

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И  
ОПТИКИ**

Факультет Компьютерных технологий и управления  
Кафедра вычислительной техники

**ОТЧЕТ**

**о практике**

Исследование сетевых технологий с помощью имитационного  
моделирования

**Студент**

Елькин А. А. группа 3105

**Руководитель практики**

Соснин В. В. доцент

2014 год

## Содержание

<b>1</b>	<b>Система компьютерной верстки <math>\text{\TeX}</math>(<math>\text{\LaTeX}</math>)</b>	<b>3</b>
1.1	$\text{\TeX}$ . . . . .	3
1.1.1	История $\text{\TeX}$ . . . . .	3
1.1.2	Особенности $\text{\TeX}$ . . . . .	3
1.2	$\text{\LaTeX}$ . . . . .	3
1.3	Достоинства и недостатки [3] . . . . .	3
1.4	Список выбранного ПО . . . . .	4
<b>2</b>	<b>Система контроля верси Git</b>	<b>5</b>
2.1	История Git . . . . .	5
2.2	Особенности Git . . . . .	5
2.3	Git-команды[5] . . . . .	5
<b>3</b>	<b>WiMAX</b>	<b>6</b>
3.1	История WiMAX . . . . .	6
3.2	Структура[8] WiMAX . . . . .	6
<b>4</b>	<b>Выполнение задания по теме</b>	<b>8</b>
4.1	ПО ns-3 . . . . .	8
4.1.1	Установка ns-3 . . . . .	8
4.1.2	Настройка . . . . .	8
4.2	Разбор первого примера . . . . .	9
	<b>Литература</b>	<b>12</b>

# 1 Система компьютерной верстки $\text{T}_\text{E}\text{X}(\text{L}\text{A}\text{T}_\text{E}\text{X})$

## 1.1 $\text{T}_\text{E}\text{X}$

### 1.1.1 История $\text{T}_\text{E}\text{X}$

$\text{T}_\text{E}\text{X}$  [1] — система компьютерной верстки, разработанная Дональдом Кнутом, которая предназначена для компьютерной верстки текста и математических формул. Кнут начал разрабатывать систему в 1977 году, и первая версия  $\text{T}_\text{E}\text{X}$  вышла в 1979 году. В 1982 году вышла заново переписанная версия  $\text{T}_\text{E}\text{X}$ 'а, которой было дано название  $\text{TeX82}$ . И с версии  $\text{T}_\text{E}\text{X}$  3.0, которая получила лучшую поддержку 8-битных символов и различных языков, используется нумерация: каждое обновление добавляет в конец номера версии десятичную цифру так, что бы она приближалась к числу  $\pi$ .

### 1.1.2 Особенности $\text{T}_\text{E}\text{X}$

В  $\text{T}_\text{E}\text{X}$  пользователь пишет текст и задает лишь структуру самого текста, а система сама формирует документ на основе выбранного шаблона. Для задания структуры используется собственный язык разметки  $\text{T}_\text{E}\text{X}$ 'а, все это содержится в файле с расширением `.tex`, и  $\text{T}_\text{E}\text{X}$  транслирует в файл `.dvi`.

$\text{T}_\text{E}\text{X}$  можно использовать для создания разных видов документов: книги, статьи, отчеты, письма и др.

## 1.2 $\text{L}\text{A}\text{T}_\text{E}\text{X}$

$\text{L}\text{A}\text{T}_\text{E}\text{X}$  [2] — макропакет компьютерной верстки  $\text{T}_\text{E}\text{X}$ . Он не добавляет возможности в  $\text{T}_\text{E}\text{X}$ , а лишь позволяет автоматизировать задачи набора текста (умерация разделов и формул, перекрестные ссылки, размещение таблиц и т. п.). Первую версию выпустил Лесли Лэмпорт в 1984 году. В 1994 году была выпущена вторая версия  $\text{L}\text{A}\text{T}_\text{E}\text{X}$  —  $\text{L}\text{A}\text{T}_\text{E}\text{X} 2_\epsilon$ , которая является текущей по сей день.

## 1.3 Достоинства и недостатки [3]

Среди достоинств можно выделить:

- Автор может не вникать в детали оформления документа, ему лишь надо задать логическую структуру текста.
- высокое качество и гибкость верстки абзацев и математических формул.

- $\text{\TeX}$  не требует большой вычислительной мощности.
- Система работает на большинстве платформах.

Среди недостатков можно выделить:

- Исходный текст не будет выглядеть так же как при печати.
- Создание нового макета документа очень трудоемкая задача.
- $\text{\TeX}$  плохо приспособлен для верстки страниц со сложным взаимодействием текста и графиков.

## 1.4 Список выбранного ПО

Для написания отчета по практике был выбран ряд программного обеспечения:

1. Сборка  $\text{\TeX}$ 'а MacTeX(<http://tug.org/mactex/>), включающий pdfLaTeX, который выдает документ с расширением pdf.
2. IDE Eclipse(<https://www.eclipse.org/>) с расширением TeXlipse(<http://texlipse.sourceforge.net/>), позволяющие удобно редактировать документ.

## 2 Система контроля верси Git

### 2.1 История Git

Git [4] — распределенная система контроля версиями. Причиной создания Git послужило ухудшение отношений между сообществом разработчиков Linux и компанией разработавшей BitKeeper, используемым сообществом с 2002 года для разработки Linux. Создателем проекта был Линус Торвальдс, и на сегодняшний день поддерживается Джунио Хамано.

### 2.2 Особенности Git

В отличие от других СКВ Git не хранит изменения файлов, а сохраняет слепок файла как она выгляид в данный момент, при чем, если файл не был изменен, он делает ссылку на ранюю сохраненную версию файла. Так же большинство операций с файлами происходит локально, то есть для просмотра истории изменения проекта, создания коммита можно не иметь доступа к Сети.

Git следит за целостностью данных, он вычисляет контрольную сумму(SHA-1 хеш), которая становится индексом данного файла. Данная система не позволяет изменять содержимое файлов или каталога.

### 2.3 Git-команды[5]

1. git help предоставляет список команд. Если использовать git -help <имя команды> даст справку об определенной команде.
2. git init создает каталог .git со всей необходимой информацией о репозитории.
3. git clone клонирует существующий репозиторий
4. git remote отображает уже подключенные репозитории
5. git remote add добавляет удаленный репозиторий
6. git remote rm удаляет ссылку на репозиторий
7. git add индексирует измененных файлом
8. git commit фиксирует измененные файлы
9. git rm удаляет файлы

Примеры ипользования приведенных команды находятся в каталоге "примеры Git команд"

### 3 WiMAX

WiMAX[6] — телекоммуникационная технология, предоставляющая универсальный беспроводной связи на большие расстояния для разных устройств. Сеть WiMAX представляет собой совокупность беспроводного сегмента, который описывается в стандарте IEEE 802.16, и базового сегмента, определенный спецификациями WiMAX-форума[7].

#### 3.1 История WiMAX

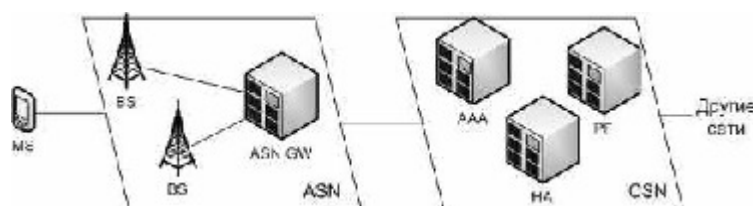
В декабре 2001 года была принята первая версия стандарта IEEE 802.16, который описывал организацию широкополосной беспроводной связи. Стандарт предусматривал скорость передачи информации 32-134 Мбит/с на расстояние в 2-5 км в радиоканалах шириной 20, 25 и 28 МГц.

Из-за того, что была необходимо построение беспроводной сети только в зоне прямой видимости, стандарт не получил широкого распространения. По этому в январе 2003 года был принято расширение 802.15а, который изменял используемые частоты на частоты в диапазоне от 2 до 11 ГГц. Это расширение должно было обеспечивать скорость передачи информации 1-75 Мбит/с на теоретическое возможное расстояние 50км(в основном 6-9 км).

Стандарт IEEE 802.16е приняли в 2005 году. Он поддерживает мобильных абонентов и систему роуминга между сетями различных беспроводных стандартов. 802.16е позволял без разрыва сеанса переключаться между стандартами 802.11 и 802.16.

#### 3.2 Структура[8] WiMAX

Сеть состоит из двух основных подсистем: Access Service Network(Сеть доступа) и Connectivity Service Network(Сеть обеспечения услуг).



Структура сети WiMAX

CSN определяется как набор функций для абонента сети:

- Распределение адресов между абонентами
- Доступ к сети Интернет

- Контроль доступа абонентов в сети
- Биллинг и межаператорское взаимодействие
- Обеспечение сервисов WiMAX
- Обеспечение аутентификации, авторизации и аудита соединения

CNS принадлежит оператору WiMAX, которому принадлежат маршрутизаторы, серверы для авторизации, аутентификации и аудита, базы данных пользователей, устройства преобразования сигнала.

ASN — набор сетевых элементов, организующие доступ в WiMAX сеть, выполняющие функции:

- Доступ абонентов в сеть по радиосвязи
- Передача сообщений между CSN и абонентом для обеспечения аутентификации, авторизации и аудита соединения
- Управление радиоресурсами
- Поиск абонента в сети
- Мобильность абонента

ASN представляет собой множество базовых станций, которые предоставляют доступ по радиосвязи по стандарту IEEE 802.16, и шлюза, предназначенного для объединения трафика от базовых станций и передачи его в CNS. ASN обязательно включает одну базовую станцию и один ASN-шлюз. Одной ASN могут пользоваться несколько провайдеров WiMAX(каждый со своей CNS).

Но так же не стоит забывать о большом количестве абонентов(мобильных станций). В качестве их могут быть мобильные телефоны, смартфоны, ноутбуки и другие устройства со встроенным или внешним адаптером.

## 4 Выполнение задания по теме

### 4.1 ПО ns-3

ns-3 — это дискретно-событийный симулятор сети, свободно распространяющийся под лицензией GNU GPLv2 license. Основное использование ns-3 фокусируется на беспроводном или IP моделирование, которое включает модели для Wi-Fi, WiMAX и LTE.

#### 4.1.1 Установка ns-3

Для установки потребовалось: Mercurial(hg)(<http://aragost.com/mercurial/>)-система контроля версии. GNU Compiler Collection(<http://gcc.gnu.org/>).

Для начала нужно было клонировать репозиторий, в котором размещались скрипты для скачивания и установки ns-3(скриншот downloadScripts.jpg).

```
hg clone http://code.nsnam.org/ns-3-allinone
```

Затем запускался скрипт для загрузки исходных файлов ns-3(скриншот downloadFileToInstall.jpg)

```
./download.py -n ns-3-dev
```

После этого запускался скрипт, запускающий сборку ns-3.(скриншот DoBuild.jpg)

```
./build.py
```

#### 4.1.2 Настройка

Настройка компилятора осуществляется при помощи команды(скриншот ns3ConfigGCC.jpg):

```
CXX=g++ ./waf configure
```

Настройка каталога для собранных проектов и включения учебных примеров:

```
./waf -d debug -o build/debug --enable-examples --enable-tests configure
```



## 4.2 Разбор первого примера

Для разработки модели в ns-3 используется язык программирования C++.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int main (int argc, char *argv[])
{
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    NodeContainer nodes;
    nodes.Create (2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

    NetDeviceContainer devices;
    devices = pointToPoint.Install (nodes);

    InternetStackHelper stack;
    stack.Install (nodes);

    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");

    Ipv4InterfaceContainer interfaces = address.Assign (devices);

    UdpEchoServerHelper echoServer (9);
```

```

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Simulator::Run ();
Simulator::Destroy ();
return 0;
}

```

Как видно из кода выше, при помощи *include* подключаются заголовочные файлы, в которых описаны нужные модули. На следующей строке *using namespace ns3;* обозначает то, что мы используем пространство имен ns3(ns3::).

Макрос *NS\_LOG\_COMPONENT\_DEFINE ("FirstScriptExample");* используется для последующей документации при помощи системы документации Doxygen.

*int main (int argc, char \*argv[])* является главной функцией и первой входной точкой программы, как для любой C/C++ программы. Это связано с тем, что для моделирования используется язык программирования C++(Код ns-3 просто программа на C++).

*NodeContainer nodes;* объявляет объект типа NodeContainer, а *nodes.Create (2);* создает 2 узла и добавляет ссылки на них в этот NodeContainer.

*PointToPointHelper pointToPoint; pointToPoint.SetDeviceAttribute ("DataRate StringValue ("5Mbps")); pointToPoint.SetChannelAttribute ("Delay StringValue ("2ms));*

Упрощает соединение и настройку PointToPointNetDevice и PointToPointChannel объектов. Так же устанавливает для устройства(объекта PointToPointNetDevice) атрибут "DataRate" значение 5Mbps и задержку в 2мс для каждого созданного канала(PointToPointChannel) впоследствии.

*NetDeviceContainer devices; devices = pointToPoint.Install (nodes);* Это позволяет настроить созданные нами узлы в начале программы и канал между ними (Параметры: скорость передачи данных 5 Мбит/с с задержкой в 2 мс).

*InternetStackHelper stack; stack.Install (nodes);* Дает TCP/UDP/IP функционал существующим узлам.

*Ipv4AddressHelper address; address.SetBase ("10.1.1.0 "255.255.255.0");* Создает объект *Ipv4AddressHelper*, которому сообщаем что он должен начать выделять адреса сети 10.1.1.0 используя маску 255.255.255.0. *Ipv4InterfaceContainer interfaces = address.Assign (devices);* присваивает адреса устройствам.

*UdpEchoServerHelper echoServer (9);* Создает серверное приложение которое ждет UDP-пакет и потом отправляет его обратно отправителю (Используется порт 9). *ApplicationContainer serverApps = echoServer.Install (nodes.Get (1)); serverApps.Start (Seconds (1.0)); serverApps.Stop (Seconds (10.0));* Устанавливаем приложение на узел 2 и задаем время начала и конца работы.

*UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);* Создает приложение которое посылает UDP-пакет и ждет пока придет его эхо. *echoClient.SetAttribute ("MaxPackets UintegerValue (1)); echoClient.SetAttribute ("Interval Time Value (Seconds (1.0)); echoClient.SetAttribute ("PacketSize UintegerValue (1024));* настройка параметров этого приложения: отправляет 1 пакет, интервал между пакетами 1 с и размер пакета 1024 бита.

*ApplicationContainer clientApps = echoClient.Install (nodes.Get (0)); clientApps.Start (Seconds (2.0)); clientApps.Stop (Seconds (10.0));* Устанавливаем приложение на узел 1, с началом работы через 2 секунды после начала симуляции, и конец после 10 секунд от начала симуляции.

*Simulator::Run ();* Запускает симуляцию. *Simulator::Destroy ();* Освобождение памяти от симулируемой модели.

## Литература

- [1] Tex - Википедия. <http://ru.wikipedia.org/wiki/TeX>.
- [2] Latex - Википедия. <http://ru.wikipedia.org/wiki/LaTeX>.
- [3] Львовский С. М. *Набор и верстка в системе LaTeX*. М.: МЦНМО, 2006.
- [4] Git - Википедия. <http://ru.wikipedia.org/wiki/Git>.
- [5] Git - book. <http://git-scm.com/book/ru>.
- [6] Wimax - Википедия. <http://ru.wikipedia.org/wiki/WiMAX>.
- [7] Wimax forum. <http://www.wimaxforum.org/>.
- [8] В. Вишневский С.Портной И.Шахнович. *Энциклопедия WiMAX путь к 4G*. Техносфера, Москва, 2009.