实验二

超文本传输协议

# 实验目的

在本次实验中，我们将使用Wireshark来研究HTTP协议，探索几个方面：基本的GET/response交互、HTTP消息格式、检索大型HTML文件、检索带有嵌入对象的HTML文件以及HTTP身份验证和安全性。在开始这些实验之前，请先复习一下课本的第2.2节[[1]](#footnote-1)。

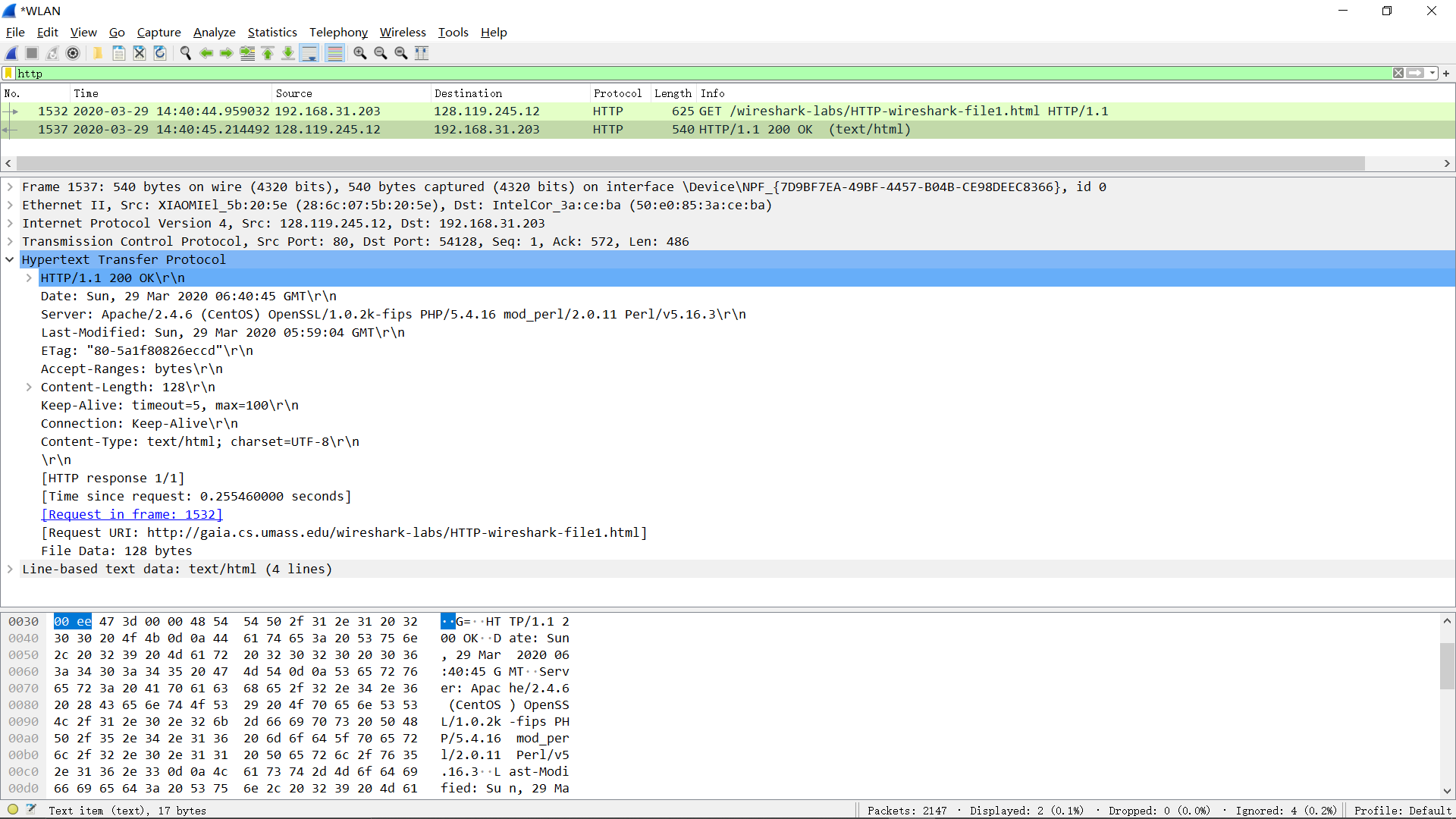
# 实验步骤

## 一、基本的HTTP GET/response交互

让我们从下载一个非常简单的HTML文件开始我们对HTTP的探索——这个文件非常短，并且不包含嵌入的对象

1. 启动网络浏览器
2. 启动Wireshark数据包嗅探器，如介绍性实验室中所述（但不要启动数据包捕获）。在“Apply a display filter”窗口中输入“http”（仅输入字母，不含引号）并**回车**，以便稍后在“Packet list”窗口中仅显示捕获的HTTP消息（这给实验里只对HTTP协议感兴趣，滤除其它捕获内容能够使界面显示内容清晰）
3. 稍等一分钟以上（后面很快会知道原因所在），然后开始Wireshark数据包捕获
4. 输入http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html到浏览器中地址后将显示一个非常简单的单行html文件
5. 停止Wireshark捕获

此时Wireshark窗口应该与图1所示的窗口类似



**图1：**浏览器取得http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html

后Wireshark上的捕获显示

图1中的示例在“Packet list”窗口中显示捕获了两条HTTP消息：GET消息（从浏览器到gaia.cs.umass.edu web服务器）和response消息（从服务器到浏览器）。“Packet detail”窗口显示所选消息的详细信息（此处为HTTP OK消息，这在“Packet list”窗口中突出显示）。回想一下，由于HTTP消息封装在TCP报文段中的，而TCP报文段则封装在IP数据报中，最终IP数据报是在以太网帧中携带的，因此Wireshark也会显示帧、以太网、IP和TCP分组信息。我们希望在界面中尽量少地显示非HTTP数据（这里主要感兴趣的是HTTP协议，本课程的后续实验室中再研究其它协议），因此请选择在帧、以太网、IP和TCP信息最左边的框有一个指向右边的三角（意味着存在隐藏的、未显示的信息），而在HTTP行则有一个向下的三角（意味着详细显示关于HTTP消息的所有信息）。

（注意：应该忽略与favicon.ico的任何HTTP GET和响应。如果看到对该文件的引用，它是你的浏览器自动询问服务器是否有一个小图标文件，以便显示在浏览器中的URL旁边。这里将忽略对这个文件的引用。）。

通过查看HTTP GET和response消息中的信息，回答下面罗列的问题。在回答这些问题时，应该打印出GET和response消息到pdf文件（有关如何执行此操作，请参阅上次实验课的内容），并指出在消息中哪里是下列问题的回答。而在实验报告中，请给输出加上注释，这样就可以清楚地知道输出中获取答案信息的位置（例如，用彩色字体的文本进行注释）。

1. 你的浏览器运行的是HTTP版本1.0还是1.1？服务器运行的是什么版本的HTTP？
2. 你的浏览器可以接受哪些语言（如果有的话）？
3. 你的电脑IP地址是多少？gaia.cs.umass.edu服务器是多少？
4. 从服务器返回到浏览器的状态码是什么？
5. 所浏览的HTML文件上次在服务器上修改时间？
6. 有多少字节的内容被返回到你的浏览器？
7. 通过检查分组内容窗口中的原始数据，你是否看到Packet list窗口中未显示数据的任何首部？如果有的话，列出一个。

在对上述问题5的回答中，你可能惊讶地发现，刚刚浏览的文档是在下载该文档之前一分钟内进行了最后一次修改。因为（对于这个文件），gaia.cs.umass.edu服务器自动将文件上次修改的时间设置为当前时间，并且每分钟执行一次。因此，如果你在两次访问之间等待一分钟，则该文件将显示为最近修改过的文件，因此浏览器将下载该文档的“新”副本。

**二、HTTP条件GET/response交互**

 回顾课本的第2.2.5节，大多数web浏览器都进行对象缓存，因此在提取HTTP对象时执行条件GET。因此在执行以下步骤之前，请先确认浏览器缓存被清空。（在chrome下请选择“History”->“Clear browsing data”，而对于Internet Explorer，请选择“工具”->“Internet选项”->“删除文件”；这些操作将从浏览器缓存中删除缓存的文件。）现在执行以下操作：

* 启动web浏览器，并确认浏览器缓存已清除
* 启动Wireshark分组嗅探器
* 输入http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html这个URL到浏览器地址栏中，会显示出一个非常简单的只有五行的html文件
* 再次在浏览器中快速输入相同的URL（或选择浏览器上的“刷新”按钮）
* 停止Wireshark捕获，并在“Apply a display filter”窗口中输入“http”，以便在“Packet list”窗口中只显示捕获的HTTP消息

回答如下问题：

1. 观察从浏览器到服务器的第一个HTTP GET请求的内容。在HTTP GET中看到一行“IF-MODIFIED-SINCE”了吗？
2. 观察服务器响应的内容。服务器是否显式返回文件的内容？你怎么知道？
3. 现在观察从浏览器到服务器的第二个HTTP GET请求的内容。在HTTP GET中看到 “IF-MODIFIED-SINCE:” 这一行了吗？如果是，那么“IF-MODIFIED-SINCE:”标题后面是什么信息？
4. 服务器返回的第二个HTTP GET响应的HTTP状态码和短语是什么？服务器是否显式返回文件的内容？请解释。

**三、提取长文档**

到目前为止上面的示例中检索的文档都是简短的HTML文件。接下来看看当下载一个很长的HTML文件时会发生什么

* 启动web浏览器，确认浏览器缓存已清除
* 启动Wireshark
* 输入http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html到浏览器中地址栏，浏览器会显示相当长的美国版权法条文
* 停止Wireshark捕获，并在display filter specification窗口中输入“http”，以便只显示捕获的HTTP消息

在“Packet list”窗口中，应该可以观察到HTTP GET消息，然后是对HTTP GET请求的多包TCP响应。这个多包响应需要进行一点解释。回想一下第2.2节（参见课本中的图2.9），HTTP响应消息由状态行、首部行、空行和实体组成。在我们的HTTP GET中，响应中的实体是请求的整个HTML文件。这个HTML文件很长，有4500字节，因为太大不适合放在一个TCP包中。因此，单个HTTP响应消息被TCP分成几部分，每一部分都有包含在单独的TCP报文段中（参见教材书中的图1.24）。Wireshark将每个TCP报文段标示为一个单独的分组，而在Wireshark显示的Info列中被标识为“TCP segment of a reassembled PDU”，这是因为单个HTTP响应分片在多个TCP包中。Wireshark的一些版本使用了continuation短语来表示HTTP消息的整个内容分割在多个TCP报文段中。需要强调的是，在HTTP中是没有“Continuation”消息的！

在实验报告中回答下列问题：

1. 浏览器发送了多少个HTTP GET请求消息？追踪中哪个分组号包含获取版权法条文的GET消息？
2. 跟踪中的哪个分组包编号包含与对HTTP GET请求的响应相关联的状态码和短语？
3. 响应的状态码和短语是什么？
4. 单个HTTP响应及版权法条文需要多少个含有数据的TCP报文段来承载？

**四、包含嵌入对象的HTML文档**

上个实验中我们已经看到Wireshark如何显示所抓取的大HTML文件形成的数据流。接下来的实验则将展示当浏览器下载一个包含嵌入对象的文件时会发生什么，该对象（下面实验中的图像文件）为存储在另一个服务器上的文件。

实验过程进行如下操作：

* 启动web浏览器，并确认浏览器缓存已清空
* 启动Wireshark数据包嗅探器
* 输入http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html这个URL到浏览器的地址栏，浏览器会显示一个包含两幅图片的简短html文件。这两幅图片在基本HTML文件中被引用，也就是说图像本身不包含在HTML文件中，只是图像的URL包含在该HTML文件中。如课本教材中所述，浏览器需要从指定的网站检索这些图片徽标。其中教材出版商的徽标来自网站gaia.cs.umass.edu，而教材第5版的封面图片则存储在caite.cs.umass.edu服务器上（二者是cs.umass.edu中的两个不同的web服务器）。
* 停止Wireshark包捕获，并在display filter specification窗口中输入“http”，以便只显示所抓取的HTTP消息。

实验报告中需要回答下列问题：

1. 浏览器发送了多少个HTTP GET请求消息？-这些GET请求被发送到哪些Internet地址？
2. 浏览器是连续下载这两张图片，还是同时从两个网站下载的？解释一下。

**五、HTTP身份验证**

在最后这个实验里我们将访问一个受密码保护的网站，并检查访问这个网站进行交换 HTTP消息的顺序。下面的URL

http://gaia.cs.umass.edu/wireshark-labs/protected\_pages/http-wireshark-file5.html受密码保护。用户名是“wireshark-students”（不含引号），密码是“network”（同样不含引号）。访问这一受密码保护的所谓“安全”站点，执行如下操作：

* 确认浏览器已清除缓存，关闭浏览器后再次启动浏览器
* 启动Wireshark数据包嗅探器
* 输入http://gaia.cs.umass.edu/wireshark-labs/protected\_pages/HTTP-wireshark-file5.html到浏览器中，在弹出对话框中键入请求的用户名和密码。
* 停止Wireshark数据包捕获，并在“Apply a display filter”窗口中输入“http”，以便稍后在“Packet list”窗口中只显示捕获的HTTP消息

现在检查Wireshark的输出。如果你想先阅读一下HTTP认证，这可以通过查看http://frontier.userland.com/stories/storyReader$2159上的关于“HTTP访问认证框架资料”。

在实验报告中回答下列问题：

1. 服务器对来自浏览器的初始HTTP GET消息的响应（状态码和短语）是什么？
2. 当在浏览器第二次发送HTTP GET消息时，HTTP GET消息中包含了什么新字段？

上面所输入的用户名（wireshark-students）和密码（network）采用字符串（d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=）进行编码，在客户端的HTTP GET消息的“Authorization:Basic”头之后。虽然你的用户名和密码可能看起来是加密的，它们只是以一种称为Base64格式的格式进行了编码。用户名和密码并未加密！要查看这个，请转到http://www.motobit.com/util/base64-decoder-encoder.asp，输入base64编码字符串d2lyzxnoyxjrlxn0dwrlbnz并解码！你已经从Base64编码转换成ASCII编码，因此应该看到你的用户名！要查看密码，请输入字符串Om5ldHdvcms=的其余部分，然后按decode键。因为任何人都可以下载Wireshark之类的工具，并嗅探通过其网络适配器的数据包（不只是他们自己的数据包），任何人都可以将Base64转换为ASCII（你刚刚所做的！），应该清楚除非采取其它措施，否则WWW站点上的简单密码是不安全的。

怕了吗！正如课本第8章中所讲述的，有很多方法可以使对WWW访问变得更加安全。然而，我们显然需要一些超越基本HTTP身份验证框架的东西！

1. 参考图和章节编号均与课本相同 [↑](#footnote-ref-1)