GSync v1 – Grid Clash Protocol (Mini-RFC Draft)

CSE361 – Phase 1 Submission

November 5, 2025

Team Members

#	**Name**	**Student ID**
1	Mohamed Tarek	2300535
2	Peter Maged Shokry	23p0192
3	John George Mikhae	23P0266
4	Rawan Hany Shaker	23P0393
5	hania mahrouse adel soliman	23p0033

1 Introduction

GSync v1 is a lightweight, UDP-based real-time multiplayer protocol used to synchronize a shared grid between one authoritative server and multiple clients. Players attempt to "acquire" ownership of cells in a global `N×N` grid. The server resolves conflicts strictly by first-arrival order and broadcasts periodic snapshots containing the full game state.

Protocol Characteristics

- **UDP** transport (no connection handshake).
- **Binary packet format** with fixed-size header.
- **CRC-32** integrity check (header + payload).
- Clients send `EVENT` messages; server does not send ACKs.
- Server broadcasts `SNAPSHOT` at a fixed rate (default 10 Hz) with 3× redundancy.

2 Protocol Architecture

2.1 Entities

- **Server** Maintains authoritative grid state, logs events, and broadcasts snapshots.
- **Client** Renders UI, sends player actions, and applies server snapshots.

2.2 Transport Layer

- Protocol: **UDP**
- Default server port: `10000`
- No retransmission except: client sends each `EVENT` twice (best-effort)

2.3 Message Flow

1. Client sends `INIT` once at startup.
2. Server adds client to broadcast list.
3. Server sends `SNAPSHOT` packets at rate f Hz.
4. Client sends `EVENT` when clicking a cell.
5. Server accepts or rejects based on current grid state.

```
Client  ---- INIT  ---->  Server
Client  ---- EVENT ---->  Server
Server  == SNAPSHOT ==>>  All Clients (10 Hz, 3 redundant snapshots)
```

3 Message Formats

3.1 Common Header (28 bytes)

Field	Size	Type	Description
proto_id	4B	char[4]	Must equal "GSYN"
version	1B	uint8	Protocol version (1)
msg_type	1B	uint8	0=SNAPSHOT, 1=EVENT, 2=ACK (unused), 3=INIT
snapshot_id	4B	uint32	Snapshot counter (used only in SNAPSHOT)
seq_num	4B	uint32	Per-packet monotonic counter
server_ts	8B	uint64	Timestamp ms (server or client origin)
payload_len	2B	uint16	Number of payload bytes
crc32	4B	uint32	CRC-32 over header(with crc=0) + payload

All integers are big-endian (**network byte order**).

3.2 Payload Formats (by msg_type)

Message Type 3: INIT (Client → Server)

Field	Size	Type	Meaning
player_id	1B	uint8	Unique ID chosen by the client

Message Type 1: EVENT (Client → Server)

Field	Size	Type	Meaning
player_id	1B	uint8	Sender identity
event_type	1B	uint8	0 = ACQUIRE_REQUEST
cell_id	2B	uint16	Index into grid (0 .. $N^2 - 1$)
client_ts	8B	uint64	Client timestamp (ms)

Message Type 0: SNAPSHOT (Server → Clients)

Payload consists of up to 3 full snapshots concatenated (redundancy). Each snapshot has this format:

Field	Size	Type	Meaning
grid_n	1B	uint8	Dimension N of grid ($N \times N$)
owners	N^2 B	uint8[]	Owner of each cell (0=free)

4 Example Message Exchanges

Example 1 – INIT (Client → Server)

Client 2 joins the game.

```
Header:
  proto_id      = "GSYN"
  msg_type      = 3 (INIT)
  payload_len   = 1
```

```
Payload:
  [02]   player_id = 2
```

Server action: Adds client UDP address to broadcast list.

Example 2 – EVENT (Client → Server)

Player 2 attempts to acquire cell 37.

```
Header:
  msg_type    = 1 (EVENT)
  payload_len = 12
Payload:
  [02]        player_id
  [00]        event_type = ACQUIRE_REQUEST
  [00 25]     cell_id = 37
  [00 00 01 97 7C 6B 20]  client_ts (ms)
```

Server behavior:

- checks if cell 37 is still unclaimed
- accepts or rejects based on first arrival
- logs result to `server_events.csv`

Example 3 – SNAPSHOT (Server → All Clients)

Server broadcasts most recent 3 grid states.

```
Header:
  msg_type      = 0 (SNAPSHOT)
  snapshot_id   = 42
  seq_num       = 42
  payload_len   = 301   (3 snapshots of 101 bytes each)

Payload (snapshot 0):
  [0A]                   grid_n = 10
  [100 bytes]            owners[0..99]
(snapshot 1)
(snapshot 2)
```

Client behavior: Reads only the *first* snapshot and updates local grid.