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

Факультет компьютерных технологий и управления
Кафедра вычислительной техники
Направление подготовки(специальность) 230100

О Т Ч Е Т
о практике

Тема задания: тема

Студент Бескоровайный Д.В., группа 3103

Руководитель практики Соснин В.В.

Оценка руководителя

Дата

Санкт-Петербург
2013г.

Содержание

1 TeX **3**
1.1 MiKTeX 4

2 Git **5**
2.1 Описание системы 5
2.2 Основные команды Git 9
2.2.1 Начальная настройка 9
2.2.2 Работа с репозиторием 9
2.2.3 Работа с удаленным сервером 10
2.2.4 Работа с метками 10
2.2.5 Работа с ветками 11

3 Симулятор NS-3 **12**
3.1 Ресурсы 14
3.1.1 Web 14
3.1.2 Mercurial 14
3.1.3 Waf 14
3.1.4 Среда разработки 15
3.2 Работа с NS-3 15
3.2.1 Обзор 15
3.2.2 Загрузка NS-3,использование Tarball. 15
3.2.3 Построение NS- 3 с build.py 16
3.2.4 Тестирование NS-3 17
3.2.5 Выполнение сценария 19
3.3 Описание Radio Mobile 20
3.4 Установка 21
3.4.1 Radio Mobile 21
3.4.2 NS-3-Wireless-planning 22
3.5 Построение 22
3.5.1 Карта и узлы 22
3.5.2 Сети и системы 24
3.5.3 Создание отчета 25
3.5.4 Преобразование сетевого отчета 27
3.5.5 Имитация сети 27

1 TeX

TEX — это созданная американским математиком и программистом Дональдом Кнутом (Donald E. Knuth) система для верстки текстов с формулами. Сам по себе TEX представляет собой машинно-независимый язык форматирования полиграфических документов. Автор TEX'а, профессор Станфордского университета США Дональд Кнут, определяет TEX как "инструмент для превращения набранного компьютерным образом манускрипта в документ, полиграфическое качество которого сопоставимо с тем, что дают самые современные печатающие устройства". Фундамент форматирующего инструмента TEX'а образуют более 300 команд-примитивов. Примитивы осуществляют операции нижнего уровня, неразложимые на более простые функциональные компоненты. На основе примитивов можно строить макрокоманды, которые могут иметь параметры и включать другие макрокоманды. Совокупность макрокоманд, подчиненных общим функциональным целям, объединяются в макронадстройки. В частности, LATEX — это созданная Лесли Лэмпортом (Leslie Lamport) издательская система на базе TEX'а. Latex не является монолитной программой и состоит из набора пакетов, причём набор пакетов не фиксирован, что позволяет создавать дистрибутивы, преследующие ту или иную цель.

Все издательские системы на базе TEX'а обладают достоинствами, заложенными в самом TEX'е. Из преимуществ данной системы следует отметить, что практически никакая другая из существующих в настоящее время издательских систем не может сравниться с TEX'ом в полиграфическом качестве текстов с математическими формулами. Также данная система обладает довольно низкими системными требованиями и на сегодняшний день реализована на всех современных компьютерных платформах,именно

благодаря этому TEX стал международным языком для обмена математическими и физическими статьями.

Однако у данной системы есть и недостатки. Среди них относительная сложность подготовки документов со сложным расположением материала на странице (для таких типов документов, практически не встречающихся в научно-технической литературе, TEX не предназначен).В добавок для TEX работа с исходным текстом и просмотр того, как текст будет выглядеть на печати, — разные операции,то есть чтобы посмотреть, как будет выглядеть на печати набираемый текст, надо запустить отдельную программу, что может увеличить временные затраты на подготовку документа.

1.1 MiKTeX

В Windows дистрибутив LATEX называется MikTeX.

Одним из существенных достоинств MiKTeX является возможность автоматического обновления установленных компонентов и пакетов. Особенности последних версии MiKTeX (2.7-2.8) является интегрированная поддержка XeTeX, MetaPost, pdfTeX и совместимость с Windows Vista, Windows Server 2008 и Windows 7.

В состав MiKTeX включены:

- классический TEX-компилятор;
- различные варианты TeX: pdfTeX, e-TeX, pdf-e-TeX, Omega, e-Omega, NTS;
- конверторы TeX в PDF;
- MetaPost(интерпретатор языка программирования META, который можно использовать для создания графических иллюстраций.);
- полный набор общеиспользуемых макропакетов: LaTeX, ConTeXt и др.;
- средство просмотра Yap;

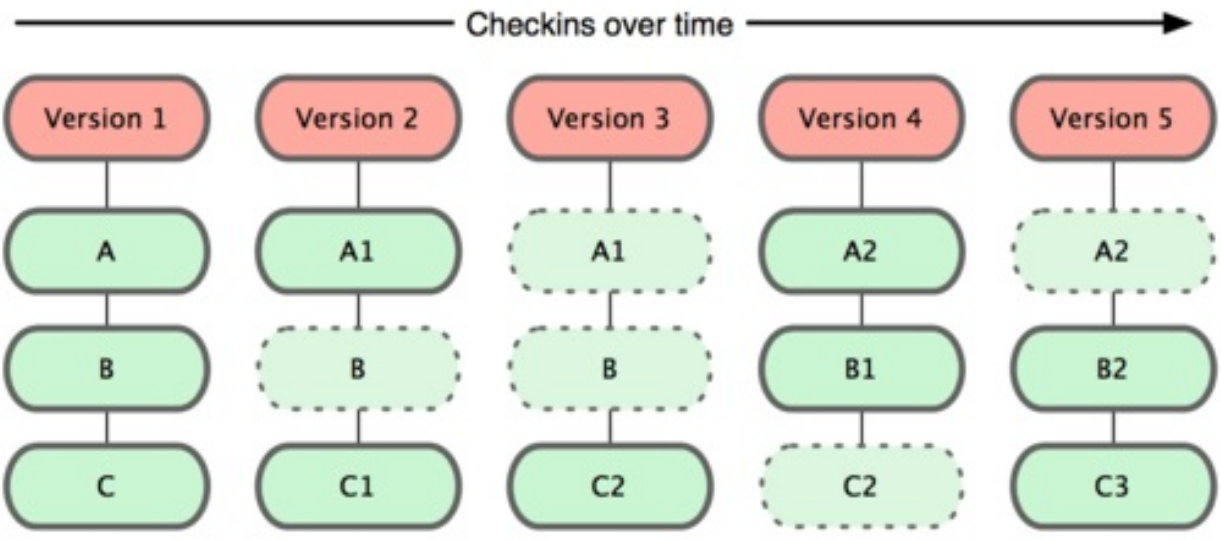
- инструменты и утилиты;

2 Git

2.1 Описание системы

Система контроля версий Git предназначена для контролируемого внесения изменений в процессе разработки программного обеспечения, а также при написании статей и другом роде деятельности, связанном с редактированием текстовых файлов (для удобства в дальнейшем будем обсуждать только тексты программ). Система позволяет отследить происходящие изменения в исходных текстах программ, что делает удобной совместную разработку.

Git ,в отличие от большинства систем контроля версий, хранит свои данные следующим образом. Он считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, Git сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет файл снова, а делает ссылку на ранее сохранённый файл. Иллюстрированный пример показан ниже.



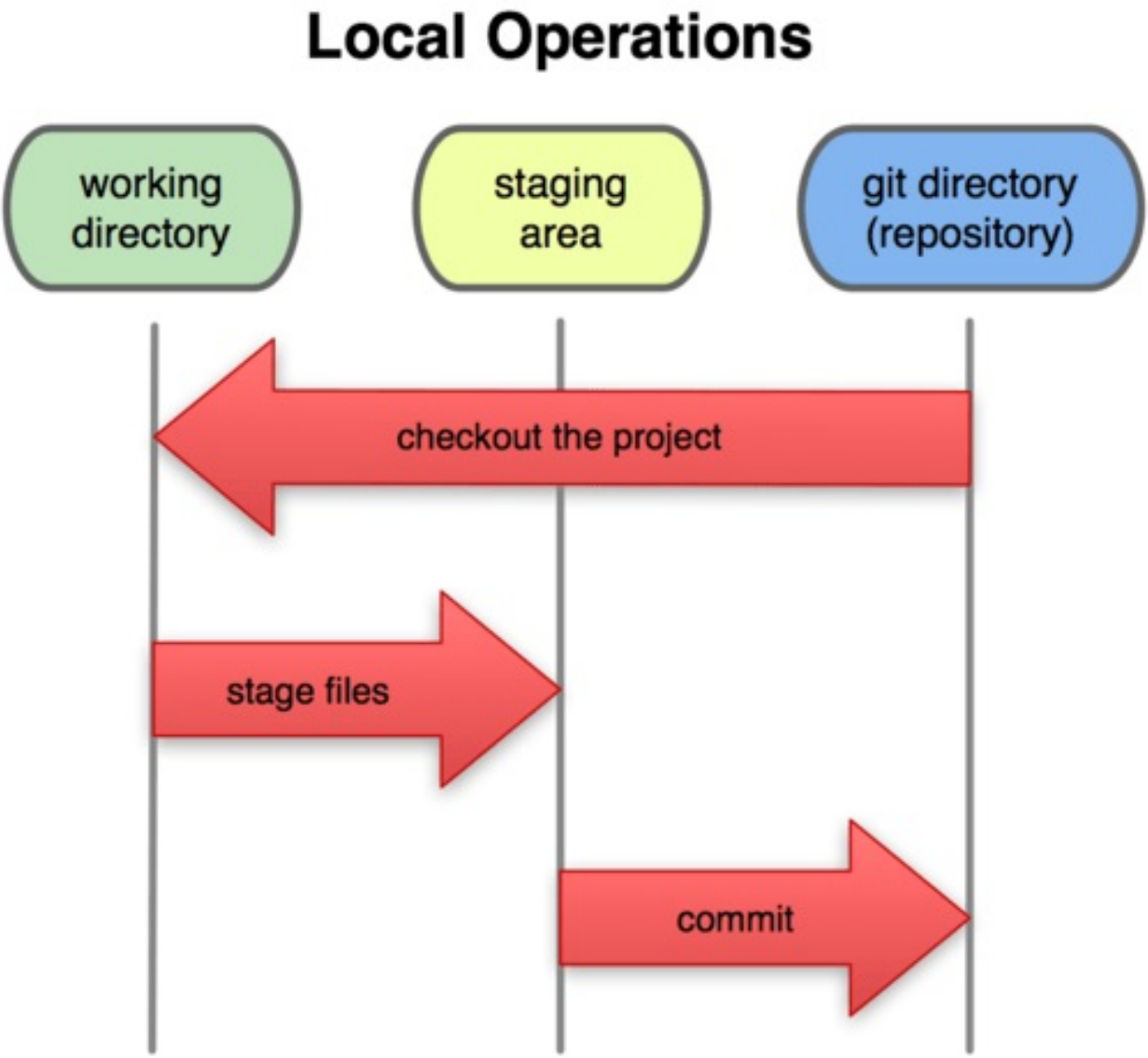
Для совершения большинства операций в Git'е необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна. Поскольку вся история проекта хранится локально у вас на диске, большинство операций кажутся практически мгновенными. К примеру, чтобы показать историю проекта, Git'у не нужно скачивать её с сервера, он просто читает её прямо из вашего локального репозитория. Поэтому историю вы увидите практически мгновенно. Если вам нужно просмотреть изменения между текущей версией файла и версией, сделанной месяц назад, Git может взять файл месячной давности и вычислить разницу на месте, вместо того чтобы запрашивать разницу у сервера или качать с него старую версию файла и делать локальное сравнение.

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит. Механизм, используемый Git'ом для вычисления контрольных сумм, называется SHA-1 хешем. Это строка из 40 шестнадцатеричных символов (0-9 и a-f), вычисляемая в Git'е на основе содержимого файла или структуры каталога. SHA-1 хеш выглядит примерно так: **24b9da6552252987aa493b52f8696cd6d3b00373**

Практически все действия, которые вы совершаете в Git'е, только добавляют данные в базу. Очень сложно заставить систему удалить данные или сделать что-то неотменяемое. Можно, как и в любой другой системе контроля версий, потерять данные, которые вы ещё не сохранили, но как

только они зафиксированы, их очень сложно потерять, особенно если вы регулярно отправляете изменения в другой репозиторий.

В проектах, использующих Git, есть три части: каталог Git'а (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).



Каталог Git'a — это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git'a, и именно она копируется, когда вы клонируете репозиторий с другого компьютера. Рабочий каталог — это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git'a и помещаются на диск для того, чтобы вы их просматривали и редактировали. Область подготовленных файлов — это обычный файл, обычно хранящийся в каталоге Git'a, который содержит информацию о том, что должно войти в следующий коммит. Стандартный рабочий процесс с использованием Git'a выглядит примерно так:

1. Вы вносите изменения в файлы в своём рабочем каталоге.

2. Подготавливаете файлы, добавляя их слепки в область подготовленных файлов.

3. Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог Git'a на постоянное хранение.

Если рабочая версия файла совпадает с версией в каталоге Git'a, файл считается зафиксированным. Если файл изменён, но добавлен в область подготовленных данных, он подготовлен. Если же файл изменился после выгрузки из базы данных, но не был подготовлен, то он считается изменённым. Существует множество различных клиентов для работы с Git,как имеющие графический интерфейс ,обеспечивающий удобство использования системы,так и представленных в виде командной строки,дающий немного большую гибкость. Далее представлены основные команды Git:

2.2 Основные команды Git

2.2.1 Начальная настройка

• Указать глобальное имя пользователя: **git config –global user.name "Den Bes"**

• Указать глобальный user email: **git config –global user.email bdvx2507@gmai**

• Указать редактор, который будет использоваться, когда нужно ввести сообщение в Git: **git config –global core.editor emacs**

• Указать утилиту сравнения, которая будет использоваться для разрешения конфликтов слияния: **git config –global merge.tool vimdiff**

• Просмотреть используемые настройки: **git config —list**

2.2.2 Работа с репозиторием

• Инициализировать репозиторий: **git init**

• Показать статус репозитория: **git status**

• Коммит с указанием комментария: **git commit -m 'some entry script fixes'**

• Коммит с указанием комментария и автоматической индексацией: **git commit -a -m 'some entry script fixes'**

• Добавить файл в индекс: **git add index.php**

• Посмотреть изменения между рабочим каталогом и индексом: **git diff**

• Посмотреть изменения между последним коммитом и индексом: **git diff –cached**

• Удалить файл из индекса и из рабочего каталога: **git rm readme.txt**

• Удалить файл из индекса, оставив его при этом в рабочем каталоге: **git rm –cached readme.txt**

• Исключить файл (который находится в индексе, но еще не за коммитен)

из индексации: **git reset HEAD readme.txt**

• Просмотр истории коммитов: **git log**

2.2.3 Работа с удаленным сервером

• Просмотреть какие удалённые серверы уже настроены, параметр -v отображает так же url сервера: **git remote –v**

• Добавить новый удалённый репозиторий под именем den: **git remote add den git:**

github.com/paulboone/ticgit.git

• Посмотреть список коммитов репозитория den ветки master, которые были выполнены после последнего pull-a: **git log den/master**

• Извлечь (fetch) всю информацию, которая есть в репозитории den: **git fetch den**

• Слить (merge) информацию, которую получили через fetch с рабочим каталогом: **git merge den/master**

• Отправить (push) код в ветку master удаленного репозитория den: **git push den master**

• Переименовать удаленный репозиторий из den на paul: **git remote rename den paul**

• Удалить удаленный репозиторий paul: **git remote rm paul**

2.2.4 Работа с метками

• Просмотр имеющихся меток: **git tag**

• Создание аннотированной метки: **git tag -a v1.4 -m 'my version 1.4'**

• Создание легковесной метки: **git tag v1.4**

• Просмотр данных метки: **git show v1.4**

• Создание метки для какого-либо существующего коммита (9fceb02 - контрольная сумма коммита, или ее часть): **git tag -a v1.2 9fceb02**

• Отправить метку на удалённый сервер: **git push den v1.5**

2.2.5 Работа с ветками

• Отобразить список веток проекта: **git branch**

• Отобразить список веток с последними коммитами: **git branch –v**

• Создать новую ветку с названием "br1": **git branch br1**

• Переключиться на ветку br1: **git checkout br1**

• Создать ветку с названием "br1"и переключиться на нее: **git checkout -b br1**

• Создать ветку с названием "articles"на основе ветки "articles"из удаленного сервера den и переключиться на нее: **git checkout -b articles den/articles**

• Слить текущую ветку с веткой hotfix: **git merge hotfix**

• Переименовать ветку "articles"в "content": **git branch -m articles content**

• Удалить ветку hotfix: **git branch -d hotfix**

• Удалить ветку hotfix на удаленном сервере den: **git push den :hotfix**

• Запустить графический инструмент для отображения конфликтных ситуаций: **git mergetool**

• Посмотреть список веток, которые уже слиты с текущей: **git branch –merged**

• Посмотреть список веток, которые содержат наработки, но еще не слиты с текущей: **git branch –no-merged**

Использованные материалы:

http://zelmanov.ptep-online.com/ctan/llang1992.pdf

http://www.intuit.ru/xml/course/LaTeX.pdf

http://ru.wikipedia.org/wiki/LaTeX

http://cluster.krc.karelia.ru/doc/rukovodstvo_GIT.pdf

3 Симулятор NS-3

NS3 является свободным программным обеспечением, распространяемым под лицензией GNU GPLv2, и ориентирован на исследовательское применение, а так же применение в образовательных целях. Исходные коды NS3 открыты для исследования, модификации и использования и доступны на сайте проекта http://www.nsnam.org.

Существует множество инструментов моделирования для исследований сетей . Ниже приведены несколько отличительных особенностей Ns -3:

• NS -3 выполнен в виде набора библиотек, которые могут быть объединены вместе, а также с другими внешними библиотеками программного обеспечения. Некоторые графические платформы и программы анализа данных могут быть использованы для работы с NS-3.Работа с ns-3 осуществляется в командной строке Ns C++ и / или Python.

• NS-3 в основном используется в системах Linux , хотя поддержка существует для FreeBSD , Windows , а также находится в процессе разработки для Microsoft Visual Studio.

Благодаря очень обширному и гибкому API, а так же благодаря полноте

документации программных интерфейсов, разработчик модели практически ничем не ограничивается. Ему предоставляется возможность построения как собственных моделей любой сложности, так и, благодаря используемой лицензии GNU GPLv2, изменение и дополнение уже существующих моделей, входящих в комплект ПО.

В NS3 разработаны модели беспроводных типов сетей, позволяющие проводить моделирование даже с движущимися объектами в трёхмерном пространстве. Разработаны модели для построения проводных топологий различной сложности, а также смешанных. Присутствует реализация различных типов Mesh-сетей на основе стека протоколов 802.11s, за счёт чего NS3 даёт фору даже многим коммерческим симуляторам. Разработан FrameWork под названием FlowMonitor, предоставляющий очень гибкие методы сбора самых различных показаний с моделируемых активных сетевых устройств и каналов связи. Симулятор не имеет собственного графического интерфейса, однако для средств визуализации моделей используются проекты NetAnimator и PyViz.

Работа над проектом NS3 не прекращается. Многие крупные компании опубликовали работы, в которых исследования основываются на NS3. Так же заявлена некоторыми компаниями и институтами разработка различных фреймворков для работы с симулятором. Ежедневно появляются новые всё более и более сложные модели, написанные членами сообщества. Данный симулятор способен удовлетворить потребности к имитационному моделированию современных телекоммуникационных систем, очень перспективен в разработке благодаря авторам проекта и сообществу, которые непрерывно улучшают проект, разрабатывая новые модели и исправляя старые ошибки. Он способен обеспечивать моделирование сетей следующего поколения, что обеспечивает возможность его использования в ближай-

шем будущем.

3.1 Ресурсы

3.1.1 Web

Есть несколько важных ресурсов, которые любой NS-3 пользователь должен знать. Основной сервер находится в http://www.nsnam.org и обеспечивает доступ к основной информации о системе NS-3. Подробная документация доступна через главный веб-сайт в http://www.nsnam.org/documentation/. Существует вики-проект, который дополняет основной веб-сайт по NS-3, который вы найдете в http://www.nsnam.org/wiki/.

3.1.2 Mercurial

Комплексные программные системы нуждаются в изменениях исходного кода и документации. Есть много способов для выполнения этого. Распределенная система контроля версий, вероятно, шим образом подходит для этих целей. Проект NS-3 использует Mercurial в качестве источника системы управления кодом . Mercurial имеет свой веб-сайт по адресу http://www.selenic.com/mercurial/ . Вы также можете найти важную информацию о совместном использовании Mercurial и NS-3 на главном веб-сайте NS-3.

3.1.3 Waf

Если ваш исходный код загрузился на локальную систему, вы должны будете скомпилировать код , чтобы получить полезную рабочую программу. Так же, как в случае управления исходным кодом есть много инструментов, доступных для выполнения этой функции. Вероятно, наиболее известным из этих инструментов является MAKE. Наряду с тем, самым известным, MAKE, вероятно, наиболее трудно используемый в очень большой и на-

страиваемой системе. Из-за этого, многие альтернативы были разработаны. В последнее время эти системы были разработаны с использованием языка Python. Система сборки Waf используется в NS- 3. Основной веб-сайт можно найти на http://code.google.com/p/waf/ .

3.1.4 Среда разработки

Как упоминалось выше, сценарии в NS- 3 написаны на C++ или Python. Большинство NS-3 API доступны в Python , но модели также написаны и для C++. Система NS- 3 использует некоторые компоненты GNU-инструментария для разработки. Для тех, кто работает под Windows , есть среды , которые имитируют среду Linux в различной степени .

3.2 Работа с NS-3

3.2.1 Обзор

NS-3 построена как система программных библиотек , которые работают вместе . Пользовательские программы написаны на языках C++ или Python. NS-3 распространяется в виде исходного кода , а это означает , что целевая система должна обладать средой разработки программного обеспечения для создания библиотеки , а затем создания пользовательской программы. NS- 3 может распространяться как готовые библиотеки для отдельных систем, однако в настоящее время многие пользователи работает путем самостоятельного редактирования NS-3.

3.2.2 Загрузка NS-3,использование Tarball.

Архив находится в определенном формате программного обеспечения, где несколько файлов связаны вместе и архив возможно сжат. Версии программного обеспечения NS-3 предоставляются через загружаемый архив.

Процесс загрузки NS-3 через архив прост: вы должны выбрать версию, скачать и распаковать ее.

cd

mkdir workspace

cd workspace

wget http://www.nsnam.org/releases/ns-allinone-3.17.tar.bz2

tar xjf ns-allinone-3.17.tar.bz2

3.2.3 Построение NS- 3 с build.py

Если Вы скачали архив ,у вас должен быть каталог с именем типа NS-AllInOne-3,17 в директории /workspace. Введите следующее:

./build.py –enable-examples –enable-tests

Вы увидите множество типичных выводов(сообщений)компилятора , отображаемых в виде сценария. В конце концов вы должны увидеть следующий текст:

Waf: Leaving directory

‘/path/to/workspace/ns-allinone-3.17/ns-3.17/build’

’build’ finished successfully (6m25.032s)

Modules built:

antenna aodv applications

bridge buildings config-store

core csma csma-layout

dsdv dsr emu

energy fd-net-device flow-monitor

internet lte mesh

mobility mpi netanim (no Python)

network nix-vector-routing olsr

point-to-point point-to-point-layout propagation

spectrum stats tap-bridge

test (no Python) tools topology-read

uan virtual-net-device wifi

wimax

Modules not built (see ns-3 tutorial for explanation):

brite click openflow

visualizer

Leaving directory './ns-3.17'

Regarding the portion about modules not built:

Modules not built (see ns-3 tutorial for explanation):

brite click openflow

visualizer

Это означает, что некоторые модули NS-3 , которые зависят от внешних библиотек , возможно, не были построены, или , что конфигурация специально попросила не строить их . Это не значит, что симулятор не построен успешно или что это обеспечит неправильные результаты для перечисленных модулей.

3.2.4 Тестирование NS-3

Вы можете запустить тесты NS-3 (проверку, что все настроено правильно), выполнив команду :

./test.py -c core

Эти тесты выполняются параллельно WAF. Должно вывести следующее:

92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)

Это важное сообщение. Вы также увидите вывод из тестера испытаний и вывод будет выглядеть примерно так:

Waf: Entering directory

‘/path/to/workspace/ns-3-allinone/ns-3-dev/build’

Waf: Leaving directory

‘/path/to/workspace/ns-3-allinone/ns-3-dev/build’

’build’ finished successfully (1.799s)

Modules built:

aodv applications bridge

click config-store core

csma csma-layout dsdv

emu energy flow-monitor

internet lte mesh

mobility mpi netanim

network nix-vector-routing ns3tcp

ns3wifi olsr openflow

point-to-point point-to-point-layout propagation

spectrum stats tap-bridge

template test tools

topology-read uan virtual-net-device

visualizer wifi wimax

PASS: TestSuite ns3-wifi-interference

PASS: TestSuite histogram

PASS: TestSuite sample

PASS: TestSuite ipv4-address-helper

PASS: TestSuite devices-wifi

PASS: TestSuite propagation-loss-model

...

PASS: TestSuite attributes

PASS: TestSuite config

PASS: TestSuite global-value

PASS: TestSuite command-line

PASS: TestSuite basic-random-number

PASS: TestSuite object

PASS: TestSuite random-number-generators

92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)

Эта команда обычно используется пользователями, чтобы быстро проверить, что NS- 3 успешно развернут.

3.2.5 Выполнение сценария

Как правило, мы запускаем скрипты под контролем Waf . Это позволяет системе сборки убедиться, что общие библиотечные пути установлены правильно и что библиотеки доступны во время выполнения. Для запуска программы , просто используйте –run в Waf . Давайте запустим программу NS-3,выводящую “hello world” введя следующее :

./waf –run hello-simulator

Waf сначала проверяет, чтобы убедиться, что программа построена правильно и выполняет построение, если требуется.Затем Waf выполняет программу, которая выдает следующий результат:

Hello Simulator

Если вы видите WAF сообщения, указывающие , что сборка была завершена успешно , но не видите "Hello Simulator есть вероятность, что вы

включили ваш билд режим "optimized"в "Building with Waf"раздел, но забыли вернуться к "Debug"режиму. Если вы не видите "Hello Simulator введите следующее:

./waf configure -d debug –enable-examples –enable-tests

Чтобы дать команду WAF создать отладочные версии NS -3 программы, которые включают примеры и тесты. Вы все еще должны построить текущую версию отладки кода , набрав :

./waf

Теперь, если вы запустите hello-simulator программы , вы должны увидеть соответствующий результат .

Для нашего исследования(симуляции) нам понадобятся такие программы как Radio Mobile (симулятор радиосистем) и NS-3,о котором ранее было рассказано.

3.3 Описание Radio Mobile

Radio Mobile является бесплатным инструментом для построения моделей радиосистем и прогнозирования их работы . Программа способна отрисовывать карты с использованием данных о местности, рельефа для автоматического извлечения профиля пути между излучателем и приемником.

Программное обеспечение также обеспечивает 3D-просмотр , стереоскопический вид и анимацию. Фоновая картинка может быть объединена с отсканированными картами или спутниковыми фотографиями. Radio Mobile имеет ряд серьезных недостатков : во-первых, это не свободно распространяемое ПО, но бесплатное, и исходный код не доступен. Во-вторых, она написана на Visual Basic , что исключает мультиплатформенность .

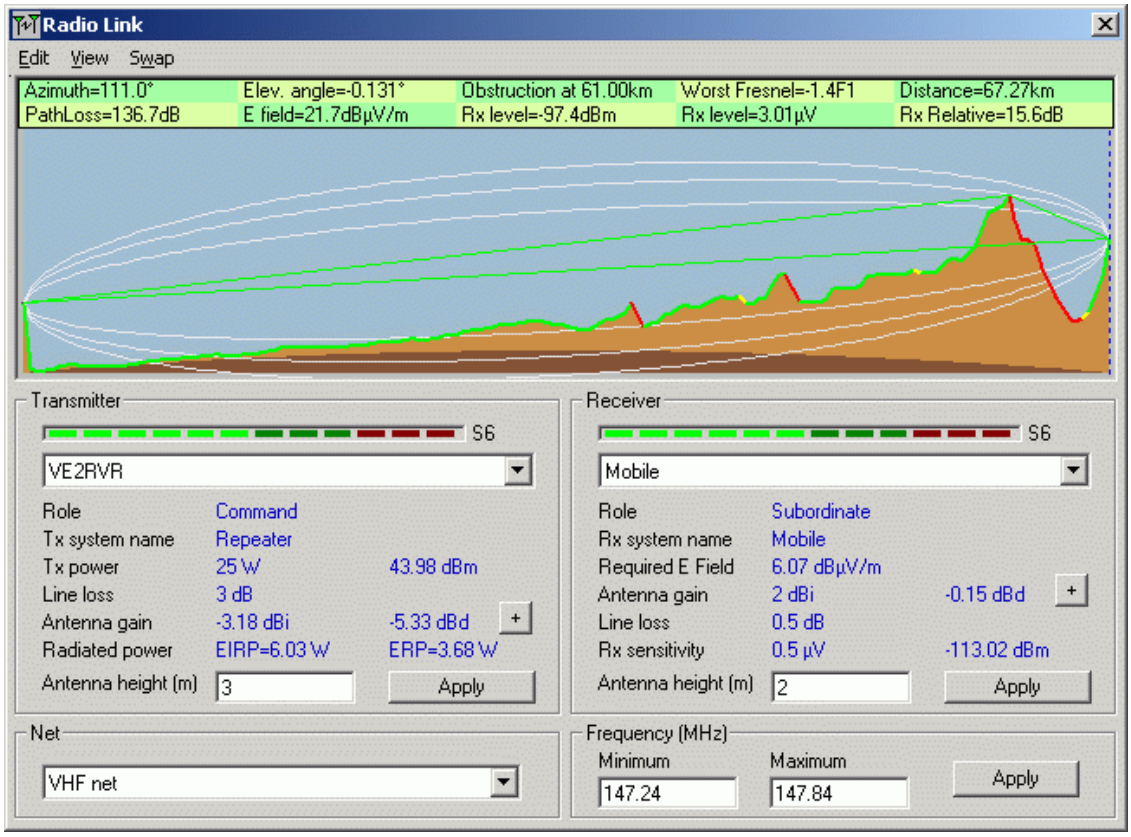


Рис. 1:

3.4 Установка

3.4.1 Radio Mobile

Устанавливаем wine :

apt − getinstallwine

Устанавливаем Radio Mobile в любую директорию используя команды:

cdpath/to/radiomobile/

winermweng.exe

3.4.2 NS-3-Wireless-planning

Скачать из источника:

$hgclonehttp://ns3-wireless-planning.googlecode.com/hg/ns3-wireless-planning$

3.5 Построение

Наша сеть будет беспроводной,и состоять из:

7 узлов(nodes,или хостов) (Josjojauarina1 , Josjojauarina2 , Ccatcca , Kcauri , Urpay , Huiracochan , Urcos).

4 подсети : Josjojauarina2 - Ccatcca - Kcauri (WiMAX) , Josjojauarina1 - Josjojauarina2 (WiFi), Josjojauarina1 - Urpay - Huiracochan (WiFi), Huiracochan - Urcos (WiFi) .

3.5.1 Карта и узлы

Сначала создайте узлы (в Radio Mobile именуемые units):

File -> Unit properties

И настройте свойства карты:

File -> Map properties

Теперь вы должны увидеть единицы, расположенные на карте:

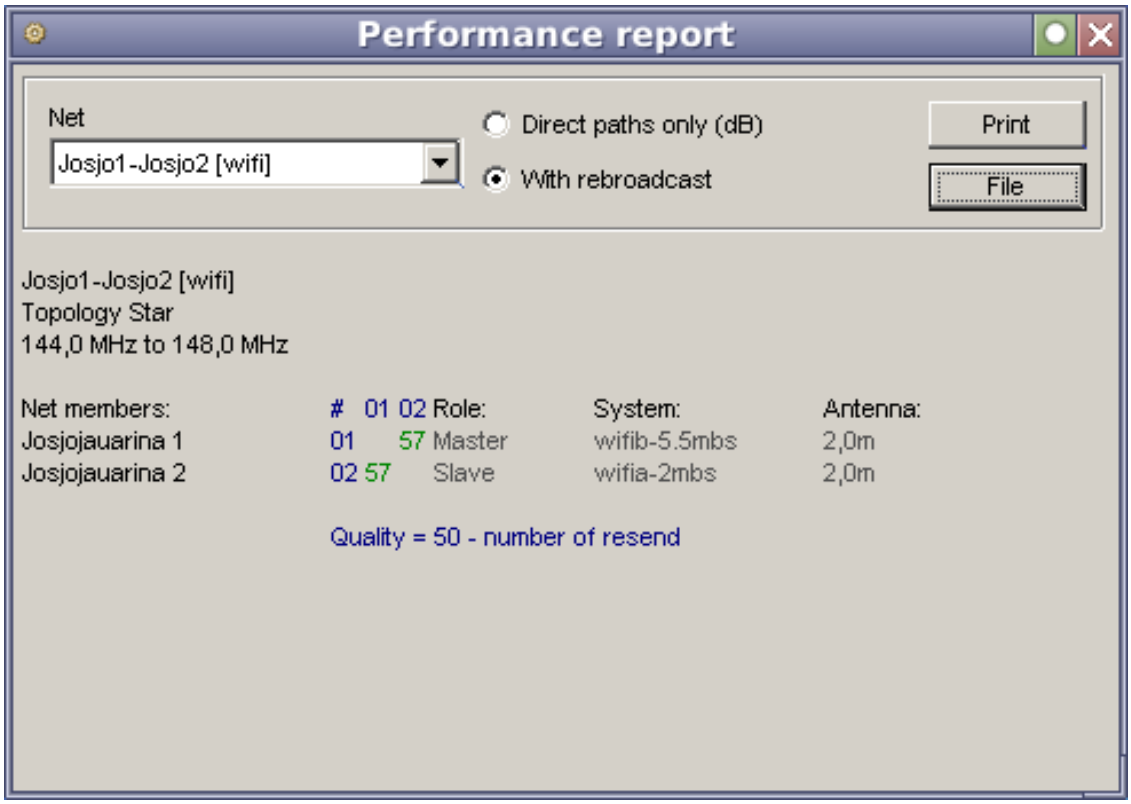

Performance report
Net
Josjo1-Josjo2 [wifi]
Direct paths only (dB)
With rebroadcast
Print
File
Josjo1-Josjo2 [wifi]
Topology Star
144,0 MHz to 148,0 MHz
Net members:
Josjojauarina 1
Josjojauarina 2
01 02 Role:
01 57 Master
02 57 Slave
System:
wifib-5.5mbs
wifia-2mbs
Antenna:
2,0m
2,0m
Quality = 50 - number of resend

Рис. 2:

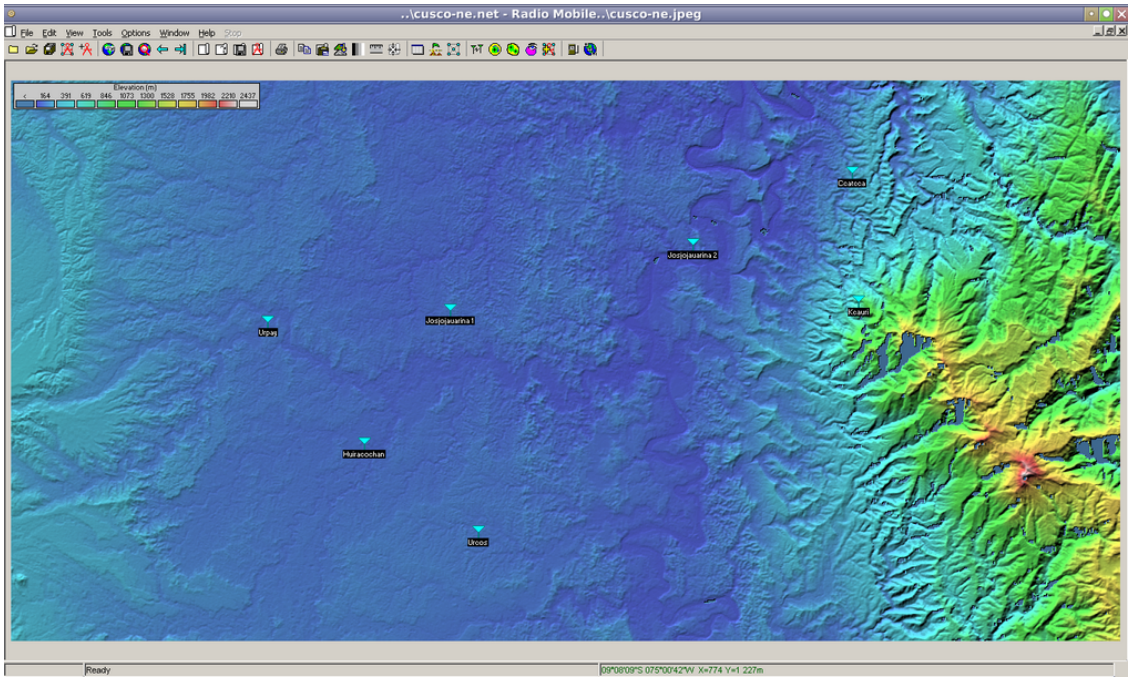

..\cusco-ne.net - Radio Mobile..\cusco-ne.jpeg
File Edit View Tools Options Window Help
Josjojauarina 2
Josjojauarina 1
Urpay
Huracochan
Ccatcca
Kayuni
Urcos
Ready

Рис. 3:

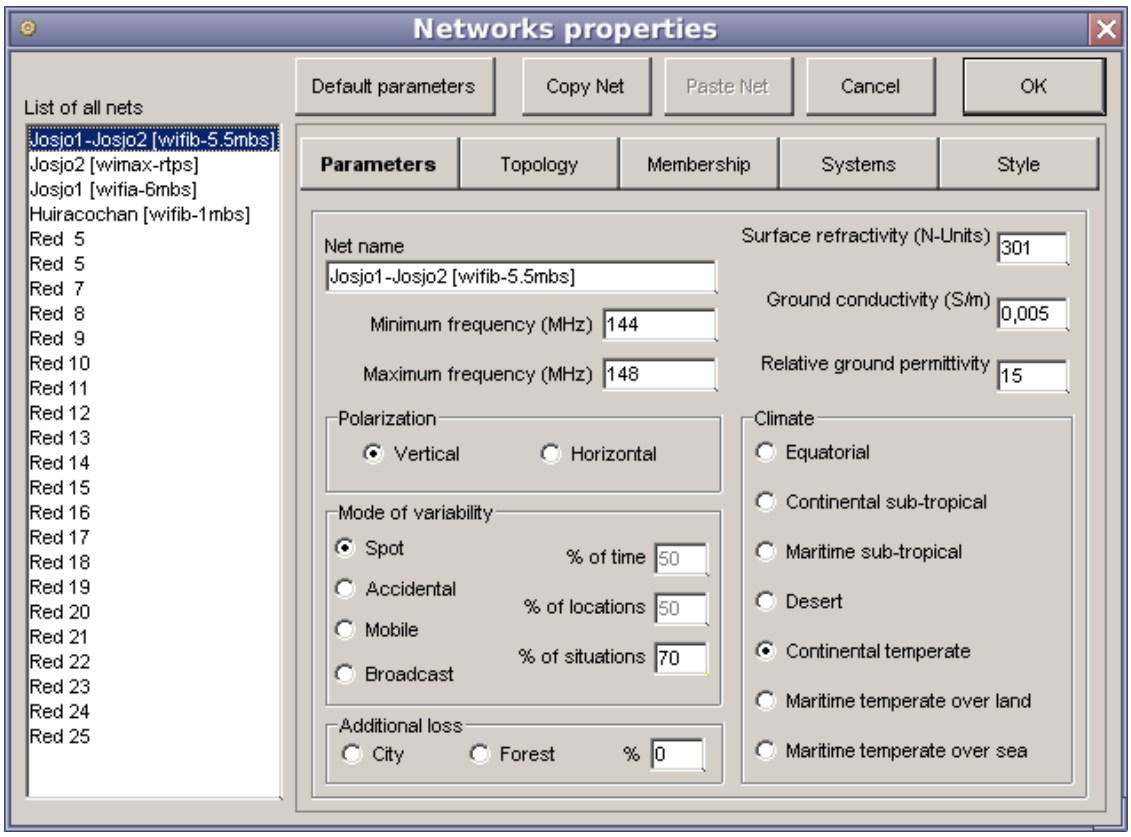


Рис. 4:

3.5.2 Сети и системы

Когда узлы помещаются на карту , мы должны указать подсети (ссылки) и системы (типы интерфейсов), которые они будут использовать.

Создаем четыре сети , с указанием типа сети (WiFi / WiMAX) в скобках имя сети .

File -> Network properties

Наименования режимов WiFi должны быть сокращены , потому что Radio Mobile сокращает названия до 20 символов. Используем эту таблицу для кодирования WiFi / WiMAX режимов :

Режим	Наименование для Radio Mobile
WiFi A 6Mbps	WFa6
WiFi A 12Mbps	WFa12
WiFi B 6Mbps	WFb1
WiFi B 12Mbps	WFb2
WiMAX BPSK 1/2	WXbk12
WiMAX QPSK 1/2	WXqk12
WiMAX QPSK 3/4	WXqk34
WiMAX 16 QAM 1/2	WX16qm12
WiMAX 16 QAM 3/4	WX16qm34
WiMAX 64 QAM 2/3	WX64qm23
WiMAX Base Station	WXall

Например, если вы создали систему под названием mysystem и вы планируете использовать ее с 802.11a при скорости 18 Мбит , вы задаете имя системы как "mysystem WFa18 ".

Наконец, необходимо определить топологию сети. Обозначьте роль точек доступа (Wi-Fi) или базовых станций (WiMAX) как Master,а роль Slave оставьте для станций WiFi и абонентских станций WiMAX :

3.5.3 Создание отчета

Извлеките всю необходимую информацию из сети с сохранением ее в текстовом файле (сетевой отчет):

Tools -> Network report -> File

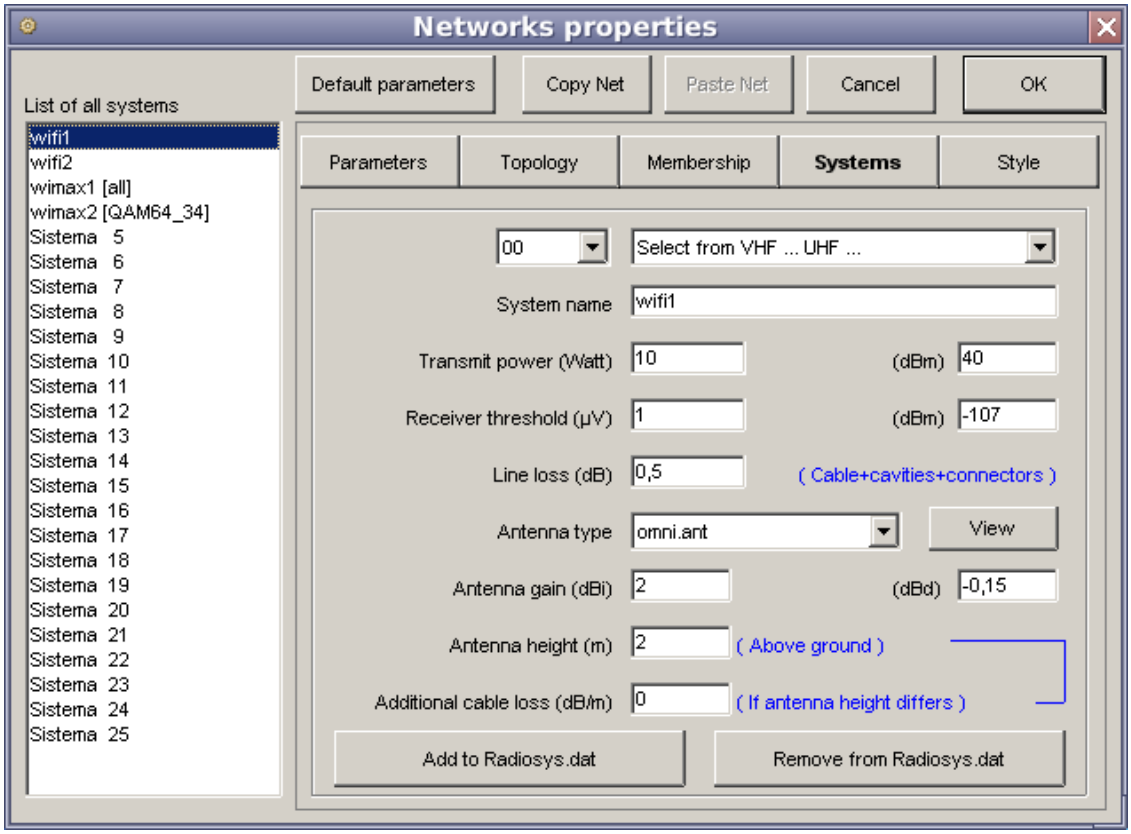

Networks properties
Default parameters
Copy Net
Paste Net
Cancel
OK
List of all systems
wifi1
wifi2
wimax1 [all]
wimax2 [QAM64_34]
Sistema 5
Sistema 6
Sistema 7
Sistema 8
Sistema 9
Sistema 10
Sistema 11
Sistema 12
Sistema 13
Sistema 14
Sistema 15
Sistema 16
Sistema 17
Sistema 18
Sistema 19
Sistema 20
Sistema 21
Sistema 22
Sistema 23
Sistema 24
Sistema 25
Parameters
Topology
Membership
Systems
Style
00
Select from VHF ... UHF ...
System name
wifi1
Transmit power (Watt)
10
(dBm)
40
Receiver threshold (µV)
1
(dBm)
-107
Line loss (dB)
0,5
(Cable+cavities+connectors)
Antenna type
omni.ant
View
Antenna gain (dBi)
2
(dBd)
-0,15
Antenna height (m)
2
(Above ground)
Additional cable loss (dB/m)
0
(If antenna height differs)
Add to Radiosys.dat
Remove from Radiosys.dat

Рис. 5:

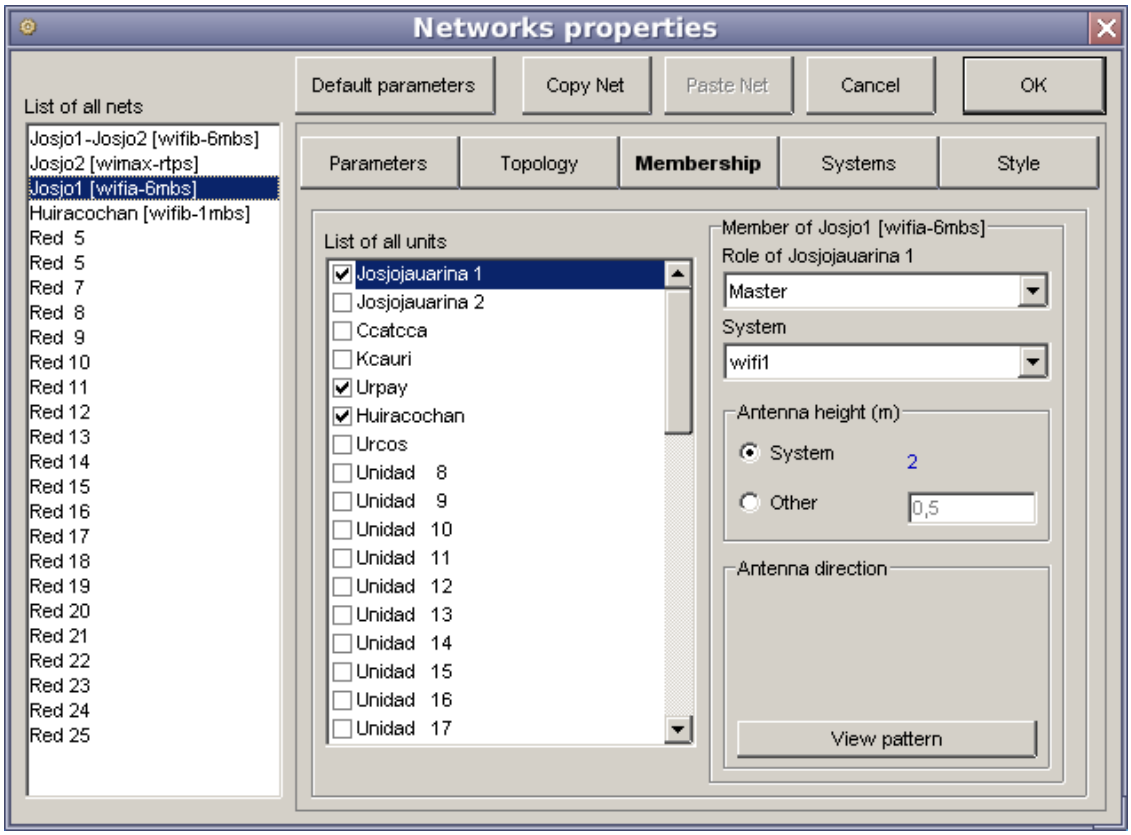

Networks properties
Default parameters
Copy Net
Paste Net
Cancel
OK
List of all nets
Josjo1-Josjo2 [wifib-6mbs]
Josjo2 [wimax-rtps]
Josjo1 [wifia-6mbs]
Huiracochan [wifib-1mbs]
Red 5
Red 5
Red 7
Red 8
Red 9
Red 10
Red 11
Red 12
Red 13
Red 14
Red 15
Red 16
Red 17
Red 18
Red 19
Red 20
Red 21
Red 22
Red 23
Red 24
Red 25
Parameters
Topology
Membership
Systems
Style
List of all units
Josjojauarina 1
Josjojauarina 2
Ccatcca
Kcauri
Urpay
Huiracochan
Urcos
Unidad 8
Unidad 9
Unidad 10
Unidad 11
Unidad 12
Unidad 13
Unidad 14
Unidad 15
Unidad 16
Unidad 17
Member of Josjo1 [wifia-6mbs]
Role of Josjojauarina 1
Master
System
wifi1
Antenna height (m)
System
2
Other
0,5
Antenna direction
View pattern

Рис. 6:

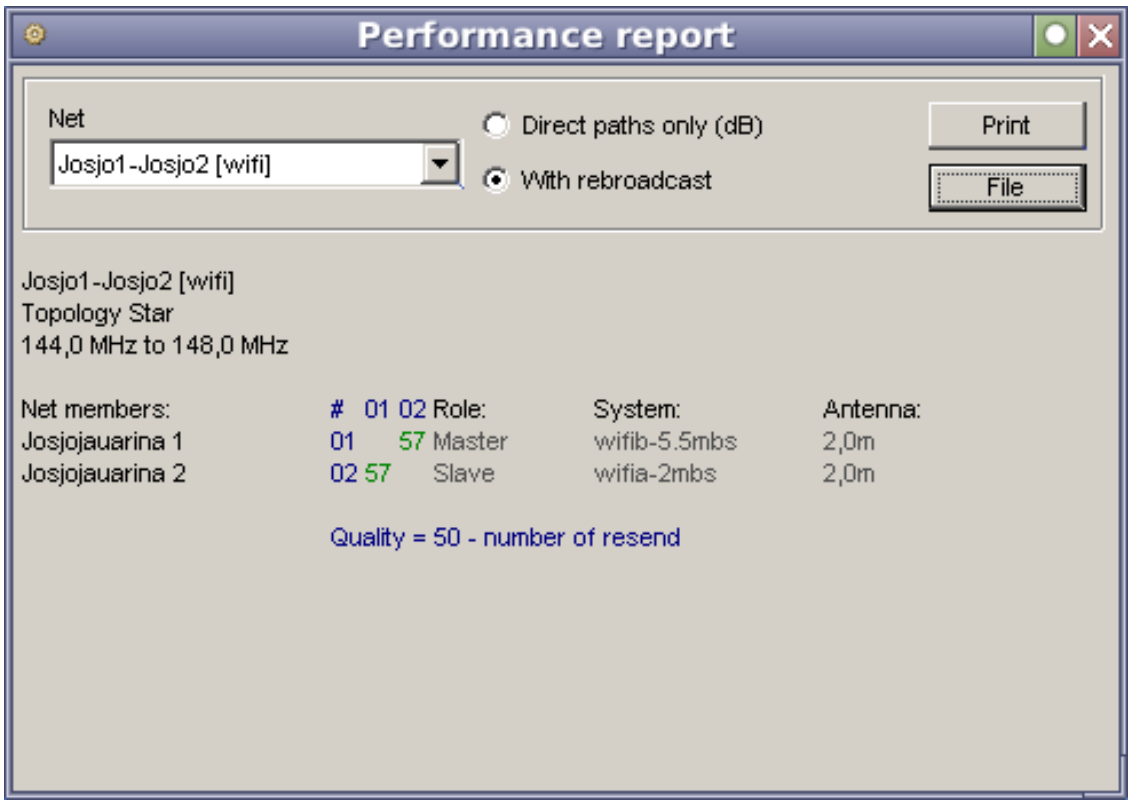


Рис. 7:

3.5.4 Преобразование сетевого отчета

Сетевой отчет(report.txt) созданный Radio Mobile должен быть преобразован в отчет NetInfo, чтобы наш модуль в NS- 3 его “понял” :

$$cdns3 - wireless - planning/ns - 3$$

$$PYTHONPATH = ../radiomobilepythonradiomobile_ns3_report.pyreport.txt > netinfo.txt$$

3.5.5 Имитация сети

Теперь пришло время для запуска моделирования с NS-3 . ns3-Wireless-planning добавляет некоторую инфраструктуру для NS-3 что делает возможным разбор NetInfo файла, созданный на предыдущем шаге . Смоделируем производительность 2 потоков , работающих одновременно (Urcos

- Huiracochan и Urcos - Ccatcca) .

Скопируем NetInfo файл в корневой каталог NS-3:

$cp/path/somewhere/netinfo.txtns-3$

$cdns-3$

Напишем код с именем scratch/wireless-planning-mysimulation.cc:

include "ns3/core-module.h"

include "ns3/simulator-module.h"

include "ns3/helper-module.h"

include "wireless-planning/net-test.h"

include "wireless-planning/create-network.h"

include "wireless-planning/network-config.h"

include "wireless-planning/print.h"

include "wireless-planning/netinfo-reader.h"

include "wireless-planning/report-2-config-data.h"

using namespace ns3;

$\mathrm{NS}_LOG_COMPONENT_DEFINE("mysimulation");$

int

main (int argc, char *argv[])

{

Time eos = Seconds (5);

/ * Обработка параметров командной строки * /
CommandLine cmd;

cmd.AddValue ("netinfo "Network Information File netInfoFile);

cmd.Parse (argc, argv);

/ * Чтение NetInfo и построение сетей * /

NetDataStruct::NetData netData = NetinfoReader::Read ((netInfoFile));
Print::Netinfo (netData);
NetworkConfig::NetworkData networkData =
Report2ConfigData::NetData2NetworkData (netData);
Print::NetworkData (networkData);

CreateNetwork createNetwork;

NodeContainer nodes = createNetwork.Create (networkData);

Print::NodeList (nodes);

/ * Начало симулятора* /

NetTest netTest;

netTest.EnablePcap("Josjojauarina 2 1);

netTest.ApplicationSetup ("Urcos "Ccatcca 1.0, 3.0, "10Mbps

1452, NULL);

netTest.ApplicationSetup ("Urcos "Huiracochan 2.0, 4.5, "10Mbps 1452, NULL);

std::vector < string > flowNames;

flowNames.push$_b ack(" Urcos - Ccatcca ")$;

$flowNames.push_b ack(" Urcos - Huiracochan ")$;

/ * Установка всех основных систем: измерение производительности ,gnuplot

* /

NetMeasure netMeasure (eos, Seconds (0.1));

netMeasure.SetupPlot ();

netMeasure.SetFlowNames (flowNames);

netMeasure.SetFlowMonitor (nodes);

netMeasure.GetFlowStats ();

Simulator::Stop (eos);

$NS_L OG_I NFO(" Startingsimulation...")$;

$Simulator :: Run();$

$Simulator :: Destroy();$

$NS_LOG_INFO("Done.");$

return 0;

}

Начнем моделирование :

$export NS_="PacketSink = level|prefix ./waf –run "scratch/wireless-planning-mysimulation$

$–netinfo=netinfo.txt"$

$gnuplot * .plt$

Gnuplot создаст изображение формата PNG для каждого участка сгенерированнонго NS- 3 . В нашем случае :

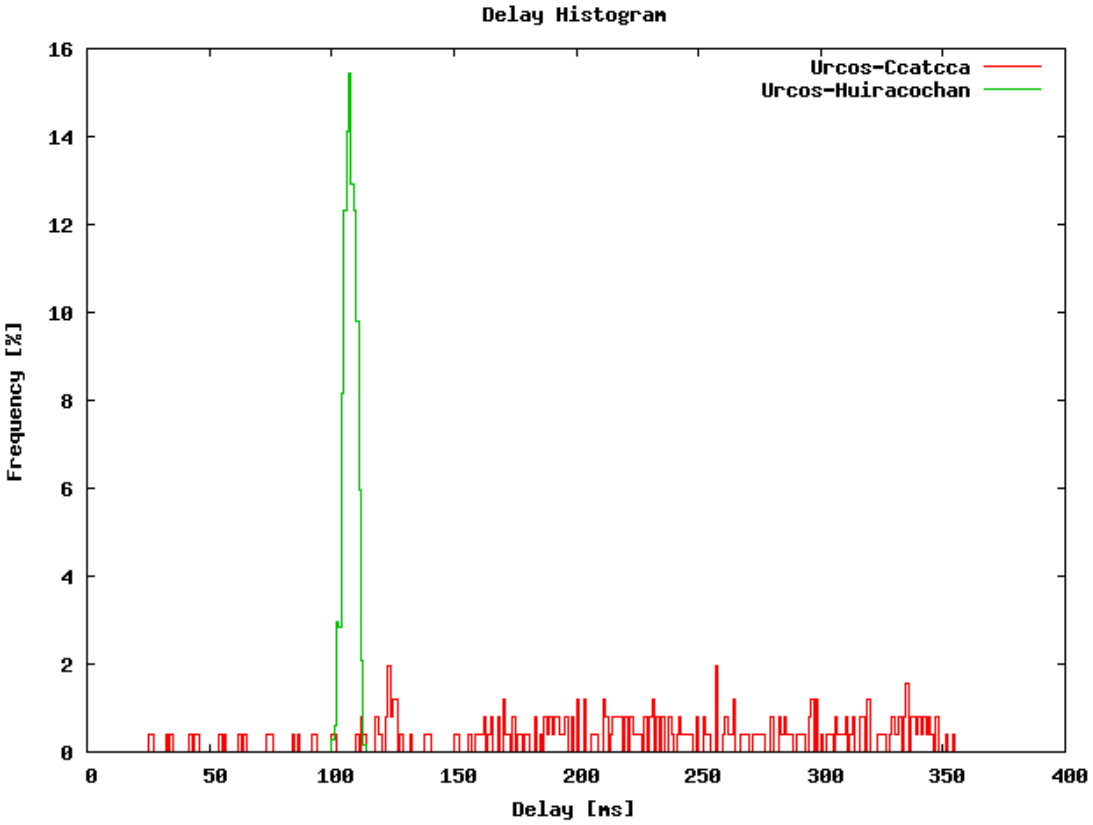

Delay Histogram
Urcos-Ccatcca
Urcos-Huiracochan
Frequency [%]
Delay [ms]
0
2
4
6
8
10
12
14
16
0
50
100
150
200
250
300
350
400

Рис. 8:

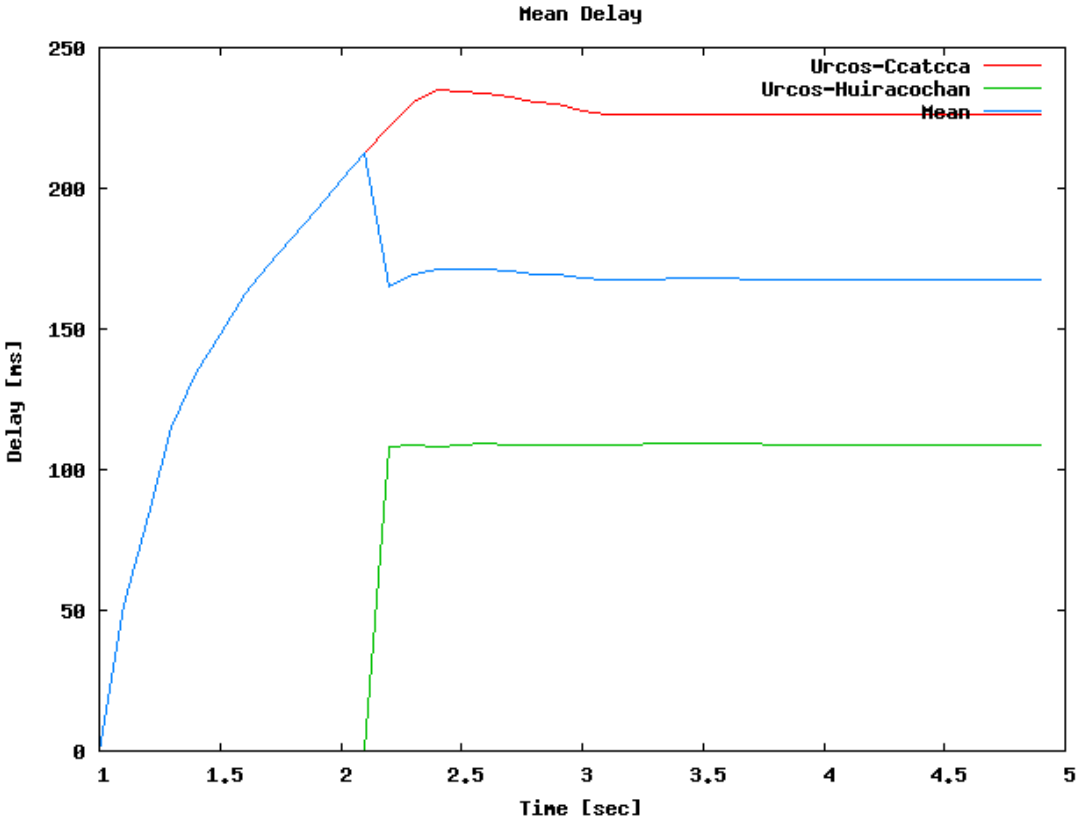

Mean Delay
Urcos-Ccatcca
Urcos-Huiracochan
Mean
Delay [ms]
Time [sec]
0
50
100
150
200
250
1
1.5
2
2.5
3
3.5
4
4.5
5

Рис. 9:

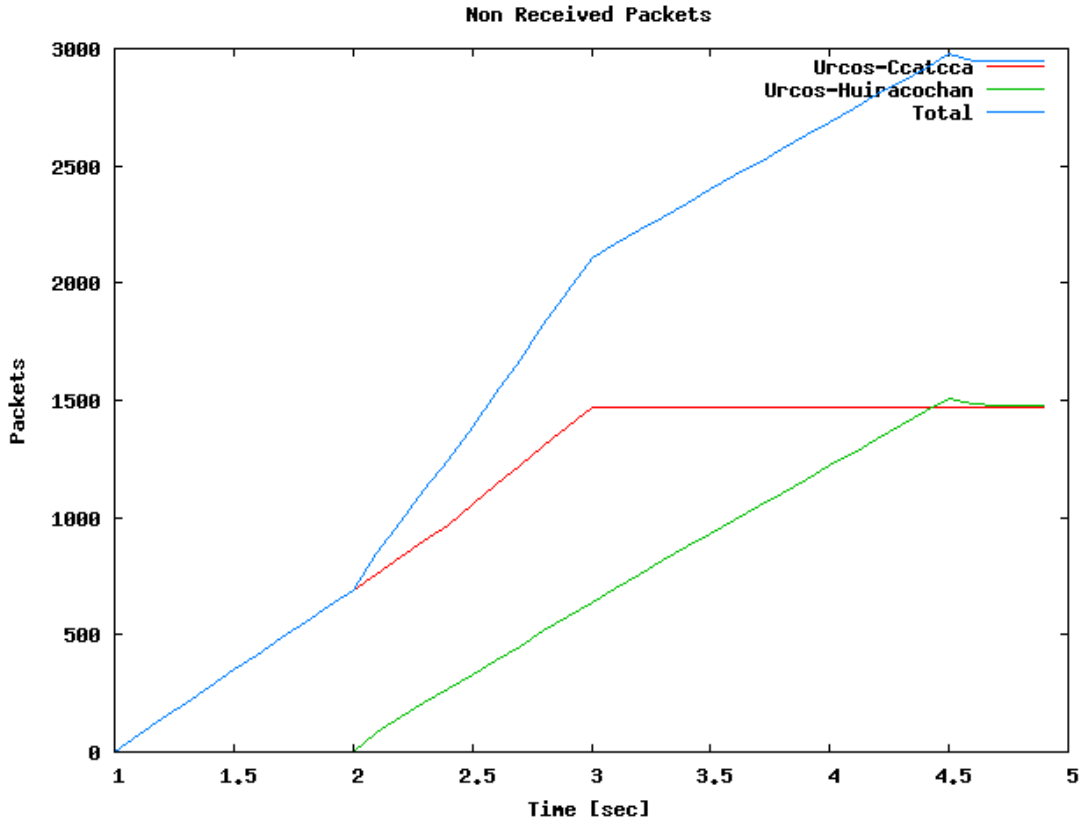

Non Received Packets
Urcos-Ccatcca
Urcos-Huiracochan
Total
Packets
Time [sec]
0
500
1000
1500
2000
2500
3000
1
1.5
2
2.5
3
3.5
4
4.5
5

Рис. 10:

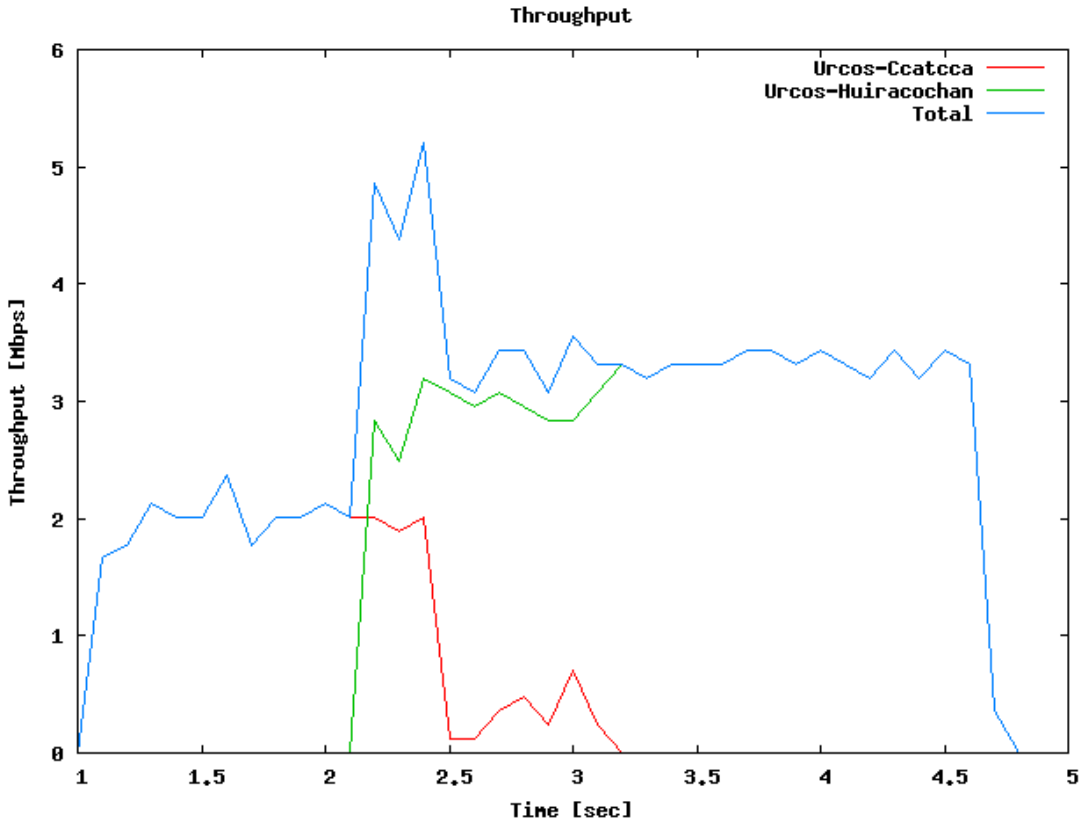

Throughput
Urcos-Ccatcca
Urcos-Huiracochan
Total
Throughput [Mbps]
Time [sec]
0
1
2
3
4
5
6
1
1.5
2
2.5
3
3.5
4
4.5
5

Рис. 11:

Использованные источники:

Latex

http://zelmanov.ptep-online.com/ctan/llang1992.pdf

http://www.intuit.ru/xml/course/LaTeX.pdf

http://ru.wikipedia.org/wiki/LaTeX

Git

http://cluster.krc.karelia.ru/doc/rukovodstvo_ GIT.pdf