МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

Факультет физико-математических и естественных наук
Кафедра теории вероятностей и кибербезопасности

«Допустить к защите»

Заведующий кафедрой
теории вероятностей
и кибербезопасности
д. т. н., профессор
_____ К. Е. Самуйлов
«___» _____ 20__г.

Выпускная квалификационная работа бакалавра

Направление 02.03.02 «Фундаментальная информатика и информационные технологии»

Тема «Название работы»

Выполнил студент Тагиев Байрам Алтай оглы

Группа НФИбд-02-20
Студенческий билет № 1032200531

Руководитель выпускной
квалификационной работы
профессор кафедры
теории вероятностей
и кибербезопасности
д. ф.-м. н., профессор
Д. С. Кулябов

_____

Автор _____

Москва
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

Аннотация

выпускной квалификационной работы

Тагиева Байрама Алтай оглы

на тему: Название работы

Здесь приводится текст аннотации.

Автор ВКР _____ Тагиев Байрам Алтай оглы

Содержание

Введение

Актуальность темы

Текст

Цель работы:

Текст

Краткое содержание работы

Текст

Глава 1. Название главы

1.1. Название секции

Фильтрующий маршрутизатор фильтрует IP-пакеты на основе групп следующих полей заголовка пакета:

- IP-адрес отправителя;
- IP-адрес получателя;
- порт отправителя;
- порт получателя.

На всю литературу надо ссылаться [1; 2].

1.2. Название секции

На рисунке 1.1 представлена упрощенная схема построения современного МЦОВ.

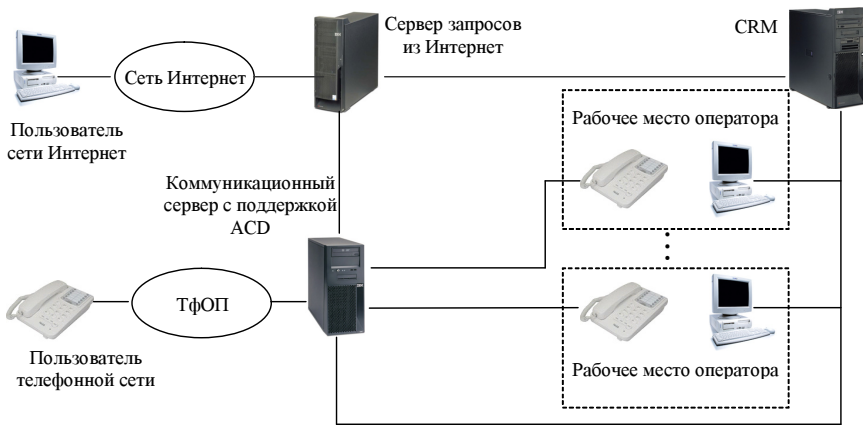

Рис. 1.1.. Упрощённая схема построения МЦОВ

На рисунке 1.2 представлен граф интенсивностей переходов для рассматриваемой СМО с параметрами $c = 2$ и $r = 3$.

На рис. 1.3 представлены графики зависимости π_1 от ρ_1 для различных μ_2.

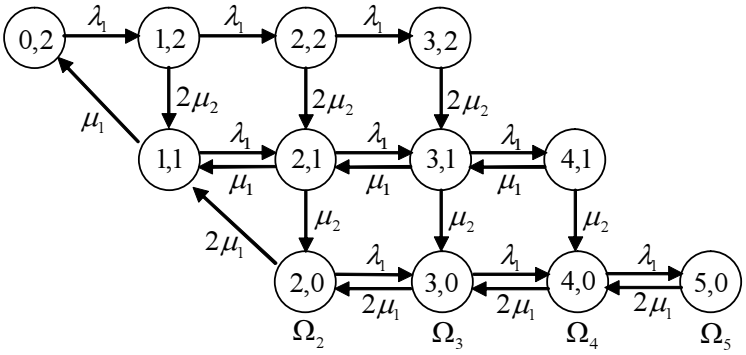

Рис. 1.2.. Граф интенсивностей переходов

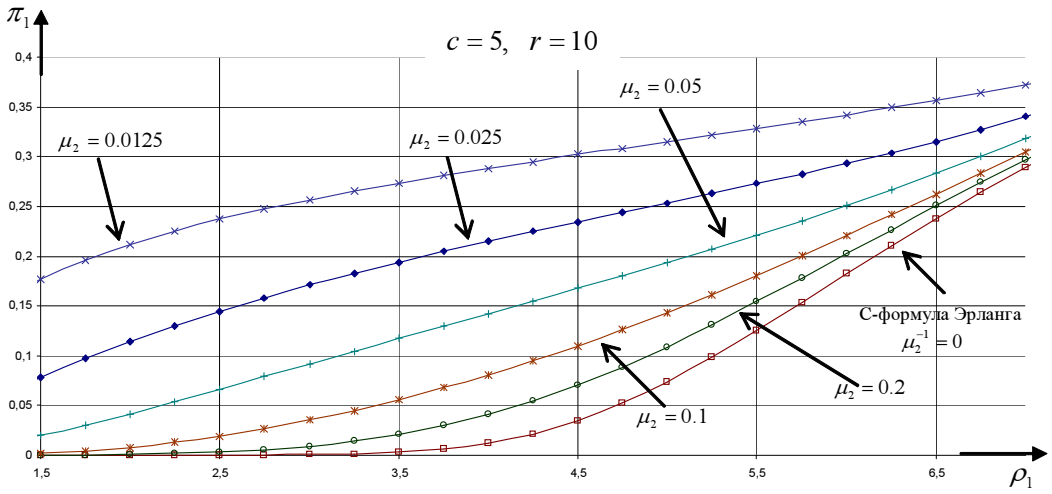

Рис. 1.3.. Зависимость π_1 от ρ_1 для различных μ_2

1.3. Название секции

Введём два случайных процесса $X_1(t)$ — суммарное количество 1-вызовов на приборах и в накопителе, $X_2(t)$ — количество 2-вызовов на приборах в момент времени t, $X_1(t) = \overline{0, R}$, $X_2(t) = \overline{0, c}$, $X_\bullet(t) = \overline{c, R}$. Тогда функционирование системы может быть описано ступенчатым Марковским процессом $\overrightarrow{X}(t) = (X_1(t), X_2(t))$ со следующим пространством состояний:

$$\Omega = \coprod_{\alpha=c}^{R} \Omega_\alpha, \quad \Omega_\alpha = \{(i, j) : i + j = \alpha\}, \quad \alpha = \overline{c, R}.$$

Глава 2. Название главы

2.1. Название секции

А теперь попробуем сравнить стоимость нашей реализации со стоимостью обыкновенного дисплейного класса (сервер в обычном ДК используется только как хранилище информации). Рассмотрим таблицу 2.1

Таблица 2.1.

Сравнительная стоимость ДК на основе обычных ПК и X-терминалов

Тип	Комплектация	Стоимость	Полная стоимость (20 шт)
Стандартный компьютер	Pentium IV, ОЗУ 512, диск 40Гб, видеокарта Radeon 8700	10000 руб	200000 руб
X-терминал	Pentium II, ОЗУ 128 (можно меньше), диск 1 ГБ (можно меньше), видеокарта Radeon 8700	менее 5000 руб	менее 100000 руб

2.2. Название секции

Текст.

2.3. Название секции

Текст.

Глава 3. Название главы

3.1. Название секции

Для этого на сервере был запущен виртуальный сервер `xserv`, с IP-адресом `10.130.64.15`:

```
vzctl create 3006  --os template gentoo-x86
vzctl set 3006  --name /xserv  --save
vzctl set 3006  --nameserver 10.130.64.15
vzctl start 3006
vzctl enter 3006
```

Запускаем ssh:

```
/etc/init.d/sshd start
```

Добавим запуск демона ssh по умолчанию:

```
rc-update add sshd default
```

Далее запускаем NX-сервер:

```
nxserver --start
```

Если все в порядке, появляется сообщение:

```
NX> 100 NXSERVER~--- Version 1.4.0-44 OS (GPL)
      NX> 122 Service started
      NX> 999 Bye
```

3.2. Название секции

Текст.

3.3. Название секции

Текст.

Заключение

Текст.

В работе было рассмотрено:

1. Принципы работы тонких клиентов, различные способы организации системы тонких клиентов

2. Сделан обзор продуктов компании NX NoMachine, а также проекта FreeNX, созданного на основе открытых библиотек NX, выделены их преимущества

3. Произведен сравнительный анализ стоимости различных конфигураций дисплейных классов, сделан вывод в пользу класса на основе X-терминалов.

4. Произведено тестовое подключение компьютера с установленным на нем клиентом NX к FreeNX серверу, а также запуск на нем приложений с оценкой скорости их работы. Скорость работы оказалась вполне приемлемой.

Итог: разработанный нами метод развертывания системы X-терминалов рекомендуется к применению в государственных и коммерческих учреждениях ввиду обеспечиваемого им снижения затрат на организацию и администрирование.

Список литературы

1. *Медведовский И. Д.*, *Семьянов П. В.*, *Платонов В. В.* Атака через Internet. — М. : НПО Мир и семья-95, 1997.
2. *Романец Ю. В.*, *Тимофеев П. А.*, *Шаньгин В. Ф.* Защита информации в компьютерных системах и сетях.

Приложение A. Название первого приложения

A.1. Название секции

Текст.

A.2. Название секции

Текст.

Приложение B. Название второго приложения

B.1. Название секции

Текст.

B.2. Название секции

Текст.

В данном приложении представлен исходный код программы для решения стохастических дифференциальных уравнений, написанный на языке C с использоваеним библиотеки GSL.

Приложение C. Заголовочный файл diffur.h

```
/*
    Name: Header file for SDE computing
    Author: Andrew "Atcher" Tchernoivanov
            tchernoivanov@gmail.com
    Copyright: Raccoon Programming Division
*/

# include <stdio.h>
# include <math.h>
# include <gsl/gsl_errno.h>
# include <gsl/gsl_matrix.h>
# include <gsl/gsl_odeiv.h>

# define N 1 // Количество узлов
# define Q_MAX 60 // Максимальное пороговое значение пакетов для
   ↪ алгоритма RED
# define Q_MIN 20 // Минимальное пороговое значение пакетов для
   ↪ алгоритма RED
# define W_MAX 32 // Максимальный размер TCPокна-
# define R 100 // Размер буфера
# define Tp 0.01 // Время прохождения пакета от источника до узла
# define wq 0.0007 // Вес очереди
# define delta 0.01 //
# define C_SMALL 1600 // Количество обслужеваемых за 1 секунду пакетов

double Q_TR_L;
double Q_TR_R;
double Q_TR;
double ALPHA;
double BETA = 0.9;
double BETA_powared = 5; // Compress factor
double P_MAX = 0.1;  // Максимальная вероятность сброса
int K = 3; // Possible values are - 2,3,4

// Описываем нашу индикаторную функцию
double tau (double x)
```

```
{
 if (x > 0.0 )
  return 1.0;
 else
  return 0.0;
}

// Описываем функцию T
double T (double x)
{
 return (Tp+x/(double)C_SMALL);
}

// Описываем функцию C
double C (double x)
{
 if (C_SMALL < x)
  return (double)C_SMALL;
 else
  return x;
}

// Задаем функцию вычисления вероятности сброса
double p_RED (double x) // RED
{
  double p1,p2;

 if ((0.0 <= x) && (x < (double)Q_MIN))
  return 0;
 else if (x > (double)Q_MAX)
  return 1;
 else
  {
   p1 = (double)(x -(double)Q_MIN);
   p2 = (double)((double)Q_MAX - (double)Q_MIN);
   return (double)((p1/p2)*P_MAX);
  }
```

```
}

double p_ARED (double x) // ARED
{
  double p, p1, p2;

// Computing P_MAX
  ALPHA = fmin(0.01, P_MAX/4);

  if ((x > Q_TR) && (P_MAX <= 0.5))
    p = P_MAX+ALPHA;
  else if ((x <= Q_TR) && (P_MAX >= 0.01))
    p = P_MAX*BETA;

  if (p<0.01) P_MAX=0.01;
  if (p>0.5) P_MAX=0.5;
  P_MAX = p;

// Computing p
  if ((0.0 <= x) && (x < (double)Q_MIN))
    return 0;
  else if (x > (double)Q_MAX)
    return 1;
  else
   {
     p1 = (double)(x -(double)Q_MIN);
     p2 = (double)((double)Q_MAX - (double)Q_MIN);
     return (double)((p1/p2)*P_MAX);
   }
}

double p_RARED (double x) // RARED
{
    int a,b;
    double c,d,p,p1,p2;
// Computing P_MAX
    if ((x > Q_TR) && (P_MAX <= 0.5))
```

```
    {
        a = Q_TR - x;
        c = (double) a / (double) Q_TR;
        d = c*P_MAX;
        ALPHA = 0.25 * d;
        p = P_MAX + ALPHA;
    }
    else if ((x <= (double) Q_TR) && (P_MAX >= 0.01))
    {
        a = Q_TR - x;
        b = (int)Q_TR - (int)Q_MIN;
        c = (double)a / (double)b;
        d = 0.17*c;
        BETA = 1 - d;
        p = P_MAX * BETA;
    }
    if (p<0.01) P_MAX = 0.01;
    if (p>0.5) P_MAX = 0.5;
    P_MAX = p;
// Computing P
    if ((0.0 <= x) && (x < (double)Q_MIN))
       return 0;
    else if (x > (double)Q_MAX)
       return 1;
    else
     {
     p1 = (double)(x -(double)Q_MIN);
     p2 = (double)((double)Q_MAX - (double)Q_MIN);
     return (double)((p1/p2)*P_MAX);
     }
}

double p_POWARED (double x) //POWARED
{
    double p, sigma, dev, p1,p2,p3,p4,p11,p21;
    //Computing p_max
    dev = x - Q_TR;
```

```
    if (dev < 0)
    {
        p1 = (double)dev / (double)Q_TR;
        p3 = p1 / (double) BETA;
        p4 = pow(p3,K);
        sigma = fabs(p4);
        p = P_MAX-sigma;
        if ( p < 0) p = 0;
        P_MAX = p;
    }
    else if (dev > 0)
    {
        p11 = (double) R - (double)Q_TR;
        p1 = dev / p11;
        p3 = p1 / (double) BETA;
        p4 = pow(p3,K);
        sigma = fabs(p4);
        p = P_MAX+sigma;
        if ( p > 1 ) p = 1;
        P_MAX = p;
    }
    else if (dev == 0) p = P_MAX;
//Computing p
    if ((0.0 <= x) && (x < (double)Q_MIN))
       return 0;
    else if (x > (double)Q_MAX)
       return 1;
    else
     {
     p1 = (double)(x -(double)Q_MIN);
     p2 = (double)((double)Q_MAX - (double)Q_MIN);
     return (double)((p1/p2)*P_MAX);
     }
}

double W_Reno_RED (double y[])
{
```

```
  return (double)((tau((double)W_MAX-y[0])*(1/T(y[1])))
  +(-((y[0])/2)*(y[0]/T(y[1])))*p_RED(y[2]));
}

double W_Reno_ARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])*(1/T(y[1])))
  +(-((y[0])/2)*(y[0]/T(y[1])))*p_ARED(y[2]));
}

double W_Reno_RARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])*(1/T(y[1])))
  +(-((y[0])/2)*(y[0]/T(y[1])))*p_RARED(y[2]));
}

double W_Reno_POWARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])*(1/T(y[1])))
  +(-((y[0])/2)*(y[0]/T(y[1])))*p_POWARED(y[2]));
}

double W_FReno_RED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])/T(y[1]))
  + (-(((y[0]))*((y[0]))/2)*(1-p_RED(y[2]))
  *p_RED(y[2]) / T(y[1]))+ ((y[0])*(1-(y[0]))*(p_RED(y[2])
  *p_RED(y[2])) / T(y[1])));
}

double W_FReno_ARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])/T(y[1]))
  + (-(((y[0]))*((y[0]))/2)*(1-p_ARED(y[2]))
 *p_ARED(y[2]) / T(y[1]))+ ((y[0])*(1-(y[0]))
 *(p_ARED(y[2])*p_ARED(y[2])) / T(y[1])));
}
```

```
double W_FReno_RARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])/T(y[1]))
  + (-(((y[0]))*((y[0]))/2)*(1-p_RARED(y[2]))
  *p_RARED(y[2]) / T(y[1]))+((y[0])*(1-(y[0]))
  *(p_RARED(y[2])*p_RARED(y[2])) / T(y[1])));
}

double W_FReno_POWARED (double y[])
{
  return (double)((tau((double)W_MAX-y[0])/T(y[1]))
  + (-(((y[0]))*((y[0]))/2)*(1-p_POWARED(y[2]))
  *p_POWARED(y[2]) / T(y[1]))+((y[0])*(1-(y[0]))
  *(p_POWARED(y[2])*p_POWARED(y[2])) / T(y[1])));
}

double Q_RED (double y[])
{
  return (double)(-C(y[1])+(tau((double)R-y[1]))
  *(y[0]/T(y[1]))*(1-p_RED(y[2]))*N);
}

double Q_ARED (double y[])
{
  return (double)(-C(y[1])+(tau((double)R-y[1]))
  *(y[0]/T(y[1]))*(1-p_ARED(y[2]))*N);
}

double Q_RARED (double y[])
{
  return (double)(-C(y[1])+(tau((double)R-y[1]))
  *(y[0]/T(y[1]))*(1-p_RARED(y[2]))*N);
}

double Q_POWARED (double y[])
{
```

```
  return (double)(-C(y[1])+(tau((double)R-y[1]))
  *(y[0]/T(y[1]))*(1-p_POWARED(y[2]))*N);
}

double Qe (double y[])
{
  return (double)(((log(1-wq)/delta)*y[2])
  -((log(1-wq)/delta)*y[1]));
}

int func (double t, const double y[], double f[], void *params)
{
  f[0]=W_FReno_ARED(y);
  f[1]=Q_ARED(y);
  f[2]=Qe(y);

 return GSL_SUCCESS;
}
```

Приложение D. Файл diffur.c

```
/*
    Name: Main file for SDE computing
    Author: Andrew "Atcher" Tchernoivanov
            tchernovanov@gmail.com
    Copyright: Raccoon Programming Division
*/

# include <stdio.h>
# include <math.h>
# include <gsl/gsl_errno.h>
# include <gsl/gsl_matrix.h>
# include <gsl/gsl_odeiv.h>
# include "diffur.h"

int main ()
{
  // Задаем границы нашего временного интервала
 double t0 = 0.0, t1 = 200.0;
 // Задаем точку начала отсчета
 double t = t0;
 // и определяем желаемый шаг, с которым у нас будет вычисляться
   ↪ значения
 double h = 1e-3;

// Размерность системы
 int dim_ode = 3;

 // Векторстолбец-, задающий начальные условия
 double y[3] = {1.0, 0.0, 0.0};

 // Определяем метод, который будет использоваться для решения данной
   ↪ системы уравнений
 const gsl_odeiv_step_type *P = gsl_odeiv_step_rk4;

 // Программная: возвращает указатель на начало массива координат
 // для заданного шага и размерности системы
```

```
gsl_odeiv_step *s = gsl_odeiv_step_alloc (P,dim_ode);
// Программная: создание переменной, в которой будет храниться
// накопленная при вычислениях ошибка
gsl_odeiv_control *c = gsl_odeiv_control_y_new (h, t0);
// Программная: возвращает указатель на массив для
// заданной размерности системы
gsl_odeiv_evolve *e = gsl_odeiv_evolve_alloc (dim_ode);

// Определяем нашу общую систему уравнений, передавая
// func - указатель на нашу систему диффуров
// NULL - здесь указывается якобиан, если он есть
// dim_ode - размерность нашей системы уравнений
// NULL - дополнительные параметры, если имеются
gsl_odeiv_system sys = {func, NULL, dim_ode, NULL};

ALPHA = fmin (0.01, P_MAX/4);
Q_TR_L = (Q_MIN + (0.4*(Q_MAX-Q_MIN)));
Q_TR_R = (Q_MIN + (0.6*(Q_MAX-Q_MIN)));
Q_TR = (Q_TR_L + Q_TR_R) / 2;

// Запускаем наш таймер
while (t < t1)
{
  // Считаем значения нашей системы в заданный момент
  // времени при заданных условиях
 int status = gsl_odeiv_evolve_apply (e,c,s,&sys,&t,t1,&h,y);


 if (status != GSL_SUCCESS) // В случае ошибки
   break;                   // прерываем выполнение
 // Выдаем необходимые нам параметры
 printf ("%f %f %f %f\n", t, y[0], y[1], y[2]);
}

// Освобождаем память
gsl_odeiv_evolve_free (e);
gsl_odeiv_control_free (c);
```

```
 gsl_odeiv_step_free (s);

 exit (0);
}
```

Список иллюстраций

Список таблиц