Pacman Protocol Specification – Version 1

Student Number: 25064762

ENGF0034

1 Introduction

1.1 Terminology

In this specification, the terms "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" are to be interpreted as defined in RFC 2119.

1.2 Protocol Overview

This Pacman Protocol Version 1 enables cooperative Pacman gameplay over a network. The protocol uses TCP on port 5432 for reliable delivery of critical events and UDP on port 5433 for low-latency position updates.

The protocol supports 9 message types:

1. PASSWORD_EXCHANGE
2. SYNC_START
3. MAZE_UPDATE
4. GAME_MODE_UPDATE
5. PACMAN_POSITION
6. GHOST_POSITION
7. PACMAN_EVENT
8. EAT
9. LIVES_SCORE_UPDATE

Each player controls their own Pacman on their local screen. When a Pacman enters a tunnel on one edge of the screen, it transitions to the other player's screen and appears from the opposite tunnel. The protocol synchronises positions, game events, scores, and lives between both players.

1.3 Game Mechanics Relevant to Protocol

1.3.1 Screen Layout

Each player views their own screen (local) and the other player's screen (remote). The coordinates in the protocol messages are the positions of the pixels relative to the origin of the screen.

1.3.2 Game Object States (Referencing Assignment Brief)

- LOCAL: A game object currently on the local screen
- AWAY: Our Pacman is currently on the remote screen
- REMOTE: A game object on the remote screen
- FOREIGN: The other player's Pacman when visiting our screen

1.4 Game Mode Updates

The game mode MUST be communicated via the GAME_MODE_UPDATE message, where the game board will be in one of the following states:

- STARTUP
- CHASE
- FRIGHTEN
- GAME_OVER
- NEXT_LEVEL_WAIT
- READY_TO_RESTART

The gameplay itself occurs only in the CHASE and FRIGHTEN game modes, where FRIGHTEN indicates that a player's Pacman has recently swallowed a power pill.

1.4.1 Ghost Management

- Four ghosts exist on each player's local screen.
- Ghosts are unable to traverse tunnels between screens.
- Ghost positions are synchronised with the remote player via GHOST_POSITION messages to enable detection of collisions with foreign Pacman.

1.4.2 Food Synchronisation

- Dots and power pills exist on both screens.
- When food is eaten on either screen, an EAT message removes it from both screens.

1.5 Transport Protocol Rationale

This protocol uses TCP (port 5432) for reliable delivery of critical events and UDP (port 5433) for low-latency position updates.

TCP is used for:

- PASSWORD_EXCHANGE: Connection establishment
- SYNC_START: Game synchronisation
- MAZE_UPDATE: Large data transfer, MUST NOT be lost
- GAME_MODE_UPDATE: State changes
- PACMAN_EVENT: Critical Pacman events
- EAT: Critical eating events
- LIVES_SCORE_UPDATE: Score and lives synchronisation

UDP is used for:

- PACMAN_POSITION: High-frequency position updates (20 fps)
- GHOST_POSITION: High-frequency position updates (20 fps $\times$ 4 ghosts)

1.5.1 Rationale

Position updates are sent frequently and benefit from UDP's lower latency. If one position update is lost, the next update replaces it anyway. Critical events like eating food or changing the game mode MUST arrive reliably, so TCP is used. This hybrid approach provides speed and reliability where each is needed.

2 Connection Establishment

2.1 Prerequisites

Before connection, both players MUST agree upon:

- Server's IP address
- Shared password (maximum 15 ASCII characters)

2.2 Connection Sequence

1. Client opens TCP connection to server on port 5432
2. Client sends PASSWORD_EXCHANGE message
3. Server sends PASSWORD_EXCHANGE message
4. Both verify passwords match (if there is a mismatch, see section 5 [Error Handling])
5. Both bind UDP sockets to port 5433
6. Both players send MAZE_UPDATE messages
    - Server sends its maze to client
    - Client sends its maze to server
    - Order does not matter
    - Both MUST receive the other's maze before proceeding
7. Server sends SYNC_START message
8. Game begins (UDP position updates + TCP events)

3 General Message

Messages are fixed-format and binary-encoded, with all integer fields sent in network byte order (big-endian). No external file formats, such as pickle, HTML, XML, JSON, etc., are used. As the message type is fixed format, no explicit length field is required.

3.1 Message Structures

TCP Message Structure:

- Type field (4 bits) identifies the message
- The remaining bits contain payload data

UDP Message Structure:

- Sequence number (16 bits, big-endian)
- Type field (4 bits) identifies the message
- The remaining bits contain payload data

The sender MUST increment the sequence number for every UDP message sent, wrapping to 0 after reaching 65535. The receiver SHOULD discard messages with sequence numbers less than or equal to the most recently processed sequence number as it would mean it is an out-of-order message.

3.2 Message Type Assignments

Each message begins with a 4-bit type field that identifies the message type. The following 4-bit type values are:

Type	Message	Transport
0x1	PASSWORD_EXCHANGE	TCP
0x2	SYNC_START	TCP
0x3	MAZE_UPDATE	TCP
0x4	GAME_MODE_UPDATE	TCP
0x5	PACMAN_POSITION	UDP
0x6	GHOST_POSITION	UDP
0x7	PACMAN_EVENT	TCP
0x8	EAT	TCP
0x9	LIVES_SCORE_UPDATE	TCP

Receivers MUST ignore messages with unassigned type values.

4 Message Content, Timing, and Encoding

This section describes what each message type contains, when it is sent, which transport protocol it uses, and how it is encoded in binary format.

4.1 PASSWORD_EXCHANGE

4.1.1 Contents

- Type: PASSWORD_EXCHANGE
- Password: Password entered by the player. Passwords are limited to printable ASCII characters with a maximum length of 15 characters.

4.1.2 Timing

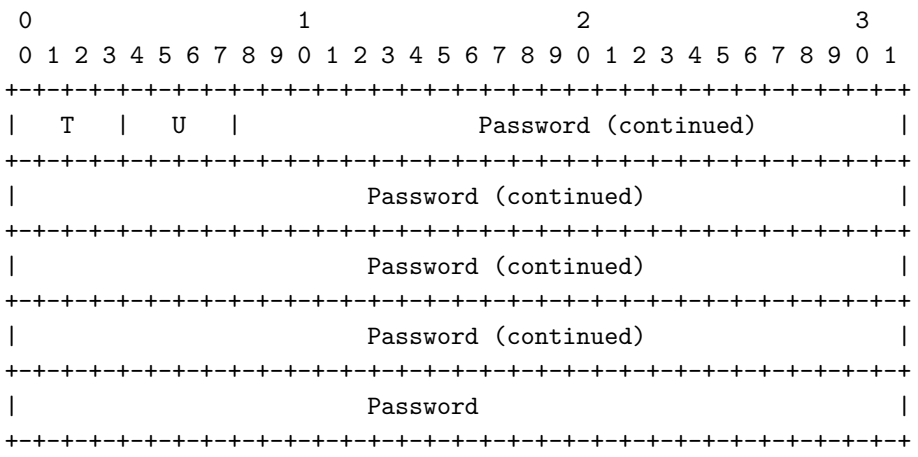
PASSWORD_EXCHANGE messages MUST be sent once when a client connects to the server.

4.1.3 Transport

TCP (port 5432)

4.1.4 Encoding

PASSWORD_EXCHANGE messages consist of 17 bytes:



Fields:

- T: 4-bit type field = 0x1
- U: 4 bits unused (must be zero)
- Password: 128 bits (16 bytes)
  - Maximum 15 ASCII characters – 15 bytes
  - Null-terminator to indicate end of password (0x00) – 1 byte

4.2 SYNC_START

4.2.1 Contents

- Type: SYNC_START
- Start Time: The server sends a Unix timestamp (current time + 1 second) to synchronise the start of the game. Both clients wait until this time before starting gameplay.
  - **Note:** The 1-second delay accounts for network transmission and processing time.

4.2.2 Timing

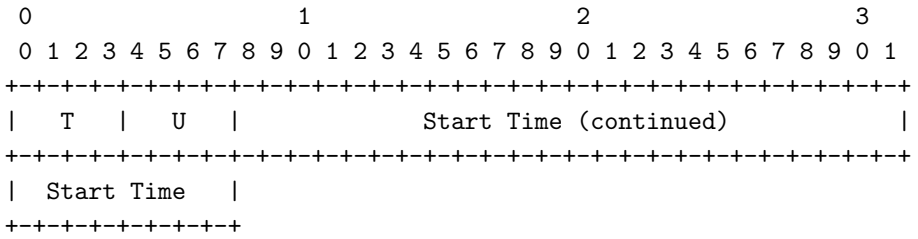
SYNC_START MUST be sent after maze exchange, before the game starts.

4.2.3 Transport

TCP (port 5432)

4.2.4 Encoding

SYNC_START messages consist of 5 bytes:



Fields:

- T: 4-bit type field = 0x2
- U: 4 bits unused (must be zero)
- Start Time: 32-bit unsigned integer (Unix timestamp)
  - Current time + 1 second
  - Big-endian byte order

4.3 MAZE_UPDATE

4.3.1 Contents

- Type: MAZE_UPDATE
- Maze: The maze layout consists of 84 columns × 31 rows = 868 squares. Each square is represented by 4 bits, encoding one of 12 possible square types

Square encoding (4 bits per square):

- 0x0: “ /-” (top-left corner)
- 0x1: “-/ ” (top-right corner)
- 0x2: “—” (horizontal wall)
- 0x3: “-\\” (bottom-right corner)
- 0x4: “ \\-” (bottom-left corner)

- 0x5: “ — ” (vertical wall)
- 0x6: “###” (ghost house wall)
- 0x7: “ ” (empty space)
- 0x8: “ . ” (food pellet)
- 0x9: “ * ” (power pill)
- 0xA: “ A ” (left tunnel entrance)
- 0xB: “ B ” (right tunnel entrance)

4.3.2 Timing

MAZE_UPDATE MUST be sent once by **both players** during connection establishment:

- Each player sends their own maze to the other player
- Both players MUST wait to receive the remote maze before proceeding

It MAY be sent again when the game restarts or level changes.

4.3.3 Transport

TCP (port 5432)

4.3.4 Internal Format

[“maze”, maze_object]

4.3.5 Encoding

MAZE_UPDATE messages consist of 435 bytes:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   T   |   U   | Sqr 1 | Sqr 2 | Sqr 3 | Sqr 4 | Sqr 5 | Sqr 6 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Sqr 866|Sqr 867|Sqr 868|
+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

- T: 4-bit type field = 0x3
- U: 4 bits unused (must be zero)
- Squares: 868 squares × 4 bits each = 434 bytes
  - 4 bits for each square → 2 squares per byte

4.4 GAME_MODE_UPDATE

4.4.1 Contents

- Type: GAME_MODE_UPDATE
- Game Mode: The local game board has six states where a 3-bit value represents each:
  - 0: STARTUP
  - 1: CHASE
  - 2: FRIGHTEN
  - 3: GAME_OVER
  - 4: NEXT_LEVEL_WAIT
  - 5: READY_TO_RESTART

4.4.2 Timing

GAME_MODE_UPDATE messages MUST be sent when the game mode changes:

- STARTUP → CHASE
- CHASE → FRIGHTEN (power pill eaten)
- FRIGHTEN → CHASE (frighten mode expires)
- Any mode → GAME_OVER (all lives are lost)
- GAME_OVER → READY_TO_RESTART
- READY_TO_RESTART → STARTUP

4.4.3 Transport

TCP (port 5432)

4.4.4 Internal Format

[“status”, [status]]

4.4.5 Encoding

GAME_MODE_UPDATE messages consist of 1 byte:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|   T   |U|  GM |
+-+-+-+-+-+-+-+-+
```

Fields:

- T: 4-bit type field = 0x4
- U: 1 bit unused (must be zero)
- GM: 3-bit game mode

4.5 PACMAN_POSITION

4.5.1 Contents

- Type: PACMAN_POSITION
- Position X, Y: Pixel coordinates (0–1023)
  - X: 0 = left edge, 1023 = right edge
  - Y: 0 = top edge, 1023 = bottom edge
- Direction: 2-bit value
  - 0: UP
  - 1: LEFT
  - 2: RIGHT
  - 3: DOWN
- Speed: 1-bit value
  - 0: Stationary
  - 1: Moving

4.5.2 Timing

PACMAN_POSITION messages SHOULD be sent every 50ms (20fps).

4.5.3 Transport

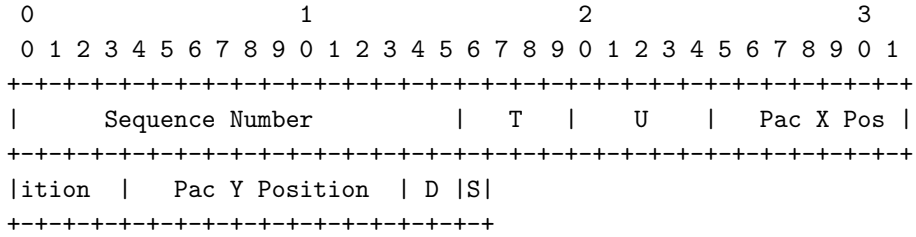
UDP (port 5433)

4.5.4 Internal Format

["pacman", [position, direction, speed]]

4.5.5 Encoding

PACMAN_POSITION messages consist of 6 bytes total:



Fields:

- Sequence: 16-bit unsigned integer
- T: 4-bit type field = 0x5
- U: 5 bits unused (must be zero)
- Pac X Position: 10 bits (0–1023)
- Pac Y Position: 10 bits (0–1023)
- D: 2-bit direction
- S: 1-bit speed

4.6 GHOST_POSITION

4.6.1 Contents

- Type: GHOST_POSITION
- Ghost Number: 2-bit identifier
  - 0: Red (Blinky)
  - 1: Pink (Pinky)
  - 2: Cyan (Inky)
  - 3: Orange (Clyde)
- Position X, Y: Pixel coordinates (0–1023 for both X and Y)
  - X: 0 = left edge, 1023 = right edge

  - Y: 0 = top edge, 1023 = bottom edge

- Direction: 2-bit value
  - 0: UP
  - 1: LEFT
  - 2: RIGHT
  - 3: DOWN

- Mode: 3-bit ghost mode
  - 0: SCATTER
  - 1: CHASE
  - 2: FRIGHTEN
  - 3: FRIGHTEN_TRAPPED
  - 4: EYES

- Speed: 1-bit value
  - 0: STATIONARY
  - 1: MOVING

4.6.2 Timing

GHOST_POSITION messages SHOULD be sent every 50ms (20 fps) for each ghost. Each message specifies information about only one ghost, and four messages are sent together, one for each ghost.

4.6.3 Transport

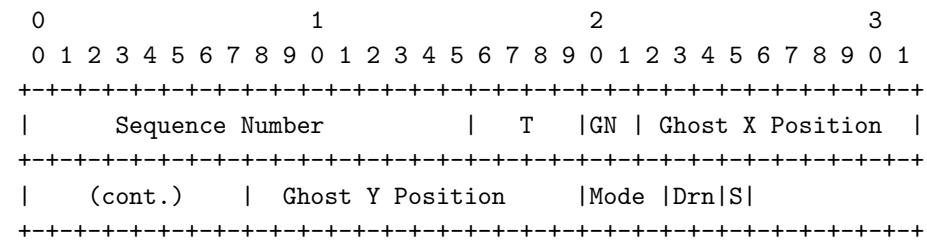
UDP (port 5433)

4.6.4 Internal Format

[“ghost”, [ghostnum, position, direction, speed, mode]]

4.6.5 Encoding

GHOST_POSITION messages consist of 6 bytes total:



Fields:

- Sequence: 16-bit unsigned integer
- T: 4-bit type field = 0x6
- GN: 2-bit ghost number
- Ghost X Position: 10 bits (0–1023)
- Ghost Y Position: 10 bits (0–1023)
- Mode: 3 bits
- Drn: 2-bit direction
- S: 1-bit speed

4.7 PACMAN_EVENT

4.7.1 Contents

- Type: PACMAN_EVENT
- Foreign Status: 2-bit value indicating foreign Pacman presence
  - 0: No foreign Pacman / foreign Pacman left
  - 1: Foreign Pacman arrived from the left tunnel
  - 2: Foreign Pacman arrived from the right tunnel
  - **Note**: "Left" and "right" refer to the tunnel positions from the perspective of the local player's screen.
- Died: 1-bit flag (1=died, 0=alive)
- Go Home: 1-bit flag indicating forced return (1= go home, 0= normal operation)

4.7.2 Timing

PACMAN_EVENT messages MUST be sent immediately when Pacman's status changes:

- Foreign Pacman arrived (with tunnel side)
- Foreign Pacman left
- Pacman died
- Pacman go home

4.7.3 Transport

TCP (port 5432)

4.7.4 Internal Format

```
["newpacman", [side]]  → Foreign Status = 1 (if side="left") or 2 (if side="right")
["pacmanleft", []] → Foreign Status = 0
["pacmandied", []] → Died = 1
["pacmanhome", []] → Go Home = 1
```

4.7.5 Encoding

PACMAN_EVENT messages consist of 2 bytes:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   T   |       U       | F |D|H|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

- T: 4-bit type field = 0x7
- U: 8 bits unused (must be zero)
- F: 2-bit foreign status
- D: 1-bit died flag
- H: 1-bit go home flag

4.8 EAT

4.8.1 Contents

- Type: EAT
- Food/Power Pill (FP): 2-bit value
  - 0: Nothing eaten (used when only a ghost is eaten)
  - 1: Food pellet eaten
  - 2: Power pill eaten
- Food Position X, Y: Pixel coordinates (0–1023 for both X and Y)
  - Set to (0, 0) when FP=0
- Ghost Eaten by Local Player (GE): 4-bit value
  - Ghost eaten by the local player's Pacman (LOCAL or AWAY)
  - 0: No ghost eaten
  - 1: Blinky eaten
  - 2: Pinky eaten
  - 3: Inky eaten
  - 4: Clyde eaten
- Ghost Eaten by Foreign Pacman (FPGE): 4-bit value
  - Ghost eaten by the foreign Pacman visiting local screen
  - 0: No ghost eaten
  - 1: Blinky eaten
  - 2: Pinky eaten
  - 3: Inky eaten
  - 4: Clyde eaten

4.8.2 Timing

EAT messages MUST be sent immediately when any of the following events are detected on the local screen:

- Food pellet or power pill eaten by any Pacman (with position)
- Ghost eaten by local player's Pacman (whether LOCAL or AWAY)
- Ghost eaten by FOREIGN Pacman visiting local screen

Note: Each computer detects and reports all eating events occurring on its own screen.

4.8.3 Transport

TCP (port 5432)

4.8.4 Internal Format

The EAT message consolidates multiple internal game events:

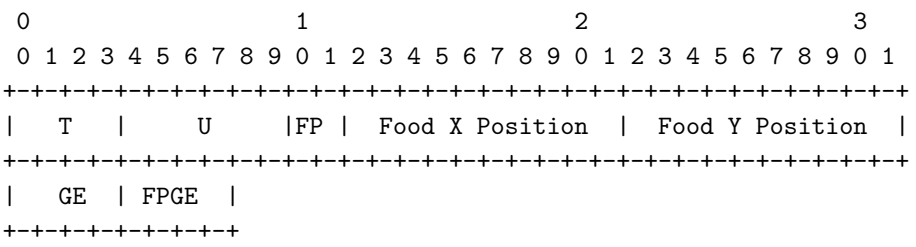
```
["eat", [position, is_powerpill]]
  → Food/power pill eaten by any Pacman on local screen
  → Maps to FP, X, Y fields
  → is_powerpill: False → FP=1, True → FP=2
  → GE=0, FPGE=0

["ghosteaten", [ghostnum]]
  → Ghost eaten by local player's Pacman (LOCAL or AWAY)
  → Maps to GE field (ghostnum 0-3 encoded as 1-4)
  → FP=0, X=0, Y=0, FPGE=0

["foreignghosteaten", [ghostnum]]
  → Ghost eaten by FOREIGN Pacman visiting local screen
  → Maps to FPGE field (ghostnum 0-3 encoded as 1-4)
  → FP=0, X=0, Y=0, GE=0
```

4.8.5 Encoding

EAT messages consist of 5 bytes:



Fields:

- T: 4-bit type field = 0x8
- U: 6 bits unused (must be zero)
- FP: 2-bit food/power pill indicator
- Food X Position: 10 bits
- Food Y Position: 10 bits
- GE: 4-bit ghost eaten by local player's Pacman
- FPGE: 4-bit ghost eaten by foreign Pacman

4.9 LIVES_SCORE_UPDATE

4.9.1 Contents

- Type: LIVES_SCORE_UPDATE
- Lives: 3-bit integer (0-5 lives) indicating remaining lives
- Score: 22-bit integer (0 to 4,194,303)
  - Maximum Pacman score is 3,333,360, thus 22 bits are required
  - $(2^{21}) - 1 < 3,333,360 < (2^{22}) - 1$

4.9.2 Timing

LIVES_SCORE_UPDATE messages MUST be sent when:

- Lives count changes (Pacman dies)
- Score changes (food eaten, ghost eaten, etc.)

4.9.3 Transport

TCP (port 5432)

4.9.4 Internal Format

Combines two internal messages:

```
["lives", [lives]]
["score", [score]]
```

4.9.5 Encoding

LIVES_SCORE_UPDATE messages consist of 4 bytes:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   T   |  U  |Lives|               Score                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

- T: 4-bit type field = 0x9
- U: 3 bits unused (must be zero)
- Lives: 3 bits (0–5)
- Score: 22 bits (0 to 4,194,303)

5 Error Handling

5.1 Password Mismatch

If passwords do not match after a PASSWORD_EXCHANGE message occurs:

1. The server MUST close the TCP connection immediately
2. The client SHOULD display an error message to the user
3. No further messages SHOULD be sent

5.2 Errors in Transport

5.2.1 TCP

If the TCP connection fails:

1. The protocol SHOULD attempt reconnection
2. The game SHOULD display a "Connection Lost" message
3. Once reconnected, continue game

5.2.2 UDP

- Lost packets are replaced by subsequent updates
- Out-of-order packets are discarded based on sequence numbers