Pacman Network Specification

Author : Daiqi Wu

Email : daiqi.wu.17@ucl.ac.uk

Student Number : 17134402

Contents

- Terminology
- General Description
- Message Types
- Message Contents and Encoding
- Message Sanity Check
- Message Timing

Terminology

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

This specification also uses the terms LOCAL, AWAY, REMOTE and FOREIGN as defined in assigment PDF of ENGF0002.

This protocol is generally a PEER-TO-PEER Network Model as definied in *Distributed Application Architecture* [Sun Microsystem, 2011].

We also define addtional terms to specifiy the computer the software instance is running on :

- LOCAL_CLIENT : The LOCAL game instance that is running on localhost.
- REMOTE_CLIENT : The REMOTE instance that is connected to the LOCAL game instance.

The Pacman protocol runs over TCP and UDP depending on message types, using port 27006.

There are 12 types of messages :

- `maze_update`
- `pacman_arrived`
- `pacman_left`
- `pacman_died`
- `pacman_go_home`
- `foreign_pacman_left`

- `pacman_ate_ghost`
- `pacman_update`
- `ghost_update`
- `eat`
- `score_update`
- `status_update`

General Description

At game start, `maze_update` message is exchanged between LOCAL_CLIENT and REMOTE_CLIENT such that both clients keeps an update of each other's maze.

When LOCAL pacman is in the LOCAL maze, LOCAL_CLIENT sends `eat` messages to REMOTE_CLIENT such that REMOTE_CLIENT updates their REMOTE maze, and vice versa.

When LOCAL pacman is AWAY, a `pacman_arrived` message is sent to REMOTE_CLIENT. Afterwards, `pacman_update` will be sent to REMOTE_CLIENT to display our AWAY pacman.

Since CLIENTs always keep an updated copy of REMOTE maze, the eating done by LOCAL pacman is calculated by LOCAL_CLIENT and updates REMOTE_CLIENT via `eat` messages at all times.

When LOCAL pacman is AWAY, `ghost_update` message will be received from REMOTE_CLIENT as it detects a FOREIGN pacman. Then, LOCAL_CLIENT is responsilbe for calculating REMOTE ghosts and their interactions with our AWAY pacman.

If a REMOTE ghost is eaten by AWAY pacman, `pacman_ate_ghost` is sent to update REMOTE_CLIENT. If a AWAY pacman is eaten by a REMOTE ghost, LOCAL_CLIENT will update REMOTE_CLIENT with `pacman_died`.

When LOCAL pacman returns from REMOTE maze, LOCAL_CLIENT will send a `pacman_left` to inform REMOTE_CLIENT of stop displaying the FOREIGN pacman.

When a level fails (e.g. one of the player runs out of lives) or the game progresses to the next level, REMOTE_CLIENT sends a `pacman_go_home` message to LOCAL_CLIENT to forcibly return their FOREIGN pacman. LOCAL_CLIENT will confirm this leave by replying a `foreign_pacman_left.`

When the score of the LOCAL pacman changes, LOCAL_CLIENT shall update the REMOTE_CLIENT by sending a `score_update` message.

In the following scenario when :

- players fail a level,
- game progresses to a next level,

- game gets initialised,
- the state of ghosts change,

a `status_update` message will be sent from the LOCAL_CLIENT to inform the REMOTE_CLIENT of its gamestate and vice versa.

Message Types

Among the above 12 types of messages,

- `maze_update`,
- `status_update`,
- `pacman_arrived`,
- `pacman_left`,
- `pacman_died`,
- `pacman_go_home`,
- `foreign_pacman_left`,
- `pacman_ate_ghost`

are sent using TCP while

- `pacman_update`,
- `ghost_update`,
- `eat`,
- `score_update`

are sent using UDP.

The reason for this is that `maze_update` is called less common and sends large amount of data at once. Other game state messages sent through TCP are generally less commonily called every round.

The messages sent using UDP are usually update messages which requires a more timely delivery of information. Since game state updates every 20 ms, it is acceptable to lose few packages since newer packages will make up the loss.

Message Contents and Encoding

Messages transmitted using TCP

`maze_update`

`maze_update` message consists of a 166 byte package as follows :

0 - 3	4 - 1253	1254 - 1303	1304 - 1323	1324 - 1327
M_TYPE	MAZE_MATRIX	PILL_LOC	END_LOC	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `maze_update` has decimal value 0.
- MAZE_MATRIX : a 1250 bit field that consists of a 25 * 25 matrix with 2 bits per element. Each element have a decimal value of 0, 1, 2 which indicates wall, road with food and road without food.
- PILL_LOC : a 50 bit field that consists of 5 pairs of coordinates X Y. The coordinate value X Y are both 5 bit unsigned integer describing the location of power pills.
- END_LOC : a 20 bit field that consists of 2 pairs of coordinates X Y with same data type as PILL_LOC that desribes the end point of the tunnel A and B.
- UNUSED : a 4 bit field used for bit alignment.

`status_update`

`status_update` message consists of a byte package as follows :

0 - 3	4 - 6	7
M_TYPE	G_STATUS	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `status_update` has decimal value 1.
- G_STATUS : a 3 bit unsigned integer that specifies the updating gamestate.

Decimal Value	Gamestate
0	STARTUP
1	CHASE
2	FRIGHTEN
3	GAME_OVER
4	NEXT_LEVEL_WAIT
5	READY_TO_RESTART
6	ILLEGAL
7	ILLEGAL

  Note : If received G_STATUS of 6 or 7, CLIENT should send `status_update` with G_STATUS of 0 to reinitialise the game.

- UNUSED : a 1 bit field used for bit alignment

`pacman_arrived`

`pacman_arrived` message consists a byte package as follows :

0 - 3	4 - 7
M_TYPE	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `pacman_arrived` has decimal value 2.
- UNUSED : a 4 bit field used for bit alignment

`pacman_left`

`pacman_left` message consists of a byte package as follows :

0 - 3	4 - 7
M_TYPE	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `pacman_left` has decimal value 3.
- UNUSED : a 4 bit field used for bit alignment

`pacman_died`

`pacman_died` message consists of a 2 byte package as follows :

0 - 3	4 - 13	14 - 15
M_TYPE	DIE_LOC	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `pacman_died` has decimal value 4.
- DIE_LOC : a pair of coordinate X Y with two unsigned 5 bit integer to describe the death location of the AWAY pacman on the REMOTE maze.
- UNUSED : a 2 bit field used for bit alignment.

`pacman_go_home`

`pacman_go_home` message consists of a byte package as follows :

0 - 3	4 - 7
M_TYPE	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `pacman_go_home` has decimal value 5.
- UNUSED : a 4 bit field used for bit alignment.

`foreign_pacman_left`

`foreign_pacman_left` message consists of a byte package as follows :

0 - 3	4 - 7
M_TYPE	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `foreign_pacman_left` has decimal value 6.
- UNUSED : a 4 bit field used for bit alignment.

`pacman_ate_ghost`

`pacman_ate_ghost` message consists of a 2 byte package as follows :

0 - 3	4 - 13	14 - 15
M_TYPE	EAT_LOC	G_NUM

- M_TYPE : a 4 bit type field that specifies message types. `pacman_ate_ghost` has decimal value 7.
- EAT_LOC : a pair of coordinate X Y with two unsigned 5 bit integer to describe the ghost eating location of the AWAY pacman on the REMOTE maze.
- G_NUM : a unsigned 2 bit integer to describe the color of the ghost that is being eaten by the AWAY pacman on the REMOTE maze.

Decimal Value	Eaten Ghost Color
0	RED
1	BLUE
2	YELLOW
3	PINK

Message Transmitted using UDP

`pacman_update`

`pacman_update` message consists of a 5 byte package as follows :

0 - 3	4 - 19	20 - 29	30 - 31	32 - 33	34 - 39
M_TYPE	SEQ	P_LOC	P_DIR	P_SPD	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `pacman_update` has decimal value 8.
- SEQ : a 16 bit unsigned integer that increases by one for every new UDP message sent. If it reaches 65535, it wraps back round to 0.
- P_LOC : a pair of coordinate X Y with two 5 bit unsigned integer to describe the location of the AWAY pacman.
- P_DIR : a 2 bit unsigned integer to describe the direction AWAY pacman currently faces.

Decimal Value	Pacman Direction
0	UP
1	LEFT
2	DOWN
3	RIGHT

- P_SPD : a 2 bit unsigned integer to describe the speed of the AWAY pacman.
- UNUSED : a 4 bit field used for bit alignment.

`ghost_update`

`ghost_update` message consists of a 10 byte package as follows :

0 - 3	4 - 19	20 - 21	22	23 - 78	79
M_TYPE	SEQ	G_ALIVE	G_MODE	G_INFO (G_LOC, G_DIR, G_SPD)	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `ghost_update` has decimal value 9.
- SEQ : a 16 bit unsigned integer that increases by one for every new UDP

message sent. If it reaches 65535, it wraps back round to 0.

- G_ALIVE : a 2 bit unsigned integer that indicates the number of ghosts alive in frightened mode (dead ghosts will return to base and revive).

  Note : If received G_ALIVE equals or more than 4, CLIENT should send `status_update` with G_STATUS of 3 to terminate the game and check for errors.

- G_MODE : a 1 bit boolean value that indicates whether in frightened mode or chase mode.

Boolean Value	Status
TRUE	FRIGHTEN
FALSE	CHASE

- G_INFO : a 56 bit field which contains 4 tuples of a pair of X Y coordinates with two 5 bit unsigned integers indicating the location of a ghost, a 2 bit unsigned integer indicating the direction of the ghost and a 2 bit unsigned integer indicating the speed of the ghost. The corresponding tuple of each ghost is ordered in the same way as it is in `pacman_ate_ghost`. The G_LOC is orderd in the same way as `pacman_update`.
- UNUSED : a 1 bit field used for bit alignment.

`eat`

`eat` message consists of a 4 byte package as follows :

0 - 3	4 - 19	20 - 29	30	31
M_TYPE	SEQ	F_LOC	F_FOREIGN	F_POWER

- M_TYPE : a 4 bit type field that specifies message types. `eat` has demcimal value 10.
- SEQ : a 16 bit unsigned integer that increases by one for every new UDP message sent. If it reaches 65535, it wraps back round to 0.
- F_LOC : a pair of coordinate X Y with two 5 bit unsigned integers indicating the location of eaten food.
- F_FOREIGN : a 1 bit boolean value that indicates whether our LOCAL pacman is eating food on a REMOTE maze.

Boolean Value	Status
TRUE	REMOTE
FALSE	LOCAL

- F_POWER : a 1 bit boolean value that indicates whether the food ate was a powerpill.

`score_update`

`score_update` message consists of a 7 byte package as follows :

0 - 3	4 - 19	20 - 51	52 - 55
M_TYPE	SEQ	SCORE_INFO	UNUSED

- M_TYPE : a 4 bit type field that specifies message types. `score_update` has decimal value 11.
- SEQ : a 16 bit unsigned integer that increases by one for every new UDP message sent. If it reaches 65535, it wraps back round to 0.
- SCORE_INFO : a 32 bit unsigned integer that indicates the score of the LOCAL pacman.
- UNUSED : a 4 bit field used for bit alignment.

Message Sanity Check

For all locations (P_LOC, G_LOC, EAT_LOC, DIE_LOC ..) in messages, if any of the X Y coordinate is out of range [0 .. 24], CLIENT SHOULD terminate the game by sending a `status_update` with G_STATUS of 3 to terminate the game.

For all messages, M_TYPE MUST fall in range [0 .. 11], otherwise messages will be regarded illegal and discarded.

All UNUSED fields in messages MAY be filled with any value as the CLIENT wishes.

Message Timing

For all messages using TCP and `eat`, the message is sent whenever necessary.

For all messages using UDP except for `eat`, update messages are sent at a time interval of 20ms when frame rate is higher than 50 per second. Otherwise, update messages will be sent every frame.