P2P协议之Bittorrent研究报告

北京大学网络与分布式实验室
周模
zhoumo@net.pku.edu.cn

December 20, 2005

1 说明

为了实现对网络中特定P2P协议的识别和流量统计，本文作者在对Bittorrent进行充分的调查和研究的基础上，编写了此文档。本文档描述了Bittorrent网络组织的特点，节点的角色及功能，以及各个阶段数据包的结构及消息数据流程。重点介绍了该协议数据流的识别方法。

本文档所描述的内容经过试验验证，证明了所描述内容的正确性，因此，可以作为Bittorrent被动流量识别及统计程序的参考文档。也可以为网络爬虫的主动识别方式提供参考。

2 协议概况

Bittorrent协议(以下简称BT)为Bittorrent以及其它功能类似的客户端所使用的通信协议。它的特点是专注于下载同样内容的节点之间的最快速的内容发布。每个节点首先使用其它方式获取种子文件，这个文件大小比较小，通常在几十k到上百k左右，和要共享的内容相比，下载种子文件所需要的网络带宽基本上可以忽略不计，因此本报告主体部分的分析基于以下前提：Peers已经通过其它途径(http,ftp,...)获取到种子文件。即本文不对peers如何获取种子文件进行分析。

2.1 协议简介

BT协议中，BT种子文件通过计算"info_hash"来和其它种子进行区分。对于每个节点来说，获取其它的正在下载同一个文件的节点信息是很重要的。这一过程可以通过向tracker进行查询来完成，也可以通过DHT网络来完成这一过程。通过DHT网络的方式在另外一篇关于Kademlia的协议分析报告中有专门介绍。

2.2 bencode编码协议

为了后面叙述的方便，在开始有必要介绍一下BT采用的一种编码方式，bencode。BT的作者使用了一种比较简单易懂的编码方式来对设计种子文件。这种编码方式能够很简单得对python中的各种数据类型，如字符串，整数，列表，字典等进行编码。而且对于类型的嵌套，如一个列表中的元素又是一个列表等情况能够进行很好得处理。

首先判断要编码的数据的类型，然后根据这个类型执行相应的编码方案：

对IntType和LongType类型进行编码的方案为一个字母"i"加上这个数值的字符串表示再加上一个字母"e"。就是说整数19851122将会被编码成"i19851122e"。

对StringType类型进行编码的方案为字符串的长度加上一个冒号":"再加上字符串本身。例如helloworld将被编码成"10:helloworld"。

对ListType和TupleType类型进行编码的方案为一个字母"l"，然后递归得调用相应的编码函数将列表或者元组的所有元素进行bencode，最后编上一个"e"结束。

对DictType类型进行编码的方案为一个字母"d"，然后递归得对每个元素进行处理。在DictType中，每个元素都由一个key和value对组成。首先以长度加":"加实际值的方式编码key，因为key通常都是简单值，所以可以这样编码。然后对value进行bencode，最后加上一个"e"结束。

通过分析以上的编码函数我们可以看出，复杂的对象被以此种编码方式进行编码后将能够无歧义地被还原出来。例如字典：{'key1':'value1','key2':'value2'}经过bencode后将变成：

d4:key16:value14:key26:value2e

这样就解决了比较复杂的数据类型在网络中传输的问题，后面在协议分析中只需要提到要传输的数据的类型，然后提到"进行bencode编码后传输"即可。

2.3 BT协议中的角色及功能

BT协议中，有普通节点和tracker服务器这两种角色。普通的节点向tracker服务器查询正在下载某个种子文件的其它的普通节点的信息。tracker服务器则维护不同的种子的下载节点列表，并且做一些统计工作(非必须)。

2.4 网络拓扑

BT协议的网络拓扑结构中不存在特殊的结构，所有正在下载同一个种子文件的普通节点之间都有可能建立连接。

3 结点加入阶段

在BT协议中，一个节点开始下载某个种子文件的内容时，它首先要从tracker服务器中获取正在下载这一文件的其它节点信息。这一部分使用标准的http协议，关于http协议的详细格式，可以查阅相关RFC文档，这里假设读者已经对http协议比较熟悉。因此本部分的协议描述以http协议的基本单元为基础。实际的网络中，该部分协议可以以http或者https协议完成。

3.1 新节点向tracker服务器进行发布的消息流程

向tracker服务器进行发布和获取信息这个过程在整个下载过程中根据需要，执行多次。每一次都是一对普通的http请求和应答。

3.2 节点向tracker服务器发送请求

这一步的目的是请求获取peers的信息，使用info_hash作为关键字来区分不同的种子，tracker服务器维护了很多种子的下载列表。另外，当一个peer向tracker发送这条请求的时候，它也同时把自己的信息告诉tracker，tracker把它的信息放在下载某个种子的列表中，并进行维护。

基本请求：

http GET /announce/info_hash=%s&peer_id=%s&port=%s&key=%s

info_hash为种子文件的"info"字段的哈希值，唯一得区分出不同的种子。该值长度为20字节，采用了URLEncoded的方法进行转化，即所有的普通字符不变，特殊字符转化成"%XX"这样的形式(这样实际在网络中进行传输时此值的长度通常要大于20字节)。

peer_id为自己的标识，长度为20字节，它的结构在下面的握手协议中有具体描述。

port为该PEERS提供BT服务的端口。

key是一个四字节长的随机值经过HEX编码后的结果(这样在网络上传输的实际值为一个8字节的字符串，每个字节的取值范围为'0'-'9'以及'a'-'f')，这个值在本次下载任务中保持不变。即完成一个下载任务时，向tracker进行发布的

次数可能不止一次，但是在对应本次下载任务中的每次publish请求的key值都是相同的。

附加请求信息：

在BT客户端中，向tracker发布信息的时候除了以上信息外，还会根据情况提交一些附加的信息，而这些信息会附带在上述基本http请求的后面，仍然保持”key1=value1&key2=value2”这样的形式。即基本请求信息和附加请求信息合并在一起以一条标准的http请求发出。这些附加信息可能有：

uploaded：自从和上一次publish相比，又上传了多少字节(仅包含peers之间的传输，和trackers之间的信息交互不计算在内)

downloaded：自从和上一次publish相比，又下载了多少字节。

left：还剩多少字节。以上三项的值都是直接使用十进制的数字表示，即100字节将会转化成字符串”100”在网络中传输。

numwant：希望tracker给自己传回多少个peers信息。注：tracker不一定会遵守这个值。

compact：tracker是否要将传回的信息进行压缩。此值为1则tracker将进行压缩以节省网络带宽。

event：事件，可以取三种值：started，completed和stopped。其中started值在第一次向tracker进行publish的时候出现，completed和stopped分别在peers完成下载和退出的时候向trackers进行发布的时候出现。

其它http请求头部信息：

在官方的Bittorrent里，请求的头部信息会包括”User-Agent”，其值为”BitTorrent/x.y.z”，其中x.y.z为版本号。

3.3 tracker服务器向节点发送回应

在正常情况下返回http response.

response头部信息有：

Content-Type=text/plain

Pragma=no-cache

如果是信息的格式是压缩的，则头部信息还有：

Content-Encoding=gzip

返回的数据类型是一个字典，其中最主要的值为关键字”peers”所对应的值，它包含了peers的IP，端口等信息。这些信息使用了”bencode”方式进行编码。

3.4 识别方法

首先使用判断http协议的方法从网络中抽取出http连接，然后在这些http请求中寻找具有模式”/announce/info_hash=”的字符串，如果匹配成功，则该http连接为一次BT发布请求。

4 数据下载或传输

在BT协议中，节点成功获取其它的正在下载该种子文件的节点信息后，即尝试和它们建立TCP连接。连接建立之后，所有的控制消息和数据消息都在这一连接上传输，即BT协议不区分控制连接和数据连接。

4.1 握手消息格式

当节点一连接到节点二时，对其发送握手消息，消息格式如下：

1字节的协议名称长度+协议名称+8字节的标志+20字节的info_hash+20字节的PEERID

下面分别解释这些部分。

协议名称，为常数字符串"BitTorrent protocol"，此项成为检测BT协议的重要依据。

8字节标志：在版本4.1.0之前(无DHT支持)，是一个由8个ASCII值为0的字节组成的串。在支持DHT的版本中(4.1.0及该版本之后)，该项由7个ASCII值为0的字节加上1个ASCII值为1的字节组成。即PEERS在通信的过程中根据这项标志确定双方是否支持DHT。

20字节的info_hash：用以确定要下载的是哪个种子文件对应的内容。

20字节的PEERID，这项信息可以区别出不同的BT客户端。在官方BT中，该值构造如下：

Mx-y-z–**********

其中x.y.z为当前Bittorrent的版本号。例如4.2.0版的Bittorrent的PEERID的前八个字节即为M4-2-0–。后面的十二个*号是将某个随机字符串进行SHA-1哈希之后，取出后面六个字节，然后再进行hex编码得到。这个随机字符串的计算方法则是使用当前系统时间加空格再加当前进程ID得到。

当节点二决定和节点一建立连接时，将发送同样格式的握手消息。

说明：当握手协议结束后，双方正式建立连接，在后面的协议中完全处于平等的地位，此时连接是由谁发起的已经不重要了。即后面的协议中的“节点一”很可能是现在的“节点二”，当然也可能还是现在的“节点一”。

4.2 正式消息的分割格式

BT协议在握手消息结束之后，所有传输的数据都可以被分割为基本消息，每一条消息都遵循下面的消息格式：

长度(4字节)+消息(字节数为前面的长度中4字节确定的整数)

例如如果在下文中出现这样的内容：“发送消息CHOKE”，那么在网络中传输的数据实际上是"00 00 00 01 00"。

所有的消息的第一个字节(不包括上述的表示长度的四字节)都是消息代码，为了后面的协议分析方便，下面列举出BT协议中的消息代码(代码均为一个字节)，在后面的分析中，直接使用消息代码来表示格式：

```
CHOKE = chr(0)
UNCHOKE = chr(1)
INTERESTED = chr(2)
NOT_INTERESTED = chr(3)
# index
HAVE = chr(4)
# index, bitfield
BITFIELD = chr(5)
# index, begin, length
REQUEST = chr(6)
# index, begin, piece
PIECE = chr(7)
# index, begin, piece
CANCEL = chr(8)
# 2-byte port message
```

PORT = chr(9)

4.3 交换块状态位图

握手协议结束后的第一条基本消息，就是双方交换块状态位图，这样就可以对接下来的下载策略提供依据。

这条消息的内容，即表示自己拥有哪些块的bitmap直接使用比特串表示，由于下载时已经知道该种子文件对应的下载内容有多少块，因此这里不再需要指出块数。

总结：在这一阶段两个节点互相发送消息BITFIELD + bitmap

4.4 确认是否对对方感兴趣

互相发送是否对对方感兴趣这一事实，对对方感兴趣的唯一标准是：存在某一块，对方已经有了，而自己还没有。如果没有兴趣的话，不发送任何消息。

总结：在这一阶段两个节点互相发送消息INTERESTED

4.5 根据策略取消应用层阻塞

根据各个客户端的不同策略，决定向和自己有连接的某些PEER上传。

节点之间根据不同的策略选择在不同的时间发送这条消息：UNCHOKE

注：一方向另一方发送取消阻塞消息并不意味着对方也会向自己发送取消阻塞消息。即每个TCP连接的两个方向上的阻塞状态是独立的，例如某个连接很可能在一个方向上处于阻塞状态，但是在另外一个方向上则不处于阻塞状态。所有的客户端自行决定向哪些PEERS发送取消阻塞消息。只有发送了取消阻塞消息后，对方发送的数据请求才会被处理。

每个客户端也可以根据一定的策略来发送阻塞(CHOKE)消息，使某个连接重新进入阻塞状态，即当一个客户端向另一个客户端发送了(CHOKE)消息后，将拒绝为其提供数据，直到它重新决定向对方发送(UNCHOKE)消息。

节点之间根据不同的策略选择在不同的时间发送这条发送消息：CHOKE

INTERESTED和UNCHOKE的关系是前者是后者的必要非充分条件。即如果节点一没有对节点二发送INTERESTED消息，那么节点二肯定不会对节点一发送UNCHOKE消息，而如果节点一对节点二发送INTERESTED消息，节点二不一定对节点一发送UNCHOKE消息。

4.6 数据传输

当一条连接的应用层阻塞已经被取消后，节点就可以发送数据请求消息。该消息格式为块序列号index(取值范围：0 - n-1，其中n是总的块数，为一四字节整型)+块内偏移地址begin(四字节整型)+请求的数据长度length(四字节整型)

总结：节点在这一阶段发送消息REQUEST+index+begin+length

节点在收到数据请求后发送数据内容。该消息的格式为块序列号index+块内偏移地址begin+数据内容piece(字节流)，注意，由于BT协议中每条消息之前均已包含了该消息的长度，因此在这里，消息内容本身的字节流不需要另一个字段来传输它的长度。

总结：收到数据请求后的节点将发送消息：PIECE+index+begin+piece

注：根据协议，如果某个连接的一个方向上被阻塞，那么数据内容将不会在这个方向上被传输，如果强行发送数据请求消息，则该消息将简单得被忽略。

4.7 其它消息

在BT协议中，还有一些其它的消息会被发送，这些消息包括：

have消息：HAVE+index。此消息在peer拥有了一块新的数据块时向其它peers发送，即peers在交换了初始的bitmap后，由于存在下载到其它块的可能性，因此需要使用这条消息通知其它peers，使它们以后可以考虑从该peer处下载此块。

cancel消息：CANCEL+index+begin+length。此消息在peer取消数据请求时发送，可以抵消前面的REQUEST消息的作用。需要发送此消息的情景如下：由于数据请求消息可能同时发送给多个peers，而这些数据请求也不一定很快能够得到回复，如果有一个数据请求因为某种策略同时向多个peer发出，而且已经得到了返回，那么其它收到同样数据请求消息的peers就没有必要重复地发送数据了。因此发送一条cancel消息给它们，可以节省网络带宽。

PORT消息：PORT+pack('!H', port)。DHT专用，其中pack是python中的一种打包格式，它把port按照第一个参数的要求打包。这个参数中，'!'表示大数在前(big-endian)，网络传输格式通常都是如此。'H'表示无符号短整形，即两个字节。由于端口号小于65536，因此使用两个字节的无符号短整形即可表示。这条消息向对方表明本节点的DHT服务(kademlia)开在哪个端口。这个端口有可能和BT协议的服务端口不相同，而且通常BT服务使用TCP端口，而DHT服务使用UDP端口。此消息在连接刚刚建立而且确定双方都支持DHT的时候会被发送(在发送完bitmap之后发送)。

NOT_INTERESTED消息：NOT_INTERESTED。这条消息在下载过程中当对方不存在自己想要下载的东西而之前已经对其发送了INTERESTED消息后将被发送，这种情况通常是下载到了一些新的块然后导致的情况变化。发送此消息后，对方将马上发送CHOKE消息。

4.8 数据连接的识别方法

BT数据连接的协议判断极度明显。因为字符串"BitTorrent protocol"的长度为19，因此可以使用如下方法判断出BT协议：即任何一个TCP连接，如果它的应用层的数据(双向)最开始的20个字节满足BT协议的握手消息格式，即判断该数据连接为TCP连接。之后可以以session为单位，对该TCP连接继续保持观察。

5 节点退出阶段

在BT协议中，节点退出的方式非常简单，即中止TCP连接。

6 关于NAT的处理

BT的官方版本在tracker服务器处会根据配置连接到节点所表现出来的IP和它声称的端口号，对NAT状况进行检测。但是从目前对代码分析来看，这些检测结果仅被作为日志保存下来以提供统计信息。

7 总结

BT协议以种子文件作为其下载的元信息，将要共享的信息从文件列表抽象到连续的块存储空间，以块为单位进行下载。节点通过tracker服务器获取其它的正在下载该种子文件的节点信息，然后自行与它们建立连接。这种连接不区分控制连接和数据连接，连接建立后，节点之间采用一定的应用层阻塞策略，这样可以起到一定的激励作用，使上传速度越快者下载速度也越快。当一个节点想停止和另外一个节点的连接时，它只是简单得断开TCP连接。