互联网系统初识实验报告

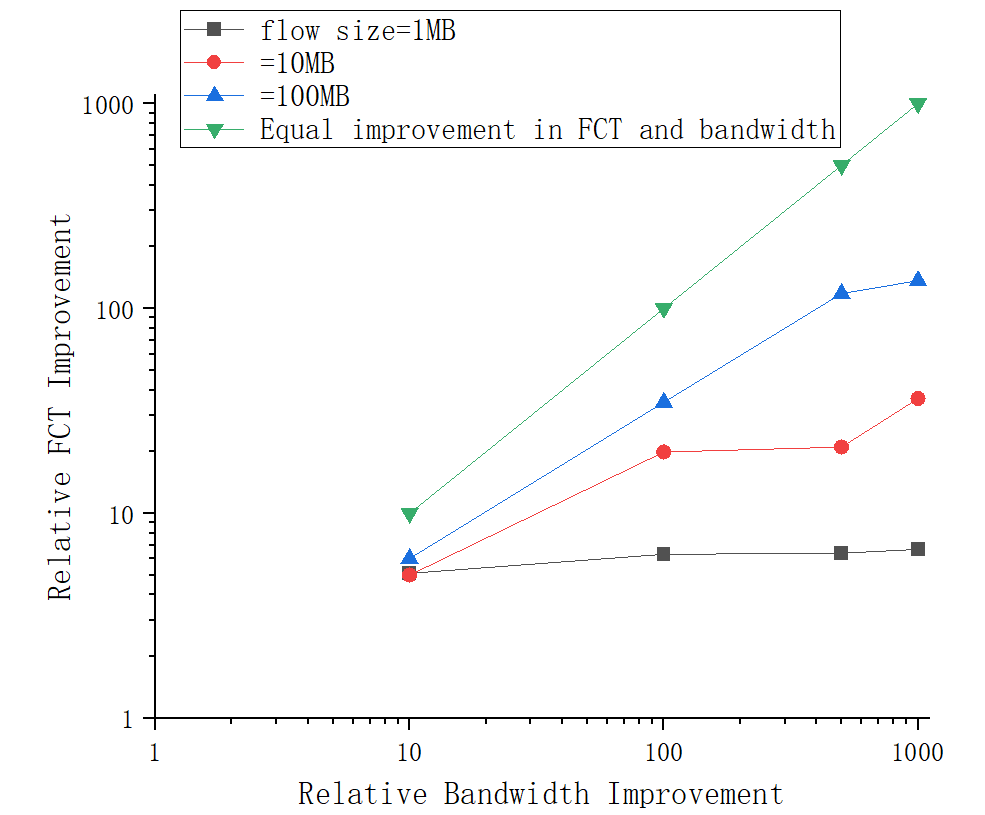
唐宇菲 2018K8009909006

[实验题目]

互联网系统初识

[实验内容]

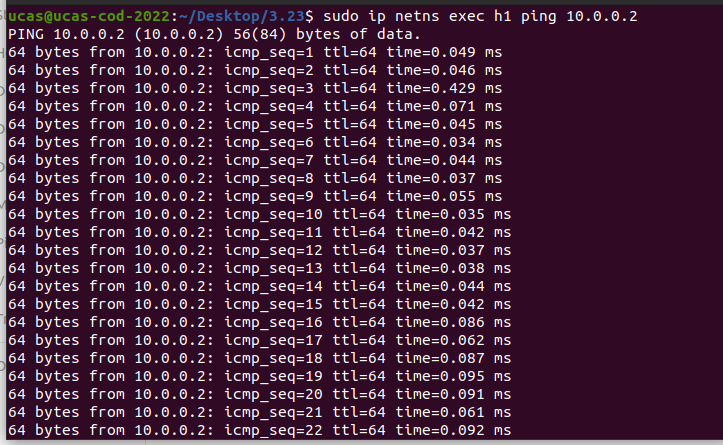
1. 互联网协议实验
2. 流完成时间实验（图像复现如下）



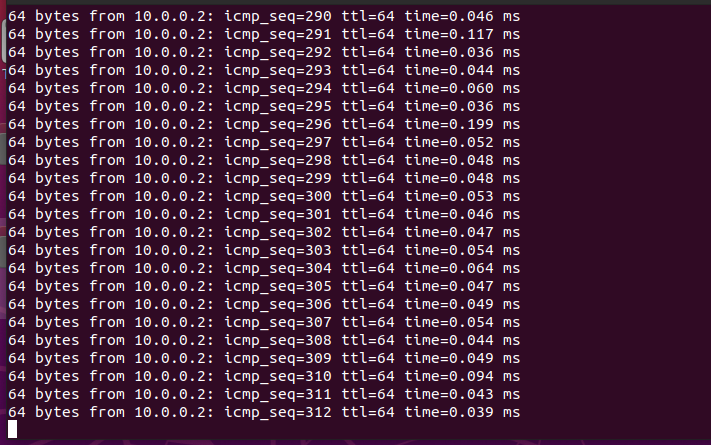
[实验流程]

1. 首先是Mininet的安装。

这里没有出现问题，前几页ppt的内容都敲了一遍以后发现和不敲结果是一样的。（指把《基于网络命名空间的网络环境搭建》这一页命令行又敲了一遍，实际和Mininet nm中定义的一致，**不需要再输入**，发现在执行最后一个测试命令的时候，经过了很久，最后直接按Crtl+C终止。查了一下icmp\_seq:是当前ICMP报文的序号，表示当前是第几个ICMP报文。ICMP(Internet Control Message Protocol)是Internet控制报文协议，是TCP/IP协议簇的一个子协议，用于在IP主机和路由器之间传递控制消息。

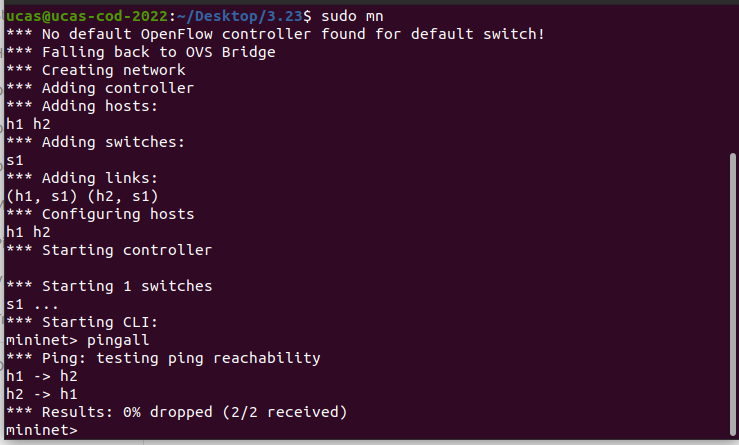


很长的序列，一直没有终止：查找了一下Windows默认ping四次，序列连续表示中间没有丢包。



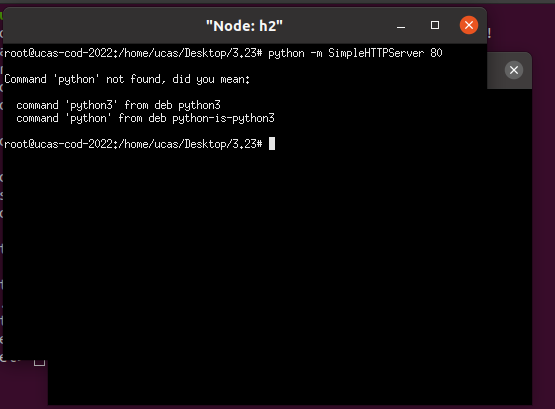
1. Mininet的安装

过程顺利，就是普通的linux下安装软件操作。

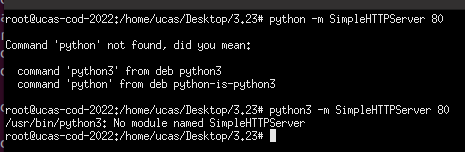




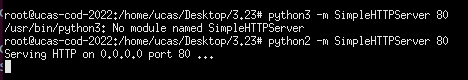
1. Mininet CLI举例



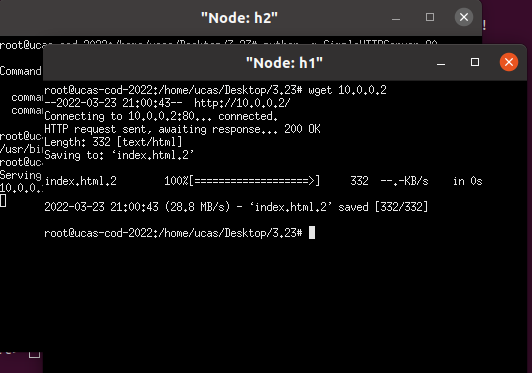
这一步骤里发现的问题，并且这个问题会反复出现，报文里容许的是python2环境，python的命令失效。



实际上改成python3也不能解决问题，正确的修改是将python改成python2。这一步的操作是在节点h2打开了一个终端，并且为这个节点设置了一个服务器。



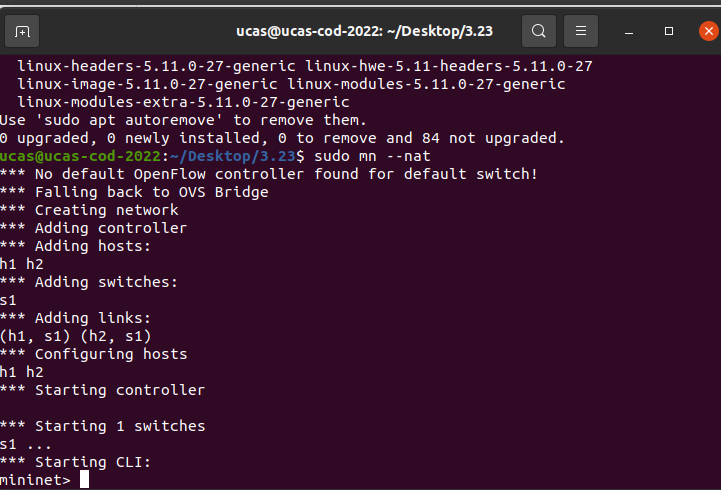
表明节点h2成功，最终效果如下：



前面是一些准备工作，下面进入第一个实验:

**互联网协议实验**

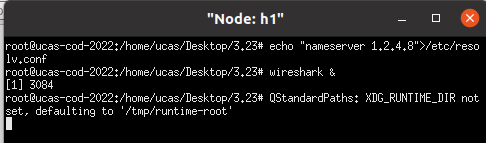
1. 实验环境搭建



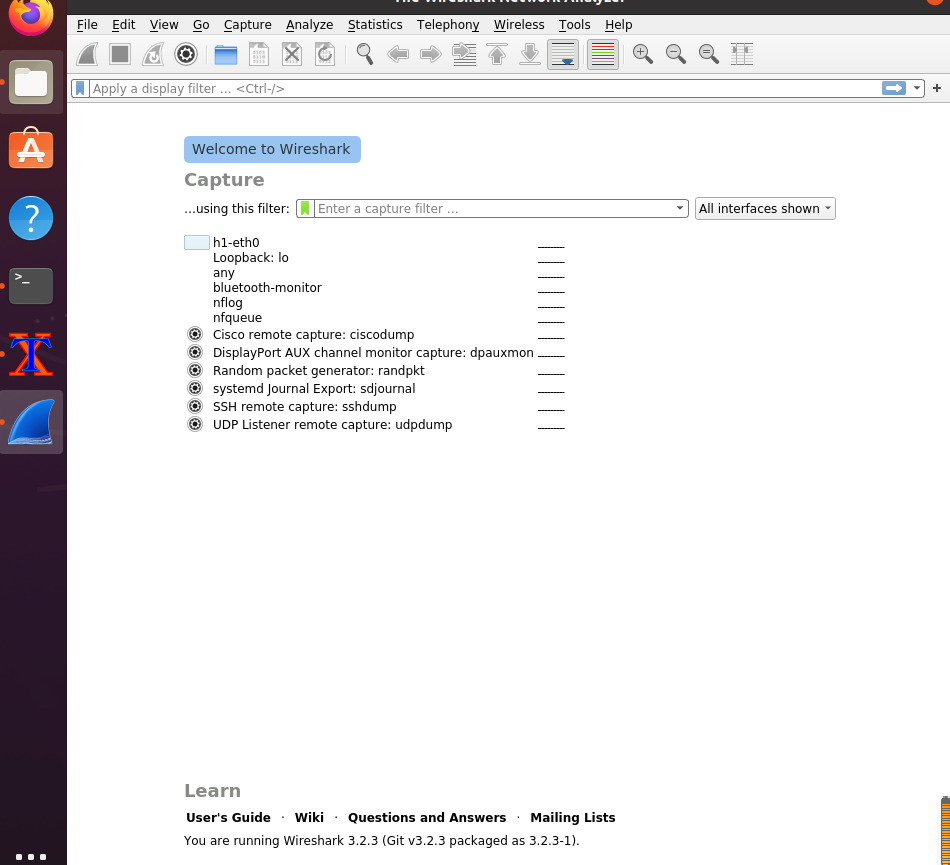
前两步没出什么问题，主要在于sudo mn --nat命令在输入的时候，前几次总是直接结束，没有mininet>这样的待输入行出现，

修正的依据：参考了《01-实验中可能会遇到的问题》7，**安装了软件包ifupdown之后问题解决。**

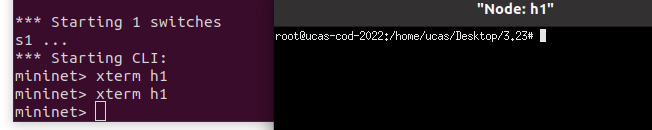
下面是关于节点h1，同时之前遇到了xterm没有弹窗的结果，在后面的ppt中有解决，需要自己手动安装。



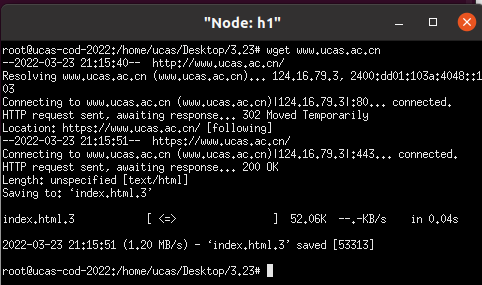
输入这两个命令之后，得到了一个正确的跳转页面。问题在于，此时节点h1的光标似乎不能和上两个命令一样接收新的命令，解决方案：h1节点再打开一个终端，也就是**h1节点开两个终端**。



重新输入一遍xterm h1



新终端中输入,得到OK标志，出现一个index.html的文件，因为操作次数多，所以重复命名。



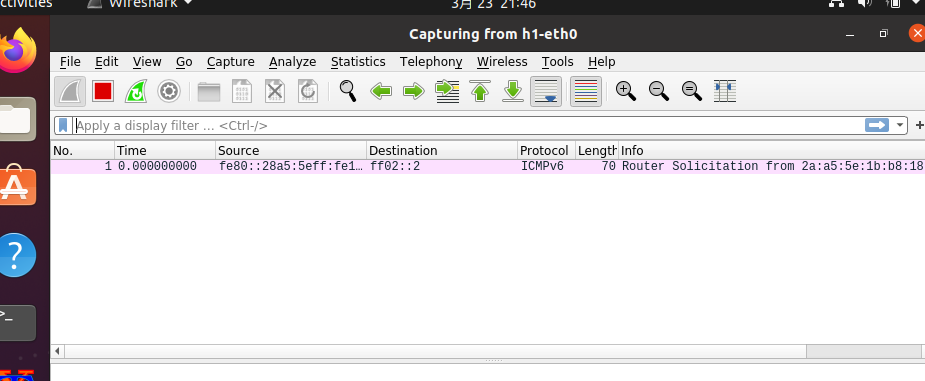
出现html文件，点开后在火狐浏览器出现未经过布局，也显示不出图片的ucas主页的网页展示。

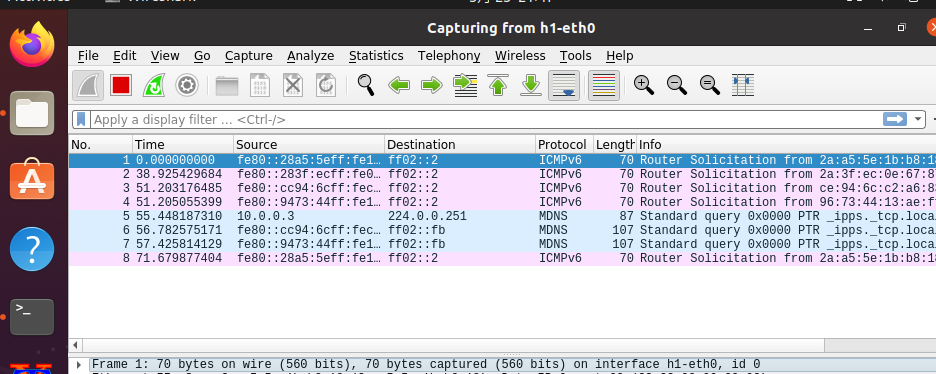


中间需要等待了很久才出现14条记录，直觉上不对。

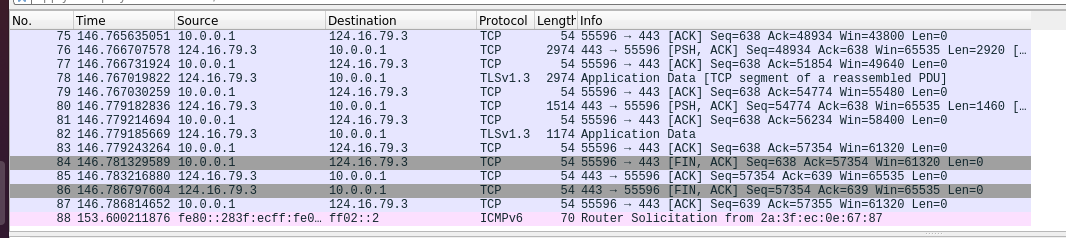
后来修正，实际上发包时间很快，**所以在输入wireshark &命令之后，先点进start\_capture，此时应该是个空闲页面**。

输入wget命令前：





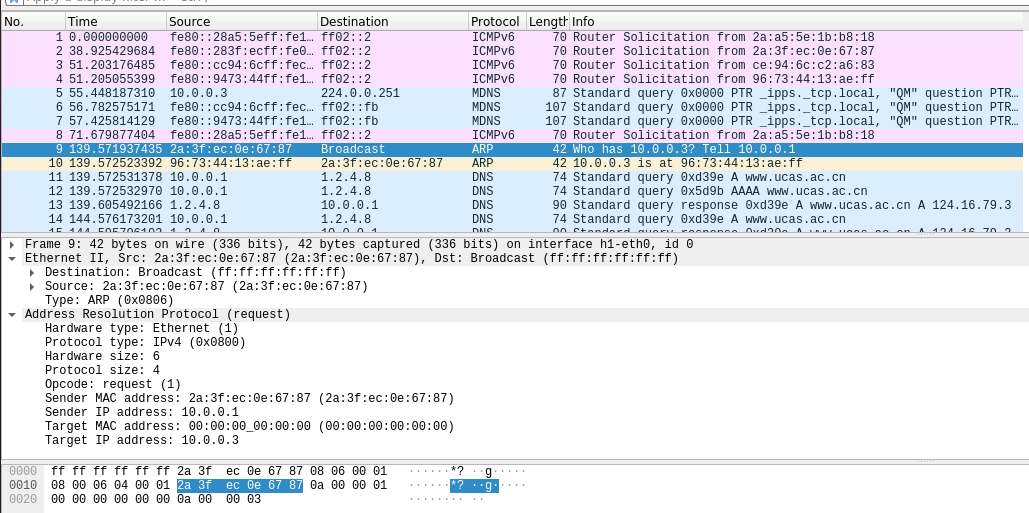
这些抓包不是我们想要的，输入wget 校园网网址之后，等待一瞬间的发包完成后，获得正确的抓包信息。



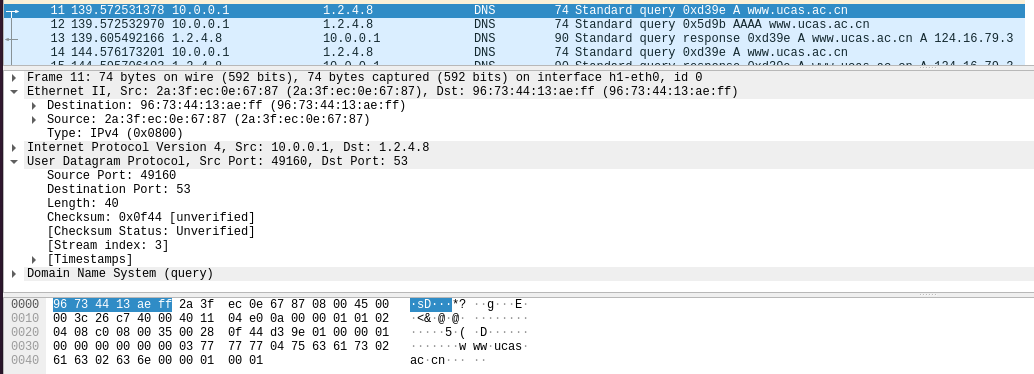
出现了87条左右的信息。

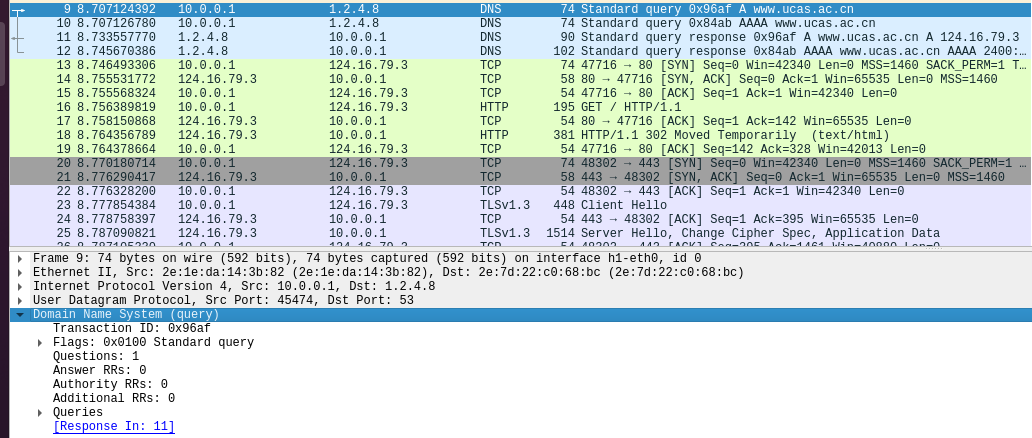
分类获得区别如下:

ARP协议:

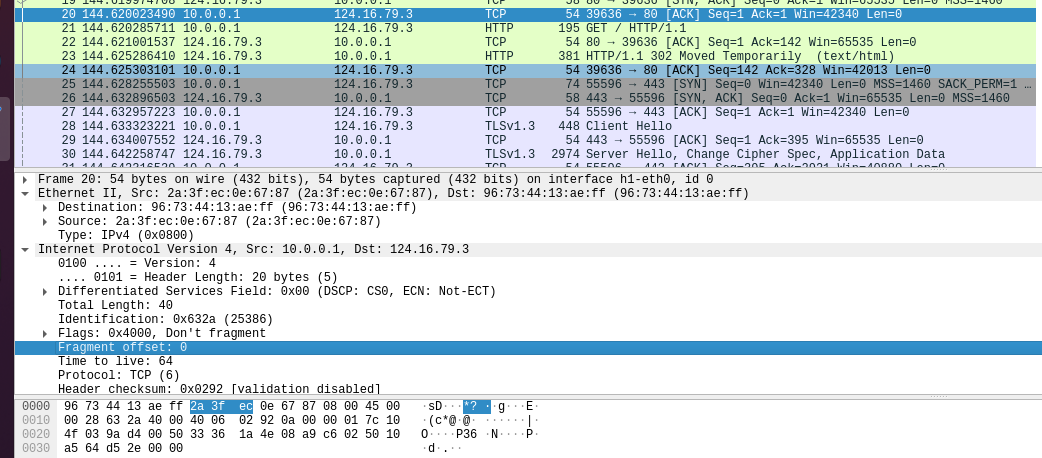


DNS协议:



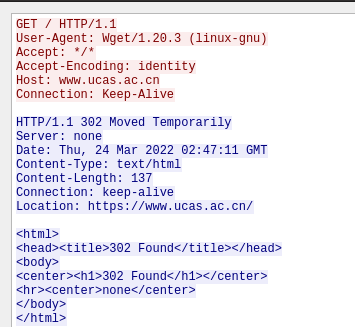


TCP协议:



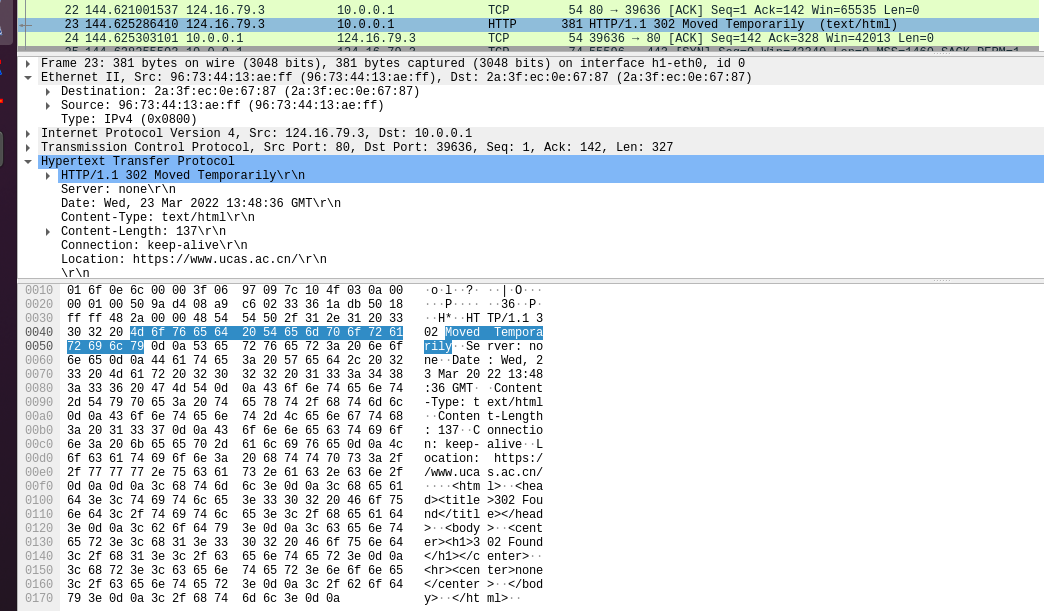
右键点击follow TCP stream

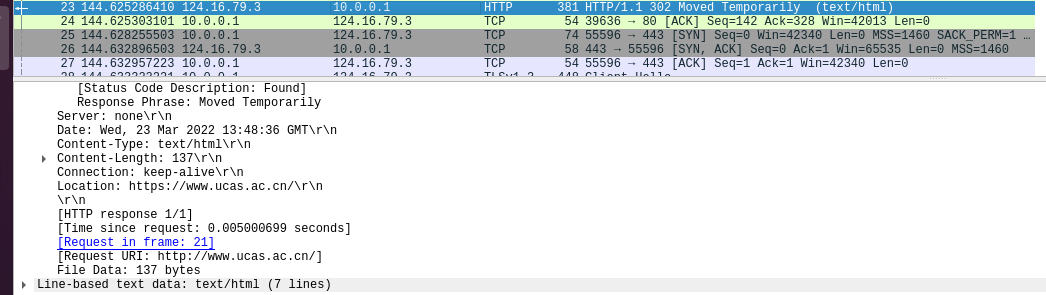




观察到标注了一些前端的信息,<html></html>,<body></body>等标记符号。

HTTP协议:

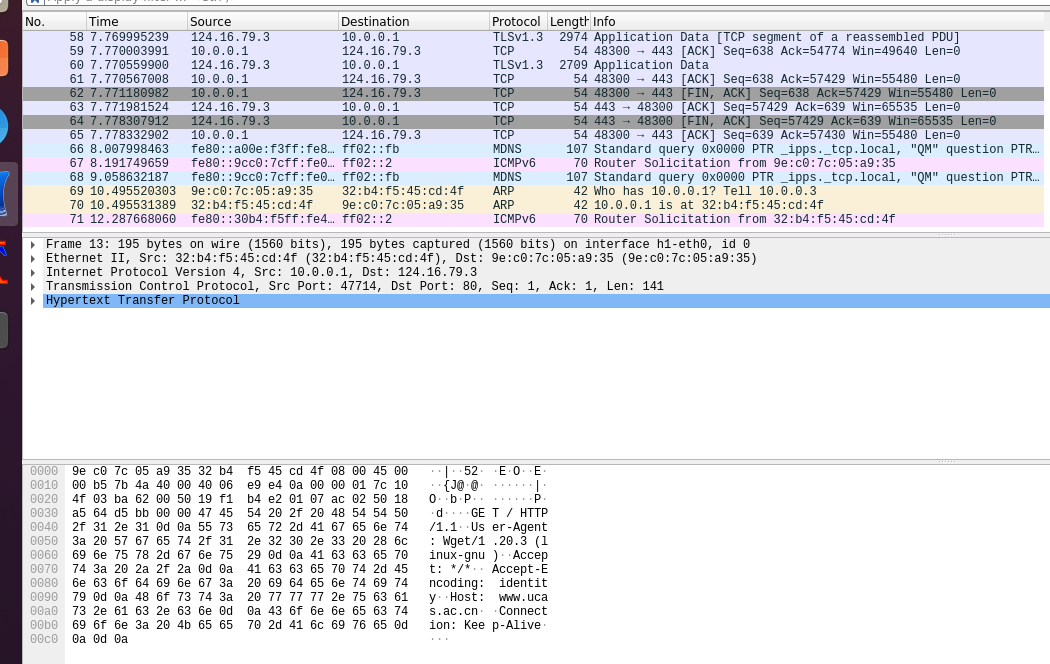




HTTPS协议:

这里没有明显区分出HTTP协议和HTTPS协议，事实上，看到的

Protocol栏均显示HTTP协议。之前由于打开wireshark和输入wget命令的间隔时间略长，中间有部分冗余信息，重新抓包后，得到:大约70条信息。

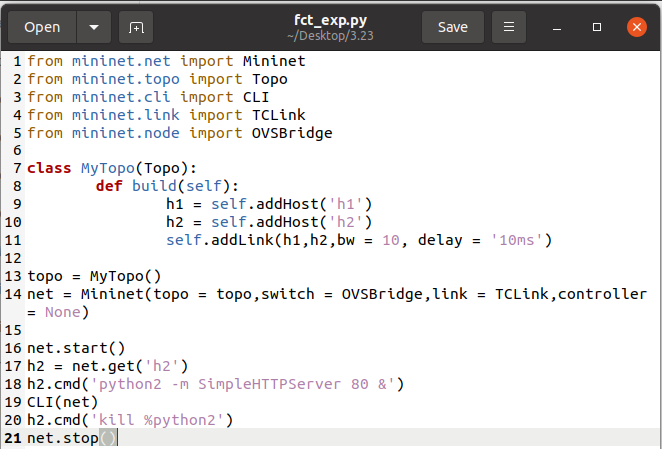


流完成时间实验

1. 实验环境搭建

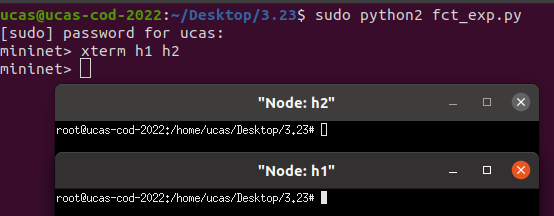
]BOXM0)L55WW0A[80K`$E1O

这里仍然需要把python改成python2,同时后续仍出现问题，需要把fct\_exp.py里面所有涉及python的地方都替换成python2。



中间遇到两个问题:(1)关于python的缩进问题，一开始没有严格关注缩进，导致13-21行均与第8行对齐，导致MyTopo没有定义，缩进表示不在class类里面，在class类之外。

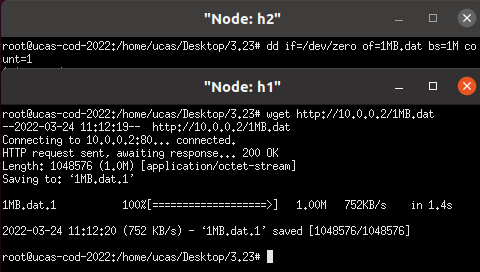
(2)18行和20行均要修改为python2。



出现1MB大小的.dat文件：



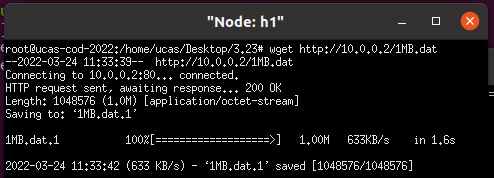
实验结果如下：



现在在给定带宽、延迟和文件大小的前提下，得到流完成时间：

（1）1MB

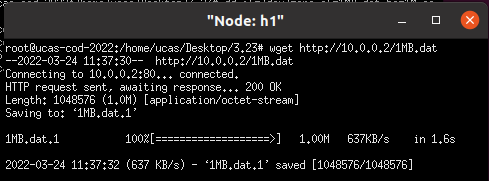
(i)10Mbps



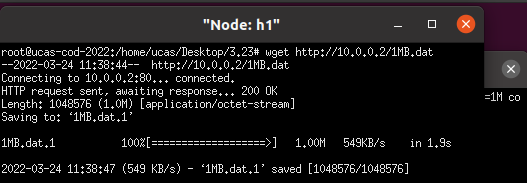
第一次:633KB/s 1.6s

![RIYN]())~P0RB67{5@R(R[Q](data:image/png;base64,)

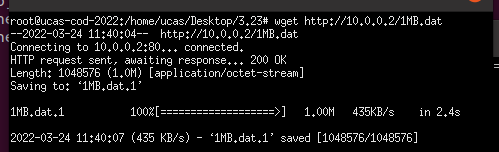
第二次:643KB/s 1.6s



第三次:637KB/s 1.6s

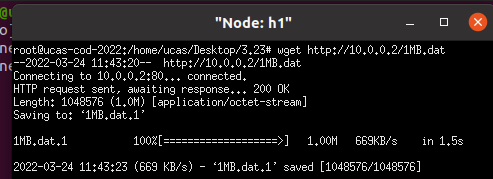


第四次:549KB/s 1.9s

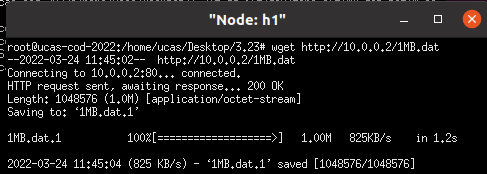


第五次：435KB/s 2.4s

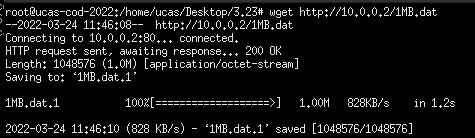
(ii)100Mbps



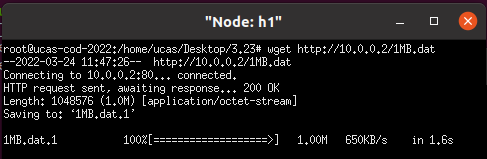
第一次:669KB/s 1.5s



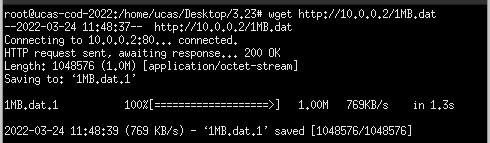
第二次:825KB/s 1.2s



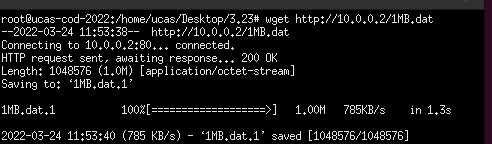
第三次:828KB/s 1.2s



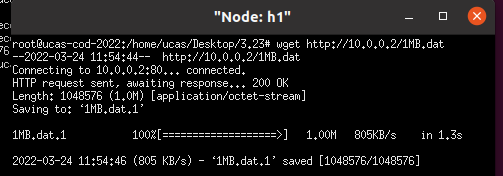
第四次:650KB/s 1.6s



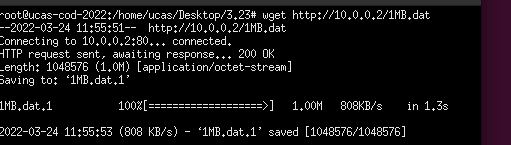
第五次:769KB/s 1.3s

(iii)500Mbps

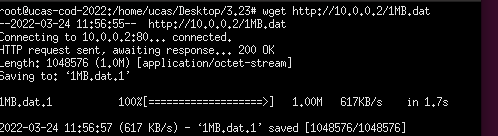
第一次:785KB/s 1.3s



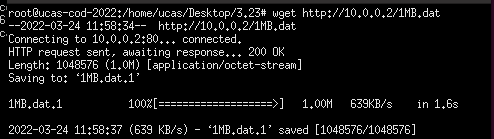
第二次:805KB/s 1.3s



第三次：808KB/s 1.3s

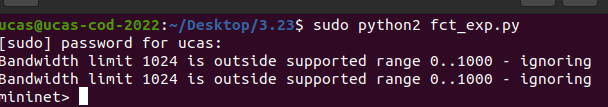


第四次:617KB/s 1.7s

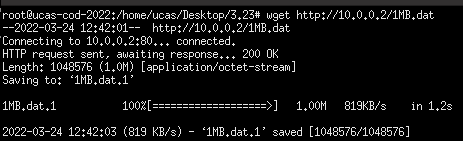


第五次:639KB/s 1.6s

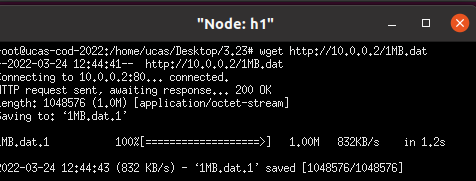
(iv)1Gbps



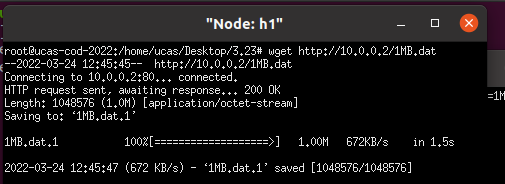
改为1000Mbps，得到



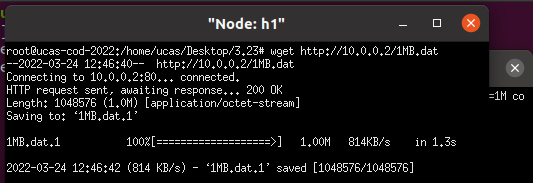
第一次：819KB/s 1.2s



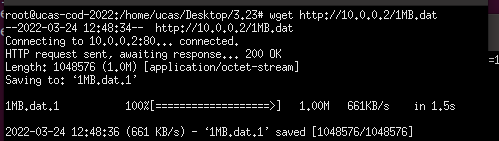
第二次:832KB/s 1.2s



第三次:672KB/s 1.5s



第四次:814KB/s 1.3s

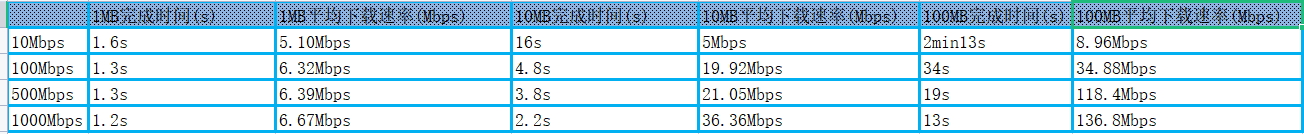


第五次:661KB/s 1.5s

五次采样取平均，

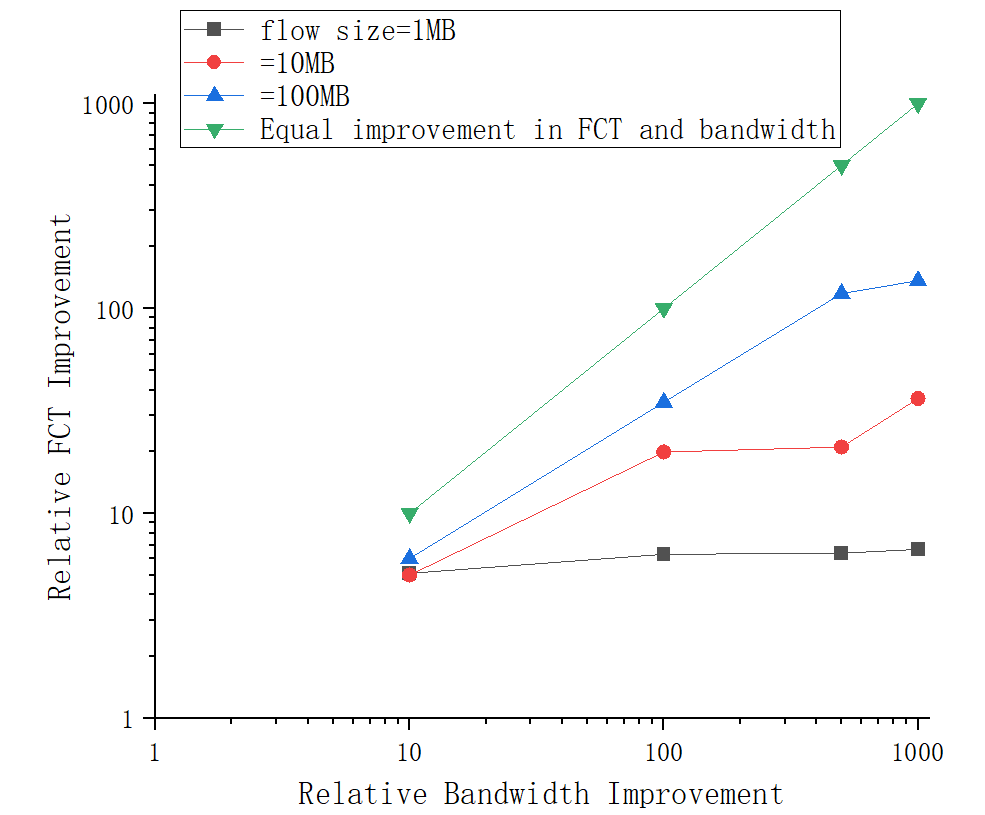
(2)10MB与100MB情况类似，不再赘述。dd命令后bs=10MB与bs=100MB，修改块大小。

流完成时间的表格如下:



[实验结果及分析]

利用Origin工具绘图如下:



**[互联网协议实验]**

1. wireshark抓到的几种协议:

**ARP:**地址解析协议，在已经知道了一个机器（主机或路由器）的IP地址，需要找出其相应的硬件地址。地址解析协议根据IP地址找硬件地址，ARP协议被划归为网络层，为了从网络层使用的IP地址解析出在数据链路层使用的硬件地址。

地址解析协议ARP解决IP地址和下面的网络硬件地址由于格式不同而不存在简单的映射关系。在主机ARP高速缓存中存放一个从IP地址到硬件地址的映射表。

例子：当主机A要向本局域网上的某个主机B发送IP数据报时，就先在其ARP高速缓存中查看有无主机B的IP地址。如有，就在ARP高速缓存中查出其对应的硬件地址，再把这个硬件地址写入MAC帧，然后通过局域网把该MAC帧发往此硬件地址。

如果查不到：

1. ARP进程在本局域网上广播发送一个ARP请求分组。
2. 在本局域网上的所有主机上运行的ARP进程都收到此ARP请求分组。
3. 主机B的IP地址与ARP请求分组中要查询的IP地址一致，就收下这个ARP请求分组，并向主机A发送ARP响应分组。
4. 主机A收到主机B的ARP响应分组后，就在其ARP高速缓存中写入主机B的IP地址到硬件地址的映射。

**DNS协议：**域名系统DNS——从域名解析出IP地址。机器在处理IP数据报时要使用IP地址而不使用域名，因为IP地址的长度是固定的32位（IPv6地址，128位，也是定长的),域名的长度不是固定的，机器处理起来比较困难。域名结构www.ucas.ac.cn，其中ac表示科研机构，cn表示中国，属于国家顶级域名，按照层次来安排。

**TCP协议:**传输控制协议，（1）TCP是面向连接的运输层协议。（2）每一条TCP连接只能有两个端点。每一条TCP连接只能是点对点的（3）TCP提供可靠交付的服务。（4）TCP提供全双工通信。（5）面向字节流。连接是TCP最基本的抽象，连接的端点叫做套接字或插口。

套接字socket=(IP地址:端口号），每一条TCP连接唯一地被通信两端的两个端点（即两个套接字）所确定。即:

TCP连接::={socket1,socket2}={(IP1:port1),(IP2:port2)}。

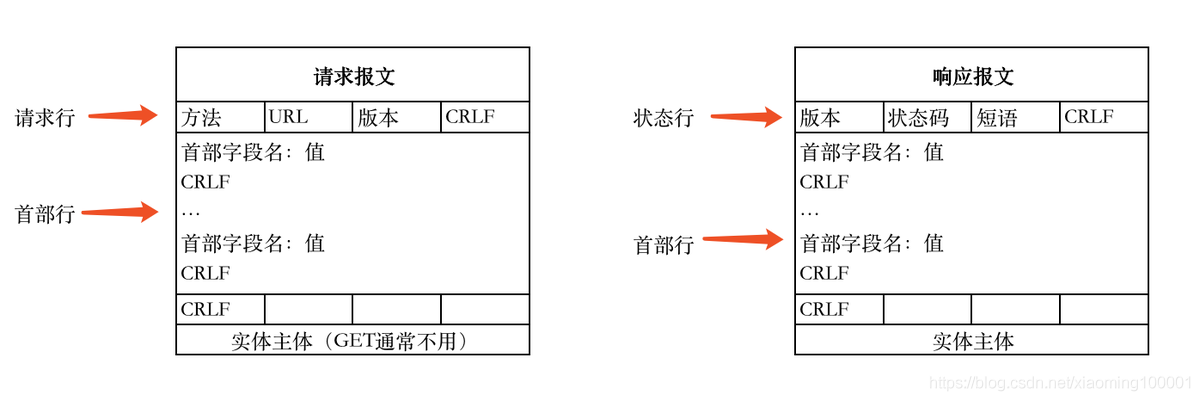
**HTTP协议：**HTTP协议定义了浏览器（万维网客户进程）怎样向万维网服务器请求万维网文档，以及服务器怎样把文档传送给浏览器。是面向事务的应用层协议，是万维网上能够可靠地交换文件（包括文本、声音、图像等各种多媒体文件）的重要基础。

**HTTP和HTTPS：**区别主要在于安全性。

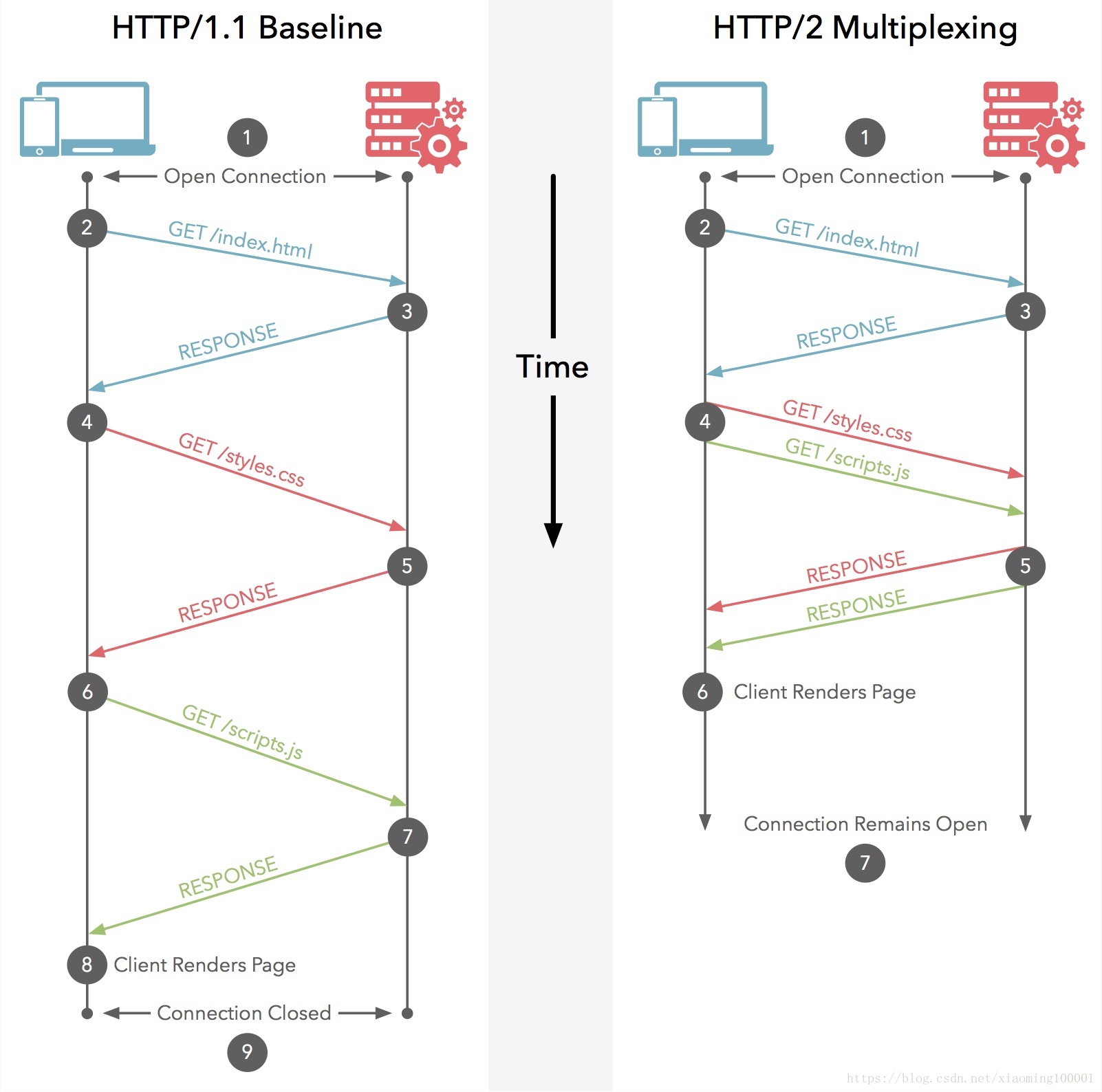
HTTP是超文本传输协议，基于请求与响应，无状态的，应用层的协议，常基于TCP/IP协议传输数据，互联网上应用最为广泛的一种网络协议，所有的www文件都必须遵守这个标准。设计HTTP的初衷是为了提供一种发布和接收HTML页面的方法。改进版本HTTP/2,自2015年，逐渐覆盖市场。

下图展示多路复用技术：通过单一的HTTP/2连接请求发起多重的请求-响应消息，多个请求stream共享一个TCP连接，实现多路并行而不是依赖建立多个TCP连接。

HTTP报文格式：



性能比较：



HTTPS是身披SSL外壳的HTTP。HTTPS是一种通过计算机网络进行安全通信的传输协议，经由HTTP进行通信，利用SSL/TLS建立全信道，加密数据包。HTTPS使用的主要目的是提供对网站服务器的身份认证，同时保护交换数据的隐私与完整性。TLS是传输层加密协议，前身是SSL协议，由网景公司1995年发布，有时候两者不区分。

2017年1月发布的Chrome 56浏览器开始把收集密码或信用卡数据的HTTP页面标记为“不安全”，若用户使用2017年10月推出的Chrome 62，带有输入数据的HTTP页面和所有以无痕模式浏览的HTTP页面都会被标记为“不安全”，此外，苹果公司强制所有iOS App在2017年1月1日前使用HTTPS加密。

**h1下载ucas页面的整个过程：**

1. 使用的链接指向了“中国科学院大学官网”的页面，其URL是https://www.ucas/ac/cn
2. h1-eth0端口分析链接指向页面的URL。
3. h1-eth0端口向DNS请求解析www.ucas.ac.cn的IP地址。
4. 域名系统DNS解析出中国科学院大学服务器的IP地址。
5. h1-eth0端口与服务器建立TCP连接（在服务器端IP地址，和端口80)
6. h1-eth0端口发出取文件命令 wget
7. 服务器www.ucas.ac.cn给出响应，把文件发送给h1-eth0端口。
8. 释放TCP连接。
9. h1-eth0显示“中国科学院大学官网”.html文件中的所有文本。只下载其中的文本部分。

[流完成时间实验]

观察到的现象：虽然传输带宽设置了10Mbps、100Mbps、500Mbps、1000Mbps，但测试得到的平均下载速率可能只有带宽的10%-50%。同时刚开始下载时，速率较慢，中间会趋于稳定，至变快。带宽越大，大部分情况下，下载速率越快。完成时间与文件大小有关，传输1MB的文件和传输100MB的文件耗时显然不同，但也不是简单的线性关系。

TCP传输：一共经历三次握手。

1. 建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN\_SENT状态，等待服务器确认;SYN:同步序列编号(Synchronize Sequence Numbers)。
2. 服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k),即SYN+ACK包，服务器进入SYN\_RECV状态；
3. 客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1),此包发送完毕，客户端和服务器进入ESTABLISHED(TCP连接成功）状态，完成三次握手。

完成三次握手后，客户端和服务端开始传送数据。

**慢启动机制：**最初的TCP在连接建立成功后会向网络中发送大量的数据包，这样很容易导致网络中路由器缓存空间耗尽，从而发生拥塞。因此新建立的连接不能够一开始就大量发送数据包，而只能根据网络情况逐步增加每次发送的数据量，以避免上述现象的发生。具体来说，当新建连接时，cwnd初始化为1个最大报文段(MSS)大小，发送端开始按照拥塞窗口大小发送数据，每当有一个报文段被确认，cwnd就增加1个MSS大小。这样cwnd的值就随着网络往返时间(Round Trip Time,RTT)呈指数级增长，事实上，慢启动的速度一点也不慢，只是它的起点比较低一点而已。我们可以简单计算下：

开始 ---> cwnd = 1经过1个RTT后 ---> cwnd = 2\*1 = 2

经过2个RTT后 ---> cwnd = 2\*2= 4经过3个RTT后 ---> cwnd = 4\*2 = 8如果带宽为W，那么经过RTT\*log2W时间就可以占满带宽。