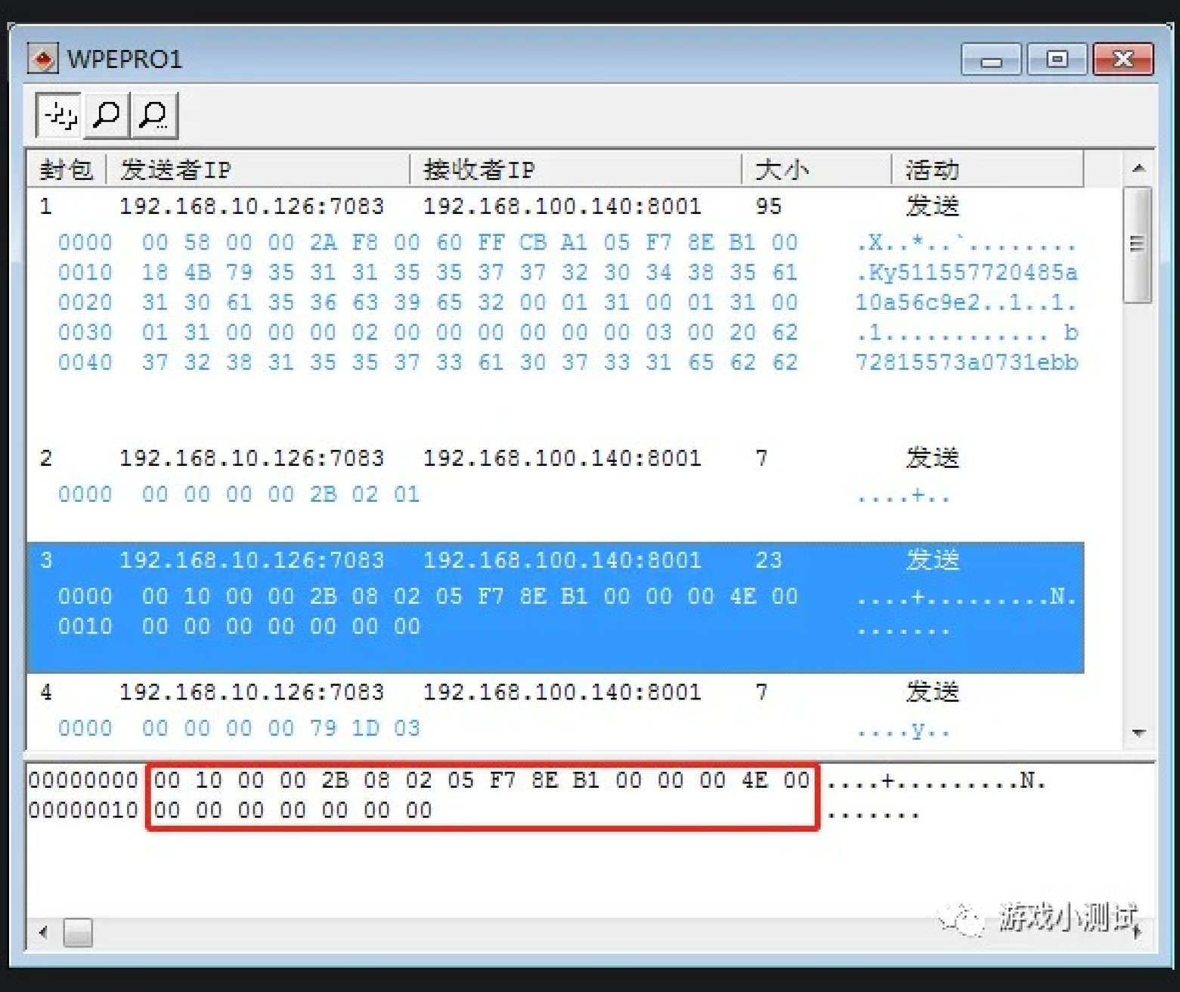
游戏抓包工具制作（零）—— 抓包工具的前世今生

原创 阿宽 游戏小测试 2024-04-08 19:07 广东

之前的文章讲到了协议测试，我们在将协议测试加入到测试必测内容之后，一直在思考一个问题，那就是如何更好更快更方便的进行协议测试。

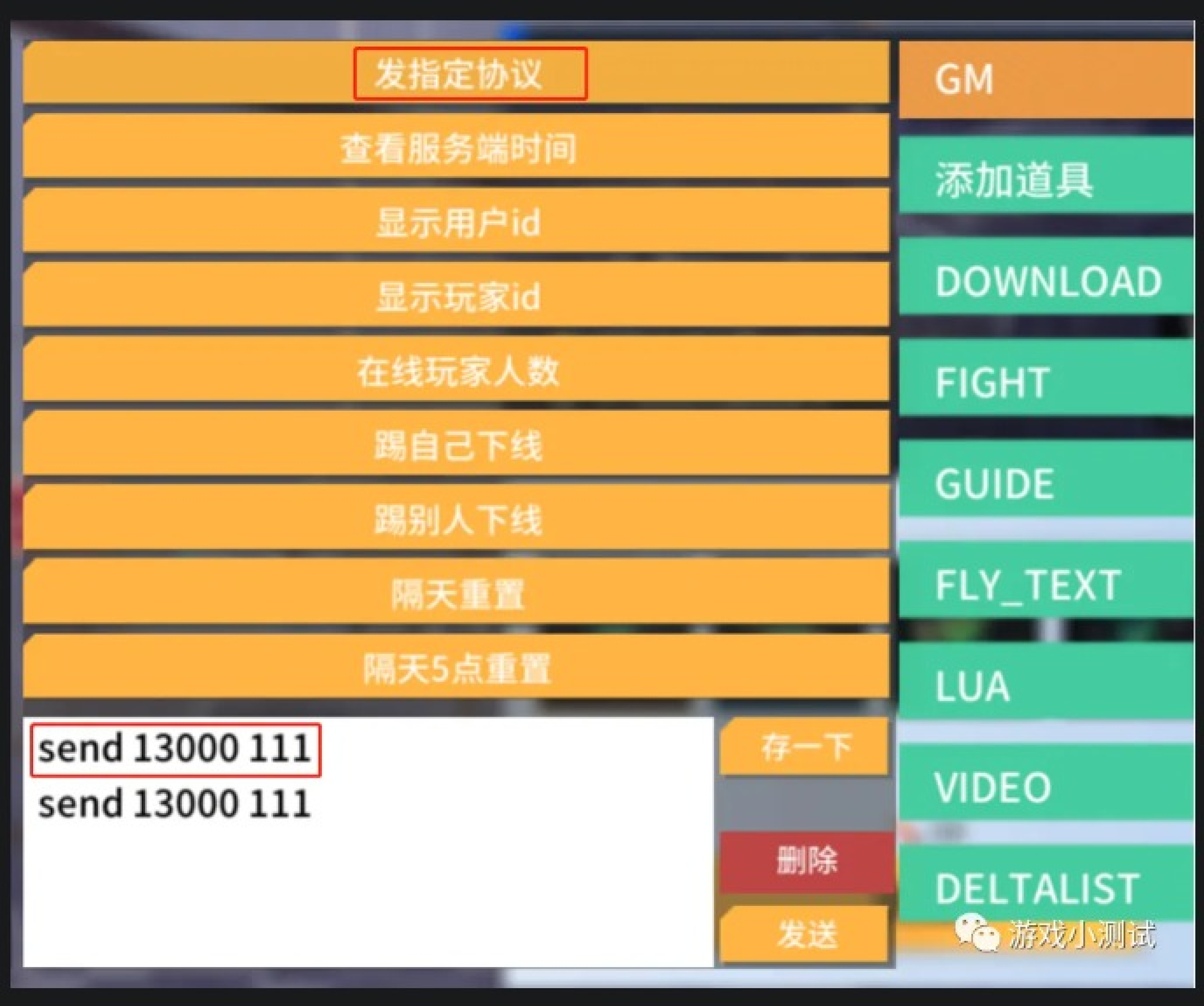
一开始我们使用的是WPE，曾几何时，WPE是一个网络游戏抓包神器，我们在做协议测试的时候，也是通过WPE来抓包、改包和发包，来测试服务器对于非法协议的处理逻辑是否正确。

WPE操作简单，只需要加载进程，启动抓包，然后停止抓包就能看到启动和停止期间所有与服务器之间的封包记录，但美中不足的是，WPE抓取的封包内容全部是16进制显示，有时候我们需要改动某个字段，还需要先破译并找到字段位置来做修改，可读性比较差，而且无法实时显示，无法清楚的知道服务器在我操作之后做了哪些响应。下图是WPE抓到的一些发送包：



另外一个就是程序同学给留的GM指令接口，在某些游戏无法通过WPE抓包的时候，我们会用GM指令来进行协议测试，它的优点是无需借用任何工具，无需考虑工具对游戏是否支持，无需考虑协议如何解析，只要在GM指令框输入对应的GM命令即可，比如图中的send 13000 111，这个命令就是发送13000协议，附带协议参数111，当然还有一些更复杂的命令，如带列表的，带对象的，它可能是这样的：send 13020 {cs_equip_strength,7,7,[{ly_item_num_msg_vo,527,1}]}。

我们刚才提到了GM命令进行协议测试的一些优势，但GM命令依然存在一些问题，比如需要输入的文字太多且可能会很复杂，而且跟WPE一样，它也无法看到实时的协议交互过程，只能判断结果是否存在异常，它也无法支持更复杂的测试辅助需求。



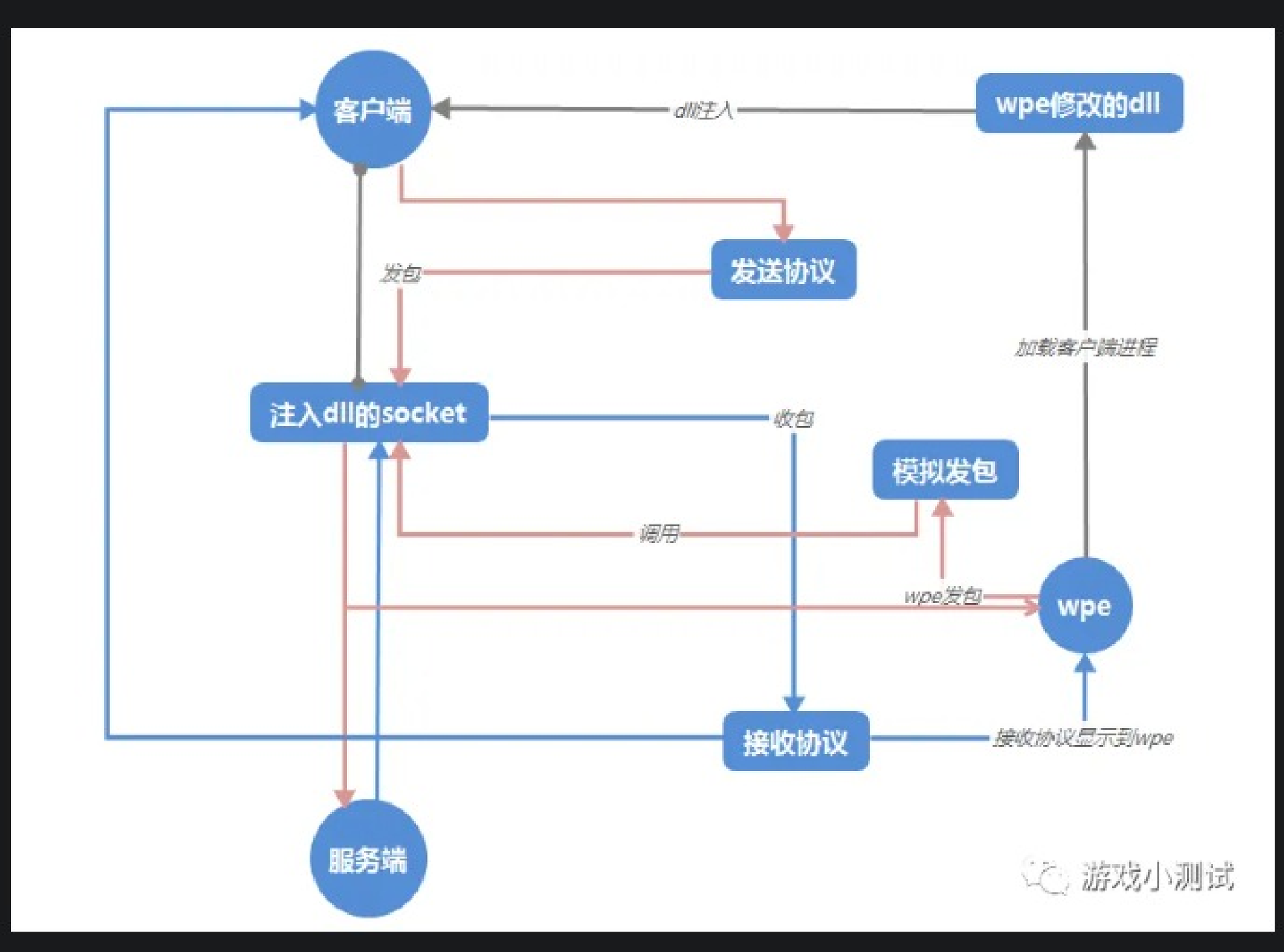
整理需求

当我们开始觉得在协议测试方面需要更多更便捷更丰富的支持的时候，我们决定自己写一个满足自己业务需求的抓包工具，这个工具的设计出发点首先是满足协议测试需求，然后能够支持我们做协议性能相关的分析，经过慢慢的功能完善和不断迭代，目前我们的抓包工具可以支持以下需求：

- **基本的协议测试**，可以通过多种方式修改或伪造协议并发送给服务器。
- **协议交互过程的实时显示**，可以实时并清楚的看到我每一步操作给服务器发送了什么协议和参数，服务器响应了我哪些协议。
- **协议统计图表**，将客户端与服务器之间的所有协议进行统计，统计内容包含发送次数（频率），包体长度以及每个协议号累积的协议长度并将这些数据转化成图表。
- **模拟弱网**，通过给协议转发加入延迟，来模拟弱网环境，观察客户端在弱网状态下的表现和逻辑。
- **快速重复的发包**，辅助进行一些重复性操作，解放双手。
- **定制测试脚本**，如概率测试等等，让抓包工具通过脚本进行指定的游戏行为，统计并输出概率计算结果，以便测试人员对概率进行分析。当然还有更多更丰富的脚本可以支持。

设计思路

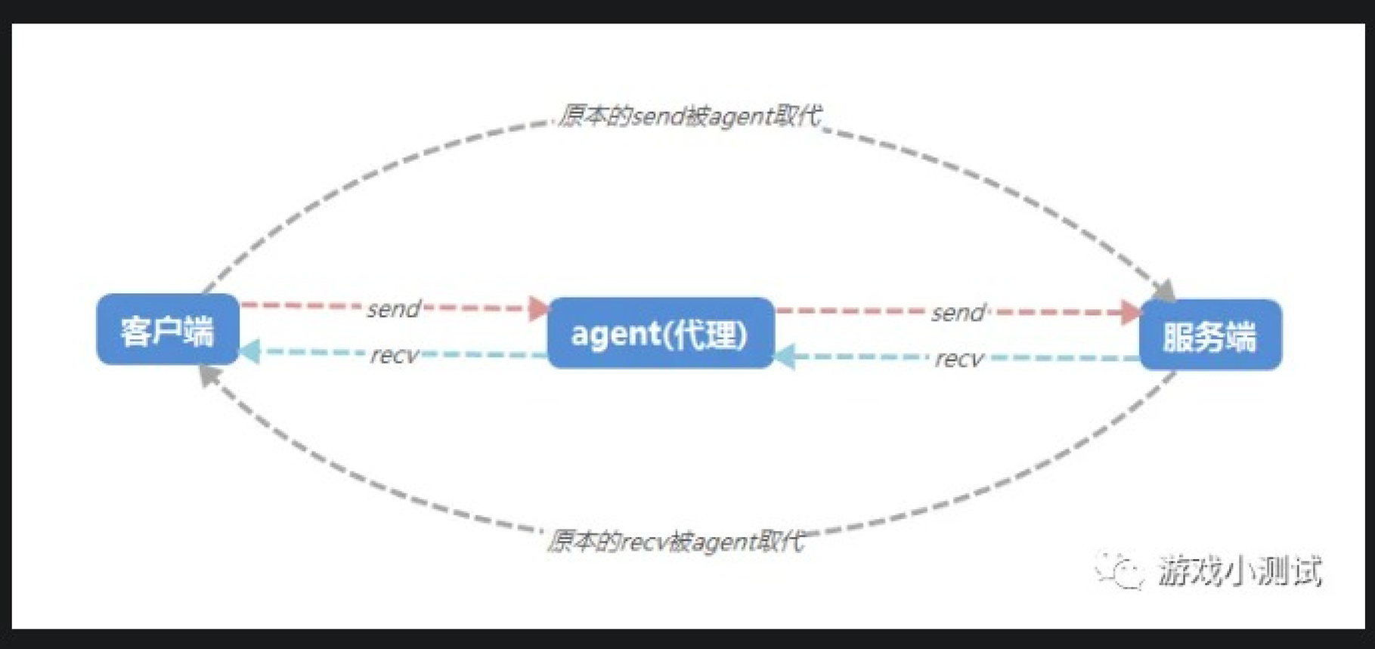
一开始，我是想模仿WPE写一个类似的工具的，然后我也对WPE的功能进行了分析，原理大概如下：



1. WPE加载客户端进程，然后将修改后的dll注入到客户端进程，hook客户端的send和recv方法调用。
2. 客户端进行发包操作，协议在通过客户端的socket时，会通过注入的dll进行处理，然后再发给服务端（处理可能包括拦截、修改等）。
3. 服务端在返回封包之后，也会通过客户端的socket，调用dll进行处理，然后再发给客户端。
4. WPE可以直接插入伪装包，通过dll调用直接将伪装的封包通过客户端的socket发给服务端。

WPE的方式有点像胁持客户端作为人质，刀架在客户端脖子上，客户端你要说什么需要通过我WPE来说，就算你不说我WPE也可以替你说，这里面最重要的技术是dll注入和钩子，这两个东西比较偏windows底层，我并非专业出身，只会一点简单的python，所以这块内容我没有弄懂，而且后来发现有些游戏的进程，WPE是找不到的，所以WPE也无法解决所有抓包问题，就放弃了这个方案。

转换一下思路，既然我不能劫持客户端，那我能不能做个中间商呢？你客户端和服务器之间不要直接对话，有啥想说的我来帮你们转达？我们简单的画图分析一下：

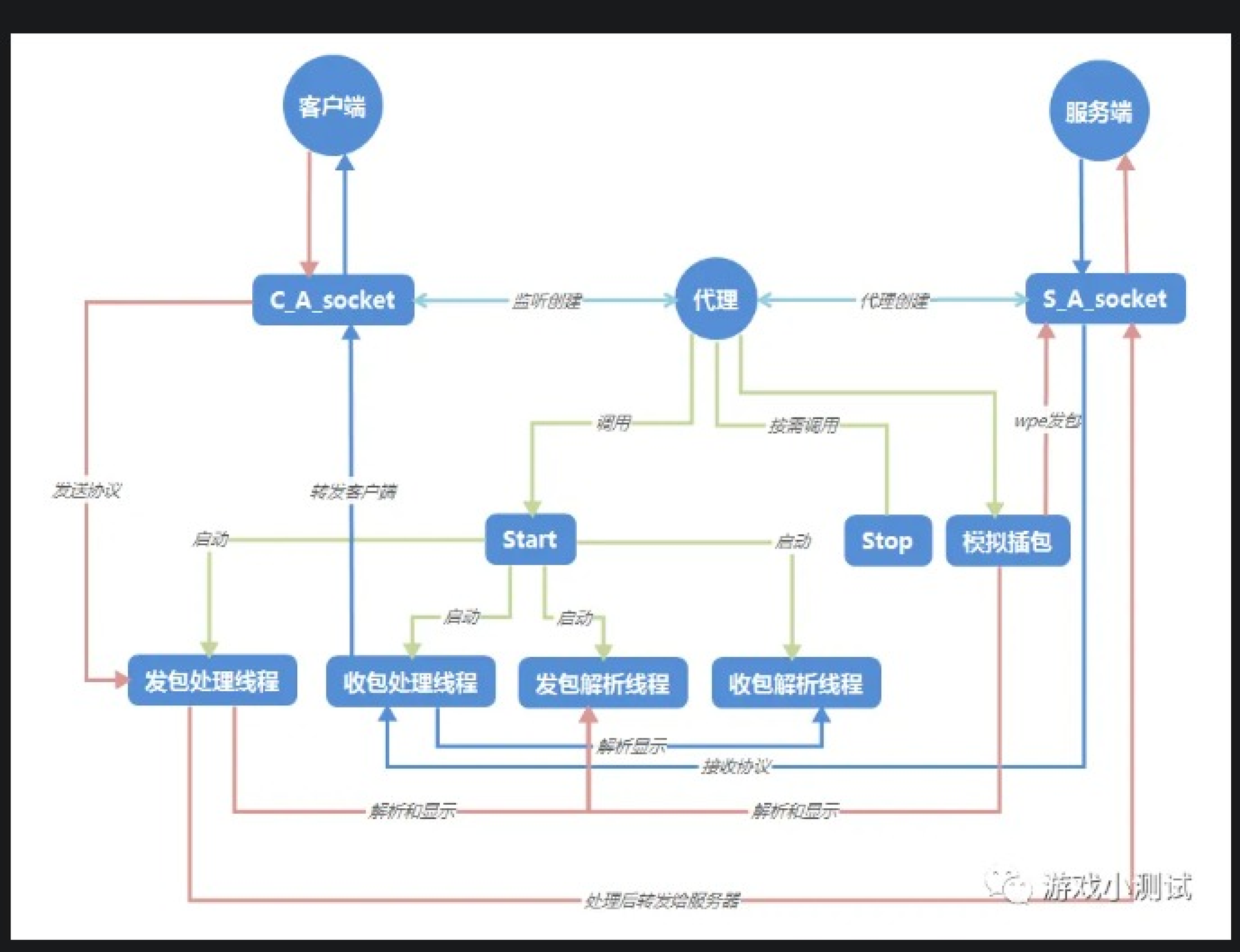


从图中可以看到，原本客户端与服务器是通过socket直连的，但是我们无法注入dll的话，就无法对收发过程进行干预，所以我们就需要在他们之间加一个代理，客户端发给服务器的包，先发到代理这里，然后代理再转发给服务器，同样的，服务器发给客户端的包，也是先发到代理这里，然后再转发给客户端。由于代理的功能是我们自己来实现的，所以在接收到客户端发给服务器的或者服务器返回的封包时，我们可以对其进行任意的修改。

我们写了一个简单的转发脚本（后续会讲），并在客户端配置了一个虚假的指向我代理地址的服务器列表，经测试，客户端成功连接到了服务器，并且打印了全部的收发协议内容，所以这个方法是可行的。

```
send-----b'\x82\x83\xda\xba\xc6\x91\xfd\xaa\xa6'
recv-----b"\x82\x06'\x11eH\x91L"
send-----b'\x82\x83\xa7\xe4\xee1\x80\xf4\x8c'
recv-----b"\x82\x06'\x11eH\x91M"
send-----b'\x82\x83\xe9\t\xa1\x15\xce\x19\xc5'
recv-----b"\x82\x06'\x11eH\x91N"
send-----b"\x82\x83'Gq\x82\x00W\x17"
recv-----b"\x82\x06'\x11eH\x91O"
send-----b'\x82\x83\x91L=\x9a\xb6\\U'
recv-----b"\x82\x06'\x11eH\x91P"
send-----b'\x82\x83\xf7c?\xb7\xd0sU'
recv-----b"\x82\x06'\x11eH\x91Q"
```


既然可行，那就这么干~！接下来我们回到我们的业务需求，对核心需求进行了分析，画了下面这张功能分析图：



1. 代理server启动监听，当客户端尝试连接服务器的时候，创建一个代理与客户端之间的链接C_A_socket。
2. 然后创建一个代理与服务器之间的链接S_A_socket，桥梁搭建完毕。
3. 调用代理的start方法，启动发包、收包、解析（协议明文化）等线程。发包线程持续从C_A_socket处recv（接收客户端发送的协议），收到协议之后根据需求进行一定的处理(修改、拦截等)，然后从S_A_socket处send（发送）给服务器，收包线程刚好相反，从S_A_socket处recv（接收服务器返回的协议），然后从C_A_socket处send（返回）给客户端，完成协议的转发。解析线程是将收发协议进行明文化解析，显示到工具上，方便查看。

这样一个最基本的抓包工具就分析完了，这些基本需求已经满足了抓包、发包、实时明文显示的功能，至少可以做到之前协议测试中提到的（逻辑验证，有效性分析，冗余性分析）测试需求，后面我们将围绕工具制作进行功能拆分并详细展开和实现，还请持续关注哦🧐

#测试工具 10 #抓包工具 15

测试工具 · 目录

‹ 上一篇
游戏抓包工具制作（四）—— 其他连接方式的实现（WebSocket）

下一篇 ›
游戏抓包工具制作（一）—— 最简单转发代理的实现

个人观点，仅供参考