

Grid Clash Multiplayer Synchronization Protocol — Project Proposal

Course: CSE361 — Computer Networks

Submitted to:
Dr. Karim Ahmed
Eng. Noha Wahdan

Submitted by :
Loaiy Mahmoud 23P0419
Omar Ayman23P0244
Ali Moataz 23P0280
Omar Osama 2300090
Jana Tamer 23P0173
Jana Mohamed 2301032

1. Introduction & Motivation

This project focuses on designing and implementing a custom application-layer protocol for low-latency multiplayer game synchronization.

We selected **Project 2: Multiplayer Game State Synchronization** from the course options, aiming to build a lightweight and efficient protocol over UDP that maintains consistent, real-time state among multiple clients connected to a central game server.

In modern online games, real-time synchronization is critical to ensure a smooth experience despite network imperfections such as delay, packet loss, or jitter. The goal of this project is to study these challenges by designing the **GridClash Binary Protocol (GCBP v1)** — a compact binary protocol optimized for speed and reliability in a competitive grid-based environment.

Objectives:

- Achieve sub-50 ms latency for real-time updates.
- Maintain synchronization consistency across 4 concurrent clients.
- Tolerate 5% packet loss without perceptible gameplay disruption.

2. Project Description

The game, **Grid Clash**, features a shared 2D grid (e.g., 20×20) where players compete to claim cells by clicking on them. Each cell can be **unclaimed**, **pending**, or **owned** by a specific player. The **server** acts as the authoritative entity, resolving conflicts (when multiple players claim the same cell) and ensuring that all clients display consistent state updates.

The focus of this project is not on graphics or complex gameplay, but on **network performance** and **synchronization accuracy**. Each client communicates with the server using UDP messages defined by our custom protocol. The system must support **at least four concurrent players** while maintaining **average end-to-end latency ≤ 50 ms** and remaining stable under **2–5% packet loss**.

3. Proposed Protocol Design (GridClash Binary Protocol)

The **GridClash Binary Protocol (GCBP v1)** is a UDP-based binary communication protocol that defines compact headers and structured message types for real-time updates.

Each message includes:

- Protocol ID (GRID)
- Version number
- Message type (e.g., SNAPSHOT, ACQUIRE_REQUEST)
- Sequence number
- Timestamp
- Payload length
- Optional checksum (CRC32)

Reliability Model:

Critical events such as cell acquisitions use **selective reliability** (ACK/NACK), while periodic updates (e.g., state snapshots) rely on **redundancy** and **sequence tracking** to handle loss gracefully. This hybrid approach provides a balance between latency and robustness.

Message Types (examples):

- CONNECT_REQUEST — initiate handshake
- WELCOME — server acknowledgment
- ACQUIRE_REQUEST / RESPONSE — cell claim and result
- GAME_STATE — broadcast of full grid state
- HEARTBEAT — keep-alive
- ACK / NACK — selective reliability control

The design prioritizes **compact headers (< 20 bytes)** to minimize bandwidth and transmission overhead, ensuring real-time operation on typical LAN/Wi-Fi networks.

4. Implementation Plan

The system will be implemented entirely in **Python 3** using standard networking and threading libraries. The following core modules are included:

Module	Function
server.py -----	Manages player registration, game state, and broadcasts periodic snapshots.
client.py -----	Handles user input, sends game actions, receives and applies state updates.
protocol.py -----	Defines binary header structure, message constants, and encoding/decoding.
run_baseline_test.py ---	Automates baseline test runs for reproducibility and logging.
game_state.py -----	Maintains grid representation and player score logic.

The architecture follows a **modular design**, enabling easy debugging, extension, and performance measurement. The code runs on both Linux and Windows systems, ensuring compatibility with the grading testbed.

5. Testing & Evaluation Plan

Performance and reliability will be evaluated using **Linux netem** to simulate network impairments (delay, loss, jitter).

The test automation script will record and analyze the following metrics:

- **Latency (ms)** — average and 95th percentile
- **Jitter (ms)** — variation in inter-arrival times
- **Packet loss rate** and duplicate rate
- **Perceived position error** — difference between server and client grid states
- **CPU utilization and bandwidth per client**

Test Scenarios:

1. **Baseline (no loss):** verify synchronization and latency < 50 ms.
2. **2% loss (LAN-like):** ensure stable gameplay, mean error ≤ 0.5 units.
3. **5% loss (WAN-like):** verify selective reliability prevents major desync.

All results will be logged in CSV format with optional .pcap traces for reproducibility.

6. Future Extensions & Conclusion

Future improvements may include:

- **Adaptive update rates** based on network conditions.
- **Delta encoding** to reduce bandwidth for state updates.
- **Forward Error Correction (FEC)** for improved reliability under higher loss.
- **Visual metrics dashboard** for real-time performance monitoring.

This project provides hands-on experience in **application-layer protocol design, UDP networking, and empirical network testing.**

The **GridClash Binary Protocol** offers an efficient, extensible framework for real-time multiplayer synchronization and serves as a strong foundation for further research and optimization in interactive network systems.