Министерство образования и науки Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

Институт Компьютерных систем и информационной безопасности

ОТЧЕТ

по лабораторной работе №8

«Случайный доступ типа МДПН/ОС

(Множественный/многостанционный доступ с прослушиванием

несущей и обнаружением столкновений) и сеть Ethernet»

Дисциплина: Архитектура ЭВМ

Выполнил: студент 2 курса

группы 20-КБ-ПР2

Нажжар Каис А.М

Проверил: Ковтун А. А.

Краснодар

2021

1 Цель работы

Разработать алгоритм, реализующий случайный доступ типа МДПН/ОС. Сделать сравнительный анализ с результатами, полученными при алгоритмизации схем Алоха.

2 Задание

Разработайте программу, моделирующую работу сети типа МДПН/ОС со схемой с 1-настойчивостью (аналогичной сети Ethernet).Понятия максимального промежутка времени ожидания и максимальной границы первоначального случайного времени передачи выбрать так, чтобы алгоритм выполнялся наиболее быстро. Количество пакетов у каждой станции считать одинаковым. Входные данные для каждого варианта взять из предыдущей работы.

**3 Текст программы на алгоритмическом языке**

using System;

using System.Linq;

namespace lab8

{

class Program

{

class Station

{

public Station(int id, int packetCount)

{

this.id = id;

this.packetCount = packetCount;

}

public int packetCount = 0;

public int nextTime = 0;

public int id;

}

static void Main(string[] args)

{

Console.Title = "Lab 8";

Console.ForegroundColor = ConsoleColor.Green;

//Кол-во станций

int n = 13;

//Интенсивность передачи

int t = 3;

//Кол-во пакетов

int pockets = 2;

Console.WriteLine("Колличество станций " + n);

Console.WriteLine("Интенсивность передачи " + t);

Console.WriteLine("Колличество пакетов " + pockets);

Console.WriteLine();

Station[] stations = new Station[n];

for (int i = 0; i < n; i++) stations[i] = new Station(i, pockets);

Random rnd = new Random();

int globalTicks = 0;

while (!AllPacketsSended(stations))

{

Station stationToStartSending;

int numberStationsReadyToSend = GetNumberStationsReadyToSend(stations,

out stationToStartSending);

if (numberStationsReadyToSend == 1)

{

DecrementAllTimes(stations, t, ref globalTicks);

stationToStartSending.nextTime = 0;

stationToStartSending.packetCount--;

Console.WriteLine("Станция {0} отправила пакет! {1} осталось", stationToStartSending.id, stationToStartSending.packetCount);

}

else if (numberStationsReadyToSend > 1)

{

Console.WriteLine("Коллизия была обнаружена!");

foreach (Station t1 in stations)

{

if (t1.nextTime <= 0)

{

int next = rnd.Next(1, 10);

t1.nextTime = next;

Console.WriteLine("Станция {0} была задержанана {1} ticks", t1.id, next);

}

}

}

else if (numberStationsReadyToSend == 0)

{

Station stationWithNearestTime = FindStationWithNearestTime(stations);

Console.WriteLine("Путь к {0}",

stationWithNearestTime.nextTime);

DecrementAllTimes(stations,

stationWithNearestTime.nextTime,

ref globalTicks);

}

}

Console.WriteLine("Время: {0}", globalTicks);

Console.ReadKey();

}

static Station FindStationWithNearestTime(Station[] stations)

{

Station minStation = stations[0];

foreach (Station s in stations)

{

if (s.nextTime < minStation.nextTime && s.packetCount > 0)

minStation = s;

}

return minStation;

}

static void DecrementAllTimes(Station[] stations,

int time,

ref int globalTicks)

{

globalTicks += time;

for (int i = 0; i < stations.Count(); i++)

{

stations[i].nextTime -= time;

}

}

static int GetNumberStationsReadyToSend(Station[] stations,

out Station st)

{

st = null;

int count = 0;

foreach (Station s in stations)

{

if (s.nextTime <= 0 && s.packetCount > 0)

{

count++;

st = s;

}

}

if (count != 1) st = null;

return count;

}

static bool AllPacketsSended(Station[] stations)

{

foreach (Station s in stations)

{

if (s.packetCount != 0)

{

Console.WriteLine("Не все пакеты были отправлены");

return false;

}

}

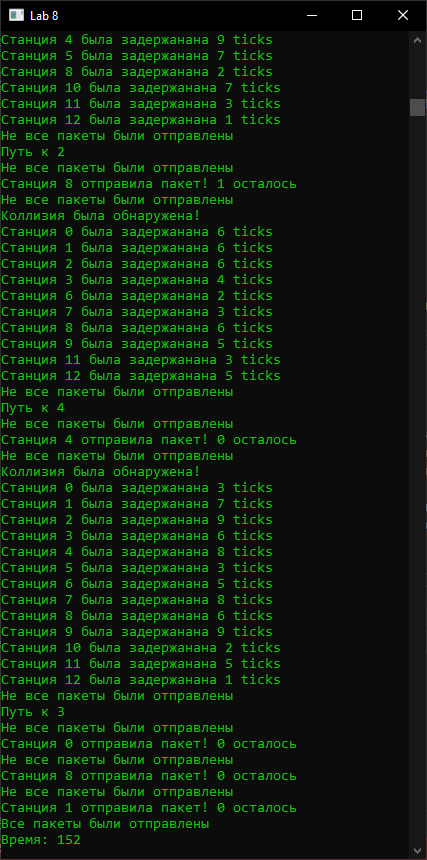
Console.WriteLine("Все пакеты были отправлены");

return true;

}

}

}

**4 Результат работы программы** 

На основании проведённых исследований можно сделать вывод, что данная система является отличным решением проблемы столкновения сообщений. При возникновении коллизий необходимо было реализовать способ повторно отправить сообщения таким образом, чтобы избежать ещё одной коллизии.