УДК 621.396.2, 004.4

АВТОМАТИЗАЦИЯ ЭКСПЕРИМЕНТОВ ДЛЯ ИССЛЕДОВАНИЯ AD-HOC СЕТЕЙ

Чистяков Антон Владимирович
старший преподаватель кафедры радиоэлектронных средств
ФГБОУ ВПО «Вятский государственный университет».
E-mail: chistyakov@vyatsu.ru.

Метелев Александр Петрович
кандидат технических наук, доцент кафедры радиоэлектронных средств
ФГБОУ ВПО «Вятский государственный университет».
E-mail: ap_metelev@vyatsu.ru.
Адрес: 610000, г. Киров, ул. Московская, д.36.

Аннотация: Беспроводные самоорганизующиеся сети (ad-hoc сети) – это децентрализованные беспроводные сети, не имеющие постоянной структуры. Абонентским терминалам в таких сетях, помимо всего прочего, отводится роль маршрутизаторов, в отличие от проводных и беспроводных управляемых сетей, в которых задачу управления потоками данных выполняют специальные устройства. Минимальное конфигурирование и быстрое развертывание позволяет применять самоорганизующиеся сети в чрезвычайных ситуациях, таких как природные катастрофы и военные конфликты. При разработке моделей протоколов различных уровней для ad-hoc сетей достаточно сложной задачей является оценка их эффективности и сравнение с уже существующими разработками. Из-за высокой сложности построения комплексной математической модели таких сетей основным инструментом в этой области стало использование симуляторов дискретного времени, наиболее распространенным из которых является NS-3. Множество публикаций не содержит детального описания процесса постановки эксперимента, ведущего к публикуемым результатам, что затрудняет их воспроизведение и сравнительный анализ. В данной статье произведена декомпозиция задачи и предложены решения на базе существующих передовых технологий. Предложена архитектура программного обеспечения, которая может упростить постановку и повторение эксперимента над ad-hoc сетями.

Ключевые слова: беспроводные ad-hoc сети, MANET, сетевой симулятор ns-3, автоматизация эксперимента.

Введение

Одним из наиболее широко используемых инструментов оценки моделей ad-hoc сетей является симулятор дискретного времени NS-3. Он предлагает широкий набор готовых программных модулей, которые могут быть использованы в симуляции. Для оценки параметров разработанных моделей, а также изучения влияния на них изменения параметров сети, необходимо выполнить моделирование с различным набором входных параметров: размер сети, тип трафика, скорость передвижения узлов, типы протоколов на разных уровнях OSI и т.д. [1-5]. Каждая дополнительная переменная увеличивает число симуляций, необходимых для полного исследования модели. В случае сетей с подвижными узлами число симуляций возрастает в несколько раз, поскольку необходимо усреднять результаты внутри одного набора параметров. Все это ведет к повышению сложности постановки и сбора результатов в эксперименте. Симуляции беспроводных сетей в NS-3 имеют еще одну важную особенность: несмотря на то, что NS-3 имеет встроенную поддержку MPI, распараллеливание процесса вычислений возможно только при использовании в симуляции соединений типа “точка-точка”. Это означает, что процесс симуляции беспроводной сети может использовать только одно вычислительное ядро, что необходимо принимать во внимание при планировании эксперимента.

Большое число симуляций, необходимое для получения достоверных результатов исследования, делает эксперимент чувствительным к управлению памятью внутри кода симу-

ляции, поскольку модели для NS-3 должны быть написаны на языке C++, не обладающем механизмами автоматического освобождения памяти. Утечка памяти в одной из симуляций, запущенной на вычислительном узле совместно с набором других симуляций, окажет влияние на все, поскольку будет потреблять разделяемый ресурс памяти, что может привести к значительному замедлению процесса моделирования. При большой размерности переменных параметров такие ошибки сложно отследить и устранить, поэтому необходимо ограничивать ресурсы, потребляемые каждой отдельной симуляцией.

В данной работе предлагается использование технологии Linux containers (LXC) для создания самодостаточных образов, которые могут быть быстро и просто запущены для точного воспроизведения симуляции в любой современной среде Linux. Это позволяет использовать технологии и существующую инфраструктуру сервисов облачных вычислений при моделировании ad-hoc сетей. В настоящей статье проводится декомпозиция задачи построения облачной архитектуры для проведения экспериментов над ad-hoc сетями с использованием сетевого симулятора NS-3 и предлагаются решения на каждом из этапов. Образы симуляций, необходимые для повторения графиков, приведенных в тексте, опубликованы в хранилище Docker по адресу *https://registry.hub.docker.com/u/nuttzipper/ns-3-rts2015/*.

Обзор существующих решений

Как указано в [6], отсутствие надежных инструментов для управления симуляциями и большое число переменных параметров, влияющих на их достоверность, привело к падению доверия к симуляции как надежному методу оценки эффективности того или иного решения в области ad-hoc сетей. В 2010 году для решения этой проблемы в рамках консорциума NS-3 был запущен проект SAFE, целью которого является разработка расширения для NS-3, состоящего из трех частей:

- Система автоматизации управления симуляциями
- Модуль генерации сценариев
- Модуль, предназначенный для упрощения обучения с помощью NS-3

Система автоматизации управления симуляциями состоит из четырех частей:

- Агрегатор событий, происходящих в симуляции, в базу данных
- Анализатор событий
- Менеджер симуляций
- Конфигуратор симуляций и их визуализация на базе веб-технологий.

Согласно данным с официального сайта NS-3 консорциума, текущий статус проекта неясен. В литературе есть недавние публикации [6-8], посвященные его разработке, однако на данный момент все еще нет завершенного инструмента для управления симуляциями. В отличие от указанного проекта, в настоящей статье сделан упор на более узкую задачу симуляции беспроводных сетей, а также затронуты дополнительные аспекты повторяемости результатов и управления ресурсами.

Кроме приведенной выше разработки, существует большое разнообразие инструментов, в основном направленных на управление системами эмуляции и тестирования: Splay, Plush, ModelNet, OMF, Emulab, NEPI и др. [9–14]. Эти системы лежат за пределами темы настоящей статьи, однако необходимо принимать их во внимание, поскольку использование функций эмуляции сетевых стеков реальных систем и реальных источников трафика крайне важно для разработки и тестирования прототипов.

Декомпозиция задачи

Подготовка моделей NS-3 и скрипта симуляции. Первым шагом при оценке разработанных математических моделей является написание модулей для системы симуляции, реализующих эти модели, и скрипта симуляции, описывающего топологию сети, модели трафика, модели подвижности узлов и т.п. Эти

модули и скрип симуляции являются основной частью программного пакета, который должен быть распространен для точного повторения эксперимента.

Для запуска симуляции с различными входными параметрами необходимо передать эти данные в командной строке в процессе запуска. NS-3 использует систему сборки и запуска waf, общий ситаксис запуска следующий:

./waf --run scratch/<name>,

где *<name>* соответствует имени файла, содержащего скрипт симуляции. При выполнении команды будет выполнена компиляция, сборка и запуск получившегося исполняемого файла. Как и в случае любого исполняемого файла, входные параметры могут быть переданы в качества аргументов командной строки. Таким образом, вызвав команду *./waf --run "scratch/ <name> --param1=val1 --param2 = val2"* и добавив соответствующий обработчик (рис. 1) в скрипт симуляции, любой параметр может быть изменен на этапе запуска бинарного файла.

```
int main (int argc, char *argv[])
{
...
CommandLine cmd;
cmd.AddValue("param1", "Parameter 1", param1);
cmd.AddValue("param2", "Parameter 2", param2);
cmd.Parse (argc, argv);
...
}
```

Рис. 1. Пример кода обработки аргументов командной строки в скрипте симуляции

После этого параметры модели также могут быть скорректированы в соответствии со входными данными. Для запуска большого набора симуляций готовый бинарный файл может быть распространен среди рабочих узлов и запущен с разными аргументами.

Запуск и воспроизведение симуляции. Симулятор NS-3 является модульной системой и распространяется в виде исходных кодов, что делает её установку и настройку нетривиальной задачей для неподготовленного пользователя. Для пакетной установки различных модулей внутри консорциума NS-3 разработана утилита Bake, однако она не предназначена для распространения модулей, не включенных в официальную поставку NS-3. Таким образом, для того, чтобы воспроизвести опубликованные результаты, необходимы исходные коды NS-3 с интегрированными авторскими модулями и скриптом симуляции. Однако всегда существует вариативность, относящаяся к середе исполнения, что становится особенно актуально с учетом возможности NS-3 Direct Code Execution [15], которая позволяет исследователю вызывать пользовательские и системные приложения изнутри симуляции. В зависимости от операционной системы, версии ядра, версий библиотек и используемых сторонних приложений, поведение симуляции может варьироваться, существенно изменяя результаты.

Это приводит к необходимости соответствия следующих элементов среды выполнения для достоверного воспроизведения результатов эксперимента:

- Версия NS-3 с интегрированными авторскими модулями вместе со скриптом симуляции
- Версии системных библиотек
- Версия компилятора
- Пользовательские библиотеки
- Внешние реализации протоколов, источников трафика и другие приложения, используемые в симуляции.

Оригинальный исходный код NS-3 поддерживается с помощью распределенной системы контроля версий mercurial, также существует версия хранилища для системы git, что позволяет организовать синхронизацию рабочей версии NS-3, в которой происходит разработка авторских моделей с текущей версией NS-3, поскольку и mercurial, и git позволяют связывать рабочее хранилище с несколькими удаленными хранилищами. Для того, чтобы связать локальное хранилище с официальным репозиторием NS-3, неободимо выполнить команду *git remote add officialNS3 https://github.com/nsnam/ns-3-dev-git*. После этого все

изменения в официальной версии NS-3 могут быть получены командой *git pull officialNS3/master && git merge officialNS3.*

Сформированное таким образом хранилище может быть опубликовано его будет достаточно для точного воспроизведения отдельной симуляции при условии идентичности всех параметров окружения. Поэтому, кроме исходного кода симуляции, необходимо также включать данные параметры в распространяемый дистрибутив. Это достаточно просто осуществить с помощью технологии Linux Containers (LXC), которая представляет собой систему виртуализации на уровне ядра ОС Linux без использования гипервизора. Эта технология позволяет упаковать в самодостаточный образ подготовленное хранилище NS-3 вместе с минимальной операционной системой и системным окружением, необходимым для запуска симуляции. Затем этот образ может быть распространен и запущен в любой современной ОС Linux, поддерживающей LXC, в точности повторяя среду запуска автора исследования.

Для создания и распространения таких образов (контейнеров) предлагается использовать платформу с открытым исходным кодом Docker. Она может быть использована в любой ОС Linux с версией ядра выше 3.8 и предоставляет удобные инструменты для создания, управления и распространения контейнеров LXC. Каждый образ может быть однозначно определен файлом, описывающим процесс его создания, т.н. Dockerfile, в котором отражаются команды, выполняемые последовательно одна за одной, и конструирующие необходимое окружение внутри контейнера. Базой для каждого образа является минимальная версия какой-либо ОС, при этом множество проектов, таких как CenOS, Ubuntu, Debian, предлагают свои официальные версии минимальных ОС для Docker. По мере выполнения команд внутри Dockerfile размер образа увеличивается. Минимальный образ операционной системы составляет около 100 Мб, в то время как контейнер, полностью подготовленный к запуску симуляции NS-3, занимает около 1.2 Гб. Сравнительные размеры образов с использованием различных минимальных ОС приведены в таблице 1.

Таблица 1. Размеры образов (Мб)

Тип ОС	Рамзер базового образа	После установки зависимостей	После установки NS-3	После сборки Ns-3
Debian 7	114	358.7	517	1166
Ubuntu wheezy	194.2	359.1	517.5	1158
CentOS 7	224	561.2	719.6	1375

Небольшой размер образа важен при запуске множества параллельных симуляций на кластере, состоящем из множества рабочих машин, поскольку этот образ должен быть распространен среди узлов кластера.

Для получения графиков, опубликованных в этой работе, использован образ, сконструированный согласно Dockerfile, текст которого приведен на рис. 2.

```
FROM debian:jessie
MAINTAINER nuttzipper@gmail.com
RUN apt-get update && DEBI-
AN_FRONTEND=noninteractive apt-get install -y --no-
install-recommends gcc  g++  python  git
RUN git clone git://github.com/nuttzipper/ns-3.git
RUN cd ns-3-dev-git/; ./waf configure; ./waf build
RUN ln -s ns-3-dev-git/waf /ns3
```

Рис. 2. Содержимое файла Dockerfile, используемое для сборки образа, использованного в данной работе

В качестве минимальной ОС использован Debian Wheezy, инсталлирован минимальный объем зависимостей NS-3, а также созданы дополнительные символьные ссылки для упрощения процесса сборки и запуска симуляции с использованием системы сборки waf внутри контейнера. Описанный образ может быть сконструирован с помощью выполнения команды *docker build ./* внутри каталога, содержащего файл с текстом, показанным на рисунке 2, либо получен непосредственно из хранилища образов, предоставляемых платформой Docker, по ссылке *https://registry. hub. docker. com/ u/nuttzipper/ns-3-rts2015/*

Управление ресурсами. Следующей задачей при планировании эксперимента, состоящего из множества симуляций, является эффективное управление вычислительными ресурсами. В зависимости от размера сети, модели трафика, числа источников трафика симуляции требуют различный объем вычислительных ресурсов. При условии отсутствия вызовов внешних программ через механизм DCE основным ресурсом, который требует управления, является оперативная память, так как NS-3 как симулятор дискретного времени полностью потребляет процессорное время. Симуляции беспроводных ad-hoc сетей могут использовать только одно вычислительное ядро на экземпляр симуляции, поскольку распределение вычислений через MPI доступно только при использовании связей типа “точка-точка”, которых нет в подобных сетях. Таким образом, наиболее эффективный способ максимально загрузить доступные вычислительные мощности - распределение экземпляров симуляции среди вычислительных узлов в соответствии с числом процессорных ядер. Как только какая-то из симуляций завершается на определенном вычислительном узле, освобождая процессорное ядро, следующий экземпляр симуляции с другим набором входных параметров должен быть немедленно запущен на этом узле. При этом общий объем потребляемой всеми запущенными на машине симуляциями памяти не должен превосходить её общий доступной объем, чтобы избежать задействования файла подкачки.

Чем больше размер симулируемой сети, тем больше памяти требуется для ее успешного завершения. Аналогичное утверждение справедливо и для длительности симуляции. Временные графики потребления памяти в зависимости от числа узлов в симуляции показаны на рис. 3.

Горизонтальная шкала представляет внутреннее время симуляции, вертикальная – параметр VmRSS процесса, полученный через считывание /proc/self/status изнутри скрипта симуляции каждые две секунды внутреннего времени. Представленные графики получены для сетей с различным числом узлов, движущихся с постоянной скоростью и использующих протокол маршрутизации OLSR [16]. В каждой симуляции присутствует один источник трафика, представленный модулем NS-3 OnOffApplication, генерирующим UDP пакеты с полезной нагрузкой размером 20 бит со скоростью 5,3 кБит/сек – параметры кодека G.723.1. Источник и узел назначения выбираются случайным образом до начала симуляции. Таким образом, представленный график отражает ситуацию передачи VoIP трафика в беспроводной подвижной ad-hoc сети.

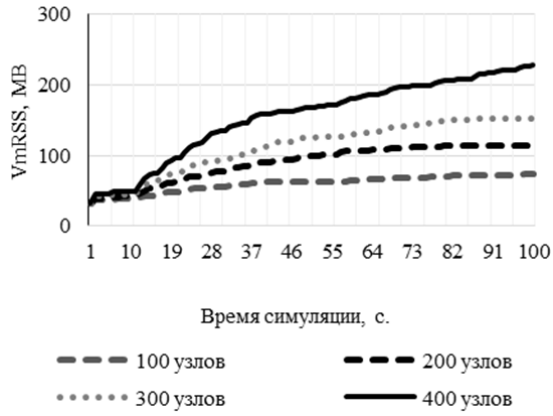

Рис. 3. Временной рост количества потребляемой памяти в зависимости от размера симулируемой сети

Как можно видеть на рис. 3, потребление памяти растет с увеличением времени симуляции, при этом, чем больше сеть, тем быстрее это происходит. Этот факт особенно важен при планировании экспериментов над моделями, которым необходимо некоторое время на инициализацию перед началом оценки их параметров, к примеру, иерархических протоколов маршрутизации. Таким протоколам необходимо некоторое время на кластеризацию перед тем, как они станут полностью работоспособными.

Для того, чтобы оценить требования к памяти, можно использовать значение VmRSS в последнюю секунду симуляции. На рис. 4 показан размер памяти, необходимый для завершения процесса симуляции без использования

файла подкачки, в зависимости от размера исследуемой сети.

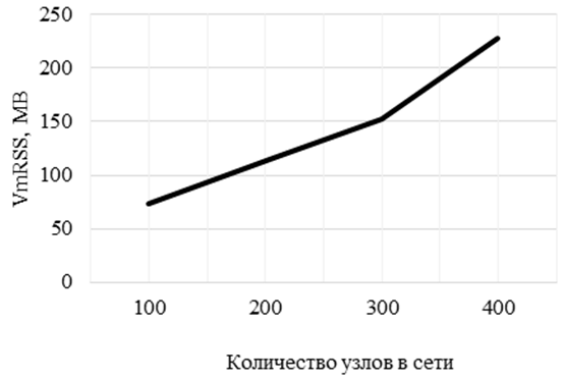

Рис.4. Размер памяти, потребленный симуляцией длительностью в 100 с

Для ограничения ресурсов, потребляемых каждой отдельной симуляцией, могут быть использованы такие технологии, как libvrt или платформа Docker. Последняя была описана в предыдущем разделе как способ достижения высокой повторяемости эксперимента. Как и libvrt, эта платформа использует cgroups для ограничения доступа LXC контейнера к вычислительным ресурсам. Каждая симуляция может быть запущена внутри контейнера с ограниченным объемом доступной памяти и процессорного времени.

Кроме ограничения ресурсов, крайне важно проводить мониторинг потребляемых ресурсов каждым отдельным контейнером с целью детектирования потенциальных ошибок в программном коде исследуемой модели. Без наличия такой информации отладка программного кода крайне трудоемка.

Масштабирование и распределение заданий. Для параллельного запуска большого числа симуляций с различными параметрами необходим формат описания параметров эксперимента и механизм распределения симуляций между рабочими узлами. Этот механизм должен обладать следующими возможностями:

- Простое масштабирование. Для любого числа вычислительных узлов необходимо иметь возможность отслеживать нагрузку, а также распределять задачи в соответствии с имеющимися свободными ресурсами.
- Интеллектуальное управление ресурсами кластера. Механизм распределения задач не должен приводить к использованию файла подкачки.
- Детектирование экземпляров симуляций, завершившихся аварийно.

В настоящее время существует множество инструментов распределения задач, от относительно простых до комплексных: Google Omega [17], Flynn, часть Google Cloud Platform - Google Kubernetes, Apache Mesos [18], модуль fleet операционной системы CoreOS, Hadoop YARN, panamax, и т.п. Большинство подобных инструментов разрабатывается и используется для управления системами типа Platform-As-A-Service и чрезмерно сложны.

Среди упомянутых решений следует отметить CoreOS – дистрибутив Linux, предназначенный для управления узлами кластера, загрузкой узлов через PXE и прозрачной процедурой удаления и введения узлов в кластер. Вместо традиционного менеджера пакетов эта ОС использует Docker. Любой пользовательский процесс, запущенный внутри CoreOS, должен быть запущен внутри контейнера. Запуском процессов управляет домен systemd. Для эффективного масштабирования и распределения задач на узлы кластера, CoreOS предлагает набор инструментов, взаимодействующих с systemd, основными из которых являются etcd и fleet. Эти инструменты доступны под лицензией Apache License и могут быть свободно модифицированы.

Сервер etcd позволяет организовать децентрализованное хранилище ключ-значение, доступное для чтения и записи всем членам кластера. Он предоставляет возможности репликации и динамического выбора мастер-узла, таким образом, даже при выключении текущего мастер узла все данные в хранилище будут актуальны. В контексте рассматриваемой задачи это хранилище может быть использовано для поддержания актуальной информации о всех узлах кластера: числе свободных узлов, доступном объеме оперативной памяти, а так-

же для текущего статуса каждой конкретной симуляции: на каком физическом узле она запущена, внутреннее время симуляции, и т.д. Таким образом, вся информация, необходимая для принятия решения при отправке нового экземпляра симуляции на выполнение, может быть помещена в etcd. Этот сервер предоставляет http long-polling интерфейс для подписчиков на определенные события, возникающие внутри кластера, к примеру, успешное завершение очередного экземпляра симуляции. События могут быть помещены в хранилище либо с помощью утилиты etcdctl, либо посылкой http запроса на определенный URL.

Для управления распределения задач, CoreOS предлагает систему fleet, состоящую из сервера fleetd и утилиты командного интерфейса fleetctl. Для описания задач используются файлы того же формата, что используются system, называемыми unit файлами. Каждый unit файл представляет собой задачу, предназначенную для отправки на рабочий узел. Таким образом, каждый unit file соответствует экземпляру симуляции с определенными параметрами.

Наиболее важные параметры: ExecStartPre, ExecStart и ExecStartPost

Поле ExecStart содержит команду, которая запускает задачу, ExecStartPre выполняется непосредственно перед запуском ExecStart и может быть использована для сигнализирования etcd о том, что на определенном узле начинается запуск новой симуляции и, следовательно, следует изменить число доступных процессорных ядер для этого узла. ExecStopPost запускается по завершении процесса, успешном или аварийном. На этом этапе должна происходить запись в etcd информации об освобождении ресурсов.

Сбор событий симуляции. При наличии большого количества высокоскоростных источников трафика необходимо выгружать генерируемые для анализа данные в стороннюю базу данных, поскольку хранение их в памяти для последующей обработки, к примеру, расчета статистики, может вызвать избыточное потребление памяти.

Кроме метаданных, связанных с обработкой трафика, потенциально полезными могут быть также метаданные, генерируемые различными модулями. NS-3 предлагает мощный механизм сбора таких данных. Каждый модуль имеет набор источников событий, к которым проектировщик симуляции может подключиться и собирать возникающие внутри него события. Такая возможность помогает глубже понять происходящие внутри симуляции процессы и упрощает отладку.

Большое разнообразие генерируемых данных ведет к значительному потреблению памяти, что, в свою очередь, мешает на этапе запуска нескольких симуляций на одном вычислительном узле. Поэтому важно внутри симуляции предусмотреть механизм адаптации системы сбора данных NS-3 для подключения к внешней базе данных. В настоящий момент существует большое количество методов и библиотек, реализующих протоколы обмена сообщениями, предусмотренных для цели подобно описанной выше: Apache Kafka, zeroMQ, iromMQ, и т.п.

Визуализация. На настоящий момент существует два основных метода визуализации симуляции NS-3: PyViz и NetAnim. PyViz является визуализатором реального времени, в основном предназначенным для целей отладки. NetAnim использует xml файлы, генерируемые с помощью системы сбора сообщений NS-3 в процессе выполнения симуляции. Проект SAFE[6] анонсировал создание визуализатора на базе javascript библиотек D3.js и Rickshaw. Однако, текущий статус проекта неясен.

Задача визуализации симуляции может быть разделена на несколько подзадач:

- Визуализация передвижения узлов сети
- Визуализация трафика
- Отображение различных пользовательских данных в процессе симуляции.

Движение узлов в NS-3 реализуется с помощью модуля MobilityModel, имеющий ис-

точник данных CourseChange. Этот источник является основным поставщиком данных для визуализации передвижения узлов. Необходимые для визуализация трафика данные можно получить из источников Rx и Drop модуля ns3::Ipv4L3Protocol. Кроме этих событий, визуализация пользовательских данных, генерируемых пользовательскими скриптами и модулями, также является важной задачей. К примеру, при исследовании иерархического протокола маршрутизации полезной информацией является состояние кластеризации.

Технологии HTML5 позволяют выстроить канал связи между симуляцией и веб-браузером через WebSockets, а javascript фреймворки, такие как D3.js, Processing.js, достаточно производительны, чтобы визуализировать большие сети, состоящие из сотен узлов. Он-лайн визуализация накладывает существенные ограничения на размер передаваемых данных и скорость кодирования – декодирования сообщений. Существует большое количество форматов сериализации для различных областей применения. В существующих публикациях, сравнивающих между собой разные форматы [19-20], показывается превосходство формата protobuf над наиболее широко распространенным JSON. Другой формат с высокой степенью эффективности сериализации данных BSON, используемый в NoSQL базе данных MongoDB, не представлен в литературе, однако авторы MongoDB заявляют о превосходстве этого формата над соперниками в задаче хранения и передачи данных.

Предлагаемая архитектура

На рис. 5 изображена полная схема предлагаемой архитектуры программного обеспечения. Разработчик помещает весь необходимый исходный код (NS-3 вместе в минимальной ОС и необходимыми бибиотеками) в контейнер Docker, который используется инфраструктурой CoreOS для запуска и управления симуляциями. Запуск симуляций осуществляется сервером fleetd, управляемым через командный интерфейс fleetctl. Для интеллектуального распределения задачи по рабочим узлам fleetd использует хранилище etcd, в котором содержится вся информация о текущем состоянии кластера. Технические подробности, такие как потребление контейнерами памяти и процессорного времени, можно видеть через сервер приложений, собирающий данные с помощью cAdvisor. Этот же сервер приложений выступает в качестве прокси узла, сохраняющего данные симуляции в MongoDB и отправляющего данные визуализации в браузер оператора через WebSockets. На рисунке представлен только один рабочий узел, однако кластер может содержать неограниченное количество идентичных рабочих узлов – fleetd прозрачно управляет включением узлов в кластер и их выходом.

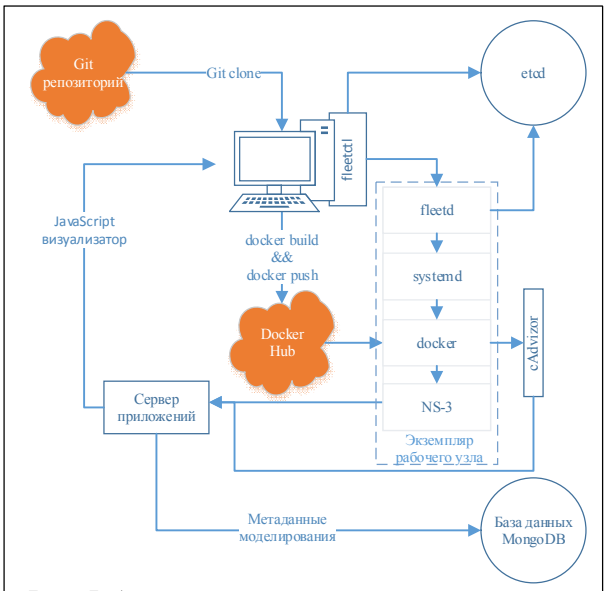

Рис. 5. Архитектура предлагаемй системы автоматизации эксперимента для NS-3

Заключение

В настоящей статье предложена архитектура программного обеспечения для автоматизации экспериментов с использованием NS-3 с учетом особенностей симуляции мобильных беспроводных ad-hoc сетей. В её основе лежит использование контейнеров Docker для создания образов, которые можно легко распространить и с помощью них воспроизвести результаты исследования. Графики, опубликованные в настоящей статье, получены с использованием образа, доступного по ссылке

https://registry.hub.docker.com/u/ nuttzipper/ns-3-rts2015/.

Настоящая работа может быть использована при создании законченной интегрированной системы автоматизации планирования и управления экспериментом с использованием NS-3.

Литература

1. Романов С.В. Анализ иерархического протокола маршрутизации manet-сетей / С.В. Романов, Д.Е. Прозоров, И.С. Трубин // Перспективы науки. – 2012. – № 4 (31). – С. 86-89.
2. Жолобов А.Н. Симуляторы беспроводных manet-сетей / А.Н. Жолобов, Д.Е. Прозоров, С.В. Романов // Инфокоммуникационные технологии. – 2012. – Т. 10. – № 3. – С. 28-33.
3. Прозоров Д.Е. Протоколы геомаршрутизации самоорганизующихся мобильных сетей / Д.Е. Прозоров, А.П. Метелев, А.В. Чистяков, С.В. Романов // T-Comm: Телекоммуникации и транспорт. – 2012. – Т. 6. – № 5. – С. 16-19.
4. Романов С.В. Методы гибридной и иерархической маршрутизации manet / С.В. Романов, Д.Е. Прозоров // Перспективы науки. – 2013. – № 6 (45). – С. 69-73.
5. Жолобов А.Н. Принципы формирования кластеров в ad-hoc сетях / А.Н. Жолобов, В.А. Лесников, С.В. Романов // Научное обозрение. – 2012. – № 4. – С. 264-273.
6. L. F. Perrone, C. S. Main, and B. C. Ward. SAFE: Simulation Automation Framework for Experiments. Simulation Conference (WSC), Proceedings of the 2012 Winter, 2012, no. 2002, pp. 1 – 12, doi: 10.1109/WSC.2012.6465286.
7. L. F. Perrone, T. R. Henderson, M. J. Watrous, and V. Daly Felizardo. The design of an output data collection framework for NS-3. Proceedings of the Winter Simulation Conference, 2013, pp. 2984 – 2995, doi: 10.1109/WSC.2013.6721666.
8. C. S. Main and L. F. Perrone. User interfaces for the simulation automation framework for experiments. WSC '12 Proceedings of the Winter Simulation Conference, 2012, №396.

A. Quereilhac, M. Lacage, C. Freire, T. Turletti, W. Dabbous, and S. Antipolis. NEPI : An Integration Framework for Network Experimentation. Proc. Software, Telecommunications and Computer Networks (SoftCOM), 2011, pp. 1 – 5.

9. L. Leonini and P. Felber. P2P experimentations with S PLAY : from idea to deployment results in 30 min. Proc. Peer-to-Peer Computing, 2008, pp. 189 – 190, doi: 10.1109/P2P.2008.14.
10. J. Albrecht, C. Tuttle, A. Snoeren, and A. Vahdat. Distributed application management using Plush. Proc. High Performance Distributed Computing 14th IEEE International Symposium, 2005, pp. 281 – 282, doi: 10.1109/HPDC.2005.1520975.
11. K. V. Vishwanath, D. Gupta, A. Vahdat, and K. Yocum. ModelNet: Towards a datacenter emulation environment. Proc. Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on, 2009, pp. 81 – 82, doi: 10.1109/P2P.2009.5284497.
12. C. Siaterlis, A. P. Garcia, and B. Genge. On the Use of Emulab Testbeds for Scientifically Rigorous Experiments. Proc. Commun. Surv. Tutorials, IEEE, vol. 15, no. 2, pp. 929–942, 2012, doi: 10.1109/SURV.2012.0601112.00185.
13. G. Di Stasi, R. Bifulco, F. D'Elia, and S. Avallone. Experimenting with P2P traffic optimization for wireless mesh networks in a federated OMF-PlanetLab environment. Proc. Wireless Communications and Networking Conference (WCNC), 2011, pp. 719–724, doi: 10.1109/WCNC.2011.5779251.
14. H. Tazaki, F. Urbani, E. Mancini, and M. Lacage. Direct Code Execution : Revisiting Library OS Architecture for Reproducible Network Experiments Categories and Subject Descriptors. Proc. of the ninth ACM conference on Emerging networking experiments and technologies, 2013, pp. 217–228.
15. P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. Proc. Multi Topic Conference, IEEE INMIC 2001. Technology for the 21st Century, 2001, pp. 62–68, doi: 10.1109/INMIC.2001.995315.
16. M. Schwarzkopf, A. Konwinski, M. Abd-el-malek, and J. Wilkes. Omega : flexible , scalable schedulers for large compute clusters. Proc. IGOPS European Conference on Computer Systems (EuroSys), ACM, 2013, pp. 351–364, doi: 10.1145/2465351.2465386.
17. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos : A Platform for Fine-Grained Resource Sharing in the Data Center. Proc. 8th USENIX conference on Networked systems design and implementation, 2011, pp. 295–308.
18. K. Maeda. Performance evaluation of object serialization libraries in XML, JSON and binary formats. Proc. Digital Information and Communication Technology and it's Applications (DICTAP), 2012, pp. 177–18, doi: 10.1109/DICTAP.2012.6215346.

A. Sumaray and S. K. Makki. A comparison of data serialization formats for optimal efficiency on a mobile platform. Proc. The 6th International Conference on Ubiquitous Information Management and Communication, 2012, p. 48, doi: 10.1145/2184751.2184810.

Исследование проведено при финансовой поддержке РФФИ в рамках проекта № 14-07-00086.

Поступила 04 июня 2015 г.

English

Experiments automation for ad-hoc networks research

Anton Vladimirovich Chistyakov – Senior Teacher Radio-electronics Department Federal State Budgetary Educational Institution of Higher Education «Vyatka State University».

E-mail: chistyakov@vyatsu.ru.

Alexander Petrovich Metelev – Candidate of Technical Sciences Associate Professor Radio-electronics Department Federal State Budgetary Educational Institution of Higher Education «Vyatka State University».

E-mail: ap_metelev@vyatsu.ru.

Address: 610000, Kirov, Moskovskaya St., 36.

Abstract: Wireless independent networks (ad-hoc networks) are decentralized wireless networks which do not have existing infrastructure. What is more terminal stations serve as routers in such networks unlike wired and wireless controlled networks in which data flow control is performed by special devices. Minimum configuring and rapid deployment enables to utilize independent networks in emergency situations such as natural disasters and military conflicts. When developing protocol models of various levels for ad-hoc networks it is rather complex problem to evaluate their efficiency and compare them with already existing projects. Discrete time simulators of which the most widespread is NS-3 became main tool for mathematical modeling of such networks due to high complexity of this modeling. Many published works do not contain detailed process description of setting experiment that led to published results that complicates their duplication and comparative analysis. Task decomposition is done and solutions based on current advanced technologies are proposed in this article. Software architecture is proposed which can simplify setting and duplication experiment with ad-hoc networks.

Key words: wireless ad-hoc networks, MANET, ns-3 network simulator, experiment automation.

References

1. Romanov S.V. Analysis of hierarchical routing protocol of mobile ad-hoc networks (MANET). - S. V. Romanov, D.E. Prozorov, I.S. Trubin. - Perspektivy nauki. - 2012. - No. 4 (31). – P. 86-89.
2. Zholobov A.N. Wireless MANET simulators. - A.N. Zholobov, D.E. Prozorov, S. V. Romanov. - Infokommunikatsionnye tekhnologii. - 2012. - T. 10. - No. 3. – P. 28-33.
3. Prozorov D.E. Georouting protocols of independent mobile networks. - D.E. Prozorov, A.P. Metelev, A.V. Chistyakov, S. V. Romanov. - T-Comm: Telekommunikatsii i transport. - 2012. - T. 6. - No. 5. – P. 16-19.
4. Romanov S. V. MANET hybrid and hierarchical routing methods. - S. V. Romanov, D.E. Prozorov. - Perspektivy nauki. - 2013. - No. 6 (45). – P. 69-73.
5. Zholobov A.N. Clusters generation principles in ad-hoc networks. - A.N. Zholobov, V.A. Lesnikov, S. V. Romanov. - Nauchnoye obozreniye. - 2012. - No. 4. – P. 264-273.
6. L.F. Perrone, C. S. Main, and B. C. Ward. SAFE: Simulation Automation Framework for Experiments. Simulation Conference (WSC), Proceedings of the 2012 Winter, 2012, no. 2002, pp. 1 - 12, doi: 10.1109/WSC.2012.6465286.
7. L.F. Perrone, T.R. Henderson, M.J. Watrous, and V. Daly Felizardo. The design of an output data collection framework for NS-3. Proceedings of the Winter Simulation Conference, 2013, pp. 2984 - 2995, doi: 10.1109/WSC.2013.6721666.
8. C.S. Main and L. F. Perrone. User interfaces for the simulation automation framework for experiments. WSC ’12 Proceedings of the Winter Simulation Conference, 2012, No. 396.

A. Quereilhac, M. Lacage, C. Freire, T. Turletti, W. Dabbous, and S. Antipolis. NEPI : An Integration Framework for Network Experimentation. Proc. Software, Telecommunications and Computer Networks (SoftCOM), 2011, pp. 1 - 5.

9. L. Leonini and P. Felber. P2P experimentations with S PLAY : from idea to deployment results in 30 min. Proc. Peer-to-Peer Computing, 2008, pp. 189 - 190, doi: 10.1109/P2P.2008.14.
10. J. Albrecht, C. Tuttle, A. Snoeren, and A. Vahdat. Distributed application management using Plush. Proc. High Performance Distributed Computing 14th IEEE International Symposium, 2005, pp. 281 - 282, doi: 10.1109/HPDC.2005.1520975.
11. K.V. Vishwanath, D. Gupta, A. Vahdat, and K. Yocum. ModelNet: Towards a datacenter emulation environment. Proc. Peer-to-Peer Computing, 2009. P2P ’09. IEEE Ninth International Conference on, 2009, pp. 81 - 82, doi: 10.1109/P2P.2009.5284497.

12. C. Siaterlis, A. P. Garcia, and B. Genge. On the Use of Emulab Testbeds for Scientifically Rigorous Experiments. Proc. Commun. Surv. Tutorials, IEEE, vol. 15, no. 2, pp. 929-942, 2012, doi: 10.1109/SURV.2012.0601112.00185.

13. G. Di Stasi, R. Bifulco, F. D'Elia, and S. Avallone. Experimenting with P2P traffic optimization for wireless mesh networks in a federated OMF-PlanetLab environment. Proc. Wireless Communications and Networking Conference (WCNC), 2011, pp. 719-724, doi: 10.1109/WCNC.2011.5779251.

14. H. Tazaki, F. Urbani, E. Mancini, and M. Lacage. Direct Code Execution : Revisiting Library OS Architecture for Reproducible Network Experiments Categories and Subject Descriptors. Proc. of the ninth ACM conference on Emerging networking experiments and technologies, 2013, pp. 217-228.

15. P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. Proc. Multi Topic Conference, IEEE INMIC 2001. Technology for the 21st Century, 2001, pp. 62-68, doi: 10.1109/INMIC.2001.995315.

16. M. Schwarzkopf, A. Konwinski, M. Abd-el-malek, and J. Wilkes. Omega : flexible, scalable schedulers for large compute clusters. Proc. IGOPS European Conference on Computer Systems (EuroSys), ACM, 2013, pp. 351-364, doi: 10.1145/2465351.2465386.

17. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos : A Platform for Fine-Grained Resource Sharing in the Data Center. Proc. 8th USENIX conference on Networked systems design and implementation, 2011, pp. 295-308.

18. K. Maeda. Performance evaluation of object serialization libraries in XML, JSON and binary formats. Proc. Digital Information and Communication Technology and it's Applications (DICTAP), 2012, pp. 177 - 18, doi: 10.1109/DICTAP.2012.6215346.

A. Sumaray and S. K. Makki. A comparison of data serialization formats for optimal efficiency on a mobile platform. Proc. The 6th International Conference on Ubiquitous Information Management and Communication, 2012, p. 48, doi: 10.1145/2184751.2184810.