**Задание:**

ЛВС состоит из **сервера** и **8 рабочих станций**. Поток заявок от рабочих станций распределен по закону Пуассона со средним интервалом 15, 12, 17, 17, 20, 20, 30 секунд соответственно. Время обработки заявки распределено по экспоненциальному закону с таким же средним.

Сервер обрабатывает одновременно до 8 заявок.

На каждой рабочей станции создается очередь в 2 заявки, новые заявки от пользователей не принимаются рабочей станцией до освобождения места в очереди.

Повторно заявка посылается на сервер через промежуток времени, распределенный по нормальному закону со средним 20 секунд и отклонением 3 секунды.

Написать программу, моделирующую ЛВС.

Оценить загрузку сервера и рабочих станций за 8 часов работы.

Подсчитать общее количество обработанных и необработанных заявок.

**Код работы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace \_4labaKSIT

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

struct TClient

{

public float interval; //интервал (пауза)

public bool[] Free; //свободный

public float extratime; //дополнительное время

}

public struct TTaskOfServer

{

public int client\_number; //номер клиента

public float runtime; //время выполнения задачи

public bool free; //свободен

}

TTaskOfServer[] ServerQueue; //Очередь сервера

TClient[] ClientMass = new TClient[8]; //Массив клиентов

float[] Counter = new float[8]; //счётчик

int Complete, NoComplete; //выполнено, не выполнено заданий

int CurTime, AllTime; //время работы, всё время (указывается пользователем)

int NumberOfThreads; //количество потоков на сервере (указывается пользователем)

Random r = new Random();

public bool SendTaskToServer(TTaskOfServer t) //посылаем задачу на сервер

{

bool res = false;

for (int i = 0; i < NumberOfThreads; i++)

{

if (ServerQueue[i].free == true)

{

ServerQueue[i].client\_number = t.client\_number;

ServerQueue[i].runtime = t.runtime;

ServerQueue[i].free = false;

res = true;

break;

}

else

res = false;

}

return res;

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8) // цифры, клавиша BackSpace

{

e.Handled = true;

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && number != 8) // цифры, клавиша BackSpace

{

e.Handled = true;

}

}

void ModelNet()

{

int i, j;

TTaskOfServer buff; //для передачи парметров

while (CurTime != AllTime)

{

for (i = 0; i < NumberOfThreads; i++)

{

ServerQueue[i].runtime = ServerQueue[i].runtime - 1; //время выполнения для текущих задач

}

//генерация нового состояния сервера

for (j = 0; j < NumberOfThreads; j++)

{

if (ServerQueue[j].runtime <= 0)

{ ServerQueue[j].free = true;

Complete = Complete + 1;

}

};

//цикл добавления времени к ожиданиям клиентов

for (i = 0; i < 8; i++)

{

ClientMass[i].Free = new bool[2];

Counter[i] = Counter[i] + 1; //добавляем время

if ((ClientMass[i].interval + ClientMass[i].extratime) <= Counter[i])

{

//клиент готов к генерации запроса?

Counter[i] = 0;

ClientMass[i].extratime = 0;

buff.client\_number = i;

buff.runtime = ClientMass[i].interval;

buff.free = false;

if (SendTaskToServer(buff))

{

if (ClientMass[i].Free[1] == false)

{

ClientMass[i].Free[1] = true;

ClientMass[i].extratime = 0;

}

else

{

ClientMass[i].Free[0] = true;

ClientMass[i].extratime = 0;

}

}

else

{

if (ClientMass[i].Free[0] && ClientMass[i].Free[1])

{

ClientMass[i].extratime = 20 + (r.Next(3) - 6);

ClientMass[i].Free[0] = false;

}

else

{

if (!(ClientMass[i].Free[0]) && ClientMass[i].Free[1])

{

ClientMass[i].extratime = 20 + (r.Next(3) - 6);

ClientMass[i].Free[1] = false;

}

else

{

if (!(ClientMass[i].Free[0]) && !(ClientMass[i].Free[1]))

{

NoComplete = NoComplete + 1;

}

}

}

}

}

}

CurTime += 1;

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || textBox2.Text == "")

{

MessageBox.Show("Данные не введены");

}

else

{

Complete = 0; //выполненные задания

NoComplete = 0; //невыполненые задания

CurTime = 0; //время работы

AllTime = Convert.ToInt32(textBox2.Text) \* 3600; //перевод времени в секунды

//интервалы поступления сообщений

ClientMass[0].interval = 15;

ClientMass[1].interval = 12;

ClientMass[2].interval = 17;

ClientMass[3].interval = 17;

ClientMass[4].interval = 20;

ClientMass[5].interval = 20;

ClientMass[6].interval = 30;

ClientMass[7].interval = 30;

//очереди всех клиентов пусты

for (int i = 0; i < 8; i++)

{

ClientMass[i].Free = new bool[2];

ClientMass[i].extratime = 0;

ClientMass[i].Free[0] = true;

ClientMass[i].Free[1] = true;

};

//инициализация сервера

NumberOfThreads = Convert.ToInt32(textBox1.Text); //количество потоков

ServerQueue = new TTaskOfServer[NumberOfThreads]; //Потоки на сервере

//очередь сервера пуста

for (int i = 0; i < NumberOfThreads; i++)

{

ServerQueue[i].client\_number = 0;

ServerQueue[i].runtime = 0;

ServerQueue[i].free = true;

};

ModelNet(); //запуск сети

//вывод результатов работы

double Z, W, N;

Z = Complete + NoComplete;

W = Complete / Z \* 100;

N = 100 - W;

textBox3.Text = Convert.ToString(Z);

textBox4.Text = Convert.ToString(Complete);

textBox5.Text = Convert.ToString(NoComplete);

label1.Text = "(" + String.Format("{0:0.##}", W) + "%)";

label2.Text = "(" + String.Format("{0:0.##}", N) + "%)";

}

}

}

}

