Distributed Pac-Man

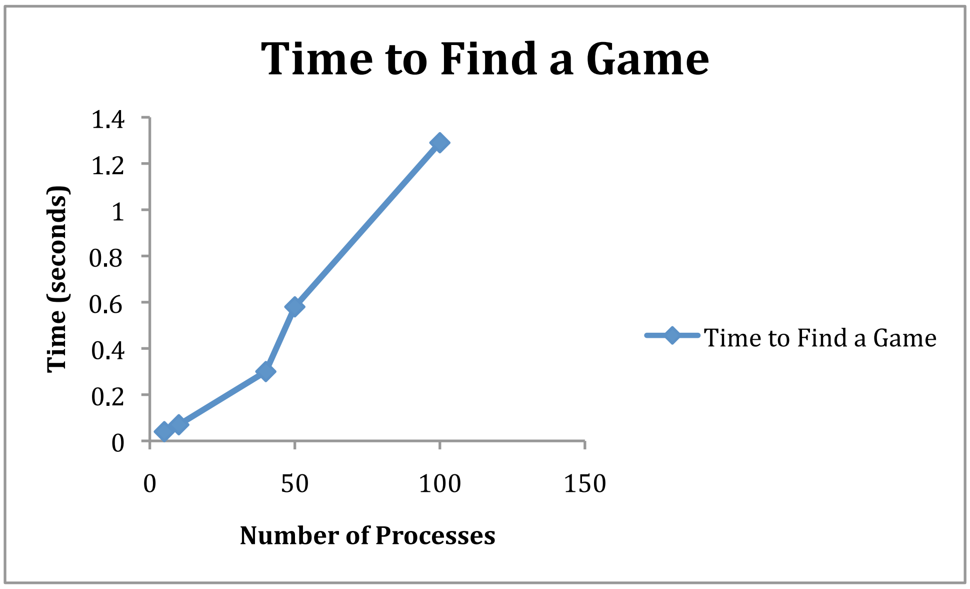
CS 425

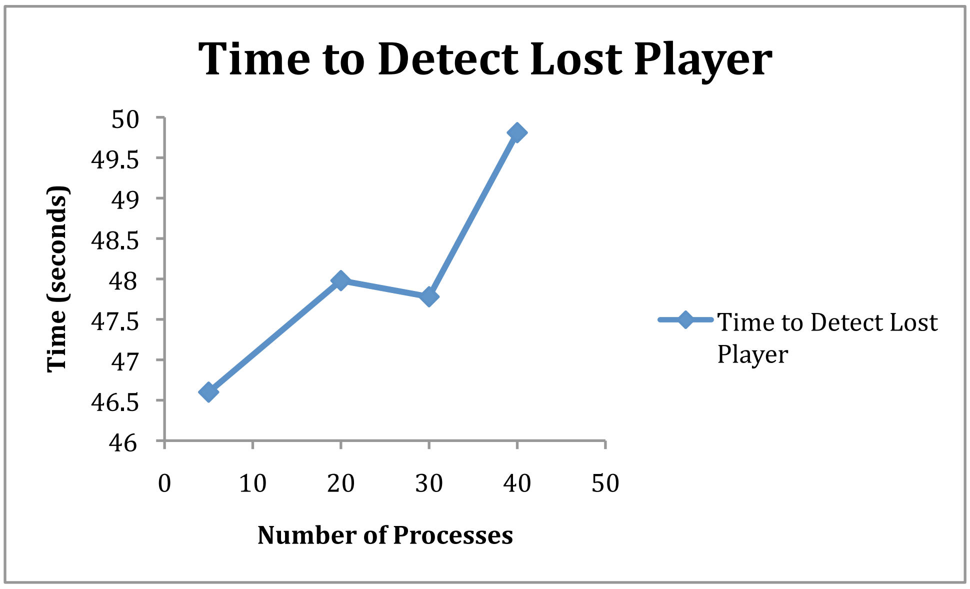
Myles Megyesi, Paul Schorfheide, Thomas Thompson

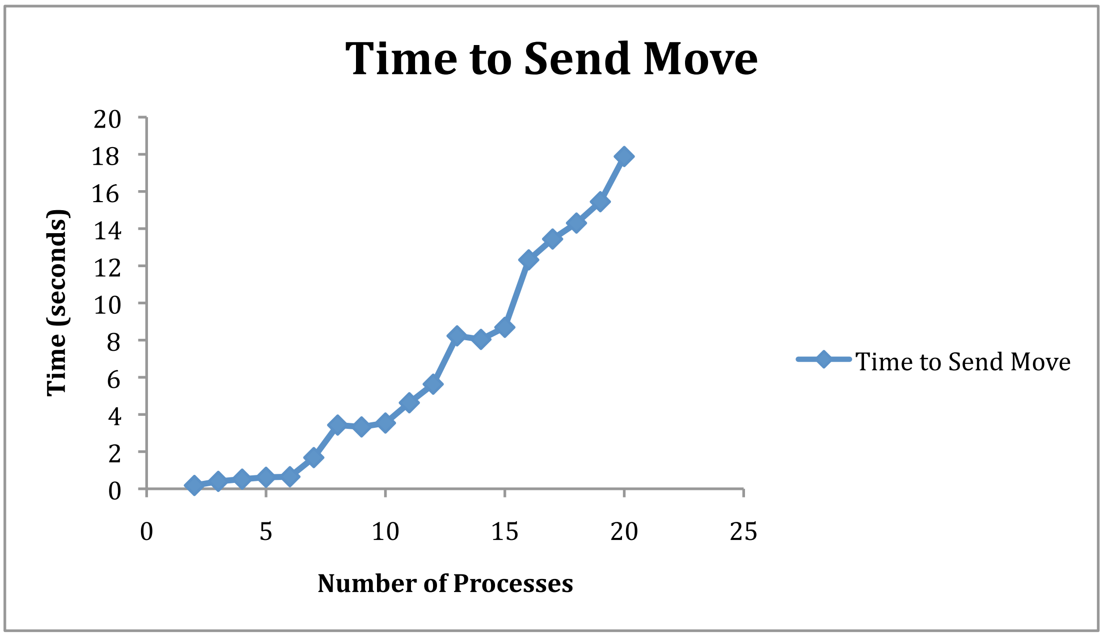
Contents

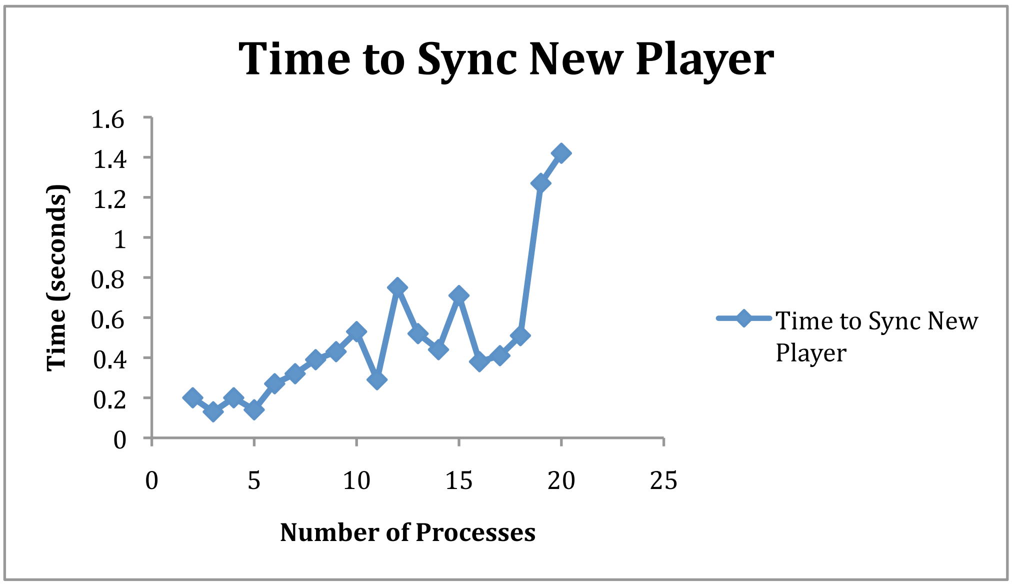
Chapter 1

Perfomance Metrics






Time to Send Move
Time (seconds)
20
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
Number of Processes
Time to Send Move


Time to Sync New Player
Time (seconds)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
Number of Processes
Time to Sync New Player

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Package board

Defines the board.

Classes

- class board

  A class to define the board.

Variables

- dictionary bops = {'FLOOR':0, 'WALL':1, 'DOT':2, 'SUPER_DOT':3}

  Defines options for tiles on the board.

- dictionary dirs = {'LEFT':0, 'RIGHT':1, 'UP':2, 'DOWN':3}

  Defines the directions for movement.

4.1.1 Detailed Description

Defines the board.

Author

Myles Megyesi

4.2 Package client

Higher level client handling group management.

Classes

- class client

  a higher level client handling group management

Variables

- tuple NAMESERVER = socket.gethostbyname(socket.gethostname())

  the default server name

- int NSPORT = 5555

  the default server port

- tuple file_lock = threading.Lock()

  lock for the log file

- int TIMEOUT = 90

  the default timeout interval

4.2.1 Detailed Description

Higher level client handling group management.

Author

Paul Schorfheide

4.3 Package game

Controls the game state and handles the operation of the game.

Classes

- class state

  Defines the player's state.

- class game

  A class to control the game.

Functions

- def kbpass

  get input from keyboard

- def intervalExecute

  Executes a function repeatedly at the given interval.

Variables

- dictionary dirs = {'LEFT':0, 'RIGHT':1, 'UP':2, 'DOWN':3}

  Defines the directions for movement.

- dictionary sops = {'PACMAN':0, 'GHOST':1}

  Defines the player types.

- tuple board = board.board()

  The global playing board.

- tuple mlock = threading.RLock()

  A semaphore used to when pushing or popping the messages queue.

- int update_interval = 1

  The interval at which to run the the game loop.

- int pacx = 0

  store pac's position

- int pacy = 0

  store pac's position

4.3.1 Detailed Description

Controls the game state and handles the operation of the game.

Author

Myles Megyesi

4.3.2 Function Documentation

4.3.2.1 def game.intervalExecute (*interval*, *func*, *args*, *argd*)

Executes a function repeatedly at the given interval.

Parameters

interval executes func(∗args, ∗∗argd) each interval

Returns

a callable object to enable you terminate the timer

4.4 Package matchmaker

Controls the interface with the matchmaking server.

Classes

- class matchmaker

  A class to help control the interface with the matchmaking server.

4.4.1 Detailed Description

Controls the interface with the matchmaking server.

Author

Paul Schorfheide

4.5 Package server

Defines the server.

Functions

- def listenForRequests

  the listener

- def joinGame

  handle join requests from client

- def addPlayer

  increment the player waiting count for a game

- def makeTimer

  start a timer to cancel a game

- def parseRequest

  parse a message from a client

- def changeLeader

  change the leader of a game

- def clearGame

  remove a game

- def logAndSend

  send a message and log the send event

- def log

  log a message

- def parseAddr

  parse an address from a message

Variables

- list games = [ ]

  The list of games waiting for players.

- int LISTEN_PORT = 5555

  The port to listen on.

- string LOGFILE_NAME = 'server.log'

  The name of the file to log messages to.

- int TIMEOUT = 10

The number of seconds to keep games in the queue.

- int logfile = 0

  The logfile handle.

4.5.1 Detailed Description

Defines the server.

Author

Paul Schorfheide

4.5.2 Function Documentation

4.5.2.1 def server.addPlayer (*client*, *client_addr*)

increment the player waiting count for a game

Parameters

client the client socket

client_addr the address of the client

4.5.2.2 def server.changeLeader (*old*, *new*)

change the leader of a game

Parameters

old the old leader

new the new leader

4.5.2.3 def server.clearGame (*game*)

remove a game

Parameters

game the game to remove

4.5.2.4 def server.joinGame (*client*, *client_addr*)

handle join requests from client

Parameters

client the client socket

client_addr the address of the client socket

4.5.2.5 def server.log (*s*)

log a message

Parameters

s the message to log

4.5.2.6 def server.logAndSend (*client*, *client_addr*, *msg*)

send a message and log the send event

Parameters

client the socket to send to
client_addr the address to send to
msg the message to send

4.5.2.7 def server.makeTimer (*game*, *create* = `False`)

start a timer to cancel a game

Parameters

game the game to wait on
create whether or not to create a timer if one does not exist

4.5.2.8 def server.parseAddr (*s*)

parse an address from a message

Parameters

s the string to parse

4.5.2.9 def server.parseRequest (*s*, *client*, *client_addr*)

parse a message from a client

Parameters

s the message
client the client socket
client_addr the address of the client socket

Chapter 5

Class Documentation

5.1 board.board Class Reference

A class to define the board.

Public Member Functions

- def __init__

    Constructor.

- def canMove

    Given a direction and your current position, returns whether you a making a valid move.

- def eatDot

    Removes a dot from the board, updates score.

- def pacmanStart

    Defines the starting position for a PACMAN player.

- def ghostStart

    Defines the starting position for a Ghost player.

- def pacScores

    Returns pac's score.

- def ghostScores

    Returns ghost's score.

Public Attributes

- board

    An array representing the board.

- totalScore

The total game score.

- pacScore

  PacMan's score.

5.1.1 Detailed Description

A class to define the board.

5.1.2 Member Function Documentation

5.1.2.1 def board.board.canMove (*self*, *dir*, *x*, *y*)

Given a direction and your current position, returns whether you a making a valid move.

Parameters

dir The direction to move
(x,y) A tuple of your x and y coordinates

Returns

A boolean indicating a valid move

5.1.2.2 def board.board.eatDot (*self*, *x*, *y*)

Removes a dot from the board, updates score.

Parameters

(x,y) A tuple of x and y coords

5.1.2.3 def board.board.ghostStart (*self*)

Defines the starting position for a Ghost player.

Returns

A tuple coordinates and position of the Ghost start

5.1.2.4 def board.board.pacmanStart (*self*)

Defines the starting position for a PACMAN player.

Returns

a tuple coordinates and position of the PACMAN start

The documentation for this class was generated from the following file:

- src/board.py

5.2 client.client Class Reference

a higher level client handling group management

Public Member Functions

- def findGame

  connect to a new game

- def disconnect

  disconnect from the current game

- def getSelf

  return the (ip, port) for this client

- def getLeader

  return the leader for this client

- def getPlayers

  return the other players in the game

- def send

  send a message to another player

- def sendToAll

  helper to send a message to all clients

- def __init__

  constructor

- def log

  log a message to a file

5.2.1 Detailed Description

a higher level client handling group management

5.2.2 Member Function Documentation

5.2.2.1 def client.client.__init__ (*self*, *servername* = `socket.gethostbyname(socket.gethostname())`, *port* = `5555`, *onMessageReceived* = `None`, *onPlayerAdded* = `None`, *onPlayerRemoved* = `None`, *onLeaderChange* = `None`, *isSafe* = `True`)

constructor

Parameters

servername the server ip

port the server port
onMessageReceived the message received handler
onPlayerAdded handler for player added
onPlayerRemoved handler for player removed
onLeaderChange handler for when the leader is changed
isSafe determines the number of listener threads to run

5.2.2.2 def client.client.findGame (*self*)

connect to a new game

Returns

the other players in the game

5.2.2.3 def client.client.getLeader (*self*)

return the leader for this client

Returns

the (ip, port) of the leader

5.2.2.4 def client.client.getPlayers (*self*)

return the other players in the game

Returns

a list of the other players in the game

5.2.2.5 def client.client.getSelf (*self*)

return the (ip, port) for this client

Returns

the (ip, port) of this client

5.2.2.6 def client.client.log (*self*, *msg*)

log a message to a file

Parameters

msg the message to log

5.2.2.7 def client.client.send (*self*, *target*, *msg*)

send a message to another player

Parameters

target the client to send the message to
msg the message to send

5.2.2.8 def client.client.sendToAll (*self*, *msg*)

helper to send a message to all clients

Parameters

msg the message to send

The documentation for this class was generated from the following file:

- src/client.py

5.3 game.game Class Reference

A class to control the game.

Public Member Functions

- def disconnect

  Disconnects the player from the socket, used upon exit of game.

- def draw

  Draws the board on the screen, with the players.

- def __init__

  Constructor.

- def update

  The game loop.

Public Attributes

- gameOver

  game is over

- isAI

  False if human-controlled.

- gameCon

  better control of the screen

5.3.1 Detailed Description

A class to control the game.

5.3.2 Member Function Documentation

5.3.2.1 def game.game.__init__ (*self*, *server_ip*, *server_port* = `5555`, *wait_time* = `None`, *isSafe* = `True`, *printStates* = `True`, *aiType* = `False`)

Constructor.

Parameters

server_ip The IP address of the server

server_port The port of the server

wait_time The time to run the game. If not set, the game will run indefinitely

isSafe Boolean to toggle the timeout threads on and off

printStates Boolean to toggle the drawing on and off

aiType Boolean to toggle the AI

The documentation for this class was generated from the following file:

- src/game.py

5.4 matchmaker.matchmaker Class Reference

A class to help control the interface with the matchmaking server.

Public Member Functions

- def getLeader

  Return the current game leader.

- def getAddress

  returns the (ip, port) of the current client

- def changeLeader

  change the current leader

- def getPlayers

  get a list of all other players

- def removePlayer

  remove a player from the game

- def findGame

  join a new game if not in one

- def disconnect

  disconnect from the current game

- def send

  send a message to a client

- def __init__

  constructor

5.4.1 Detailed Description

A class to help control the interface with the matchmaking server.

5.4.2 Member Function Documentation

5.4.2.1 def matchmaker.matchmaker.__init__ (*self*, *servername* = `socket.gethostbyname(socket.gethostname())`, *port* = `5555`, *handler* = `None`, *onLeaderChanged* = `None`)

constructor

Parameters

servername the ip address of the server

port the server port

handler a function to handle (high level) messages

onLeaderChanged function to be called when a new leader is elected

5.4.2.2 def matchmaker.matchmaker.findGame (*self*)

join a new game if not in one

Returns

the other players in the game

5.4.2.3 def matchmaker.matchmaker.getAddress (*self*)

returns the (ip, port) of the current client

Returns

the (ip, port) of the current client

5.4.2.4 def matchmaker.matchmaker.getLeader (*self*)

Return the current game leader.

Returns

the game's leader

5.4.2.5 def matchmaker.matchmaker.getPlayers (*self*)

get a list of all other players

Returns

the (ip, port) of all other players

5.4.2.6 def matchmaker.matchmaker.removePlayer (*self*, *player*)

remove a player from the game

Parameters

player the player to add

5.4.2.7 def matchmaker.matchmaker.send (*self*, *addr*, *message*)

send a message to a client

Parameters

addr the address to send the message to

message the message to send

The documentation for this class was generated from the following file:

- src/matchmaker.py

5.5 game.state Class Reference

Defines the player's state.

Public Member Functions

- def changeType

  Changes the type of player.

- def getState

  State getter.

- def setState

  State setter.

- def move

  Move the player one space in a given direction.

- def __init__

  Constructor.

5.5.1 Detailed Description

Defines the player's state.

5.5.2 Member Function Documentation

5.5.2.1 def game.state.__init__ (*self*, *type*)

Constructor.

Parameters

type The type to make the player

5.5.2.2 def game.state.changeType (*self*, *type*)

Changes the type of player.

Used during leader election

Parameters

type The player type to change to

5.5.2.3 def game.state.getState (*self*)

State getter.

Returns

The state of the player

5.5.2.4 def game.state.move (*self*, *dir*)

Move the player one space in a given direction.

Parameters

dir The direction to move the player

5.5.2.5 def game.state.setState (*self*, *x*, *y*, *dir*, *type*)

State setter.

Parameters

x The X coordinate

y The Y coordinate

dir the direction to face

type The type of the player

The documentation for this class was generated from the following file:

- src/game.py