Pac-man Protocol Specification

# student number: 23158200

Terminology

This specification uses the terms **MUST, SHOULD and MAY** as defined in RFC 2119 [rfc2119]

The pac-man protocol runs over UDP, using a well known port 5432. The messages type including: `INIT`, `POSITION_UPDATE`, `PLAYER_STATUS`, `GHOST_STATUS`, `BOARD_UPDATE` and `CHECK`.

The board where each player's pac-man spawns is refered to as LOCAL MAP, the other one is called REMOTE MAP.

When to send the message

At the moment when the connection is established or a new level starts, each player sends the `INIT` message which contains the maze shape detail and the level information to the other side. When the player receives the message, it creates the remote map maze locally.

During the gameplay, no matter which board the pac-man is based on (locally or remotely), the pac-man is always controlled by the computer where it is generated. (e.g Player 1's pac-man entered the Player 2's maze, but all the status/position updates of this pac-man are still simulated and generated from Player 1's PC)

On top of that, both computers continuously send the `POSITION_UPDATE` message in each frametime (typically t = 1/60 s). When the message is received on the other side, the pac-man and the ghosts' positions can be correctly updated.

When the status of the pac-man (entered, left, eat, died, lives left) or ghost (been eaten) changed, the player sends a `PLAYER_STATUS` or `GHOST_STATUS` message accordingly to the other side to inform the change of player or ghost status. The other player **SHOULD** receive the messge to operate on the their display (local map or remote map) and store the changed information locally until receive the next `PLAYER_STATUS` or `GHOST_STATUS` message.

When the food on the local map is eaten or the score is updated, the `BOARD_UPDATE` message is sent to the other side to update the scores and the food on the window.

(e.g when Player1's Pac-Man eats a food on the remote map, Player1's PC **MUST** send a message to Player2 with the coordinate of the food just been eaten *(by Player1's Pac-Man)*. Player2 receives the message, and then he **SHOULD** check the lastest `PLAYER_STATUS` message he had received and restored in the memory to find out which map the Player1's pac-man is currently on)

As UDP may lose packets, a `CHECK` is send to confirm the recipt of `INIT`, `PLAYER_STATUS`, `GHOST_STATUS` and `BOARD_UPDATE` messages. it **MUST** hold the same *type number and sequence number* as the received message, and be sent immediately after receiving. The messages that require a `CHECK` confirmation **SHOULD** repeatedly sending the information if no `CHECK` is received within 20ms.

The reason why we don't reply `CHECK` message to `POSITION_UPDATE` is that even though with the potential package loss, `POSITION_UPDATE` message is sent in a high frequency so that even we **MAY** lose a few packages in between, the receiver can still draw the objects correctly in a majority of time, which the player can hardly notice.

Content of messages

`INIT` type

- Type: `INIT`
- Value: shape of the maze and level information. The maze info consists of 2634 ASCII encoded characters. Therefore, the size of the maze **SHOULD** be 2634 bytes. However, the maximum size per package supported by UDP is typically 1500 bytes. Hence, two messages are required to send, each containning half of the maze info (1317 bytes). To make sure the receiver understand which half of the maze is received, an sequence number is also contained in the message. The level ranges from 0 to 15 inclusive, when player enters the next level, the number increment by 1.

`POSITION_UPDATE` type

- Type: `POSITION_UPDATE`
- Value: the positions of the player and the ghosts. This would cover the information including `object`, `positonX`, `postionY` and `Direction`. This covers the postion for all objects, the pac-man and the ghosts. The `postionX` and `positionY` are integers ranging from 0 - 650, 0 - 800 respectively, where postion (0,0) is the top left of the canvas and (649,799) is the bottom right corner of the canvas. The direction is represented by an integer between 0 - 3 inclusive. 0,1,2,3 represents the direction **'up', 'right', 'down', 'left'** accordingly.

`PlAYER_STATUS` type

- type: `PLAYER_STATUS`
- Value: `PLAYER_STATUS` message contains the current status information of the pac-man, including
  - is pac-man on remote map or local map
  - is the pac-man alive
  - do pac-man has power pills
  - how many lives left
- The first three status (pac-man on remote map, pac-man alive and powerpill) can be represented by a one bit bool value. The lives left can be expressed using a 3 bits integer, since the maximum lives of the pac-man is 5.

`GHOST_STATUS` type

- type: `GHOST_STATUS`
- Value: The `GHOST_STATUS` represents whether the ghosts are eaten at the moment. This is only use a 1 bit bool.

BOARD_UPDATE type

- type: BOARD_UPDATE
- Value: The BOARD_UPDATE message contains the exact position of the one food been eaten at the moment as well as the updated score. The position including x and y coordinates, ranging from (0,0) *[- top left]* to (649, 799) *[- bottom right]*. Here the score is set to be a integer taken no more than 32 bits, ranges from 0 ~ (2^32-1) inclusive.

CHECK type

- type: CHECK
- Value: The CHECK message is sent once receive a message containing type value 0, 2, 3, 4. The message reply its own type number and the same type and sequence numbers contained in the received message. (type: 0-4, sequence number: 0 - (2^29-1))

Message encoding

Sequence numbers

Due to the potential package loss using UDP, we use sequence numbers to ensure that the message we received is in the order. Therefore, when a message is received, the receiver **MUST** compare its sequence number to the lastest same type of message that has been previously received. If the number is out of the order or sent before the lastest number, this message **SHOULD** be ignored.

Addtionally, when the sequence number reaches the end, it **MUST** be reset to zero, causing the next sequence number to be smaller than the previous one. Hence, the receiver **should** pay extra care when comparing the sequence numbers.

INIT message format

INIT message consist of **1322 bytes** for *each message*, encoded as follows:

```
 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  |                   sequence number                       | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Maze                              | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Maze(cont)                          | (...)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Maze(cont)                          | 330
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Maze(cont)  | LEVEL |   U   |                                 331
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = INIT has decimal value 0

- S: squence number, 29 bits unsigned integer. Represents which part of the maze information it contain.
  - even value: the first half
  - odd value: the second half
- Maze: maze information, containning 1317 ASCII encoded characters. Since each ASCII encoded character takes 8 bits of space, hence, the size taken by the maze info is 1317 bytes where each byte represents a character.
- LEVEL: The current level of this round of game. 4 bits unsigned integer giving in big-endian order.
- U: unused, 4 bits, not used in order to maintain byte alignment. **MUST** be set to zero.

`POSITION_UPDATE` message format

`POSITION_UPDATE` message consist of **18 bytes**, encoded as follows:

```
 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  |                  sequence number                        | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P-D|    P positionX    |    P positionY    |    G1 positionX   | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G1D|    G1 positionY   |    G2 positionX   |    G2 positionY   | 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G2D|    G3 positionX   |    G3 positionY   |    G4 positionX   | 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G3D|    G4 positionY   |G4D| U |                                 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = `POSITION_UPDATE` has decimal value 1
- sequence number: a 29-bit unsigned integer, incremented by one for every new message sent. If the number reaches (2^29 - 1), the number **MUST** be reset back to 0
- P positionX, P positionY | G1 positionX, G1 positionY | G2 positionX, G2 positionY| G3 positionX, G3 positionY | G4 positionX, G4 positionY:
  - represent Pac-man, Ghost1, Ghost2, Ghost3 and Ghost4's positions.
  - 10 bits unsigned integer for each position, giving an unsigned integer in big-endian byte order.
- P-D, G1D, G2D, G3D, G4D:
  - Pac-man, Ghost1, Ghost2, Ghost3 and Ghost4 direction
  - 2 bits for each, representing 'up', 'right', 'down' and 'right' 4 directions, giving an unsigned integer in big-endian byte order.
- U: unused, 2 bits, not used in order to maintain byte alignment. **MUST** be set to zero.

PLAYER_STATUS message format

PLAYER_STATUS message consist of **5 bytes**, encoded as follows:

```
 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  |                   sequence number                       | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| U |M|A|P|LIVES|                                                 2
+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = PLAYER_STATUS has decimal value 2
- sequence number: a 29-bit unsigned integer, incremented by one for every new message sent. If the number reaches (2^29 - 1), the number **MUST** be reset back to 0
- U: unused, 2 bits, not used in order to maintain byte alignment. **MUST** be set to zero
- M: is the pac-man on remote map. 1 bit bool. 0 for on remote map, 1 for on local map
- A: is pac-man alive or died. 1 bit bool. 0 for died, 1 for alive.
- P: has power pill. 1 bit bool. 0 for not, 1 for has.
- LIVES: how many lives left. 3 bits, giving an unsigned integer in big-endian order.

GHOST_STATUS message format

GHOST_STATUS message consist of **5 bytes**, encoded as follows:

```
 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  |                   sequence number                       | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   U   |W|X|Y|Z|                                                 2
+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = GHOST_STATUS has decimal value 3
- sequence number: a 29-bit unsigned integer, incremented by one for every new message sent. If the number reaches (2^29 - 1), the number **MUST** be reset back to 0
- U: unused, 4 bits, not used in order to maintain byte alignment. **MUST** be set to zero.
- W, X, Y, Z:
  - representing Ghost1, Ghost2, Ghost3 and Ghost4's life status respectively
  - 1 bit unsigned bool for each

`BOARD_UPDATE` message format

`BOARD_UPDATE` message consist of **11 bytes**, encoded as follows:

```
  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  |                   sequence number                       | 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   U   |   Food PositionX  |   Food PositionY  |    scores       | 2
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  scores (cont)                |                 3
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = `POSITION` has decimal value 4
- sequence number: a 29-bit unsigned integer, incremented by one for every new message sent. If the number reaches (2^29 - 1), the number **MUST** be reset back to 0
- U: unused, 4 bits, not used in order to maintain byte alignment. **MUST** be set to zero.
- Food PositionX, Food PositionY: The coordinate of the food been eaten. 10 bits unsigned integer in big-endian order.
- scores: the current score. 32 bits unsigned integer in big-endian order.

`CHECK` message format

`CHECK` message consist of **8 bytes**, encoded as follows:

```
  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  |                   sequence number                       | 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | R T |                  R sequence number                      | 2
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- T: 3 bit type field. Type = `CHECK` has decimal value 5
- sequence number: a 29-bit unsigned integer, incremented by one for every new message sent. If the number reaches (2^29 - 1), the number **MUST** be reset back to 0
- RT: 3 bit type field. containing the type number of the received message
- R sequence number: The sequence number from the received message