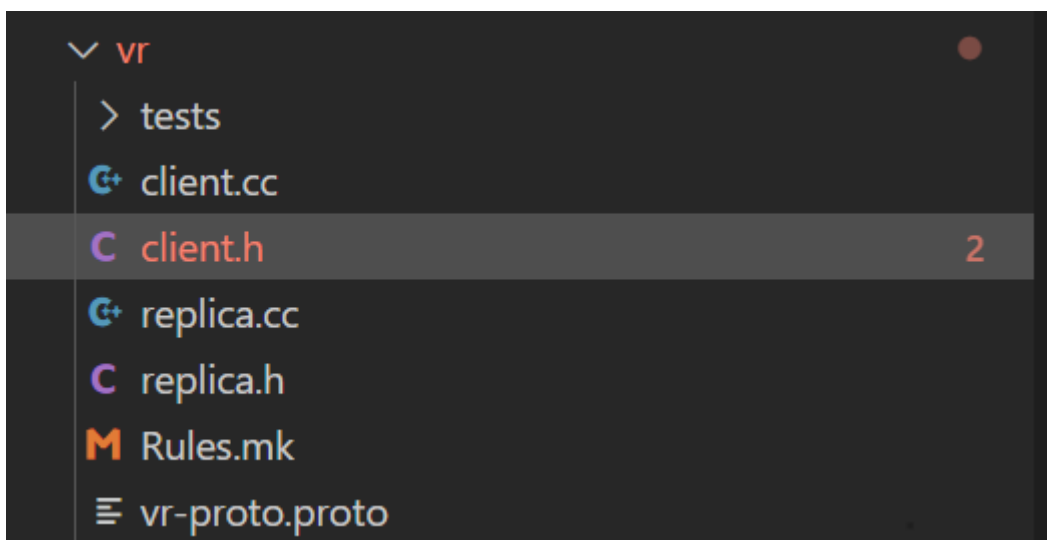


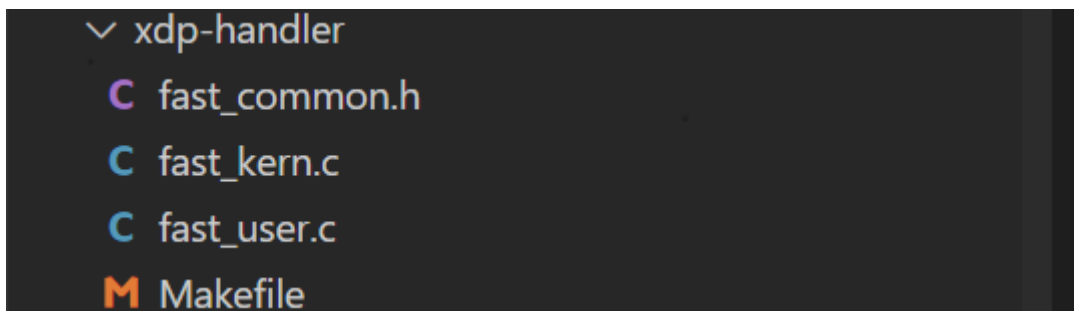
电极源码分析

电极源码中主要包含两个部分

1. 在“VR/”中实现 VR（Viewstapped Replication）协议，代码来自[推测性 Paxos](#). 其中做了一些修改来实现功能。

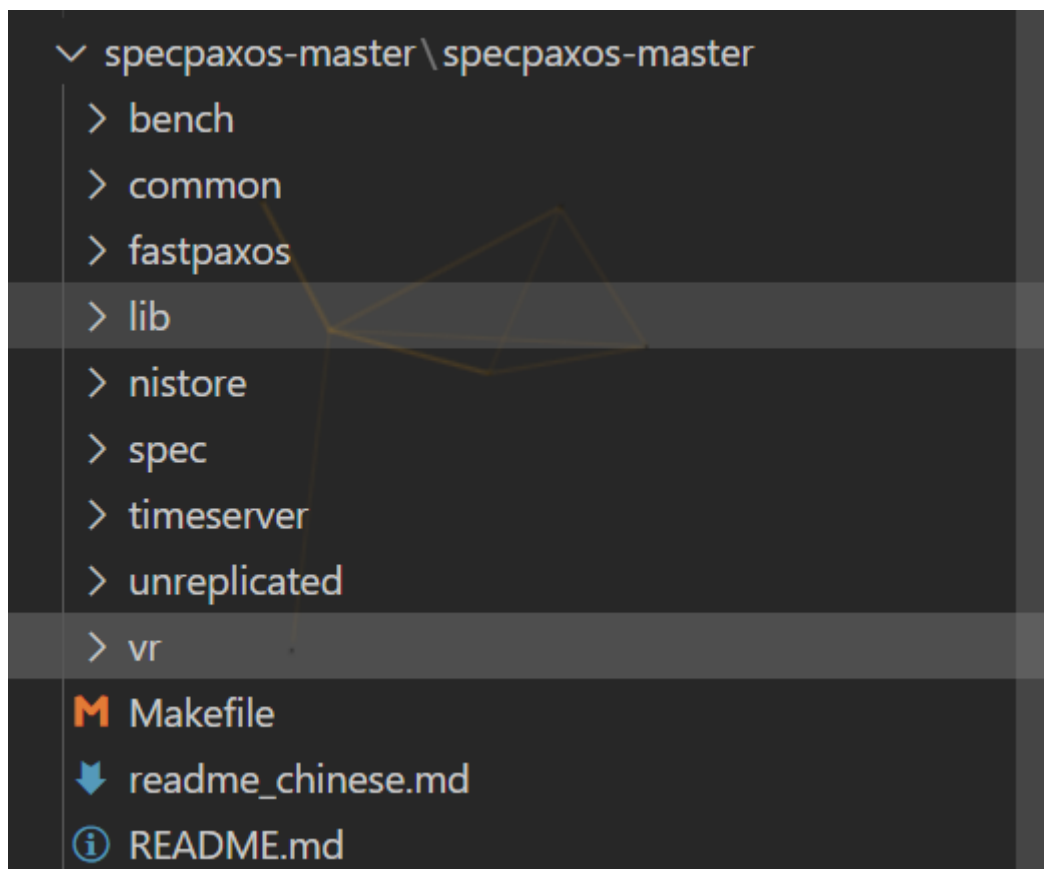


2. 通过 eBPF 中实现三个优化，实现在“xdp handler/”中。原始结构来自[BMC]的源代码 (<https://github.com/Orange-OpenSource/bmc-cache>).



先前分析中可知，电极中关于paxos协议的代码都来自于[推测性 Paxos](#)，下载该项目源码，对其进行分析，和电极中的代码进行比较。

specpaxos项目结构：



- bench: 一个以尽可能快的速度进行的基准测试
- fastpaxos: fast paxos, 只实现了基本情况（不知此处的fast paxos指的是什么，要探究可能需要进一步阅读论文）
- nistore: 一个事务性键值存储（“nistore”），支持使用任意一个严格的多版本控制和并发控制两阶段锁定或乐观并发控制。（这是源自中使用的代码库TAPIR）
- spec: Speculative Paxos, 推测性paxos, 实现包括正常操作、同步和协调协议。（同样不知区别）
- timeserver: 一个用于分布式的存储系统简单的时间戳服务器。
- unreplicated: 一个简单用于比较的原始版本
- vr: Viewstamped Replication (VR)，又名 Multi-Paxos, 论文"[Viewstamped Replication Revisited](#)"中实现的，包括可选的批处理优化

根据电极中readme的信息，其对paxos主要进行的修改集中在vr文件夹中，因此，当前我认为电极仅对 Multi-Paxos进行加速优化，和论文的信息也较为符合。

比较代码的改变，电极改变的代码内容集中在replica.cc和replica.h两个文件中。

电极关键代码

replica.cc

归纳其中主要函数及其主要功能

1. request_polling函数

```
#ifdef FAST_BATCH
```

函数接受三个参数：ctx（上下文指针），data（数据指针），data_sz（数据大小）创建一个UDPTTransportAddress对象，表示发送者的地址，并调用HandleRequest方法处理请求

2. prepare_polling

```
#ifdef FAST_REPLY
```

创建了一个PrepareMessage类型的静态变量prepare，然后调用prepare.ParseFromString方法将数据解析到prepare变量中，HandlePrepare_Kernel处理请求

3. VRReplica构造函数

值得一提，bpf_obj_get：通过文件名称获取本进程中的句柄fd

```
#ifdef FAST_BATCH---> request_buffer_fd
```

```
#ifdef FAST_REPLY--->prepare_buffer_fd
```

```
#if defined FAST_REPLY || defined FAST_BATCH || defined FAST_QUORUM_PRUNE---  
>paxos_ctr_state_fd
```

4. ModifyKernelState函数

```
#if defined FAST_REPLY || defined FAST_BATCH || defined FAST_QUORUM_PRUNE
```

修改内核状态。定义了一个结构体paxos_ctr_state，包含了一些状态信息

当前视图号、当前状态、最后一次操作、当前节点的索引、领导者节点的索引、批处理大小
调用bpf_map_update_elem函数将其更新到名为paxos_ctr_state_fd的BPF映射

5. GenerateNonce函数

生成随机数

6. AmLeader

GetLeaderIndex(view) == myIdx，查看当前视图是不是领导结点

7. CommitUpTo

- 提交日志中的操作，直到指定的操作号。这个过程包括执行操作、更新状态、发送回复等。
- @param upto 指定需要提交到哪个操作号

8. SendPrepareOKs

- 向其他副本发送PREPAREOK消息，用于确认新的未提交操作
- 遍历从oldLastOp到lastOp的所有操作，对那些新且未提交的操作，发送PREPAREOK消息。PREPAREOK消息包含操作号、视图号和发送者的索引
- @param oldLastOp 旧的最后一个操作号
- 观察到SendMessageToReplica函数，多了参数，所有发送文件的消息的函数都多了这个参数，后面详解。

9. SendRecoveryMessages

- 发送恢复信息给所有副本，以请求恢复操作。

10. RequestStateTransfer

- 请求状态传输的函数。当发现当前状态陈旧时，会调用此函数来请求更新状态。

11. EnterView

- 用来进行视图切换操作的，根据当前节点是否为领导者进行不同的逻辑处理，并且在进入新视图时进行相应的状态更新和超时计时器管理

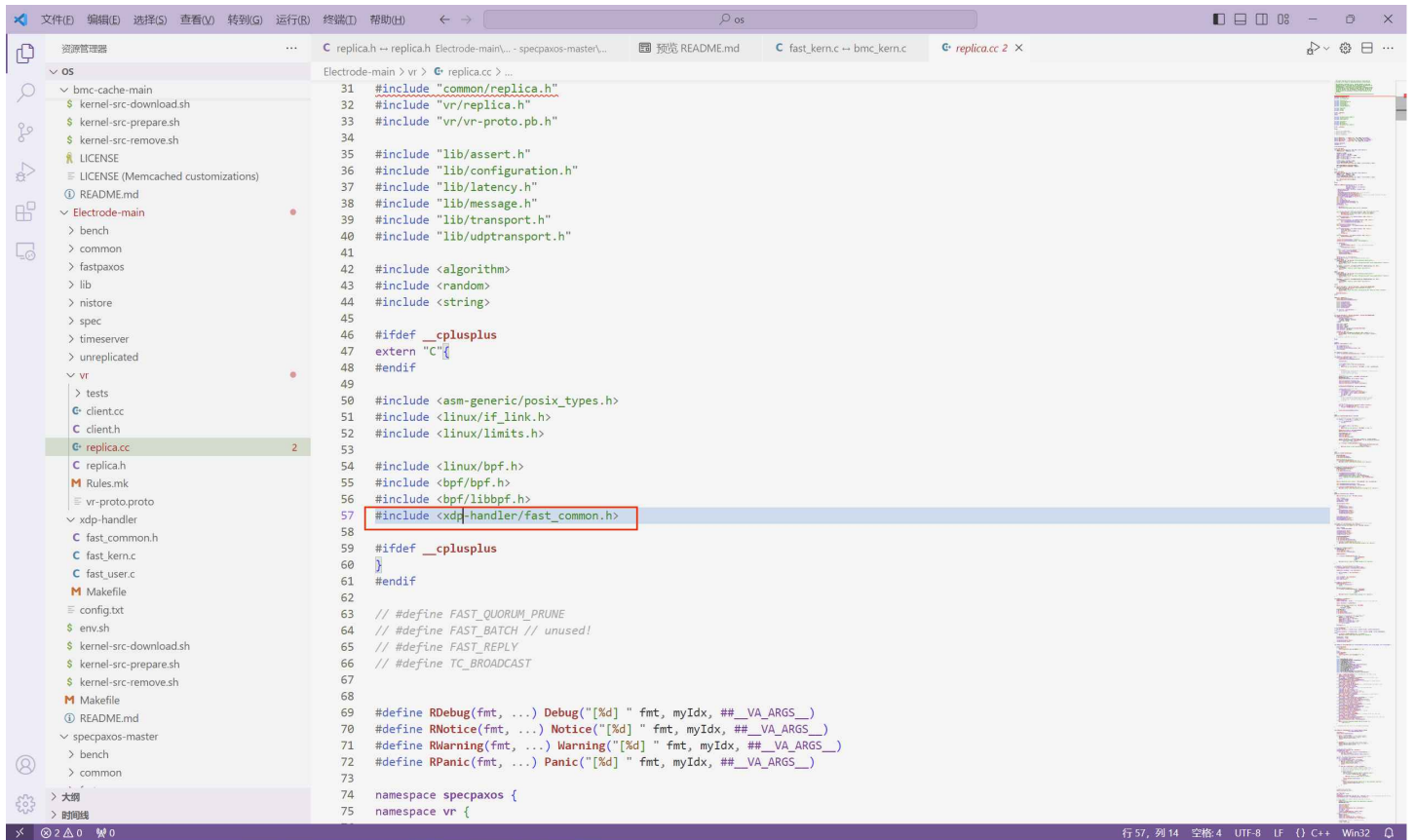
xdp-handler

观察对比源码以及修改后代码可发现与ebpf相关部分主要包含在文件夹xdp-handler中以及文件replica.cc和replica.h文件中：

```
> G+ replica.cc Electrode-main\vr 16
v C replica.h Electrode-main\vr 2 6
#include <linux/bpf.h>
#include <bpf/bpf.h>
#include <bpf/bpf.h>
#include <bpf/libbpf.h>
#include <bpf/libbpf.h>
// asd123www: about the eBPF.
> C fast_kern.c Electrode-main\xdp-handler 54
> C fast_user.c Electrode-main\xdp-handler 77
```

主要的优化代码实现在xdp-handler/中，这个结构来自于bmc的源代码，即此优化代码是根据bmc改编的。

对于xdp-handler中fast_common.h进行直接调用位于文件replica.cc中：



而fast_common.h代码内容分析如下：

以上代码是一个头文件，主要定义了一些常量、枚举类型和结构体，以及一些预定义的XDP程序和TC程序的操作码。

1. 定义了常量：

- ETH_ALEN：以太网地址的字节数
- CLUSTER_SIZE：集群大小，值为3
- FAST_REPLICA_MAX：最大副本数量，值为100
- NONFRAG_MAGIC和FRAG_MAGIC：用于识别非分片和分片数据的魔数
- MAGIC_LEN、REQUEST_TYPE_LEN、PREPARE_TYPE_LEN、PREPAREOK_TYPE_LEN、MYPREPAREOK_TYPE_LEN、FAST_PAXOS_DATA_LEN：用于定义数据的长度
- BROADCAST_SIGN_BIT：广播标志位
- QUORUM_SIZE：法定数量大小
- QUORUM_BITSET_ENTRY：用于记录法定数量的位集合条目大小

2. 定义了枚举类型ReplicaStatus，包括三个状态：STATUS_NORMAL、STATUS_VIEW_CHANGE、STATUS_RECOVERING，表示副本的状态

3. 定义了两个枚举类型，分别表示XDP程序和TC程序的操作码

4. 定义了一个结构体paxos_configure，用于存储Paxos配置信息，包括IP地址、端口和以太网地址

总的来说，该头文件主要用于定义一些常量、枚举类型和结构体，在XDP和TC程序中使用这些定义来进行相关操作。