МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования 

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий   
Кафедра   
Вычислительные системы и технологии

Лабораторная работа №2

по дисциплине

Сети и телекоммуникации

РУКОВОДИТЕЛЬ:

В.Е.Гай

СТУДЕНТ:

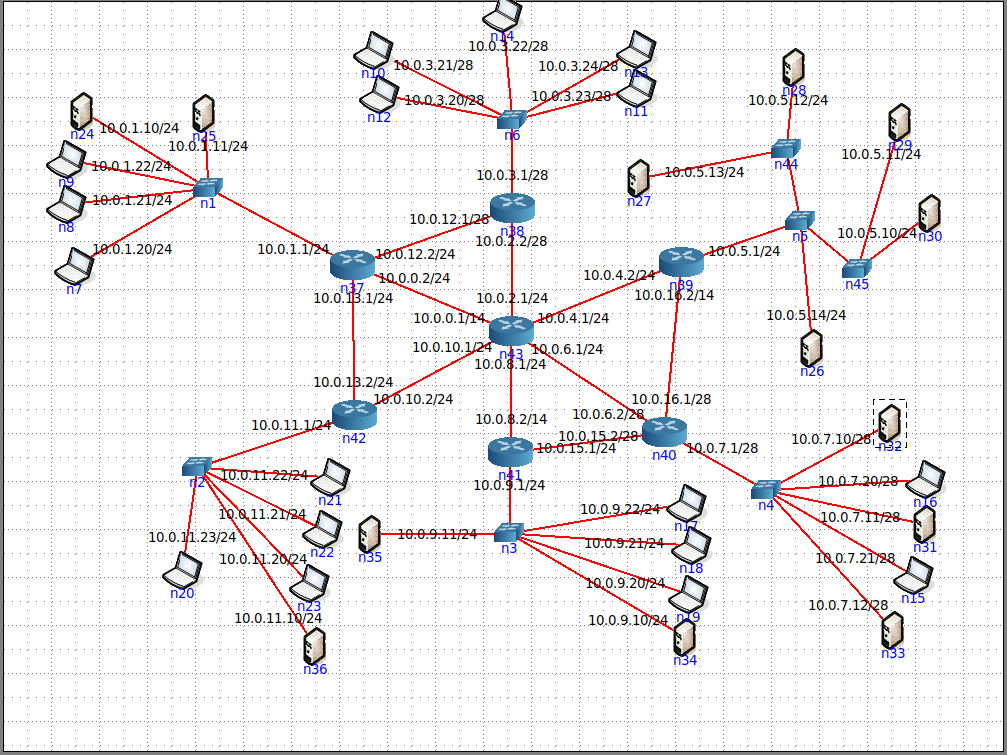
Тарасов А. В.

19-B-1

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

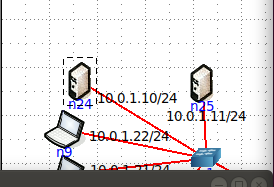
С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

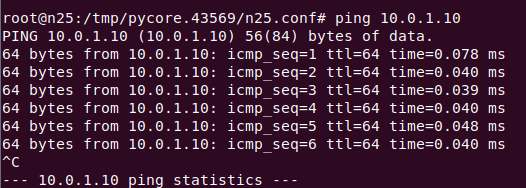
Нижний Новгород 2022



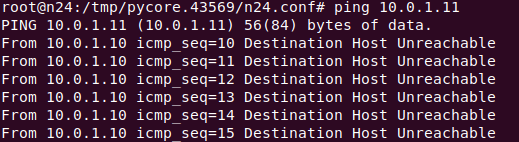
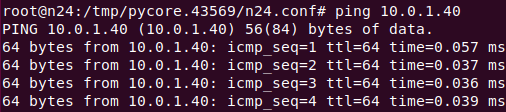
Для начала докажем, что реконфигурация сети в процессе её работы не влияет на хранимое состояние сети.

Запустим эмуляцию

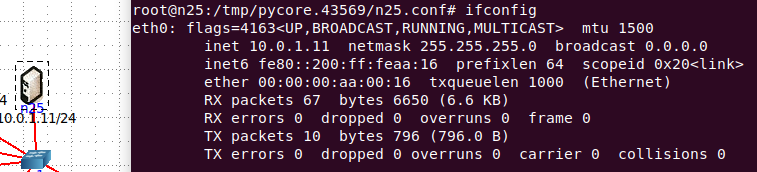


Выполним ping между узлами 10.0.1.10/24 и 10.0.1.11/24

Сменим IP узла 10.0.1.10/24 через терминал  

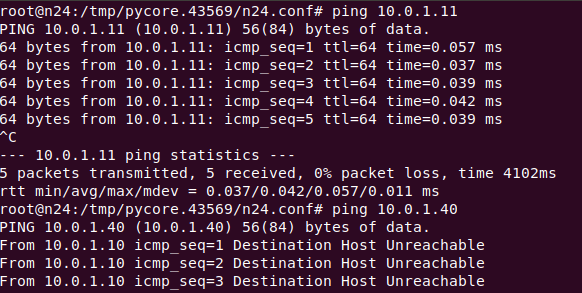

Выполним ping по старому и новому адресу  
Всё так, как должно быть.

Перезапустим сеть.



Как видно, ip прежний.

Выполним ping ещё раз.



Закономерно, всё вернулось к прежним значениям.

Предположительно происходит так: при эмуляции из файла конфигурации эмулятора читаются значения и записываются в память/виртуальные файлы, и после запуска эмуляции изменение этих значений никак не влияет на конфигурационный файл.

Следовательно, для изменения этих значений нужно лезть в конфигурационный файл эмулятора, что проще всего сделать через изменение параметров внутри эмулятора и сохранение файла. Так в будущем и будем поступать.

На первом этапе нужно определить, какие узлы сети нам доступны. Для этого используем следующий bash-script:

#!/bin/bash

#own ip

ipPattern="[0-9]\*\.[0-9]\*\.[0-9]\*\.[0-9]\*"

ip=$(ifconfig | grep -o -m 1 $ipPattern | head -1)

nmap -sP -T 5 10.0.0.0/20 | grep -o $ipPattern>/home/core/Документы/nres/"$ip.txt"

Этот скрипт сохранит весь вывод команды nmap в файл с ip узла вместо имени.

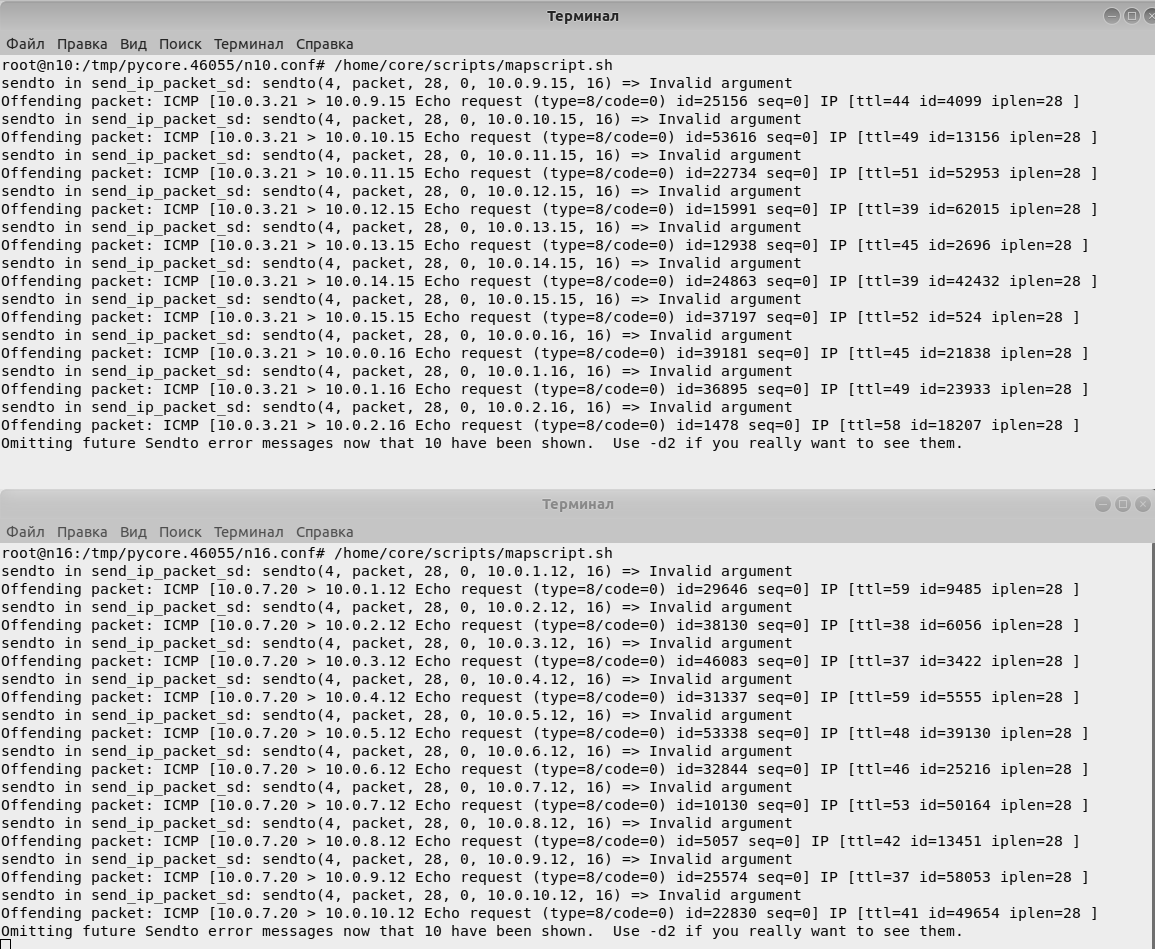
Запустим скрипт из произвольного узла каждой подсети.

Самый “большой” IP-адрес сети - 10.0.16.0, однако самый большой значимый для хадачи адрес - 10.0.11.0. Следовательно, будем искать компьютеры в сети 10.0.0.0/20

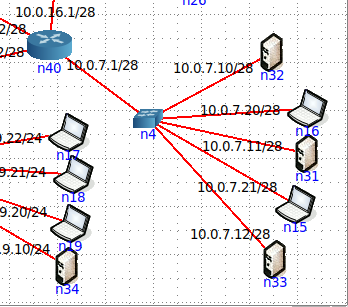
Запустим скрипт из произвольного узла каждой подсети.

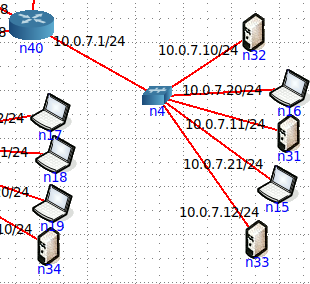


На некоторых узлах всё хорошо, но есть и такие:

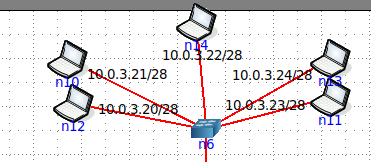


проанализируем содержащие эти узлы сети

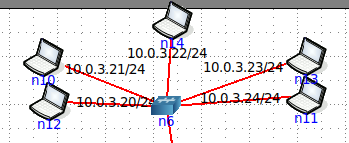


Неисправность в используемой маске сети. Компьютеры с ip соответственно меньшим и большим 10.0.0.16/28 находятся в разных сетях. Исправим маску на /24. 

запустим скрипт вновь.



аналогичная ситуация с n10



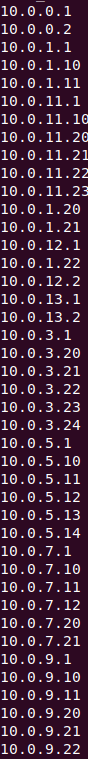
В процессе дальнейшей работы скрипт был переписан следующим образом:

#!/bin/bash

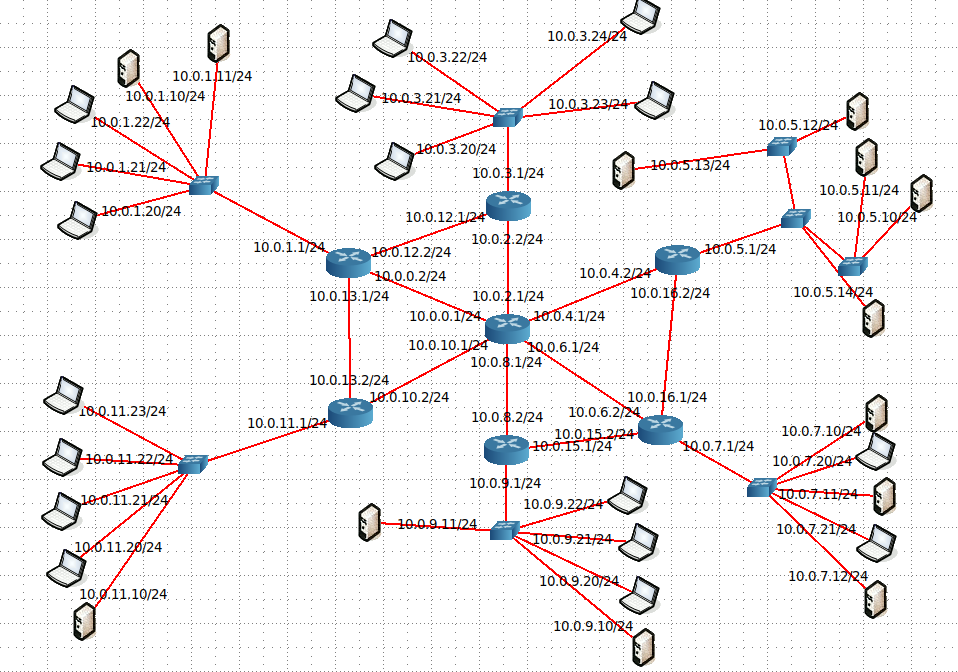
nmap -sP -T 5 10.0.0.0/20 | grep -o "10\.0\.[0-9]\*\.[0-9]\*" | sort

По-сути этот скрипт представляет собой фильтрацию вывода nmap.

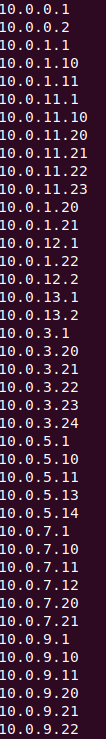
Проанализируем:



Во всех интересующих нас сетях доступно по 6 интерфейсов => сеть вполне работоспособна.. Для подтверждения данного факта nmap был запущен ещё несколько раз с других узлов. Результат - аналогичный.



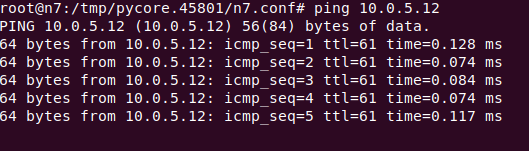
Запустим скрипт.



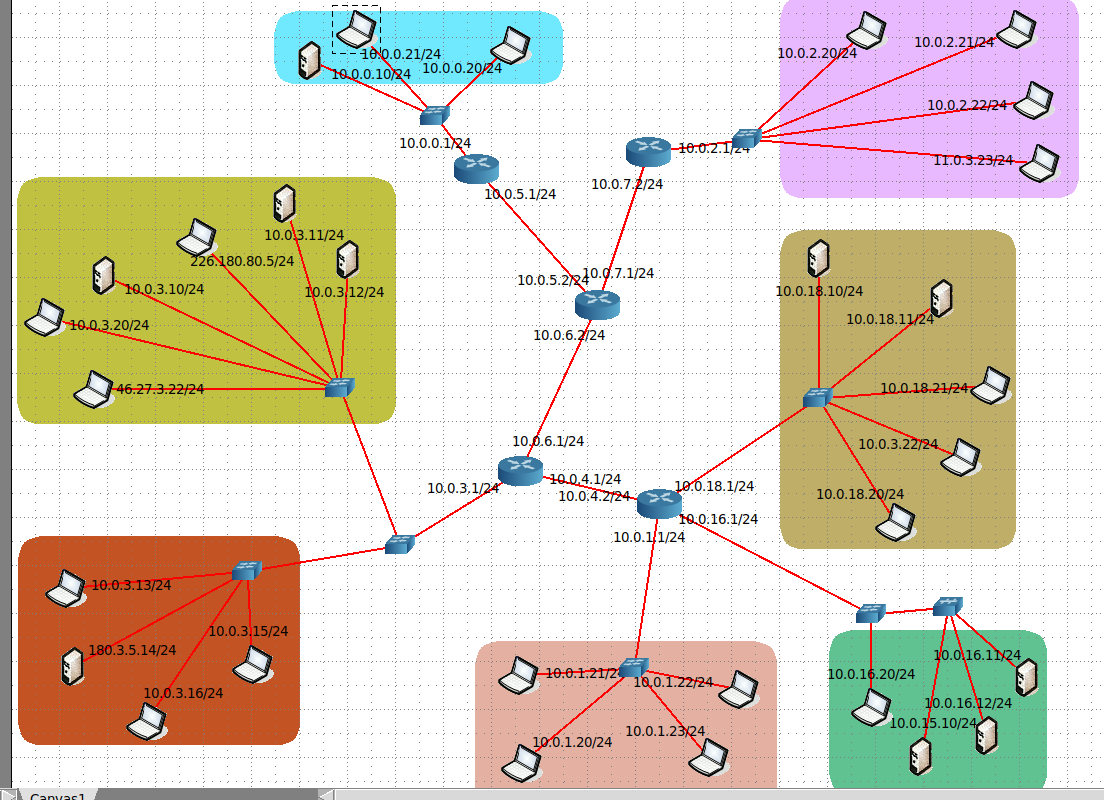
В сети 10.0.5.0/24 недоступен узел 10.0.5.12

Был выявлено отсутствие defaultRoute. Исправим:

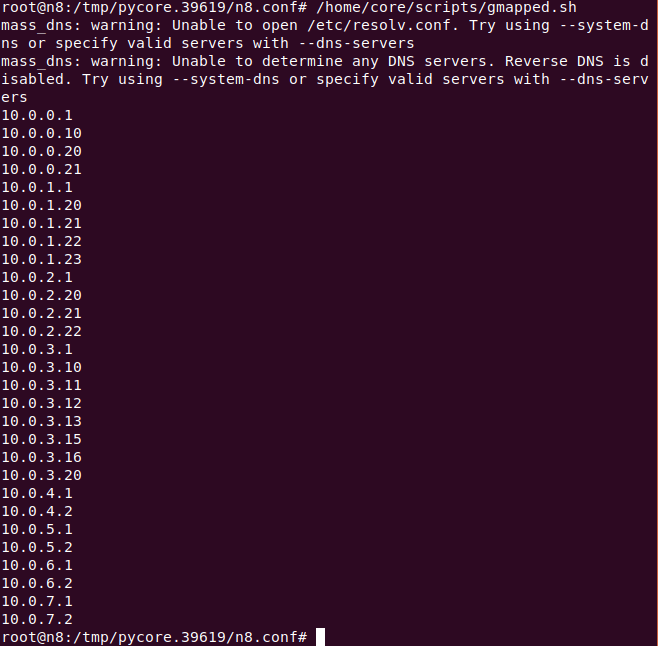




Пинг проходит => узел доступен. Для подтверждения работоспособности сети nmap был запущен ещё несколько раз с узлов других подсетей.



Запустим скрипт.

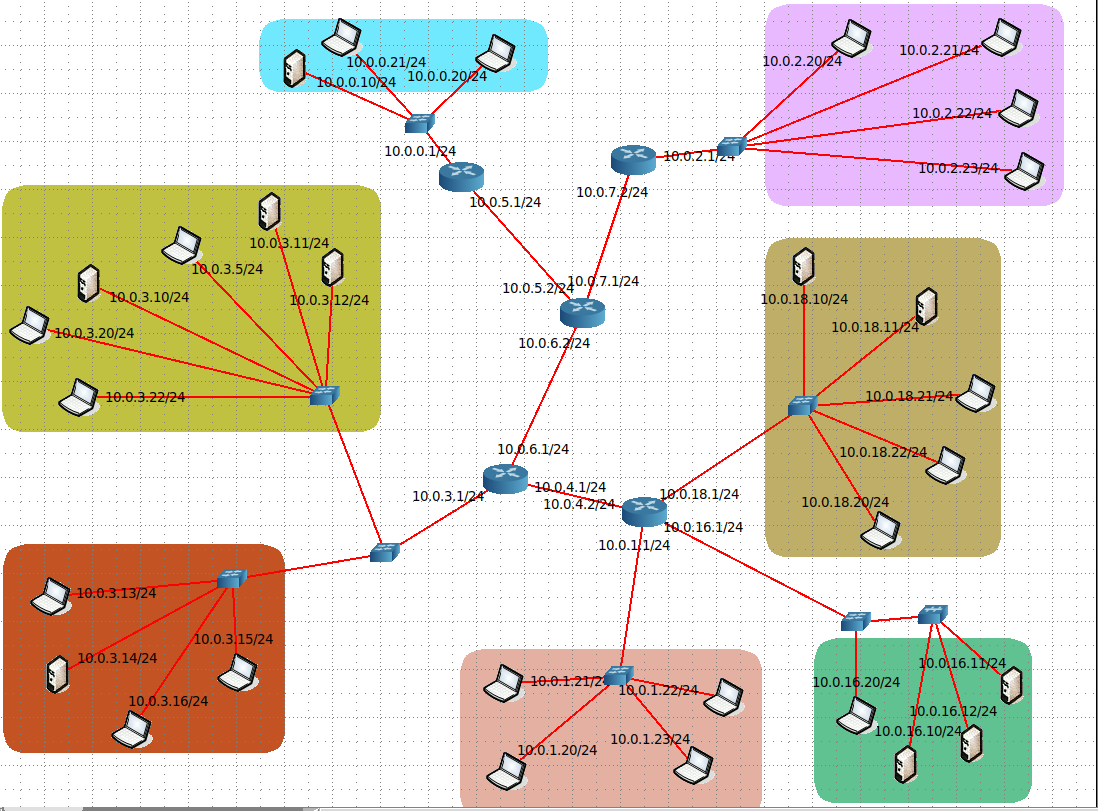


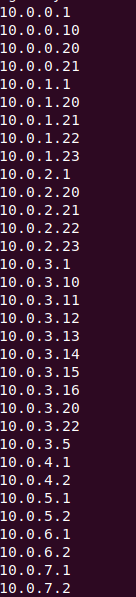
В данном случае у узлов неверно указаны адреса сетей.

Вот список таких узлов:

1. 46.27.3.22
2. 180.3.5.14
3. 11.0.3.23
4. 10.0.3.22 (в сети 10.0.18.0/24)
5. 10.0.15.10 (в сети 10.0.16.0/24)

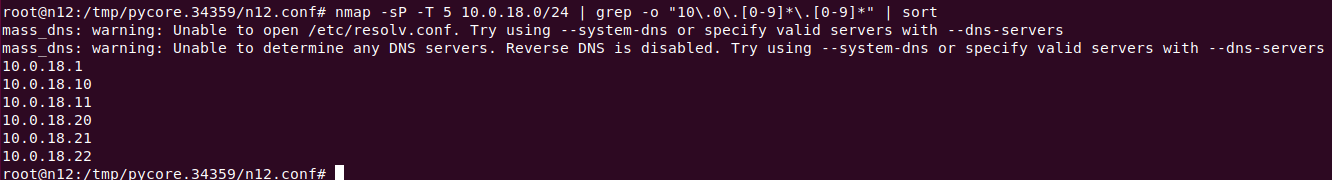
Изменим ip узлов в соответствии с ip сети, в которой они находятся.

Получившаяся сеть:



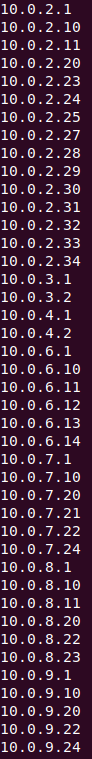
Все интересующие сети доступны в полном объёме. Поскольку используемая маска /20, сети 10.0.16.0/24 и 10.0.18.0/24 - не попадают в этот диапозон, потому проверим их доступность иначе:



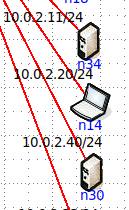
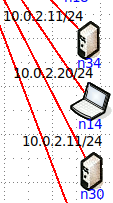


Таким образом, сеть вполне доступна.

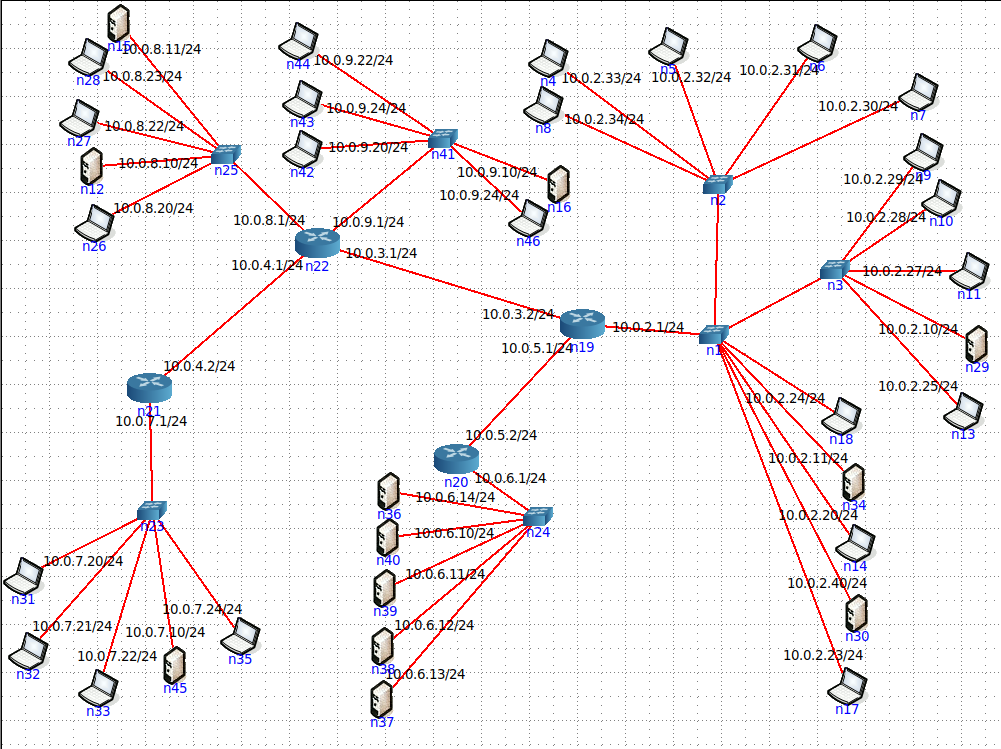
Запустим скрипт.

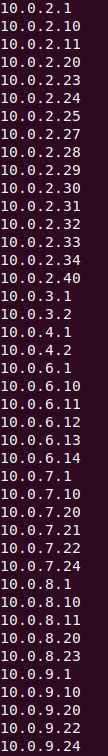


Не хватает одного узла в сети 10.0.2.0/24 Его ip дублируется, исправим это.



Получившаяся сеть.





Неисправность устранена.

Вывод:

В ходе лабораторной работы были приобретены навыки написания bash-скриптов, пользования утилитой nmap, навыки поиска и устранения различных ошибок, возникающих в связи с неправильной настройкой сети.