|  |  |  |  |
| --- | --- | --- | --- |
| **《计算机网络》实验报告** | | | |
| **实验编号** | 实验7 | **实验名称** | TCP |
| **姓名** | 杨伟康 | **学号** | 202200201095 |
| **班级** | 网安班 | **成绩** | （空着） |
| 1. **实验目的**   我们来详细看一下UDP（用户数据报协议）。UDP是一种在互联网上广泛使用的传输协议，当不需要可靠性时，它可以作为TCP的替代方案。这部分内容在您的课本的6.4节中有所介绍。在进行本实验之前，请回顾一下该部分内容 | | | |
| 1. **实验要求**   wireshark：使用wireshark软件工具用于捕获和检查数据包跟踪。  wget / curl: 我们主要使用了wget（在Linux和Windows系统上）和curl（在Mac系统上）这两个命令行工具来从网络上获取资源。  Browser: 可以使用任何Web浏览器来查找或获取页面作为工作负载。 | | | |
| 1. **实验内容** 2. Capture a Trace. 3. Inspect the Trace 4. TCP Segment Structure 5. TCP Connection Setup/Teardown 6. TCP Data Transfer | | | |
| 1. **实验过程**   Step 1: Capture a Trace.  要捕获发送适量数据的单个TCP连接的跟踪信息，请按照以下步骤操作；或者，您也可以使用提供的跟踪信息。许多应用程序都使用TCP作为传输层协议，包括网络浏览器。因此，我们将通过执行一个网络下载来测试TCP连接。但请注意，TCP能够同时双向传输数据，但在下载过程中，内容仅从远程服务器发送到本地计算机（在初始请求之后）。  找到一个中等大小资源的URL，并且该资源可以通过HTTP（而非HTTPS）下载。您可以使用浏览器进行搜索，可能是在寻找一张图片（.jpg）或PDF文档（.pdf）。您需要确保这是一个单独的资源，而不是包含许多内联资源的网页（例如，.html）。作为建议，SIGCOMM会议将其技术论文作为PDF文件提供。找到一篇有趣的网络论文的URL！  使用wget或curl获取该URL，以检查您是否能够在至少几个网络时间秒（network time seconds）内检索到至少500KB的内容。    使用提供的跟踪，获得了如下的数据，你现在应该有一个类似于下图所示的短跟踪    Step 2: Inspect the Trace  在您的跟踪中，选择位于中间位置的任意一个长数据包，其协议标记为TCP。在中面板中展开TCP协议部分（使用“+”扩展器或图标）。除了初始的HTTP GET请求和HTTP响应的最后一个数据包外，所有数据包都应被列为TCP。选择一个长数据包可以确保我们正在查看从服务器下载到您的计算机的数据包。在查看协议层时，您应该在TCP块之前看到一个IP块。这是因为TCP段是在IP中传输的。我们在图中展示了展开的TCP块。  您将看到大致如下的字段：  • 首先是源端口，然后是目的端口。这是TCP在IP地址之外添加的寻址信息。由于数据包是由Web服务器发送的，且标准Web服务器端口是80，因此源端口很可能是80。  • 接下来是序列号字段。它给出了字节流中第一个有效负载字节的位置。  • 紧接着是确认字段。它指示反向字节流中最后收到的位置。  • 头部长度字段给出了TCP头部的长度。  • 标志字段包含多个标志位，用于指示TCP段的类型。您可以展开它并查看可能的标志。  • 接下来是校验和，用于检测传输错误。  • 可能还有一个包含各种选项的选项字段。如果您愿意，可以展开此字段进行探索，但我们稍后会更详细地讨论这些选项。  • 最后，可能有一个TCP有效负载，携带正在传输的字节。    除了上述字段外，Wireshark还可能提供其他信息行来帮助您解释数据包。我们仅涵盖了通过网络传输的字段。  Step 3: TCP Segment Structure  为了展示您对TCP的理解，请绘制一张您研究的TCP段的图示。该图示应显示您使用Wireshark可以观察到的TCP头部字段的位置和字节大小。不要细分标志字段或任何选项字段，如果您发现某些TCP字段共享一个字节，则将它们组合在一起。像往常一样，您的图示可以简单地将帧表示为一个长而薄的矩形。尝试不要查看您文本中的TCP段图示；之后检查它以注意并研究任何差异。  为了确定大小，请注意，当您在中间面板中点击一个协议块（块本身，而不是“+”扩展器）时，Wireshark会在下面的面板中突出显示数据包中相应的字节，并在窗口底部显示长度。您也可以使用“长度”列或“帧详细信息”块中显示的总数据包大小。请注意，这种方法不会告诉您子字节位置。  提交：提交您绘制的TCP段图示。在这里插入图片描述  Step 4: TCP Connection Setup/Teardown  **三次握手**  为了观察“三次握手”的实际过程，请查找带有SYN标志的TCP段，这通常位于您的跟踪的开头，以及它之后的数据包。SYN标志在“信息”列中注明。您也可以使用过滤器表达式“tcp.flags.syn==1”来搜索带有SYN标志的数据包。一个“SYN数据包”是三次握手的开始。在这种情况下，它将从您的计算机发送到远程服务器。远程服务器应以设置SYN和ACK标志的TCP段进行回应，或者称为“SYN ACK数据包”。在收到此段后，您的计算机会对其进行确认（ACK），认为连接已建立，并开始发送数据，在这种情况下，数据将是HTTP请求。您的交换应遵循此模式，但如果数据包丢失并需要重传，则可能会略有不同。  在您的跟踪中，绘制一个三次握手的时间序列图，直到并包括您的计算机在连接建立后发送的第一个数据包（HTTP GET请求）。将您的计算机放在图的左侧，远程服务器放在右侧。像往常一样，时间从上到下流动，横跨页面的线条表示段。结果将图表类似。在您的图表中包括以下特征：  • 每个段上的序列号（Sequence Number）和确认号（ACK Number），如果它们存在的话。只有当段设置了ACK标志时，才会携带ACK号。  • 每个段在您的计算机上发送或接收的时间（以毫秒为单位），从0开始计时。  • 到服务器的往返时间（RTT, Round-Trip Time），估计为SYN段和SYN-ACK段之间的时间差。  在这里插入图片描述  连接选项  除了建立连接之外，TCP的SYN数据包还通过“选项”（Options）在两端之间协商参数。每一端通过在其SYN数据包中包含适当的选项来描述其能力（如果有的话）给另一端。通常，两端都必须支持某种行为，以便在数据传输期间使用它。  回答以下问题：  在您的跟踪中，SYN数据包上携带了哪些TCP选项？  常见的选项包括最大段大小（MSS, Maximum Segment Size），用于告知对方可以接收的最大段大小；以及时间戳（Timestamps），用于在段中包含信息以估计往返时间（RTT, Round-Trip Time）。还有诸如NOP（无操作）和选项列表结束符之类的选项，它们用于格式化选项但不宣传能力。在上面的回答中，您不需要包含这些格式化选项。  选项也可以在连接建立后，在常规段上携带，当它们在数据传输中发挥作用时。这取决于选项本身。例如：MSS选项不在每个数据包上携带，因为它不传递新信息；时间戳可能包括在每个数据包上以保持RTT的最新估计；而诸如SACK（选择性确认）之类的选项仅在数据无序接收时才使用。为了娱乐，您可以查看您跟踪中数据包上的选项。  FIN/RST 拆除  最后，在下载完成后，TCP连接会被拆除。这通常是通过FIN（终结）段来完成的。每一方都向另一方发送一个FIN，并对接收到的FIN进行确认；这个过程与三次握手类似。另外，当一端发送RST（重置）包时，连接可能会被突然拆除。这个包不需要另一方的确认。  在您的跟踪中，从第一个FIN或RST包发出开始，直到连接完全拆除，绘制一个拆除过程的图示。与之前一样，在每个段上显示序列号和确认号。如果您有FIN包，请使用时间差来估计往返时间（RTT）。  在这里插入图片描述  Step 5: TCP Data Transfer  TCP连接的中段在我们跟踪中代表的是数据传输或下载，这是主要事件。为了对其有一个整体了解，我们首先将查看随时间变化的下载速率。  在“统计”菜单下，选择“IO图”。默认情况下，此图显示的是随时间变化的包速率。通过以下更改将其调整为显示下载速率。您可能会想用“统计”菜单下的“TCP流图”工具，但这些工具对我们的情况并不适用，因为它们假设跟踪是在发送数据的计算机附近进行的；而我们的跟踪是在接收数据的计算机附近进行的。    • 在x轴上，调整刻度间隔和每刻度的像素数。刻度间隔应足够小，以便能够观察到跟踪中的行为，但又不能太小以至于没有平均效果。对于几秒钟的跟踪，0.1秒是一个不错的选择。每刻度的像素数可以调整，以使图形更宽或更窄，从而填满窗口。  • 在y轴上，将单位更改为位/刻度。默认是包/刻度。通过更改它，我们可以通过取y轴值并适当缩放（例如，对于0.1秒的刻度，乘以10）来轻松计算出每秒的位吞吐量。  • 添加一个过滤器表达式以仅查看下载包。到目前为止，我们一直在查看所有包。假设下载来自常见的Web服务器端口80，您可以使用“tcp.srcport==80”作为过滤器进行筛选。不要忘记按Enter键，并且您可能需要点击“图形”按钮以使其重新显示。  • 要查看上传流量的相应图形，请在下一个框中输入第二个过滤器。同样假设是常见的Web服务器端口，过滤器为“tcp.dstport==80”。按Enter键并点击“图形”按钮后，您应该在图形上看到两条线。  此过程的图形如下所示。从图中我们可以看到，示例下载速率迅速从零增加到稳定速率，带有一点指数增长的特性    **当TCP连接运行良好时，下载方向的大致数据包速率（packets/second）和比特速率（bits/second）是多少？**  回答：要确定下载方向的数据包速率和比特速率，您需要查看网络抓包工具中的相关数据。数据包速率通常可以通过观察单位时间内通过连接的数据包数量来计算。比特速率则是这些数据包中实际传输的数据量（以比特为单位）除以时间。在TCP连接稳定后，这些数据将呈现出一个相对稳定的值。  **下载速率中内容所占的百分比是多少？请展示您的计算过程。为了找到答案，请查看一个典型的下载数据包；通常会有许多类似的大下载数据包。您可以看到它的长度以及它包含的TCP有效载荷的字节数。**  回答：要计算下载速率中内容所占的百分比，首先需要确定一个典型下载数据包中TCP有效载荷的字节数。然后，将该字节数与数据包总长度（包括TCP头部、IP头部等）进行比较。通常，TCP有效载荷占据了数据包的大部分，但具体比例取决于数据包的具体内容和大小。通过计算有效载荷占总数据包的百分比，并假设这一比例在整个下载过程中保持不变，可以估算出下载速率中内容所占的百分比。然而，请注意，由于TCP和IP头部的存在，内容所占的百分比永远不会达到100%。  **由于ACK包的存在，上传方向的大致数据包速率（packets/second）和比特速率（bits/second）是多少？**  回答：在TCP连接中，每当一方成功接收到数据包时，它都会发送一个ACK（确认）包作为响应。因此，上传方向的数据包速率和比特速率将受到ACK包数量的影响。为了计算这些速率，您需要观察网络抓包工具中上传方向的数据包，并识别出哪些是ACK包。然后，您可以通过计算单位时间内通过连接的ACK包数量来得到数据包速率，并将这些ACK包中的TCP头部（以及可能的IP头部）的总字节数除以时间来得到比特速率。请注意，由于ACK包通常比下载数据包小得多，因此上传方向的比特速率可能会显著低于下载方向。  **如果最近从服务器接收到的TCP段的序列号为X，那么下一个传输的TCP段将携带什么ACK号？**  除了携带数据的常规TCP段之外，您可能还会看到各种其他情况。您可以按“信息”列对跟踪进行排序，并浏览类型为“[TCP xxx ...”的数据包。根据下载的不同，您可能会看到重复ACK、乱序数据、重传、零窗口、窗口更新等。这些段通常不会通过TCP头部中的标志（如SYN或FIN段）来区分，而是用于命名在传输过程中可能发生并需要处理的情况。  Explore on your own  我们鼓励您在完成这个实验后自行探索TCP。以下是一些想法：  • 探索TCP的拥塞控制及其经典的AIMD（加性增乘性减）行为。为此，您可能希望在TCP连接上发送（而非接收）适量数据时捕获跟踪信息。然后，您可以使用“TCP流图”工具以及其他分析工具来观察拥塞窗口随时间的变化情况。  • 更深入地探索TCP的可靠性机制。捕获包含数据包丢失的连接跟踪信息。观察是什么触发了重传以及何时触发。同时查看往返时间估算器。  • 查看包括SACK（选择性确认）在内的选项的使用情况，以了解详细信息。在数据包丢失期间，您应该能看到关于已接收字节范围的信息。  • TCP是Web底层的传输层协议。您可以通过设置并发连接来查看浏览器如何利用TCP。 这些建议为深入理解和探索TCP提供了丰富的方向。确实，通过实际操作和数据分析，可以更好地掌握TCP的工作机制和特性。   1. 探索TCP的拥塞控制是理解网络性能优化的关键。通过捕获发送数据时的跟踪信息，并使用分析工具观察拥塞窗口的变化，可以帮助我们了解网络在拥塞时如何动态调整发送速率，以保证网络的稳定性和效率。 2. TCP的可靠性机制确保了数据的完整性和正确性。通过捕获包含数据包丢失的连接跟踪信息，并分析重传触发条件和往返时间估算器，可以深入理解TCP在应对网络问题时的应对策略和效果。 3. SACK选项的使用为TCP在数据包丢失时提供了更精细的控制和确认机制。通过分析SACK信息，我们可以更准确地了解哪些数据已经成功接收，哪些数据需要重传，从而优化数据传输效率。 4. 了解浏览器如何利用TCP建立并发连接是理解Web性能的重要一环。通过观察和分析浏览器在加载网页时建立的TCP连接，我们可以了解并发连接如何帮助提高网页加载速度和用户体验。 | | | |
| **四．实验心得**  学习TCP后，我深刻体会到TCP作为互联网核心协议之一的重要性。它不仅确保了数据的可靠传输，还通过复杂的拥塞控制和流量控制机制维护了网络的稳定性和公平性。TCP的三次握手和四次挥手过程展示了其严谨的连接管理机制，确保数据在传输前后的准确性和安全性。同时，TCP的窗口机制和拥塞控制算法也让我对网络性能优化有了更深入的理解。整体而言，TCP的学习让我对网络通信的底层机制有了更加全面和深入的认识。 | | | |