CS244B MazeWar Answers

Yunlu Li

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

   Strengths:
   In the joining phase, we listen for packets to gather player information. We generate a 16 bit random integer as rat ID. The chance for ID conflict is pretty low in this case.

   In the game playing phase, each player sends out a heart beat message periodically to drive the system to a consensus. It helps to reduce inconsistencies.

   We track both voluntary leave and accidental leave. We process leave packet explicitly, or treat a player as gone if heartbeat message hasn't been heard for 10 seconds.

   We used MissileHitACK to handle the scenario that a rat is hit by multiple missiles. It acks only one shooter.

   Weaknesses:
   Score inconsistency may occur if MissileHitACK is dropped. Only the victim's score is changed in this case, not the shooter's.

   We could handle position conflict and ID conflict better.

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?

   LAN scenario:
   Performance on a single server is pretty good. Packet drop rate is very low, and almost no delay is observed. Since network is reliable, the system has fairly good consistency.

   Multiple players:
   If the number of players increases to a point that each player needs to spend a long time on updating peers' state, heart beat packets may be dropped, and the game may be in a weird state.

   The heartbeat messages will also generate a lot of unnecessary traffic, which may lead to bad network performance.

   WAN scenario:
   Higher packet drop rate may lead to loss of missile hit / missile hit ack packets, leading to inconsistent scores among players. Heart beat messages may also be dropped, so players may observe obsolete state of peers.

   Different Network Capacity:
   If the packet drop rate increases, or latency increases, or network bandwidth decreases, the consistency of the game would be affected significantly.

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

   If the missile hit packet is dropped, it would seem weird to the player fires the missile. Neither its own score of that of the victim will change.

   If the missile hit ack packet is dropped, it's possible that only the victim decreases its score, but the shooter doesn't increase that of its own.

If heart beat message is delayed or dropped, there may be position conflict. Player may observe obsolete state of peers as well.

Another scenario we don't have enough time to deal with is multiple players trying to join the game when it's almost full. Say if there are already 7 players in the game. Rat 8 and rat 9 both try to join at the same time. Since in the joining state, they cannot hear each other's heartbeat, they cannot detect the full game error. This can be handled to some extent by increasing the capacity of the internal array of rats.

We didn't spend any effort on detecting packet corruption. Checksum could be used, but we didn't have enough time to implement it.

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

   Our design does not handle malicious behavior correctly. A malicious user can change his score arbitrarily by sending out fake packets. He can also report random position without traversing a valid path, so that he gains advantage over other players.