深入剖析UE4网络

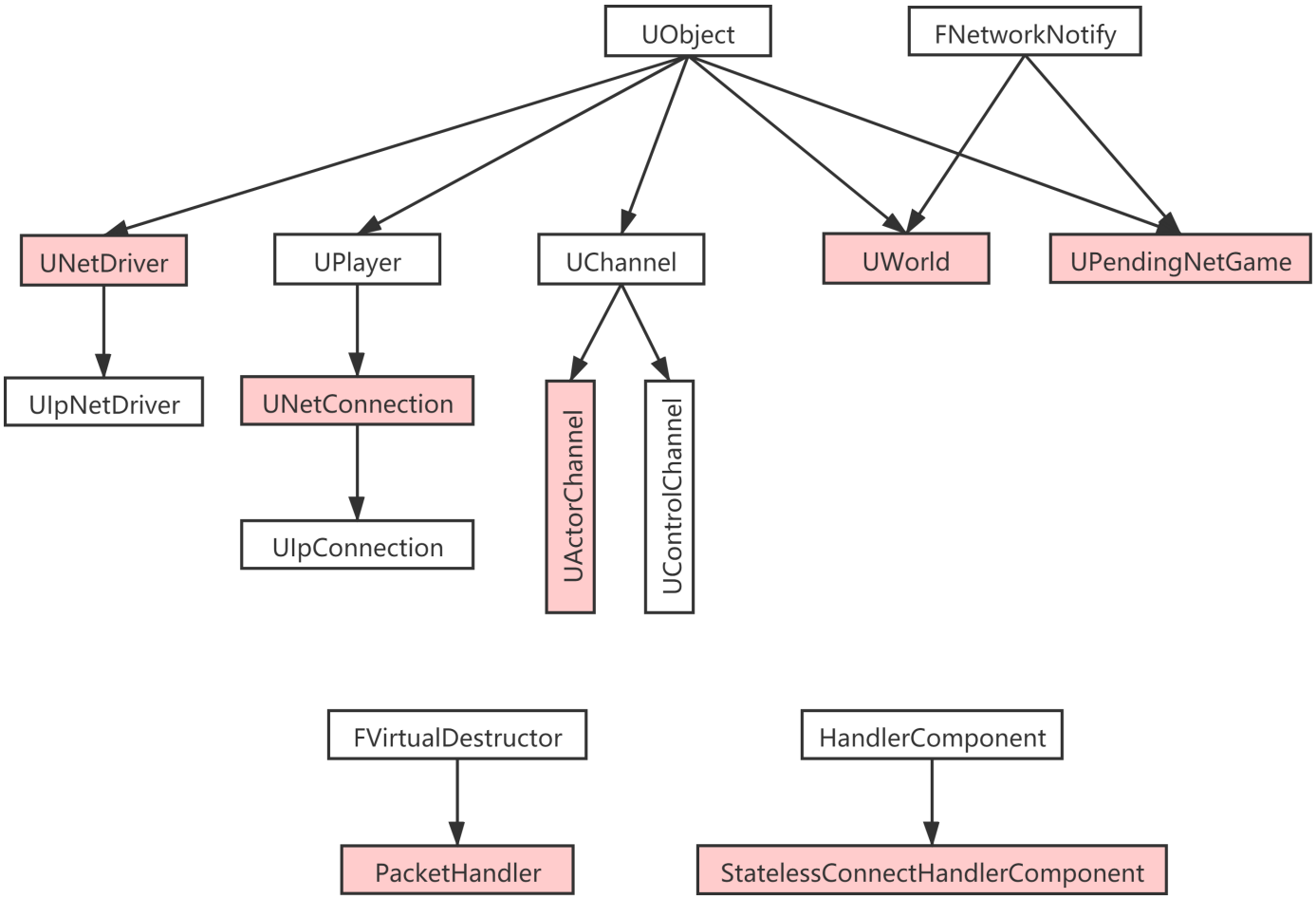
UE4的网络设计

1. UE状态同步的实现是基于UDP，并且以Actor为同步单位，与UE整体结构强耦合，从而分为多个层次保证可靠性的同时提高整体效率
2. ipNetDriver中RecvForm代码编码了MAX_PACKET_SIZE的接收缓冲大小为1024字节
3. UNetConnection::InitSentBuffer函数中设置了SendBuffer为1024字节，可在初始化Connection的时候传入
4. 通过准确计算将UDP数据严格限制1024字节内

架构特点

- 客户端和服务端共用一套代码
- 服务器为游戏逻辑核心，单个服务器为核心，多个客户端连接
- 默认通信协议为UDP(应用层实现数据可靠的UDP)
- 收发UDP数据包都在主线程(GameThread)执行

关键类



NetDriver

网络驱动，实际上我们创建使用的是他的子类IPNetDriver,里面封装了基本的同步Actor的操作，初始化客户端与服务器的连接，建立属性记录表，处理RPC函数，创建Socket。构建并管理当前Connect信息，接收数据包等基本操作。NetDriver和world一一对应，在一个游戏世界里只存在一个NetDriver，UE里面默认都是基于UDPSocket进行沟通的。

Connection

表示一个网络连接，服务器上，一个客户端到服务器的一个连接叫ClientConnect；在客户端上，一个服务器到一个客户端的连接叫ServerConnect；

LocalPlayer

本地玩家，一个客户端的窗口ViewportClient对应一个LocalPlayer，LocalPlayer在各个地图切换时不会改变

Channel

数据通道，每一个数据通道只负责交换某一个特定的数据信息

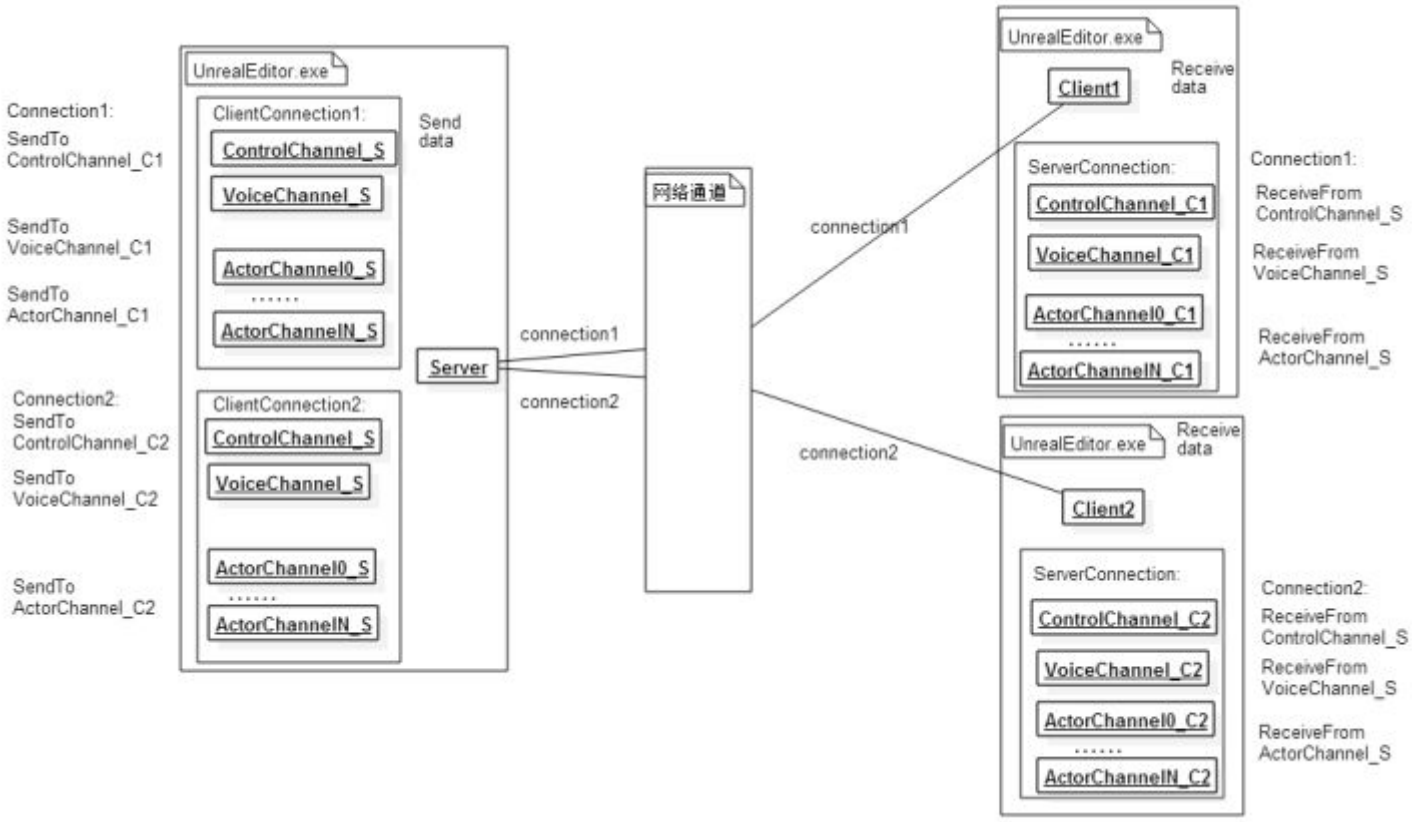
- ControlChannel
  客户端和服务器之间发送控制信息，主要是发送接受连接和断开的相关信息
- VoiceChannel
  用于发送接收语音信息
- ActorChannel
  处理Actor本身相关信息的同步，包括自身的同步以及子组件，属性的同步，RPC调用等

PlayerController

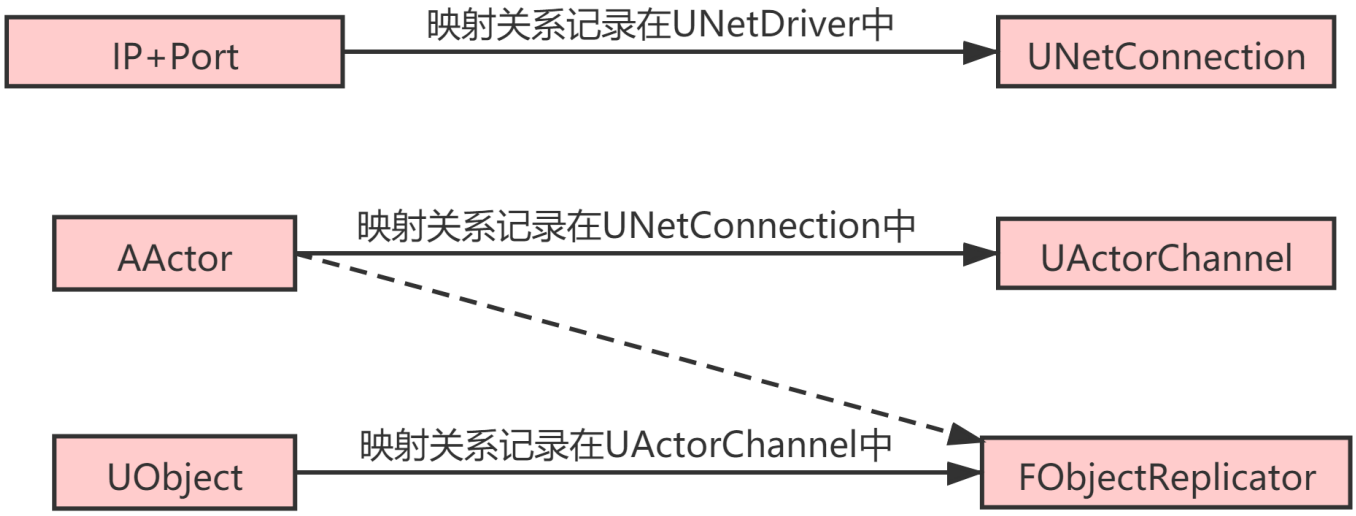
玩家控制器，对应一个LocalPlayer，代替本地玩家控制游戏角色，同时对应一个Connection，记录当前的连接信息，这和RPC条件属性复制都是密切相关的，

World

游戏世界，任何游戏逻辑都在world里面处理的，actor的同步也受world控制，world知道那些actor应该同步，保存了网络通道的基础设施NetDriver



Actor和Channel的关系



- 在服务器上
  1. NetDriver中管理着多个Connection，每个Connection对应着一个客户端IP+Port
  2. Connection中管理着多个ActorChannel,每个ActorChannel对应着一个Actor
  3. ActorChannel中管理多个FObjectReplicator,每个FObjectReplicator负者一个UObject的序列化和反序列化操作
- 在客户端上
  1. NetDriver中只有一个Connection，就是ServerConnection
  2. 一个Actor只对应一个ActorChannel，因为只有一个Connection

Actor承载的同步内容

- 针对一个Actor
  - Actor自己(SubObject同步的第一个Bunch的附带)
  - Properties,DeltaProperty,RPC
  - SubObjects
- 针对一个SubObject
  - SubObject自己(SubObject同步的第一个ContentBlock附带
  - Properties,DeltaProperty,RPC
  - Sub SubObjects