NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
SEMESTER EXAMINATION FOR
Semester 1 AY2013/2014

CS4344 Networked and Mobile Gaming

November 2013 Time Allowed 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains FOUR (4) questions and comprises SIX (6) 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.

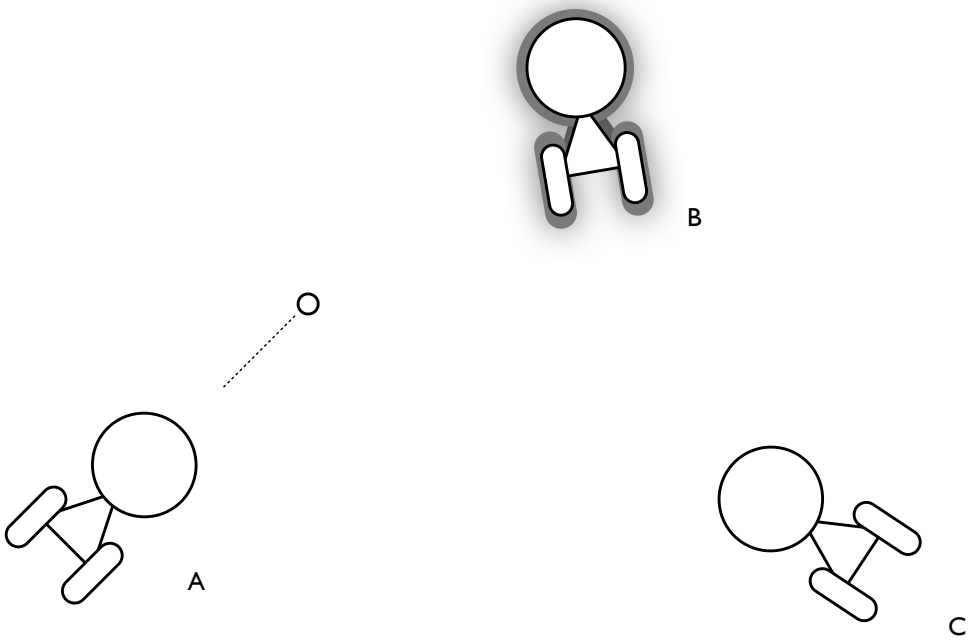

Figure 1: A screenshot of "Angry Khan". Starship A just fired a torpedo at B. Starship B has its shield on.

1. (30 points) The screenshot above (Figure 1) shows a client/server-based multi-player game called "Angry Khan," where each player controls a starship.

   A player can issue three possible types of event: (i) move: using the keyboard, the player can move the starship to any position in the game world and face any direction; (ii) raise/drop shield: A player can raise the shield of its starship, to protect itself from the opponent's torpedo. Shield starts at 100% and is reduced by 10% each time it is hit with a torpedo; the player can also drop a raised shield when appropriate; (iii) fire: a player can command its starship to shoot a torpedo at the other starships. A starship caught with the shield down or 0% shield would be destroyed if hit with a torpedo. A starship cannot fire a torpedo when its shield is raised (a common strategy is thus to raise the shield just before the torpedo hits). The torpedo always travels in a straight line. The goal of the game is to destroy as many other starships as possible.

   The game server is authorative and decides whether a torpedo hits or misses the target, and whether a shield is raised in-time or not.

   The client implements short circuiting with a *local lag* of t seconds for the fire event. In other words, after the fire event is triggered by the user, the starship takes t seconds to load the torpedo before launching the torpedo. The fire event, however, is sent immediately to the server after it is issued.

   We can assume that, other than the local lag for the fire event and the network delay for message exchanges between the server and the clients, there is no other delay in the system.

   Consider two players A and B. The one-way network delay between A and the server

is d_A, and the one-way network delay between B and the server is d_B.

Sometime during the game, A fires a torpedo at B. The game implements local perception filter for the movement of the torpedo. The torpedo moves at a slower speed when rendered at A and a faster speed when rendered at B. Player B raises the shield just before the torpedo hits B. Figure 2 shows the message exchange between player A and player B.

Assume that A and B remain still. The distance between A and B is L. The torpedo would have travelled at a speed of v without local perception filter.

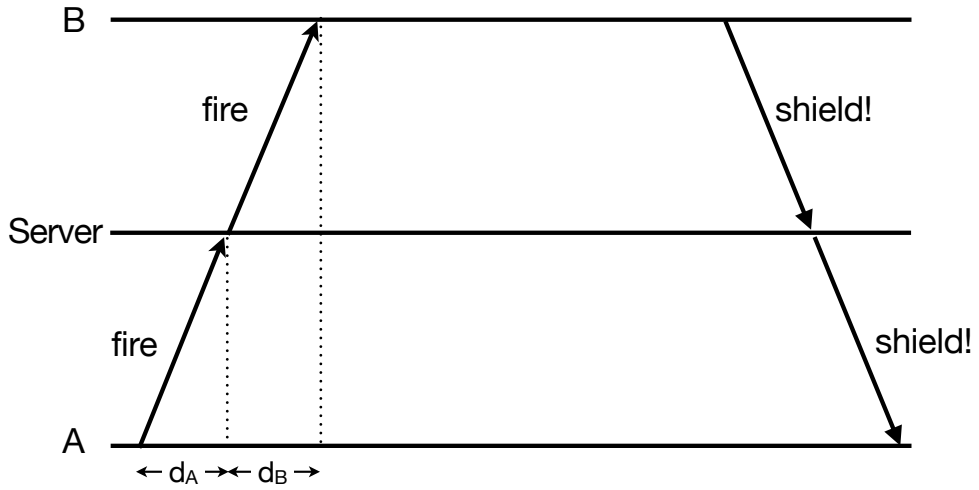

Figure 2: Timing diagram for the fire and raise shield message exchanged between A, B, and the server. Note that t is not labelled in the diagram.

(a) (10 points) At what speed should the torpedo travel at A, to ensure that when the torpedo reaches B on the screen of A, the raise shield message from B has reached A? Express your answer in terms of t, d_A, d_B, L, and v (you may not need to use every variable).

(b) (10 points) Consider a third player C that observes this sequence of events (A firing a torpedo at B, B raised its shield before the torpedo hits). The one-way delay between C and the server is d_C.

At what speed should the torpedo travel on the screen of C?

Express your answer in terms of t, d_A, d_B, d_C, L, and v (you may not need to use every variable).

(c) (10 points) Suppose now the game allows the starship to cloak, i.e., to become invisible to other starships, for a short period of time. A cloaked starship cannot raise shield nor fire, but it can still move and get hit by the opponent's torpedo.

Can the cloaking ability be exploited to reduce the number of messages exchanged between the players and the server? If so, how?

2. (30 points) Consider a real-time strategy game that supports 32 players, implemented as a point-to-point game using stop-and-wait bucket synchronization. As every player needs to send an update to every other player in every round, a heavy message overhead is incurred – about 1000 messages are exchanged every round!

   In an effort to reduce the number of messages, the following communication architecture is proposed:

   - Group the players into four groups, each with 8 players.
   - Players within a group communicate in a point-to-point manner.
   - One of the players in each group is designated as the group *leader*.
   - The four group leaders also communicate in a point-to-point manner. During a round, each leader consolidates the eight event updates from its group into one message and sends it to the other three leaders.
   - Each leader then forwards the consolidated event updates from the other three leaders to players in its group.
   - Event updates sent during a round are put into a bucket. After a predetermined lag, when all 32 event updates from all players are available, a player processes the event updates and updates the states.

   Figure 3 below illustrates the communication structure among the players.

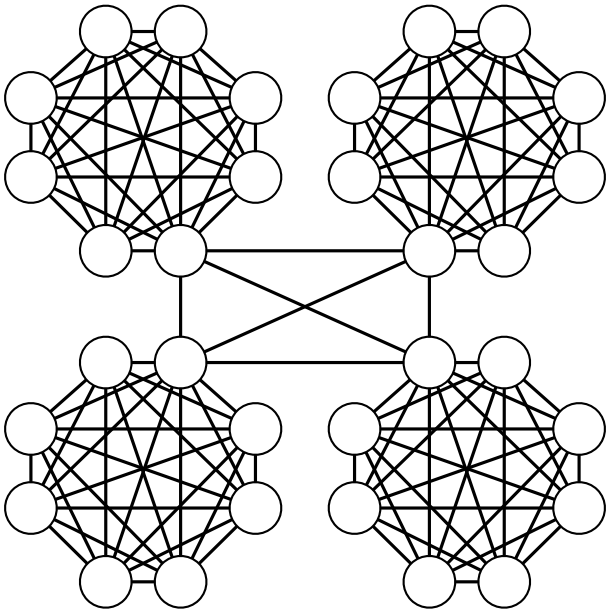

Figure 3: The proposed communication structure for the 32-player RTS game.

(a) (5 points) Estimate the number of messages exchanged in each round using the proposed scheme.

(b) (5 points) A student noted that, despite a reduction in the number of messages, about 1000 event updates are still being exchanged every round. The student then concluded that the proposed scheme still leads to the same amount of network traffic overhead as the original scheme.

Do you agree with the student? Justify your answer.

(c) (5 points) One of the key parameters of bucket synchronization is lag. Explain how you can determine the appropriate value of lag in the proposed scheme.

(d) (15 points) Sketch a method to group the players into four groups of eight each and to designate one player from each group as a leader, such that the lag is reduced.

You should first explain the intuition behind your method and how it could lead to a lower lag, before presenting the method step-by-step. You can assume that the delay between every pair of players is known.

3. (15 points) One of the drawbacks of Voronoi Overlay Network (VON) is that players still occasionally exchange position updates even when they are not within the area of interest (AOI) of each other. Such scenario occurs because a player always connects to, and exchanges position updates with, its enclosing neighbors regardless of whether the enclosing neighbors are inside its AOI or not.

(a) (5 points) Explain why it could be useful for a player to exchange position updates with an enclosing neighbor that is outside of its AOI.

(b) (10 points) Consider two players A and B that use VON for interest management. A and B have the same AOI radius and their respective AOI overlaps. A and B, however, are not in each other's AOI.

i. (5 points) Is the following statement true? "If A and B do not have any common neighbor in their AOIs, then they must be enclosing neighbors of each other."
Explain your answer with a diagram.

ii. (5 points) Is the following statement true? "If A and B are enclosing neighbors of each other, then they have no common neighbor in their AOIs."
Explain your answer with a diagram.

4. (25 points) Consider a multi-player game that is designed to be played on a local area network (LAN), where packet loss is rare and network latency is small. The game uses TCP as its transport protocol and adopts the point-to-point architecture.

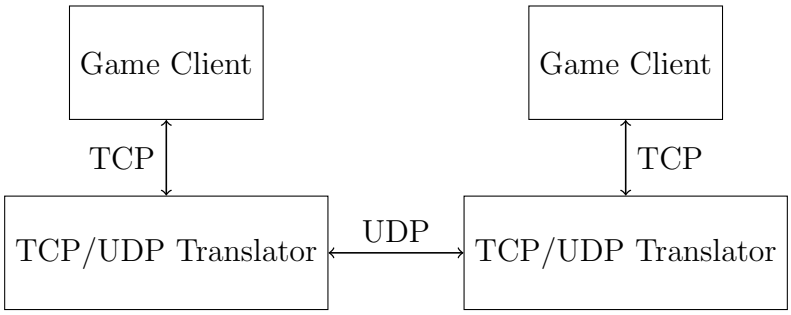

Figure 4: Using Proxies to Translates between UDP and TCP.

(a) (5 points) Explain why playing such a game over the Internet may affect its playability.

(b) (10 points) To improve the playability of such LAN game over the Internet, it has been suggested that every player installs and runs a proxy that creates a UDP tunnel. Every game client runs a corresponding proxy. An outgoing TCP packet from the game client is converted into a UDP packet by its proxy before sending it to other proxies. Incoming UDP packets at a proxy are converted back to the TCP packets by the proxy before delivering it to the corresponding game client. (See Figure 4). Explain how running such proxies that translates between TCP and UDP can lead to better playability of LAN games over the Internet.

(c) (10 points) There are two design choices for when a proxy should send the TCP ACK back to the corresponding game client:

1. when a TCP packet is received correctly at the proxy, or
2. when a TCP packet is received correctly at the receiving game client.

Discuss the pros and cons of each design choice above.

END OF PAPER