**计算机网络实验1报告**

学号 2017K8009929059

姓名 於修远

互联网协议实验

一、实验内容

* 使用wireshark软件，在mininet环境中观察结点h1执行wget www.baidu.com命令下载页面时，在网站上产生的各种包。
* 调研说明wireshark抓取的ARP，DNS，TCP，HTTP等协议。
* 解释h1下载baidu页面的整个过程中，不同协议的运行机制。

二、实验流程

（一）必要程序安装

1、 重新编译linux-4.9内核，并设置为boot的默认选项。

2、 利用aptitude工具下载安装xterm和mininet和wireshark。

（二）网络环境创建

1、 使用`sudo mn --nat`指令创建mininet实验环境。

2、 在mninet环境中执行`xterm h1`，调出h1的shell。

3、 执行`echo “nameserver 1.2.4.8”>/etc/resolv.comf`维护环境配置。

（三）命令执行

1、 执行`wireshark &`命令调出wireshark的gui，并选中h1对应网络，开始监控。

2、 在mninet环境中执行命令`wget www.baidu.com`，观察wireshark抓取网络包的输出情况。

三、结果分析

（一）各种网络协议的功能与联系

1. ARP协议

地址解析协议，实现从IP地址到MAC地址的映射，属于链路层协议。主机和交换机间通过广播查询，单播回复，获取映射关系。

1. DNS协议

域名解析协议，实现从域名到IP地址的映射，属于应用层协议。主机首先查询自身的DNS缓存表，若无结果则向上级域名服务器查询，依次包括本地域名服务器、权限域名服务器、顶级域名服务器和根域名服务器。一旦查询成功，再向下回复查询结果。

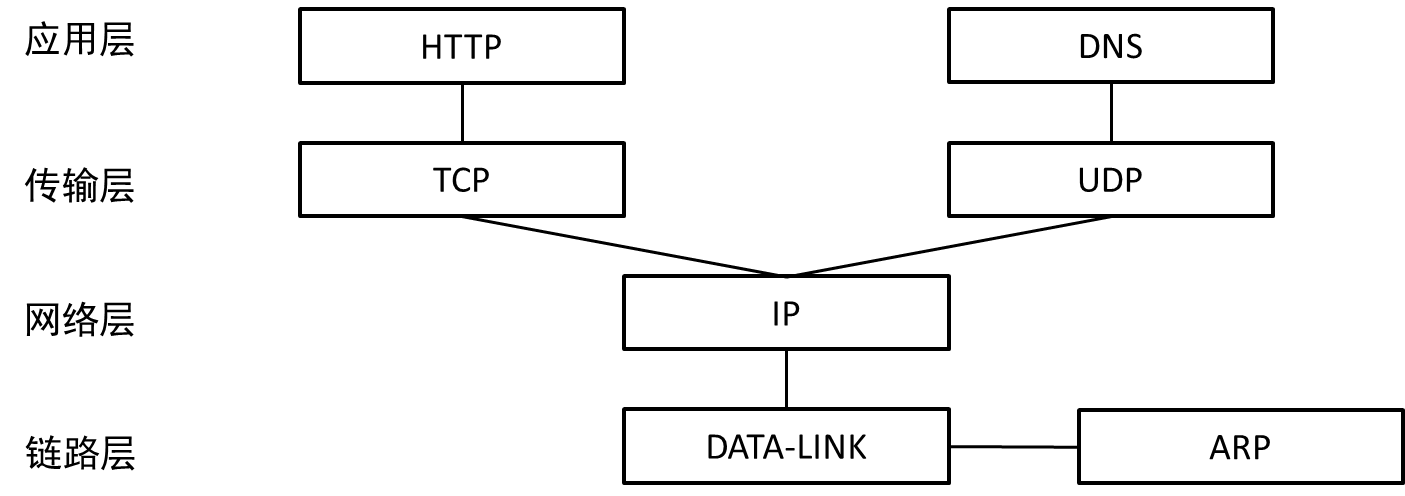
1. TCP协议

传输控制协议，实现从IP地址到MAC地址的映射，属于传输层协议。TCP与处于网络层的IP协议一起工作。TCP传输的最大特征是建立连接需要3次握手、解除连接需要4次挥手。TCP还包含了慢启动、拥塞窗口等机制

1. HTTP协议

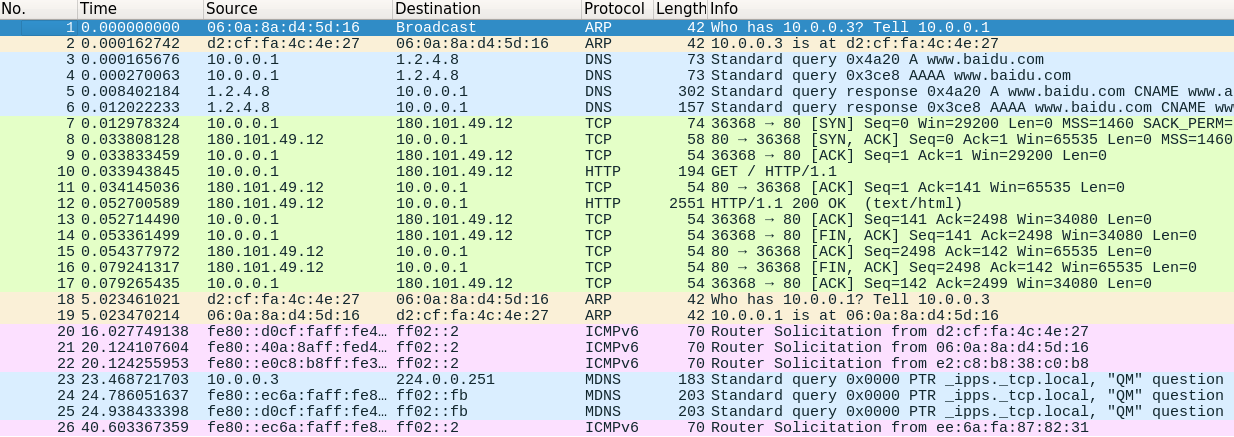
超文本传输协议，从www服务器传输超文本到本地浏览器，属于应用层协议。HTTP协议基于TCP协议，由请求和响应构成，具有多次请求和无状态等特点。

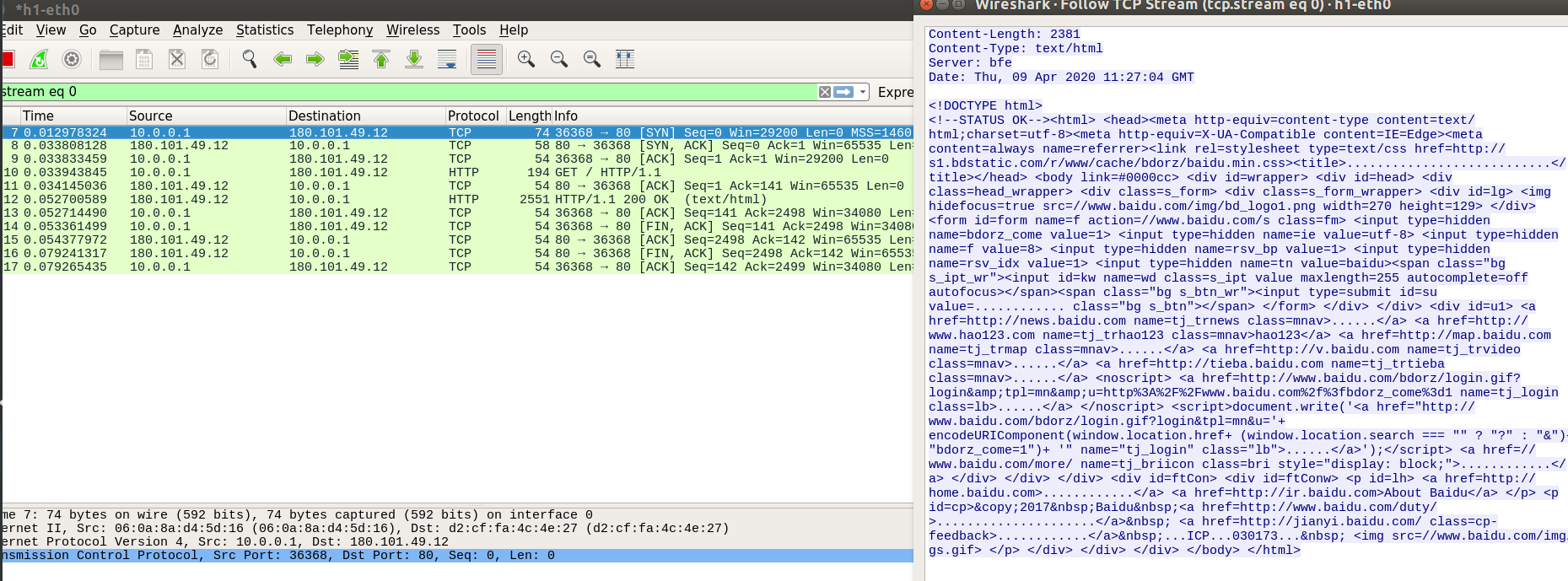
1. 协议间关系

****

（二）实验过程中各协议运行机制

1. 抓包结果





1. 下载过程
2. h1通过ARP协议获得交换机的mac地址。如上图所示，第一个ARP包为h1广播的信号，目的是让拥有10.0.0.3条目的交换机回复其mac地址。第二个ARP包即为该交换机对h1单播的回复信号。
3. h1通过DNS协议获得域名www.baidu.com的IP地址。如上图所示，前两个DNS包（编号3和4）向自身对应的域名服务器（启动时设置为1.2.4.8）发出域名查找，后两个DNS包（编号5和6）是域名服务器回复的IP地址查询结果。
4. h1通过TCP协议与baidu服务器建立连接。首先发送含seq=0的SYN包（编号7，第一个TCP包）到服务器，并进入SYN-SEND状态，收到服务器回复的含ack=1且seq=0的SYN+ACK包（编号8，第二个TCP包）后，对应返回一个含ack=1的ACK包（编号9，第三个TCP包）。至此三次握手全部完成。
5. h1通过HTTP协议进行数据传输。h1向服务器发送HTTP包（编号10，第一个HTTP包）进行下载请求，服务器收到后向h1发出ACK包（编号11，第四个TCP包）进行通道连接，随后发出包含下载内容的HTTP包（编号12，第二个HTTP包）。h1在接收完成后发出ACK包（编号13，第五个TCP包）断开本次传输通道。
6. h1通过TCP协议与baidu服务器解除连接。首先向服务器发送含seq=141的FIN+ACK包（编号14，第六个TCP包），并进入FIN\_WAIT\_1状态。收到该包后，服务器首先发送一个含ack=142的ACK包确认（编号15，第七个TCP包），再发送一个含seq=2498的FIN+ACK包（编号16，第八个TCP包），来关闭数据传输。收到这两个挥手包后h1发送含ack=2499的ACK包（编号17，最后一个TCP包）确认。至此四次挥手全部完成，网页下载从建立到传输到中断的全过程也就此结束。

（三）ARP与DNS机制

1. ARP缓存机制

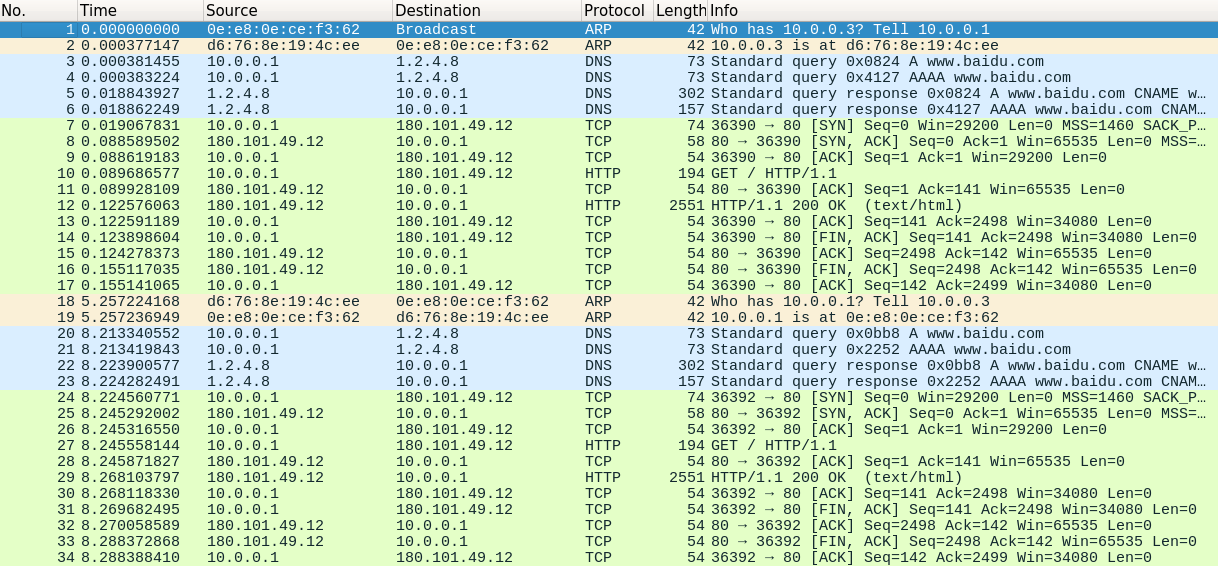
为了提高性能，主机和交换机上一般都会维护一个ARP缓存表，来保存IP地址到MAC地址的映射。这一缓存表会定期更新，保留近期常用的通讯地址；每条条目也都会有一定的生命周期。在自身的缓存表中缺少需要通信的条目时，通过广播得到持有此条目映射的主机或交换机的回复。

1. DNS缓存机制

与ARP缓存机制类似，主机或交换机维护一个DNS缓存表，来保存域名到IP地址的映射。当所需条目缺失时，机器会向一个设定的上级域名服务器发出请求，以获取映射关系。但是，虽然Windows系统中包含有DNS缓存，许多Linux发行版并不具有，所以域名解析只能靠收发DNS包来完成。

1. 缓存机制验证

如下图所示，在很短的时间间隔内连续执行多次`wget www.baidu.com`命令，并观察收发包情况即可。如下图所示，第一次执行页面下载时，h1既收发了ARP包也收发了DNS包，但第二次则只收发了DNS包（编号为18和19的ARP包不是wget后进行的收发包，从内容上也可以看出这是由10.0.0.3发送的确认包）。重复实验多次，可以发现从第二次起每次都要重新进行DNS包的收发，但无需进行ARP包收发。这就说明了ARP存在缓存机制，但本机版本的ubuntu不具有DNS缓存。



另一方面，在h1的shell中执行`ip neigh`，可以看到d6:768e:19:4c:ee的状态为STABLE，即需要发送验证报文。上述编号为18的ARP包也印证了这一点。

流完成时间实验

一、实验内容

* 使用fct\_exp.py脚本复现课件中流完成时间变化曲线图。
* 调研解释图中现象。

二、实验流程

（一）网络环境创建

1、 执行`sudo python fct\_exp.py`以创建设定带宽和延迟等参数的mininet实验环境。

2、 在mninet环境中执行`xterm h1 h2`，调出h1和h2两个结点的shell。

（二）命令执行

1、 在h2的shell中执行命令`dd if=/dev/zero of=1MB.dat bs=1M count=1`，生成等待下载的1MB大小文件。改变文件名和文件大小，同样生成10MB和100MB的文件。

2、 在h1的shell中执行命令`wget http://10.0.0.2/1MB.dat`，并更改文件名，尝试下载3种不同大小的文件，每种至少5次。同时记录下载耗时。

3、 更改脚本中带宽的设置（10MBps~1GBps），重复上述步骤。

（三）数据处理和计算

1、 汇总三种不同大小的文件在五种不同带宽下的下载耗时（流完成时间），并计算各条件下的均值。

2、 根据不同的文件大小计算在平均流完成时间下文件传输的平均速率。

3、 对每种文件大小，以带宽为10MBps情况下的平均速率为基准，对其余带宽下的平均速率进行规格化处理。

4、 以对数尺度横纵坐标作出流完成时间图。

三、结果分析

（一）数据处理

1. 原始数据记录表

如下三表所示，分别为延时为10ms条件下不同大小文件每次的流完成时间。







1. 数据处理与规格化

根据FCT均值计算出下载速率的均值，如下表所示。

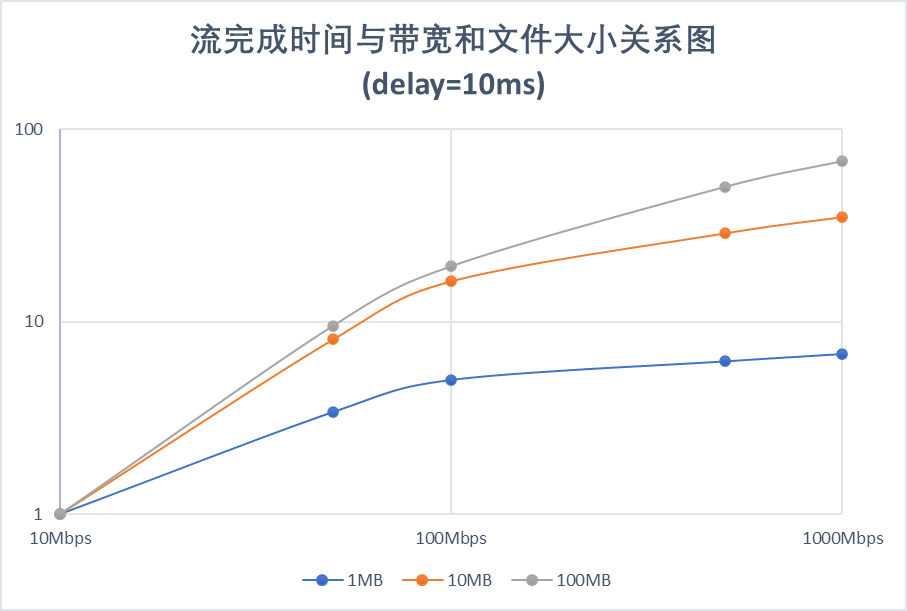


取每列首项进行规格化，得到如下表的FCT提升率。



1. 图像绘制

取对数坐标轴，得到如图的曲线图。其中横坐标为带宽，单位为Mbps；纵坐标为下载加速比。



（二）现象分析

从图像中可以看到，对于一定大小的文件，增加带宽可以提高传输的平均速率，缩短平均流完成时间。但是这一增幅并不是随带宽线性变化的。随着带宽增大，平均速率的增加也会逐步放缓。

究其原因，是TCP传输存在慢启动机制：通过拥塞窗口来限制发送方的数据传输量，避免在大规模数据的突发传输引起网络瘫痪。这也与下载过程中观察到的现象相符：尤以100MB数据下载时最为明显，瞬时下载速率有一个缓慢增长的过程，从起初的几百KB，经过一定的时间后才增长到MB的数量级。这也很好地解释了为什么图像中100MB数据的加速比较10MB数据而言增长不明显，慢启动导致100MB数据下载的开始阶段被迫被拖到和10MB数据相近的传输率。

但是，拥塞窗口的大小限制了发送流量的上限，同时慢启动机制本身也拖慢了达到最大传输速率的时间，所以传输速率无法获得对应于带宽的线性增长，最终趋于平缓。