DCF)

The MAC layer knowledge required for this project includes:

- **CSMA/CA Mechanism:** Carrier sensing and contention-based channel access.
- **RTS/CTS Exchange:** The handshake procedure followed by DATA and ACK transmissions.
- **NAV and Timing:** Maintenance of the Network Allocation Vector (NAV), including inter-frame spaces (DIFS, SIFS) and slot time.
- **Backoff Procedure:** Random backoff selection with contention window (CW) adaptation and decrement by slot time.
- **Retransmission Policy:** Handling of failed transmissions, with packets discarded once retry limits are exceeded.

Learning Outcomes

- Demonstrate deep understanding of how WiFi PHY and MAC layers work
- Develop practical protocol implementation skills
- Analyse key metrics such as SNR, BER, throughput, and fairness to interpret protocol performance.
- Gain insight into key trade-offs that matter in real-world wireless networking.

Project Tasks

The project tasks are divided into **two main parts**: implementation and report writing.

Part I: Implementation (70% of total marks)

Before starting, ensure that MATLAB is installed and running correctly on your computer. You have already used MATLAB in earlier labs, so you should be familiar with the basic interface and workflow. In this part, you will complete simplified WiFi simulations consisting of two components: a **physical layer** model for image transmission (which may be implemented in **MATLAB or Python**), and a **MAC layer** model where you will implement contention-based access (*RTS/CTS/DATA/ACK*) using the supplied **MATLAB** framework. The outputs and logs generated from this part will serve as the basis for the analysis and evaluation tasks in Part II.

1. Physical Layer Simulation (40% of total marks)

In this component, you will model a simple point-to-point link in which station A transmits an image message to station B, focusing on the physical layer aspects of signal transmission.

Your implementation should follow the core principles of the WiFi PHY, by directly mapping the input bitstream into modulation symbols and transmitting them over a simplified channel model. Use an **802.11a-style OFDM grid** with **64 subcarriers over 20 MHz**. Among these, only **48 subcarriers** will be used to carry data symbols (consistent with the 802.11a specification), while the remaining subcarriers are reserved for pilots or left unused. **For simplicity, you do not need to explicitly simulate pilot signals; you may treat them as unused (null) subcarriers.**

Important: even though you only modulate data on 48 subcarriers, all calculations related to channel bandwidth and noise power (e.g., per-subcarrier energy, thermal noise floor) should be based on the full set of **64 subcarriers**.

At least two suitable modulation schemes (e.g., BPSK, QPSK) must be implemented. Within this simplified framework, you are free to organize the simulator in your own way, provided that it captures the essential steps of physical layer operation. The detailed requirements are divided into core tasks and advanced extensions as listed below.

- **Core requirements (sufficient for a passing/high-credit grade):**

- **Channel modeling:** Use a simple propagation model, such as Friis free-space path loss.
- **Modulation and demodulation:** Implement at least two coherent modulation/demodulation schemes (e.g., BPSK, QPSK, 8-PSK). Demodulation can be performed using the minimum-distance decision rule.
- **Error performance measurement:** Evaluate received signal quality in terms of bit error rate (BER), provide per-frame error statistics, and perform comparative analysis such as BER vs. SNR plots across different modulation schemes as well as constellation diagrams.

- **Advanced extensions (for higher marks/distinction):**

- Explore alternative propagation models such as the two-ray ground reflection model.
- Implement a higher-order QAM scheme (e.g., 16-QAM).

- Investigate adaptive modulation: dynamically adjust the modulation scheme in response to varying channel conditions.
- Add coding and decoding (e.g., simple Hamming code) to demonstrate error correction capability.
- Model channel estimation errors to analyze their impact on system performance.

You may make your own design choices, but must **state and justify them clearly in your report**. For this assignment, the PHY tasks are deliberately simplified: it is sufficient to model transmission and reception of modulated symbols over OFDM subcarriers with noise.

A step-by-step workflow is provided below as guidance. It includes a detailed walk-through for the basic implementation and brief suggestions for possible extensions. You are encouraged to refer to these hints, but you are not required to follow them exactly. **What matters most is that you demonstrate your own understanding of PHY layer operation in WiFi communication.**

Suggested Workflow (Basic and Advanced Hints)

In this section, two levels of guidance are provided. The first part gives a step-by-step workflow for the core PHY tasks, and the second part provides brief hints on possible extensions for higher marks.

Basic Workflow

Your main function could be called `simulate_ofdm(params)`. This function is expected to run the entire end-to-end simulation: it should take the system parameters, generate the payload, perform modulation and map symbols onto OFDM subcarriers, transmit the signal through the chosen channel model, add noise, recover the signal at the receiver by demodulation, and finally compute the BER and other performance statistics to evaluate the PHY simulation.

You may also write external scripts to call this function with different parameter settings (e.g., varying transmit power, distance, or modulation schemes) to run experiments and collect results for analysis.

To keep your implementation clean and modular, we recommend dividing the work into helper functions, each handling one stage of the pipeline. While you are free to adapt this structure, note that **code clarity and readability are part of the marking criteria**, so a well-organized design will be to your advantage.

For illustration, our examples use QPSK modulation and the Friis free-space path loss model. In this assignment, you are required to implement at least two modulation schemes (e.g., BPSK, QPSK). To achieve higher marks, you are encouraged to extend your design with advanced options such as higher-order QAM (e.g., 16-QAM) or alternative channel models like the two-ray ground reflection model.

- `friis_received_power`: Compute the free-space path loss (FSPL) using the Friis equation, then calculate the total received power at the receiver (in dBm). The total power is evenly distributed across all subcarriers, and the function returns the FSPL value, the total received power, and the per-subcarrier received power.

- `image_to_bits / bits_to_image`: Convert images into binary bitstreams for transmission and reconstruct images from received bitstreams. The `image_to_bits` function reads an input image, records its size and datatype, and produces a bit sequence representation. **For this simulation, we assume that the receiver already knows the true size and datatype of the transmitted data.** The `bits_to_image` function then uses this metadata to recover the original image.
- `modulate / demodulate` (example shown with QPSK): In this simulation, we make the simplifying assumption that bits are directly mapped to I/Q symbols in the frequency domain, and additive noise is applied to these symbols. The example below uses QPSK, but any consistent scheme is acceptable. The ideal constellation diagrams can be seen in Figure 1.

In QPSK, each bit pair maps to a point in the I/Q plane as follows:

$$\begin{aligned} 00 &\rightarrow (+1, +1), & 01 &\rightarrow (-1, +1), \\ 11 &\rightarrow (-1, -1), & 10 &\rightarrow (+1, -1). \end{aligned}$$

The receiver demodulates by determining the closest constellation point. In this project, you are required to apply **the minimum-distance rule**, i.e., selecting the constellation point with the smallest Euclidean distance to the received symbol.

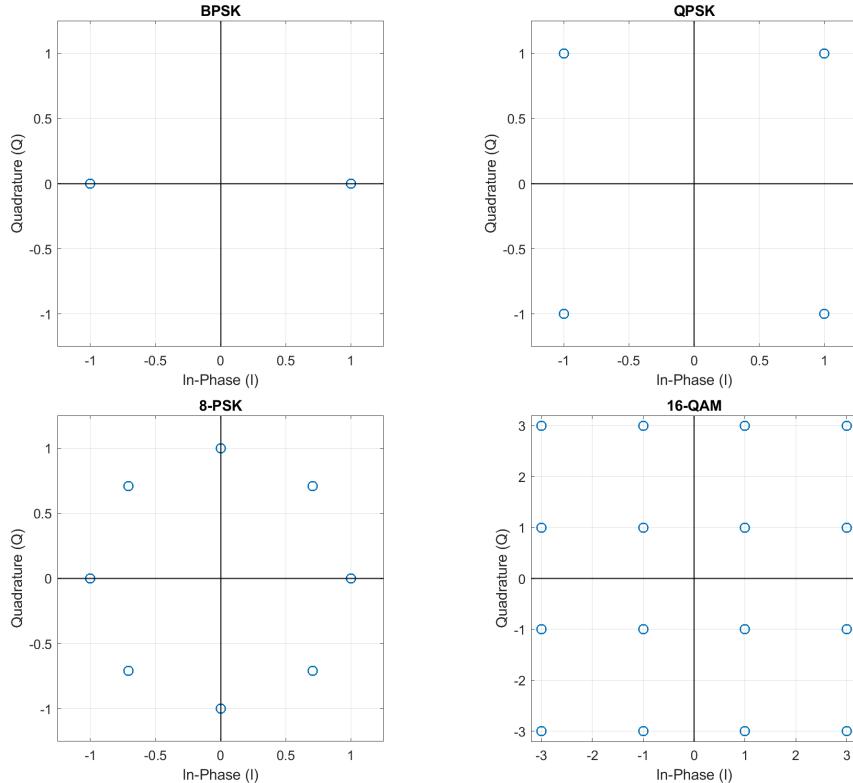


Figure 1: Ideal constellations for different modulations.

- `add_awgn_per_subcarrier`: This step models the effect of an **additive white Gaussian noise (AWGN) channel**, which contributes only random Gaussian noise. The per-subcarrier noise level is converted from dBm to linear power, used to compute the variance, and the resulting complex Gaussian noise is **added directly to the received signal**:

$$Y = X + N, \quad N \sim \mathcal{CN}(0, \sigma^2). \quad (1)$$

Thus, the AWGN channel is represented as a clean transmission path followed by stochastic noise.

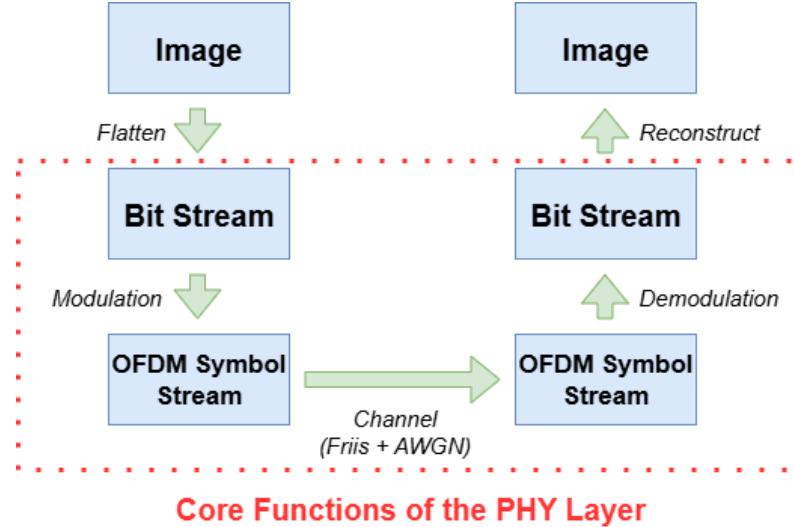


Figure 2: End-to-end logical flow of the OFDM simulator.

Putting these functions together, your simulator should follow the logical flow shown in Figure 2. By structuring your implementation in this modular way, you will produce clear, testable code while also gaining insight into the role of each component in the physical layer.

Extensions for Higher Marks

- **Alternative propagation models:** As an extension, you may replace the Friis free-space model with a two-ray ground reflection model. Use the same assumptions as in the basic case, but modify the received power calculation according to the two-ray principle.
- **Higher-order QAM (e.g., 16-QAM):** Unlike phase-only schemes such as BPSK or QPSK, QAM varies both the amplitude and phase of the signal. For example, if you try 16-QAM, you will need to handle multiple amplitude levels along each axis, which makes the mapper/demapper design more involved than in QPSK. You can also consult standard constellation definitions if you want to explore other QAM schemes.
- **Adaptive modulation:** In realistic wireless systems, the choice of modulation is not fixed but depends on the channel quality. As an extension, you may investigate adaptive modulation, where the system dynamically selects between different modulation schemes depending on the observed SNR. This highlights the trade-off between robustness (lower-order modulation at low SNR) and efficiency (higher-order modulation at high SNR), and allows you to demonstrate how link adaptation improves performance.

- **Error correction coding:** Beyond simple modulation and demodulation, many systems employ forward error correction to improve reliability. As an extension, you may add coding and decoding functions (for example, a simple Hamming code) around the modulation stage. This allows you to compare uncoded and coded BER curves, and to analyze how coding gain reduces the required SNR for a given error rate.
- **Channel estimation errors:** In practice, the receiver does not have perfect knowledge of the channel response. You may model channel estimation errors by perturbing the estimated channel coefficients with random noise or bias, and then using this imperfect estimate for equalization. This extension allows you to analyze how estimation inaccuracy degrades demodulation performance, and to compare ideal versus imperfect channel knowledge scenarios.

Result Visualization

Once your end-to-end simulation is complete, you should include visualization results in your report. These figures are an essential part of presenting your work, as they help verify correctness and illustrate the impact of different channel and modulation settings. In the following, we outline several forms of visualization **that you are expected to show**.

Input image selection

In your report, include the input image you use for testing. Pick one that is colorful and information-rich, but not too large, to keep simulation time reasonable. A well-chosen image makes the later visualizations clearer and more meaningful.

By also showing the received image after transmission, you can demonstrate how parameter adjustments (distance, for instance) influence the SNR, which in turn directly impacts the perceived image quality. The images below (Figure 3), generated from the QPSK-based simulation, illustrate this effect.



Figure 3: Input and received images under different SNR (resolution: 1280×720 , size: 22,118,400 bit).

BER vs. SNR curves

Another essential visualization is the bit error rate (BER), defined as **the ratio of erroneously received bits to the total number of transmitted bits**, plotted as a function of SNR. In your report, include BER vs. SNR plots for at least two different modulation schemes (e.g., BPSK and QPSK). These plots form the basis for discussing how modulation choice affects system performance under different channel conditions. Figure 4 shows an example format of such a plot.

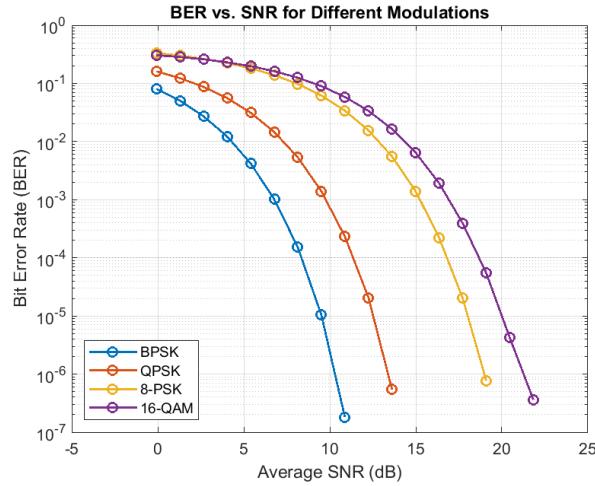


Figure 4: BER vs. SNR for different modulation schemes.

Constellation diagrams

Constellation plots are another useful way to visualize system performance. By displaying the received symbols on the I/Q plane under different SNR values, you can see how noise affects the spread of points around their ideal locations. In your report, include constellation diagrams at multiple SNR levels and discuss what they reveal about the reliability of symbol detection. Figure 5 shows an example format of such a plot. **Remember to use post-equalization when doing so.**

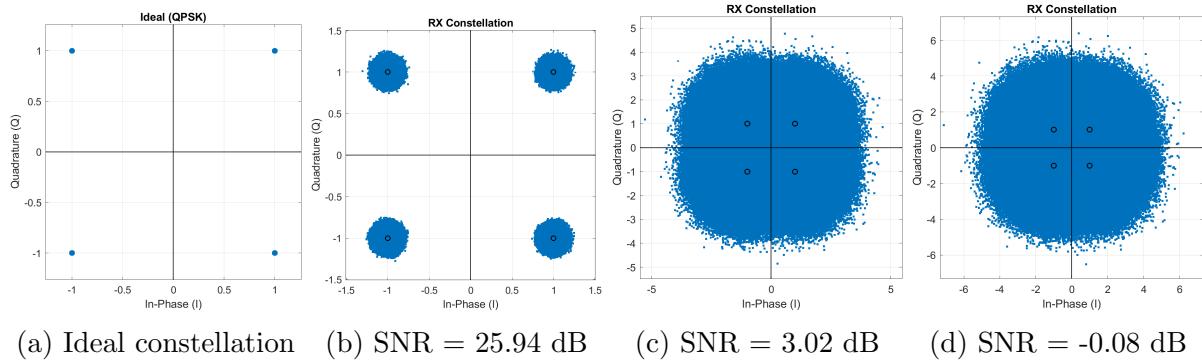


Figure 5: Constellation diagrams at different SNR levels.

Beyond the required visualizations, you are encouraged to include additional plots or analyses in your report. These figures should be accompanied by clear discussion and analysis of the underlying data, rather than presented without explanation. **A more detailed and insightful discussion, supported by simulation data, will be taken into account when assessing the quality of your work.**

2. MAC Layer Simulation (30% of total marks)

In this part of the assignment, you will complete a simplified simulation of the WiFi MAC protocol in MATLAB. The scenario has two stations (A and B) contending to send data to a common access point (C). The access mechanism follows the Distributed Coordination Function (DCF) with RTS, CTS, DATA, and ACK, together with backoff and carrier sensing. The provided framework also models NAV updates and probabilistic energy detection and frame decoding.

You are provided with a partially implemented framework `MAC_Layer_Simulation` that already handles WiFi state logging and the overall control flow. Your task is to fill in the missing functions in the `mac.student` file so that the simulation reproduces the expected channel access behavior. **You must not modify the logging code or the random number setup, since grading will compare your printed logs and generated CSV files with reference outputs.**

By completing this part, you will gain hands-on experience with MAC operations such as contention, collision handling, NAV based deferral, timeout and retry control, and fairness measurement.

Simplifying Assumptions (aligned with the framework)

Our simulator is a teaching-oriented abstraction of IEEE 802.11 DCF. To keep the scope manageable and match the provided code, we make the following assumptions:

- Topology: two senders (A, B) contend for one receiver (C). By default, all nodes are within mutual carrier-sense range, though hidden-terminal effects can also be explored.
- Time abstraction: slot-timed model with no propagation delay; events occur at slot boundaries.
- Medium perception and decoding are probabilistic as configured in the framework (energy detection and frame decoding probabilities). The simulator does not include a capture effect model, meaning that if two transmissions overlap in time, they always result in a collision rather than one being successfully decoded.
- PHY effects (noise, fading, waveform-level collisions) are abstracted into the above probabilities; collisions occur when transmissions overlap in time.
- RTS/CTS is used for all DATA transmissions. Control-frame durations (RTS/CTS/ACK) are deterministic under fixed configs; DATA duration is determined by payload size and the selected MCS (no rate adaptation).
- Retransmissions use a simple retry limit; no fragmentation/aggregation or Block ACK is modeled.
- NAV is updated only when the relevant control/data frames are (probabilistically) decoded per the framework's rules.
- Goodput is measured as successfully ACKed payload bytes over the simulation time; fairness is evaluated using Jain's index on station goodputs.

Fairness Metric: Jain's Index

Besides total throughput, we also care about how fairly the channel is shared between stations. A widely used metric is **Jain's Fairness Index**, which gives a number between 0 and 1:

$$J = \frac{(\sum_{i=1}^n x_i)^2}{n \cdot \sum_{i=1}^n x_i^2},$$

where x_i is the goodput of station i , and n is the number of stations.

- $J = 1$: perfectly fair (all stations achieve equal goodput).
- $J \rightarrow 0$: highly unfair (one station dominates).

Examples:

- If A = 5 Mbps and B = 5 Mbps, then $J = 1$ (completely fair).
- If A = 9 Mbps and B = 1 Mbps, then $J \approx 0.61$ (unequal share).

At the end of each run, the simulation framework automatically computes Jain's index based on the goodputs of A and B¹. The resulting value (provided by `calc_overlap_fairness`) is available for you to use in your report for further analysis and discussion.

¹https://en.wikipedia.org/wiki/Fairness_measure

What to Implement

Complete the placeholders in `mac.student`. Each function corresponds to a core element of DCF. Read the provided framework carefully to understand when and how each function is called.

Contract and constraints:

- Follow the given function interfaces strictly: all required inputs are already provided in the framework and function headers, and your code must return outputs exactly as specified.
- Do not modify existing structs (`sta`, `cfg`, `Metrics`, `onair`); you may update the values of existing fields, but must not add new fields.
- Do not alter logging strings or random number seeds; grading relies on exact log and CSV matches, and reproducibility of results.
- Use only framework-defined constants and timing variables; do not redefine them.

Functions to Complete

The **six** functions to complete are:

- `update_backoff(sta, DIFS_slots, waiting, underNAV, sensedBusy)`: Update the backoff counter based on medium state and detect when it reaches zero.
- `can_attempt_rts(Metrics, SlotTime)`: Decide if a station is eligible to start an RTS in the current slot.
- `cw_update_after_event(sta, event, CWmin, CWmax)`: Update contention window and retry counter after transmission events.
- `navUntilFor(ftype, end_slot, lenCTS, lenDATA, lenACK, SIFS)`: Compute until which slot NAV remains asserted after hearing a frame.
- `compute_wait_deadline(txType, start_slot, len_slots, SIFS_slots, CTS_slots, ACK_slots)`: Compute the slot index until which a station waits for CTS or ACK.
- `handle_drop_and_advance(sta, CWmin)`: Drop a packet after exceeding retry limits and advance to the next one if available.

How to Run the Simulation

To run the simulator, first paste the provided `test.txt` file to the `data` folder, then navigate to the `scripts` folder and execute the provided script.

```
>> runSim
```

This script:

- Sets up paths and builds the configuration struct internally.
- Calls the framework to perform the simulation.
- Produces results in `output/out_<tag>/`, including:
 - `log.txt`: detailed slot-by-slot events.
 - `metrics_summary.csv`: key throughput and fairness results.
 - Additional CSV files containing traces of delays, contention windows, and other per-node statistics.

Provided Test Cases

We provide two parameter settings and their expected logs for self-checking. You can run them using `runSim.m` by adjusting the configuration, while final grading will use hidden configurations. The required input files are located under the project's `data` directory. The details of the two provided settings are summarized in Table 1. For any parameters not listed in the table, please keep their default values as provided in the downloaded framework without modification.

	Test 1 (Baseline)	Test 2 (Asymmetry)
Transmitted file (A,B)	(test.txt, test.txt)	(test.txt, test.txt)
RNG seed	42	42
Slot time	9 μ s	9 μ s
SIFS	16 μ s	16 μ s
Retry limit	4	4
CWmin / CWmax	15 / 1023	15 / 1023
MTU (A,B)	(200, 200)	(400, 200)
$P_{C,A}$ (RTS/DATA)	1.0 / 1.0	0.9 / 0.8
$P_{C,B}$ (RTS/DATA)	1.0 / 1.0	0.9 / 0.9
$P_{A,B}$ (RTS/DATA)	1.0 / 1.0	0.9 / 0.9
$P_{B,A}$ (RTS/DATA)	1.0 / 1.0	0.5 / 0.5
A: CTS (self / side)	1.0 / 1.0	1.0 / 0.9
B: CTS (self / side)	1.0 / 1.0	1.0 / 0.9
A: ACK (self / side)	1.0 / 1.0	1.0 / 0.9
B: ACK (self / side)	1.0 / 1.0	1.0 / 0.9
Sensing prob. (A,B)	(1.0, 1.0)	(0.9, 0.9)

Table 1: Two provided parameter settings for self-checking.

Part II: Report and Evaluation (30% of total marks)

In addition to completing the simulation code, you are required to write a report (maximum 10 pages) that documents your design, validation, and analysis. The report should be written in a clear and concise style, with a logical structure that follows the tasks you implemented. If you have large figures or illustrations, **you may include them in an Appendix, which will not count towards the 10-page limit.**

Expected Content

Your report should contain the following elements:

- **Code Logic Explanation:** Describe how you implemented the required functions in both PHY and MAC parts. Focus on the reasoning and pipeline (how the main simulator calls helper functions, how state is updated), rather than reproducing code listings.

Important: We will review your code guided by your written description. Provide careful, detailed justification of your implementation choices—especially in the

PHY simulation. Insufficient, vague, or missing explanations will **directly incur mark deductions**.

- **MAC Layer Self-Designed Scenario and Log Output:** Design one or more additional test scenarios of your own (beyond the provided test cases), run your MAC simulator, and include excerpts of the generated `log.txt`. Clearly state the parameter values you chose (e.g., payload sizes, sensing probabilities, retry limits), and explain what specific DCF behavior the scenario is intended to highlight. Examples include: observing the exponential backoff process after repeated collisions, showing how sensing errors lead to unnecessary backoff growth, or illustrating channel access asymmetry when one station consistently wins. For each scenario, briefly explain how the observed log lines demonstrate the expected DCF behavior.
- **DCF Timing Diagram:** Draw a timing diagram that illustrates channel access in one of your scenarios. **You may follow the style shown in the lecture slides on IEEE 802.11 Basics**, using time along the x-axis and marking RTS, CTS, DATA, ACK, backoff, and idle slots.
- **Metrics-based Analysis – PHY Simulation:** As outlined earlier, your report must include the required PHY visualizations (i.e., received images, BER vs. SNR plots, constellation diagrams). In addition, you are encouraged to incorporate further figures or tables of your own, using results from the simulator to explore how performance changes with parameters such as distance, transmit power, antenna gains, or modulation scheme. A thorough discussion that combines these visualizations with clear interpretation of the underlying trends will be a key factor in the assessment.
- **Metrics-based Analysis – MAC Simulation:** Using the outputs produced by your simulator (`log.txt`, `metrics_summary.csv`, `cw_trace.csv`, and per-station delay files), design and analyze scenarios of your choice. Depending on what you wish to highlight, you may focus on throughput/fairness (from `metrics_summary.csv`), timing and access behavior (from `log.txt`), contention dynamics (from `cw_trace.csv`), or per-packet delays (from the delay files).

As an example, Figure 6 contrasts two configurations used for *log testing*. In this example, we provide two sample parameter sets: besides parameter differences, their random seeds are also not identical. However, when you design your own experiments, you should **control variables** carefully by fixing a particular set of random seeds.

The left plot shows contention-window (CW) traces over time (per station), and the right plot shows *goodput*, i.e., the rate of **successfully delivered payload bits** over wall-clock time. These are only examples—you are free to choose other metrics/plots, but strive for a rich and comprehensive analysis.

Note on Randomness. Because both the PHY and MAC simulations involve random processes (e.g., noise realisations, random backoff, probabilistic sensing/decoding), the outcome of a single run may not strictly follow the expected trend. For example, in a single trial, one station may dominate due to luck. **A reasonable approach in such cases is to repeat the simulation with different random**

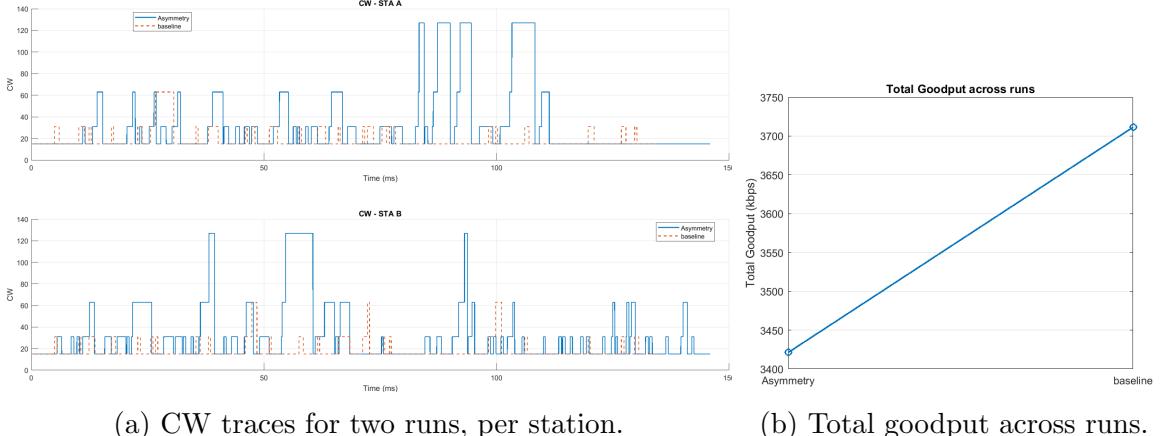


Figure 6: Example MAC-level visualizations: contention dynamics (left) and throughput outcome (right).

seeds and average the results, which smooths out random variation and reveals the underlying protocol behaviour.

- **Conclusion:** Summarise what you have learned about WiFi operation from building and analyzing these simulations (Both PHY and MAC layers). Your conclusion should not be generic: it should be grounded in the results you obtained. Reflect on the trade-offs observed between throughput, delay, error rate, and fairness. For example, you might comment on:
 - How increasing payload size can raise throughput but also leads to higher average delay.
 - How asymmetric channel conditions between A and B reduce Jain's fairness index.

Marking Rubric

Task	Criteria
Physical Layer Simulation (40%)	<p>Excellent (85–100): Complete PHY pipeline with modular code. Mandatory outputs (BER vs. SNR for ≥ 2 modulations including at least one QAM, constellation diagrams at multiple SNRs, and input/output image examples) are clear and critically analyzed. Extensions (higher-order QAM, two-ray ground reflection, adaptive modulation, error correction and channel estimation errors) are implemented and explained. Code is reproducible and well documented.</p> <p>Average (50–84): Core PHY runs correctly with required plots, but analysis shallow or extensions absent. Code functional but less clear.</p> <p>Poor (0–49): Major parts missing/incorrect, plots unclear or absent, code unreliable.</p>
MAC Layer Simulation (30%)	<p>Excellent (85–100): Implements CSMA/CA with RTS/CTS, back-off, NAV, and retries. Logs/CSVs match DCF behavior. At least one extra scenario with parameters, log excerpts, and explanation. Code clear and reproducible.</p> <p>Average (50–84): Most MAC functions present but partly wrong/missing. Logs/CSVs generated but inconsistent. Extra scenario attempted with limited discussion.</p> <p>Poor (0–49): Incomplete/incorrect MAC, missing or invalid logs, or code fails.</p>
Report and Evaluation (30%)	<p>Excellent (85–100): Well-structured report (≤ 10 pages) with required figures/analysis. Explanations clear, extra metrics used, figures labeled and referenced, conclusions concise.</p> <p>Average (50–84): Covers most elements but shallow analysis or unclear figures; conclusions generic.</p> <p>Poor (0–49): Disorganized or superficial, key parts missing, figures unclear, or conclusions absent.</p>

Submission Instructions

Submit your work via Moodle before the deadline. Your submission must include:

- A single `zip` archive containing only your MATLAB/Python code:
 - For the PHY part, the complete simulation pipeline you were asked to build.
 - For the MAC part, **only the required functions**.
 - Any small helper scripts you wrote for your own data analysis or plotting.

Do not include the large output folders.

- A single PDF report (maximum 10 pages) following the guidelines in Part II.

Both files should be named with your student ID (e.g., `z1234567_proj.zip`, `z1234567_proj.pdf`). Your code must run correctly with the provided `runSim.m` script without modification.

End of Project Specs