Student No: 18100537
Pacman Networking Specification
Lilly Neubauer

Terminology

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

This protocol sends data over TCP, using port 1024.

There are 12 message types:

- maze update
- Pacman arrived
- pacman update
- Eat
- Ghost update
- Foreign pacman ate ghost
- Ghost was eaten
- Pacman left
- Foreign Pacman died
- Pacman go home
- Score update
- Status update

Client and Server

The game must be played in client and server mode, with one computer acting as the server and other computer(s) acting as the client. Once the network connection has been established, there is no difference between how the client and server behave.

The game is mostly the same as the traditional pacman game. There is a pacman that operates within a maze, eats food, power pills and can be killed by collision with any one of four ghosts. In this networked version, the pacman can traverse tunnels to visit other player's mazes, where it can also eat food, power pills and be killed by ghosts. The pacman originating in the local computer is always modelled in the local computer, even when visiting a remote computers maze. The ghosts cannot traverse tunnels. When foreign and local pacman encounter one another, they must pass through one another.

This protocol specifies the format and contents of messages sent across the network to play pacman. This allows each computer in the game to correctly model the other computers maze and pacman behaviour.

Message encoding

Messages are fixed format, binary encoded, with all integer fields sent in network byte order (i.e. big endian order). As message type is fixed forward, no explicit length field is required. More than one message may be sent consecutively in a single packet. Each message begins with a 4 bit message type signifier. It is then followed by 0 or more fields containing the binary encoding of other information.

Maze display

The position of moveable objects (pacmen, ghosts) is communicated in terms of a pixel coordinate value located on a "pixel-grid" of size 560 (max x value) x 620 (max y value). The maze is based on a 28 x 31 "maze-grid", with each maze-grid square taking up 20 x 20 'pixels' of the 'pixel-grid'. The actual number of pixels used to represent each 'pixel' on the 'pixel grid' could be adjusted by the local computer to suit different screen sizes etc.

The game frame rate must be set to be the same on all computers playing the game. Moveable objects can have a speed of between 0 and 10. Pacmen always move at either 0 (stationary) or 10 (they are moving). The number of pixels per frame that the pacmen move when moving (i.e. the number of pixels per frame that speed '10' indicates) must be decided and set as constant between all computers playing the game. Ghosts can move at different speeds (e.g. 5, 7, 8, 9).

Types of Messages

This document uses the following terminology to distinguish between visiting pacman and various 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.

1) MAZE UPDATE

The first message to be sent from all players is a MAZE UPDATE message which allows all computers to construct an identical copy of each player's maze in local memory, including location of food and power pills.
The Maze walls object is a list of binary integers between 0 and 5, where elements of the maze are represented as follows [decimal value - binary encoding - meaning]:

0 - 0000 - Space
1 - 0001 - wall
2 - 0010 - food
3 - 0011 - powerpill
4 - 0100 - tunnel A
5 - 0101 - tunnel B

The maze has 31 'rows' and 28 'columns', with information provided row by row starting at the top left corner and continuing to the right along each row and then down a row. So the first 28 decimal digits (represented by 112 binary bits, 4 bits for each decimal digit) must be interpreted as the top row of the maze, the next 28 as the row below that, and so on.

The maze picture is constructed as follows: Each element of the list represents a 20 x 20 pixel square. If the element of the list is '1', indicating a wall, the wall must be on the upper left corner of the square. The alignment of the wall (vertical, horizontal etc.) is decided by context. If the element of the list is '2' or '3', representing food or a powerpill, the centre 'pixel' coordinate of the food or powerpill is at the upper left corner of the pixel-grid square. Please see Appendix 1 for a diagram illustrating the maze construction.

When pacman exits at tunnel A it must appear at tunnel B on the remote screen. When pacman exits tunnel B it must appear at tunnel A on the remote screen.

Message length: 3476 bits
Message fields: 2

1) Message Type 0000 [4 bits]
2) Copy of Maze.walls in four bit binary encoding (where values above 0101 are unused) [3472 bits]

2) PACMAN ARRIVED

When our pacman leaves its home screen to visit the remote screen, it becomes "away". Our computer sends a 'pacman arrived' message, so the remote computer can initialise any state.
Message fields: 1

1) 0001 [4 bits]
   only one field is required, the message type field, which indicates that this is a pacman arrived message and therefore the pacman has arrived

Message length: 4 bits

3) PACMAN UPDATE

Following the pacman arrived message, our computer now sends ‘pacman update’ messages every frame to the remote computer, giving the current position, direction and speed of our pacman. These messages are not sent when the pacman is home.

Message length: 28 bits
Message fields: 5

1) Message type 0110 [4 bits]
2) x position in pixels [decimal integer from 0 to 560 encoded in 10 bit unsigned binary]
3) y position in pixels [decimal integer from 0 to 620 encoded in 10 bit unsigned binary]
4) Direction - RIGHT - 000, LEFT - 001, UP - 010 , DOWN - 011, NONE - 100 (3 bits)
5) Speed - 0 (not moving) or 1 (moving at speed 10) [1 bit]

4) EAT

There are two types of eat message:
1) An eat message is sent whenever a remote pacman eats food or power pills in its home maze, so that the local copy of the remote maze can be updated with these changes. This happens even if our pacman is currently local.
2) If our away pacman eats food or power pills on the remote screen, this is detected by the model running on our own computer, using its copy of the remote maze. Our computer then sends an ‘eat’ message to the remote computer informing it that food or a powerpill has been eaten.
The type of message being sent is indicated by the fourth message field, a boolean value. A ‘True’ value of ‘1’ indicates the first type - a foreign pacman ate food on our screen. A ‘False’ value of 0 indicates that our pacman ate food in the foreign maze.
Message length: 16 bits
Message fields: 5

1) Message type 1010 [4 bits]
2) x co-ordinate of food or powerpill eaten (value between 0 and 30 inclusive, encoded in a 5 bit unsigned binary integer)
3) y co-ordinate of food or powerpill eaten (value between 0 and 27 inclusive, encoded in a 5 bit unsigned binary integer)
4) Boolean - To determine type of ‘eat’ message - See above - 1 bit value where true is 1 and false is 0
5) Boolean - True indicates that the eaten food was a powerpill - 1 bit value where true is 1 and false is 0

5) GHOST UPDATE

While our pacman is away, the remote computer sends ‘ghost update’ messages every frame giving the position, direction, speed and mode of each ghost. Amongst other things, (like what?) thee mode includes whether the Ghose is in “FRIGHTEN” mode (having turned blue and being edible, and having the speed reduced to half). Our model uses this information to update a local model of the remote ghosts to determine if our pacman has eaten one or was killed by one.
Message length:
Message fields: 6

1) Message Type [4 bits] 0111
2) x position in pixels [decimal integer from 0 to 560 encoded in 10 bit unsigned binary]
3) y position in pixels [decimal integer from 0 to 620 encoded in 10 bit unsigned binary]
4) Direction - RIGHT - 000, LEFT - 001, UP - 010 , DOWN - 011, NONE - 100 [3 bits]
5) Speed [integer between 1 and 10 - binary encoded in 4 bits]
6) Mode [3 bits] (The following list represents: decimal integer - binary encoding - meaning. Only the binary encoding should be sent)
    0 - 000 - SCATTER
    1 - 001 - CHASE
    2 - 010 - FRIGHTEN
    3 - 011 - FRIGHTEN_TRAPPED
    4 - 100 - EYES
    5 - 101 - REMOTE

6) FOREIGN PACMAN ATE GHOST
This is sent to update the remote system when our model detects that our away pacman has eaten a remote ghost.
Message length: 6 bits
Message fields: 2

1) Message Type 1001 [4 bits]
2) Number of eaten ghost [2 bits]- this is a decimal number between 0 and 3 inclusive, encoded in 2 binary bits, to indicate which is the ghost that has been eaten.
    00 - ghost '0' was eaten
    01 - ghost '1' was eaten
    10 - ghost '2' was eaten
    11 - ghost '3' was eaten

7) GHOST WAS EATEN
This is sent to indicate that a foreign pacman visiting the local screen has eaten one of the ghosts on the local model.
Message length: 6 bits
Message fields: 2

1) Message Type 1000 [4 bits]
2) Number of eaten ghost [2 bits]- this is a decimal number between 0 and 3 inclusive, encoded in 2 binary bits, to indicate which is the ghost that has been eaten.
    00 - ghost '0' was eaten
    01 - ghost '1' was eaten
    10 - ghost '2' was eaten
    11 - ghost '3' was eaten

8) FOREIGN PACMAN DIED
If our model detects that our AWAY pacman was killed by a REMOTE ghost, it sends a "foreign pacman died" message.
Message length: 4 bits
Message fields: 1

1) 0011 [4 bits] (indicates that our away pacman died remotely)

9) FOREIGN PACMAN LEFT
If our model detects that our AWAY pacman has traversed the tunnel again, and is now LOCAL, it sends a "foreign pacman left" message. The remote side will stop displaying the pacman.
Message length: 4 bits
Message fields: 1

1) 0010 [4 bits]

10) PACMAN GO HOME
Some events require than our AWAY pacman be forcibly sent home. This happens when the level is competed on the remote screen, for example. The remote system sends a "pacman go home" message. Our system then resets our pacman to LOCAL, and sends a a "foreign pacman left" message in reply.
Message length: 4 bits
Message fields: 1

1) 0101 [4 bits]

11) SCORE UPDATE
This is sent whenever our pacman's score changes, whether it is home or away. Score increases by 200 whenever a pacman eats a ghost, and increases by 10 every time it eats a piece of food or a powerpill.
Message fields: 2

1) Message Type 1011 [4 bits]
2) Score [14 bit unsigned binary integer - max score as a decimal integer is therefore 16,383, which should be far beyond the possibility of the game]

Message length: 18 bits
Message type signifier:

12) STATUS UPDATE

The local game board also has states associated with it.

- STARTUP
- CHASE
- FRIGHTEN
- GAME OVER
- NEXT LEVEL WAIT
- READY TO RESTART

Changes between these states are communicated using “status update” messages.

Gameplay only happens in CHASE and FRIGHTEN state (the different being whether a powerpill has recently been eaten). The software is in STARTUP state while playing the startup jingle.

If either player loses their last life, the game ends. The losing player’s computer goes to GAME OVER state, and sends a status update message. The other side then also moves to GAME OVER state.

From GAME OVER state, if the local player presses “r” to restart, the local computer goes to READY TO RESTART state and sends an update message. The game restarts when the second player also presses “r”, and sends a replying “READY TO RESTART” status update.

When a level is cleared on a screen, that screen’s system goes to NEXT LEVEL WAIT while it plays the jingle and the player gets ready. Completing a level does not affect the level being played on the other screen, except the pacmen positions are reset.

Message fields: 2

1) Message Type
2) Status indicator:
    - 0 - 000 - STARTUP
    - 1 - 001 - CHASE
    - 2 - 010 - FRIGHTEN
    - 3 - 011 - GAME OVER
    - 4 - 100 - NEXT LEVEL WAIT
    - 5 - 101 - READY TO RESTART

Message length: 7 bits

Message type signifier: 1100

Appendix 1: Maze Construction

The following diagram can be used to understand how the 560 x 620 pixel grid and the 28 x 31 maze grid should be re-constructed on screen. The graphical interpretation of the game may be decided by the user, as long as both grids are aligned in the correct ratio.