Engf0002 CA5.0 Pacman Sepcification

Student number: **19072161**

Terminology

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

The Pacman game has four status, we use following terminology to distinguish between pacman and variuos game objects:

```
  - LOCAL: the game object is a local game object, and is currently on the
local screen.

  - AWAY: our pacman is current away on the remote screen.

  - REMOTE: a game object on the remote screen that our AWAY pacman might
interact with.

  - FOREIGN: the other player's pacman, when it is visiting our screen.
```

Self defined terms for abbreviation:

```
  - L_MODEL: the local model of pacman game, on local machine.

  - F_MODEL: the other player's L_MODEL.
```

Since the intreactions require low response time, the game model MUST be drawn locally.

The Pacman protocol runs over **UDP**, using port number **5432**. There are ( ) message types and can be split into two larger groups by reliability required:

NEED TO BE RELIABLE:

- **MAZE_UPDATE**

  MUST to be send when initial connection of two players is estabilished and the start of next level. This message is used to tell F_MODEL our LOCAL maze.

- **SCORE_UPDATE**

  When there is a changement in score of our pacman in L_MODEL. We MUST send this message to notify F_MODEL to update the score.

- **LIVES_UPDATE**

  When there is a changement in lives of our pacman in L_MODEL. We MUST send this message to notify F_MODEL to update lives.

- **STATUS_UPDATE**

  When there is a changement in status of game in LOCAL. we MSUT send this message to notify F_MODEL to update status for REMOTE.

- **PACMAN_ARRIVED**

  When LOCAL pacman become AWAY, we MUST send a PACMAN_ARRIVED message to F_MODEL. When F_MODEL recieve this message, it can create a FORIGN pacman.

- **PACMAN_LEFT**

  When FOREIGN pacman become REMOTE, we MUST send a PACMAN_LEFT message to F_MODEL. When F_MODEL recieve this message, it can create a LOCAL pacman.

- **PACMAN_DIED**

  When LOCAL or FOREIGN pacman died, we MUST send this message to tell F_MODEL to reset died pacman.

- **PACMAN_GO_HOME**

  When there's an update on level. We MUST send this message to tell F_MODEL to reset both pacman's postion

- **GHOST_EATEN**

  When a ghost is eaten, no matter on LOCAL maze or REMOTE maze. We MUST send this message to tell F_MODEL that a ghost was eaten, to make sure F_MODEL also perform the same action as L_MODEL (update its display).

- **EAT**

  When a powerpill or food is eaten, no matter on LOCAL maze or REMOTE maze. We MUST send this message to tell F_MODEL that a powerpill is eaten, to make sure F_MODEL also perform the same action as L_MODEL (update its display).

NO NEED TO BE RELIABLE:

- **PACMAN_UPDATE**

  When there is an change on **position** or **direction** or **speed** of either our pacman or the other player's pacman. We MUST send this message to tell F_MODEL so pacmans can be synchronized on both L_MODEL and F_MODEL.

- **GHOST_UPDATE**

  When there is an change on **position** or **direction** or **speed** or **mode** of either LOCAL ghost or REMOTE ghost. We MUST send this message to tell F_MODEL so ghosts can be synchronized on both L_MODEL and F_MODEL.

[UDP] Dealing with data need to be reliable VS. no need to be reliable

The strategy for specify data into this two categories is because some messages for some events are only send once. Due to the property of UDP, our packets might lost. Hence, in order to make sure the game looks the same on both computers, These messages must be delivered. So, for these data, we need to ensure the lost packets is resend.

Also, a sequence number is used to determine the order of packets and whether the message needs to be reliable or not.

For message don't need to be reliable, we simply take the latest packet (MAY use higher sequence number to represent later packet)

For message requires reliability, we firstly make a copy of packets and store it in a sender's buffer. Then wait for an amout of time (MAY use 1.5 times the time used for one trip between connection) after we send packet. If we don't recieve any acknowledgement (MAY contain sequence number of packet) back from target side, We resend the packet. If we got the acknowledgement, We delete that packet from local buffer.

Message Contents

--CAUTION--

- for **Type** in a message, Should be an integer (more clarify in encoding part)
- Assuming Coordinate has fix range 0-1023 and top left is 0,0.
- Assuming maze has fixed number of blocks 0-1023

The contents of a MAZE_UPDATE message

Note: Assuming maze has fixed number of blocks

- Type: **MAZE_UPDATE**
- Value: **Status**, **Symbol type**, **position x, y coordinate**. **Status** SHOULD be value between 0 and 2. **Symbol type** SHOULD be an integer in range 0 and 9. **x** and **y** SHOULD be positive integer between 0 and 1023.

Status

```
status = 0 # Start of maze
status = 1 # middle of maze
status = 2 # end of maze
```

Symbol type

```
SYMBOL TABLE
  "/" = 0
  "-" = 1
  "\" = 2
  "|" = 3
  "#" = 4
  " " = 5
  "." = 6
  "*" = 7
  "A" = 8
  "B" = 9
```

The contents of a SCORE_UPDATE message

- Type: **SCORE_UPDATE**
- Value: **scores** of player1 and player2. **scores** should be an **posirive integer**.

The contents of a LIVES_UPDATE message

- Type: **LIVES_UPDATE**
- Value: **num_lives** of player1 and player2. **Number of lives** should be an **positive integer**.

The contents of a STATUS_UPDATE message

- Type: **STATUS_UPDATE**
- Value: **current_status**. Value of **current state** SHOULD be **integer between 0 and 6**.

```
    LOCAL = 0   # local object, currently local
    AWAY = 1    # local object, currently on vacation
    FOREIGN = 2 # foreign object visiting us, needs displaying
    REMOTE = 3  # foreign object on their screen, we don't display
    LOCAL_DYING = 4   # dying locally
    AWAY_DYING = 5   # dying while on vacation
    REMOTE_DYING = 6   # foreign object dying there
```

The contents of a PACMAN_ARRIVED message

- Type: **PACMAN_ARRIVED**

The contents of a PACMAN_LEFT message

- Type: **PACMAN_LEFT**

The contents of a PACMAN_DIED message

- Type: **PACMAN_DIED**

The contents of a PACMAN_GO_HOME message

- Type: **PACMAN_GO_HOME**

The contents of a GHOST_EATEN message

- Type: **GHOST_EATEN**
- Value: **ghost number**. A integer, SHOULD only be 0 to 4 (defined in pacman game)

The contents of a EAT message

- Type: **EAT**
- Value: **postion x, y** and **is_foregin**, **is_powerpill**. **x** and **y** SHOULD be positive float between 0 and 1023. **is_foregin** and **is_powerpill** SHOULD be a boolean value either 0 or 1.

The contents of a PACMAN_UPDATE message

- Type: **PACMAN_UPDATE**
- Value: **postion x, y** and **direction**, **speed**. Position **x** and **y** SHOULD be a positive float between 0 and 1023. **Direction** SHOULD be an integer value between 0 and 4. **Speed** should be a floating value greater or equal to 0 represent speed in m/s.

The contents of a GHOST_UPDATE message

- Type: **GHOST_UPDATE**
- Value: **ghost_num**, **position x and y**, **direction**, **speed**. **ghost_num** SHOULD be an integer in range 0 to 4 (defined in pacman game). Position **x** and **y** SHOULD be a positive float between 0 and 1023. **Direction** SHOULD be an integer value between 0 and 4. **Speed** should be a floating value greater or equal to 0.

Message Timing

All messages SHOULD be sent every 20ms. If a computer cannot maintain 50 frames per second, then all messages MAY be sent once per frame.

Message Encoding

Messages are fixed format, binary encoded, with all integer fields send in network byte order (i.e, big endian order). As as message type is fixed forward, no explicit length field is required. More than one message MAY be send consecutively in a single packet.

Type table reference

```
# In decimal

  MAZE_UPDATE = 0
  SCORE_UPDATE = 1
  LIVES_UPDATE = 2
  STATUS_UPDATE = 3
  PACMAN_ARRIVED = 4
  PACMAN_LEFT = 5
  PACMAN_DIED = 6
  PACMAN_GO_HOME = 7
  GHOST_EATEN = 8
  EAT = 9
  PACMAN_UPDATE = 10
  GHOST_UPDATE = 11
```

About sequence number used for encoding

Due to the use of UDP, messages may be lost or arrive out of order. We do not want to update the screen with out of order data. The receiver therefore keeps track of the seuqence number of the last message received of each type. If it receives a message of a specific type with a lower sequence number than the last one received, the message MUST be discarded. When performming this comparison, care must be taken to account for the potential for sequence numbers to wrap.

The squence number takes **8 bit**. However, only **7 bit** is used for actual numbers. Hence If it reaches (2^7)-1, it wraps back round to zero.

The **8th** bit is used to store the information about needs in reliability. There should only be two state.

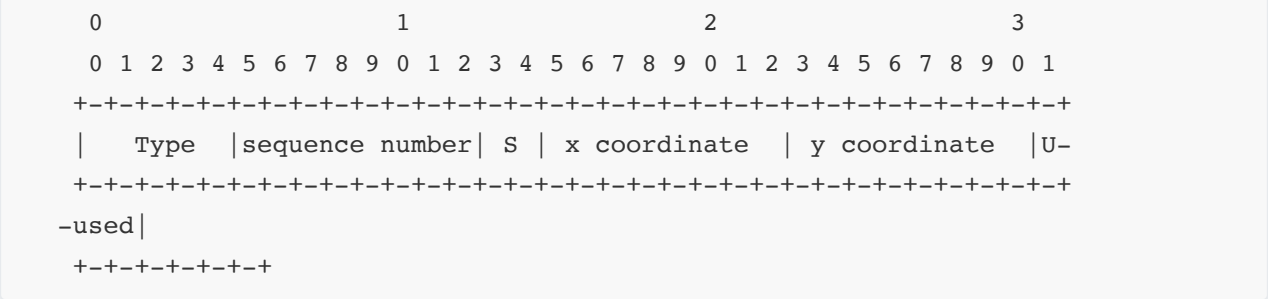
- 0: This packet doesn't need to be reliable. So it will just send the packet via UDP.
- 1: This packet needs to be reliable. Hence when a packet loss is detect, we can ask packet resend.

More handling strategy refer to previous **[UDP] Dealing with data need to be reliable VS. no need to be reliable** section in this specification.

Also, we MUST use two separate counter to count sequence number that require reliability and not requiring it. Hence we are able to track lost packet number for only packet that required to be reliable.

MAZE_UPDATE message format

MAZE_UPDATE message consist of 4 bytes, encoded as follows:



Type: **5 bit** type field. Type = MAZE_UPDATE message SHOULD have decimal value 0.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

S: Status. **2 bit** unsigned integer in big-endian byte order.

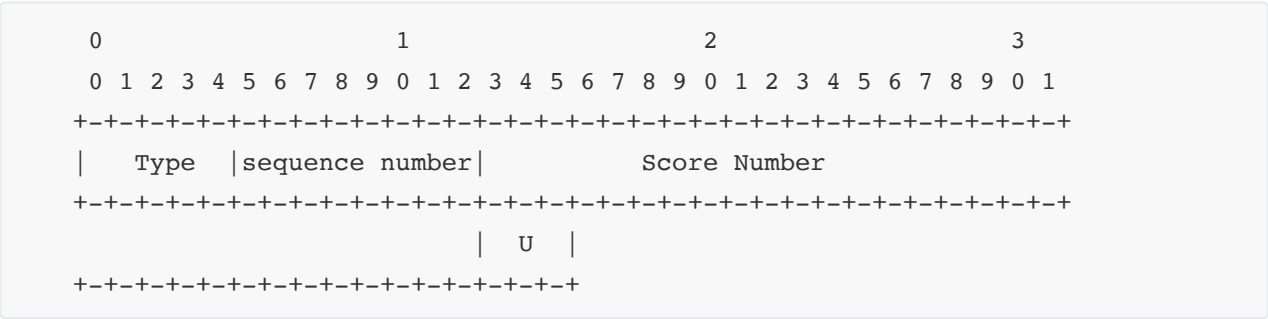
x coordinate: **8 bit** unsigned integer in big-endian byte order.

y coordinate: **8 bit** unsigned integer in big-endian byte order.

U-used: **3 bit**, not used, but needed to maintain byte alignment.

SCORE_UPDATE message format

SCORE_UPDATE message consist of 6 bytes, encoded as follows:



Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

Score Number: **32 bit** unsigned Integer in big-endian byte order

U: **3 bit**, not used, but needed to maintain byte alignment.

LIVES_UPDATE message format

LIVES_UPDATE message consist of 4 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number|          Lives_Number         |  U  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

Lives_Number: **16 bit** unsigned Integer in big-endian byte order. **first 8 bit** is used for lives of our pacman, the next **8 bit** is used for lives of the other's pacman.

U: **3 bit**, not used, but needed to maintain byte alignment.

STATUS_UPDATE message format

STATUS_UPDATE message consist of 3 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number|    cur_stat   |  U  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field. Type = STATUS_UPDATE message SHOULD have decimal value 3.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

cur_stat: current_status. **8 bit** unsigned Integer in big-endian byte order.

U: **3 bit**, not used, but needed to maintain byte alignment.

PACMAN_ARRIVED/LEFT/DIED/GO_HOME message format

This encoding applys to **PACMAN_ARRIVED** / **PACMAN_LEFT** / **PACMAN_DIED** / **PACMAN_GO_HOME** messages.

This encoding consist of 2 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number|  U  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field.

- Type = PACMAN_ARRIVED message SHOULD have decimal value 4.
- Type = PACMAN_LEFT message SHOULD have decimal value 5.
- Type = PACMAN_DIED message SHOULD have decimal value 6.
- Type = PACMAN_GO_HOME message SHOULD have decimal value 7.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

U: **3 bit**, not used, but needed to maintain byte alignment.

GHOST_EATEN message format

GHOST_EATEN message consist of 2 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number|  G  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

G: ghost number, **3 bit** unsigned Integer in big-endian byte order. (Since the range SHOULD in 0-4). And use an extra bit to maintain byte alignment.

EAT message format

EAT message consist of 4 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number|F|P| x coordinate  | y coordinate  |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

F: is_foregin. **1 bit** Boolean in big-endian byte order. **1** means the object is FOREGIN and **0** means not.

P: is_powerpill. **1 bit** Boolean in big-endian byte order. **1** means the object is a powerpill and **0** means not.

x coordinate: **8 bit** unsigned integer in big-endian byte order.

y coordinate: **8 bit** unsigned integer in big-endian byte order.

U-used: **1 bit**, not used, but needed to maintain byte alignment.

PACMAN_UPDATE message format

PACMAN_UPDATE 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type  |sequence number| x coordinate  | y coordinate  | DIR |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      SPD      |
+-+-+-+-+-+-+-+-+
```

Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

x coordinate: **8 bit** unsigned integer in big-endian byte order.

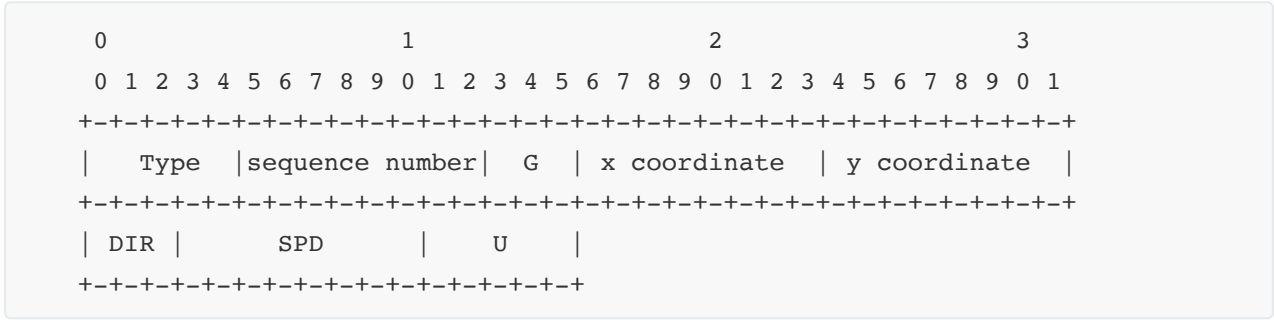
y coordinate: **8 bit** unsigned integer in big-endian byte order.

DIR: Direction of pacman. **3 bit** unsigned integer in big-endian byte order. (Since value SHOULD be in range 0-4.)

SPD: Speed of pacman. **8 bit** unsigned float in big-endian byte order.

GHOST_UPDATE message format

GHOST_UPDATE message consist of 6 bytes, encoded as follows:



Type: **5 bit** type field. Type = SCORE_UPDATE message SHOULD have decimal value 1.

Sequence number: **8 bit** unsigned Integer in big-endian byte order. Incremented by one for every new message sent. If it reaches (2^7)-1, it wraps back round to zero.

G: ghost number, **3 bits** unsigned Integer in big-endian byte order. (Since the range SHOULD in 0-4). And use an extra bit to maintain byte alignment.

x coordinate: **8 bit** unsigned integer in big-endian byte order.

y coordinate: **8 bit** unsigned integer in big-endian byte order.

DIR: Direction of pacman. **3 bit** unsigned integer in big-endian byte order. (Since value SHOULD be in range 0-4.)

SPD: Speed of pacman. **8 bit** unsigned float in big-endian byte order.

U: **5 bit**, not used, but needed to maintain byte alignment.

END