游戏中的网络同步机制——Lockstep

2015-03-10 宾狗 工程 网络

0x00 前言

每个人或多或少都接触过网游，那个虚拟的世界给予了我们无穷的乐趣，而这个虚拟世界是如何完美的将身处天南地北的玩家连接在一起的呢？我们每个人的电脑配置都不一样，网络延迟也不同，但是在玩FPS（第一人称射击）游戏时，战斗感受与真实世界并无二致，网游是如何做到这一点的呢？

本文将介绍和分析**早期**广泛在RTS（即时策略）游戏中应用的同步机制——Lockstep

RTS游戏有很多，比如我们都玩过的的`Warcraft III`（大家耳熟能详的`Dota`是它的一张地图）和`StarCraft`，还有EA的代表作命令与征服系列（Command & Conquer）等等，以及现在非常流行的`Dota2`和`LOL`

那么为什么要强调早期呢？因为`Dota2`和`LOL`等新兴的游戏使用的同步机制不再是传统的Lockstep了。严格来说，Warcraft和现在意义上的网游有很大区别，因为它所谓的网是局域网（LAN）。早期RTS游戏出现时互联网还没有现在那么普及，网速也很慢，更没有什么像样的网游，能够支持局域网对战已经很不错了。

有人可能会有疑问，我们平时经常在对战平台上和全国各地的人打`Dota`，你为什么说`Warcraft III`只支持局域网呢？这又是一个很有意思的话题，实际上，对战平台使用了虚拟局域网（VLAN）技术，通过进程注入，HOOK `WinSock`函数调用，将数据包发送到对战平台服务器上，由服务器分配虚拟IP，这里还能够进行天梯匹配等等，在随后的游戏过程中游戏数据包都是通过对战平台的服务器进行转发，但是这一切对`Warcraft III`进程本身来说是透明的，它依然感觉自己在一个局域网环境中。

0x01 为什么要有同步机制

一致性

在虚拟世界中，保证游戏的一致性是一个基本前提。什么是一致性？通俗的说就是虚拟世界中的事实，比如在一个FPS游戏中，大家的延迟都很高，A、B两个玩家同时发现了对方，并向对方射击，如果没有很好的同步机制，那么A的屏幕上显示B还没有开枪就被击杀，而B的屏幕上显示A还没有开枪就被击杀，这就出现了不一致的问题，那么这个游戏还怎么愉快的进行下去？

可以这么说，延迟是造成不一致问题的主要原因。如果延迟都为0（即A玩家作出行动的同时B玩家就能看到），那么也就不存在不一致的问题了，就像在真实世界中一样。而同步机制除了基本的通信作用外，最重要的任务就是解决不一致问题，即保证游戏的一致性。同步机制有许多种，根据游戏类型、技术条件甚至时代背景的不同，选择的同步机制也会不同。

分类

游戏的网络同步机制有很多，国外也有这方面的论文，抛开具体实现细节，总体来看可以分为下面几类

- Peer-to-Peer，在这类方法中，没有服务器，游戏参与者的身份是对等的，依靠参与游戏的玩家电脑自行解决同步问题的，最为典型的就是Lockstep
- Client-Server，在这类方法中，Server端是绝对的权威，所有计算基本在Server上完成。例如，在游戏中向前移动一步，要等待服务器确认“你向前移动了一步”之后，才可以在客户端上进行这个行为。（延迟较低的时候是察觉不到这个过程的，延迟高时会有明显的卡顿现象）
- Client-Side Prediction，严格来说这并不是一类方法，而是对第二类方法的改进。试想如果所有的操作都必须在得到服务器的确认，然后才在客户端上进行，在延迟较高时用户体验会非常的差。这时可以把常用一部分计算转移到客户端进行，服务器辅助校正即可。

0x02 什么是Lockstep

Lockstep最初是军队行进中使用的，后来在19世纪的时候广泛在美国监狱使用，成为那个时期美国监狱的一个标识。就像这样

或者这样

意思就是大家同步的走，谁超前了要等待，落后了的要赶上。后来就引申到游戏的网络同步机制上了

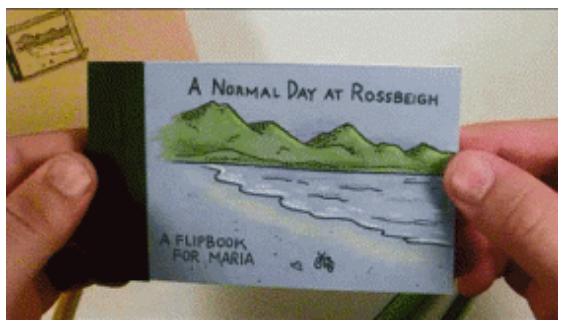
上一章节中我们说到Lockstep是Peer-to-Peer架构中的一种同步方式，而我们平时在局域网中玩`Dota`时，也的确没有大型的游戏服务器，只有一台所谓的主机，那么你可能会想，是不是所有的计算都是在那台主机上完成的呢？也就是说其他玩家的机器只发送`施放了某某技能`这样的数据包给主机，而`造成多少伤害、某某效果`是由主机计算并返回的。

但事实并不是这样的，玩`Dota`时**所有的一切都是在本地计算完成的**。包括技能伤害、效果，命中与否，随机刷新野怪等等。也就是说**每个玩家的电脑都完整计算了整盘游戏的全过程，且计算过程与计算结果都一模一样**。而主机只是负责把每个玩家的操作指令（鼠标点击、键盘按键等等）广播给其他玩家。

是不是感到难以理解？

想要理解Lockstep的机制，先看看下面三个问题。

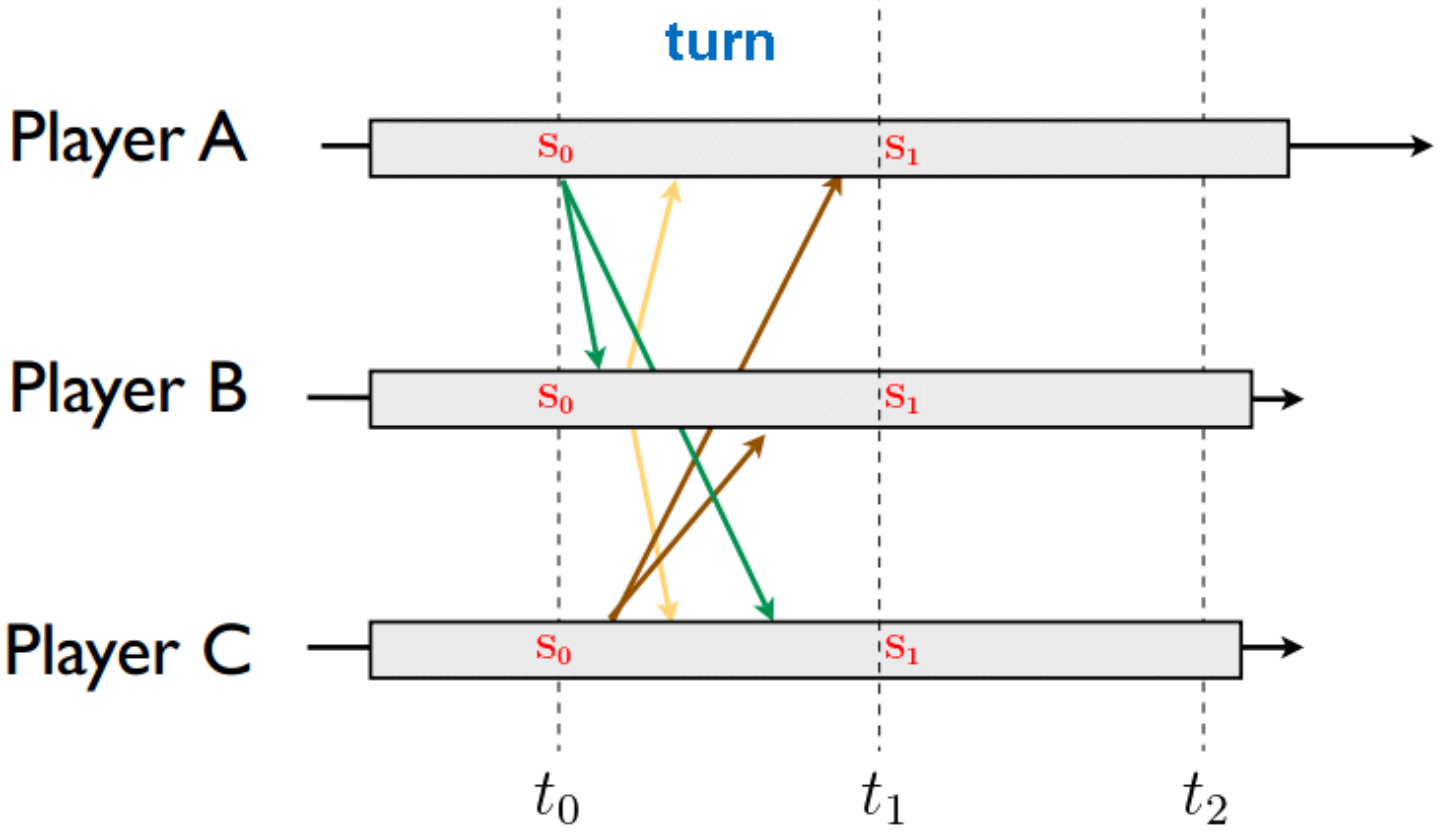
- 什么是动画？是会动的画吗？当然不是，人眼的记忆时间为`0.1s`，也就是大约`100ms`，只要把一帧一帧的静态图像快速播放一遍，我们就会感觉画面就动了起来，比如下面这样



- 最容易实现同步的游戏类型是什么？当然是回合制游戏，比如棋类游戏和卡牌游戏，它们有严格的先后顺序，不容易出现逻辑错误，更不会出现不一致的情况；而且回合时间较长，能够容忍高延迟。
- 什么是状态机？状态机是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型。给定一个状态机模型`F`，处在某一个状态`S1`，这时给定一个输入`I`，此时状态机会转移到一个新的状态`S2`。在这个过程中，只要`F`、`S1`和`I`是确定的，那么`S2`就是确定的。

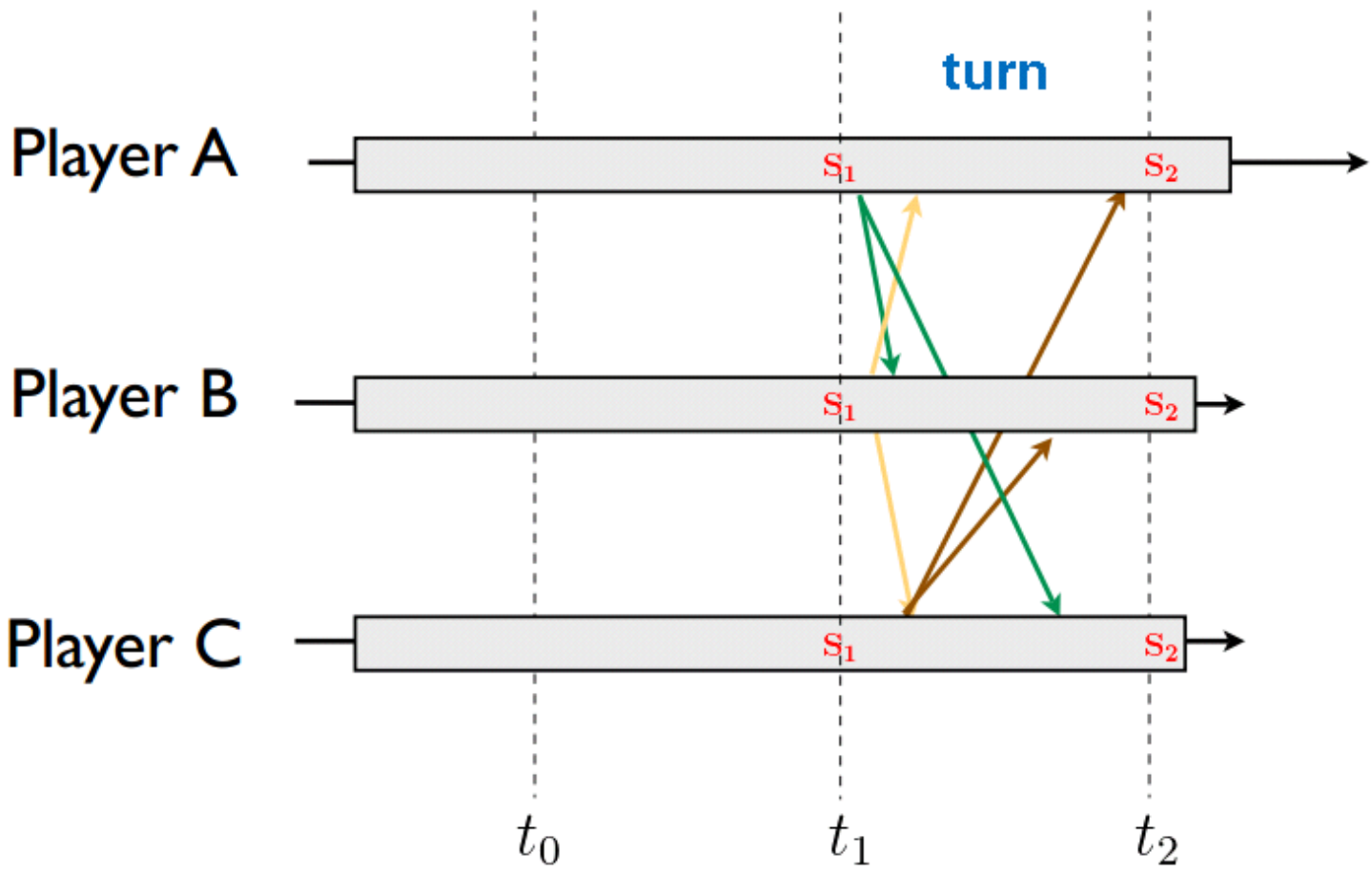
其实把以上三点结合起来，就是Lockstep的基本思路

我们来看下面这张图



图中是A、B、C三个玩家的时间轴，这个时间轴不是电脑上的本地时间，而是A、B、C联机时定义的一个时间轴。虚线分隔出来时间片称为turn，可以理解成一个回合。箭头表示该玩家将自己的操作指令广播给其他玩家。我们把一盘游戏看成一个大型的状态机，因为大家玩的是同一款的游戏，因此`F`是相同的，初始状态`S0`也是相同的。在第一个turn结束时，所有玩家都接收到了完全一样的输入`I`，注意这里的`I`不是一个值，而是包含了当前游戏中所有玩家的操作指令集合。`t1`时刻所有玩家的电脑自行计算结果。由于`F`、`S0`和`I`是固定的，所以每个玩家电脑上计算出的下一个状态`S1`一定是相同的。

同理，第二个turn也是如此



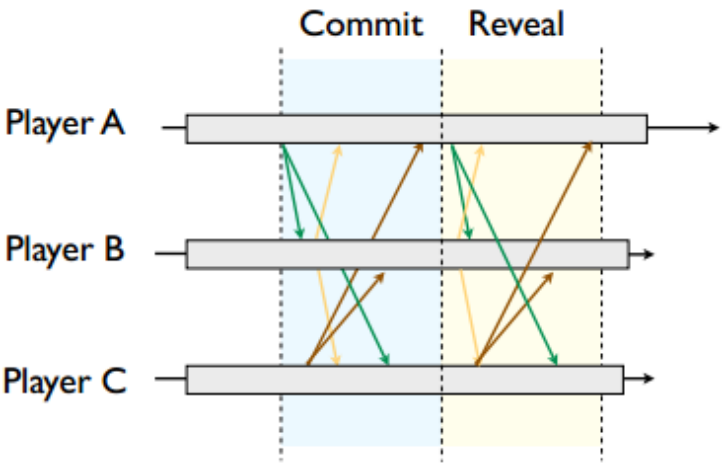
可以看出，Lockstep其实也是“回合制”的，当然这个所谓的回合与我们理解的棋类、卡牌游戏的回合是不太一样的。Lockstep的回合（也就是turn）中，所有玩家都可以采取行动，最终结果是在回合结束时统一计算的。在同一个turn接收到的操作指令，是不分行动先后顺序的，只要是在同一个turn里，就认为是同时发生的。

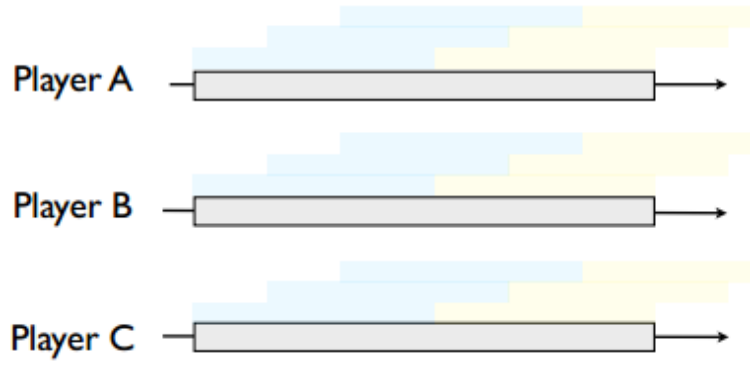
举个例子，假设A、B、C是游戏中3个互相敌对的单位，攻击力都为100。在某一个turn内，A和B都右键点击了C（warcraft这类游戏好像都是右键普攻），C右键点击了A，这些操作指令都广播到了其他玩家电脑上，则该turn的输入为“A攻击C、B攻击C、C攻击A”。那么该turn结束后，每个人的电脑都开始计算，且计算结果是相同的，即“A损失100生命值，B不变，C损失200生命值”。

这就是Lockstep同步机制，其实也没有多复杂是吧~这里还有几点需要注意：

- Lockstep把游戏过程划分成了一个个turn，为什么游戏不会出现卡顿的现象呢？回到前面动画的那个问题，人眼是容易被欺骗的，人的反应其实也是很慢的（相对电脑来说）。人眼的记忆时间为`0.1s`，只要每秒进行`10`turn，我们是感觉不出卡顿的。而通常游戏的帧数为60fps，即每秒60幅图像在屏幕上显示，你会感觉游戏非常流畅~当然“帧”和Lockstep中的“turn”并不是一一对应的，这里只是想说明一个turn的时间是非常短的，至少比我们的反应要快的多。
- Lockstep对网络延迟的要求是非常高的，因为每个turn要向所有玩家广播操作，同时也要接收来自其他玩家的操作。只有当每个turn集齐了所有玩家的操作指令，也就是输入确定了之后，才可以进行计算，进入下一个turn，否则就要等待最慢的玩家。当然局域网可以很好的满足这个要求，延迟基本都在1ms左右。

- Lockstep中会不会出现延迟导致的不一致问题？显然不会，从上面的分析可以知道，使用Lockstep的游戏是严格按照turn向前推进的，如果有人延迟比较高，其他玩家必须等待该玩家跟上之后再继续计算，不存在某个玩家领先或落后其他玩家若干个turn的情况。**使用Lockstep同步机制的游戏中，每个玩家的延迟都等于延迟最高的那个人。**（当然这个说法还有待讨论，后面还会说到这个问题）
- Lockstep是非常严格的，要求每一步的的计算结果都完全一样，任何计算错误都有可能导致蝴蝶效应，产生严重的后果，因为状态不能同步的话游戏根本就没有办法进行下去，最终将崩溃退出。
- 《Algorithms and Networking for Computer Games》这本书中关于Lockstep的部分与本文的说法有些不同，我觉得也没有谁对谁错之分，Lockstep更多的是一种思想，算是不一样的理解吧，感兴趣的同学也可以看看书中的章节。书中将Lockstep分为commit和reveal两个阶段，并且采用了流水线的方式。如下图所示





0x03 与Lockstep无关的细节问题

为什么要讨论与Lockstep无关的问题呢？因为这些问题虽然与Lockstep无关，但却与游戏本身有关，而且很多问题是由Lockstep引起的

野怪刷新与暴击

我们都知道，`Dota`中有许多问题是与概率相关的，比如整点时野怪是随机刷新的，出了水晶剑之后是有概率暴击的。那么按照Lockstep同步机制，计算都是在每个玩家自己电脑上完成的，那么在有概率存在的情况下，怎么可能保证每台电脑的计算结果一致呢？！

这时就轮到伪随机数派上用场了，我们来看下面这个`Java`程序

```
import java.util.Random;
public class PseudoRandom {
    public static void main(String[] args) {
        Random r1 = new Random(10);//这里的10就是随机种子（Random Seed）
        for (int i = 0; i < 10; i++) {
            System.out.print(r1.nextInt(100) + "\t");
        }
        System.out.println();
        Random r2 = new Random(10);
        for (int i = 0; i < 10; i++) {
            System.out.print(r2.nextInt(100) + "\t");
        }
    }
}

//Output
//13    80      93      90      46      56      97      88      81      14
//13    80      93      90      46      56      97      88      81      14
```

如果你的`jdk`版本没有什么问题的话，那么你看到的结果一定也是上面那样，而且无论你运行多少次都是这个结果。大部分编程语言内置库里的随机数都是利用线性同余发生器产生的，如果不指定随机种子（Random Seed），默认以当前系统时间戳作为随机种子。一旦指定了随机种子，那么产生的随机数序列就是确定的。

所以，游戏开始前，参与游戏的玩家电脑协商确定一个随机种子，就可以保证在游戏进行过程中大家产生的随机数序列是相同的，也就可以保证计算结果一致。

例如一个英雄的暴击率为`30%`，对某个目标持续普攻，如果随机数序列为`12 32 90 25`，小于等于`30`判定暴击，大于`30`判定不暴击，那么每个玩家电脑的计算结果都是`暴击 不暴击 不暴击 暴击`。（这里只是举例说明原理，游戏中的实现细节不一定是这样）

外挂问题

在对战平台打`Dota`的人都见识过全图挂，而且屡禁不止，那么为什么`Warcraft III`中会有全图挂，而没有其他游戏中的变态挂（如加强攻击力，瞬间移动等等）呢？为什么无法从根本上杜绝全图挂呢？

前面已经说过了，使用Lockstep同步机制，所有计算都是在本地完成的，每轮turn都必须接收来自其他所有玩家的操作指令才能完成计算，也就是说其他玩家的一举一动你的电脑其实是知道的，只不过没有在你的屏幕上展现出来罢了（被战争迷雾遮挡了）。因此，全图挂只要修改`Warcraft III`的内存数据，就可以去除战争迷雾，达到开全图的效果。而对战平台不可能改变Lockstep这种同步机制，只能在本地检测是否有其他程序修改`Warcraft III`的内存数据（就像病毒查杀一样），而外挂程序总有办法绕过检测，所以全图挂总是层出不穷。

而另一方面，恰恰因为Lockstep同步机制，其他比较变态的外挂根本不可能在`Warcraft III`上存在。比如强化攻击力，我们同样可以利用修改内存的方式将增加自己的攻击力，可以直接秒杀其他任何单位，但是别忘了其他玩家电脑也在同步的进行计算，而我们的攻击力在其他玩家电脑上仍然是不变的，这就造成了状态不一致。前面说过了，Lockstep中出现状态不同步的情况时很容易产生蝴蝶效应，最终崩溃退出。

断线重连

这个问题一直被广大玩家诟病，因为其他网游从来没有断线之后连不回去的情况，为什么`Warcraft III`不支持断线重连呢？

Lockstep同步机制是非常严格的，中途加入游戏是从技术上来讲是非常困难的。首先中途加入的玩家要进行状态同步，而状态同步本身包含的内容太多了，其中包括时间戳、伪随机数序列、所有单位的位置属性信息等等，不是简单的复制粘贴就能同步的。这说起来容易，要具体实现起来没那么简单，而且在局域网中掉线情况并不常见，因此早期暴雪的开发人员可能也就忽略这个问题了。

事实上使用对战平台的人才会经常遇到这个问题，在互联网中，延迟高、掉线的情况时有发生。那么11平台的掉线重连功能是怎么来的呢？个人观点，那并不是真的掉线重连，只是我们的丢包情况比较严重，暂时卡了而已。11平台可能做了一些优化，将其他玩家的操作指令保存起来，等延迟稳定的时候再发送给我们。但由于此时我们电脑所处的turn可能落后了其他玩家，所以此时游戏就会像快放一样赶上其他玩家的进度。如果是真的掉线，如停电、程序崩溃等直接退出的情况，我们是无法重新加入游戏的。

Warcraft III是不是使用严格的Lockstep机制？

接上一个问题，如果`Warcraft III`是严格的Lockstep同步机制，那么一定会出现一人卡、大家都卡的情况，而事实上只有当我们是主机时才会出现这种情况，其他情况下我们延迟高甚至掉线都不影响其他玩家的操作。

因此，Lockstep实际是一种理想的模型，如果在实际中使用会造成非常差的用户体验。那么`Warcraft III`使用的到底是什么同步机制呢？参考What every programmer needs to know about game networking这篇文章后面一个评论的说法

> TOADCOP
>
> FEBRUARY 10, 2010 AT 9:15 AM
>
> btw afaik StarCraft don’t use peer-to-peer it uses client-server model with lockstep (at least warcraft 3 does so). It has the advantage what theoreticaly laggers will not affect gameplay/response latency at all (but to not let them fall behind server do timeouts so the lagger can catch up, also doing temporary local game speed increasing) and imo it’s the only and true way to do sync in RTS like games. (and for some reasons this technic isn’t good covered in the web)

然后是作者的回复

> GLENN FIEDLER

> FEBRUARY 10, 2010 AT 10:20 AM
>
> You are correct. Also something cool is that in a C/S RTS model the server could also theoretically arbitrate to ignore turns from lagging players, and kick them if they don't catch up – removing various exploits where you can time-shift your packets and lag out other players.

也就是说`Warcraft III`使用的是基于Client-Server的Lockstep模型。这就是为什么`Warcraft III`中有主机这个概念，当然这里主机的作用并不是完成所有计算。

（以下为个人观点）

`Warcraft III`中的主机的主要功能是广播并设置`timeout`，也就是说在每个turn内，游戏玩家并非直接将自己的操作指令广播给其他玩家，而是先发送给主机，由主机负责广播，且每个turn都有`timeout`，如果超过了`timeout`仍然没有收到某个掉线玩家的操作指令，则忽略该玩家在该turn的行为，即认定他什么都没有做，并与其他延迟正常的玩家同步进入下一个turn。而当掉线玩家网络恢复时，主机会将之前保存的turn中操作指令集合发送给该名玩家，而该名玩家为了赶上进度，就会出现游戏快放的情况。

所以`Warcraft III`中只有在主机延迟高或掉线时，其他玩家才会受影响，否则不受影响。在局域网中，如果主机是正常退出的，那么会选定另一玩家电脑作为主机，如果是崩溃退出的，则所有人都会直接掉线。至于在对战平台上是否有优化就不太清楚了。

0x04 总结

Lockstep是出现较早的一种同步机制，不过现在很多RTS游戏中依然能够看到它的影子，当然都对它进行了一定程度的改进。

国内关于游戏编程和网络同步的教材、文献寥寥无几，不知道是文化因素还是什么其他原因，难道与游戏相关的技术都是玩物丧志、不学无术？

在我查询资料的过程中，发现国外不仅有游戏编程和网络同步的理论教材，还有大学开设的游戏课程，甚至还有硕士论文是关于设计一个MMORPG游戏的……看看国外繁荣的游戏（使命召唤、刺客信条、魔兽世界），再看看国内繁荣的游戏市场（页游？手游？），看看国外知名的游戏公司（Blizzard、EA、UBISOFT），再看看国内知名的游戏公司（……企鹅？）

这里不是想黑什么，而是觉得计算机作为一门多样化的学科，我们不能固步自封，排斥其中的一些东西。有人说研究游戏有什么用？真的没用吗？看看下面这些应用

- 网络游戏中广泛应用的航位推测法（Dead Reckoning）美国军方也在使用
- 游戏中的自动寻路算法，在机器人和自动驾驶中也有应用
- 游戏中的决策与博弈论也有关联
- 游戏服务器集群之间的负载均衡与一致性，和现在热门的大数据息息相关

虽然不一定是由游戏本身推动的，但是这些研究之间是相辅相成的，不会做无用功的，何况游戏本身就是增加GDP的一个好手段ヽ(￣▽￣)ﾉ

0x05 相关资料

- 教材：《Algorithms and Networking for Computer Games》
- 课程：CS4344: Networked and Mobile Games
- 硕士论文：Hybrid Peer-to-Peer Solution forMMORPGs