CS244B Project 1 Answers

Wei Shi
weishi@stanford.edu

April 23, 2013

1 Design evaluation

1.1 Evaluate the portion of your design that deals with starting, maintaining, and exiting a game - what are its strengths and weaknesses?

The design for starting aims for low packet traffic and good ID conflict detection.

- Strength: We have a special game phase(Join) for starting the game. Instead of boardcasting many dedicated Join packets, we choose to listen passively for other players game heartbeats. This way we don't generate many packets that stress the current players. We also use those information to generate the ID. Since our ID is a 32 bit random int. There is a small chance that our ID will conflict with another player. By listening other players first and generating a different ID, we reduce the chance of conflict so as to avoid the unpleasant restart.

- Weakness: Instead of super robust ID conflict resolution, we choose a simpler approach. If we later detect a conflict, we just clear the game state and restart the game. Although the chance of this happening is really low, especially with a good random number generator, user will lost his game state/score if there is conflict.

The design of maintaining game state aims for good consistency.

- Strength: We use a heartbeat to sync game states. Each heartbeat includes the absolute position and missile information instead of relative move instructions. This way even if there is packet loss, we still have a consistent state upon network recovery. If there is no movement we

send a heartbeat at every 200ms. If there is a movement, we send out a heartbeat immediately to sync with other players quickly. The position conflict resolution is handled locally by asking the player with lower ID to yield.

- Weakness: Since the player with lower ID always yield when there is a position conflict. This might not be fair. However, since the IDs are randomly generated every time a new game starts, it is fair across rounds. The boardcasting of heartbeat message will also be a bottleneck if game scales.

The design of exiting game state aims for good consistency.

- Strength: We use two mechanisms to detect leaving game. One is by boardcasting a Leave packet. The other is the absence of heartbeat packet. The first one ensures fast sync in good network condition while the second ensures eventual consistency in poor network condition.

- Weakness: Since we only attempt to send 1 leave packet. If that is lost, the time for other players to detect a leave might be a little long(5sec in this case).

1.2 Evaluate your design with respect to its performance on its current platform (i.e. a small LAN linked by ethernet). How does it scale for an increased number of players? What if it is played across a WAN? Or if played on a network with different capacities?

The performance in a small LAN is very good. It is only bounded by the heartbeat interval. If we set the heartbeat interval low enough, player movement and score updates feel like "instant".

If the number of player increases in a LAN, the bottleneck will be the heartbeat packets. It it reaches a point where the heartbeat alone saturates the network, the performance will degrade dramatically. However before that point, our game should scale well.

If it is played across a WAN, the gaming experience will not be that good due to the high latency. There will be many local position conflict because the movement updates have not reached when a local move is issued. Player with lower ID will from time to time be pushed away by higher ID player. Missile hits will also not register until after a delay.

On a network with different capacity, there are many aspects that can affect the game experience, eg bandwidth, packet loss rate, and latency. If any of those attributes are poor, the game experience can be affected.

1.3 Evaluate your design for consistency. What local or global inconsistencies can occur? How are they dealt with?

- Position: Although we check position conflicts locally before moving into a cell, there will be positional conflict if there is a packet loss from a player who wants to move into the same cell. When this happens, they both think they are in the same cell. This is resolved by asking the player with lower ID to move away locally and update its new position in the next heartbeat.

- Missile hits: The missile hits are reported by victims. So multiple player can think they are hit, but logically only one of them can be count as victim. We resolve it by asking victim to report to the shooter and the shooter will choose the first victim he sees and sends back a confirmation. So the player gets the confirmation knows he is the victim and others who waits for the confirmation will know that they are not victim.

- Score: Since player only tracks his own score. The shooter increases his score when he responds with a killconfirm packet. However the victim only decreases his score when he receives that killconfirm packet. In the case where that packet is lost, the victim will not consider himself killed. Therefore we have a score inconsistency. Although we have the victim keeps asking for killconfirm, in extreme poor network condition, it is still possible the packet never reaches the victim before he times out.

- Leaving: Player's leave packet might be lost. So other players will see him as idling. This is dealt with a heartbeat timeout. After the timeout, the player is removed from game.

1.4 Evaluate your design for security. What happens if there are malicious users?

Our design does not account for malicious behaviours. A malicious user can sniff the packet over the network and impersonate by setting the packet ID.

He can also report invalid positions and arbitrarily change his score since our protocol asks each player to only track his own score. Moreover, he can avoid being killed by not reporting a missile hit.