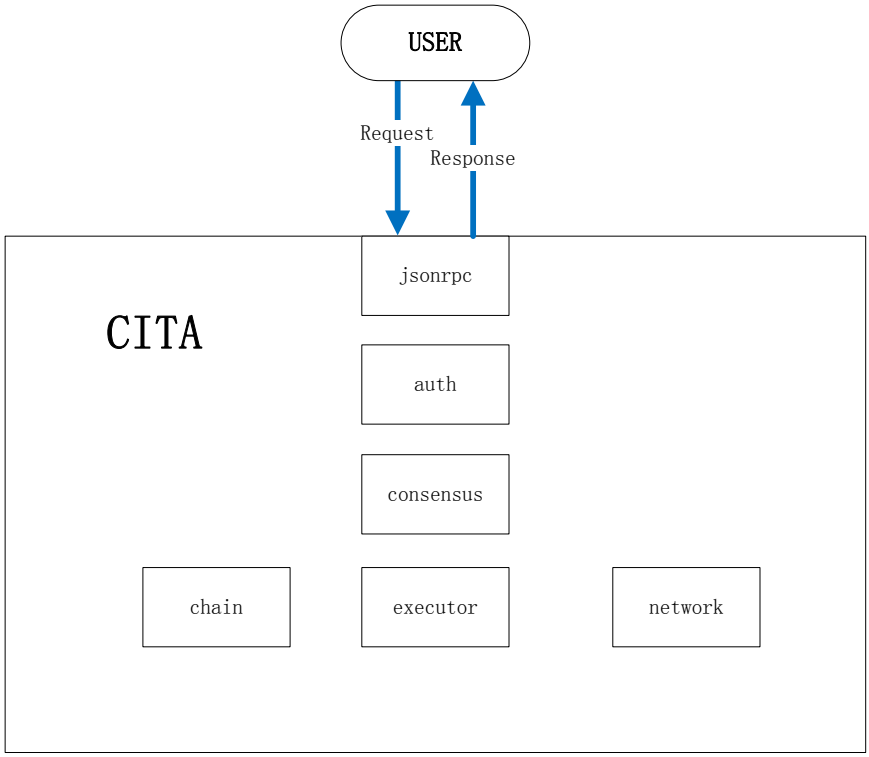
解读 CITA

在用户看来，cita 是个抽象的整体（见下图），可以想象成一个远程服务器，用户向 cita 发出各种请求（Request），然后得到响应（Response）。

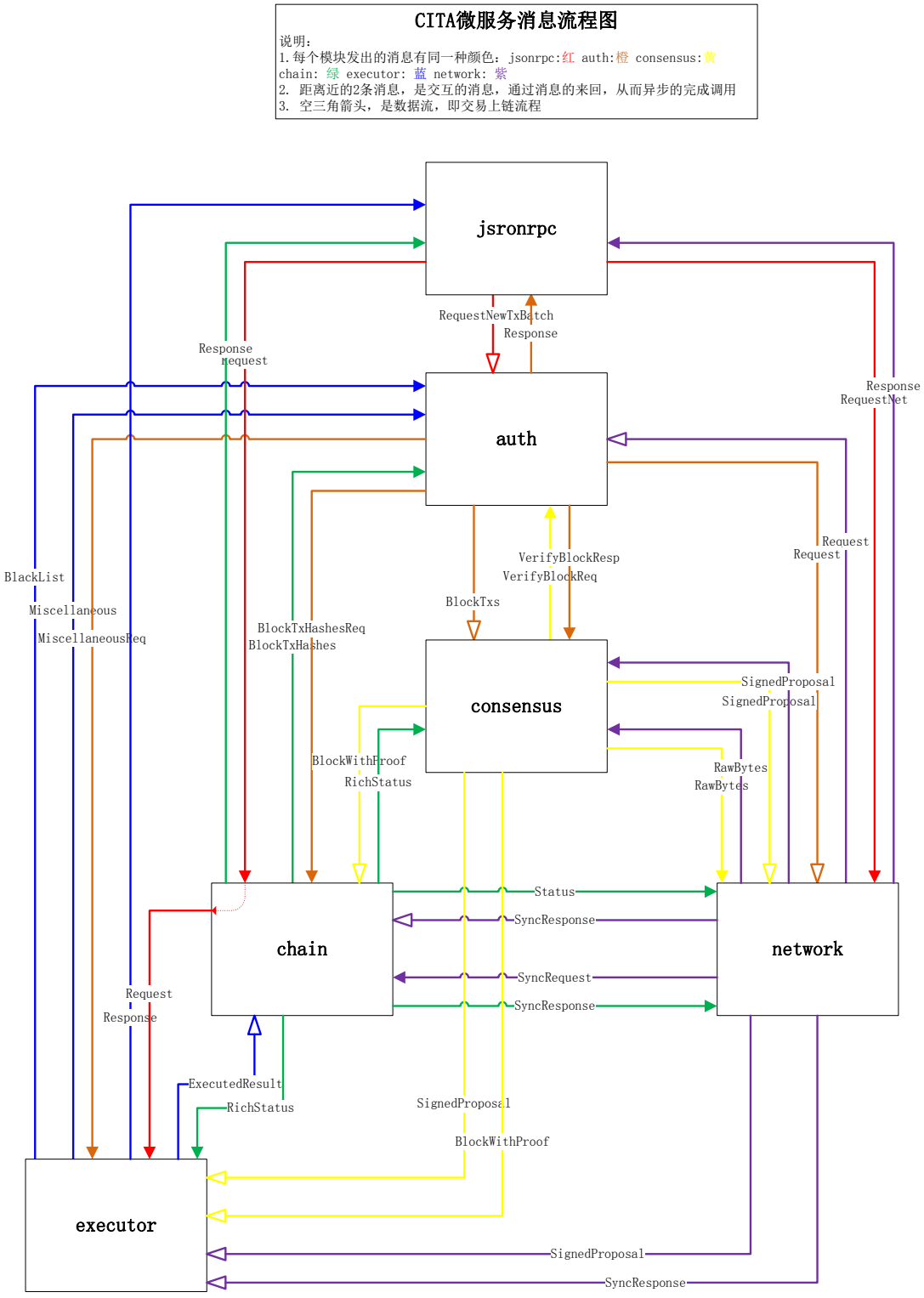
在 cita 内部看来，Jsonrpc 是接口模块，用户与 cita 的交互全部是通过 Jsonrpc 模块，是 jsonrpc 将 Request 引入到 cita 内部，从而产生各条脉络，最后的处理结果经过 jsonrpc 返回给用户。



我们先从功能上对 cita 有个整体的认识。

可以做个类比，cita 就像是银行系统，用户就像是客户。用户首先需要在银行系统中生成账户，这是后续所有业务的起始条件。这个账户可以是银行账户，可以存钱、查账、转账等。但这个账户可以比银行账户更高级，可以完成商业上的合同，多个实体之间可以在系统内签订合同、执行合同等。转账是非常具体的业务，合同比转账的抽象级别更高，可以以合同的形式实现转账。

然后我们需要理清 cita 内部的脉络，从代码实现上进行分析，这是接下来我们要做的所有工作。下图是整理出来的内部脉络图。第一眼看上去应该会感觉很乱，这是因为 cita 的微服务架构是通过异步的消息传输来完成调用关系的，所以会比宏架构的程序看上去复杂很多。微服务架构将整体的功能分成了这 6 个模块，我们接下来的分析方法就是逐个分析这 6 个模块，但是在脑子中要始终记得分析的原则：**理清 Request 的脉络走向**。



经验之谈

1. 看懂一个模块的第一步是看懂配置文件。
2. 分析线程模型、消息订阅关系以及消息分发关系

3. 掌握核心的数据结构
4. 看代码的时候要带着这样一种心态：我能否优化这个模块，使得这个模块更好？有没有更好的方式能够替代现有的实现？
举例来说，jsonrpc 模块，我能否通过优化使得这个接口模块更好，从而能够更快的响应用户请求呢？或者是否有其他更好的接口技术，而不（只）是 Jsonrpc？