引子：当你拿到这串神秘链接，迫不及待地想要进入维斯特洛大陆领略各方势力倾轧的精彩故事时，究竟是谁决定了你还有多久才能看上权力的游戏？

这是一张BT的架构图，我们首先可以看到这里有一台Torrent服务器。当你通过磁力链接创建下载任务的时候，BT软件就会向Torrent服务器发出请求，获得torrent文件，也就是俗称的种子。种子储存了你所下载文件的全部信息。接着，你使用的BT软件通过解析种子获得Tracker服务器的地址。那么Tracker服务器有什么用呢？Tracker翻译成中文就是追踪者，它会告诉你谁拥有你正在下载的文件。当然，并不是所有人都拥有完整的文件，这里就要提到BT的文件分片技术，拥有部分分片的人可以向没有这些分片的人进行传输，比如你，而拥有完整文件的人将只进行传输，这就是下载者和做种者的区别。这两者结合组成了一个去中心化的网络，在有人做种的情况下，下载的人越多，下载的速度就越快。

那么在BT中，有三个比较重要的组件，第一个就是我们之前提到的Tracker，它是活跃peer的追踪者；第二个是DHT网络，网络中的节点同时扮演Peer和DHT node的角色，这将是我们在之后重点介绍的。第三个就是种子文件，它构成了bt协议的核心。

我们知道，利用tracker服务器找到其他peer效率很高，可是如果请求很多，tracker服务器处理不过来，或者tracker服务器宕机怎么办？这时就要提到BT协议的另一种结构：DHT网络。在这个网络中，每一个主机都会维护一些文件的索引，这样每个peer都相当于一个小型tracker，一个查找请求可以通过这个网络慢慢传播，将压力分摊到每个人身上，这就提高了系统的鲁棒性。

好的，讲完Tracker我们接着来介绍torrent文件。这个文件采用的是bencoding编码，这里就不展开讲了。种子文件主要由一些字段组成。它的announce字段就记录了BT Tracker的地址，下面的一些字段提供了一些次要的辅助信息，最重要的字段就是最后的info字段。它给出了资源的名称、分片的长度以及每个分片的哈希编码，这个段哈希值就是实现文件分片的重要依据，它的每20字节都表示一个分片的SHA1哈希编码。最后这个字段呢相当于一个文件目录，记录了文件的相对路径和大小。

好的，最重要的部分来了，接下来我将稍微深入地介绍一下DHT网络。可能会有点难，但是如果听懂了会发现非常有意思。DHT网络主要采用KAD协议。这个协议会给网络中的每个节点分配一个随机生成的SHA1哈希编码作为ID，需要注意的是，这个ID和通过torrent文件info字段生成的info\_hash使用的是同一种方法，他们的长度都是160位。那么KAD协议的核心思想就是：如果某个节点的ID和某个文件的哈希值距离较近，那么这个节点就有责任知道这个文件存放在哪些节点上。节点维护的信息就是一个路由表，它采用最长公共前缀的方法来划分成多个桶以支持快速的查找请求，每个桶最多保存K个节点，所以也叫K桶，在之后会举例说明。那么两个不同的哈希编码怎么比较距离呢？KAD协议利用了异或的算法，值越大，距离越远。

这里我们看一个例子。实际的哈希编码是160位，这里我们就简化成5位。假设文件1的哈希值是10000，节点C和D的ID分别是10000和10001，节点ID和文件哈希的异或距离是多少？一个是0，一个是1。也就是说，节点C和节点D都有责任知道文件1的存储位置。那我们假设有一个新加入的节点想要下载文件1，它就会在网络中找到节点C或者节点D，从而知道文件1保存在哪个节点上。这就引出了一个问题，当利用torrent文件创建任务后，怎么找到与文件哈希编码距离最近的DHT节点？这就要提到路由表的结构。

这里我们假设节点的ID是01010，如果一个节点的ID是01011，只有最后一位不同，它就会被存在第1个桶中，这时候进行异或运算可以得到一个距离范围，是0到2的1次方。如果一个节点的ID从倒数第2位开始不同，它会被存放在第2个桶中，异或运算的距离范围是2的1次方到2的2次方。以此类推，编码有多长就会有多少个桶，每个桶中所有节点ID和这个节点ID的距离都在对应的范围内。那么对于文件哈希为10000，我们怎么找到和它距离最近的节点？我们可以发现文件从第1位就开始不同，那么我们就在第5个桶中选择异或距离和10000最近的节点，问问它知不知道文件放在哪，如果它也不知道，那我们再从它的第4个桶中选择。这样不断迭代，最终就可以确定文件的位置，这实质上是一个高效的二分查找过程。桶的标号每减1，范围就缩短一半。接下来，我们一起来看一个视频，以一种更生动的方式了解BT的工作原理。