NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
SEMESTER EXAMINATION FOR
Semester 2 AY2009/2010

CS4344 Networked and Mobile Gaming

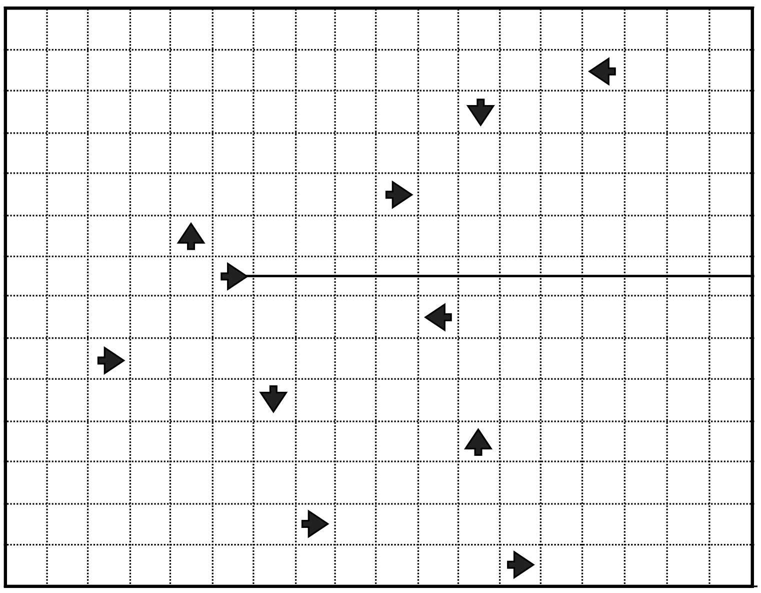
May 2010 Time Allowed 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains THREE (3) questions and comprises FOUR (4) printed pages, including this page.
2. Answer **ALL** questions. State your assumptions, if any, clearly.
3. The total marks for this paper is A HUNDRED (100).
4. Answer all questions in the answer book provided.
5. This is an **OPEN BOOK** examination.

1. (45 points) Consider the following multiplayer-player networked game, to be played within a large 2D game world consists of a grid of $k \times k$ cells. Each player controls a spaceship that occupies exactly one cell. A cell can only hold one spaceship at a time. A spaceship can face either up, down, left, or right. A spaceship can shoot a laser of *infinite* length in a straight line in the direction that it is facing, which would *immediately* damaged any spaceship that lies along the path of the laser. The *damage point* of a spaceship counts how many times the spaceship has been hit.

   The figure below shows a screenshot of the game, showing part of the game world. Each arrow represents a spaceship, facing in the same direction as the arrow. The figure shows one of the spaceship is shooting its laser.



   The objective of the game is to damage other players' spaceships as many times as possible. Players can issue three commands (*without any arguments*):

   - rotate - rotate the spaceship by 90 degrees clockwise.
   - move - move the spaceship forward by one cell in the direction that the spaceship is facing, when possible.
   - fire - shoot the laser immediately.

   An implementation of this game uses a *permissable* client/server architecture with *short circuiting*. It works as follow:

   - The states of the game consists of the position, direction, and damage point of each spaceship;
   - Each client maintains a local copy of the game states;
   - The server maintains the authorative copy of the game states;
   - Each command is simulated locally and sent to the server at the same time;
   - The server, upon receiving each command, simulates the game and sends the authorative states to each client;

- If there is any inconsistency between the server's states and a client's states, the client will overwrite its own states with the server's states.

(a) (5 points) Explain, by tracing the game states stored on two sample clients and the server along a timeline, how this implementation can lead to inconsistent positions and damage points of spaceships.

(b) (10 points) Suppose that after the fire command is issued, the laser takes a small amount of time to warm up before firing. Similarly, after the move or rotate command is issued, the spaceship's engine takes a small amount of time to start before the spaceship moves or rotates.

Explain how this change in design can help to improve the consistency of the game.

(c) (10 points) For each of the commands, move, rotate, and fire,

i. explain if it is necessary to send the command using a reliable transport protocol,

ii. and if so, is it possible to modify the command such that it need not be sent reliably? If your answer to the second question is yes, explain how the command can be modified.

(d) (20 points) Suppose we want to implement interest management so that the server only sends every update from a player p to other players that are relevant to p. For players that are not relevant to p, the server sends p's updates to them at a lower frequency.

i. (10 points) Argue why both visibility-based interest management and distance-based interest management are not appropriate in this game.

ii. (10 points) Suggest an alternative interest management scheme for this game. Specifically, explain how the server determines who are the players relevant to p.

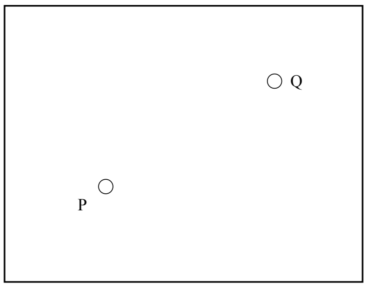
2. (20 points) Recall that frontier set is used for visibility-based interest management for a game world with lots of occlusions and can be divided into cells. A frontier of two cells X and Y consists of two sets, F_{XY} and F_{YX} such that any cell in F_{XY} is not visible from any cell in F_{YX}. An ideal frontier is one where the two sets are as large as possible and have equal size. To be more precise, we will refer to frontier set as *frontier cells.*

In a game where players move in an open area with no occlusions, visibility-based interest management and frontier set are not suitable. In such scenario, distance-based interest management are more appropriate. Here, two players are interested in each other only if they are within distance d of each other.

In this question, we change the definition of frontier set to the following so that it is suitable for distance-based interest management. A frontier of two points p and q (on a 2D plane) consists of two regions (or, equivalently, two sets of points), F_{pq} and F_{qp}, such that any point in the region F_{pq} is not within distance d away from any point in region F_{qp}. Now consider two players, P and Q, whose initial positions are p and q respectively. If P and Q are of distance more than d apart, P can move anywhere within F_{pq} without needing to update Q, and Q can move anywhere within

F_{qp} without needing to update P. We will refer to this new definition of frontier set as *frontier regions.*

(a) (10 points) The following diagram shows the top-down view of two players P and Q in the virtual world.



In the answer book, draw the diagram above, and sketch the frontier regions F_{pq} and F_{qp} on the diagram. The total area of F_{pq} and F_{qp} has to be as large as possible. Label the distance d clearly. You may assume that the distance between p and q is larger than d.

(b) (10 points) Unlike frontier cells, frontier regions cannot be precomputed offline as it depends on the positions of two players P and Q. Suggest how you can approximate F_{pq} and F_{qp} in such a way that an approximated frontier regions can be precomputed offline.

3. (35 points) Recall that a game with n players that adopts point-to-point architecture requires every player to connect to $n - 1$ other players. A total of $O(n^2)$ connections need to be maintained.

To reduce the number of connections to $O(n)$, we can use a generic proxy. Every player connects to and sends event messages to the proxy. The proxy serves as a relay node that forwards messages among the players. The proxy is generic in the sense that it does not simulate any game logic nor store any game-specific states.

(a) (10 points) Suppose that the proxy forwards each message received from a player to every other players and suppose that bucket synchronization is used to synchronize the states among the players. Does the transport protocol used by the proxy to relay the messages need to support in-order delivery? Justify your answer.

(b) (15 points) Assuming that the players can trust the proxy.

Suggest how the proxy can be used to prevent look-ahead cheat without resolving to lock-step protocol.

What is the bounding factor that determines the frame rate of the game?

(c) (10 points) Suggest how the proxy can help in making message arrivals at the clients more predictable, allowing clients on mobile devices to switch their WNIC between active/suspend mode to save power.

END OF PAPER