Lockstep protocol

Maksym Kurylovych
Special Assignment in Distributed Systems (MTAT.08.034)
Institute of Computer Science
University of Tartu
max.kurylovych@gmail.com

Abstract
Current article describes the way cheaters used look-ahead cheating method to receive advantage over other players. Among that, the article contains information about lock-step protocol, how can the above problem can be solved with it, disadvantages the protocol has and how to avoid them with asynchronous lockstep protocol. Lack of scientific based information forced author to use all the available resources to collect all the details that were necessary for the report however, current article focused on research papers written by scientists and other resources were used just for better understanding the protocol. All the articles that were used during the research mentioned in the Reference section.

1 Introduction

To understand the main idea of lock-step protocol first, we have to try to understand for what purpose it was created. Nowadays, we have a huge variety of different games we can play online when we have some free time with our friends, which is great. However, many of games demand from a player dozens of hours in front of the computer, nice reaction, analytical skills or just rely on randomness; that is why some people started searching for bugs to find the way around long gaming process to save the time and gain more in-game artefacts. In fact, it is possible to find some cheats almost to all modern games and it doesn't really matter if it is a single-player game, where you have to play a story mode, or multiplayer, where you are dropped directly into the action. The cheats might be based on in-game bags or interacting with server. Today, we will consider look-ahead cheating witch is based on fooling the server by giving it the wrong information.

2 Look-ahead cheating

As we figured out from above, cheating is an activity that modifies the game experience to give the player some advantages over the players that are playing in a "fair" game. In case of look-ahead cheating the cheating client receives his/hers unfair advantage by delaying his/hers actions to see other player actions before own one(action). Basically, current cheating occurs within peer-to-peer multiplayer architecture. Current cheating method is difficult for detection as it makes think that user is suffering from high latency. In other

words, the outgoing packet is changing by attaching a new or editing the current time which helps to wrap around the finger other players makes them think that the answer has been sent at the correct time but arrived with a delay. The solution for this is lock-step protocol.

3 Example of Look-ahead cheating

The most common and basic example of the look-ahead cheat is Rock-Paper-Scissors game. Imagine that we are playing this funny game with our friend. We and our friend have equal chances to win or to lose. However, let's try imagine that we would know what our friend would do. In that case we can reach 100% win rate.

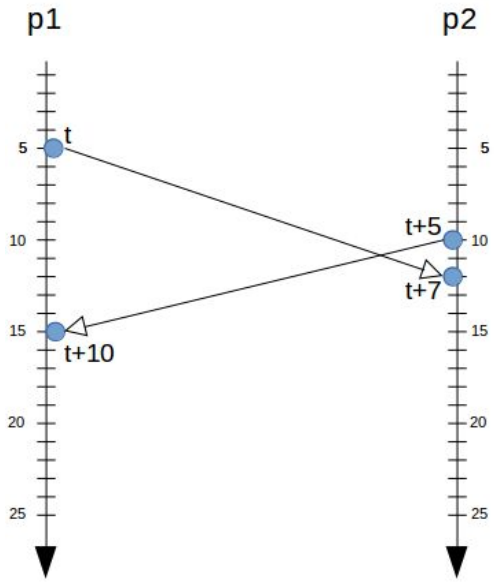

Fig.1. Fair game

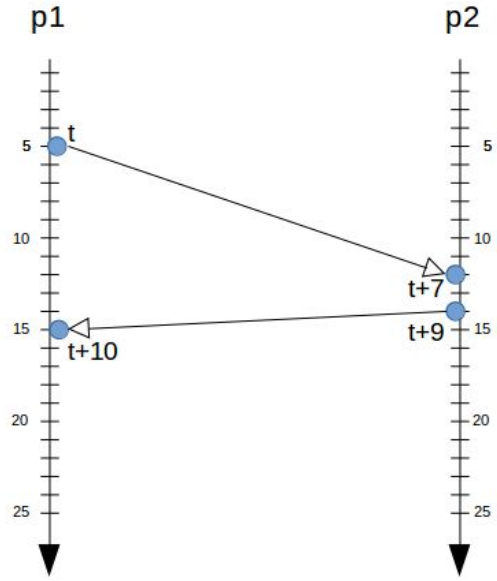

Fig.2. Unfair game

From the above figures, where p = player, t = latency time when player makes a move, we can see the example of two games: fair (fig.1) and not-fair (fig.2). Let's assume that the latency of the first player are 5 points and the difference in latency between players are also 5. (1) If they play in a fair game - then the first player can be sure that the second player will make a move before he/she receives the result move of the first player. (2) However, if the latency of the first player equal to 5 points and the latency of the second player equal to 3 points but he/she makes to believe that the latency p2 > p1 by 5, then we would see a nice example of a look-ahead cheat that allows the second player to receive the move of the first player and make a decision to win in a game.

4 Lockstep

And here we are, finally reached the chapter that describes the solution to the look-ahead cheating. Now, when we understood the problem we might go further and think about the solution to the following questions: how to fight with the above problem; how to detect a player who uses that kind of cheat to win the game?

To counter the cheating problem developers started to use different approaches and techniques and the most useful one was lockstep protocol. But what kind of technology is that and how does it work? Usually, when we are thinking about games and have they work, we imagine a Client-Server Model as it is the one that most of the games use nowadays. However our case works a bit different. This protocol based on peer-to-peer and prevent the lookahead cheat using synchronization of peer's actions, called lockstep synchronization. For example, if we want to move the character or make any other action, in authoritative model all the simulations will be on the server side. The client will send the request to the server with the information about where the character should move or what action has to be done. After that, the server would process the request and start to make changes that we have required. Additionally, the server will send the information about the new action to all the clients so all the players would have the most updated information. In fact, this must be done to all the characters in the game as in real-time strategy, for example, we have hundreds of units we have to move.

On the other hand, in a lockstep, once player has decide to move the unit, that action communicated to every client. After that, every client will perform path finding and update the characters position. Only the first communication will need to be sent over the internet as every client will have their own physics simulation and can update the characters position themselves.

From the above information we can assume that in lockstep network model every client runs the entire game simulation. The main advantage of the current model is that it reduces the amount of information that needs to be sent over the wire. Shortly, only unset inputs need to be sent to each other.

To better understand the protocol we can look at Jouni and Hakonen protocol representation in their article; pseudocode for a lockstep protocol you can see below. Based on pseudocode, let's try to figure out what is going on. As we can see, lock-step protocol solves the problem by requiring all players to announce their commitments first. After every players receive the commitments they show everyone their real move. This real move is comparing to the initial commitment to ensure that the commitment is indeed the sent action. The important are the two rules all the commitments must follow:

1. they cannot be used to infer the action;
2. they should be easy to compare whether an action corresponds with a commitment.

```
LOCKSTEP(a_p, P)
  in:    action a_p of the local player p; set of remote players P
  out:   set of players' actions A
  local: commitment c; action a; set of commitments C
 1: c ← HASH(a_p)
 2: SEND-ALL(P, c)                    ▷ Announce commitment to other players.
 3: C ← {c}
 4: C ← C ∪ RECEIVE-ALL(P)            ▷ Get other players' commitments.
 5: SYNCHRONIZE(P)                    ▷ Wait until everyone is ready.
 6: SEND-ALL(P, a_p)                  ▷ Announce the action.
 7: A ← {a_p}
 8: A ← A ∪ RECEIVE-ALL(P)            ▷ Get other players' actions.
 9: for all a ∈ A do
10:     find c ∈ C for which sender(c) = sender(a)
11:     if c ≠ HASH(a) then          ▷ Are commitment and action different?
12:         error action a does not comply with commitment c
13:     end if
14: end for
15: return A
```

Although, current method is works good in turn-based games and players will not be able to notice anything, it is pretty bad in real-time games as the teleport/significant speed increasing effects occurs.

What can help in this situation is asynchronous lockstep protocol. The idea of it is very simple. The player is allowed to move asynchronously from different players, however, when the interaction is necessary they fall into the lock-step mode. This specific mode basically represents the sphere of influence of every player. In other words, until the players are not interacting their decisions will not have any effect on other players.

5 Protocol disadvantages

Although, the protocol is very powerful and nice constructed it has two main significant disadvantages that force developers not to use lockstep in their games:

a. Each client must run their simulation in synchronization with each other. That, in fact, means that every single simulation must update the exact same number of times and in the exact same order for each action. Otherwise, the client might get ahead or behind of the other players. Moreover, in that case when a new command is sent the path would be different for those players that are ahead/behind. All of these may cause

that every client would play a different game.

b. Another issue is determinism across different machines and platforms. In other words, even a small, tiny difference in calculations might be a reason for butterfly effects which lead to very different games.

6 Future

Although, the protocol was widely used, right now we can say that it is no longer that popular how it was several years ago. Nowadays, almost all games use the new standard - Client-Server network model. Basically, lockstep was widely used because of network bandwidth was very low. However, right now the situation is different and network bandwidth is no longer that bad. Nevertheless, the main reason why developers no longer want to work with the technology is that desynchronizations are very common when using lockstep and every little issue may lead to game breaking desynchronization.

Below, you can find a very short list of popular games that used the protocol and what is used right now by the new versions of these games:

StarCraft 1	**StarCraft 2**
Peer-to-peer lockstep	Client-Server
DOTA(Warcraft 3)	**DOTA 2**
Lockstep	Client-Server
DOOM	**Quake**
Lockstep	Client-Server
Supreme Commander 1	
Lockstep	

7 Conclusion

The lockstep protocol was the best technique to prevent the look ahead cheating, although it is not the best solution in all cases. Game types like shooters or other games where action occurs in real time and players are forced to interact with each other all the time would work horrible with the described protocol. Moreover, nowadays there are nicer techniques without that strong disadvantages.

8 References

1 Online game protocol for P2P using Byzantine agreement, http://ieeexplore.ieee.org/document/5273978/authors?part=1

2 Yeung S. F., Lui J. C., Lui J. and J. Yan. Detecting Cheaters for Multiplayer Games: Theory, Design and Implementatio, http://www.cse.cuhk.edu.hk/~cslui/PUBLICATION/detect_cheat.pdf

3 Smed J. and H. Hakonen. Perverting Look-Ahead Cheating with Active Objects, 2006, https://rd.springer.com/chapter/10.1007/11674399_21 .

4 Herik H., Bjornsson Y. and N. Netanyahu. Computers and Games, 2003,

http://www.springer.com/gp/book/9783540324881

5 Lockstep implementation in Unity3D, http://clintonbrennan.com/2013/12/lockstep-implementation-in-unity3d/

6 Lock-step simulation is child's play Experience Report – Extended Version, https://arxiv.org/pdf/1705.09704.pdf

7 A short history of game networking techniques, February 24, 2010, https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/