Network Design Project – Phase Proposal & Design Document (Phase 1 of 5)

> **Purpose:** This document is your team's *proposal* for how you will implement the current phase **before** you start coding.
> Keep it clear, concrete, and lightweight.

Team Name: Virtual team
Members: Zach Garnes, zachary_garnes@student.uml.edu, Lucas Lomba, Lucas_Lomba@student.uml.edu, Jonathan Rubio, Jonathan_Rubio@student.uml.edu
GitHub Repo URL (with GitHub usernames): https://github.com/zgarnes-uml/Garnes_Phase1.git (zgarnes-uml)
Phase: 1
Submission Date: 1/26/2026
Version: v1

---

0) Executive summary

In Phase 1, the goal is to introduce everyone to python, git, and the fundamentals of networking. In part one, the goal is to implement the standard user datagram protocol (UDP) sockets by creating a UDP server and client to send the text "Hello". The client will send the text "Hello", once received the server will echo the same text by sending it back to the client. In part two, the goal is to implement RDT 1.0 protocol to transferring a file from the UDP client to the UDP server. This will be done by each team member individually.
—

1) Phase requirements

1.1 Demo deliverable

You will submit a **screen recording** demonstrating the required scenarios.

- **Private YouTube link:** *(fill in at submission time)*
  - Link:
  - Timestamped outline (mm:ss → scenario name):

1.2 Required demo scenarios

Fill in the scenarios required by the phase spec.

Scenario	What you will inject / configure	Expected observable behavior	What we will see in the video
1	Run UDP server and client python code	Server client will wait for message from client, client will send "hello",server will receive message and send it back to client	Two terminal windows, one that is the server and one that is the client. Client window will send the text "hello". Server window will receive "hello" and then send the text back to the client. Client window will say hello.
2	Run UDP server and client python code	Server client will wait for file from client, client will take the transfer file (an image) and break it up into packets, client will send packets one by one,server will receive packets one at a time and create a new file that will be the same image as what was transferred.	Two terminal windows, one that is the server and one that is the client. Client window will use a .bmp file in the folder of the project and break it up into packets to send to the server. Server window will receive the packets, saving them into a new .bmp file. The new file will be the same image and the image in the file that was already existing in the folder, showing the data was sent from the client to the server.

---

2) Phase plan (company-style, lightweight)

2.1 Scope: what changes/additions this phase

- **New behaviors added:** UDP server and client to send text. UDP server and client to send files.
- **Behaviors unchanged from previous phase:** N/A
- **Out of scope (explicitly):** N/A

2.2 Acceptance criteria (your checklist)

List 5–10 measurable checks that mean you're done (examples below).

☐ Sender/receiver run echoing the text correctly
☐ All required scenarios demonstrated in the video
☐ Output file matches input file (byte-for-byte)

☐ Image is recreate correctly
☐ Being able to send the same file multiple times

---

3) Architecture + state diagrams

The architecture for part 2 will follow the RDT 1.0 protocol described in Section 3.4.1 of the course textbook. See below image.

3.1 How to evolve the provided state diagram

image ### 3.2 Component responsibilities - **Sender** - responsibilities: take file, break it up into packets, send to server. - **Receiver** - responsibilities: wait for packets, assemble all packets into a new file, save file.

3.3 Message flow overview

```
[image] -> Sender -> UDP -> Receiver -> [output image]
```

---

4) Packet format (high-level spec)

Define your on-the-wire format **unambiguously**.

4.1 Packet types

List the packet types you will send: - Data packet - ACK packet - (Optional) end-of-transfer marker / metadata packet

4.2 Header fields (this is the "field table")

What this means: you must specify the *exact* fields in each packet header and their meaning.
This ensures everyone can encode/decode packets consistently.

Field	Size (bytes/bits)	Type	Description	Notes
seq	4	Unsigned 32-bit integer	sequence number	
len	4	Unsigned 32-bit integer	payload length	last packet may be smaller

Field	Size (bytes/bits)	Type	Description	Notes
total	4	Unsigned 32-bit integer	total number of packets	
payload	~1024B	bytes	file chunk	binary-safe

---

5) Data structures + module map

This section prevents "random globals everywhere" and helps keep code maintainable.

5.1 Key data structures

List the core structures you will store in memory

Server IP

Server port

Client IP

Client Port

Payload size

---

6) Protocol logic (high-level spec before implementation)

This section is your "engineering spec" that you implement against. Keep it precise but not code-heavy.

6.1 Sender behavior

Describe behavior as steps or a state machine: - when packets are sent - retransmission rules - termination conditions - (if applicable) window advance rules

Sender pseudocode (recommended):

```
initialize state
while not done:
  send/queue packets according to phase rules
  if ACK received:
    validate (seq)
    update state (advance)
  if timeout/event:
    retransmit according to phase rules
```

6.2 Receiver behavior

Describe receiver rules: - accept/discard conditions - duplicate/out-of-order handling - file write rules (safe and deterministic)

Receiver pseudocode (recommended):

```
on packet receive:
  if corrupt: discard; respond according to phase rules
  else if expected: accept; write/buffer
  else: handle duplicate/out-of-order according to phase rules
```

---

7) Repo structure + reproducibility

Your repo must contain at minimum:

```
src/
scripts/
docs/
results/
README.md
```

State where phase artifacts live: - Design docs: `docs/` udp_client.py, udp_server.py, rdt1_client.py, rdt1_server.py - Figures/plots + CSV: `results/` image.bmp, received.bmp, state diagram.PNG

---

10) Team plan, ownership, and milestones

10.2 Milestones (keep it realistic)

- Milestone 1: Completed the whole phase 1/30/2026

Appendix (optional)