RFC-001: R-Type Game Protocol (RTGP)

Metadata	Details
Version	1.4.3 (Chat system + Admin commands)
Status	Draft / Experimental
Date	2026-01-13
Authors	R-Type Project Team
Abstract	This document specifies the binary application-layer protocol used for real-time communication between the R-Type Client and Server, including a reliability layer over UDP.

1. Introduction

The R-Type Game Protocol (RTGP) is a lightweight, binary, datagram-oriented protocol designed to facilitate real-time multiplayer gameplay. It prioritizes low latency and bandwidth efficiency while providing a selective reliability mechanism (RUDP) to ensure critical game events are delivered.

2. Terminology & Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2.1. Data Types

- **Byte:** 8-bit unsigned integer.

- **uint16:** 16-bit unsigned integer.
- **uint32:** 32-bit unsigned integer.
- **int32:** 32-bit signed integer.
- **float:** 32-bit IEEE 754 floating point.
- **String:** NOT SUPPORTED in standard packets to avoid allocation overhead, unless specified in the payload.

DisconnectReason Enum

The `DisconnectReason` is a uint8 value indicating why a connection was terminated:

Value	Name	Description
0	Timeout	Connection timed out due to lack of response
1	MaxRetriesExceeded	Maximum retry attempts exceeded
2	ProtocolError	Protocol violation or invalid packet
3	RemoteRequest	Disconnect requested by remote peer
4	LocalRequest	Disconnect requested locally

2.2. Byte Order

All multi-byte numeric fields **MUST** be transmitted in **Network Byte Order (Big-Endian)**. Implementations on Little-Endian architectures (x86/x64) MUST convert data before transmission (htons, htonl) and after reception (ntohs, ntohl).

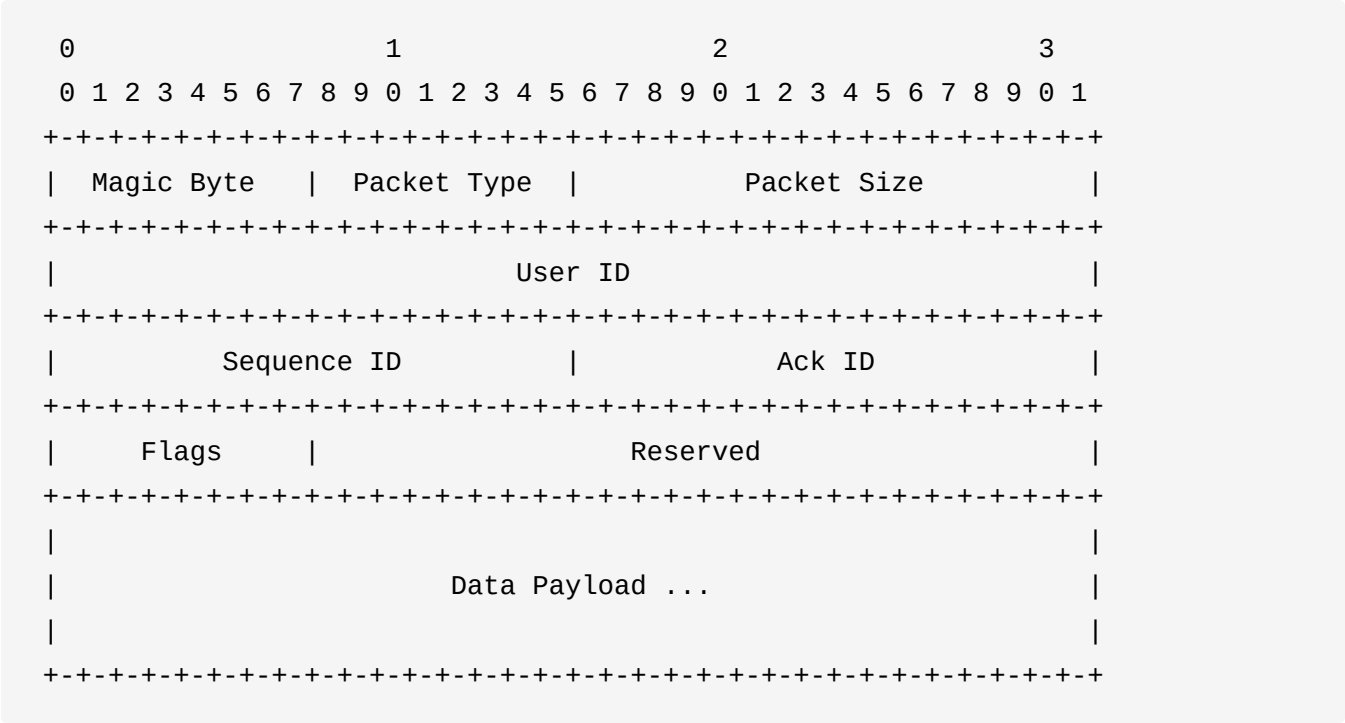
3. Transport Layer

- **Protocol:** UDP (User Datagram Protocol).
- **Default Port:** 4242.
- **MTU Safety:** The total packet size (Header + Payload) **SHOULD NOT** exceed 1400 bytes to avoid IP fragmentation on standard networks.

4. Packet Structure

Every RTGP packet consists of a fixed **16-byte Header** followed by a variable-length **Data Payload**.

4.1. Header Format



Field Definitions:

Field	Type	Description
Magic Byte	uint8	**MUST** be 0xA1. Used to filter spurious traffic.
Packet Type	uint8	The Operation Code (OpCode) defined in Section 5.
Packet Size	uint16	Length of the **Data Payload** in bytes. Excludes Header size.
User ID	uint32	The sender's unique identifier.
Sequence ID	uint16	Incremental ID of the packet sent. Wraps at 65535.
Ack ID	uint16	The Sequence ID of the last packet successfully received.
Flags	uint8	Bitmask for packet attributes (see 4.3).

Field	Type	Description
Reserved	3 bytes	Padding for 16-byte alignment. MUST be 0.

4.2. User ID Conventions

- **Server Authority:** 0xFFFFFFFF (-1). Only the Server uses this.
- **Unassigned Client:** 0x00000000. Used during handshake.
- **Assigned Client:** 0x00000001 to 0xFFFFFFFE.

4.3. Reliability Mechanism (Flags)

The `Flags` field is used to manage the reliability layer (RUDP) and compression.

Flag Bitmask Values:

- **0x01 - RELIABLE:** The sender requests an acknowledgement for this packet. The receiver **MUST** acknowledge this packet (either via a dedicated ACK or piggybacking).
- **0x02 - IS_ACK:** The `Ack ID` field in this header is valid and acknowledges a previously received packet.
- **0x04 - COMPRESSED:** The payload is compressed using LZ4 frame format. The receiver **MUST** decompress the payload before processing.

Behavior:

1. **Sequence ID:** MUST be incremented by 1 for every new packet sent.
2. **Ack ID:** MUST always contain the Sequence ID of the last valid packet received from the remote peer.
3. **Retransmission:** If a packet marked `RELIABLE` is not acknowledged within a specific timeout (e.g., 200ms), the sender MUST retransmit it.

4.4. Compression

RTGP supports optional LZ4 compression for payloads to reduce bandwidth usage.

Compression Behavior:

1. **Threshold:** Payloads smaller than 64 bytes **SHOULD NOT** be compressed (overhead exceeds benefit).
2. **Format:** LZ4 frame format is used, which includes decompressed size metadata for safe buffer allocation.
3. **Header Field:** `Packet Size` (payloadSize) contains the **COMPRESSED** size, not the original size.
4. **Flag:** When payload is compressed, the COMPRESSED flag (0x04) **MUST** be set.
5. **Backward Compatibility:** Receivers **MUST** support both compressed and uncompressed packets. If COMPRESSED flag is not set, payload is processed as-is.

Decompression Requirements:

1. Receivers **MUST** check the COMPRESSED flag before processing payload.
2. If decompression fails, the packet **MUST** be dropped and **MAY** be logged.
3. Decompressed size **MUST NOT** exceed `kMaxPayloadSize` (1384 bytes).

5. Protocol Operations (OpCodes)

5.1. Session Management

0x01 - C_CONNECT

- **Sender:** Client
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Request to establish a connection.
- **Payload:** Empty.

0x02 - S_ACCEPT

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Connection accepted. Assigns the User ID to the client.
- **Payload:**
  - New User ID (uint32)

0x03 - DISCONNECT

- **Sender:** Client OR Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Graceful termination of the session.
- **Payload:**
  - Reason (uint8): DisconnectReason enum value indicating why the connection is being terminated.

0x04 - C_GET_USERS

- **Sender:** Client
- **Reliability: RELIABLE**
- **Description:** Request a list of currently connected players (Lobby).
- **Payload:** Empty.

0x05 - R_GET_USERS

- **Sender:** Server
- **Reliability: RELIABLE**
- **Description:** Server responds to C_GET_USERS.
- **Payload:**
  - Count (uint8): Number of users.
  - UserIDs (uint32[]): Array of User IDs.

0x06 - S_UPDATE_STATE

- **Sender:** Server
- **Reliability: RELIABLE**
- **Description:** Notifies clients of a global game state change.
- **Payload:**
  - StateID (uint8): 0=Lobby, 1=Running, 2=Paused, 3=GameOver.

0x07 - S_GAME_OVER

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Notifies clients that the game has ended with the final score. This is sent when

the game reaches a terminal state.

- **Payload:**
  - Final Score (uint32): The final accumulated score.

0x08 - C_READY

- **Sender:** Client
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Client signals ready/unready state in the lobby. When all players are ready, the server initiates the game start sequence.
- **Payload:**
  - Is Ready (uint8): 1 if ready, 0 if not ready.

0x09 - S_GAME_START

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Server signals that all players are ready and the game is starting. Includes countdown duration for client-side countdown display. Sent when the server determines all required players are ready.
- **Payload:**
  - Countdown Duration (float): Duration in seconds for the countdown timer (e.g., 3.0 for 3 seconds).

0x0A - S_PLAYER_READY_STATE

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Broadcasts the ready/unready state of a specific player to all clients in the lobby. Typically sent by the server in response to a client's **C_READY** message, or when synchronizing lobby state after a client joins.
- **Payload:**
  - Player ID (uint32): Unique identifier of the player whose state changed.
  - Is Ready (uint8): 1 if the player is ready, 0 if not ready.

5.2. Gameplay & Entity Management

0x10 - S_ENTITY_SPAWN

- **Sender:** Server
- **Reliability: RELIABLE** (Critical: Prevents invisible enemies)
- **Description:** Instructs clients to instantiate a new game object.
- **Payload:**
  - Entity ID (uint32)
  - Type (uint8): 0=Player, 1=Bydos, 2=Missile, 3=Pickup, 4=Obstacle.
  - PosX (float)
  - PosY (float)

0x11 - S_ENTITY_MOVE

- **Sender:** Server
- **Reliability: UNRELIABLE** (Flag 0x00)
- **Description:** Regular state update. If lost, the next update fixes it.
  Uses fixed-point quantization to reduce bandwidth and includes a server tick to allow client-side interpolation.
- **Payload:**
  - Entity ID (uint32)
  - Server Tick (uint32) — Monotonic server tick for interpolation
  - PosX (int16) — Quantized/fixed-point world X coordinate
  - PosY (int16) — Quantized/fixed-point world Y coordinate
  - VelX (int16) — Quantized/fixed-point X velocity
  - VelY (int16) — Quantized/fixed-point Y velocity
- **Notes:**
  - Total payload size: **16 bytes** (4 + 4 + 2 + 2 + 2 + 2)
  - Position and velocity components are **quantized signed 16-bit** values (fixed-point). The scale (units per LSB) is implementation-defined and must be consistently shared by client and server.

0x12 - S_ENTITY_DESTROY

- **Sender:** Server
- **Reliability: RELIABLE**

- **Description:** Instructs clients to remove an entity.
- **Payload:**
  - Entity ID (uint32)

0x13 - S_ENTITY_HEALTH

- **Sender:** Server
- **Reliability: RELIABLE**
- **Description:** Synchronizes entity health state with clients. Sent when an entity takes damage or is healed. Critical for displaying health bars and handling death events.
- **Payload:**
  - Entity ID (uint32) - The network ID of the entity
  - Current Health (int32) - The entity's remaining health points
  - Max Health (int32) - The entity's maximum health capacity

0x14 - S_POWERUP_EVENT

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Notifies all clients that a player has collected a power-up. This allows synchronization of power-up effects across all game clients.
- **Payload:**
  - Player ID (uint32) - The User ID of the player who collected the power-up
  - Power-Up Type (uint8) - The type of power-up collected (implementation-specific enumeration)
  - Duration (float) - Duration in seconds for temporary power-ups (0.0 for permanent)

0x15 - S_ENTITY_MOVE_BATCH

- **Sender:** Server
- **Reliability: UNRELIABLE** (Flag 0x00)
- **Description:** Batched position/velocity updates for multiple entities. More bandwidth-efficient than individual S_ENTITY_MOVE packets, especially when combined with LZ4 compression. Each tick, all dirty entity updates are grouped into a single packet.
- **Payload:**
  - Count (uint8): Number of entities in the batch (1-114)
  - Server Tick (uint32): Shared server tick for all entries

    - Entries (EntityMoveBatchEntry[]): Compact per-entity updates, each containing:
        - Entity ID (uint32)
        - PosX (int16) — Quantized/fixed-point world X coordinate
        - PosY (int16) — Quantized/fixed-point world Y coordinate
        - VelX (int16) — Quantized/fixed-point X velocity
        - VelY (int16) — Quantized/fixed-point Y velocity

Notes:

- Maximum entities per batch are limited by payload size: (kMaxPayloadSize - 5) / 12 = 114 (5 = 1 byte count + 4 byte serverTick; 12 = size of each EntityMoveBatchEntry)
- If more than 114 entities need updating, multiple batch packets are sent
- LZ4 compression is automatically applied when batch size exceeds 64 bytes (4+ entities)
- Estimated bandwidth savings: ~25% per-entity compared to non-compacted per-entity payloads (12 bytes vs 16 bytes) and larger savings when compressed

5.3. Input & Reconciliation

0x20 - C_INPUT

- **Sender:** Client
- **Reliability: UNRELIABLE** (Sent frequently)
- **Description:** The client sends its current input state.
- **Payload:**
    - Input Mask (uint8):
        - 0x01=UP,
        - 0x02=DOWN,
        - 0x04=LEFT,
        - 0x08=RIGHT,
        - 0x10=SHOOT.

0x21 - S_UPDATE_POS (Reconciliation)

- **Sender:** Server
- **Reliability: UNRELIABLE**
- **Description:** Correction of client position.
- **Payload:**

    - Authoritative X (float)
    - Authoritative Y (float)

5.4. Chat System

0x30 - C_CHAT

- **Sender:** Client
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Client sends a chat message to the server for broadcast to all players.
- **Payload:**
    - User ID (uint32): The sender's User ID (can be 0 for system messages)
    - Message (char[256]): The chat message content, null-terminated

Notes:

- Maximum message length is 255 characters plus null terminator
- Messages longer than 255 characters **MUST** be truncated by the client before sending
- The server **SHOULD** validate the User ID matches the sender

0x31 - S_CHAT

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Server broadcasts a chat message to all connected clients. This can be a player message or a system notification.
- **Payload:**
    - User ID (uint32): The original sender's User ID (0 for system messages)
    - Message (char[256]): The chat message content, null-terminated

Notes:

- When User ID is 0, the message **SHOULD** be displayed as a system message
- Clients **SHOULD** display the sender's username (if known) alongside the message
- Total payload size: **260 bytes** (4 + 256)

5.5. System & Diagnostics

0xF0 - PING

- **Sender:** Client or Server
- **Reliability: UNRELIABLE**
- **Description:** Latency measurement request. The receiver should respond with a PONG.
- **Payload:** Empty (timestamp can be tracked via seqId).

0xF1 - PONG

- **Sender:** Client or Server
- **Reliability: UNRELIABLE**
- **Description:** Latency measurement response. Echoes the seqId from PING via ackId.
- **Payload:** Empty.

5.6. Admin & Debug Commands

These opcodes are reserved for administrative and debugging purposes. Access is restricted to localhost connections only.

0xD0 - C_ADMIN_COMMAND

- **Sender:** Client
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Client requests an administrative action. The server **MUST** validate that the request originates from localhost (127.0.0.1, ::1, or 127.x.x.x) before executing.
- **Payload:**
  - Command Type (uint8): The admin command to execute (see AdminCommandType enum)
  - Parameter (uint8): Command-specific parameter (0 = off, 1 = on, 2 = toggle)

AdminCommandType Enum:

Value	Name	Description
0x01	GodMode	Toggle player invincibility

Notes:

- Additional command types may be added in future versions
- Non-localhost requests **MUST** be rejected with an error response
- Total payload size: **2 bytes** (1 + 1)

0xD1 - S_ADMIN_RESPONSE

- **Sender:** Server
- **Reliability: RELIABLE** (Flag 0x01)
- **Description:** Server responds to an admin command with success/failure status and a human-readable message.
- **Payload:**
  - Command Type (uint8): Echoed from the request
  - Success (uint8): 0 = failed, 1 = success
  - New State (uint8): The resulting state after the command (0 = off, 1 = on)
  - Message (char[61]): Human-readable response message, null-terminated

Notes:

- The Message field provides feedback to the user (e.g., "God mode enabled", "Access denied: localhost only")
- Total payload size: **64 bytes** (1 + 1 + 1 + 61)

6. Security Considerations

1. **Header Validation:** Any packet where Header[0] != 0xA1 **MUST** be silently dropped.
2. **Sequence Validation:** Packets with a `Sequence ID` significantly older than the last received ID SHOULD be discarded to prevent replay attacks and processing old state.
3. **Spoofing Protection:** The Server **MUST** verify User ID against IP/Port.
4. **Authority Check:** Clients **MUST** ignore packets claiming to be 0xFFFFFFFF (Server) if they do not originate from the known Server IP endpoint.
5. **Admin Command Restriction:** C_ADMIN_COMMAND packets **MUST** only

be processed if the sender's IP address is localhost (127.0.0.1, ::1, or 127.x.x.x). Remote admin requests **MUST** be rejected.

7. Payload Size Reference

OpCode	Payload Size (bytes)	Notes
C_CONNECT	0	Empty
S_ACCEPT	4	uint32
DISCONNECT	1	uint8 (reason)
C_GET_USERS	0	Empty
R_GET_USERS	Variable	1 + (count * 4)
S_UPDATE_STATE	1	uint8
S_GAME_OVER	4	uint32
C_READY	1	uint8
S_GAME_START	4	float
S_PLAYER_READY_STATE	5	uint32 + uint8
S_ENTITY_SPAWN	14	uint32 + uint8 + uint8 + float + float
S_ENTITY_MOVE	16	uint32 + uint32 + 4 * int16 (quantized)
S_ENTITY_MOVE_BATCH	Variable	5 + (count * 12), max 1373 bytes (114 entries)
S_ENTITY_DESTROY	4	uint32
S_ENTITY_HEALTH	12	uint32 + int32 + int32
S_POWERUP_EVENT	9	uint32 + uint8 + float

OpCode	Payload Size (bytes)	Notes
C_INPUT	1	uint8
C_SET_BANDWIDTH_MODE	1	uint8 (0=normal,1=low)
S_BANDWIDTH_MODE_CHANGED	6	uint32 + uint8 + uint8 (userId, mode, activeCount)
S_UPDATE_POS	8	2 * float
C_CHAT	260	uint32 + char[256]
S_CHAT	260	uint32 + char[256]
C_ADMIN_COMMAND	2	uint8 + uint8
S_ADMIN_RESPONSE	64	uint8 + uint8 + uint8 + char[61]
PING	0	Empty
PONG	0	Empty

8. Changes from Previous Versions

Version 1.4.3 (2026-01-13)

- **Added OpCode 0x30 - C_CHAT:** Client sends chat message to server (RELIABLE). Payload: uint32 userId, char[256] message (260 bytes total).
- **Added OpCode 0x31 - S_CHAT:** Server broadcasts chat message to all clients (RELIABLE). Payload: uint32 userId, char[256] message (260 bytes total).
- **Added OpCode 0xD0 - C_ADMIN_COMMAND:** Client requests admin action (RELIABLE, localhost only). Payload: uint8 commandType, uint8 param (2 bytes total).
- **Added OpCode 0xD1 - S_ADMIN_RESPONSE:** Server responds to admin command (RELIABLE). Payload: uint8 commandType, uint8 success, uint8 newState, char[61] message (64 bytes total).
- **AdminCommandType Enum:** GodMode (0x01) for toggling player invincibility.
- **Security:** Admin commands are restricted to localhost connections only.

Version 1.4.2 (2026-01-10)

- **Added OpCode 0x16 - C_SET_BANDWIDTH_MODE:** Client toggles bandwidth mode preference (RELIABLE). Payload: uint8 mode (0=normal,1=low).
- **Added OpCode 0x17 - S_BANDWIDTH_MODE_CHANGED:** Server broadcasts which player changed their bandwidth mode and the current number of players requesting low mode (RELIABLE). Payload: uint32 userId, uint8 mode, uint8 activeCount (6 bytes total).
- **Behavior:** Clients MAY toggle local bandwidth mode via a configurable keybind; the server maintains per-lobby low-bandwidth state and notifies all clients via S_BANDWIDTH_MODE_CHANGED.

Version 1.4.1 (2026-01-05)

- **Added OpCode 0x15 - S_ENTITY_MOVE_BATCH:** Batched entity position/velocity updates for bandwidth optimization
- **Optimization:** Entity updates are now grouped into a single packet per tick, enabling LZ4 compression
- **Estimated bandwidth savings:** 50-60% for entity movement traffic
- **Added OpCode 0x08 - C_READY:** Client lobby ready/unready signal
- **Added OpCode 0x09 - S_GAME_START:** Server-initiated game start with countdown
- **Added OpCode 0x0A - S_PLAYER_READY_STATE:** Broadcast player ready state changes
- **Lobby Workflow:** Clients send C_READY when toggling ready state. Server broadcasts S_GAME_START when all players ready

9. Future Extensions