P2PCache框架设计

## Session

在新的框架中，一个session管理一种协议类型的交互，如果RCS支持多种协议，比如支持BT、PPS等，那么就应该创建BtSession和PpsSession等。

框架中实现了一个抽象的Session，各协议需要各自从Session派生，构建各自的Session。

Session的创建

SessionManager在初始化的时候，会读取配置文件，查看哪些协议开关打开，如果协议开关打开就会创建相应的session。

## Socket

系统涉及到服务端和客户端两类socket，使用socket的地方包括

1. 协议连接。等待用户向服务器发起协议连接或者主动向客户端发起协议连接。
2. Tracker通信。向tracker请求peer、请求metadata或者发起announce。
3. UGS和RCS通信。系统内部的心跳检测，状态报告等。

### socket的创建

协议连接socket的主要功能是进行协议监听，当然，UDP和TCP socket的通信机制不太一样，对于TCP来说，会有一个建立连接的过程，然后再进行数据通信；对于UDP来说是直接进行数据通信的。对于上层来说，其实是不需要关注通信机制的，关注的是接收数据后的处理以及数据的发送。基于此，我们将socket进行更高层的抽象。

协议连接的socket由session进行管理，一个session可以同时在多个socket上进行监听，这也就意味着，一个session可能需要管理多个服务端socket，在实现的时候，需要保证接收报文和发送报文的socket一致。

### Socket报文处理的线程模型

系统的socket实现使用的是boost的asio库，其本身已经是一个异步处理模型，不过当前所有的异步处理都是在session的主线程执行，如果性能达不到要求，后续可以考虑使用多线程进行处理。

## PeerConnection

连接是本端与对端进行交互的逻辑载体，可以认为是一个逻辑管道，一个连接的最重要属性就是两个端点：本端和对端。因此对于我们的系统来说，创建一个连接的必要条件就是知道对端信息。

### 连接创建

这里所说的连接指的是协议连接，就是要进行协议通信的连接。系统在两种情况下会建立协议连接，第一种情况是本端正在进行监听，对端主动向本端发起连接/发送数据；第二种情况是本端需要主动去连接对端，从对端下载资源或者向对端推送资源。

框架中用PeerConnection来表示一个连接

连接的校验

1. 如果是外网用户发起的连接，此连接关联的资源并不存在，则不应该创建连接。
2. 如果配置了IP地址过滤，则在过滤地址段的IP地址不应该建立连接。

### 连接管理

协议连接都是与具体资源关联的，一个协议连接必定是属于某一个torrent的，因此，连接都是由torrent来进行管理的。

连接管理的内容

1. 连接的中断

在协议交互过程中，发生了严重错误，如超时、垃圾数据等，则应该将连接中断。

1. 连接的暂停

如果资源还没有下载完成，且连接数已经达到上限，此时有新的连接到达，同时，连接队列中存在下载速度较常慢的peer，则可以先将其暂停，优先创建新的连接。

1. 连接的重启

暂停的连接在满足一定条件后，又可以重新启动进入协议交互阶段。

1. 连接的清理

对于已经中断/停止的连接，需要及时清除，session的tick任务负责这一工作。

1. 连接的带宽控制

对内网用户和外网用户分别实施不同的带宽限制策略。

### 协议处理

理论上，框架的实现应该不涉及到具体协议，各种协议的实现细节不尽相同，很难将这些特殊元素纳入到框架中来。不过，为了将协议实现较好的融入到框架中来，我们有必要将协议进行适当抽象，这就协议状态机的由来。

协议状态机隶属于协议连接，每一个协议连接都运行各自独立的协议状态机，协议状态机由事件驱动。

## Torrent

Torrent代表一个资源，和资源相关的属性都由torrent管理维护，同时torrent又由session进行管理。框架中定义用一个InfoHash来标识一个torrent，也就是说，只要能获取到一个资源的InfoHash就能够建立此资源对应的torrent。

Torrent内容

1. InfoHash：标识一个资源。
2. File：包含的文件，可能存在多个文件。
3. FastResume：快速启动文件，系统启动时，会通过FastResume来读取已经存在的资源。
4. MetaData：元数据，对于BT协议来说就是种子文件。

### Torrent的创建

当前，创建torrent的时机有两个，一个是系统启动加载是，会读取fastresume文件，此时需要创建torrent；另一个就是当内网用户发起连接，并发送了包含了InfoHash的报文，此时应该建立torrent，立即获取资源并提供服务。

### InfoHash的获取

如果资源已经存在，则应该从FastResume中读取InfoHash。如果资源不存在，则必须等待用户来主动连接，并带来InfoHash。

### File的创建

一个Torrent包含哪些文件是由MetaData来描述的，因此，MetaData就成了非常重要的数据。在获取到了MetaData后，就可以创建资源相关的所有文件了。

### FastResume

为了支持系统的快速启动，需要使用FastResume文件。FastResume定义了启动一个任务的最小数据集合，一个FastResume文件对应一个资源。系统启动时，会逐一读取所有FastResume文件建立必要的数据结构，来快速提供服务。当然，FastResume文件也需要进行校验，校验机制可以有多种，再次不详细描述。

FastResume文件的格式采用Bencode编码，文件的内容对于不同协议不尽相同，由各个协议自己定制。

FastResume写入的时机：

1. 资源下载完成。
2. Torrent被停止。

### MetaData

Torrent的元数据，用来描述资源的概要信息，它包含了建立下载任务的必要数据。对于BT协议来说，元数据就是种子文件，对于其他协议，不尽相同。MetaData是torrent的最重要数据之一，在启动一个任务后，第一件事情就是要获取到MetaData。

MetaData的获取是一个比较关键的技术之一。

## Peer

Peer可以用一个IP地址和Port来标识，可以认为代表一个下载客户端。

系统中所有Peer都是由Policy来管理，每一个Torrent都包含一个Policy。

### Peer的获取

系统启动后是没有Peer的，此时我们要等待用户来主动连接我们，当第一个用户连接系统并创建了torrent后，就需要快速下载资源，这时系统需要想办法获取更多的peer，一般都是通过向Tracker服务器请求peer。

因此，获取peer的途径可以归纳为：

1. peer主动连接过来。内网peer主要是通过前端UGS的重定向获取的，外网peer则是通过不断的发布announce，将系统的地址发布出去。
2. 主动获取peer。比如向Tracker请求peer、通过DHT网络获取。

### Peer的连接策略

Peer连接策略，简单来说就是应该向哪个peer发起连接。Peer连接策略可以做的很复杂，也可以做的很简单，刚开始我们可以做的比较简单，可以认为所有peer都是平等的，为了适应复杂的网络环境，后续可以继续优化。

## IO