Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ярославский государственный технический университет»
Кафедра «Информационные системы и технологии»

Отчет защищен
с оценкой _____
Преподаватель
А.Н. Вологин
«20» октября 2022

IPv4-CALCULATOR

Отчет о лабораторной работе №3
по дисциплине «Компьютерные сети»

ЯГТУ 09.03.04 – 003 ЛР

Отчет выполнил
студент группы ЦПИ-21
Д.В. Аристов
«19» октября 2022

2022

Цель работы: изучить способы назначения адресации в стеке протоколов TCP/IP, получить навыки назначения адреса устройства и маски подсети.

Задание: написать программу – IPv4 калькулятор на одном из языков программирования, работающий без использования двоичной системы исчисления. На вход программы подается IPv4 адрес и маска подсети в префиксной форме. На выходе программа должна выводить адрес сети, маску в нормальном виде, широковещательный адрес, количество хостов.

Рисунок 1 – Основная функциональная блок-схема

```
public static int[] normalMask(int prefixNetmask) {
    int[] netmask = new int[4];
    int countZeroBitMask = 32 - prefixNetmask;

    if (countZeroBitMask == 0) {
        for (int i = 0; i < 4; i++) {
            netmask[i] = 255;
        }
    } else if (countZeroBitMask > 0 && countZeroBitMask <= 8) {
        for (int i = 0; i < (prefixNetmask / 8); i++) {
            netmask[i] = 255;
        }
        netmask[(prefixNetmask / 8)] = 256 - (int) pow(2, countZeroBitMask);
    } else if (countZeroBitMask > 8 && countZeroBitMask <= 16) {
        for (int i = 0; i < (prefixNetmask / 8); i++) {
            netmask[i] = 255;
        }
```

```
            countZeroBitMask -= 8;
            netmask[(prefixNetmask / 8)] = 256 - (int) pow(2, countZeroBitMask);
        } else if (countZeroBitMask > 16 && countZeroBitMask <= 24) {
            for (int i = 0; i < (prefixNetmask / 8); i++) {
                netmask[i] = 255;
            }
            countZeroBitMask -= 16;
            netmask[(prefixNetmask / 8)] = 256 - (int) pow(2, countZeroBitMask);
        } else if (countZeroBitMask > 24 && countZeroBitMask <= 32) {
            for (int i = 0; i < (prefixNetmask / 8); i++) {
                netmask[i] = 255;
            }
            countZeroBitMask -= 24;
            netmask[(prefixNetmask / 8)] = 256 - (int) pow(2, countZeroBitMask);
        }
        return netmask;
    }
```

Метод normalMask() возвращает массив из 4 целочисленных элементов – маску сети в нормальном виде. На вход функции подается маска сети в префиксном форме. Определяем количество нулевых элементов маски в двоичном виде. Если количество нулевых элементов маски равно 0, то всем октетам маски в цикле присваивается значение 255. Если количество нулевых элементов от 0 до 8, не включая 0, то всем октетам маски, кроме последнего в цикле присваивается значение 255, а затем последнему присваивается значение 256 минус 2 в степени количества нулевых элементов этого октета маски сети в двоичной форме. Аналогично формируется маска в случае, когда количество нулевых элементов находится в пределе от 8 до 16, от 16 до 24 и от 24 до 32.

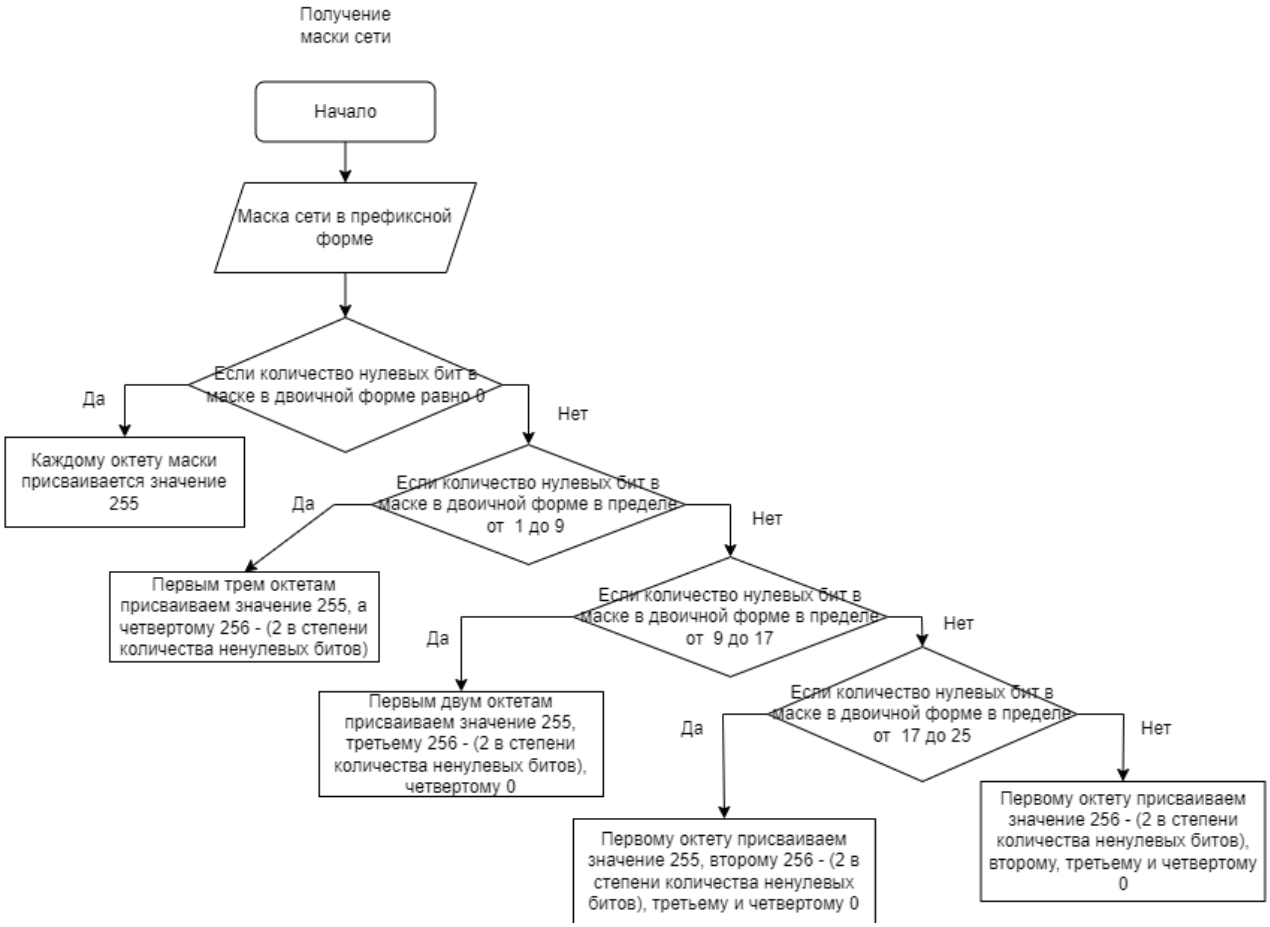


Рисунок 2 – Блок-схема получения маски сети

```java
public static int[] getNetAddress(int[] ip, int[] mask) {
    int[] ipNetwork = new int[4];
    for (int i = 0; i < mask.length; i++)  {
        if (mask[i] < 255) {
            int a = 256 - mask[i];
            for (int j = ip[i]; j > 0; j--) {
                if (j % a == 0) {
                    ipNetwork[i] = j;
                    break;
                }
            }
        } else ipNetwork[i] = ip[i];
    }
    return ipNetwork;
}
```

Метод getNetAddress() возвращает массив из 4 целочисленных элементов – адрес сети. На вход функция получает ip адрес и маску сети в нормальном виде. Цикл проходит по маске: если значение маски 255, то октету адреса сети присваивается значение этого октета IP адреса, в противном случае октету адреса сети присваивается первое значение меньшее этого октета IP адреса, которое делится на 256-значение маски этого октета.

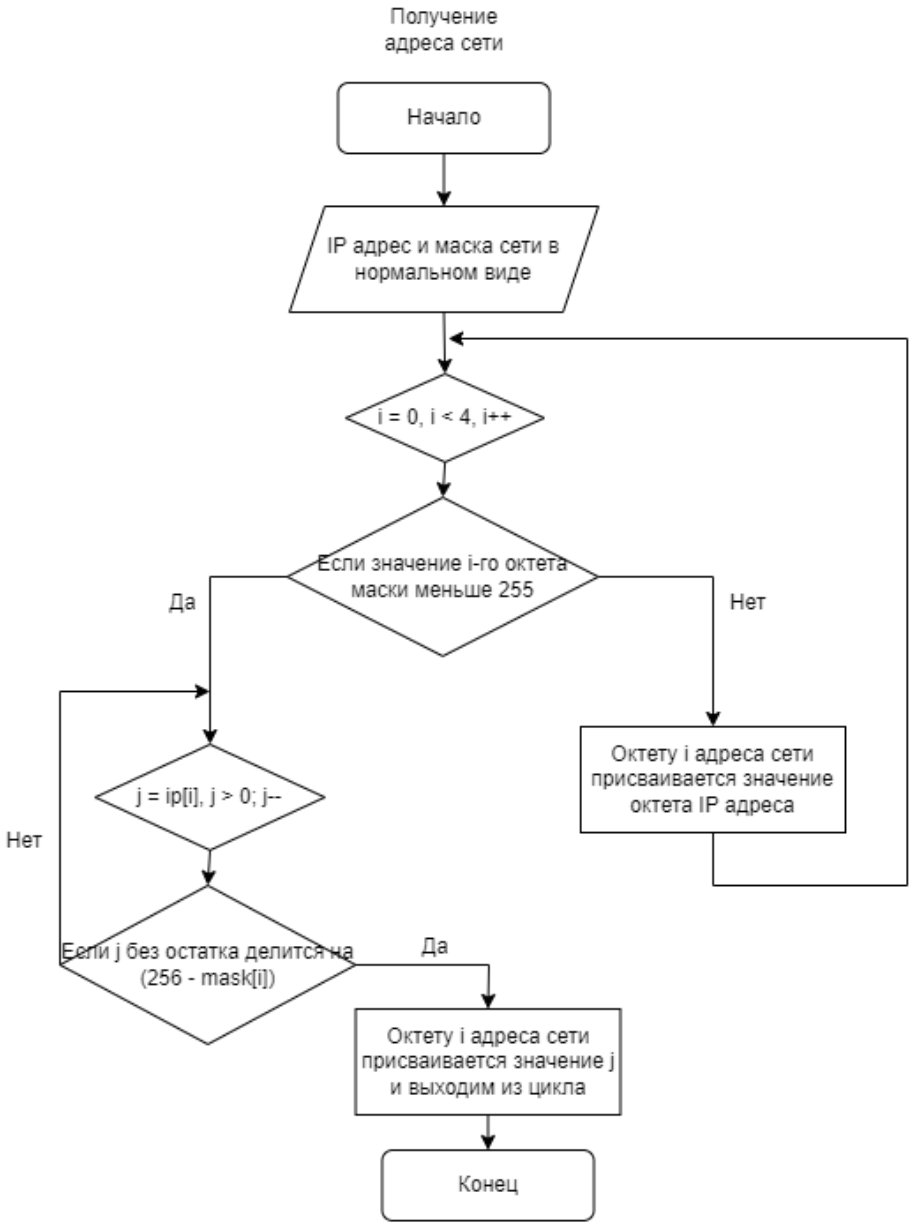


Рисунок 3 – Блок-схема получения адреса сети

```java
public static int[] getBroadcastAddress(int[] netmask, int[] ipNetwork) {
    int[] broadcast = new int[4];
    for (int i = 0; i < netmask.length; i++) {
        if (netmask[i] == 255) {
            broadcast[i] = ipNetwork[i];
        } if (netmask[i] == 0)  {
            broadcast[i] = 255;
        } else {
            int a = 256 - netmask[i];
            broadcast[i] = ipNetwork[i] + (a-1);
        }
    }
    return broadcast;
}
```

Метод getBroadcastAddress() возвращает массив из 4 целочисленных элементов – широковещательный адрес. На вход функция получает маску сети в нормальном виде и адрес сети. Цикл проходит по октетам маски сети в нормальной форме: если значение 255, то широковещательному адресу присваивается значение IP адреса этого октета, если значение 0, то этому октету широковещательного адреса присваивается 0, иначе же октету широковещательного адреса присваивается значение этого октета IP адреса плюс 256 минус значение этого октета маски сети.

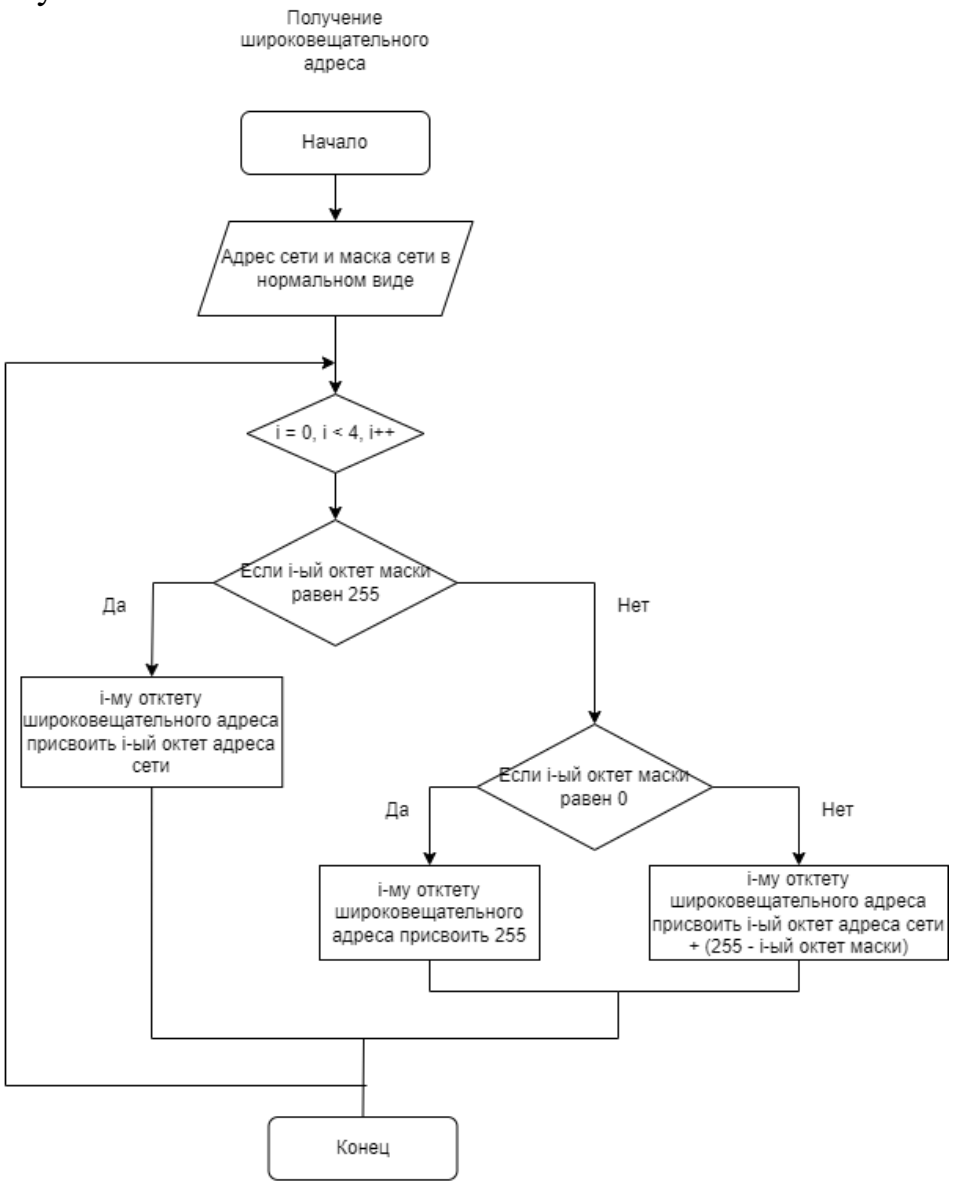


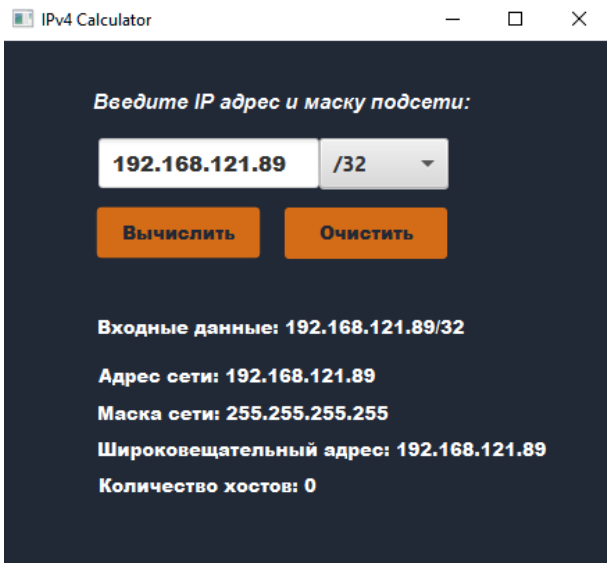
Рисунок 4 – Блок-схема получения широковещательного адреса

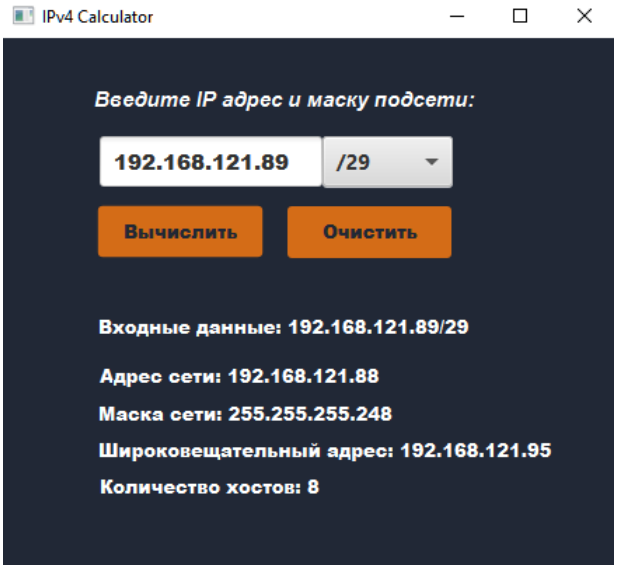
```java
public static long countHosts(int prefixNetmask) {
    if (prefixNetmask == 32) return 0;
    else return (long) pow(2, (32 - prefixNetmask));
}
```

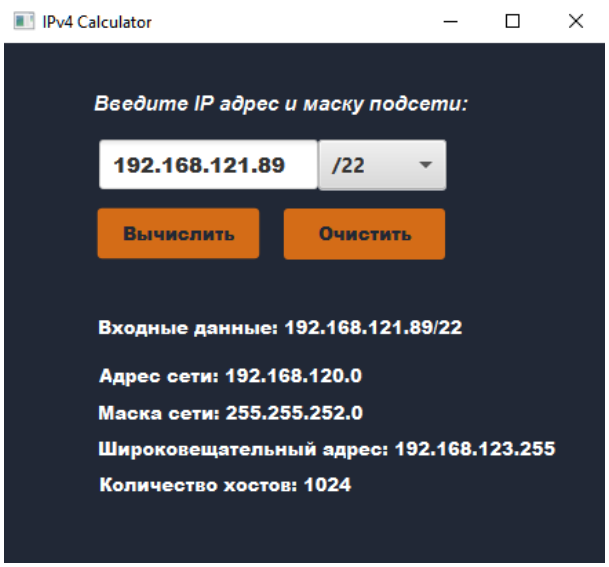
Метод countHosts() вычисляет количество хостов. На вход подается маска сети в префиксной форме. Количество хостов вычитывается по формуле $2^{32-\text{маска}}$. Если маска равна 32, то функция возвращает 0, в противном случае количество хостов.

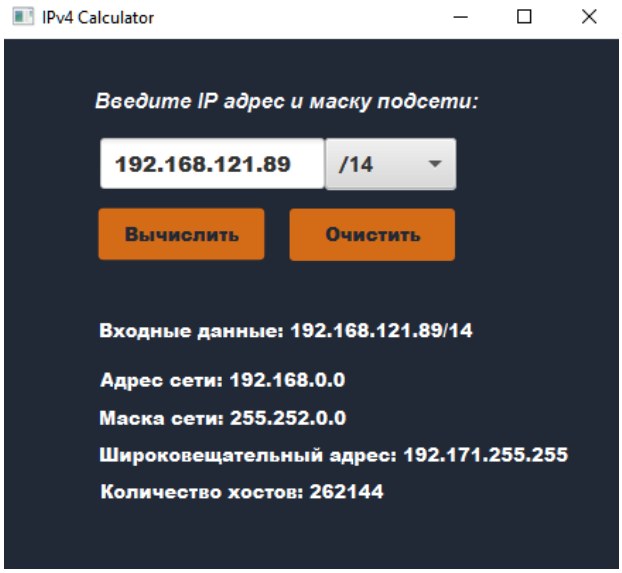
Рисунок 5 – Блок-схема подсчета количества хостов

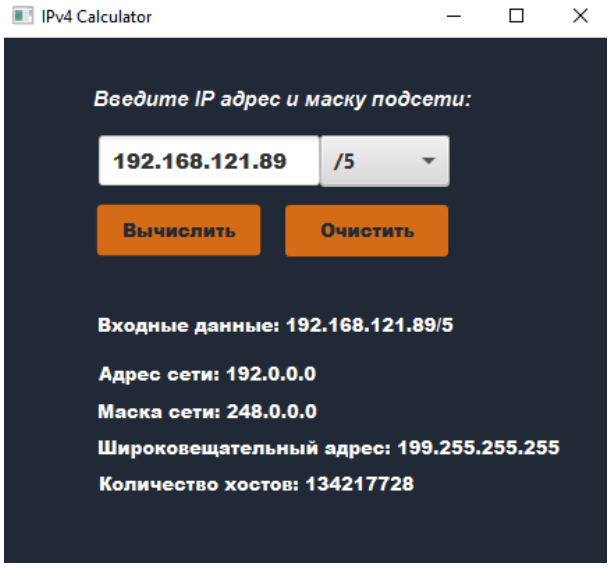
Тесты:

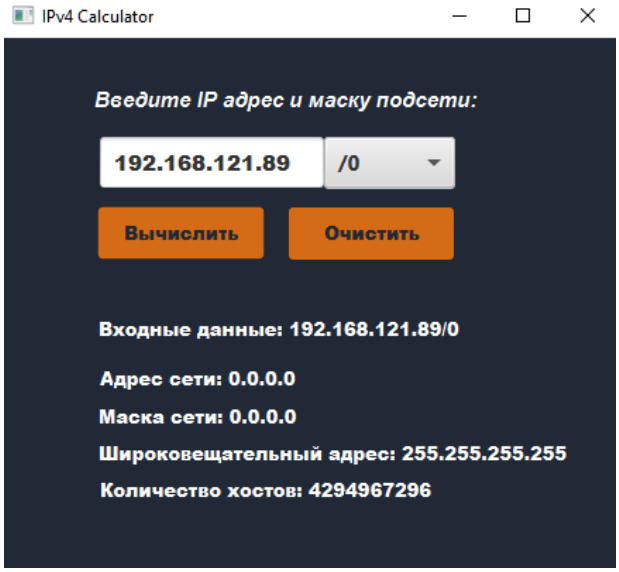












Вывод: в ходе лабораторной работы я изучил способы назначения адресации в стеке протоколов TCP/IP, получил навыки назначения адреса устройства и маски подсети.