

ET

deepwaterooo

February 20, 2023

Contents

1 一个网络上搜到的 ET 框架的应用	1
2 参考网络上的一个重构，重用复用大部分 ET 框架关于网络部分的源码，重构出一个不可热更新的服务器端	1
2.1 吐槽	1
2.2 抽离：我会用 ET7, 用最精简精炼的设计封装，ETTask 流式异步语法等	2
2.3 重构	2
2.4 模块化	3

1 一个网络上搜到的 ET 框架的应用

- 原仓库：https://gitee.com/NKG_admin/NKGMobaBasedOnET.git
- 相比于 ET 与斗地主框架，它很小很轻量，可以快速翻一遍。翻的过程中总是会有不少收获。
- 把 ET7 的框架，与这个小游戏的源码，Actor 消息，客户端位置消息的发送与服务器端的处理，转发等，以及异步协程锁等相关的逻辑连了一下

2 参考网络上的一个重构，重用复用大部分 ET 框架关于网络部分的源码，重构出一个不可热更新的服务器端

- 主要参考：<https://www.jianshu.com/p/2aaf4ab0682e> 其实现在他说的，可能很多我还仍然不是很懂。
- 小小服务器的起点：可以参考上面的重构，模块化转移，把可热更新的动态双端框架变成为自己的静态不可热更新最小带数据库的注册登录（客户端热更新资源包）
- 它的重构后的项目地址：<https://github.com/Bian-Sh/ET-Network-Module.git> 这不是把自己的服务器的工作量又压缩了很多了吗？【还没有运行，要试运行一下】

2.1 吐槽

- 当然，非要说细节上为啥抗拒，其实是有几点的，作为 ET 0 基础小白斗胆小声 BB 几句：
 - 为了热重载，严格约束用户按照 ET ECS 架构开发，因此开发一个应用得整好几个解决方案【12 个解决方案！太多了，尤其是现在系统受限制，完全不能用来开发项目】，而且还不是常规的打开方式。

- 不需要热更，ILRuntime 插件冗余了，不能忍，如果考虑热更那也只是对现有开发模式无侵入的 HCLR。【这里说的是实情，ILRuntime 代码侵入严重】
- 修改在 Model、Hotfix 程序集，需要编译才能看到修改的效果，而开启自动编译一度让我困惑。【对我来说，现在仍是困或的，我只能在 windows 系统下勉强可以运行示例项目】
- 框架文件结构限定的比较死，由于很多路径都是写死在逻辑中，原本仅有一个 Server，一个 Client，想要做一个 Server + 两个 Client 令初次接触 ET 的我无从下手（当然，现在整明白了【偶现在还木有涉及到这个问题】，改一些脚本中文件夹路径就行，按说 ET7 啥都内敛到了 Unity 内部，也不预设热更方案也就没有这些痛点了）。
- 我的服务器端可以有热更新方案，只是在我的客户端已经基本 90% 完成的情况下，我不太想再花大时间精力再严格按照 ET7 框架再次重构一次【仅只为了服务器端可以热更新?! NO】。因此，我只想要个让我感觉相对满意的不可热更新的服务器端。

2.2 抽离：我会用 ET7，用最精简精炼的设计封装，ETTask 流式异步语法等








- 更新 ET6.0 Github 最新提交后，一通复制粘贴，把网络部分梳理到新工程中【异步网络调用，这个模块也是自己服务器的重点】
- 转移 Protobuf、Litjson、ETTask、Timer、KCP 到新工程。【我仍然想要保留 Bson 内网交互消息序列化与反序列化。虽然是不可热更新的服务器，但是写这第一个服务器，也更多的是希望能够好好体会一下服务器端的写法，若存在多个服务器，注册登录用，网关服，map/location 服，要怎样处理?】
- 删除 Litjson、Protobuf 中为适配 ILRuntime 而撰写的逻辑。【暂时保留 ILRuntime，不太想要多余的 Litjson 是真的。等把这些做完，ILRuntime 还想要再深入一下】
- 此时，网络模块已经大体抽离，然后再根据报错，将缺失的、离散在周边的脚本汇集整理。
- 到了这一步，网络模块还是 ECS 架构的（由于模块无需热更，ECS 没有拆分在各个程序集）所以一通改造，把各个功能模块的 ECS 整合成单脚本（其实就是把 Component 的字段，System 中的 Awake、Update 等方法整到一起，使用 MonoBehaviour 驱动）。【这时候就明白，他们牛牛神把好好的脚本折分到不同组件系统是本事，偶们弱弱把不同组件里的机制还原回脚本里也是需要真正理解框架滴~~】
- ET EventSystem，ET 的心脏，用于在启动或热重载时根据 Attribute 做 Type 预绑定，同时也是一个事件总线。网络模块、ECS 以及事件分发都依赖它，一通删除，保留网络消息类型注册部分即可。【也就是说，它，ET 框架的心脏，切除精减到最小，只处理异步网络调用相关的回调封装逻辑】
- 断开对 ET Entity 的依赖：删除继承关系，同时将 GetChild、AddChild 进行简单的改写成实例查询和实例构建就好了。【感觉刷过题，对基本数据结构的掌握还不错，读起框架里对各种数据结构的再封装，方便框架使用时的书写，都是活龙活现，如有灵犀，想保持或封装更多底层结构，尽量减少框架上层的重复与不便利写法。知道回有无数样写法，谁不爱写最简单的呢?】
- 两个有意思的地方：LitJson 保留是为了网络消息实例以 json 输出，方便 log；KCP 最后被干掉了是因为 ET6.0 为了通用，即便是 KServer、KChannel 也默认不使用 KCP。【这两三种服务，再考虑一下，有没有必要都保留，还是说留用一种暂时?】

2.3 重构

- 管理器性质的组件都依赖了 MonoBehaviour 驱动，但他们多为管理器性质，所以笔者将他们改为静态类。【静态类，vs 单例模式】
- 对管理器性质的组件的初始化放在 ET Eventsystem 中统一按顺序进行（比如 opcode 管理器就要优先于其他消息分发管理器）。
- 对管理器性质的组件的 Update 统一放在 ET Eventsystem 中统一使用 PlayerLoop 驱动。
- 作为底层核心网络消息，将 Ping 消息 OuterMessage 中单独抽离出来，实现网络模块对 OuterMessage 零依赖。
- 重新设计非 RPC 网络消息的监听与取消监听，实现非 RPC 消息在继承了 MonoBehaviour 的类更方便的被捕获。【这个部分还要再想想】
- 代码生成：实现非 RPC 网络消息处理器的一键生成；修正 OuterMessage 的一键生成逻辑，跳过 Ping 消息。
- 引入 Loom 做线程间数据传递，剔除原有的带 Thread 关键字的组件，避免了不必要的依赖注入，使用 using static Loom; 的形式简化多线程间数据的投递。【这个没有接触过，还没有想明白，什么情境上下文下需要这个】

2.4 模块化

- 借助 Unity 官方 Assembly Definition File ，我们可以在 Unity 项目中轻松的模块化 ET 网络模块。
 - 首先，将依赖的插件抽离，放在 ThirdPart，为他们加上各自的.asmdef 文件，使其模块化 【以前不懂，应该很容易生成修改】
 - 然后，将用户自定义的消息类（OuterMessage.cs 和 OuterOpcode.cs）抽离，网络模块对其 0 依赖，方便随时剔除，随时生成，随时更新。【这些独立设计比较喜欢】
 - 网络模块重构成以下形式：

	com.network.editor.dll	2022/9/24 22:31	应用程序扩展	16 KB
	com.network.ettask.dll	2022/9/24 22:31	应用程序扩展	23 KB
	com.network.generated.dll	2022/9/24 22:31	应用程序扩展	18 KB
	com.network.runtime.dll	2022/9/24 22:31	应用程序扩展	66 KB
	com.network.timer.dll	2022/9/24 22:31	应用程序扩展	6 KB
	com.sharelib.litjson.dll	2022/9/24 22:30	应用程序扩展	59 KB
	com.sharelib.protobuf.dll	2022/9/24 22:30	应用程序扩展	181 KB

- 【去年的这一天，偶滴舅舅的生日，我在 WSU 看足球赛!!!】
- 可见：
 - ettask 、 timer 、 litjson、 protobuf 都成了独立的程序集。【不想要 Litjson, 保留 bson】
 - 网络核心又拆分出 editor + runtime + generated （generated 为一键生成的 outermesage 和 messagehandler）