模式：

DS-Client

内容包含：

1.一场游戏的创建过程（GameMode、GameState）

2.PlayerController的初始化过程

3.Character的初始化过程

4.客户端与服务器的连接建立过程

5.客户端加入游戏的过程

6.如何进行游戏模式的规则的开发

7.游戏流程是如何逐阶段推进的

**1、一场游戏的创建过程（GameMode、GameState）**

UE4中存在一个GameInstance游戏实例，自游戏开始时由Engine实例创建，GameInstance生存周期一直存在直到游戏shut down，**客户端和服务器都会有一个GameInstance**。GameInstance会读取默认的关卡，每一个关卡对应一个WorldSetting，UWorld一般会加载PersistentLevel的WorldSetting，调用UWorld::SetGameMode通知GameIntance创建GameMode，GameInstance调用UWorld::SpawnActor在世界中创建GameMode。客户端不创建GameMode。

在服务器中，GameState的创建在GameMode::PreInitializeComponents()发生，也就是说随着GameMode而产生。然而在客户端上，建立好连接后（PlayerContoller）后，GameState由服务器复制而来。

**2、PlayerController的初始化过程**

World在UWorld::LoadMap本地加载地图的时候，都会调用ULocalPlayer::SpawnPlayerActor**创建一个本地APlayerContoller**。

服务器同意客户端连接后，客户端发出NMT\_Join消息，这个时候客户端本地已经执行了加载地图。服务器收到NMT\_Join消息解析完毕后，会调用UWolrd::SpawnPlayerActor来生成一个PlayerContoller在服务器端上，**有多少个客户端连接，就会产生多少个PlayerContoller**。

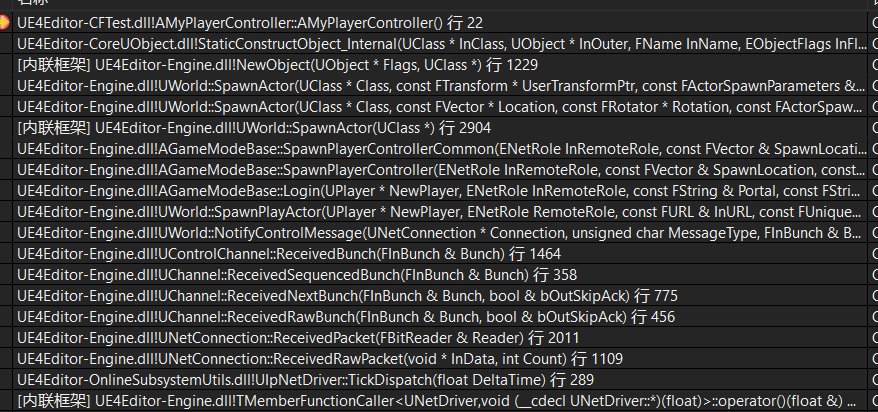


图 服务器收到客户端join信息

服务器产生了PlayerContoller，建立了与客户端的连接后，会把这个**PlayerContoller信息同步给客户端**，客户端本地调用构造函数后，再复制服务端PlayerContoller的消息，这样就得到了服务器PlayerContoller的复制版本。

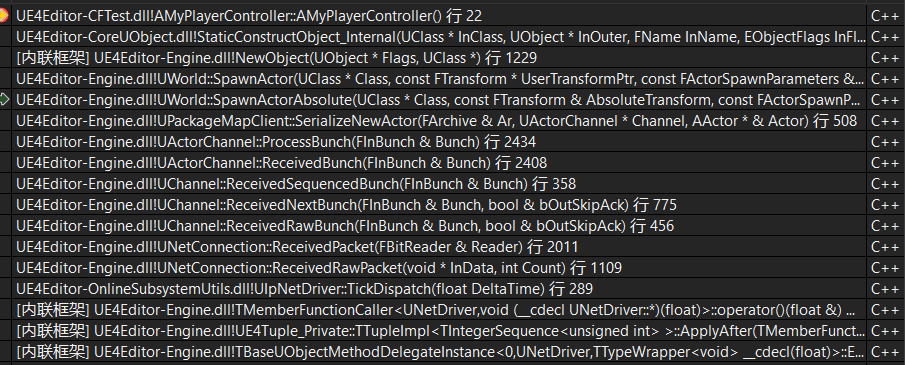


图 客户端收到服务器同步消息

**3、Character的初始化过程**

如果没有指定默认的Pawn，那么何时何地产生人物角色由用户决定，在世界产生想要的Charater并用PlayerController 持有后，会把Character和Contoller信息同步至客户端。

如果指定了默认的Pawn，在服务器上，得到出生点后，GameMode会调用AGameModeBase::SpawnDefaultPawnAtTransform，将人物产生到固定的地方，并且持有，然后同步至客户端。

**4、客户端与服务器的连接建立过程**

首先，在服务器和客户端上，GameInstance都需要初始化通信相关的内容。在服务器上，GameInstance创建一个IPNetDriver，创建一个Socket绑定端口后等待客户端的连接。在客户端上，因为world暂时没有创建，会额外创建一个UpendingGame来接受和处理连接信息。

客户端会创建一个**ServerConnection**，继而创建一个Control Channel来发送控制信息。第一次会**发NMT\_Hello给服务器**。

服务器收到Hello信息解析完毕，创建**ClientConnection，**如果验证成功，会**发送NMT\_Chanllenge给客户端。**

客户端收到challenge消息解析完毕，**发送NMT\_Login给服务器。**

服务器收到Login解析完毕，会执行GameMode::PreLogin，**发送NMT\_Welcome给客户端，同时发送地图信息。**

客户端收到welcome信息，会执行地图加载，**发送NMT\_Join消息给服务器**。

服务器收到join消息，会执行GameMode::Login产生PlayerContoller和注册玩家，执行GameMode::PostLogin。

最后**将Actor和PlayerContoller同步至客户端。**

5、客户端加入游戏的过程

GameMode创建PlayerContoller并且同步至客户端后，客户端也获得了这个PlayerContoller，客户端和服务器都可以通过PlayerContoller找到客户端与服务器之间的连接，进而执行游戏逻辑。

**6、如何进行游戏模式的规则的开发**

GameMode主要用来定义游戏的规则，负责游戏中非常权威的、重要相关逻辑计算和执行，那些全局一致的逻辑需要写进仅存在于服务器的GameMode，比如游戏什么时候开始，什么时候该结束了，爆炸的发生、天气的转变，都需要在服务器执行，通知到客户端。GameState存在于客户端和服务之中，可以当做服务器和客户端沟通的媒介，把需要让所有客户端知道的状态集放入其中。PlayerContoller也可以充当客户端和服务器沟通的媒介，也可以把要同步的一些状态直接写入其中，能够达到和GameState一样的效果。对于每个客户端，都会保存其他玩家的PlayerState，所以可以把一些玩家需要公开给其他玩家的状态放入其中（也可以放入GameState中）。

**7、游戏流程是如何逐阶段推进的**

首先，无论是客户端还是服务器，需要有UEngine、GameInstance，用这些来初始化建立连接所必要的属性。

游戏要进行前，必须有段时间，等待所有当前要参与游戏的玩家连接到DS，这个流程结束后，基本得到了所有玩家的连接，所有玩家有了同步过来的PlayerContoller。

连接建立后，服务器需要把Actor等必要的信息同步至客户端，客户端得到Actor的模拟或自治版本。

开始游戏所必要的内容初始完毕后，客户端和服务器开始执行相关的逻辑，通过他们所建立的连接执行RPC、属性同步来通信。

除有些玩家提前结束游戏外，会有结束游戏的规则，达到结束条件后，所有玩家游戏结束。

后续DS可以断开与玩家连接，玩家也可以主动断开连接。