网络编程大作业东南大学网络拓扑自动构造

71118225 赵君亮

2021 年 6 月 20 日

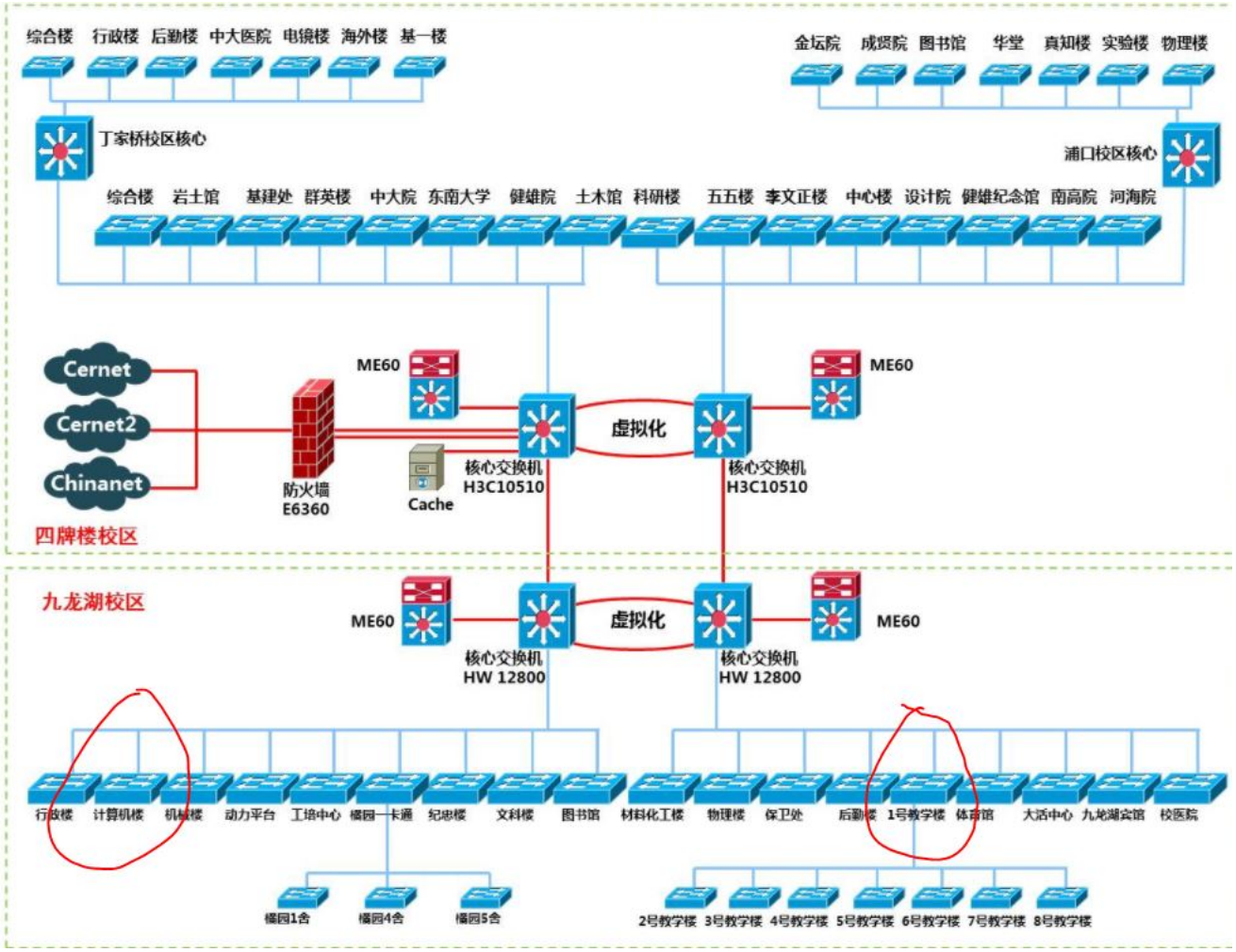
1 网络拓扑构造原理

网络拓扑结构是指用传输媒体互连各种设备的物理布局（将参与 LAN 工作的各种设备用媒体互连在一起有多种方法，但是实际上只有几种方式能适合 LAN 的工作）。要构造网络拓扑图，可以通过 ping 出网络结构种的活跃节点，并且通过 tracert 追踪指令去追踪该活跃节点，通过多次追踪，找到不同路径，确定网络拓扑种的主干节点。

2 东南大学校园网结构

为了实现本次网络拓扑图的自动构造，首先要去探究实验网络环境, 对于东南大学，是校园网的一种，它需要划分不同的子网来提供给学生和老师使用。

东南大学依据教学区、生活区以及专业楼等不同场所划分不同的子网结构，下图是通过信息网获取的官方东南大学拓扑结构。



从东南大学的网络拓扑结构，我们很明显可以看出来，其各个子网是由交换机控制的，并且该拓扑结构相对比较简单，仅仅是交换机的二层结构，对于网络拓扑结构的构建大大减少了工作难度。校园内部网络是有防火墙阻隔，防止外网入侵。在这个子网拓扑图中，本次实验主要选择教学楼宿舍区以及计算机楼这三个地区作为锚点，进行实验，分析出东南大学传输的主干交换机 IP。

分析了东南大学的网络拓扑结构，接下来需要了解的是各个子网的 IP 分配。通过查询东南大学信息化中心，可以获得官方给定 IP 地址。信息来源：https://nic.seu.edu.cn/wlgl/xnIPgl.htm

通过九龙湖校区 IP 分配，可以很明显得出，东南大学划分的子网掩码是 16 位。这里采用宿舍区 10.208.0.0/16 以及 10.203.0.0/16 两个子网进行互 ping 实验，并且找出其中的主干节点，即可以获取东南大学交换机的固定 IP 地址。

由于要能 ping 通，必须处于一个子网下才可以转发，所以可以推测东南大学交换机的固定地址存在 10.208 开头或者 10.203 开头。有了这些想法和前提条件，可以开始设计实验。

3 实验条件说明

3.1 工具选用

在本次实验中，由于需要进行大量数据数据处理和拓扑图的构造，实现较为方便的 python 语言进行编程。对于网络拓扑结构图的构建，采用了图形数据库 neo4j 进行保存存储拓扑图。neo4j 是一个比较常用的知识图谱数据库，可以方便的构造出有向图。

3.2 环境说明

本次编程是在 Windows10 的环境下，并且连接了东南大学的校园网的条件下实现的。系统环境需要存在 ping 指令以及 tracert 指令。

3.3 实验锚点 IP 选定

这里选定了:

教一 10.19.0.0/16

计算机楼以及除了教一以外的其他教学楼 10.203.0.0/16

宿舍区 10.208.0.0/16

上述三个为锚点选中。为什么是这样的呢。首先，从日常的校园网使用，不难发现教一、宿舍楼以及计算机楼的三个地方的 wifi 命名是不不同的。此外，通过我的实际测验发现，当我迁移位置时，比如从教一到计算机楼的时候，ip 地址发生了改变，而且需要重新验证连接。但是当我从教二、教四到计算机楼的时候，ip 地址是不变的，连接也是会自动连不需要验证的。宿舍楼同理，所以这里可以推测学校的这三个位置分属三个子网，分别从每个子网取出几个锚点，可以更好的进行主干 IP 的提取。

4 实验设计与实现

4.1 通过 ping 获取网络结构的在线活跃点

首先对于东南大学的校园网而言是一个局域网，各个子网都是由交换机控制的，之间是可以相互转发的。所以针对不同子网而言原理上是可以 ping 通的。但是由于并不完全处于一个子网下，ping 会有一定的延迟，相较于广域网的延迟而言，这种延迟对本实验并不会产生太大的影响。此外还有一个因素也会影响本实验的效率，那便是访问的活跃节点很多事个人主机存在着防火墙，有时候并不能 ping 通，会造成长时间的延迟无反馈。这里就涉及到一个等待时间的问题。

```
def ping_ip(ip_str):  #调用ping程序 获取活跃ip  主要通过cmd的ping实现
    mycmd = ["ping", "-{op}".format(op=get_os()),
             "1", ip_str]
    output = os.popen(" ".join(mycmd)).readlines()
    for line in output:
        if str(line).upper().find("TTL") >= 0:
            global allipOnline
            allipOnline.append(ip_str)
            break
```

在上述的代码种，首先是先确定了本机的操作系统，方便去调用系统的 ping 指令并且在 ping 指令种设置了 TTL 作为是否 ping 成功的判定依据。如果成功则将该活跃节点保存用于后续的节点追踪。

第二个问题是要解决 ping 不同的情况，因为其实并非一个子网下的所有动态 IP 都是 ping 得通的，所以这里是不可以用手动去 cmd 上进行 ping 指令的。这里需要写一个脚本去遍历所有的动态子 IP。举个例子，比如我在教学楼，然后需要 ping 通宿舍楼的 IP，那么这里的话就要 ping 10.208.0.0/16 子网下的所有 IP，10.208.0.0-10.208.255.255。

第三个问题是在第二个问题的基础上的，作为 cmd 的 ping 指令，其实尝试过都可以轻易发现，每个 ping 指令都是需要很长时间的检查的，ping 一个地址，约为 3-5s 反馈结束。所以作为几百上千个的 ping，这里就需要调用的一个多线程的操作。

```
def find_ip(ip_prefix):  #扫描所有ip段  采用多线程扫描
    '''''
    给出当前的ip地址段 ，然后扫描整个段所有地址
    '''
    threads = []
    for i in range(1, 256):
        ip = '%s.%s' % (ip_prefix, i)
        threads.append(threading.Thread(target=ping_ip, args={ip, }))
    for i in threads:
        i.start()
    for i in threads:
        i.join()
```

由于不同代码段的对称性，比如 10.208.5.0/8 与 10.208.6.0/8 路径是相似的所以，对于问题二这里可以有一个比较好的修正方案。除了多线程，对于代码段的选取也可以不用全选。比如可以从 10.208.0.0/8-10.208.40.0/8。然后再在高位代码段选取一段，比如 10.208.128.0/8-10.208.192.0/8。这样就解决了 ping 大量 IP 的时间运行问题了。

如下图所示，是一个从教一 ping 宿舍 10.208.192.0/8-10.208.193.0/8IP 段的实验结果

```
开始扫描时间: Sun Jun 20 17:03:40 2021
扫描接口 ip地址为: 10.208.192 请等待
ip: 10.208.192.1 在线
ip: 10.208.192.224 在线
扫描接口 ip地址为: 10.208.193 请等待
ip: 10.208.193.239 在线
ip: 10.208.192.1 在线
ip: 10.208.192.224 在线
ip: 10.208.193.239 在线
扫描结束时间 Sun Jun 20 17:03:54 2021
本次扫描共检测到本网络存在3台设备
```

4.2 通过 tracert 追踪活跃 ip

在上述过程完成了活跃节点的获取，并将其保存。接下来就是要对这一列的活跃节点进行追踪来获取主干静态 IP。首先是要了解与 tracert 相关的原理和指令。

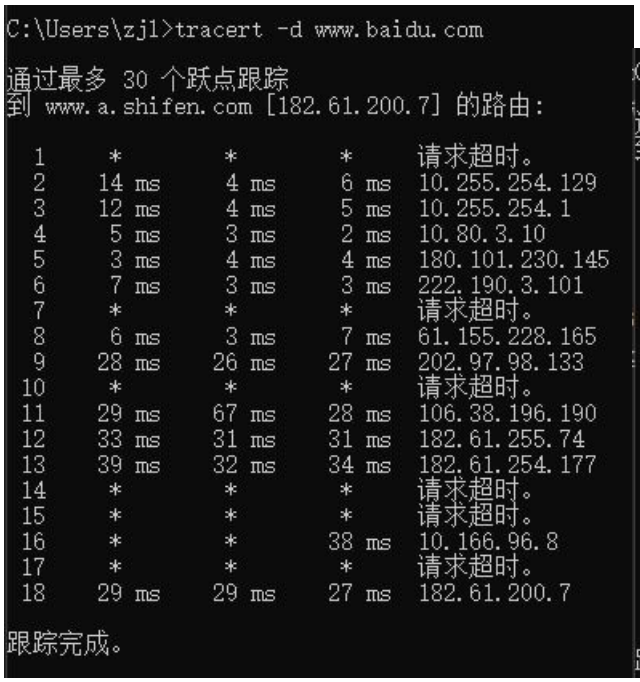
tracert 是一个 windows 系统的 IP 追踪指令（linux 是 traceroute，其多了可以自主选定源节点的功能，tracert 只支持 IPV6 的源节点选中，IPV4 不支持），tracert 具有很多的功能

如下图所示是 tracert 一些常用的参数

```
C:\Users\zjl>tracert

用法: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
              [-R] [-S srcaddr] [-4] [-6] target_name

选项:
    -d                  不将地址解析成主机名。
    -h maximum_hops     搜索目标的最大跃点数。
    -j host-list        与主机列表一起的松散源路由(仅适用于 IPv4)。
    -w timeout          等待每个回复的超时时间(以毫秒为单位)。
    -R                  跟踪往返行程路径(仅适用于 IPv6)。
    -S srcaddr          要使用的源地址(仅适用于 IPv6)。
    -4                  强制使用 IPv4。
    -6                  强制使用 IPv6。
```

tracert 指令可以通过途径各个静态 IP 追踪到目标 IP，并把路径 IP 显示出来。并且它还具有解析主机名的功能。这里有一个值得一提的问题。解析主机名虽然作为一个不错的功能，但是在本次实验中探索的是路径，并不需要解析主机名，并且解析主机名要消耗大量的时间，所以在本次实验中，需要在指令中加入 -d 的参数来取消解析主机名这一个操作，来提高运行的效率。

在进行实验前，这里做了一个小测验来探究 tracert 的功能，使用 tracert 探究从本机 IP 到 ping 通百度 IP 走过的所有 IP 节点。



在上图可以很清晰的看见一个从本机 IP 到百度 IP 的过程。当然中间可能也是会遇到一些请求超时的节点，这些节点其实是存在防火墙或者反馈时间过长，导致获取不到该节点 IP。对于上述问题，大多出现在广域网的路径追踪中。而本次实验，作为局域网，并不会出现中间访问断点的问题。

另外，从上述对 tracert 的简单测试，左边是追踪百度网址，右边是追踪 qq 网址很明显可以看出来，追踪的前三项是相同的，这里可以作一个大胆的推测，10.255.254.129 10.255.254.1 10.80.3.10 这三个 IP 地址都是学校的主干静态 IP，也就是我们需要构造的拓扑图节点

4.3 选取多个网络节点作为锚点重复追踪

完成了简单的测试之后，接下来需要在校园网内进行大量追踪实验。

这里分为从宿舍楼到教一、从宿舍楼到计算机楼、从教一到宿舍楼、从计算机楼到宿舍楼、以及从教一到计算机楼的追踪，五个部分的追踪。前面讲述过锚点选用的原因，这里不再赘述。为了找到校园网的主干拓扑，必须采用不同子网间的互相追踪才可以获取转发的主干节点。

对于追踪这部分的程序实现，因为有大量的 IP 地址需要追踪，所以同样要采用多线程的 tracert 指令运行，并且要带上 -d 参数，取消目标 IP 的主机解析。

追踪代码多线程实现如下图所示

```
def tracert_allip(desip): #追踪所有ip  采用多线程追踪加速
    '''''
    给出当前的ip地址段 ，然后扫描整个段所有地址
    '''
    threads = []
    for i in desip:
        print(i+"追踪开始")
        threads.append(threading.Thread(target=tracert_ip, args={i, }))
    for i in threads:
        i.start()
    for i in threads:
        i.join()
```

```python
def tracert_ip(ip):  #追踪某个ip
    mycmd="tracert -d "+ip
    output = os.popen(mycmd).readlines()
    global alltracertinfo
    alltracertinfo.append((ip,output))
```

4.4 通过大量追踪数据，找到其中的主干转发节点

通过一系列的追踪，程序运行效果如下图所示。

```
PS C:\Users\zjl\Desktop\网络拓扑图构建项目\networkip>  c:; cd 'c:\Users\zjl\Desk
\python\debugpy\launcher' '1223' '--' 'c:\Users\zjl\Desktop\网络拓扑图构建项目\n
开始扫描时间: Sun Jun 20 19:32:34 2021
扫描接口 ip地址为: 10.208.192 请等待
ip: 10.208.192.1 在线
ip: 10.208.192.1 在线
扫描结束时间 Sun Jun 20 19:32:40 2021
本次扫描共检测到本网络存在1台设备
开始追踪时间: Sun Jun 20 19:32:40 2021
10.208.192.1追踪开始
**************************
追踪10.208.192.1

通过最多 30 个跃点跟踪到 10.208.192.1 的路由

  1     *        *        *     请求超时。
  2     *        4 ms     *     10.255.254.129
  3    10 ms     3 ms     3 ms  10.255.254.1
  4     3 ms     1 ms     2 ms  10.80.128.150
  5     3 ms     1 ms     7 ms  10.208.192.1

跟踪完成。
**************************
追踪结束时间 Sun Jun 20 19:33:02 2021
PS C:\Users\zjl\Desktop\网络拓扑图构建项目\networkip>
```

由图可见，追踪 10.208.192.1，经过 10.255.254.129 10.255.254.1 10.80.128.150 最后到 10.208.192.1。这一系列的追踪可以构造出一个图，获得一个路线。该路线保存为一个 python 中的数据结果，map 序列。并且通过该序列，判定出非叶子节点作为主干节点。如下图所示:

```
PS C:\Users\zjl\Desktop\网络拓扑图构建项目\networkip>  c:; cd 'c:\Users
\python\debugpy\launcher' '1032' '--' 'c:\Users\zjl\Desktop\网络拓扑图
('10.19.107.35', '10.255.254.129')
('10.255.254.129', '10.255.254.1')
('10.255.254.1', '10.80.128.150')
('10.80.128.150', '10.208.192.1')
尝试连接neo4j 请稍等
成功连接neo4j 并构造拓扑图
{'10.80.128.150': '10.208/16'}
{'10.80.128.150', '10.255.254.129', '10.19.107.35', '10.255.254.1'}
```

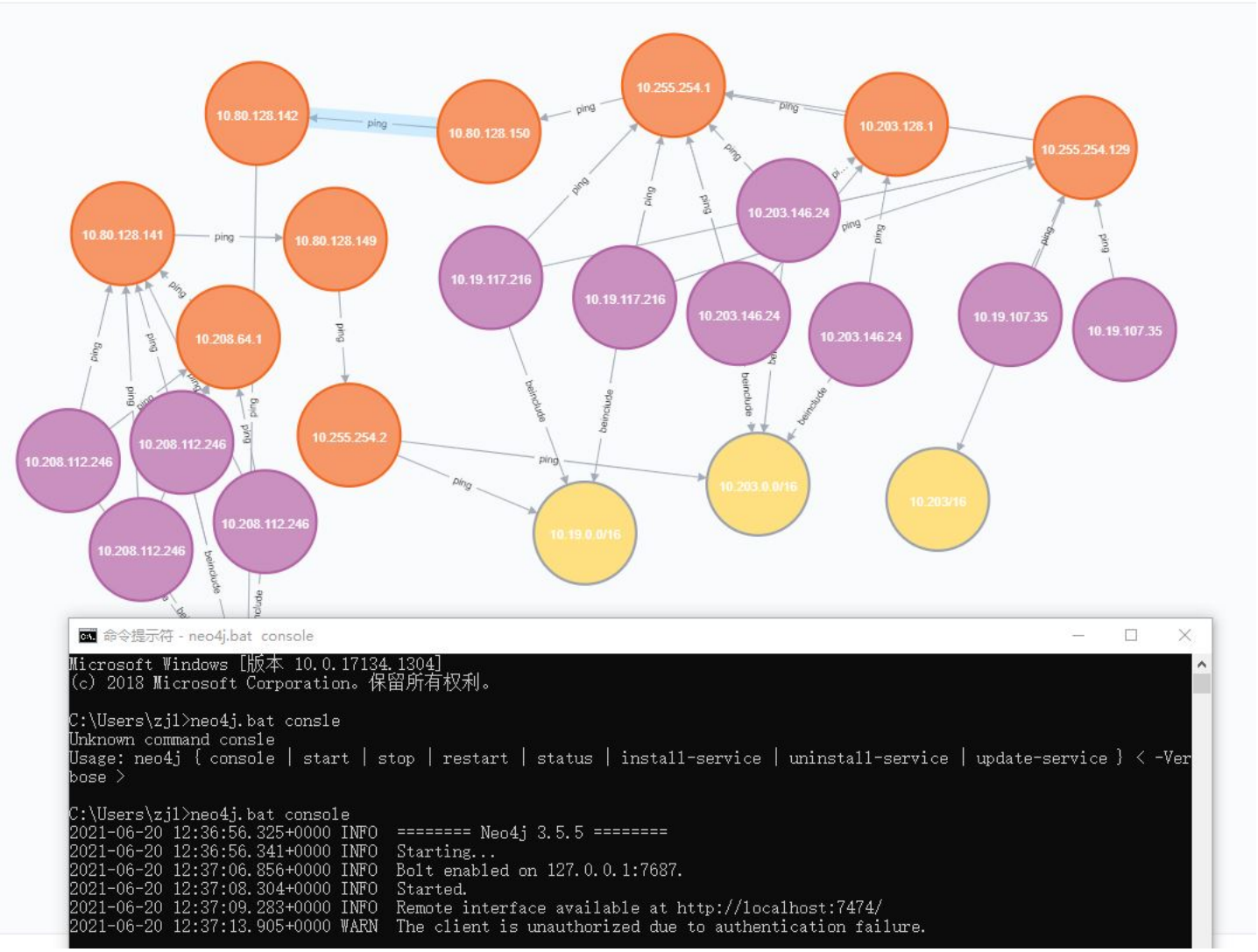
上图用 ('10.19.107.35', '10.255.254.129') 表示这两个 IP 之间互相连接转发并且通过遍历所有的 map 数

据，得到叶子节点，去除叶子节点，留下主干节点。最后获得的主干节点为'10.80.128.150', '10.255.254.129', '10.19.107.35', '10.255.254.1'

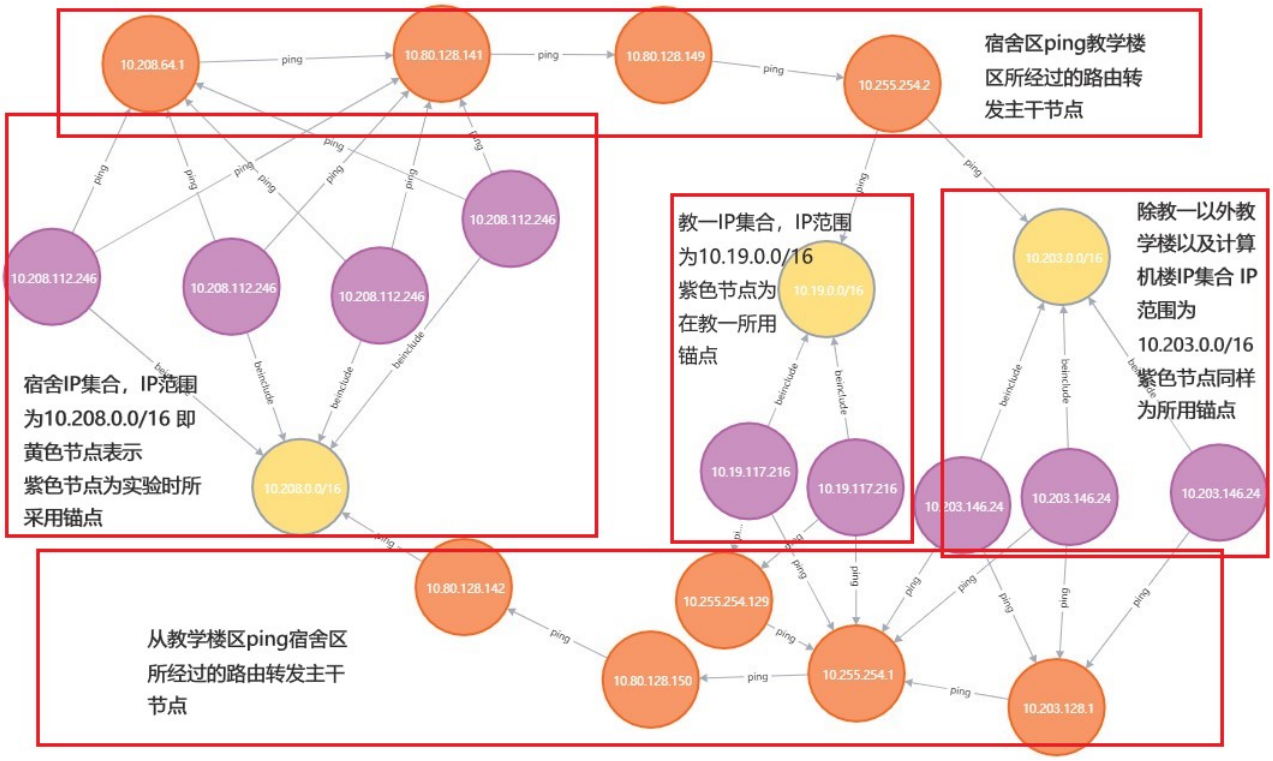
4.5 利用 neo4j 数据库构造拓扑图

完成了主干节点的提取后，最后一步是构造拓扑图。这里采用的是 neo4j 的数据库来构造网络拓扑图，首先把主干节点都写入，然后将有连接的两个主干节点互相连接起来。而由于叶子节点的众多，所以这里直接把叶子节点表示为一个掩码的形式，作为 IP 集合。比如这里的叶子节点都是 10.208.0.0/16 的 IP 集合。

下图是 NEO4J 数据库的使用



下图是构造的结果

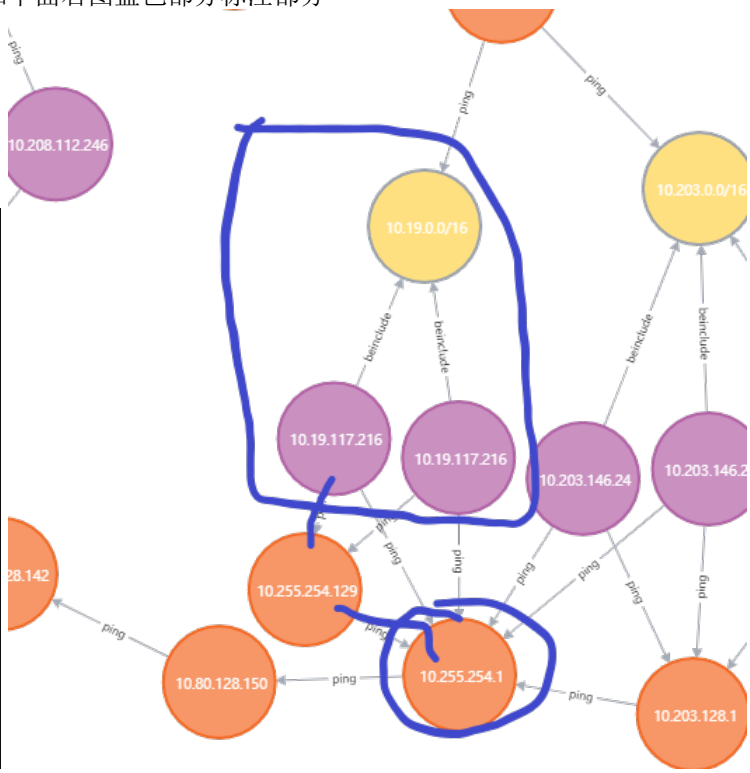


通过上图，可以很明显看出来，橙色节点为交换机静态 IP，即为主干节点。紫色节点是我本机 IP，也就是选定的锚点。黄色节点是一个子网的集合 IP，紫色节点自然也属于这部分集合。

5 实验结果分析与拓展延申

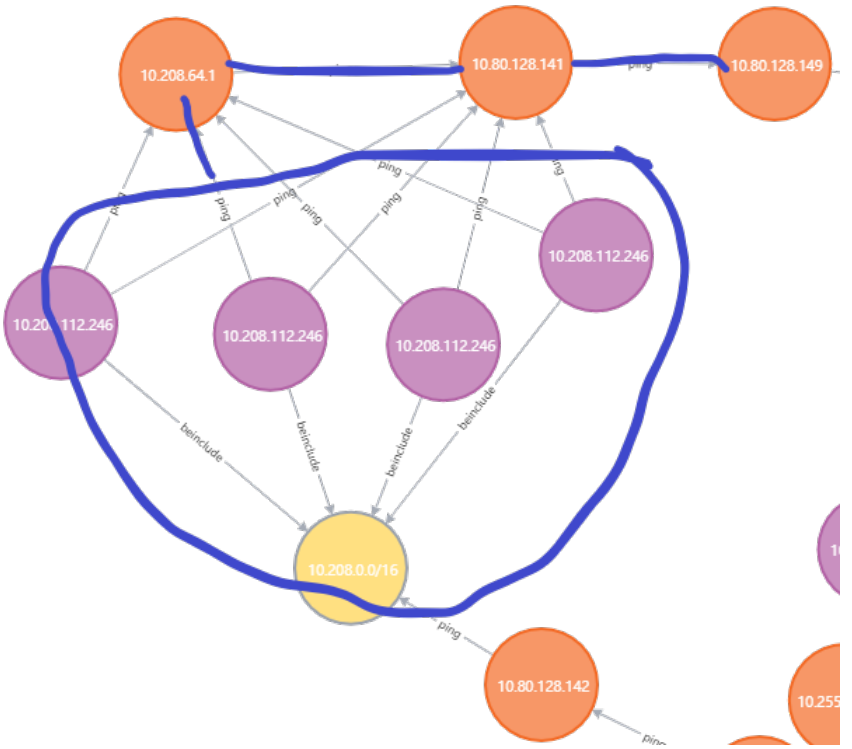
现在回过头，取看刚刚在广域网进行的 tracert 指令。可以发现一些有趣的事情，根据百度网址的追踪结果，在校园网拓扑图中标记出现过的节点，如下面右图蓝色部分标注部分

```
C:\Users\zjl>tracert -d www.baidu.com

通过最多 30 个跃点跟踪
到 www.a.shifen.com [182.61.200.7] 的路由:

  1     *        *        *     请求超时。
  2    14 ms     4 ms     6 ms  10.255.254.129
  3    12 ms     4 ms     5 ms  10.255.254.1
  4     5 ms     3 ms     2 ms  10.80.3.10
  5     3 ms     4 ms     4 ms  180.101.230.145
  6     7 ms     3 ms     3 ms  222.190.3.101
  7     *        *        *     请求超时。
  8     6 ms     3 ms     7 ms  61.155.228.165
  9    28 ms    26 ms    27 ms  202.97.98.133
 10     *        *        *     请求超时。
 11    29 ms    67 ms    28 ms  106.38.196.190
 12    33 ms    31 ms    31 ms  182.61.255.74
 13    39 ms    32 ms    34 ms  182.61.254.177
 14     *        *        *     请求超时。
 15     *        *        *     请求超时。
 16     *        *       38 ms  10.166.96.8
 17     *        *        *     请求超时。
 18    29 ms    29 ms    27 ms  182.61.200.7

跟踪完成。
```



可以很明显发现，节点追踪到 10.255.254.1 后面就是通过 10.80.3.10 对外转发。以及转发到宿舍网也是经过该节点，说明 10.255.254.1 是东南大学九龙湖教学区对外交换的跟节点。而 10.80.3.10 很可能是对外转发的 IP 节点为了验证这一个想法，下列继续进行宿舍区对百度进行追踪

同样可以可以对宿舍追踪百度进行标记，如下图所示



对比教一追踪百度网址 IP 和宿舍区追踪百度网址 IP，可以明显该处也是通过 10.80.3.10 对外转发所以这里可以推断，10.80.3.10 是整个东南大学九龙湖校区对外转发的 IP 节点。当然了在内部转发，是不用不到该节点的所以在该拓扑图上也就看不见这个节点。

6 总结与收获

本次实验可以说比较成功，完整的构造了东南大学九龙湖校区的一个网络拓扑结构。通过该图可以很容易的了解校园网的结构。

整个实验流程为：原理了解 -> 官方网络结构分析 -> 构造设计 -> 编写代码 -> 运行代码 -> 收集数据 -> 处理数据 -> 构造网路拓扑图。

不论是从工具的选择还是实验的设计，该实验都做得比较完善。当然该实验仅仅只是构造了东南大学的校园网，并没有对广域网进行一个构建，在后续的学习生活中，可以对其进行一个更好的改进，可以尝试去测试广域网的拓扑结构。

通过该实验，我学习了很多，学会了各个子网之间的转发原理，了解了一个网络拓扑结构是怎样构建的以及这样的一个网络结构是如何工作的。更好的去理解了何为 IP 地址，何为动态 IP 地址，何为静态 IP 地址。也掌握了 ping 工具和 tracert 的使用。

此外通过整个实验流程的设计和实现，提高了我动手操作的实践能力，以及项目的实现管理能力。