\*帧同步，状态同步

https://zhuanlan.zhihu.com/p/36884005

最大的区别就是战斗核心逻辑写在哪，状态同步的战斗逻辑在服务端，帧同步的战斗逻辑在客户端。战斗逻辑是包括技能逻辑、普攻、属性、伤害、移动、AI、检测、碰撞等等的一系列内容，这常常也被视为游戏开发过程中最难的部分。由于核心逻辑必须知道一个场景中的所有实体情况，所以MMO游戏（例如魔兽世界）就必须把战斗逻辑写在服务端，所以MMO游戏必须是状态同步的，因为MMO游戏的客户端承载有限，并不能把整张地图的实体全部展现出来（例如100米以外的NPC和玩家就不显示了），所以客户端没有足够的信息计算全图的人的所有行为。

\*玩家移动

有个解决方案，我给它取名叫预测拉扯，虽然有些怪异了点，不过基本上大家也能从字面上来理解它的意思。要解决这个问题，首先要定义一个值叫：预测误差。然后需要在服务器端每个玩家连接的类里面加一项属性，叫TimeModified，然后在玩家登陆的时候，对客户端的时间和服务器的时间进行比较，得出来的差值保存在TimeModified里面。还是上面的那个例子，服务器广播消息的时候，就根据要广播对象的TimeModified，计算出一个客户端的CurrentTime，然后在消息头里面包含这个CurrentTime，然后再进行广播。并且同时在玩家A的客户端本地建立一个队列，保存该条消息，只到获得服务器验证就从未被验证的消息队列里面将该消息删除，如果验证失败，则会被拉扯回P1点。然后当玩家B收到了服务器发过来的消息“玩 家A从P1到P2”这个时候就检查消息里面服务器发出的时间和本地时间做比较，如果大于定义的预测误差，就算出在T2这个时间，玩家A的屏幕上走到的地点 P3，然后把玩家B屏幕上的玩家A直接拉扯到P3，再继续走下去，这样就能保证同步。更进一步，为了保证客户端运行起来更加smooth，我并不推荐直接 把玩家拉扯过去，而是算出P3偏后的一点P4，然后用(P4-P1)/T(P4-P3)来算出一个很快的速度S，然后让玩家A用速度S快速移动到P4，这 样的处理方法是比较合理的，这种解决方案的原形在国际上被称为（Full plesiochronous），当然，该原形被我篡改了很多来适应网络游戏的同步，所以而变成所谓的：预测拉扯。