个人资料

房燕良

+ 加关注 发私信

博客专家

访问： 125183次
积分： 1743
等级： BLOG 4
排名： 第12304名

原创： 27篇 转载： 0篇
译文： 1篇 评论： 160条

原 《炉石传说》架构设计赏析(7)：使用Google.ProtocolBuffers处理网络消息

分类： Unity3D 2014-12-15 11:38 4592人阅读 评论(5) 收藏 举报

unity3d google protocolbuffe 网络 网络游戏

目录(?) [+]

这段时间琢磨了一下Unity3D网络游戏开发中的网络消息处理。网络游戏的服务端一般都是自主开发的，所以对应的网络消息处理也要自己开发。客户端/服务端之间的消息传到目前使用JSON和Google.ProtocolBuffers是两种常见的做法。打开炉石的代码看了看它的处理方式，感觉代码写的还是很好的，把它的思路分析一下，与大家分享。

整体机制描述

我们想要达到的目标大概是这样的：

- 有N个网络消息，每个消息对应一个Proto中的message描述；
- 每个消息对应一个数字ID；
- 底层在收到消息是，将其解析成为Google.ProtocolBuffers.IMessage对象，这个对象的具体类型应该是前面那个message生成的代码；
- 发送消息就简单了，[illegible]型，可以直接执行序列化；

炉石使用Google.Protocol[illegible]可以看这里：http://www.nuget.org/packages/Google.ProtocolBuffers/

消息发送

发送的机制很简单，首先[illegible]uffer生成的message类构造一个消息对象，例如：ConnectAPI.SendPing()


```csharp
public static void SendPing()
{
    Ping.Builder body = Ping.CreateBuilder();
    QueueGamePacket(0x73, body);
    s_lastGameServerPacketSentTime = DateTime.Now;
}
```

底层会构造一个“PegasusPacket”数据包对象，添加到发送队列之中，这个数据包对象主要包含3部分：消息ID，消息大小，具体消息数据。详见PegasusPacket.Encode()函数：

```csharp
public override byte[] Encode()
{
    if (!(this.Body is IMessageLite))
```

```
    {
        return null;
    }
    IMessageLite body = (IMessageLite) this.Body;
    this.Size = body.SerializedSize;
    byte[] destinationArray = new byte[8 + this.Size];
    Array.Copy(BitConverter.GetBytes(this.Type), 0, destinationArray, 0, 4);
    Array.Copy(BitConverter.GetBytes(this.Size), 0, destinationArray, 4, 4);
    body.WriteTo(CodedOutputStream.CreateInstance(destinationArray, 8, this.Size));
    return destinationArray;
}
```

消息接收与解析

接下来我们重点看一下消息的接收与解析机制。首先因为TCP是流式的，所以底层应该检测数据包头，并收集到一个完整的数据包，然后再发送到上层解析，这部分逻辑是在”ClientConnection<PacketType>.BytesReceived()“中实现的。当收到完整数据包时，会在主线程中触发”OnPacketCompleted“事件，实际上会调用到”ConnectAPI.PacketReceived()“，其内部主要是调用了”ConnectAPI.QueuePacketReceived()“，这个函数负责将TCP层接收到的byte[]解析成对应的IMessage对象。

重点来了！由于网络层发过来的数据包，只包含一个消息ID，那么客户端就需要解决从ID找到相应的消息Type的问题。想象中无非有两种方式去做：1是手动记录每个ID对应的Type；2是搞一个中间的对应关系的类，附加上自定义的Attribute，然后在使用反射机制自动收集这些类，其实和前者也差不多。炉石采用了第一种方式。整体机制是这样的：

- 客户端每个消息对应一个PacketDecoder的派生类对象；
- ConnectAPI类使用一个字典，用来保存<消息ID，Decoder对象>之间的对应关系：ConnectAPI.s_packetDecoders:SortedDictionary<Int32,ConnectAPI.PacketDecoder>；
- 如果每个消息都要写一个Decoder，而其内部代码由完全一致，岂不是很蛋疼？！好吧，我们用模板来实现，详见后续分析；
- 在ConnectAPI.ConnectInit()初始化的时候，创建Decoder对象，并保存到上述dict之中，类似这样：s_packetDecoders.Add(0x74, new DefaultProtobufPacketDecoder<Pong, Pong.Builder>());
- 最后在上述的收到完整数据包的函数中，根据数据包中记录的消息ID，去查找Decoder，然后调用其方法得到具体的消息对象，类似这样：

```csharp
if (s_packetDecoders.TryGetValue(packet.Type, out decoder))
{
    PegasusPacket item = decoder.HandlePacket(packet);
    if (item != null)
    {
        queue.Enqueue(item);
    }
}
else
{
    Debug.LogError("Could not find a packet decoder for a packet of type " + packet.Type
}
```

最后我们看一下，Decoder模板的实现技巧。首先消息解析的具体操作是有Google.ProtocolBuffers生成的代码去实现的，所以具体操作流程是完全一致的，这些写到基类的的静态模板函数中：

```csharp
public abstract class PacketDecoder
{
    // Methods
    public abstract PegasusPacket HandlePacket(PegasusPacket p);
    public static PegasusPacket HandleProtoBuf<TMessage, TBuilder>
(PegasusPacket p) where TMessage: IMessageLite<TMessage, TBuilder> where TBuilder: IBuilderLite
```

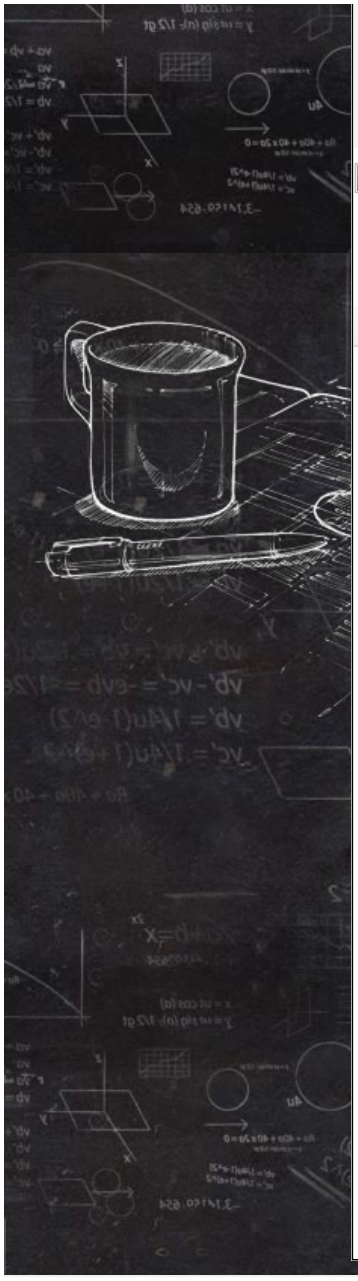
```
    {
        byte[] body = (byte[]) p.Body;
        TBuilder local2 = default(TBuilder);
        TBuilder local = (local2 == null) ? Activator.CreateInstance<TBuilder>
() : default(TBuilder);
        p.Body = local.MergeFrom(body).Build();
        return p;
    }
}
```

其次，使用一个模板派生类，实现HandlePacket()这个虚函数，主要的目的只是把TMessage和TBuilder这两个类型传给那个静态函数而已：

```
[csharp]
public class DefaultProtobufPacketDecoder<TMessage, TBuilder> : ConnectAPI.PacketDecoder where
{
    // Methods
    public override PegasusPacket HandlePacket(PegasusPacket p)
    {
        return ConnectAPI.PacketDecoder.HandleProtoBuf<TMessage, TBuilder>(p);
    }
}
```

OK，炉石是使用使用ProtocolBuffers来处理网络消息的机制就是这样，是不是已经很清晰啦！

上一篇 在Unity3D的Legacy动画系统中应用Root Motion
下一篇 Unity3D脚本中的Awake()和Start()的本质区别



查看评论

3楼 lirentai 2015-01-07 00:48发表

楼主uml图和各个类的描述都很清晰，我在架构上是小白，希望能在楼主讲解里面看到更多的对架构上的优劣分析：）

2楼 silveryw 2014-12-18 11:44发表

LZ分析项目的思路很老道，我想问那个类似UML图是用什么工具画的？我是做unity的新手，我这边也反编了代码在学习，但是遇到yield return部分，代码就会比较难分析。。有什么好的反编译C#的工具推荐吗？？

Re: 房燕良 2014-12-18 17:37发表

回复silveryw：UML图是用Microsoft Visio画的。反编译C#我用的Reflector，可以到我的QQ群里面来下载。

1楼 u010384128 2014-12-15 14:35发表

亲，给来个可以运行的源代码呗！

核心技术类目