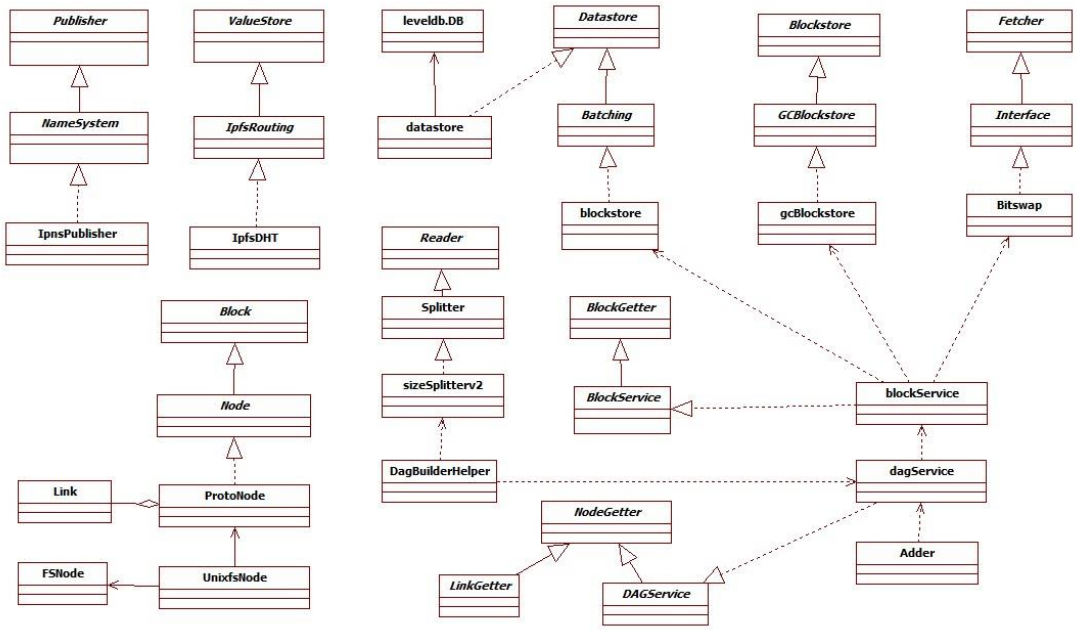
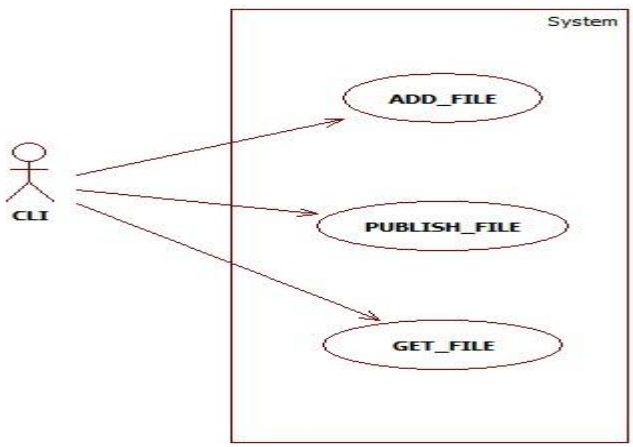
IPFS 代码解析

1 IPFS 文件操作

1.1 IPFS 文件操作数据结构

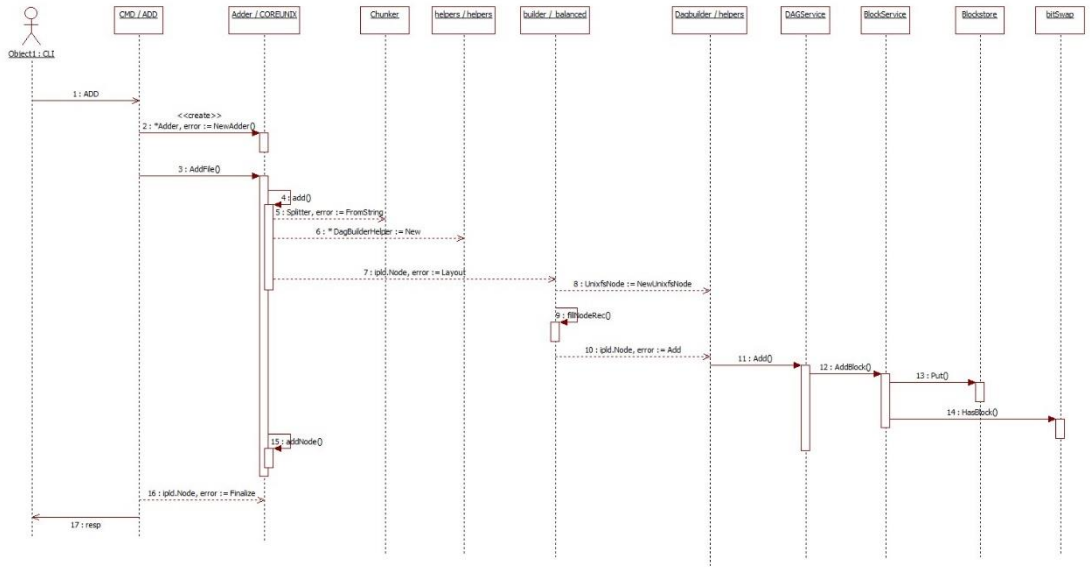


1.2 IPFS 文件操作主要 UseCase

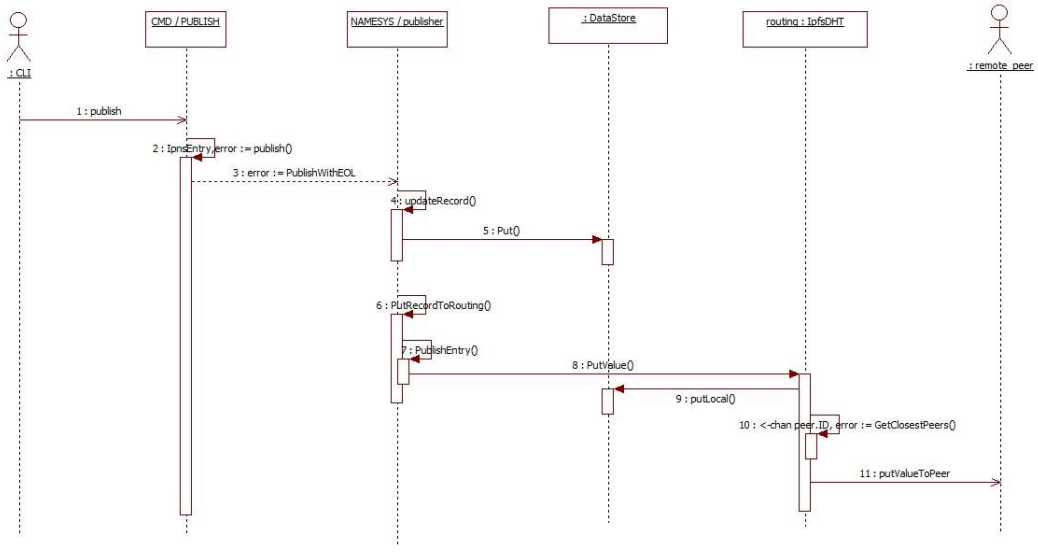


1.2 IPFS 文件操作 Usecase

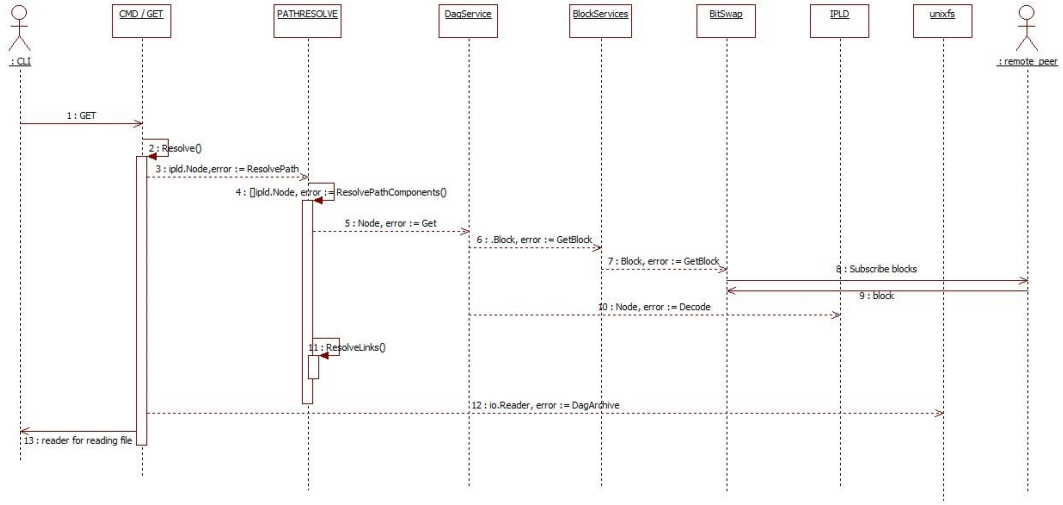
1.2.1 add file



1.2.2 publish file

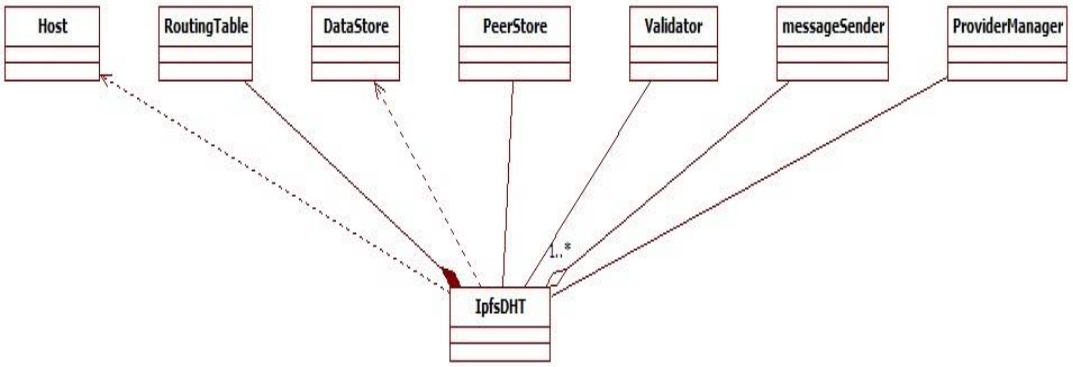


1.2.3 get file

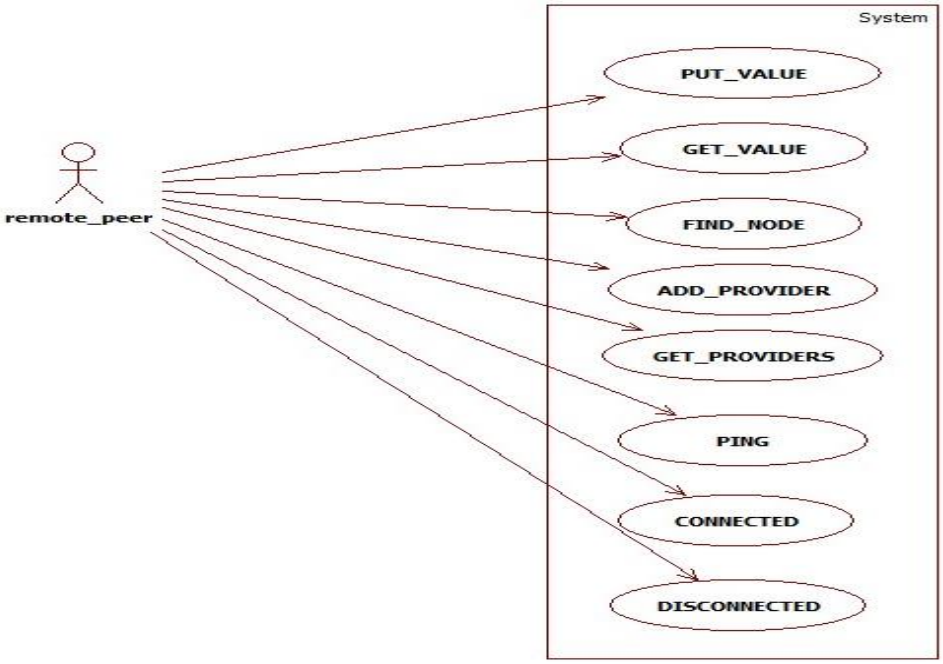


2 IpfsDHT

2.1 IpfsDHT 数据结构

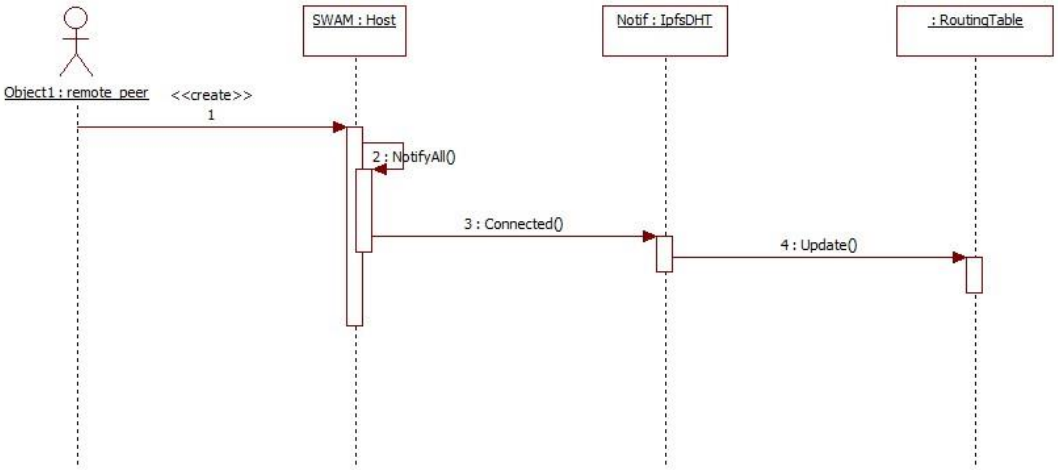


2.2 DHT 主要 UseCase

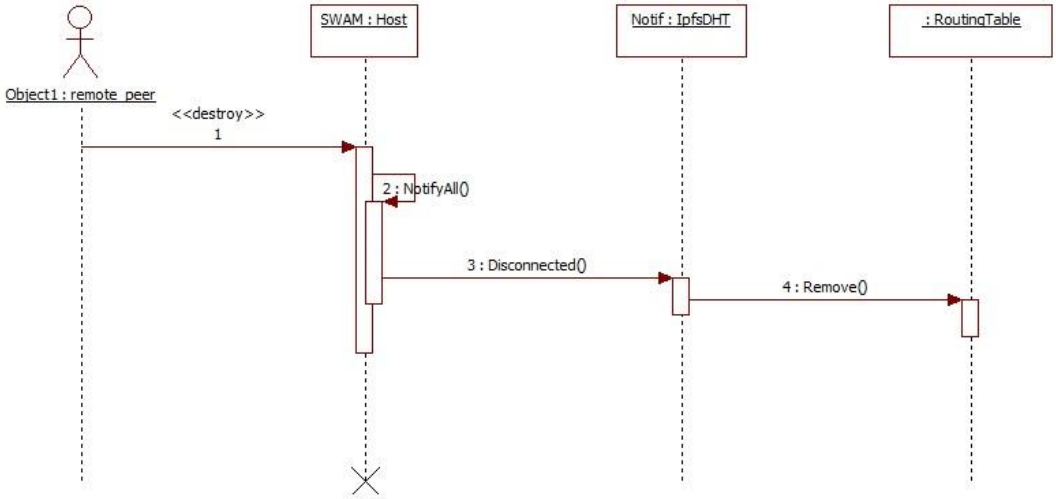


2.3 DHT UseCase 时序图

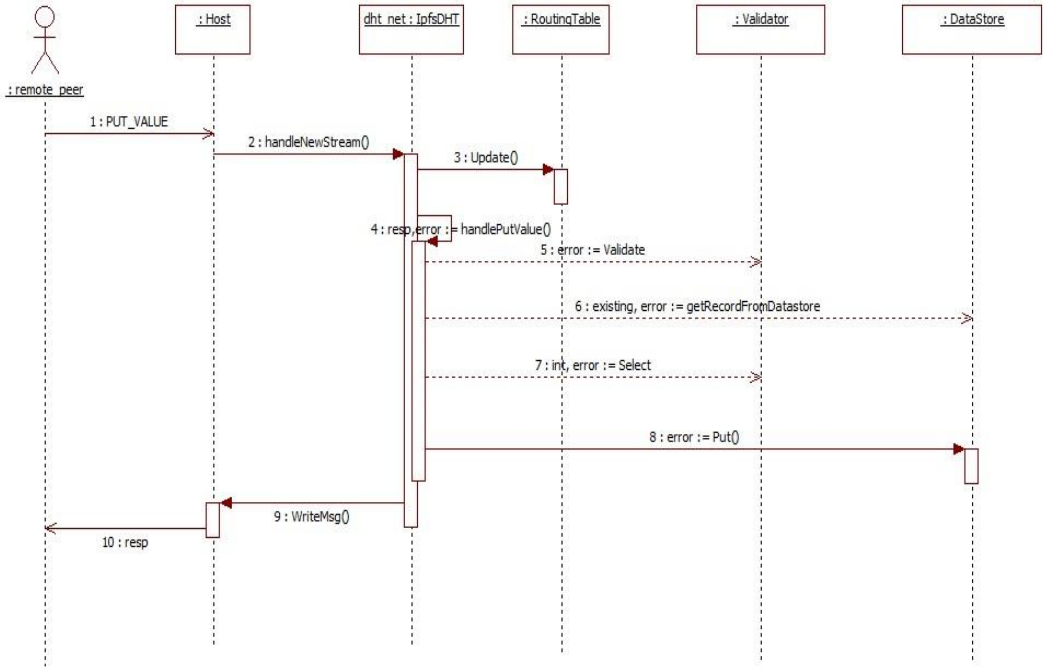
2.3.1 peer connected



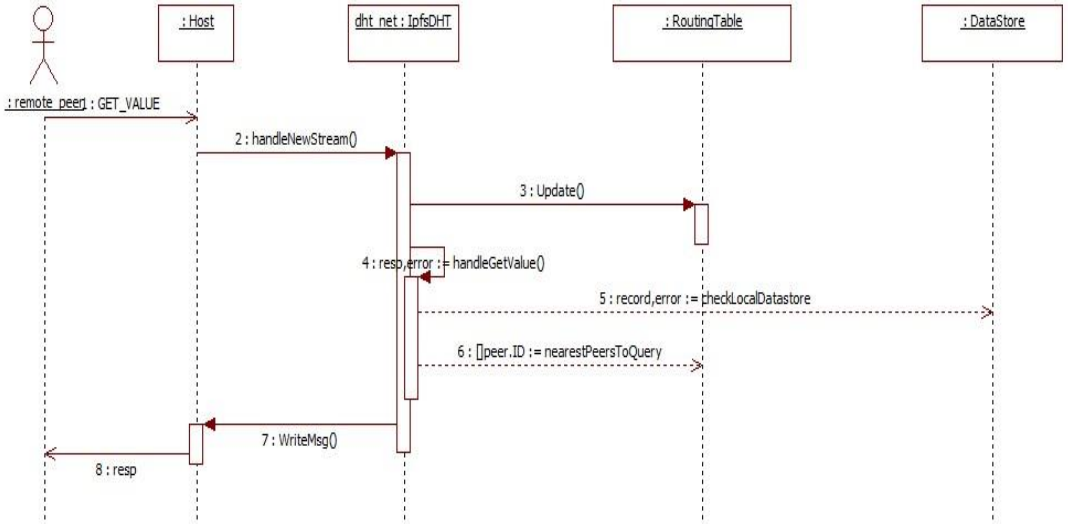
2.3.2 peer disconnected



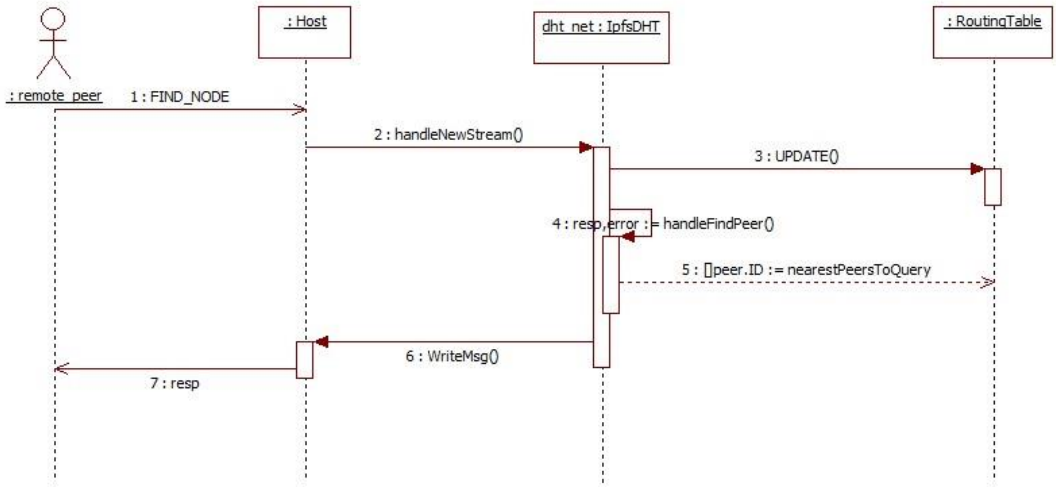
2.3.3 put value



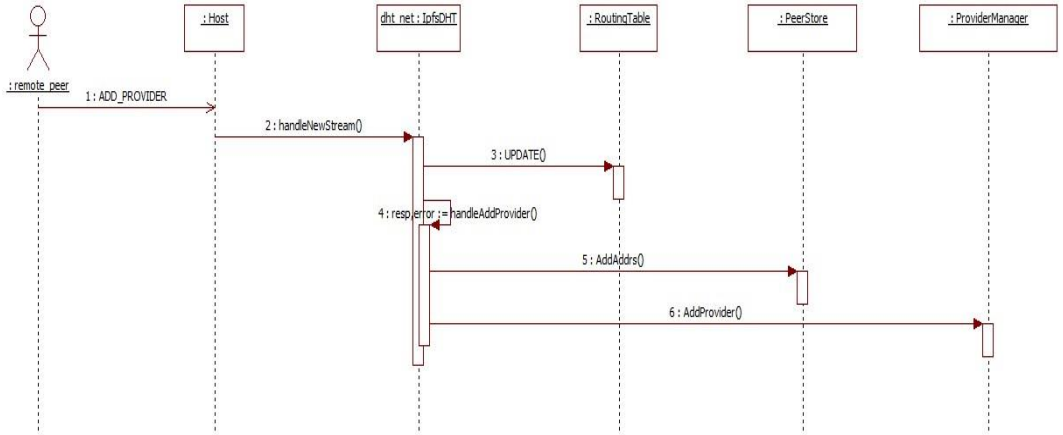
2.3.4 get value



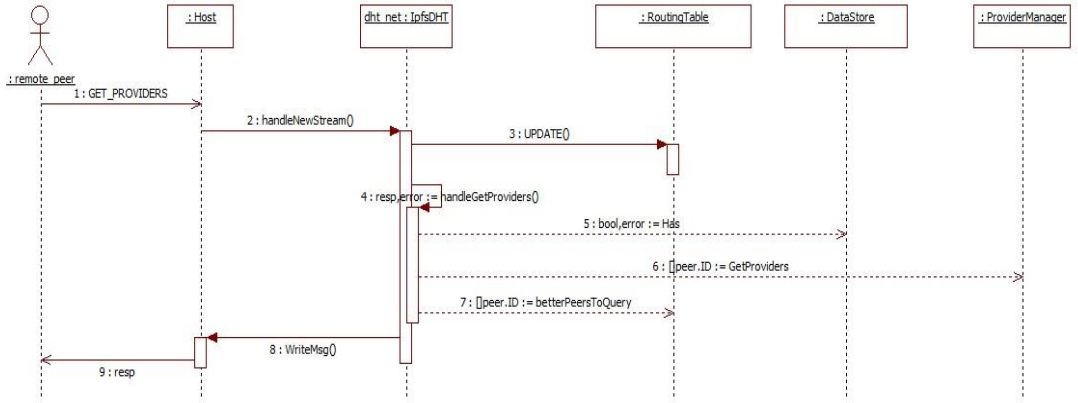
2.3.5 find node



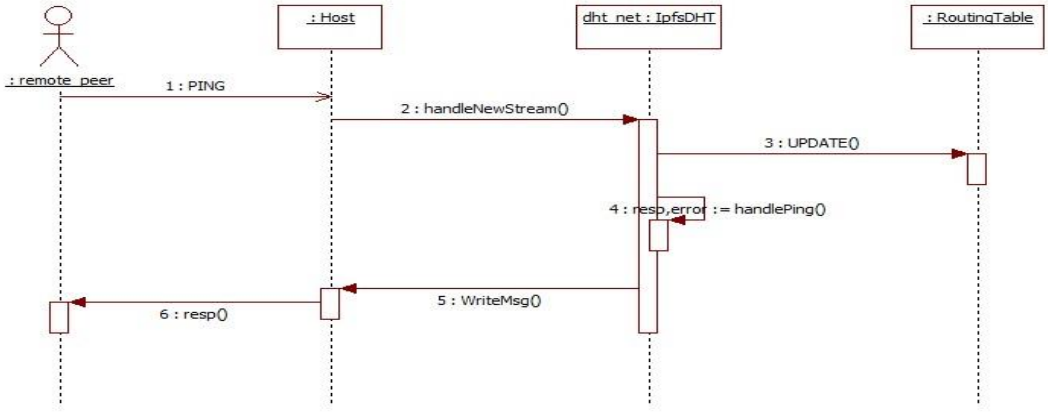
2.3.6 add provider



2.3.7 get providers

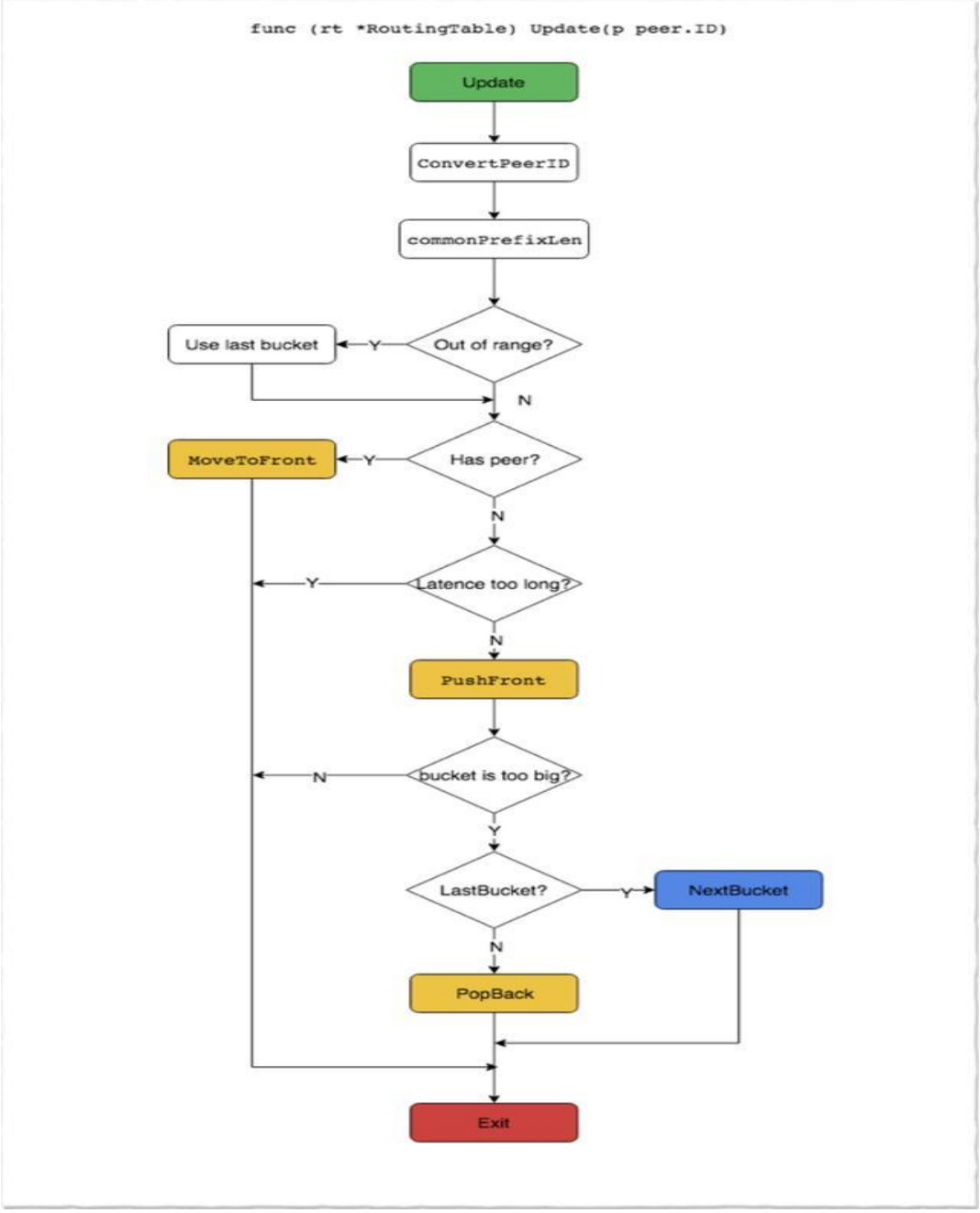


2.3.8 ping



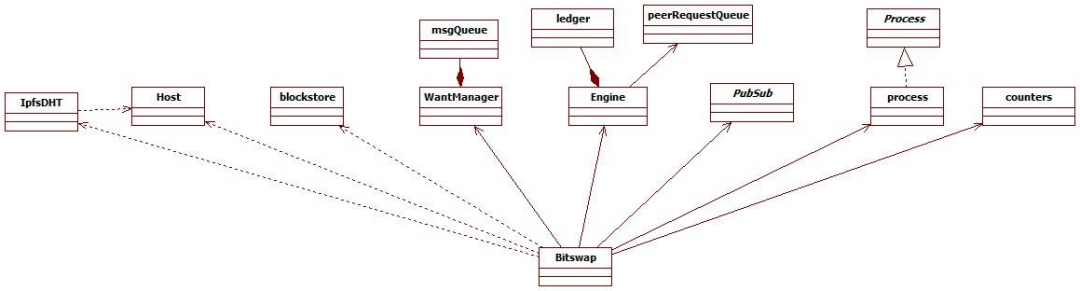
2.4 DHT KAD 路由更新算法

下图借用《IPFS 技术深度解析》

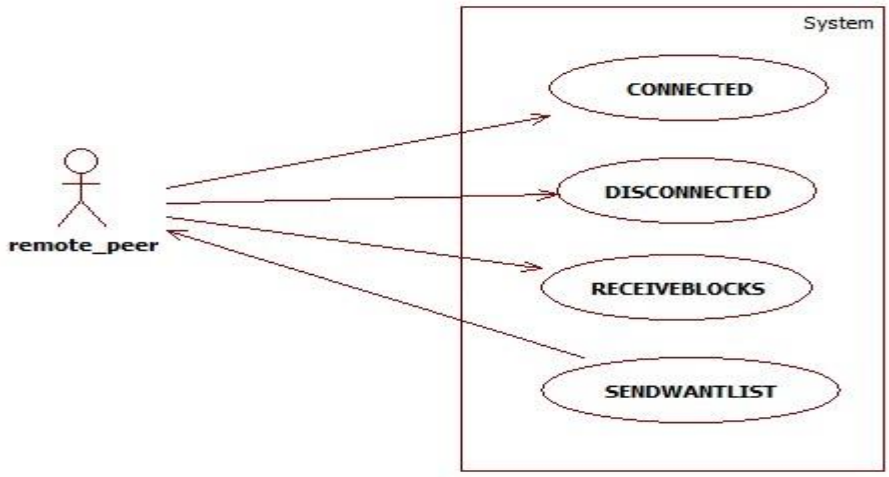


3 Bitswap

3.1 Bitswap 数据结构



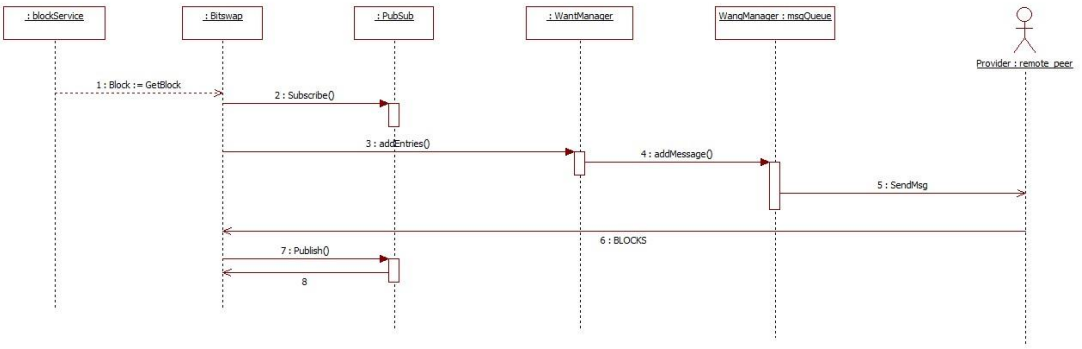
3.2 Bitswap 主要 UseCase



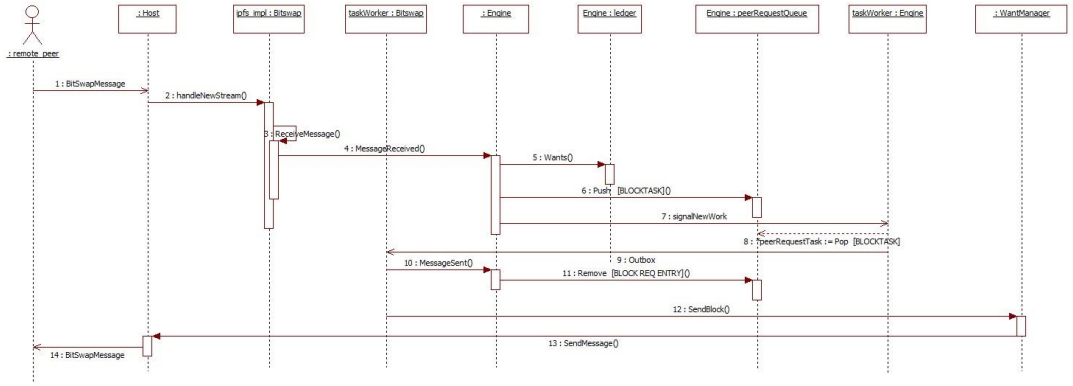
Bitswap 主要是负责本地和远端 IPFS 节点间 Block 的 GET/PUT 操作的一个协议:
1) 如果 GET 本地节点不存在的 BLOCK,则 DAG 服务会向 BitSwap 添加一个新的 want entry, 从而触发向其远端节点获取 block,远端节点通过查询 BLOCK 的 provider 来获取.
2) 在获取到这个 block 后会向远端节点回送一个 Cancel 消息，不再接受这个 BLOCK.
3)接收到 BLOCK 后不仅回保存在本地, 同时也会查询那些远端节点向本地提交了这个 block 的 want request，从而向其分发这个 BLOCK.
4）**Bitswap 通过一个 WRR 调度的队列来实现对节点的 GET 策略控制，每个节点的 weight 是通过其账本来计算生成，目前这部分功能在 IPFS 中还没有实现。**

3.3 Bitswap UseCase 时序图

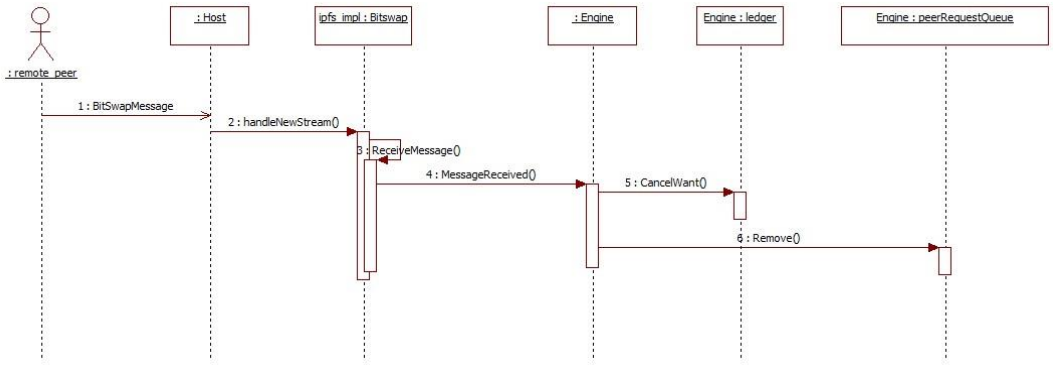
3.3.1 发送 block wantlist



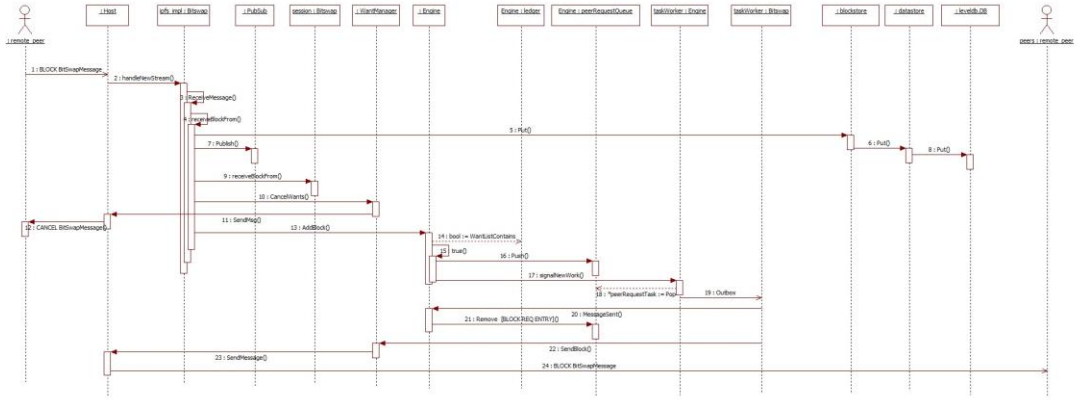
3.3.2 接收 block wantlist



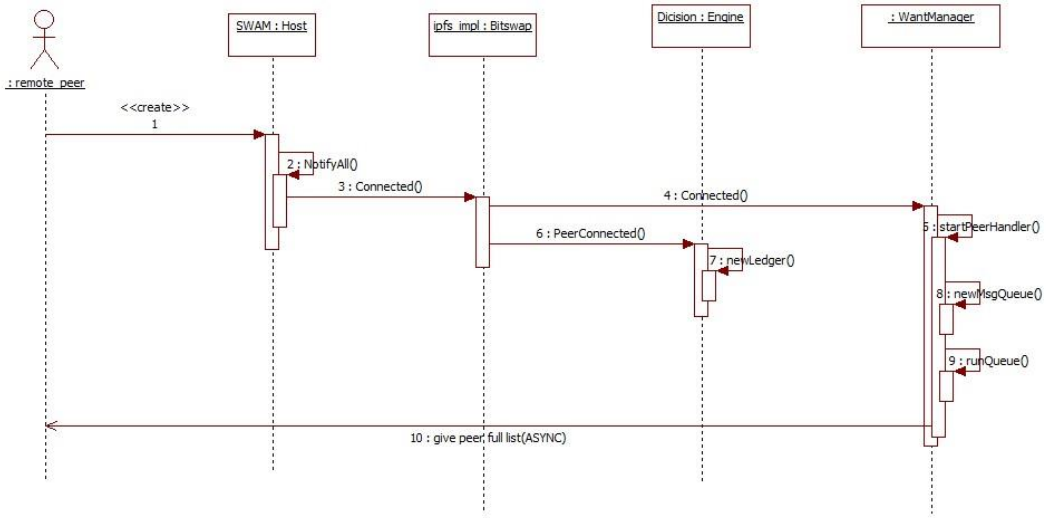
3.3.3 发送 block cancel



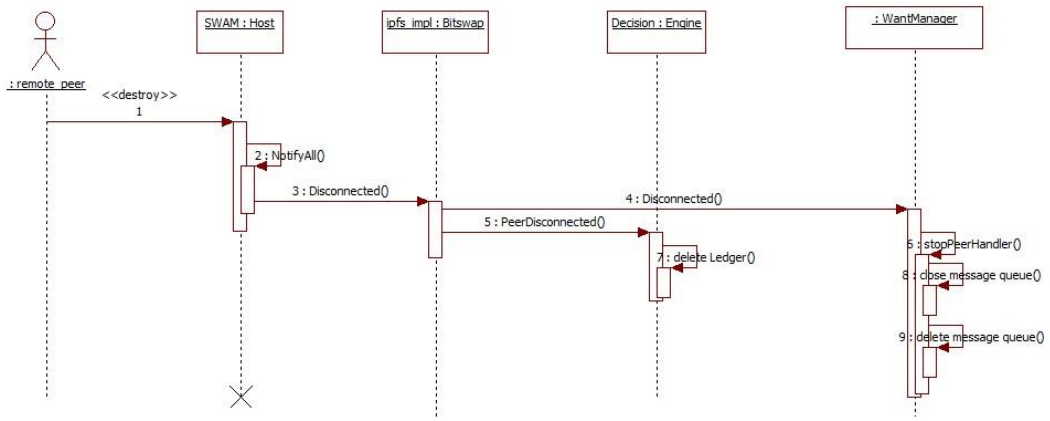
3.3.4 接收 block



3.3.5 peer conneted



3.3.6 peer disconnected



4. IPLD

IPLD(Inter Planetary Linked Data)是为了实现不同领域之间交互的数据模型，对这个模型的相关概念可以参考其它的文档说明。这里主要关注代码实现中如何添加一个新格式的 BLOCK 的主要操作流程.

在 IPLD 数据模型中把本地文件添加到 IPFS 中会根据制定的文件类型完成 Encode，获取 IPFS 中的文件则会根据 BLOCK 类型完成 Decode。不同领域的数据类型需要通过一个 plugin 的方式来集成，其主要步骤有：

1) BLOCK 接口的实现
IPFS 中定义了一个统一的 Block 抽象接口，各个不同类型的数据需要完成这个接口的实现(go-block-format)，具体可参考(go-ipld-git 中的 tag,tree 等实现)：

```
type Block interface {
   RawData() []byte
   Cid() *cid.Cid
   String() string
   Loggable() map[string]interface{}
}

type Cid struct {
   version uint64
   codec   uint64 //区分不同的数据类型，需要在 plugin 中指定
   hash    mh.Multihash
}
```

2) IPLD plugin 接口的实现(具体可参考 git plugin 实现)
IPLD 对不同的数据类型集成定义了一个统一的 plugin 接口(go-ipfs/plugin):

```
type Plugin interface {
   // Name should return unique name of the plugin
   Name() string
   // Version returns current version of the plugin
   Version() string
   // Init is called once when the Plugin is being loaded
   Init() error
}
```

```
type PluginIPLD interface {
   Plugin

   RegisterBlockDecoders(dec ipld.BlockDecoder) error
   RegisterInputEncParsers(iec coredag.InputEncParsers) error
}
```

3) IPLD plugin 加载过程

```
LoadPlugins
        |--Init //plugin 需要实现的接口
        |--runIPLDPlugin
              |
                |--RegisterBlockDecoders //decode
            |--RegisterInputEncParsers  //encode
        |--runTracerPlugin
```

4) 举例

在 IPFS IPLD 模块的代码中提供了一个 Batsh(具体参考 go-ipld-format 中的 bash 文件)的数据结构来配合完成相关和 DAG 的对接操作.外部命令可参考： ipfs dag put **–format ABA**

1. ***通过 coredag.ParseInputs(ienc, format, file, mhType, -1)完成文件到 IPLD.NODE 的转换，这个函数会根据 format 调用对应 IPLD plugin 的 encode 函数***
2. ***添加上一步得到的 ipld node 到 BATSH 中，有 BATSH 向 DAG service 完成数据提交。***