МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В. ЛОМОНОСОВА,
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

На правах рукописи
УДК 519.876.5

АНТОНЕНКО ВИТАЛИЙ АЛЕКСАНДРОВИЧ

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МОДЕЛИ ФУНКЦИОНИРОВАНИЯ ГЛОБАЛЬНОЙ СЕТИ ДЛЯ АНАЛИЗА ДИНАМИКИ РАСПРОСТРАНЕНИЯ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Специальность 05.13.11 —
«Математическое обеспечение вычислительных машин,
комплексов и компьютерных сетей.»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
член-корр. РАН, профессор
Смелянский Р.Л.

Москва – 2014

Содержание

Введение

Под термином Глобальная Компьютерная Сеть (ГКС) будем понимать компьютерную сеть, состоящую не менее чем из 10^5 узлов. Узел сети характеризуется определенным набором параметров; подробное описание параметров представлено в разделе 2.2.1. Значения параметров узла сети определяют его состояние; совокупное состояние всех узлов определяет состояние сети. Узлы соединяются каналами и тем самым образуют структуру сети, называемую топологией.

В современных ГКС актуально уметь оперативно прогнозировать динамику изменения состояния сети, например:

- прогнозировать динамику распространения вредоносного программного обеспечения (ВПО) и оценивать наносимый ущерб ГКС. Ущербом будем считать изменение параметров качества сервиса (задержек и процента потерь легитимного трафика) [1, 2];
- прогнозировать и оценивать задержку при доставке контента от сервера хранения до получателя [3, 4];
- прогнозировать скорость сходимости протоколов маршрутизации и оценивать накладные расходы (например, количество служебного трафика в ходе функционирования исследуемого протокола) [5].

Для моделирования больших сетей, сопоставимых с ГКС, обычно используют наиболее абстрактные подходы к моделированию процесса функционирования сети [6, 7]. Для таких подходов характерна невысокая точность моделирования процессов сетевого обмена между узлами сети, что существенно, например, при прогнозировании динамики распространения вредоносного программного обеспечения (ВПО).

Такой прогноз требует умения анализировать функционирование ГКС. Под функционированием ГКС понимается процесс обработки и передачи сетевого трафика между узлами сети. Основным методом исследования функционирования ГКС является

имитационное моделирование. Процесс построения имитационной модели состоит из двух этапов:

1. построения математической модели объекта;
2. реализации математической модели на вычислительной системе (ВС).

Исходя из большой размерности и сложности структуры ГКС, эксперименты без использования моделирования затруднены по финансовым причинам, а также из-за невозможности физического воссоздания сети столь большого размера. Имитационная модель ГКС — это с одной стороны комбинация математической модели и ее реализация на ВС, с другой — результат компромисса между:

- уровнем детальности описания;
- сложностью описания функционирования;
- точностью предсказания поведения;
- сложностью идентификации и калибровки построенной модели.

Необходимая детализация имитационной модели ГКС зависит от целей моделирования и определяется исследователем при подготовке эксперимента моделирования. Подробность и точность имитационной модели зависит от выбора уровня абстракции объекта моделирования, а также от выбора математического аппарата, в терминах которого строится модель. В настоящее время исследователи для моделирования ГКС чаще всего используют вероятностный математический аппарат [8–10].

Использование вероятностного математического аппарата предполагает статистическое усреднение многих параметров функционирования ГКС, либо их представление в форме функции распределения соответствующего вида. Статистическое усреднение используется для:

- упрощения модели;
- сокращения размерности модели;
- уменьшения вычислительной сложности реализации модели.

Использование функций распределения позволяет лишь приблизительно описать динамику многих параметров. Как известно из литературы [11], многие параметры функционирования сети не могут быть описаны известными математическими распределениями. В качестве примера можно привести процесс изменения длины очередей в маршрутизаторах. Как результат, использование вероятностного математического аппарата приводит к уменьшению точности моделирования реальной сети, также появляется необходимость калибровки модели сети перед проведением эксперимента моделирования. Под точностью моделирования сети понимается соответствие моделируемых процессов обработки и передачи сетевого трафика между узлами в заданной топологии с процессами, происходящими в реальной сети.

В тоже время хорошо известно из практики, что наиболее точным математическим описанием функционирования сети является композиция автоматов (в математическом смысле этого слова), представляющих функционирование стека протоколов [12]. Однако его использование сопряжено с рядом трудностей, основной из которых является сложность и большая размерность итоговой модели сети.

Целью данной работы является разработка и реализация системы моделирования процесса функционирования ГКС с возможностью анализа динамики распространения ВПО.

Для достижения поставленной цели необходимо было решить следующие задачи:

1. составить обзор математических методов описания/построения моделей функционирования сети и средств реализации моделей сети. Оценить их с точки зрения следующих критериев:
   (a) точность моделирования функционирования сети (точность);
   (b) требовательность к вычислительным ресурсам для вычисления результатов моделирования (ресурсоемкость);
   (c) зависимость количества узлов в моделируемой сети от количества вычислительных ресурсов, используемых в процессе моделирования (масштабируемость);
2. построить формальную модель ГКС, которая позволяет моделировать функционирование ГКС с возможностью анализа динамики распространения ВПО;

3. исследовать применимость эпидемических моделей, известных из медицины и биологии, для описания процесса распространения ВПО;

4. разработать систему имитационного моделирования ГКС с возможностью точного моделирования процесса распространения ВПО, то есть моделирования всех стадий (выбор жертвы, сканирование, заражение и т.д.) жизненного цикла ВПО;

5. исследовать динамику распространения тестового набора ВПО в разработанной системе имитационного моделирования ГКС.

Научная новизна работы заключается в разработке подхода к построению имитационных моделей на основе техники легковесной виртуализации [13], которая позволяет:

- масштабировать модель вплоть до размеров ГКС;
- сократить затраты на калибровку и идентификацию модели;
- избежать необходимости доказательства корректности конкретной модели сети, то есть доказательства факта воспроизведения процессов обработки и передачи сетевого трафика в заданной пользователем топологии сети.

Практическая ценность в создании системы имитационного моделирования на основе техники легковесной виртуализации заключается в упорядочении и упрощении процесса построения имитационной модели ГКС. Подобная система предназначена для:

- исследователей в области компьютерных сетей при анализе различных сетевых обменов и их влияния на различные сетевые характеристики;
- разработчиков сетевых приложений и протоколов для определения корректности работы приложения или, например, для исследования сходимости нового протокола маршрутизации;
- сетевых архитекторов на различных стадиях проектирования и реализации сети.

Построение имитационной модели при помощи техники легковесной виртуализации называется в литературе Hi-Fi моделирование [14]. При использовании техники легковесной виртуализации модель сети строится из «виртуальных контейнеров», главной целью которых является воспроизведение функционирования основных компонентов сети: сетевых интерфейсов, сетевого стека, каналов связи. Hi-Fi

моделирование позволяет точно воспроизводить процессы обработки и передачи сетевого трафика, так как функционирования сети задается настройками виртуального сетевого стека операционной системы машины, на которой осуществляется процесс моделирования, что фактически означает использование реального сетевого стека.

Впервые Hi-Fi моделирование было использовано в программно-конфигурируемых сетях (ПКС) [15] для описания процесса взаимодействия коммутаторов и ПКС контролера. ПКС контролер — вычислительная машина, определяющая правила для реализации политик маршрутизации в контролируемом сегменте сети. Детально ознакомиться с ПКС сетями можно в следующих работах [16], [17].

Основные положения, выносимые на защиту:

1. Построена математическая модель, позволяющая моделировать функционирование ГКС, при этом результат моделирования близок к функционированию реальной сети. В терминах этой модели описана задача прогнозирования динамики распространения ВПО и показано, что она разрешима.

2. На основе техники легковесной виртуализации предложен новый подход к моделированию функционирования ГКС, позволяющий строить модели сетей необходимого размера. Отличительной чертой подхода является высокая точность моделирования процесса функционирования ГКС по сравнению с существующими подходами.

3. Разработана и реализована уникальная распределенная система имитационного моделирования (СИМ) сети с использованием «виртуальных контейнеров», названная Network Prototype Simulator (NPS). Результаты моделирования в СИМ NPS продемонстрированы применительно к задачи исследования динамики распространения ВПО.

Апробация работы. Основные результаты работы докладывались на:

- 17h GENI ENGINEERING CONFERENCE (GEC17);
- SIGCOMM 2013;
- YET ANOTHER CONFERENCE 2013;
- SOFTWARE ENGINEERING CONFERENCE IN RUSSIA 2013.

Публикации. Основные результаты по теме диссертации изложены в печатных изданиях [18, 19], два из которых изданы в журналах, рекомендованных ВАК [18, 19].

Объем и структура работы. Диссертация состоит из введения, четырех глав и заключения. Полный объем диссертации составляет 108 страниц с 36 рисунками и 5 таблицами. Список литературы содержит 115 наименований.

Глава 1

Обзор предметной области. Обзор методов и средств моделирования компьютерной сети

Целью данного обзора является рассмотрение средств реализации и основных математических аппаратов, используемых для построения моделей сетей с точки зрения точности моделирования функционирования сети, требовательности к вычислительным ресурсам, зависимости количества узлов в моделируемой сети от количества вычислительных ресурсов, используемых в процессе моделирования.

1.1 Понятие глобальной компьютерной сети

Для описания упомянутых требований (точности, ресурсоемкости, масштабируемости) необходимо определить базовые понятия для такого объекта, как ГКС. ГКС — это сеть, состоящая не менее чем из 10^5 узлов. Узлом сети будем называть точку в сети, имеющую уникальный сетевой адрес. Примером узла может быть либо хост, либо сетевое устройство, которое объединяет хосты между собой, например, коммутатор или машрутизатор. Хост (компьютер с сетевым интерфейсом) — это потребитель сетевого сервиса ГКС, на котором задан уникальный сетевой адрес. Предполагается, что ГКС обладает следующими свойствами:

- размер сети не менее 10^5 хостов. Хостом является только вершина степени один в графе сети (см. рисунок 1.1);
- сеть разделена на домены. Домен — это множество связанных между собой хостов. Будем считать, что множества хостов различных доменов не пересекаются.

Мобильные хосты не рассматриваются, то есть при переходе хоста из одного домена в другой первоначальные свойства домена не изменяются;

- различные домены связаны между собой каналами типа «точка-точка» [20]. Под каналом будем понимать пару сетевых интерфейсов, которые связывают хосты с сетевыми устройствами или сетевые устройства между собой.

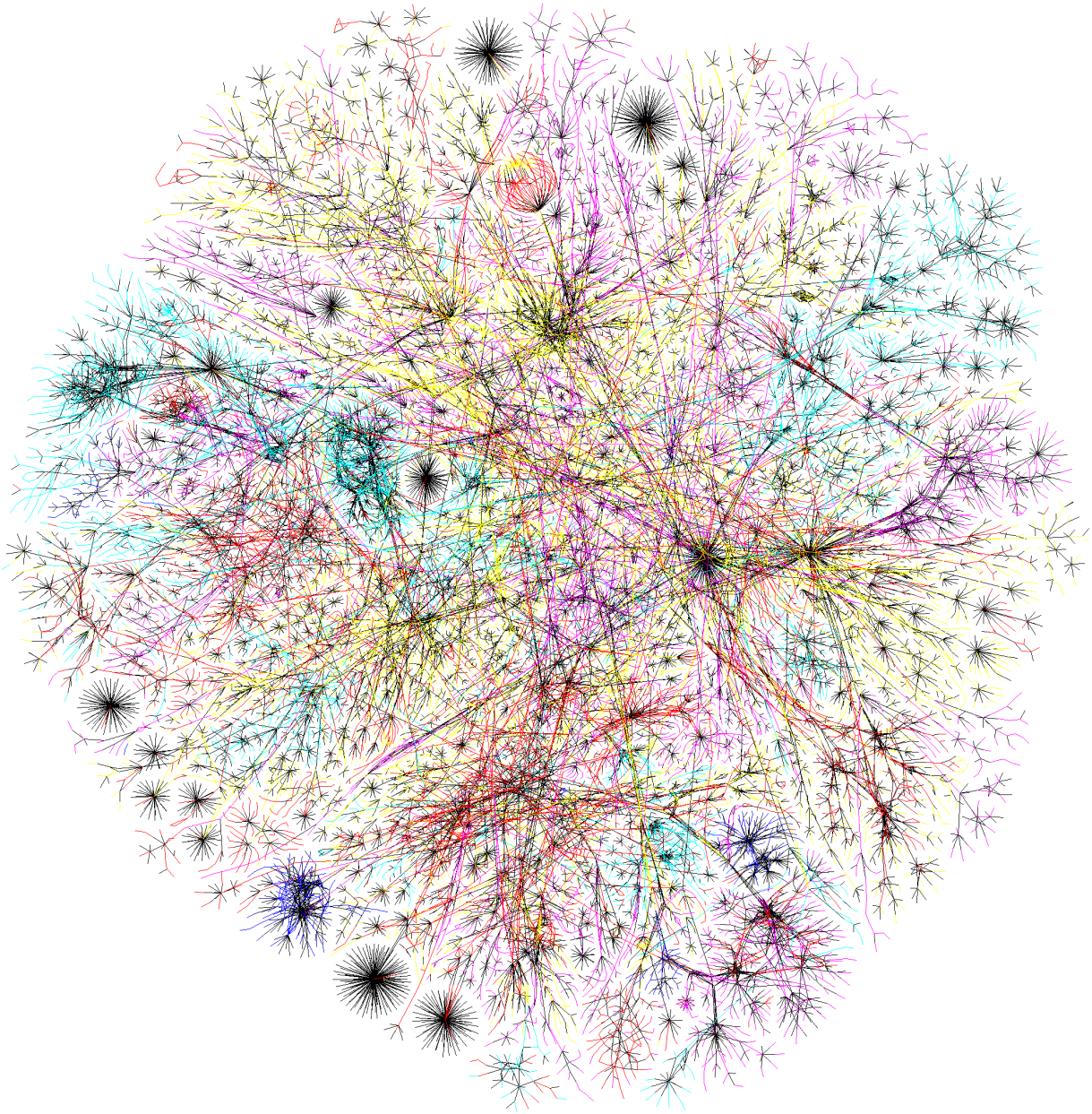

Рисунок 1.1: Пример топологии Интернета.

В основном в литературе для обозначения элемента структуры ГКС используется термин хост. Основываясь на предположении о том, что домен может состоять из

единственного хоста, далее в тексте по умолчанию будет использоваться термин хост и только в отдельно оговариваемых случаях — домен.

Сетевое приложение — распределенная система, разные части которой обмениваются между собой данными. Между сетевыми приложениями возникают потоки данных, совокупность которых образует трафик сети. Отметим, что на хосте могут работать несколько сетевых приложений (см. рисунок 1.2).

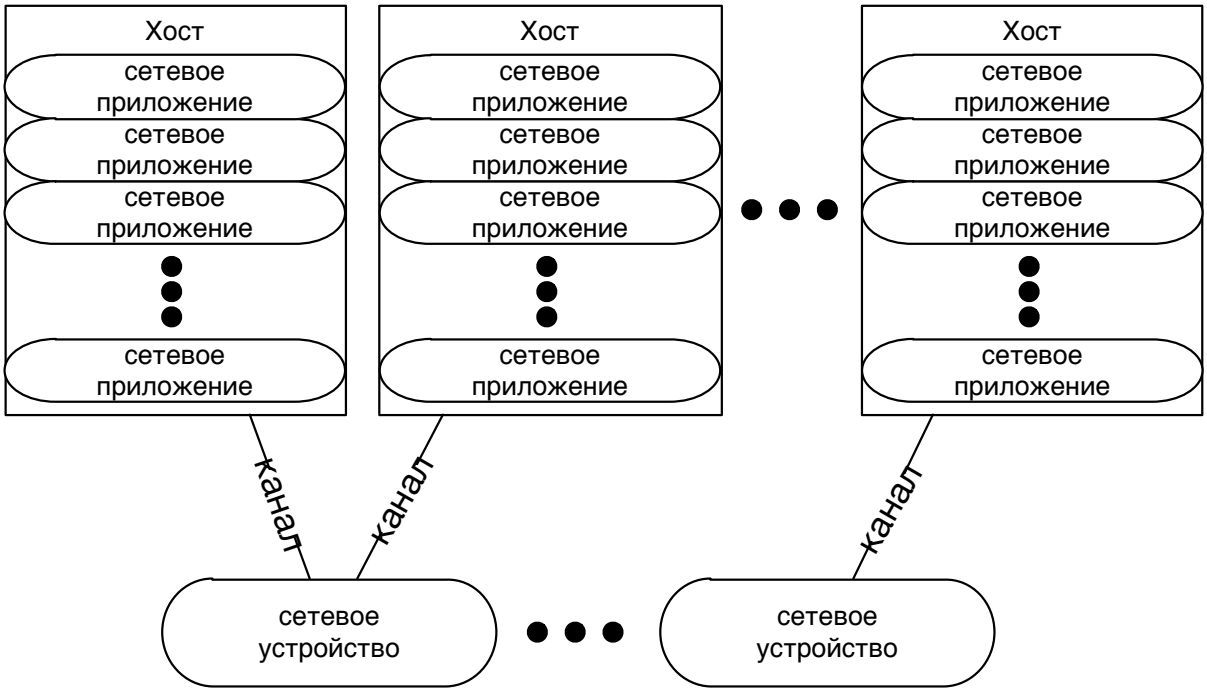

Рисунок 1.2: Структура ГКС.

Сетевая активность ГКС определяется потоками трафика, передаваемыми между хостами. Эти потоки мы будем разделять на два класса :

- легитимный трафик — трафик, генерируемый сетевыми приложениями и служебными сетевыми протоколами (ICMP, DNS, DHCP и др.);
- вредоносный трафик — трафик, генерируемый при передаче экземпляра вредоносного программного обеспечения (ВПО) и сетевой активности каждого экземпляра ВПО.

Современные ГКС обеспечивают инфраструктуру для работы сетевых приложений с высокими требованиями к сетевому сервису. Как хорошо известно из литературы, предоставляемый сетью сервис характеризуется понятием качества. Под качеством сервиса обычно понимается набор параметров сервиса, например [21]:

- доступность предоставляемого сервиса (service availability) определяет надежность соединений пользователей с сервис-провайдером;
- задержка (delay) характеризует интервал между приемом и передачей пакетов;
- отклонение от средней задержки (jitter) описывает возможные отклонения от времени задержки при передаче пакетов;
- производительность или пропускная способность (throughput) — скорость передачи пакетов в сети определяется для каждого потока трафика. Выделяют среднюю (average rate) и пиковую (peak rate) скорости. Можно встретить термин пропускная способность применительно к каналу; в этом случае имеется ввиду постоянная, заявленная скорость передачи трафика по каналу;
- доля потерянных пакетов (packet loss rate) — процент пакетов, сброшенных во время передачи по сети. В общем случае потери могут быть связанны со множеством причин: отказ оборудования, неправильная конфигурация оборудования, загрузка каналов, ошибки передачи и т.д. В данной работе будут рассматриваться только потери, происходящие вследствие перегрузок сети (congestion).

1.1.1 Требования к моделированию функционирования ГКС

На текущий момент существует ряд работ, посвященных построению моделей сети для исследования влияния различной сетевой активности на процессы функционирования сети [22, 23] [24]. Точность и детальность моделирования зависят от выбора математического аппарата, на основе которого строится имитационная модель. Моделирование функционирования ГКС выдвигает особые требования, которые необходимо учитывать при выборе математического аппарата:

- Масштабируемость. ГКС по определению является сетью с большим числом узлов, модель должна позволять варьировать в широком диапазоне число узлов в сети без особых усилий для перестроения модели.
- Ресурсоемкость. Требовательность к вычислительным ресурсам, необходимым для моделирования сети, заданного пользователем размера.
- Точность. Обычно подходы, используемые для построения моделей ГКС, обладают низкой точностью. Это связанно с высокой ресурсоемкостью «точных»

моделей [25]. В тоже время для исследования конкретных сетевых протоколов или для моделирования определенного рода сетевой активности (например, распространения ВПО) требуется точно, вплоть до работы с полями заголовка сетевого пакета, моделировать сетевое взаимодействие. В противном случае модель исследования сетевого протокола (сетевой активности) перестает быть адекватной.

Исходя из анализа открытых источников, были проанализированы существующие математические аппараты и средства реализации моделей сети по критериям: точность, ресурсоемкость и масштабируемость. С подробностями данного анализа можно познакомиться далее в тексте обзора.

1.2 Моделирование компьютерной сети

Модель – это объект или описание объекта системы для замещения (при определенных условиях, предложениях, гипотезах) одной системы (т.е. оригинала) другой системой для лучшего изучения оригинала или воспроизведения каких-либо его свойств [26, 27]. Модель должна строиться так, чтобы она наиболее полно воспроизводила те свойства объекта, которые необходимо изучить в соответствии с поставленной целью, либо которые оказывают определяющее влияние на изучаемые свойства [28].

Одним из основных подходов к моделированию является имитационное моделирование. Имитационная модель сети — это модель, которая описывает топологию, процессы передачи и обработки сетевого трафика, ресурсы сети. Имитационное моделирование сети позволяет проводить исследования состояния сети, опираясь на базовые правила функционирования сети. При использовании подобного подхода необходимые свойства и функционал сети описываются в терминах выбранного математического аппарата, следовательно, не требуется затрат на специализированное сетевое оборудование.

Результатом имитационного моделирования сети является собранные в ходе наблюдения за протекающими событиями данные о наиболее важных характеристиках сети: временах реакции, коэффициентах использования каналов и узлов, вероятности потерь пакетов и т.п.

В процессе имитационного моделирования сети могут быть определены следующие параметры:

- предельные пропускные способности различных сегментов сети;
- зависимость потерь пакетов от загрузки отдельных хостов и каналов;
- время отклика хостов сети;
- влияние установки новых хостов на перераспределение сетевых потоков;
- решение оптимизации топологии при возникновении узких мест в сети;
- выбор внутреннего протокола маршрутизации и его параметров (например, метрики определения оптимального маршрута);
- определение предельно допустимого числа пользователей того или иного ресурса сети;
- оценка влияния вредоносного трафика на работу сети, например, при распространении сетевого червя.

Обычно для имитационного моделирования сети необходимо задать набор входных параметров исследуемой сети (например, задержки на каналах связи, пропускные способности каналов связи, характеристики сетевого оборудования, принятая в сети политика маршрутизации). Набор параметров зависит от целей моделирования, и, соответственно, от интересующих нас выходных параметров модели. При этом, чем точнее будет воспроизведено функционирование сети, тем больше вычислительных ресурсов потребуется для обсчета построенной модели. Кроме того, необходимо сделать некоторые предположения относительно распределения загрузки для конкретных хостов и других сетевых элементов, задержек в каналах, времени обработки запросов в различных точках сети.

Как было сказано во введении, процесс построения имитационной модели делится на два этапа:

1. выбор математического аппарата и построение математической модели сети;
2. решение математической модели через ее программную реализацию на вычислительной системе.

1.2.1 Выбор математического аппарата для моделирования ГКС

В основе любой формальной модели лежит определенный математический аппарат, либо их комбинация [29]. Примером математического аппарата может быть:

- теория вероятностей;
- сети Петри;
- теория графов;
- теория автоматов.

Формальное представление модели описывает объект в терминах конкретного математического аппарата. При этом решение конкретной модели может быть представлено в одном из двух видов:

1. аналитическое решение, когда математическая модель имеет решение в аналитическом виде, то есть результат моделирования представляется в виде выражения в элементарных функциях. Эти функции определяют зависимость входных параметров модели от выходных. В литературе подобный подход можно встретить под названием аналитическое моделирование [30];
2. численное решение, когда математическая модель имеет решение в неаналитическом виде. Как правило, это решение носит приближенный характер. Данный подход применяется в случаях, когда аналитическое решение неизвестно или способ его нахождения трудоёмок.

Теория вероятностей

В настоящие время элементы теории вероятности (ТВ) используются для моделирования динамических характеристик сетевого трафика (например, интенсивность трафика, доля потерь, средняя задержка). Существуют работы, которые используют марковские процессы для моделирования конкретных протоколов, например, TCP [31]. Отличительной чертой таких моделей является их узконаправленность, что делает их применимыми только для специализированных задач.

При моделировании сетевого трафика, помимо использования марковских процессов, можно выделить еще два класса вероятностных моделей:

1. основанные на модулированных случайных процессах, например, модель Generaly Modulated Procces (GMP) [32];

2. учитывающие статистическое самоподобие сетевого трафика (фрактальные модели), например, модель на основе хаотических отображений Chaotic Map (CMAP) [33]

В основе GMP лежит идея управления законом распределения при помощи вспомогательного стохастического процесса. Стохастический процесс — это процесс изменения во времени состояния или характеристик некоторой системы под влиянием различных случайных факторов, для которого определена вероятность того или иного его течения [34]. При этом источник трафика может находиться в одном из множества состояний, задающих параметры случайного процесса генерации нагрузки. Переход между состояниями обуславливается дополнительным моделирующим случайным процессом.

Частным случаем GMP моделей являются обобщенные процессы с детерминированной модуляцией Generaly Modulated Deterministic Process (GMDP) [35] и процессы с марковской модуляцией Markov Modulated Procces (MMP) [36].

В GMDP моделях источник трафика может быть в одном из нескольких состояний. Система остается в конкретном состоянии в течение определенного интервала времени, при этом источник трафика генерирует нагрузку заданную распределением случайной величины, которая зависит от времени. Время в данной модели задается дискретно.

В MMP моделях управляющий процесс является марковским процессом с непрерывным временем и конечным множеством состояний. Марковский процесс — это случайный процесс, эволюция которого после любого заданного момента времени не зависит от эволюции в предшествовавшие моменты времени при условии, что значение процесса в этот момент фиксировано [34]. То есть "будущее" и "прошлое" процесса не зависят друг от друга при известном "настоящем". В этом случае текущий закон распределения нагрузки полностью определяется текущим состоянием марковского процесса.

Также находят применение модели Markov Modulated Poisson Process (MMPP) [37], которая представляет собой разновидность процессов с марковской модуляцией, в которых источник трафика генерирует пуассоновскую нагрузку.

Вторым классом моделей, основанных на ТВ, являются модели, использующие статистическое самоподобие сетевого трафика. Очевидно, что данные модели используются только для моделирования сетевого трафика. Свойством, характеризующим самоподобие сетевого трафика, является масштабная

инвариантность по времени, так что при изменении временной шкалы его структура остается неизменной [38].

Модели, использующие данное свойство, называются фрактальными. Примерами таких моделей являются Fractional Brown Motion (FBM) [39], и Fractional Gaussian Noise (FGN) [39].

FBM и FGN успешно использовались в гидрологических исследованиях, откуда их применение было перенесено в область моделирования сетевого трафика.

Общим недостатком моделей, базирующихся на ТВ, является их специфичность и отсутствие универсальности. Кроме того, применение их на практике приводит к большому объему исследовательской работы, требуемой для адаптации моделей к параметрам сетевой конфигурации. Из вышесказанного следует, что ТВ не удовлетворяет требованиям масштабируемости и точности к моделированию функционирования ГКС.

Теория массового обслуживания

Теория массового обслуживания (ТМО) — раздел теории вероятностей, целью исследований которого является выбор структуры системы обслуживания потоков сообщений, поступающих в систему, оценка длительности ожидания и длины очередей сообщений [40]. Как было сказано выше, сеть представляется в виде совокупности узлов. Узлы обмениваются между собой данными; порции данных будем называть сообщениями. Сообщение, пришедшее на узел, ждет некоторое время до того, как оно будет обработано. При этом может образоваться очередь таких сообщений, ожидающих обработки. Полное время передачи складывается из трех составляющих:

1. время распространения;
2. время обслуживания;
3. время ожидания.

Обычно модели на основе ТМО строят для определения среднего значения общего времени передачи сообщения (заявки) [41]. Стоит отметить, что при больших загрузках основной по продолжительности составляющей является время ожидания обслуживания. Для описания процессе обработки трафика сетевым устройством (например, маршрутизатором или коммутатором) используется понятие очередей, формальное определение можно найти в [42]. Наиболее распространенными

дисциплинами обслуживания очереди являются FIFO (First-In-First-Out), LIFO (Last-In-First-Out) и FIRO (First-In-Random-Out). Например, запись M/M/2 означает очередь, для которой времена прихода и обслуживания имеют экспоненциальное распределение. При этом имеется два узла, длина очереди и число клиентов могут быть сколь угодно большими, а дисциплина обслуживания является FIFO [43].

В настоящее время в сетях протоколы канального уровня используют основанные на разделении во времени методы доступа к среде. В этом случае, как и во всех случаях разделения ресурсов со случайным потоком запросов, возникают очереди. Для описания этого процесса тоже используются элементы ТМО. Например, механизм протокола Ethernet упрощенно описывается моделью типа M/M/1 - одноканальной моделью с пуассоновским потоком заявок и показательным законом распределения времени обслуживания. Она хорошо описывает процесс обработки случайно поступающих заявок на обслуживание системами с одним обслуживающим прибором со случайным временем обслуживания и буфером для хранения поступающих заявок на время, пока обслуживающий прибор занят выполнением другой заявки. Передающая среда Ethernet представлена в этой модели системой массового обслуживания, а пакеты соответствуют заявкам [30].

Используя элементы ТМО, зачастую удается получить решения достаточно широкого круга задач по исследованию рассматриваемой сети. В то же время он имеет ряд существенных недостатков, к числу которых относятся упрощения моделируемого объекта, которое ведет к потере точности моделирования [43]:

- упрощения моделируемого объекта: представление трафика как простейших потоков заявок; предположение об распределении длительностей обслуживания заявок и др.

- высокая ресурсоемкость для сложных моделей, например, представления в модели процесса функционирования стека TCP/IP в виде системы очередей, предъявляет существенные требования к вычислительным ресурсам [44];

- невозможность использования ТМО для описания специфицированной сетевой активности. Например, отправка определенного вида пакетов, либо генерация определенных сетевых запросов пользователем.

В итоге, ТМО не удовлетворяет требованиям точности и ресурсоемкости к моделированию функционирования ГКС.

Сети Петри

Часто исследователи для построение моделей КС обращаются к формальным системам, основанным на использовании математического аппарата сетей Петри (СП). Моделирование в СП осуществляется на событийном уровне. Определяются какие действия происходят в сети, какие состояния предшествовали этим действиям и какие состояния примет сеть после выполнения этих действий. Выполнение событийной модели в сетях Петри описывает поведение сети. Анализ результатов выполнения может сказать о том, в каких состояниях пребывала или не пребывала сеть, какие состояния в принципе не достижимы. Однако такой анализ не дает числовых характеристик, определяющих состояние КС [45].

Среди достоинств аппарата СП можно указать следующие:

- позволяет моделировать асинхронность и параллелизм (в СП могут одновременно и независимо друг от друга сработать несколько переходов);
- предоставляет возможность введения любой степени иерархической детализации описываемых сетевых компонент;
- позволяет формальными средствами доказывать существование или отсутствие определенных состояний СП (речь о простых СП).

Однако формальная модель СП в силу своей универсальности имеет ряд недостатков, затрудняющих практическое применение для построения модели сети. К основным таким недостаткам можно отнести следующие:

- высокая трудоемкость анализа сетей большой размерности;
- описательная мощность СП недостаточна для содержательного моделирования. Достаточно сложно в терминах меток описать поведение сложного сетевого приложения;
- обычные СП не отражают временные характеристики моделируемой системы, то есть нет модельного времени;
- невозможность проведения логических преобразований и, как следствие, невозможность управления продвижением фишек по сети. Данный недостаток существенно осложняет применения аппарата СП, например, для моделирования маршрутизации в сети.

Следовательно, СП не удовлетворяют требованиям точности к моделированию функционирования ГКС.

Теория графов

Другим распространенным математическим аппаратом для анализа структурных свойств сетей является теория графов (ТГ) [46]. Данный математический аппарат является основным для исследования структурных вопросов в области моделирования компьютерных сетей, например, для нахождения кратчайшего пути между узлами сети. В случае, когда необходимо помимо топологии моделировать сетевого трафик, ТГ используется совместно с другими математическими аппаратами, например, ТМО.

ТГ позволяет точно описывать структуру сети, представляя сеть в виде графа, обычно ориентированного, где дуги графа соответствуют каналам связи, а вершины — узлам сети. Однако при моделировании сети на уровне хостов, ТГ не позволяет предложить масштабируемое решение для сетей больших размеров. В первую очередь это связанно с высокой ресурсоемкостью построения графа модели сети, сравнимой по размеру с ГКС.

Перспективным для моделирования ГКС является подход, когда уровень абстракции понятия узла можно менять. В определенных случаях узел может представлять отдельный хост, в других — набор хостов и сетевых устройств. За счет абстрагирования от структуры связей внутри таких наборов мы получаем возможность масштабирования модели сети.

В итоге ТГ с точки зрения моделирования структуры ГКС полностью удовлетворяет требованиям, но с точки зрения моделирования процессов обработки и передачи сетевого трафика между узлами:

- не применима без использования дополнительных математических аппаратов;
- представляет компромисс между уровнем абстракции понятия узла и точности моделирования процессов обработки и передачи сетевого трафика.

Теория автоматов

Один из популярных математических аппаратов для исследования сетевых политик маршрутизации является теория автоматов (ТА). ТА предполагает представление каждой компоненты модели сети в виде автомата. ТА утверждает, что любой процесс может рассматриваться как последовательная смена состояний некоторого объекта

во времени. Следовательно, процесс обмена между узлами сети представляет собой последовательности некоторых операций, осуществляемых в определенные моменты времени. Поэтому для их моделирования могут быть использованы методы теории автоматов [47].

Обычно исследователи используют конечные автоматы, например, для описания сетевых протоколов [48]. Это связанно с тем, что конечные автоматы предоставляют удобный способ описания логики функционирования, поэтому подходят для моделирования составляющих модели сети, реализующих обмен сетевым трафиком (хостов, маршрутизаторов, коммутаторов).

При использовании ТА для предсказания поведения системы необходим вычислительный эксперимент [42]. Модели на основе конечных автоматов являются удобными средствами описания протоколов [49], но попытки описания сложных составляющих сети, например, сетевого стека, приводят к большому числу переходов состояний, что затрудняет реализацию автоматной модели.

Стоит отметить, что представить всю сеть, особенно масштабов ГКС, в виде единого автомата является чрезвычайно ресурсоемкой задачей, а при определенных размерах сети и вовсе невыполнимой. Для моделирования больших сетей возможно использование нескольких автоматов, что позволяет построить масштабируемое решение.

ТА представляется удобным инструментом для описания процесса обработки трафика сетевыми устройствами, а также описания процесса функционирования сетевых приложений. Однако, несмотря на универсальность ТА, использование данного математического аппарата без комбинации с другими не представляется возможным для моделирования функционирования ГКС.

Следовательно, ТА не удовлетворяет требованиям масштабируемости и ресурсоемкости к моделированию функционирования ГКС.

Выводы

Сравнительный анализ математических аппаратов с целью исследования возможности их использования для построения моделей ГКС, представлен в таблице 1.1.

(*) — зависит от необходимости моделирования сетевого трафика. В случае моделирования только структуры сети — это «чистый плюс».

Таблица 1.1: Сравнительный анализ математических аппаратов.

Математический аппарат	Точность	Ресурсоемкость	Масштабируемость
ТВ	-	+	-
ТМО	-	-	+
СП	-	-/+	+/-
ТГ	+/-	-/+ (*)	+
ТА	+	-	-

Итак, ни один из представленных подходов к построению математической модели функционирования сети не удовлетворяет требованиям к моделированию ГКС (см. рисунок 1.3). Необходимо предложить гибридный вариант, который будет удовлетворять требованиям: точности, масштабируемости и ресурсоемкости. Данный подход был предложен, и на его основе построена формальная модель ГКС, описанная в главе 2.

1.2.2 Средства реализации имитационной модели сети

Процесс реализации имитационной модели сети заключается в разработке программы, которая шаг за шагом воспроизводит события, происходящие в моделируемой сети. Обычно говорят о двух способах реализации имитационной модели:

1. использование специализированных языков моделирования;
2. использование специализированной системы имитационного моделирования. В данном случае разработчик строит модели, используя систему имитационного моделирования (СИМ), базируясь на математический аппарат, заложенный в основу конкретной СИМ.

Требования к средствам реализации имитационной модели сети

В настоящее время к средствам реализации имитационной модели предъявляется существенный ряд требований, как то:

- описание исследуемой системы на различных уровнях детализации без ограничений на их количество;

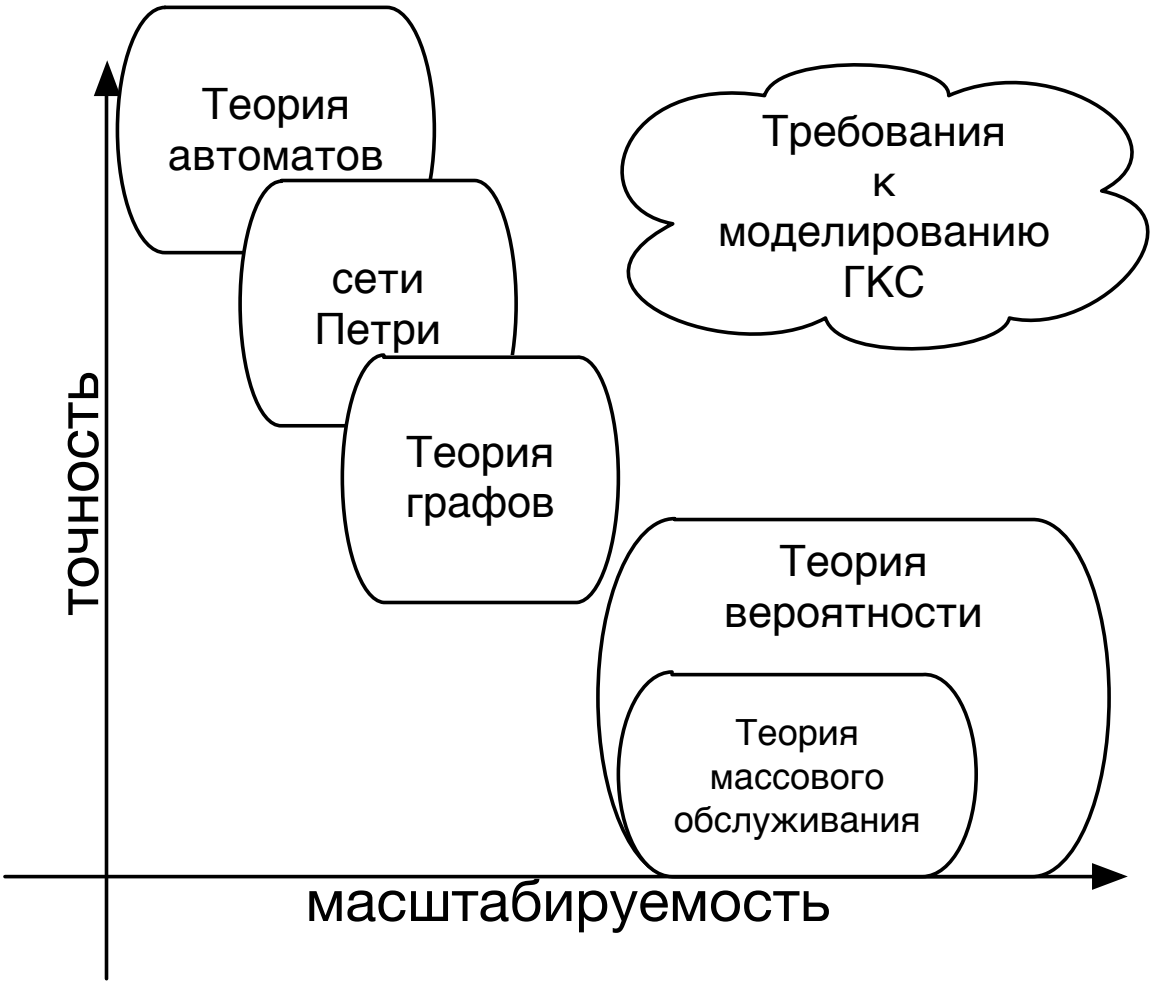

Рисунок 1.3: Обзор математических аппаратов имитационного моделирования.

- проектирование модели в графическом режиме на основе готовых блоков при минимальном объеме программирования;
- удобное и полное представление работы программного обеспечения сети (работы сетевых приложений);
- наглядное отображение процесса моделирования (использование методов анимации);
- снижение затрат времени моделирования на сбор статистики (накопление данных по требованию);
- распределенное моделирование сети на нескольких вычислителях;
- точное описание процесса обработки PDU;

- избегание процессов идентификации и калибровки модели сети.

В рамках данной работы добавляется требование: возможность моделирования сетей масштабов, сравнимых с ГКС.

Далее рассмотрим основных представителей средств реализации имитационных моделей с точки зрения предъявляемых требований.

Языки моделирования

Язык программирования — это формализованный язык, предназначенный для описания алгоритма для выполнения на вычислительной машине. Универсальные алгоритмические языки обеспечивают возможность реализации на вычислительных средствах разнообразных моделей (моделей сети в частности), однако, недостатком их использования в решении задач имитационного моделирования является сложность реализации и количество времени, затрачиваемое на разработку и исследование имитационных моделей по причине их универсальности. Примерами универсальных языков являются: C, C++, Java, Delphi и др.

Существуют специальные языки имитационного моделирования, которые облегчают процесс создания программной модели по сравнению с использованием универсальных языков программирования. Примерами языков имитационного моделирования могут служить такие языки, как SIMULA [50], GPSS [51] Языки имитационного моделирования обладают определенным рядом преимуществ по сравнению с универсальными языками:

- снижение трудоемкости реализации имитационных моделей;
- четкая классификация компонентов имитационной модели;
- обеспечение гибкости, необходимой для изменения реализации имитационной модели;
- описание связей между компонентами имитационной модели;
- возможность управления модельным временем;
- способность накапливать выходные данные;
- способность проводить статистический анализ накапливаемых данных;
- способность распределять выходные данные по заранее заданным форматам;

- возможность выявлять и регистрировать логические несоответствия и другие ситуации, связанные с ошибками.

Язык SIMULA предназначен для моделирования систем с дискретными событиями, т.е. систем, представляющих последовательность сменяемых друг друга мгновенных событий. Главную роль в языке играют параллельно функционирующие процессы, которые выступают в качестве компонентов моделируемой системы. Они имеют свою структуру данных и программу действий. В каждый момент времени активен только один процесс, который может вызывать и планировать новые процессы и события. Для этой цели в языке имеются планирующие и управляющие операторы.

Язык моделирования дискретных систем GPSS (General Purpose System Simulator) является одним из самых распространенных в мире языков имитационного моделирования. В основу языка GPSS положены результаты исследований разнообразных дискретных систем, показавшие, что любую систему можно описать с помощью необходимого набора абстрактных элементов или объектов. Формальными основными моделями, для имитации которых используется GPSS, являются системы массового обслуживания, конечные и вероятностные автоматы, сети Петри, агрегаты.

К недостаткам языков имитационного моделирования можно отнести:

- использование только стандартных форм вывода результатов моделирования;
- недостаточная распространенность языков моделирования, следовательно, затруднен перенос моделей на другие платформы;
- необходимость дополнительного обучения языкам моделирования;
- отсутствие гибкости и широких возможностей, присущих универсальным языкам программирования;
- отсутствие специализации: набора моделей базовых объектов, характерных для области моделирования.

Стоит отметить, что на данный момент не существует специализированных языков имитационного моделирования или библиотек к общим языкам имитационного моделирования для моделирования ГКС. Исходя из недостатков данных языков, можно сделать вывод лишь о частичной применимости данного средства для моделирования ГКС, например, в качестве промежуточного представления модели ГКС на этапе построение модели. Подобный подход уберет необходимость исследователя

работать с большим набором данных, однако оставит возможность использовать уже имеющиеся наработки и библиотеки, написанные для конкретного языка имитационного моделирования.

Системы имитационного моделирования

Система имитационного моделирования (СИМ) - комплекс программных средств для создания имитационной модели и ее симуляции (имитации). При использовании СИМ упрощается процесс построения имитационной модели, так как в них обычно уже реализованы атомарные компоненты для построения модели КС: каналы, сетевые устройства, модели протоколов и приложений. СИМ строят модель сетей на основе:

- исходных данных о ее топологии, алгоритмах и методах обработки трафика;
- интенсивности потоков запросов между узлами сети;
- пропускной способности и номинальной задержки каналов связи.

По способу взаимодействия с пользователем СИМ могут быть автономными и интерактивными. Автономные модели не требуют вмешательства исследователя после определения режима моделирования и задания исходных данных. Взаимодействие пользователя с такими моделями сводится только к вводу исходной информации и управлению началом и окончанием процесса моделирования. Интерактивные модели предусматривают диалог с пользователем в том или ином режиме в соответствии со сценарием моделирования, позволяющий пользователю приостанавливать моделирование, изменять значения параметров модели, корректировать перечень регистрируемых данных и т.д, примером могут служить различные симуляторы.

Применительно к сетям СИМ могут воспроизводить следующую функциональность сети:

- процессы генерации сообщений сетевыми приложениями;
- разбиение сообщений на пакеты и кадры определенных протоколов;
- задержки, связанные с обработкой сообщений, пакетов внутри хостов и сетевых устройств;
- процесс обработки поступающих пакетов сетевыми устройствами.

Результатом работы СИМ являются собранные в ходе моделирования статистические данные об интересующих характеристиках КС:

- задержка при передаче пакета;
- утилизация ресурсов каналов и узлов КС;
- доля потерянных пакетов и т.п.

По функциональности СИМ, используемые при исследовании КС, например, можно представить в виде двух классов:

- Системы, моделирующие отдельные элементы (компоненты) КС. Например, Linuxbridge [52], OpenVSwitch [53], ClickRouter [54]. Эти системы моделируют функционирование распространенных типов маршрутизаторов, каналов связи, методов доступа, протоколов. Модели отдельных элементов сети создаются на основании различных данных: результатов тестовых испытаний реальных устройств, анализа принципов их работы, аналитических соотношений. В результате создается библиотека типовых элементов сети, которые можно настраивать с помощью заранее предусмотренных в моделях параметров.
- Системы, моделирующие КС целиком. Например, системы NS-3 [55], NetEmulab [56].

На сегодняшний день известно достаточно много сетевых симуляторов, например: OPNET [57], OMNET [58], OMNET++ [58], NS-2 [59], NS-3.

Одним из самых распространённых является СИМ NS-2, созданная в 80-х годах. NS-2 является свободно распространяемым программным обеспечением, имеет достаточно большое и развитое сообщество, и как следствие — большое количество модулей, дополнений, готовых к использованию. NS-2 является объектно-ориентированным программным обеспечением, использующим два языка программирования: ее ядро реализовано на языке C++, а языком скриптов является Object oriented Tool Command Language (Otcl) [60], он используется в качестве интерпретатора. Такой подход является компромиссом между удобством использования и скоростью. В NS-2 в качестве системного языка используется C++, позволяющий обеспечить:

- высокую производительность;
- работу с сетевыми пакетами на низком уровне абстракции модели;

- возможность модифицирования ядра NS-2 с целью поддержки новых функций и протоколов.

NS-2 (см. рисунок 1.4) содержит большой набор протоколов, таких как TCP, UDP, а так же позволяет реализовывать различные типы приложений. Среди них: ftp, telnet, http, которые используют TCP в качестве основного транспортного протокола. В NS-2 на уровне ядра реализованы многие известные протоколы. Среди наиболее актуальных на данный момент можно отметить следующие: MPLS, IPv6, OSPF, RSVP, протоколы беспроводной связи. Реализовано семейство дисциплин обслуживания очередей [61]: RED, WFQ, CBQ, SFQ [62] и т.д. Также возможна реализация собственного протокола в рамках программы.

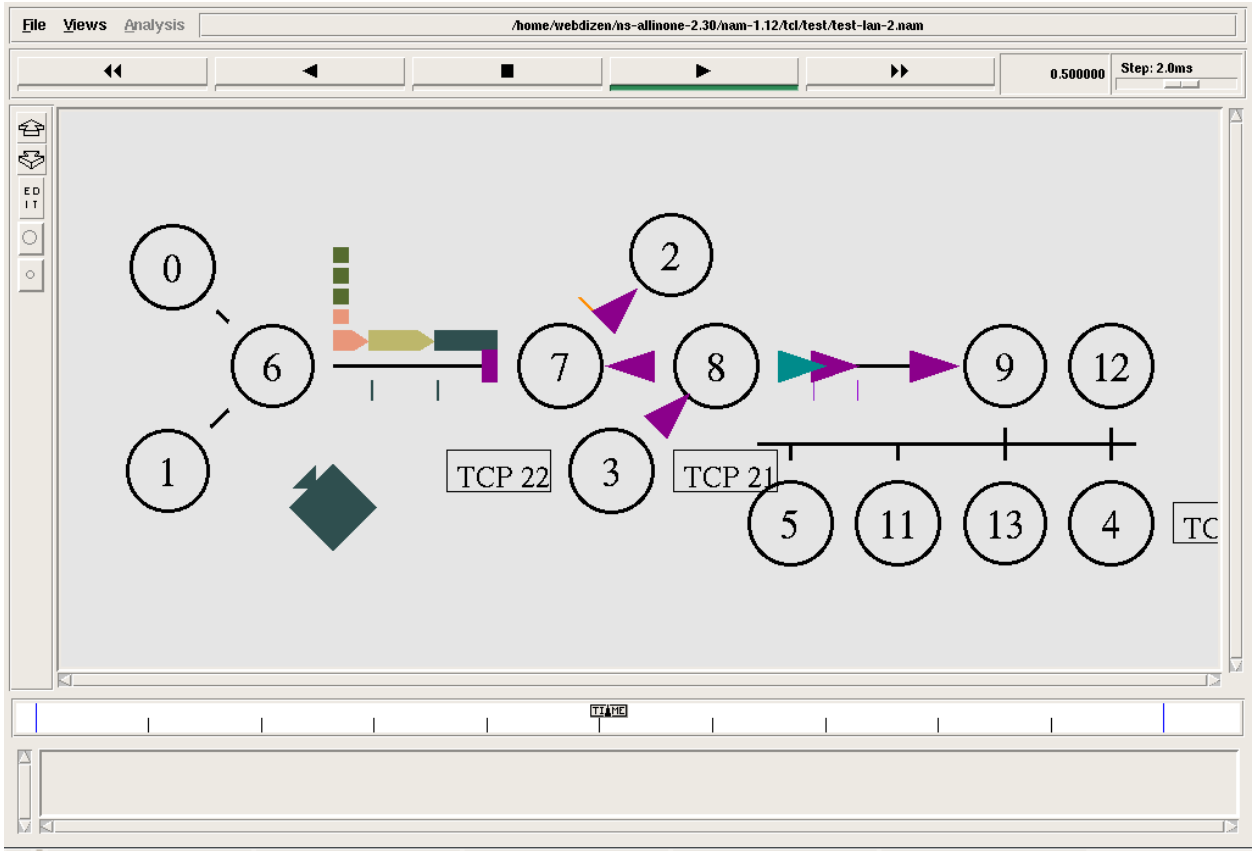

Рисунок 1.4: Интерфейс системы NS-2.

В отличии от NS-2, в NS-3 (см. рисунок 1.5) помимо C++ может использоваться Python. Оба языка в симуляторе равноправны и принимаются для описания моделей телекоммуникационных систем. Присутствует реализация различных типов Mesh-сетей на основе стека протоколов 802.11s. В NS-3 разработан модуль FlowMonitor, предоставляющий 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 [63] и PyViz [64].

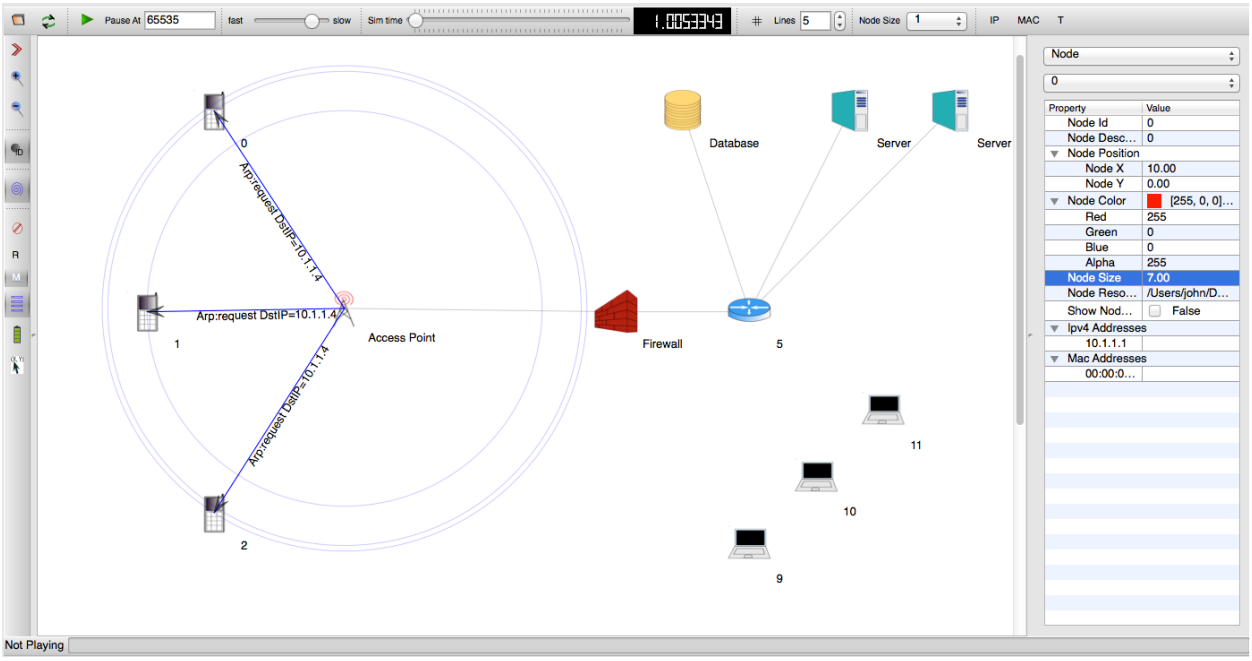

Рисунок 1.5: Интерфейс системы NS-3.

OPNET — СИМ для проектирования и моделирования локальных и глобальных сетей, компьютерных систем, приложений и распределенных систем. Имеет возможность импорта и экспорта данных о топологии и сетевом трафике. OPNET (см. рисунок 1.6) способна моделировать иерархические, многопротокольные, локальные и глобальные сети, с учетом алгоритмов маршрутизации. Поддерживается возможность генерации сетевой топологии - кольца, звезды, случайной топологии сети. Существенным моментом является то, что данная система является коммерческой.

Система OmNET++ является совокупностью программных библиотек, в которых хранятся функции для работы с имитационными моделями сетей. OmNET++ (см. рисунок 1.7) является объектно-ориентированным программным обеспечением, использующим язык программирования C++. Система OmNET++ является СИМ общего назначения (а не только КС). В данной системе реализован практически весь функционал, который был указан для предыдущих систем.

Система моделирования AnyLogic [65] — отечественная система имитационного моделирования, которая предназначена для разработки и исследования имитационных моделей. AnyLogic (см. рисунок 1.8) была разработана на основе идей в области теории параллельных взаимодействующих процессов и теории гибридных систем. AnyLogic

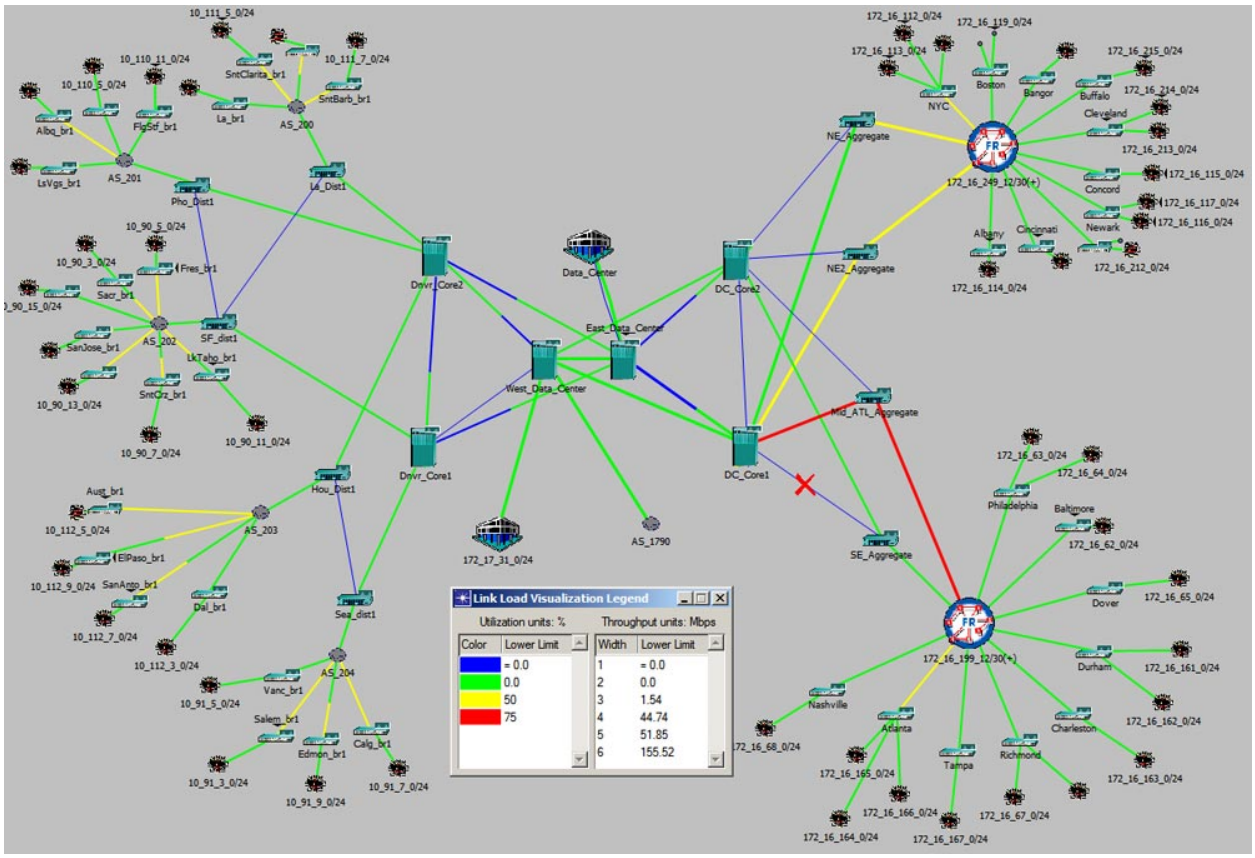

Рисунок 1.6: Интерфейс системы OPNET.

основана на объектно-ориентированной концепции, представляющая модели как набор взаимодействующих, параллельно функционирующих активностей.

Графическая среда моделирования поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов, оптимизацию параметров относительно некоторого критерия. При разработке модели на AnyLogic можно использовать концепции и средства из нескольких классических областей имитационного моделирования:

- динамических систем;
- дискретно-событийного моделирования;
- системной динамики;
- агентного моделирования.

В итоге можно сделать вывод, что существующие наиболее популярные СИМ для моделирования сети очень близки по возможностям между собой. Однако

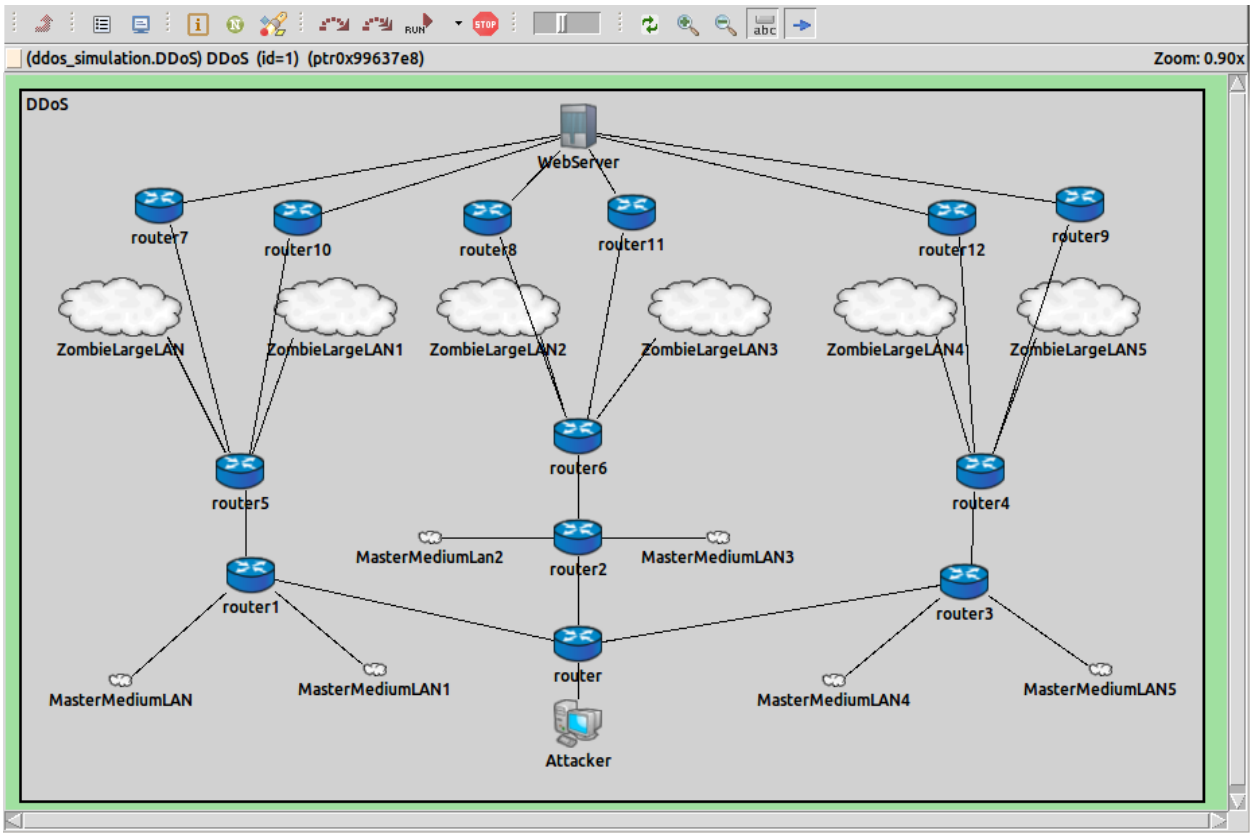

Рисунок 1.7: Интерфейс системы OmNET++.

все они обладают рядом недостатков, которые не позволяют их использовать для моделирования ГКС (см. рисунок 1.9):

- ограниченные возможности описания функционирования сетевых приложений в узлах сети;
- скрытость алгоритмов функционирования ряда модулей и отсутствие исходных кодов инструментальных средств (верно для OPNET);
- отсутствие в некоторых случаях возможности создания собственных блоков и ограничения на выбор уровней детализации создаваемый модулей;
- выполнение на единственном вычислителе, отсутствие интерфейсов сорганизованной работы одновременно нескольких вычислителей, моделирующих разные части одной модели сети.

Качество результатов имитационного моделирования функционирования в значительной степени зависит от точности исходных данных о сети, переданных на вход СИМ сети. В вышеописанных СИМ [57, 58, 66, 67] моделируемая сеть описывается

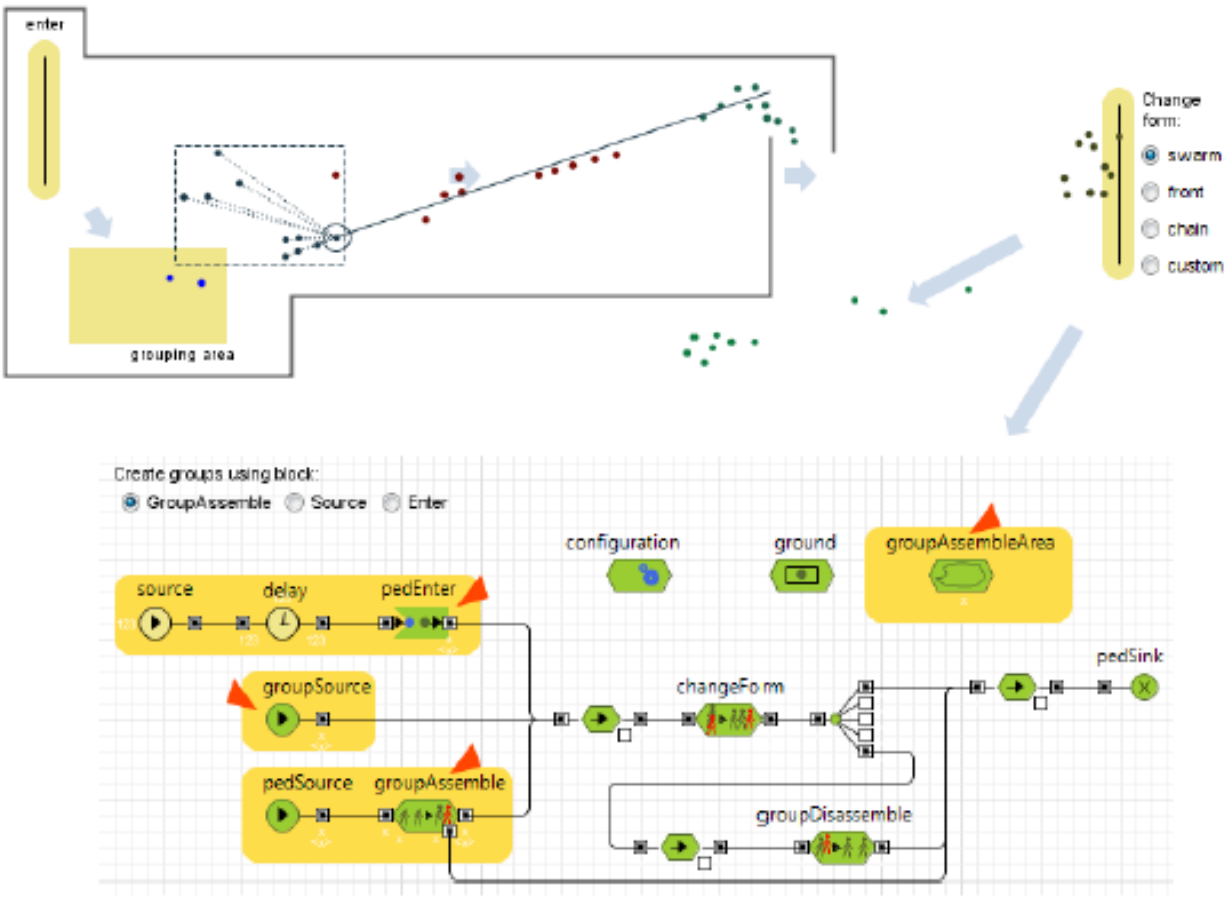

Рисунок 1.8: Интерфейс системы AnyLogic.

на уровне отдельных хостов и маршрутизаторов, что непригодно для моделирования ГКС по следующим причинам:

- чрезмерная вычислительная сложность. Согласно исследованиям, проведенным в работе [68], максимальный размер сети, при котором возможно применение подобного подхода, составляет 10^4 узлов. Размер ГКС — 10^5 и более, вследствие чего существующие вычислительные системы не позволяют получить решение модели за приемлемое время;

- сложность сбора данных для описания структуры сети и определения потоков трафика. Для того чтобы провести моделирование сети на уровне хостов, необходимо определить топологию сети, а также для каждого хоста указать его основные свойства, в частности набор установленных сетевых приложений, их сетевую активность и т.д. Настолько подробные сведения обо всех хостах в масштабах ГКС на практике получить очень сложно. Более того, описать

корректно подобную сеть невозможно из-за трудоемкости и высокой вероятности ошибки.

Сравнительный анализ средств реализации имитационных моделей сетей с целью исследования возможности их использования для построения моделей ГКС представлен в таблице 1.2.

Таблица 1.2: Сравнительный анализ средств реализации имитационных моделей сетей.

Требование	SIMULA	GSPP	NS-2	NS-3	OPNET	OMnet++	Anylogic
различные уровни детализации	+	+	-	-	-	-	+
графический режим	-	-	+	+	+	+	+
представление работы ПО сети	-/+	-/+	+/-	+/-	-/+	-/+	-
отображение процесса моделирования	-	-	+	+	+	-	+
накопление данных по требованию	+	+	-	-	+	-	-
распределенное моделирование	-/+	-/+	-	-	-	-	-
описание процесса обработки PDU	+/-	+/-	+	+	+	+	-
избегание процессов идентификации и калибровки (*)	-	-	-/+	-/+	-/+	-/+	-/+
моделирование сетей размера ГКС	-/+	-/+	-	-	-	-	-

(*) — «-/+» означает отсутствие калибровки только на заранее подготовленных моделей сетевых приложений и протоколов.

Следовательно, необходима разработка собственной СИМ, удовлетворяющей требованиям описанным в 1.2.2. Данная СИМ должна быть свободна от указанных недостатков и по своим выразительным возможностям не должна существенно уступать рассмотренным СИМ.

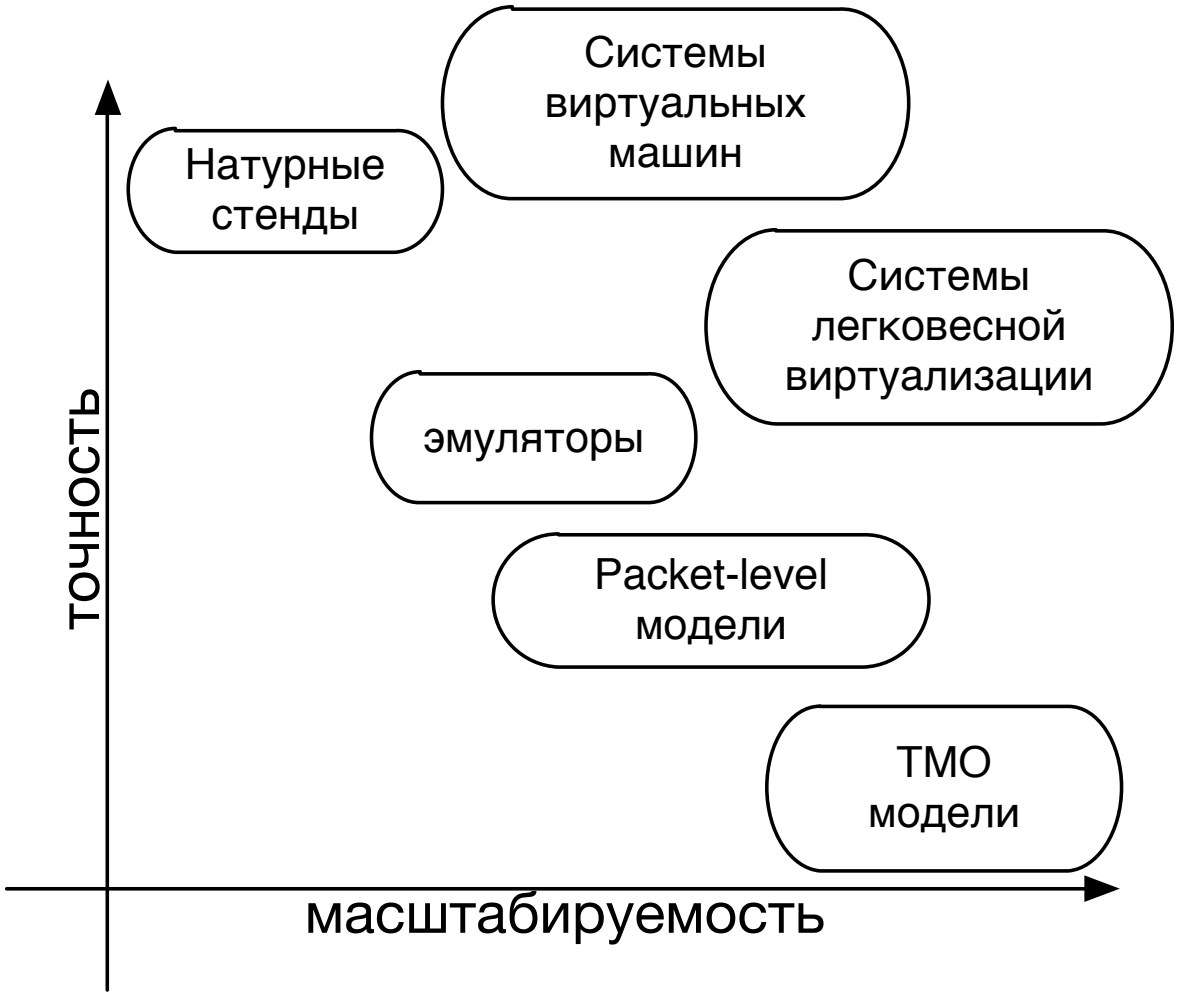

Рисунок 1.9: Обзор средств реализации имитационной модели.

Hi-Fi системы имитационного моделирования

Напомним что, моделирование процесса функционирования сети заключается в возможности воспроизводить:

- топологию и ресурсы сети;
- процессы обработки сетевого трафика:
- процессы передачи сетевого трафика:

- сетевой трафика:
  - способы генерации трафика/нагрузки в сети;
  - инициализацию сетевых приложений.

СИМ, моделирующие с высокой точностью процесс функционирование сети, называют high-fidelity системами (или Hi-Fi СИМ). Под высокой точностью обычно понимается точность реального оборудования, либо эмуляторов оборудования. Hi-Fi СИМ являются удобным инструментом для исследования, так как модели построенные при помощи Hi-Fi СИМ либо существенно упрощают, либо вовсе не требуют доказательство корректности и адекватности построенной модели сети. Как известно, процесс доказательства корректности и адекватности построенной модели в некоторых случаях (например, при построении модели ГКС) может требовать больших трудозатрат, чем процесс построения модели [19].

Одной из основных особенностей ГКС является большой размер, следовательно, ресурсов одного вычислителя для имитационного моделирования ГКС недостаточно. Несмотря на привлекательность Hi-Fi СИМ, существующие реализации применимы только для моделирования сетей малого размера. Очевидным вариантном решения данной проблемы является разработка распределенного варианта Hi-Fi СИМ с целью задействовать в процессе моделирования несколько вычислителей. Для любой распределенной системы является актуальным вопрос синхронизации времени между компонентами данной системы.

На данный момент единственной доступной реализацией Hi-Fi подхода является система Mininet. Система Mininet, разработанная в университете Стэнфорда [69], — система прототипирования компьютерной сети с открытым исходным кодом [70], основанная на техниках легковесной виртуализации в операционной системе машины, на которой выполняется процесс моделирования [71]. Данная система разрабатывалась для быстрого построения сегментов программно-конфигурируемых сетей (ПКС сетей). Целью построения данных сегментов является разработка и отладка сетевых приложений для ПКС контроллера, основного компонента ПКС сегмента [72–74], отвечающего за маршрутизацию пакетов. Использование метода легковесной виртуализации для моделирования впервые применено в системе Mininet. Методы, используемые в данном подходе, позволяют детально моделировать процесс функционирования сети.

Главной особенностью системы Mininet является возможность создавать сегменты виртуальной сети в рамках одной локальной машины. Эмпирические исследования показали, что Mininet может строить сеть, топология которой состоит не более чем из 2000 узлов.

Система Mininet использует две основные техники виртуализации, реализованные в ОС Linux [75]:

- Network namespaces. Network namespace — это логически отделенный от других стек сетевых протоколов в ОС Linux. Представляет собой контейнерную виртуализацию для сетевых интерфейсов. При помощи данной виртуализации эмулируется полностью сетевой стек: сетевые интерфейсы, таблица маршрутизации, сетевой экран и т.д. Работа осуществляется на уровне ядра ОС и для определенных процессов ОС;
- Virtual Ethernet pairs. Два контейнера виртуализации (виртуальных сетевых интерфейса) соединенных между собой. Представляется в виде специального виртуального устройства, соединяющего мост с виртуальным интерфейсом контейнера.

Одно из существенных преимуществ системы Mininet — это отсутствие необходимости вносить изменения в разрабатываемое или тестируемое сетевое приложение (при переносе на реальное оборудование). Это достигается за счет обработки сетевого трафика в узлах моделируемой сети. Стоит отметить, что Mininet не является СИМ. Основной задачей этой системой является создание тестовой площадки для отладки новых сетевых протоколов и приложений. Следовательно, в системе Mininet не было уделено должное внимание методам управления временем, библиотекам специализированных сервисов и т.п. В сочетании с требованием масштабируемости и ограничения размера топологии Mininet сети (до 2000 узлов) делает ее неприменимой для моделирования ГКС.

Одной из основных особенностей ГКС является большое количество узлов, следовательно, ресурсов одного вычислителя для имитационного моделирования ГКС недостаточно. Очевидным вариантном решения данной проблемы является разработка распределенного варианта Hi-Fi СИМ с целью задействовать в процессе моделирования несколько вычислителей.

Выше были представлены только наиболее существенные представители СИМ сети. Подробное описания других рассмотренных систем моделирования можно найти в [76].

1.3 Выводы

В результате обзора математических аппаратов для моделировании ГКС было показано, что невозможно выбрать единственный математический аппарат для моделирования функционирования ГКС. Предлагается использовать комбинацию математических аппаратов: теории графов, теории массового обслуживания, теории автоматов. Предполагается, что модель, построенная с использованием возможности данной комбинации математических аппаратов удовлетворит требованиям точности, ресурсоемкости и масштабируемости модели ГКС. Построенная модель ГКС описана в разделе 2.2. Доказательство этого предложения приводится в экспериментальном исследовании, представленном в главе 4.

В результате обзора существующих средств реализации имитационной модели обнаружилось, что ни одно не удовлетворяет описанным требованиям (см. раздел 1.2.2). Наиболее перспективными для реализации модели ГКС были выбраны Hi-Fi СИМ (см. рисунок 1.9). Однако по причине отсутствия готовых СИМ, удовлетворяющих требованиям модели ГКС, необходима разработка и реализация собственной СИМ. Описание реализованной СИМ представлено в главе 3.

Глава 2

Формальная модель глобальной компьютерной сети

В данной главе приведено описание формальной модели ГКС при помощи математических аппаратов: теории графов, элементов теории массового обслуживания, а также теории автоматов. Основная цель данной формальной модели — спрогнозировать динамику распространения ВПО. Приведены основные понятия и термины. Доказано, что задача исследования динамики распространения ВПО в построенной формальной модели имеет решение.

Будем называть доменом (см. рисунок 2.1) — множество хостов и сетевых устройств. Хосты и сетевые устройства обмениваются между собой данными. Процесс обмена представляется в виде потока данных между доменами, который будем называть поток данных или поток. Каждый домен имеет определенный набор истоков и стоков:

- исток — это точка подключения канала к домену, через которую поток поступает в домен;
- сток — это точка подключения канала к домену, через которую поток уходит из домена.

Допускается наличие нескольких стоков и/или истоков у одного домена.

Домены связаны между собой каналами связи или каналами. По каждому каналу поток может проходить только в одном направлении. Каждый канал имеет фиксированную пропускную способность. Канал может быть подключен только к одному стоку и одному истоку соответствующего домена. Другими словами, канал не может соединять один и тот же сток с несколькими истоками как одного, так и разных

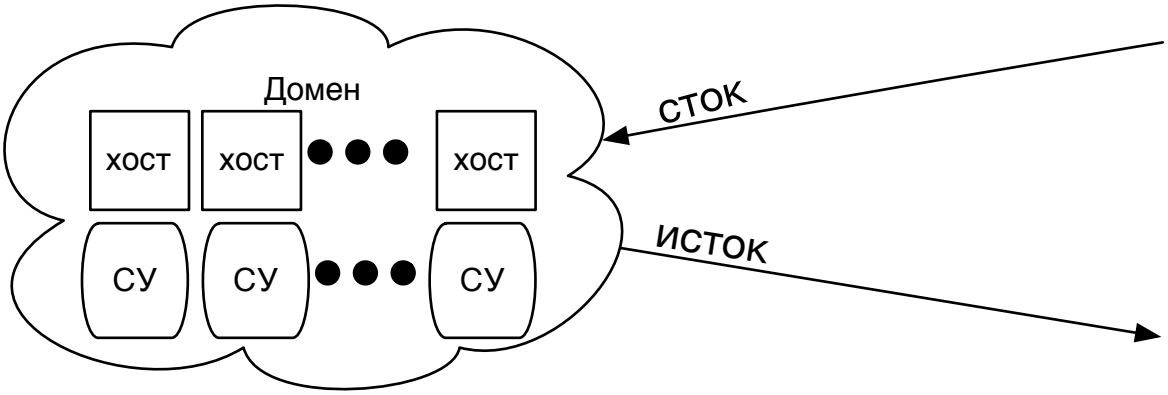

Рисунок 2.1: Структура домена.

доменов. Верно и обратное: несколько стоков не может быть соединено с одним и тем же истоком.

В модели каждый соединяющий домены канал однонаправлен и ориентирован от стока к истоку. Домены, соединенные каналом, будем называть смежными. Дуплексный канал в модели ГКС будет представлен двумя разнонаправленными каналами. В этих предположениях сеть доменов может быть описана в терминах конечно порожденной частичной алгебры [77].

Трафик в сети рассматривается в виде совокупности потоков [15], где каждый поток характеризует обмен данными между сетевыми приложениями. Сетевое приложение — это источник и/или потребитель сетевого трафика, ассоциированный с доменом. Совокупность потоков, проходящих через один и тот же канал, будем называть трафиком канала. Под интенсивностью потока будем понимать количество данных, прошедших по каналу в единицу времени.

Правомерность использования понятия потока основана на том, что в современных сетях преобладает использование виртуальных соединений между взаимодействующими сетевыми объектами. Маршрутизация на уровне пакета в настоящее время на практике применяется редко [78]. Говоря о потоке, мы фокусируем внимание на взаимодействии двух сетевых объектов, а не пытаемся охватить все взаимодействия, проходящие через определенный канал.

Рассматривая сеть на уровне доменов, мы пренебрегаем локальным трафиком внутри домена, который для оценки динамики распространения ВПО и влияния трафика ВПО на уровне потоков не важен. В рамках поставленной задачи необходимо

оценить динамику заражения ВПО хостов внутри домена, что достаточно точно описывается эпидемическими моделями [79], которые будут рассмотрены ниже.

Заметим, что домен в предлагаемом подходе играет двоякую роль. С одной стороны, в нем сконцентрированы ресурсы (хосты, сетевые приложения и т.п.), с другой — он выполняет функции коммутатора.

Предположим, что у нас есть сеть взаимосвязанных доменов, по которым проходят потоки различных типов: легитимные и вредоносные. Поток ориентирован от стока одного домена к истоку смежного домена. Внутри домена может быть порожден новый поток. Некоторые потоки, вошедшие в домен, могут быть «потреблены» доменом, который является конечной точкой маршрута, или же часть потока может быть потеряна по причине исчерпания ресурсов домена или перегрузкой канала. Динамику обработки потоков внутри домена будем описывать в виде автомата, который будет подробно рассмотрен ниже.

В рамках построенной модели рассматривается следующая задача: при заданных начальных условиях оценить динамику распространения ВПО среди хостов в ГКС и долю вредоносного потока в общем трафике ГКС.

2.1 Исходные предположения

При построении имитационных моделей критичным является вопрос о представлении модельного времени. Будем предполагать, что в сети один наблюдатель [77] с едиными часами, значение которых изменяется дискретно. С целью упростить технику описания нашей модели будем считать, что время едино и изменяется дискретами, которые мы будем обозначать Δt.

Сразу отметим, что это достаточно сильное предположение, так как на практике оно уместно для части сети, но не для всей ГКС в целом. Предположение о едином наблюдателе для одного домена оправдано в случае использования распределенной системы вычислителей для реализации модели. В этом случае у каждого домена будут свои часы, свой наблюдатель. Случай моделирования с несколькими часами и описанием процесса их синхронизации рассмотрен в разделе 3.4.1.

Обозначать время будем t, где $t \in \mathbb{N}$. Под словами «текущий момент времени» будем понимать время, прошедшее за дискрет Δt.

2.2 Описание формальной модели ГКС

Для построения математической модели функционирования ГКС нам необходимо определить математические абстракции для понятий: домен, канал и поток. На основе этих абстракций нужно предложить способы описания коммутации потоков в доменах для решения задач оценки:

- динамики изменения числа зараженных хостов под влиянием вредоносного потока для каждого домена;
- динамики изменения доли вредоносного потока в общем объеме сетевого трафика в ГКС.

2.2.1 Описание

Будем представлять ГКС (см. рисунок 2.2) в виде ориентированного мультиграфа G. Направление дуг графа G указывает направление передачи трафика. Пусть некоторые вершины данного графа будут отмечены особым образом. Отмеченные вершины назовем полюсами, а остальные вершины – внутренними:

$$G = (D, L, P),$$

где:

- $D = \{d_1, d_2, ..., d_n\}$ — множество доменов в ГКС,
- $P = \{p_1, p_2, ..., p_q\}$ — множество полюсов в ГКС,
- $L = \{l_1, l_2, .., l_m\}$ — множество каналов в ГКС, т.е. это упорядоченная пара доменов (полюсов) (v_1, v_2), где $v_1, v_2 \in D \cup P$. Можно сказать, что канал ведет от домена (полюса) v_1 к домену (полюсу) v_2.

У каждого домена и полюса есть стоки и истоки. Будем обозначать:

- $in(v_*)$, где $v_* \in D \cup P$ — множество истоков домена (полюса);
- $out(v_*)$, где $v_* \in D \cup P$ — множество стоков домена (полюса).

Каждый канал $l \in L$ в соответствии со своей направленностью соединяет сток одного домена с истоком другого домена. Определим канал как вектор, скалярная

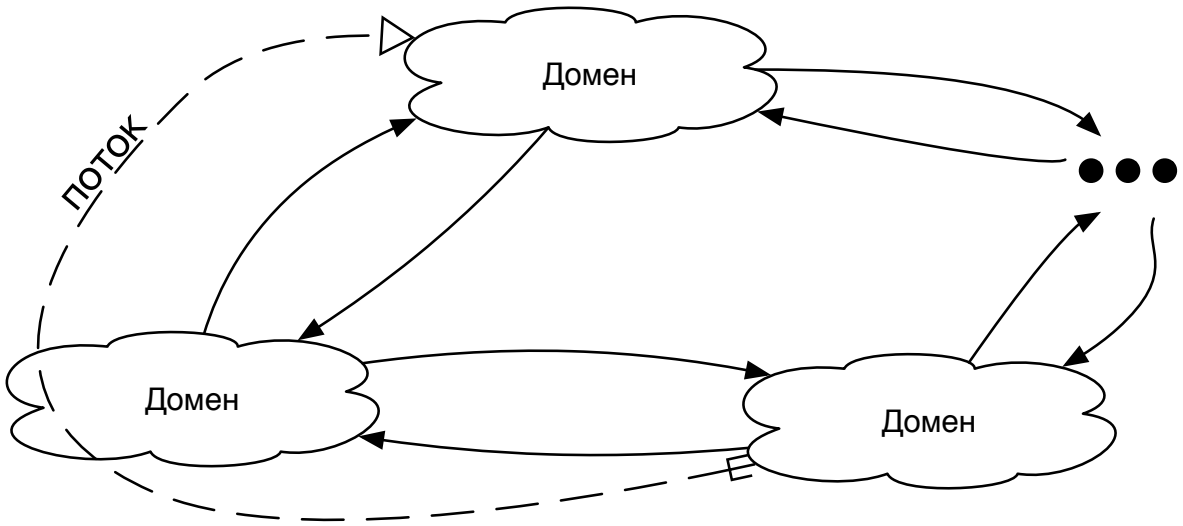

Рисунок 2.2: Общая структура модели ГКС.

компонента которого определяет пропускную способность канала, а направление указывает направление движения трафика в канале:

$$\bar{l} = (TP_l),$$

где $TP_l \in \mathbb{R}_+$ — пропускная способность канала; будем считать ее величиной постоянной.

Введем следующие обозначения:

- N_d — число хостов в домене d;
- I_d — множество типов ВПО, которые могут заразить домен d. Будем предполагать, что $\forall d_1, d_2 : d_1, d_2 \in D \cup P \Rightarrow I_{d_1} = I_{d_2}$.

Обозначим S_d, где $d \in D$, состояние домена как пятерку:

$$S_d = (n, SV, IV, Res, FPS),$$

где:

- $n \in [1, N_d]$ — число активных хостов в домене. Хост считаем активным, если на нем функционирует сетевое приложение, являющееся источником или потребителем потока;
- $SV : |SV| = |I_d|$ и $\forall i : SV_i \in [0, 1], i \in I_d$ — параметр, определяющий защищенность домена. $\forall i : |1 - SV_i|$ представляет собой вероятность заражения хоста ВПО типа

$i \in I_d$. Тем самым предполагается, что каждый хост в домене одинаково защищен от вредоносной активности одного и того же типа;

- $IV : |IV| = |I_d|$ и $\forall i : IV_i(t) \in [0, N_d], i \in I_d$ — параметр, определяющий число хостов в домене, зараженных ВПО i-го типа. Отметим, что если вышеперечисленные компоненты состояния являются исходными и задаваемыми в том или ином виде до начала моделирования, то этот компонент вычисляется на каждом дискрете времени.

2.2.2 Описание модели заражения домена

Для описания процесса заражения хостов внутри домена воспользуемся эпидемической моделью (см. рисунок 2.3) [80]. Введем параметры, необходимые для вычисления популяции зараженных хостов в домене с течением времени:

- $\frac{IV_i(t)}{n} \in [0, 1], i \in I_d$ — доля инфицированных ВПО i-го типа хостов в домене;
- $vul_i(t) = \frac{n*(1-SV_i(t))-IV_i(t)}{n} \in [0, 1], i \in I_d$ — доля уязвимых для ВПО i-го типа хостов в домене;
- $\beta_i(ED(tr_i, ES_i, t), IV_I(t - \Delta t)) \in \mathbb{N}, i \in I_d$ — скорость распространения ВПО внутри домена (средняя скорость распространения ВПО). Функция, определенная на множестве новых экземпляров ВПО и уже существующих внутри домена. $ED(tr_i, ES_i, t)\ inN, i \in I_d$ — функция, ставящая в соответствие входящему в домен вредоносному трафику (tr_i) и размеру экземпляра ВПО (ES_i) количество экземпляров ВПО каждого типа, проникших в домен за дискрет времени t.

Зная количество экземпляров ВПО, попавших в домен в заданный промежуток времени, и коэффициент защищенности домена SV_i от данного типа ВПО, мы можем вычислить количество ВПО, прошедшего через защиту домена (t_0 – начальный момент времени), следующим способом:

$$IV_i(t) = \frac{vul_i(t) * n}{1 + (\frac{1}{IV_i(t_0)} - 1) * e^{-\beta_i * t}}$$

Стоит отметить, что использование простых эпидемических моделей не является обязательным, и выбор модели распространения ВПО зависит от известной информации о распространении конкретного экземпляра ВПО. Наиболее точным будет моделирование, которое основывается:

- на алгоритме распространения исследуемого ВПО;
- данных о его размерах;
- характеристиках сетевого приложения, при помощи которого осуществляется распространение.

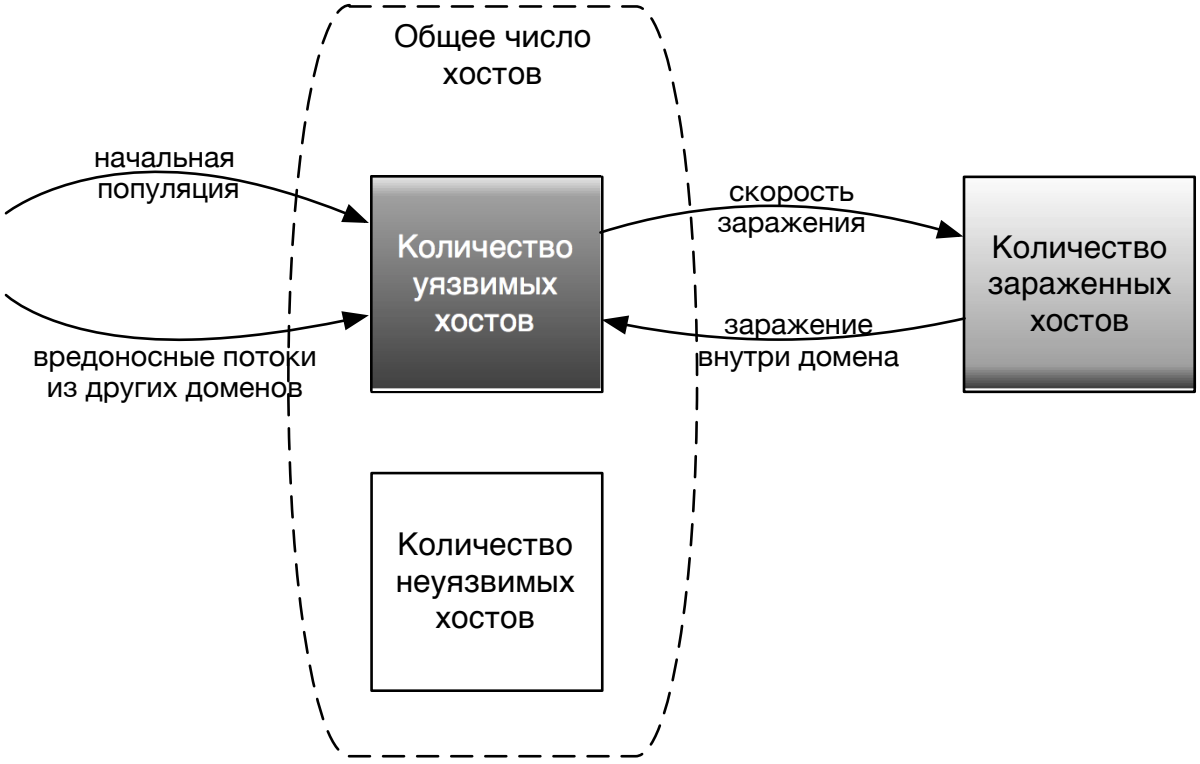

Рисунок 2.3: Модель заражения домена.

2.2.3 Описание сетевой активности

Компонент состояния домена Res определим как величину, характеризующую загрузку ресурсов домена в определенный момент времени. Рассмотрим домен как многоканальную систему массового обслуживания (СМО) с ожиданием и с ограниченной длиной очереди. Следовательно, для обработки трафика все потоки по всем истокам домена суммируются, формируя очередь заявок на каждый сток домена $Res_{d_n} = \{q(t)_{d_{n_1}}, q(t)_{d_{n_2}}, ..., q(t)_{d_{n_k}}\}$, где [81]

- $q(t)_{d_n} = (\lambda(t), \mu, m)$ — СМО для каждого стока домена:
  - $\lambda(t) \in \mathbb{N}$ — интенсивность входящих данных по всем истокам домена за дискрет времени;

– $\mu \in \mathbb{N}, \mu = const$ — интенсивность обслуживания, то есть объем данных, который может обработать домен за дискрет времени;

– $m \in \mathbb{N}, m = const$ — размер очереди, то есть размер буфера для хранения входящих данных, ожидающих обработки;

- k — число стоков домена d_n.

Заявки, для которых не оказалось места в очереди по причине ее заполненности, сбрасываются. Таким образом, поток входящих данных с интенсивностью $\lambda(t) > \mu$ увеличивает очередь заявок, $\lambda(t) < \mu$ — уменьшает очередь заявок, $\lambda(t) = \mu$ — не изменяет очередь заявок.

Под загруженностью домена $DL_{d_n} \in [0, m_{d_n}]$ будем понимать длину очереди: количество заявок, ожидающих обработку (см. рисунок 2.4).

Под задержкой транзитного трафика TD будем понимать время пребывания входящих данных в очереди $TD_{d_n} = \frac{DL_{d_n}}{\mu_{d_n}}$.

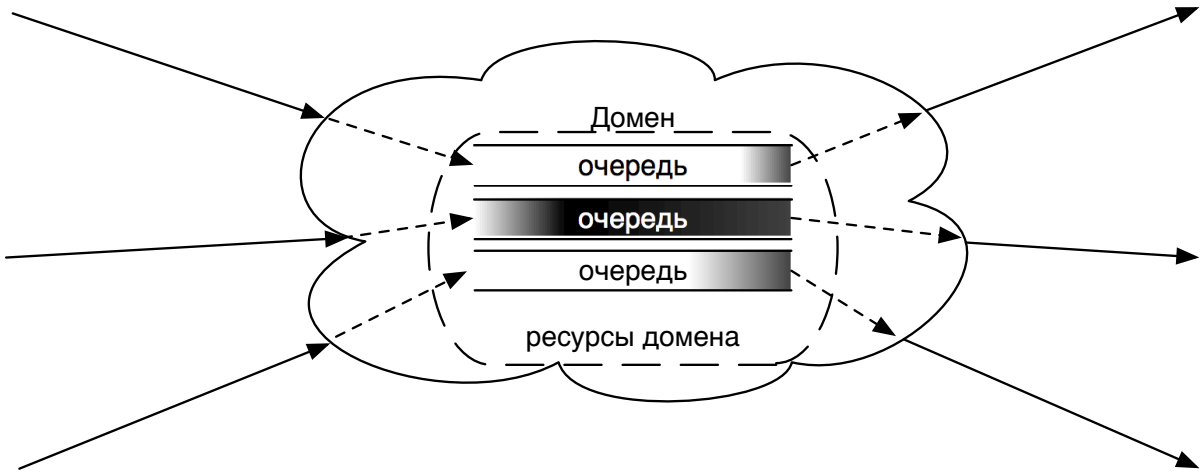

Рисунок 2.4: Загруженность ресурсов домена.

Обозначим множество всех потоков в ГКС через $F = \{f_1, f_2, \ldots, f_u\}$. Множество вредоносных потоков трафика будем обозначать $F_{inf}, F_{inf} \subseteq F$.

Потоком в сети назовем функцию $f_j(s_d, t), f_j \in F$ в вершинах мультиграфа G, зависящую от времени и состояния домена d, через который проходит поток. Данная функция принимает значения на множестве натуральных чисел со следующими свойствами:

1. для любого канала $l \in L$ выполняется $f_j(s_d, t) \leqslant TP_l, d \in D \cup P$, где канал l подключен к стоку домена d. Из этого следует, что интенсивность потока в канале никогда не превышает пропускную способность данного канала;

2. для любого домена $d_x \in D \cup P$, принадлежащего маршруту потока f_j, выполняется

$$(1) \sum_{out(d)} f_j(s_d, t) = \sum_{in(d)} f_j(s_d, t) - \sum_{d} f_j(s_d, t),$$

   где:

   - $s_d \in S_d$ — состояние домена $d \in D$;
   - $\sum_d f_j(s_d, t)$ — часть потока, потерянная по причине исчерпания ресурсов домена. Данное соотношение определяет, что из вершины необязательно выходит тот объем потока, который пришел в данную вершину, то есть $\sum_{out(d)} f_j(s_d, t) \leqslant \sum_{in(d)} f_j(s_d, t)$ в момент времени t;

3. поток обладает свойством аддитивности, то есть в момент времени t^*:

$$(2) \sum_{out(d')} f_j(s_{d'}, t^*) = \sum_{in(d')} f_j(s_{d'}, t^*) - \sum_{d'} f_j(s_{d'}, t^*)$$
$$\sum_{out(d'')} f_j(s_{d''}, t^*) = \sum_{in(d'')} f_j(s_{d''}, t^*) - \sum_{d''} f_j(s_{d''}, t^*)$$
$$\sum_{out(d''')} f_j(s_{d'''}, t^*) = \sum_{in(d''')} f_j(s_{d'''}, t^*) - \sum_{d'''} f_j(s_{d'''}, t^*). \quad (2.1)$$

Из этого свойства следует, что:

$$(3) \sum_{out(d''')} f_j(s_{d'''}, t^*) =$$
$$\sum_{in(d')} f_j(s_{d'}, t^*) - (\sum_{d'} f_j(s_{d'}, t^*) + \sum_{d''} f_j(s_{d''}, t^*) + \sum_{d'''} f_j(s_{d'''}, t^*)) \quad (2.2)$$

$$(d^{'}, d^{''}) \in L, (d^{''}, d^{'''}) \in L.$$

Назовем интенсивностью потока величину $M(f_j) = div_{f_j}(d_{src})$, где $div_{f_j}(d_{src})$ — дивергенция потока [82], обладающая свойствами (1) – (3), $d_{src} \in D$ — домен-источник потока. Тогда для мультиграфа G справедливо равенство:

$$div_{f_j}(d) = \sum_{out(d)} f_j(s_d, t) - \sum_{in(d)} f_j(s_d, t) + \sum_{d} f_j(s_d, t)$$

Отметим, что $div_{f_j}(d) = 0$ для всех внутренних вершин, которые не изменяют поток. Следовательно, являются транзитными, не перегруженными и не осуществляющими фильтрацию потока. Дивергенция $div_{f_j}(d_{dst})$, где $d_{dst} \in D$ — домен-потребитель потока определяет функцию поглощения потока. Данная функция определяет потребление потока доменом d, если этот домен является конечной точкой в маршруте потока.

На интенсивность потоков, проходящих через домен, существенно влияет порядок их обработки в домене. Известно [83] несколько стратегий обработки:

1. FIFO стратегия. Очередность обработки потока соответствует очередности поступления данного потока в домен. Однако, в описываемой модели потоки приходят в домен одновременно в каждый момент времени, следовательно, применение данной стратегии в чистом виде не представляется возможным;

2. приоритетная стратегия. Очередность обработки потока напрямую зависит от приоритета потока (чем выше приоритет, тем быстрее обработается поток доменом);

3. справедливая стратегия. Очередность обработки потоков не важна, так как все ресурсы домена в равной степени разделяются между потоками (на каждый поток выделяется $\frac{Res}{num}$, где Res ресурсы домена, а num - число потоков);

4. стратегия, основанная на интенсивности потока. Очередность обработки потока зависит от интенсивности потока (чем более интенсивный поток поступает в домен, тем менее высокий приоритет у него на обработку);

5. смешанная стратегия. Очередность обработки зависит от интенсивности, но в отличие от стратегии, описанной в пункте 4, зависимость обратная (чем более интенсивный поток поступает в домен, тем более высокий приоритет у него на обработку).

Выбор стратегии основывается на целях эксперимента и на количестве предварительно известной информации о потоках. Для работы с низкоскоростными или нагруженными высокоскоростными каналами необходимо использовать стратегию, основанную на интенсивности потока; для работы с не перегруженными каналами (загрузка каналов не превышает примерно 90% [83]) – справедливую стратегию; для обеспечения наименьших задержек и потерь для конкретных потоков – приоритетную стратегию. В случае отсутствия детализированной информации о потоках, например,

известно только количество источников и протокол потока, целесообразно использовать смешанную стратегию.

Для описания прохождения потока через домены каждому потоку $f_j \in F$ сопоставим набор тэгов потока $FTS = \{tag_1, tag_2, \ldots, tag_r\}$. Каждый тэг используется для определения политики обработки данного потока трафика при прохождении через домен. Наличие у потока определенного тэга определяет принадлежность данного потока соответствующему классу. В один класс можно объединить, например, все потоки, потребителем которых является данный домен, либо потоки принадлежащие ВПО определенного типа.

Для описания процесса изменения каждого потока из F, проходящего через домен, будем представлять домен в виде автомата следующего вида:

$$DS = (S_d, a, \delta, q_0, Fin),$$

где:

- S_d — множество состояний домена, которое одновременно будет являться множеством состояний этого автомата;
- q_0 — начальное состояние автомата;
- Fin — заключительное состояние автомата. В данной модели любые состояния, кроме начального, могут быть заключительными;
- a — допустимый входной алфавит (множество тэгов);
- $\delta = a \times S_d \rightarrow < action >< out >< S_d^* >$ — функция переходов, аргументами которой являются текущее состояние, набор тэгов входящего потока и загруженность ресурсов текущего домена, а значением – новое состояние:
  - $action$ — одно или несколько из нижеперечисленных действий, ассоциированных с данным переходом:
    - добавление нового тэга;
    - изменение тэга;
    - удаление тэга;
    - определение стока домена, на который нужно перенаправить входящий поток;

* генерация потока;
* поглощение потока;
* сброс потока;

– S_d^* — новое состояние домена;

– $out \in out(d)$ — сток домена, на который необходимо направить поток.

2.2.4 Доказательство корректности формальной модели ГКС

Сформулируем ранее упомянутую задачу анализа динамики заражения хостов и доли вредоносного трафика в ГКС в терминах построенной модели, а затем докажем, что эта задача имеет решение. Пусть дана сеть G с источниками d_{src} и получателями d_{dst} потоков из множества F, где $d_{src}, d_{dst} \in P$. Каждому потоку приписана функция начальной интенсивности $div_f(d_{src})$. Каждому каналу $l \in L$ приписана пропускная способность TP_l. Каждой вершине $d \in D \cup P$ приписана величина Res_d, обозначающая ресурсы, и конечный автомат для изменения потоков DS_d. Необходимо для каждого дискрета t найти решение двух задач:

1. оценить число зараженных хостов в доменах ГКС:

$$\sum_{i \in I} \sum_{d \in D \cup P} IV_d(t),$$

где I — множество типов ВПО;

2. оценить долю трафика, порождаемого ВПО, в общем трафике ГКС ($f_* \in F_{inf}$):

$$\sum_{f_* \in F_{inf}, f \in F} \sum_{d \in D \cup P} \frac{div_{f_*}(v)}{div_f(v)}.$$

Сформулируем и докажем теорему, обосновывающую корректность этих задач.

Теорема 1. Пусть дана сеть G с источниками d_{src} и получателями d_{dst} потоков из множества F, где $d_{src}, d_{dst} \in P$, тогда если в сети G выполнены следующие условия:

- каждому потоку приписана функция начальной интенсивности $div_f(d_{src})$;
- каждому каналу $l \in L$ приписана пропускная способность TP_l;
- каждой вершине $d \in D \cup P$ приписана величина Res_d, обозначающая ресурсы и конечный автомат для изменения потоков DS_d;

то существует алгоритм воспроизведения динамики заражения хостов и доли вредоносного трафика в ГКС, представляющее оценку числа зараженных хостов в каждой вершине d сети G и доли вредоносных потоков в общем трафике сети G.

Доказательство. Поскольку:

1. в $|L_p|$-мерном пространстве переменных f ограничение $0 \leqslant f(s_d, t) \leqslant TP_l$ определяет ограниченную замкнутую область $l \in L_p$, где L_p — множество каналов, через которые проходит поток $f, L_p \subseteq L$;

2. в $|D \cup P|$-мерном пространстве переменных f ограничение $0 \leqslant f(s_d, t) \leqslant Res_d$ определяет ограниченную замкнутую область;

3. действия работы с тэгами (добавление, изменение, удаление) не влияют на изменение интенсивности потока;

4. маршрутизация потока определят сток, связанный с каналом, по которому пройдет поток, но никак не влияет на интенсивность потока;

5. поглощение и сбор потока только уменьшают интенсивность, играя роль потребителя;

6. учитывая, что при генерации потока, связанного с работой автомата в домене, создается новый поток, следовательно, невозможно увеличение интенсивности в транзитных точках маршрута;

   Примечание: важно для выполнения ограничений, чтобы на пути потока не было источников потока (начальной нагрузки), кроме как в точке начала потока;

7. из пунктов 3-6 следует, что все возможные действия автомата не выводят за рамки ограничений, описанных в пунктах 1-2 данного доказательства;

8. для каждого потока начальная интенсивность $div_f(d_{src})$ — непрерывный линейный функционал на множестве дискретных моментов времени модельного времени, так как:

   (a) пусть есть множество дискретов времени $Time = \{t_0, t_1, \ldots, t_{end}\}$. Введем на данном множестве дискретную метрику $\sigma(t_x, t_y)$ такую, что $\sigma(t_x, t_y) = 0$, если $t_x = t_y$ и $\sigma(t_x, t_y) = 1$ во всех остальных случаях;

(b) в силу теоремы о свойстве системы открытых множеств метрического пространства [84] семейство подмножеств множества $Time$ удовлетворяет аксиомам топологического пространства. Таким образом, данное метрическое пространство является топологическим;

(c) дискретное топологическое пространство является частным случаем сепарабельного пространства, поскольку топологическое пространство называется сепарабельным, если оно содержит не более чем счетное число подмножеств, замыкание которых совпадает с данным пространством. В нашем случае в силу конечности пространства $Time$ его можно разделить на подмножества конечным количеством вариантов;

(d) на данном сепарабельном пространстве линейный функционал $div_f(d_{src})$ можно считать непрерывным, так как на сепарабельном пространстве любой линейный функционал является непрерывным [85] ;

Из сказанного выше следует, что $div_f(d_{dst})$ достигает своего максимального и минимального значения. Следовательно, возможно оценить число зараженных хостов в каждой вершине d сети G и долю вредоносного трафика в сети G. ■

2.3 Описание службы управления временем в модели ГКС

В начале главы 2 было сделано предположение о наличие глобальных часов в формальной модели ГКС. Данное предположение было удобно для введения основных определений и доказательства корректности. Однако, как было сказано ранее, данное упрощение является серьезным ограничением на практике. Это связанно с большой размерностью сети ГКС и невозможностью выполнять модель на единственном вычислителе.

В случае использования нескольких вычислителей необходимо сделать предположение о наличии нескольких часов, которые необходимо синхронизировать при взаимодействии соответствующих компонент модели ГКС. В этом случае модель ГКС представляется распределенной системой, где каждый домен обладает собственными часами — t_D, где $t \in \mathbb{N}$.

В случае распределённой системы классическим вариантом является использование оптимистических, консервативных алгоритмов или их комбинации [86]. Однако при

решении задачи моделирования ГКС данные алгоритмы не применимы, так как данные алгоритмы ведут:

- либо к очень маленькому доверительному интервалу в консервативном случае;
- либо к частым откатам в оптимистическом случае. В данном случае так же трудоемким является вопрос разработки и реализации системы откатов.

Исходя из вышесказанного, предлагается особый вариант синхронизации времени, который условно можно назвать «упрощенно оптимистическим» алгоритмом синхронизации. Отличие данного алгоритма от оптимистического алгоритма заключается в отсутствии механизма отката при получении «события из прошлого», вместо этого трафик, нарушивший синхронизацию, будет сброшен.

В контексте имитационного моделирования ГКС нетривиальным становится вопрос определения причин временных рассинхронизаций. В случае использования Hi-Fi подхода единственной причиной рассинхронизации является несоответствие модельных компонент характеристикам физического оборудования, на котором вычисляется модель. Например, если модельный канал не соответствует по временным характеристикам физическому каналу, использование вышеописанного «упрощенно оптимистического» алгоритма синхронизации времени приведет к постоянному сбросу всего трафика, проходящего по этому каналу.

Для реализации синхронизации времени между компонентами модели предлагается уточнить модель времени, добавив агента времени на каждый домен. Напомним, что доменом является пятерка (n, SV, IV, Res, FPS). Уточнение понятия домена заключается в добавлении компонента TA (time agent). Использование TA предполагает следующий алгоритм синхронизаций времени между компонентами модели:

1. на этапе инициализации будет определено максимальное отклонение реальных задержек трафика от предполагаемых модельных задержек;
2. если выявлены недопустимые отклонения, то об этом факте оповещается пользователь;
3. максимальное отклонение модельного времени от физического определяется отношением $\Delta = \frac{t_m}{t_{ph}}$, далее все временные характеристики всех каналов в топологии модельной сети умножаются на заданную Δ . Также необходимо

изменить временные характеристики в функционирующих сетевых приложениях (например, всевозможные таймауты).

Предложенный подход гарантирует, что к моменту передачи трафика домены будут синхронизированы между собой по времени, и сетевые пакеты в модели сети будут приходить в том же порядке, как и в реальной сети. Достигается это по сути замедлением работы модели сети по наиболее медленному физическому каналу, который используется для развертывания модели в кластере.

Глава 3

Система имитационного моделирования Network Prototype Simulator

В данной главе описывается архитектура системы имитационного моделирования (СИМ) Network Prototype Simulator (NPS) [87].

Для реализации имитационной модели ГКС (в рамках Hi-Fi подхода, описанного в разделе 1.2.2) будем использовать следующие предположения (см. рисунок 3.1):

- каналы моделируются при помощи пар виртуальных сетевых интерфейсов, работающих на пакетном уровне;
- сетевые устройства представляют собой программируемые виртуальные коммутаторы. Использование подобного подхода позволяет моделировать процесс функционирования различных сетевых устройств (машрутизаторов, коммутаторов) путем программной конфигурации сетевых устройств;
- хосты (см. рисунок 3.2) представляют полноценную систему, которая в рамках своей файловой системы предоставляет возможность работы сетевым приложениями. Под «системой» может пониматься как полноценная виртуальная машина со своей операционной системой, так и процесс в операционной системе. Второй вариант предпочтителен, когда хосты не представляют из себя «тяжеловесный» набор сетевых приложений, и есть необходимость моделировать работу нескольких хостов в рамках одной операционной системы. В общем случае не важно, моделируется ли хост на физической или на виртуальной машине.

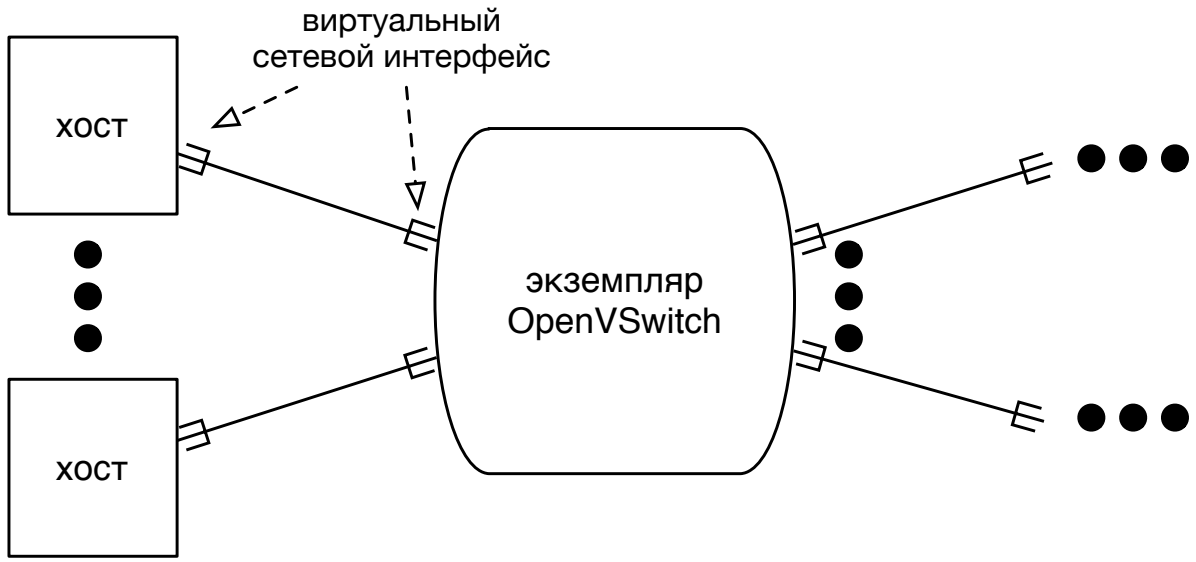

Рисунок 3.1: Реализация имитационной модели ГКС.

3.1 Требования СИМ NPS

СИМ NPS должна удовлетворять следующим требованиям:

- реализовывать формальную модель ГКС, описанную в главе 2;
- обладать свойством масштабируемости (см. 3.2);
- избегать процесса калибровки модели сети, а также необходимости доказательства корректности модели;
- позволять изменять количество узлов в моделируемой сети без существенных затрат времени на перестройку модели сети.

Целью разработки СИМ NPS является реализация формальной модели ГКС, описанной в главе 2.

3.2 NPS кластер

Под NPS кластером будем понимать несколько вычислительных машин, объединенных между собой в сеть. В этом случае не имеет значения использование виртуальных или физических вычислительных машин. Далее вычислительную машину, входящую в состав NPS кластера, будем называть NPS узлом кластера (см. рисунок 3.3).

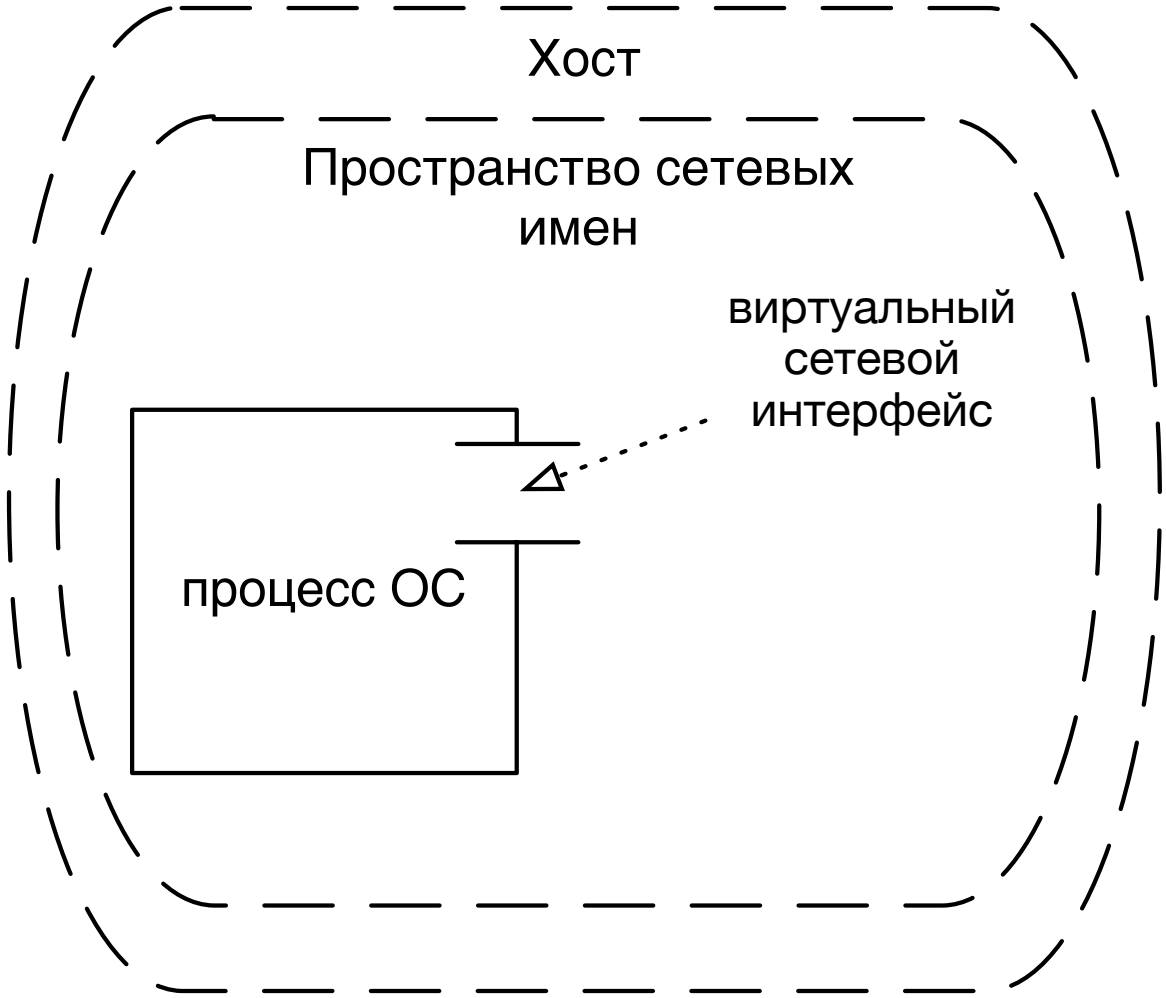

Рисунок 3.2: Реализация хоста.

Требование масштабируемости удовлетворяется СИМ NPS за счет увеличения максимального количества узлов в модели сети при добавление новых NPS узлов кластера. СИМ NPS сама решает, как будет разбита топология модели сети на NPS узлах кластера. Это свойство существенно упрощает перестройку модели при изменении количества моделируемых узлов в сети, так как от пользователя требуется только предоставление вычислительных ресурсов (в виде NPS узлов кластера).

Возможный размер моделируемой NPS сети зависит от количества NPS узлов в кластере. Беря во внимание, что один NPS узел кластера может моделировать топологию размером до 2000 узлов (хостов или сетевых устройств), а современный сервер может поддерживать работу порядка 25 NPS узлов кластера, итого получаем порядка 50000 хостов, моделируемых одним сервером. Учитывая, что разработанная

СИМ NPS обладает свойством масштабируемости, мы получаем возможность моделировать сети масштаба ГКС (порядка сотни тысяч узлов, см. рисунок 3.14).

3.3 Описание архитектуры СИМ NPS

СИМ NPS состоит из четырех основных частей:

- NPS главная консоль управления;
- NPS узел кластера;
- NPS ПКС контролер;
- NPS библиотека сетевых приложений.

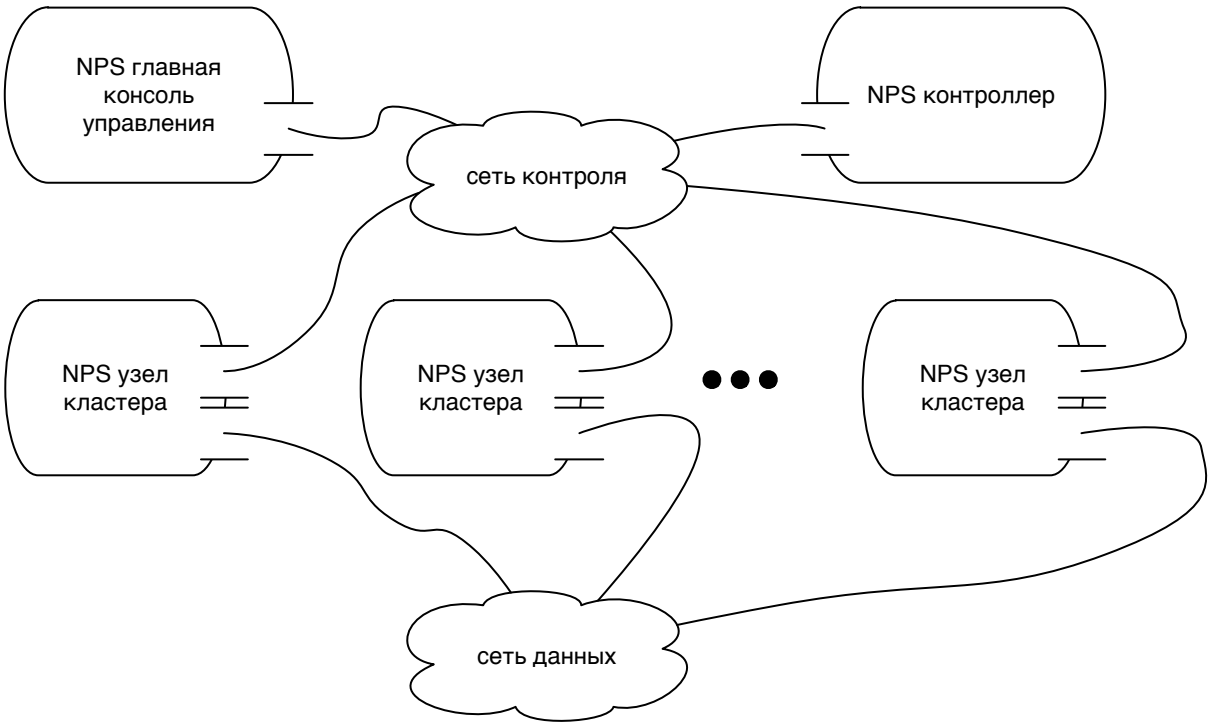

Рисунок 3.3: Общая схема NPS кластера.

NPS главная управляющая консоль — это аналог консоли управления системы Mininet, но способная конфигурировать и выводить результаты выполнения команд с нескольких NPS узлов кластера, используя SSH соединение. Для установи SSH соединения между главной управляющей консолью и каждым экземпляром Mininet в кластере используется библиотека Paramiko [88].

Основные задачи NPS главной управляющей консоли:

- разбор текстового описания графа сети с заданными сетевыми приложениями;
- разделения графа сети на несколько частей. Количество частей равно количеству узлов кластера включенных в состав NPS;
- генерация на основе разделения графа скриптов инициализации для развертывания модели сети на разных узлах кластера;
- запуск с заданными параметрами сетевых приложений на узлах кластера, исходя из заданной конфигурации;
- именование каждого созданного процесса хоста, коммутатора с целью сохранить уникальность имен в пределах всего кластера, на котором разворачивается модель сети;
- настройка сетевых интерфейсов, ассоциированных с каждым процессом хоста или коммутатора;
- настройка внешнего контролера для каждого процесса коммутатора;
- настройка параметров для логгирования необходимых данных при моделировании;
- отправка служебных скриптов на узлы кластера, необходимых для развертывания и функционирования модели сети;
- отправка команд процесса хостов и/или коммутаторов на уже развернутую модель сети;
- заверение всех процессов, созданных в ходе моделирования, удаление всех созданных экземпляров контейнеров виртуализации.

Главная консоль предоставляет интерфейс для GUI для задания начальной конфигурации сети и данные для отображения результатов моделирования.

NPS узел кластера — это виртуальная или физическая машина с установленной ОС Linux (в данной работе использовался дистрибутив Ubuntu 12.10) и с набором установленных пакетов:

1. Python2.7 [89]. Интерпретатор языка Python;

2. Scapy-Python [90]. Библиотека на языке Python для генерации специально сконструированных сетевых пакетов;

3. Mininet [70]. Экземпляр системы Mininet. Система Mininet, устанавливаемая на узлах кластера, дополняется расширением базовых классов для обеспечения функционирования сетевых приложений. Данное расширение представляет скриптовый файл на языке Python. Этот файл передается на узел кластера NPS главной консолью управления при инициализации развертывания модели сети.

Каждый NPS узел кластера (см. рисунок 3.4) должен иметь два активных Ethernet интерфейса. Первый используется для обмена управляющей информацией между NPS узлом кластера и главной управляющей NPS консолью, второй — для обмена данными между соседними NPS узлами кластера. Последним требованием для NPS узла кластера является наличие пользовательского аккаунта, созданного специально для главной управляющей NPS консоли с правами администратора "sudo".

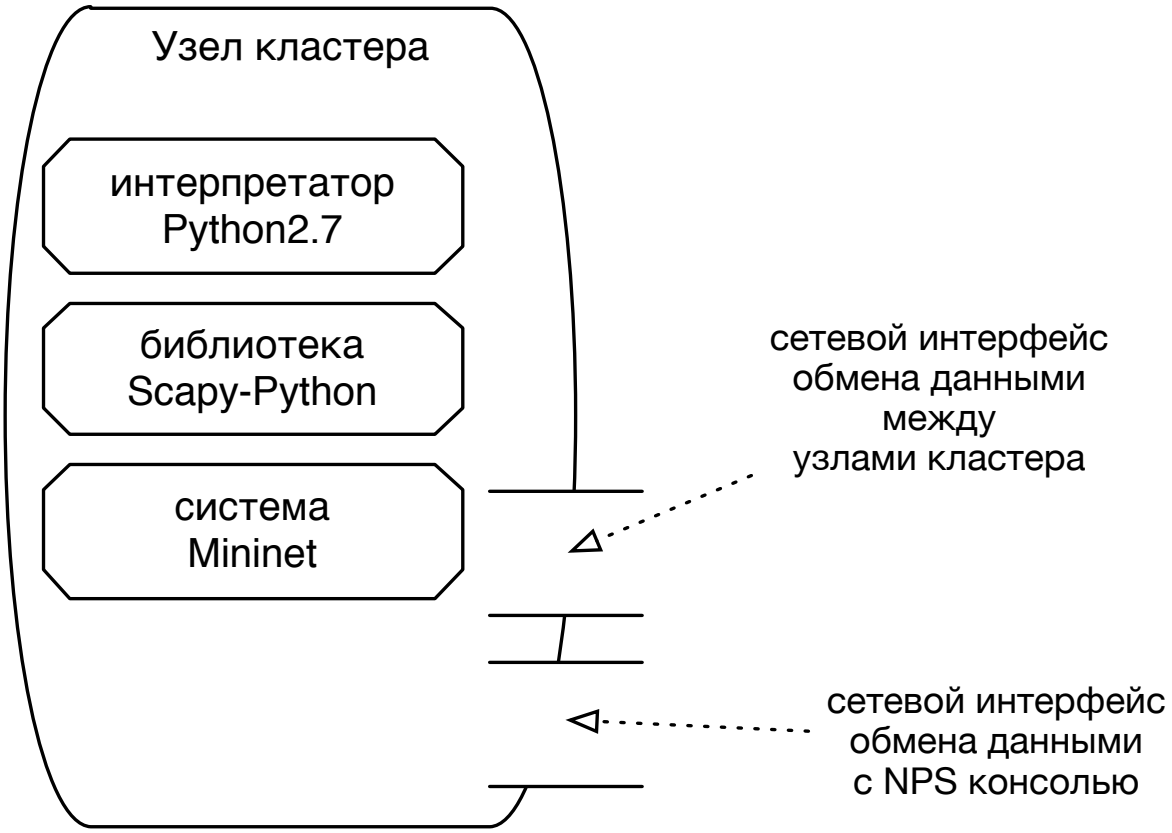

Рисунок 3.4: Узел кластера.

NPS ПКС контролер используется для программирования сетевой активности внутри сегментов Mininet сетей и между данными сегментами. В NPS кластере

предполагается использование множества NPS контролеров. В СИМ имеется возможность каждый узел кластера ассоциировать со своим контролером.

Использование нескольких контролеров в ходе конкретного эксперимента имеет свои достоинства и недостатки. В случае единственного контролера имеется возможность сконцентрировать в одной точке глобальное видение сети, что позволяет разыгрывать различные сценарии сетевого обмена между хостами, например, моделировать сетевую эпидемию червя. Однако данный подход не является масштабируемым, и при достаточно большом размере модели сети контролер становится "бутылочным горлышком". В то же время использование нескольких контролеров решает проблемы масштабируемости, но в некоторых экспериментах может потребоваться синхронизация состояний контролера, что повлечет за собой дополнительные расходы по ресурсам кластера.

Для задания сетевой активности и процесса генерации нагрузки на ресурсы моделируемой сети на каждом хосте имеется возможность запускать сетевые приложения. Для удобного запуска набора приложений на большом числе хостов используется библиотека приложений. Данная библиотека содержит интерфейсы для запуска некоторых сетевых приложений. На данный момент реализована поддержка DHCP [91], NTP [92], DNS [93], HTTP [94] приложений. Архитектура библиотеки сетевых приложений дает возможность пользователю добавлять собственные приложения.

Файловая система, пространство идентификаторов процессов и прочие глобальные ресурсы ОС разделяются между хостами. Это приводит к некоторым существенным ограничениям на конфигурацию сетевых приложений. Поскольку моделируемая сеть рассматривается и используется как единый объект, то ее хосты должны пройти инициализацию синхронизированным образом. Синхронизация позволит избежать таких некорректных ситуаций, как запуск каких-либо демонов серверных приложений после клиентских или начала исполнения тестовых сценариев на части хостов до окончания развертывания всех остальных.

Особое внимание в NPS библиотеке сетевых приложений (см. рисунок 3.5) уделяется аргументам командной строки, которые могут затребовать одни и те же ресурсы в единой файловой системе, едином пространстве UNIX-сокетов, едином пространстве идентификаторов процессов. Например, некоторые сетевые приложения для проверки факта своего запуска и недопущения случайного повторного запуска проверяют определенным образом наличие именованного файла со своим PID в фиксированной

директории файловой системы. Как правило, имя такого файла можно задать через аргументы командной строки, что позволяет добиться единовременного запуска демона в различных экземплярах.

Стоит отметить, что по возможности необходимо использовать сетевые приложения, которые прописаны в NPS библиотеке сетевых приложений, так как в данных приложениях уже прописаны временные параметры каждого приложения. Указание данных параметров для NPS позволяет избежать некорректностей при использовании системы управления временем, реализованной в СИМ NPS.

Использование NPS библиотеки сетевых приложений на примере DHCP приводиться в эксперименте, описанном в разделе 4.4.1.

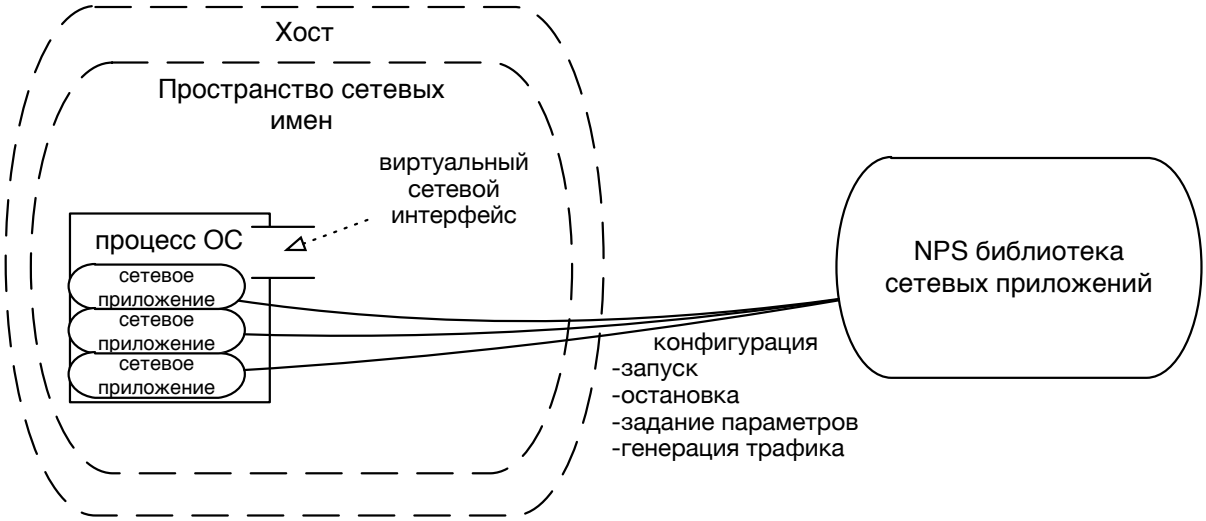

Рисунок 3.5: Работа с NPS библиотекой сетевых приложений.

3.4 Описание аппарата легковесной виртуализации ОС Linux

LXC (англ. Linux Containers) [75] — система виртуализации на уровне операционной системы для запуска нескольких изолированных экземпляров ОС Linux на одном компьютере. LXC не использует виртуальные машины, а создает виртуальное окружение с собственным пространством процессов и сетевым стеком. Все экземпляры LXC используют в разделяемом режиме ту же операционную систему, что и основная система (см. рисунок 3.6).

Контейнеры представляют из себя облегченную технологию виртуализации. Предоставляя средства для создания и использования контейнеров, ОС дает

приложениям возможность работать как бы на отдельной машине, при этом совместно используя множество базовых ресурсов. Экономия от совместного использования ресурсов в сочетании с предоставляемой изоляцией приводит к тому, что контейнеры требуют значительно меньших накладных расходов, чем истинная виртуализация.

Для изоляции процессов, сетевого стека ipc, uts и точек монтирования используется механизм пространств имён (namespaces). Для ограничения ресурсов применяются cgroups. По своим возможностям контейнеры занимают нишу между изоляцией при помощи chroot и полноценными средствами виртуализации. В состав инструментария LXC входит библиотека liblxc, набор утилит (lxc-create, lxc-start, lxc-stop, lxc-ls и т.п.), шаблоны для построения контейнеров и набор адаптеров для различных языков программирования.

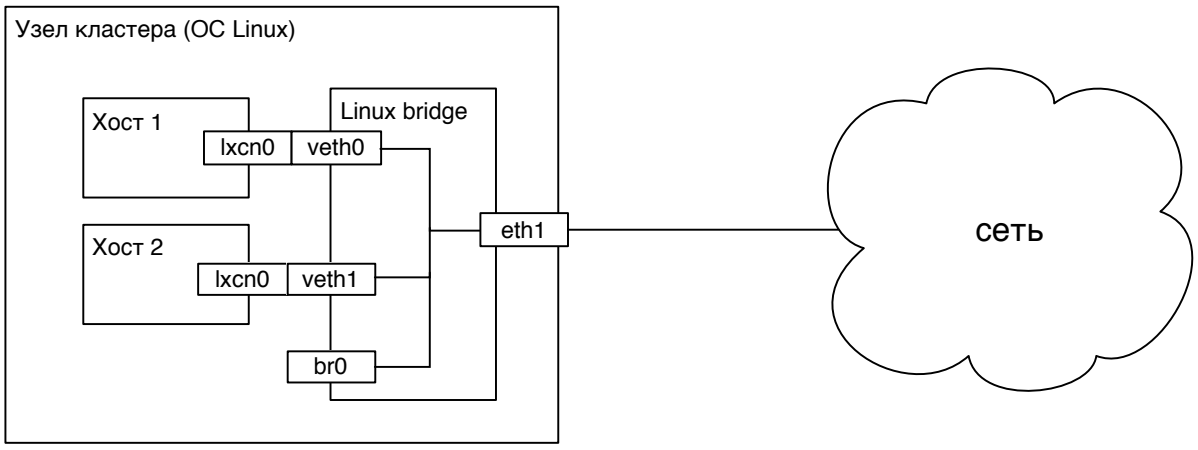

Рисунок 3.6: Контейнеры легковесной виртуализации в ОС Linux.

Контейнеры виртуализации используются в СИМ NPS для описания структуры и ресурсов моделируемой сети.

3.4.1 Методы управления модельным временем в СИМ

В процессе разработки СИМ необходимо соотносить между собой два представления времени:

- реальное время, в котором происходит функционирование моделируемой системы;
- модельное время, в масштабе которого организуется работа модели.

Целью методов управления модельным временем является установление соответствия между законами изменения времени в моделируемой системе и законами

времени, которые действуют в СИМ. С помощью методов управления модельным временем решаются следующие задачи:

- отображается переход моделируемой системы из одного состояния в другое;
- производится синхронизация работы модели.

Выбор метода управления модельным временем зависит от назначения модели, ее сложности, характера исследуемых процессов, требуемой точности результатов.

Существуют три метода управления модельным временем:

- дискретный;
- дискретно-событийный;
- непрерывный.

В случае дискретного метода продвижение времени происходит с минимально возможной постоянной длиной шага (дискрета времени) [86]. Дискретный метод применяется в случаях:

- если закон изменения времени описывается интегро-дифференциальными уравнениями. Характерный пример: решение интегро-дифференциальных уравнений численным методом. В подобных методах шаг моделирования равен шагу интегрирования. Динамика модели является дискретным приближением реальных непрерывных процессов;
- когда события распределены равномерно и можно подобрать шаг изменения временной координаты;
- когда сложно предсказать появление определенных событий;
- когда событий очень много и они появляются группами [95].

Можно выделить частный случай дискретного метода, называемый методом управления в реальном времени. В этом случае некоторые компоненты имитационной модели представлены физическими устройствами, следовательно, скорость изменения времени физического устройства равна скорости изменения астрономического времени. При использовании метода управления в реальном времени значение часов модельного времени определяется некоторой неубывающей функцией от значений аппаратно или программно реализованных часов реального времени. Примером систем, использующих

данный метод, являются тренажеры, модельное время для которых определяется линейной функцией от реального времени [96].

В случае дискретно-событийного метода время меняется тогда, когда изменяется состояние системы. В дискретной-событийном методе длина шага временного сдвига максимально возможная. Модельное время с текущего момента изменяется до ближайшего момента наступления следующего события. Применение дискретно-событийного метода предпочтительнее в том случае, если частота наступления событий невелика. Тогда большая длина шага позволит ускорить ход модельного времени [97]. Этот метод более сложен, чем дискретный в реализации, однако более эффективен с точки зрения использования вычислительных ресурсов.

В случае непрерывного метода переменные имитационной модели изменяются непрерывно, состояние моделируемой системы меняется как непрерывная функция времени, и, как правило, это изменение описывается системами дифференциальных уравнений. Соответственно продвижение модельного времени зависит от численных методов решения дифференциальных уравнений [98].

Из перечисленных методов для разработки СИМ NPS был выбран дискретный метод. Первая и основная причина выбора — это особенности использования техник виртуализации для построения имитационных моделей сети, описанных в разделе 3.4. Вторая — это возможность работы в реальном времени, что в будущем позволит включать в состав моделируемой сети реальные компоненты. Последние будут полезны для исследователей, которым необходима тестовая площадка для уже готового решения (например, сетевого приложения, или протокола).

3.4.2 Проблемы синхронизации модельного времени в распределенной СИМ

В случае распределенной СИМ, помимо метода управления временем, необходимо предусматривать методы синхронизации времени между компонентами распределенной системы. Необходимость синхронизации обуславливается отсутствием глобальных часов в модели, исполняемой на нескольких вычислителях.

Существуют подходы к реализации имитационной модели, которые предполагают использование нескольких вычислителей. Подобный подход порождает две проблемы, без решения которых невозможно построить корректную имитационную модель ГКС:

1. проблема несоответствия временных характеристик физических и моделируемых каналов, например, пропускной способности, задержки распространения, несоответствия скоростных характеристик сетевых интерфейсов и т.п. При передаче сетевого пакета между узлами, расположенными на разных вычислительных машинах в распределенной системе, временные характеристики физического канала между вычислителями могут существенно различаться от характеристик виртуального канала. При построении модели возможно возникновение несоответствия временных характеристик (например, физический канал гораздо медленнее виртуального), и, следовательно, возникновение непредусмотренных задержек и потерь сетевых пакетов;

2. проблема несоответствия сетевых стеков физических и моделируемых каналов. Примером такого различия может служить использование физических InfiniBand-каналов [99] для организации виртуальных Ethernet-соединения между вычислителями.

Решением второй проблемы является использование механизма сетевых мостов и специализированных версий сетевого стека для устройства, обеспечивающих функционирование физического канала [100]. Первая же проблема заключается в необходимости синхронизировать взаимодействие между узлами по времени, следовательно, при разработке модели необходимо уделить отдельное внимание методам управления модельным временем.

Для решения вышеописанных проблем в СИМ NPS предложен свой метод синхронизации модельного времени, описанный в разделе 2.3. Эффективность данного метода подтверждается экспериментами, описанными в разделе 4.5.

3.5 Особенности кластерной архитектуры СИМ NPS

Соединение сегментов различных узлов кластера стало возможным путем использования специализированного механизма добавления реального сетевого интерфейса в виртуальный сегмент сети. Использование данного механизма описано в примере «example/hwintf.py», который входит в стандартный дистрибутив системы Mininet. Суть механизма заключается в отображении виртуального контейнера (виртуального сетевого интерфейса) на реальный физический интерфейс.

Алгоритм построение NPS кластера состоит из нескольких шагов:

1. установка SSH соединений с каждым NPS узлом кластера;
2. конфигурирование выделенного интерфейса при построении виртуального сегмента сети на каждом NPS узле кластера;
3. разделение стандартного скрипта инициализации системы с описанием структуры модели сети на части. Количество этих частей равно количеству NPS узлов кластера;
4. подготовка скрипта инициализации для соответствующих NPS узлов кластера;
5. отправка частей скрипта инициализации на соответствующие NPS узлы кластера для построения топологии каждой части сети на NPS узле кластера;
6. отправка вспомогательных скриптов для генерации и установлении факта получения специализированных сетевых пакетов. Данные скрипты будут в дальнейшем использоваться для моделирования процесса распространения ВПО;
7. после построения сетевой структуры NPS главная управляющая консоль начинает выполнение сценария, предусмотренного экспериментом;
8. после выполнения сценария главная управляющая консоль сворачивает все экземпляры системы на NSP узлах кластера и закрывает SSH соединения;
9. (Дополнительно) визуализация результатов.

3.6 Графический интерфейс СИМ NPS

Существенным моментом для пользователя любой СИМ является процесс описания входных данных для эксперимента. Немаловажно сколько времени у пользователя тратится на описание всех входных данных для эксперимента (описание топологии, ресурсов сети, списка функционирующих приложений, процесса маршрутизации трафика, начальной популяции ВПО и т.д.). Для СИМ NPS был разработан специализированный графический интерфейс пользователя, который позволяет существенно ускорить и упростить процесс описания модельного эксперимента.

Основные возможности графического интерфейса пользователя (ГИП) СИМ NPS:

- описание структуры сети в виде графа (рисунок 3.7):
  - описание графа модели сети «вручную»;

  - генерация случайного графа модели сети;
  - загрузка графа модели сети из текстового описания (рисунок 3.8);

- описание ресурсов сети путем изменения параметров хостов, каналов и транзитных узлов графа модели сети (рисунок 3.9);
- запуск ПКС контролера (при запуске эксперимента) с возможностью конфигурирования его параметров. Помимо стандартных параметров (IP адреса и порта контролера) имеется возможность задания набора приложений контролера, отвечающих за маршрутизацию в моделируемой сети. Такой подход позволяет быстро и гибко задавать роль каждого транзитного узла в модели сети, сделав его коммутатором, маршрутизатором, или другим сетевым устройством;
- задание списка сетевых приложений, ассоциированных с каждым хостом в модели сети. При этом имеется возможность управлять параметрами запуска сетевых приложений, а так же задавать профили фоновой активности (например, HTTP-приложениям отправлять случайные запросы на HTTP-сервера в сети, или SMTP-приложениям отправлять случайные письма между активными SMTP-клиентами запущенными в модели сети). Последнее позволяет задавать фоновую активность, что существенно увеличивает ценность результатов проводимого эксперимента;
- сохранение (и загрузка) эксперимента в специальный текстовый формат файла (формат *.nps).

Основой задачей ГИП СИМ NSP является визуализации результатов проведения эксперимента. При этом важна визуализация как модельных данных («выхода» модели), так и процесса работы СИМ NPS (процесса разбиения модели сети на узлы кластера, загрузка ресурсов узла кластера, статистическая информация и т.д.). Решая данную задачу, в ГИП были предусмотрены следующие возможности визуализации процесса развертывания модели сети на кластера и результатов проведения экспериментов:

- использование графических примитивов для отображения зависимостей и соотношений параметров модели как с модельным временем, так и между собой. Примерами графических примитивов являются (рисунок 3.10):
  - график зависимости;

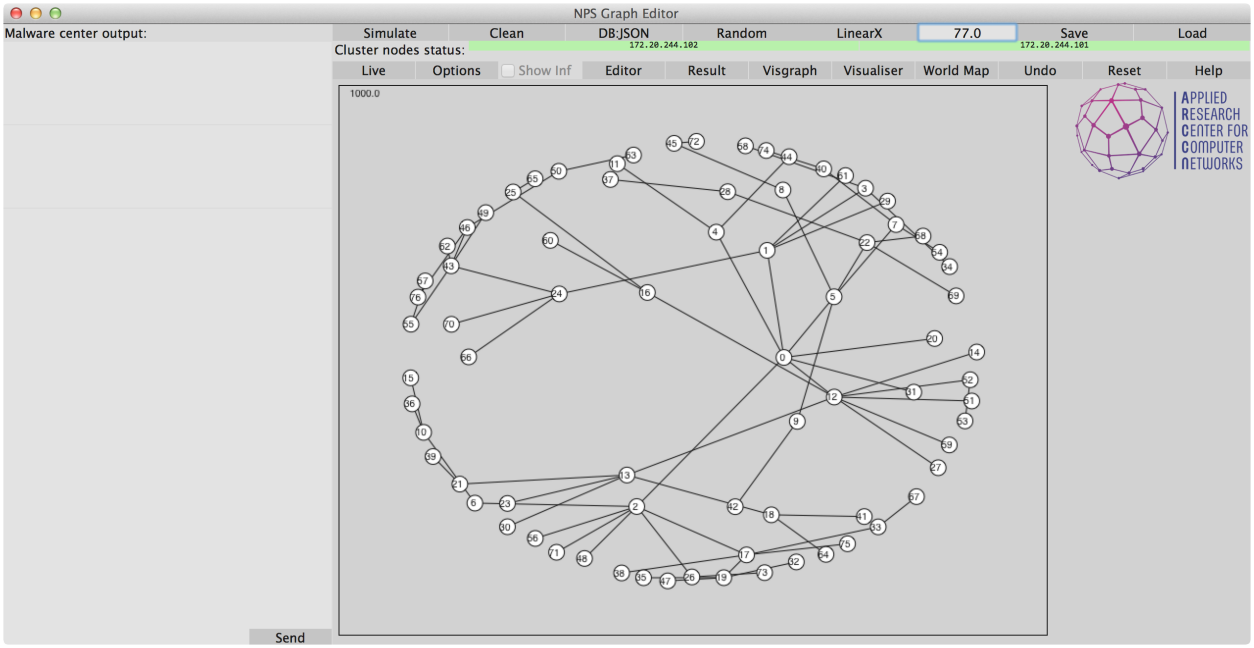

Рисунок 3.7: Процесс редактирования графа модели сети.

  - таблица;
  - гистограмма;
  - круговая диаграмма;

- отображение хостов на географической карте мира с использованием функциональности geoIP [101] (рисунок 3.11);
- отображение результатов разделения модели сети между NPS узлами кластера (рисунок 3.12);
- отображение процесса заражения хостов в моделируемой сети в режиме реального времени (рисунок 3.13).

3.7 Выводы

Описанная архитектура СИМ NPS удовлетворяет требованиям, описанным в разделе 3.1. Так масштабируемость достигается за счет добавление новых NPS узлов кластера при подготовки эксперимента. Отсутствует необходимость процесса калибровки модели сети за счет использования аппарата легковесной виртуализации ОС Linux, описанного в разделе 3.4, а также NPS библиотеки сетевых приложений. Возможность системы автоматического разбиения топологии модели сети на NPS узлы

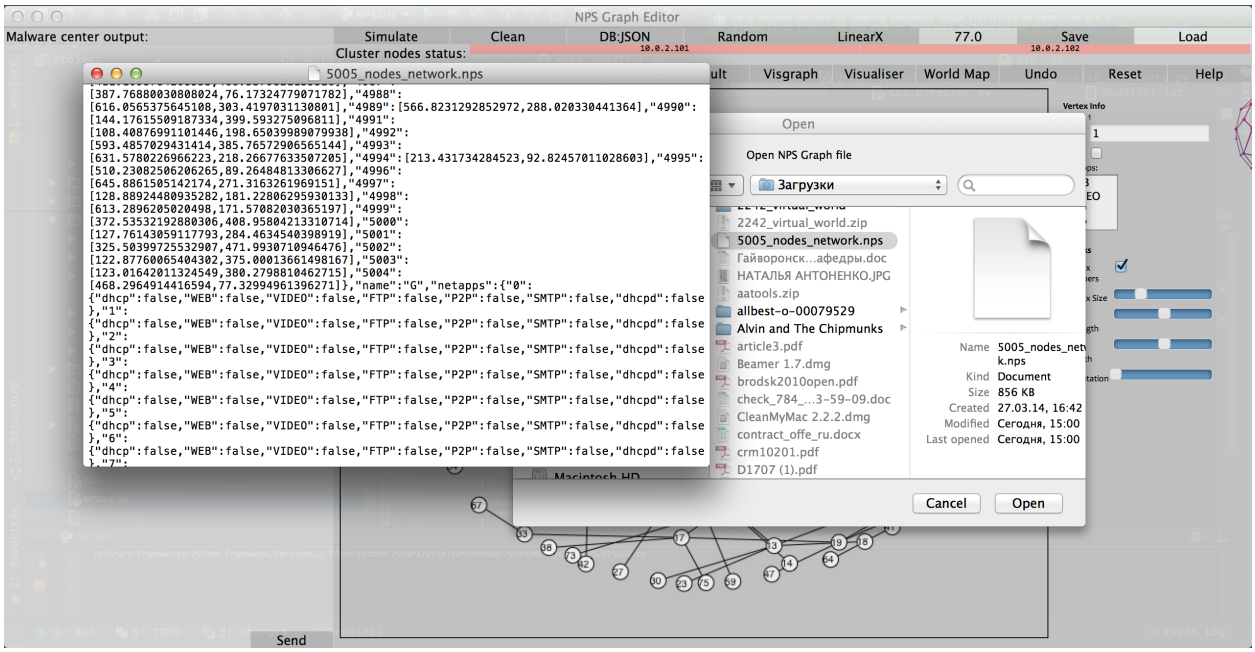

Рисунок 3.8: Запись эксперимента в *.nps файл.

кластера позволяет без особых временных затрат изменять количество узлов в модели сети.

Адекватность модели ГКС, реализуемой СИМ NPS, доказана в экспериментальном исследовании, представленном в главе 4.

Подробное описание реализации и процесса установки СИМ NPS: примеры конфигурации, скриншоты, схемы развертывания, настройка сетевых стеков физических машин приведены на странице проекта [102].

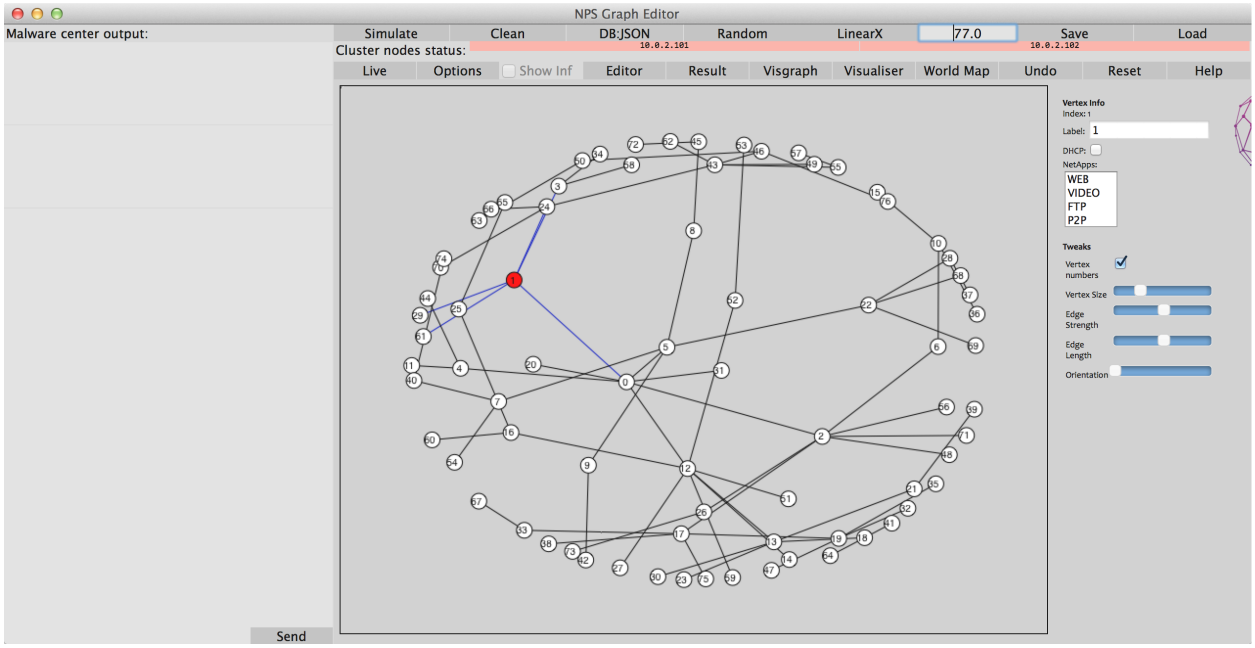

Рисунок 3.9: Изменение параметров хостов модели сети.

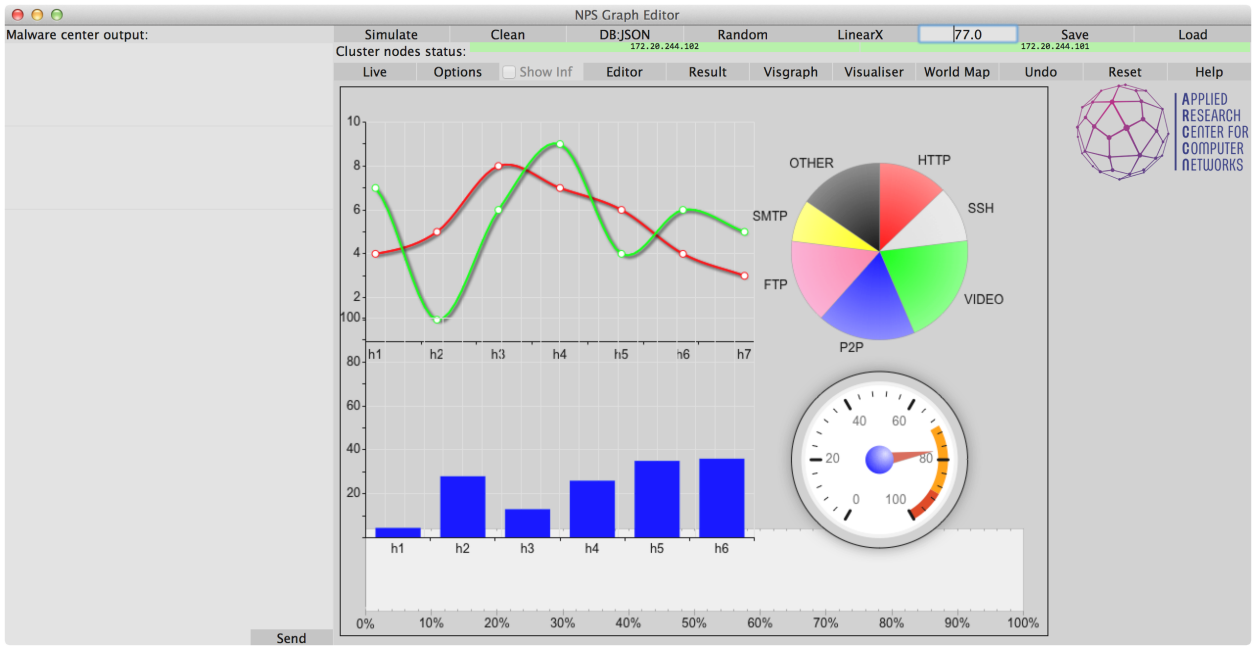

Рисунок 3.10: Визуализация результатов моделирования (графики, гистограммы и т.д.).

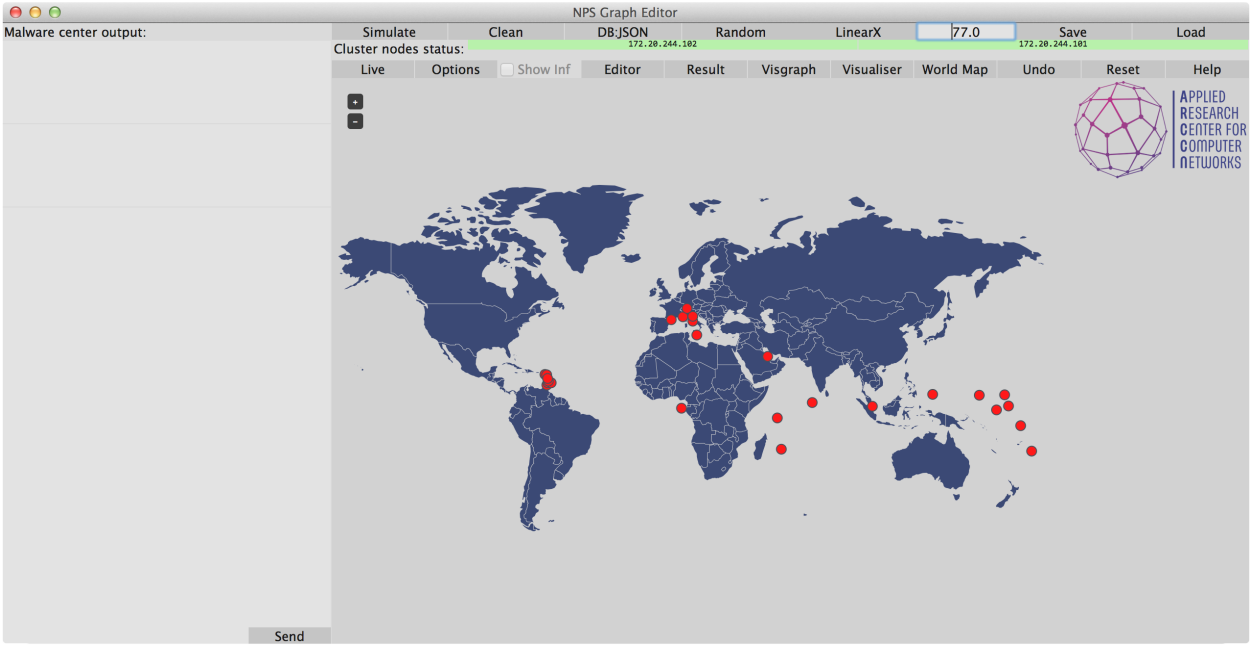

Рисунок 3.11: Визуализация результатов моделирования (карта Мира, geoIP).

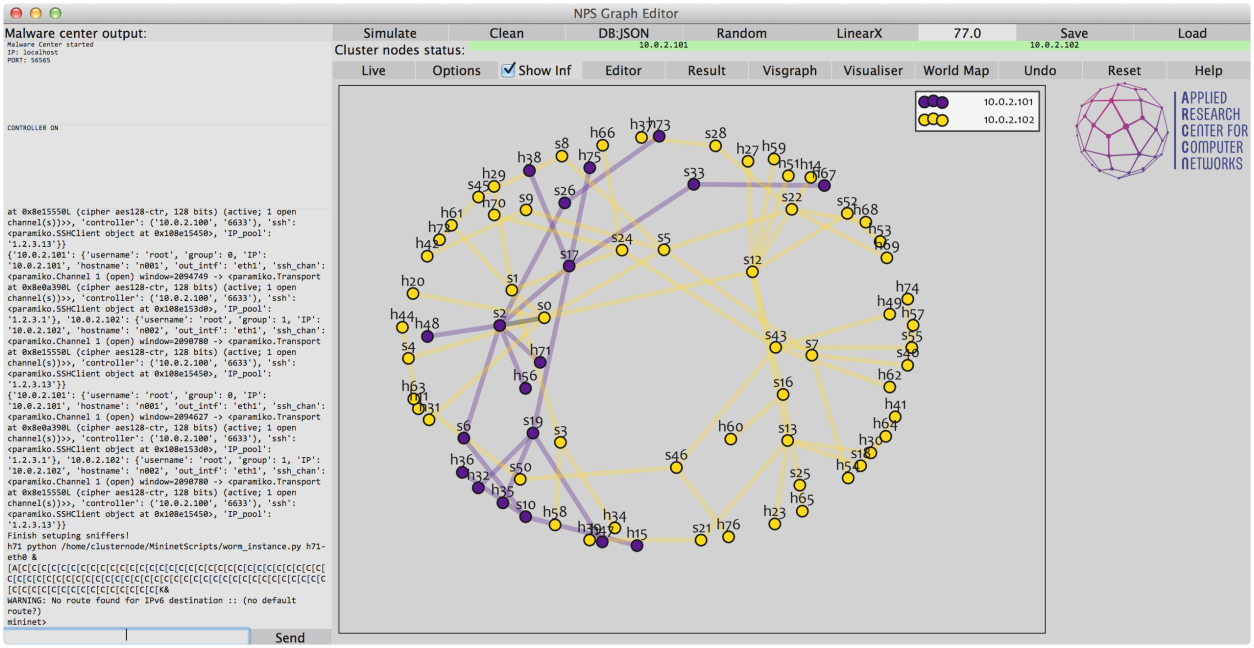

Рисунок 3.12: Визуализация распределения модели по NPS кластеру.

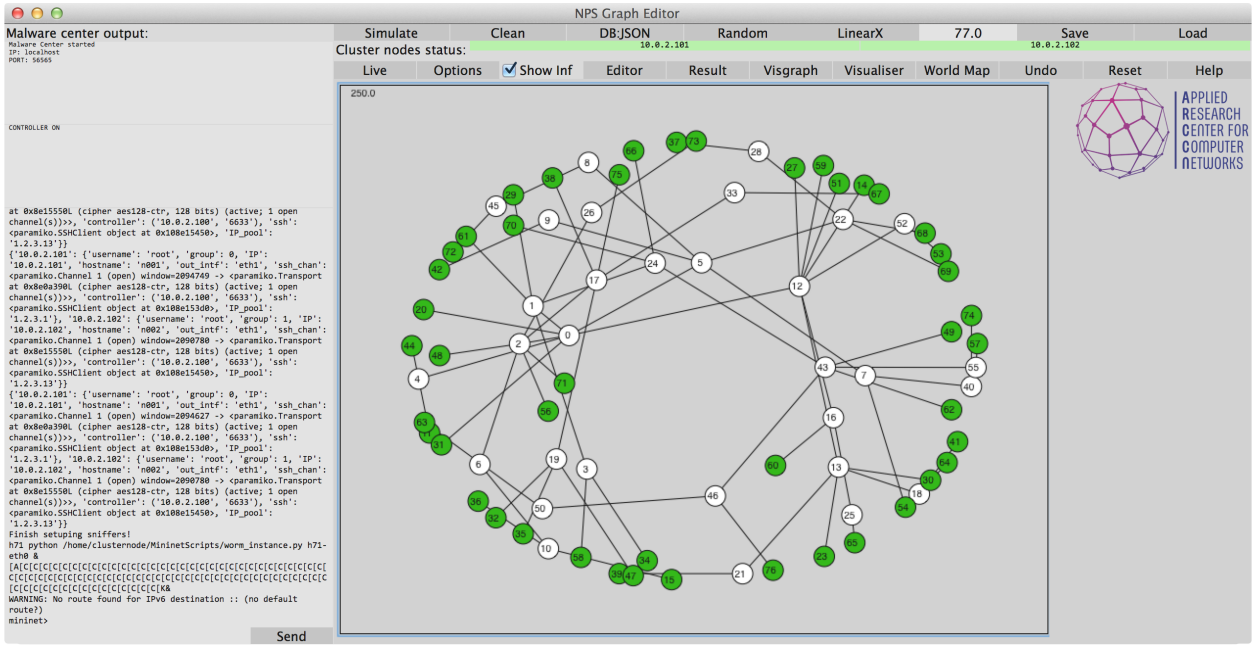

Рисунок 3.13: Визуализация процесса распространения ВПО.

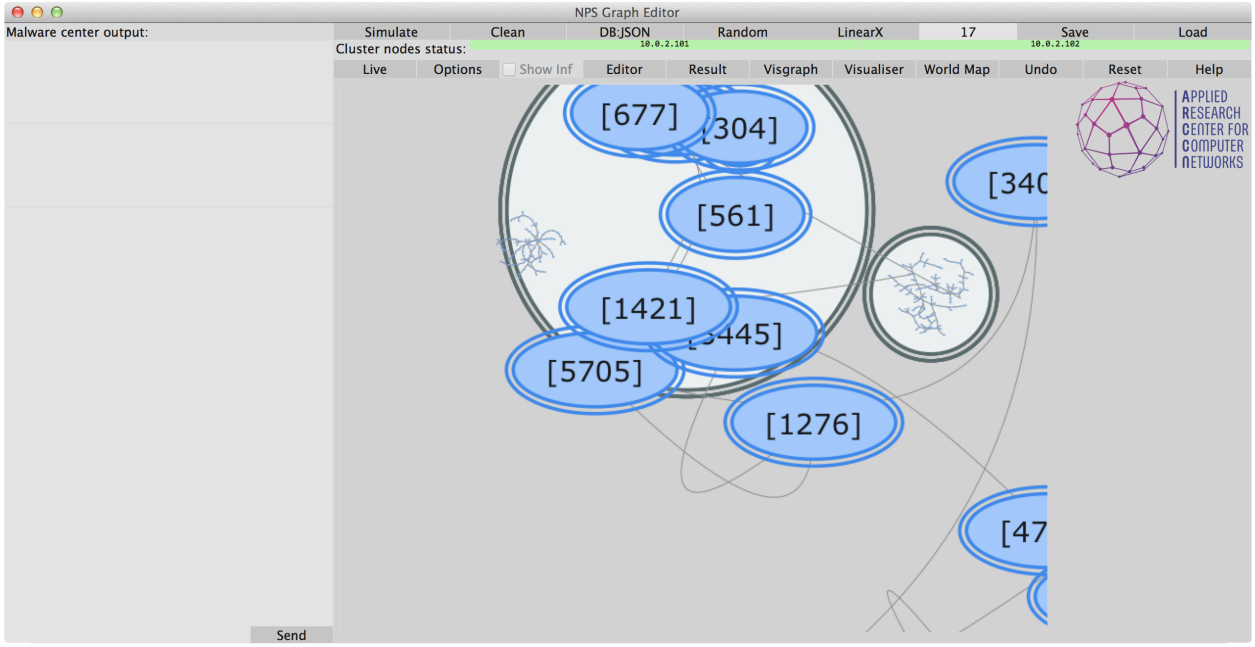

Рисунок 3.14: Визуализация больших топологий в СИМ NPS.

Глава 4

Экспериментальное исследование СИМ NPS

В данной главе представлено экспериментальное исследование разработанной и реализованной СИМ NPS на примере моделирования динамики распространения активных сетевых червей (один из типов ВПО). Продемонстрирована работа системы управления временем, входящей в состав СИМ NPS.

Экспериментальное исследование разбито на две части:

- экспериментальное исследование свойства масштабируемости разработанной СИМ NPS на примере моделирования динамики распространения ВПО;
- экспериментальное исследование синхронизации времени между компонентами СИМ NPS.

Возможность моделировать ГКС в СИМ NPS проверялась на примере распространения активных сетевых червей CodeRedv2 и Sasser. Данные сетевые черви являются одними из наиболее известных представителей своего класса. Использование в экспериментальном исследовании именно этих сетевых червей обусловлено наличием подробной аналитической информации либо о процессе распространения сетевого червя, либо о реализации экземпляра сетевого червя (например, алгоритм поиска новой жертвы, используемые порты при распространении или количество сетевых пакетов, содержащих экземпляр сетевого червя).

Результаты данных исследований экспериментально доказывают, что СИМ NPS способна моделировать процесс функционирования ГКС с возможностью моделирования динамики распространения ВПО.

4.1 Основы распространения ВПО

Сетевой червь — это один за классов ВПО. Его отличает самостоятельность предпринимаемых попыток найти новую жертву. Характеристикой этого класса ВПО является быстрое распространение в сети.

Примером червя является CodeRedv2 (2001) [103], распространявшийся через уязвимость в Microsoft's Internet Information Services (IIS) [104], или червь Slammer [105] (иногда называемых Sapphire, 2003), распространявшийся через ошибку переполнения буфера в Microsoft's SQL Server или Microsoft SQL Server Desktop Engine (MSDE) 2000 [106].

Последняя из крупных эпидемий была вызвана червем Stuxnet [107]. Данный червь захватывает компьютеры под управлением операционной системы Windows. Особенность данного червя заключалась не только в возможности захвата машин рядовых пользователей, но и промышленных систем, управляющих авторизованными производственными процессами. Уникальность его атаки заключалась в том, что впервые в истории компьютерных сетей червь физически разрушал инфраструктуру.

Потенциальными жертвами сетевого червя являются хосты, участвующие в сетевом взаимодействии. Хосты называются уязвимыми, если они обладают уязвимостью, используемой червем в процессе распространения. Соответственно, по аналогии, определяются неуязвимые хосты.

Жизненный цикл экземпляра сетевого червя разделяется на несколько стадий [108]:

1. выбор жертвы. На данной стадии экземпляр сетевого червя определяет адрес жертвы. Представление адреса зависит от уровня сетевого стека, на котором функционирует сетевой червь (от MAC-адреса на канальном уровне до номера IM-клиента на уровне приложений). Далее в работе будут рассматриваться сетевые черви, функционирующие на сетевом и транспортном уровнях стека TCP/IP. Следовательно, адрес жертвы представляется IP-адресом и портом;

2. сканирование. Каждый экземпляр сетевого червя имеет один или несколько эксплойтов, используемых для заражения жертвы. Перед тем как применить эксплойт, экземпляр сетевого червя убеждается, что необходимое сетевое приложение функционирует на выбранном хосте. Обычно это определяется по открытому порту. Примером сетевого приложения может быть популярный IM-

клиент или клиент для работы с базой данных. Выделяют несколько тактик сканирования, подробнее об устоявшихся тактиках можно прочитать в [109];

3. заражение. После нахождения необходимого сетевого приложения на хосте жертвы экземпляр сетевого червя эксплуатирует уязвимость в сетевом приложении. В случае успешной эксплуатации экземпляр сетевого червя обычно получает права пользователя, от имени которого запущено уязвимое приложение (в некоторых случаях права администратора);

4. самораспространение. Для продолжения процесса распространения и наращивания популяции зараженных хостов экземпляр сетевого червя посылает свою копию с захваченного хоста к новой жертве;

5. вредоносная активность. Данная стадия не является обязательной, она предполагает осуществление экземпляром сетевого червя дополнительных действий, направленных, например, на установку троянских программ для удаленного управления зараженным хостом.

4.2 Центр управления распространения ВПО

СИМ NPS предлагает особые возможности для исследования динамики распространения ВПО, а также последствий распространения ВПО. Примером задач, решаемых в результате имитационного моделирования динамики распространения ВПО, могут быть:

- исследование эффективности размещения систем сетевой информационной безопасности (СОИБ) на заданной топологии сети;

- определение наиболее подходящие точки для размещения СОИБ для быстрого предотвращения распространения ВПО;

- определение наиболее эффективной топологии при условии наличия ограниченного количества экземпляров СОИБ.

В системе NPS предусмотрена специализированная подсистема моделирования процесса распространения ВПО.

Данная система состоит из следующих компонент:

- центр управления распространением ВПО (ЦУР ВПО);

- центр мониторинга распространения ВПО (ЦМР ВПО);
- служба поддержки экземпляров ВПО (СПЭ ВПО).

4.2.1 Архитектура подсистемы моделирования распространения ВПО

Основными задачами ЦУР ВПО являются:

- определение захваченных и уязвимых хостов;
- определение начальной популяции ВПО в моделируемой сети.

ЦУР ВПО реализован в виде программного модуля, который запускается вместе с главной консолью управления NPS. Рабочий процесс ЦУР ВПО описывается следующими шагами:

1. определение множества уязвимых хостов из общего числа хостов;
2. определение начальной популяции ВПО из общего числа хостов;
3. отправка специализированных скриптов СПЭ ВПО на узлы NPS кластера для поддержки процесса моделирования распространения ВПО;
4. запуск ЦМР ВПО;
5. запуск экземпляров ВПО на хостах, входящих в начальную популяцию ВПО.

Для моделирования динамики распространения ВПО необходимо точно воспроизводить следующие процессы:

- процесс захвата хостов в сети;
- процесс распределения потоков трафика (как относящихся к процессу распространения ВПО, так и сторонних потоков).

Мониторинг процесса заражения и мониторинг потоков трафика являются основными задачами ЦМР ВПО. ЦМР ВПО реализован в виде программного модуля, подключенного к базе данных. Отметим, что локальная база данных имеется на каждом узле NPS кластера, один раз в определенный интервал времени (который может изменяться в зависимости от настроек эксперимента) проводит синхронизацию с главной базой управляющей NPS консоли. Главная база данных управляющей

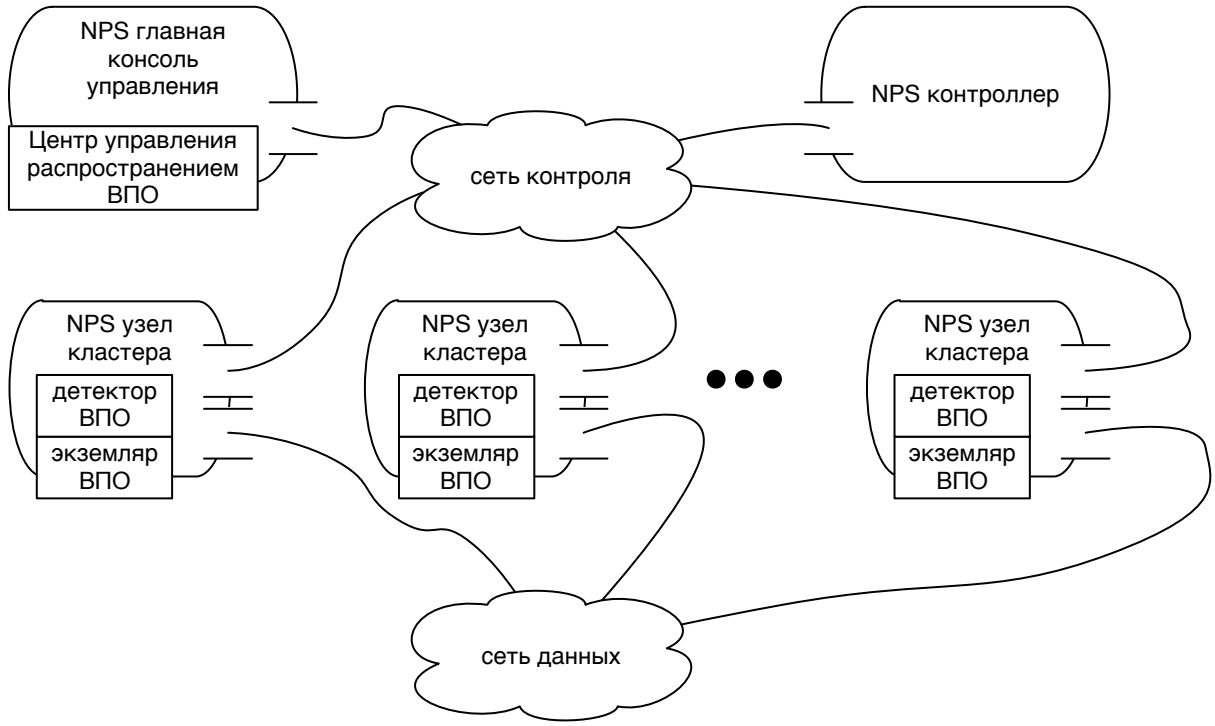

Рисунок 4.1: Центр управления распространением ВПО.

NPS консоли является основной для протоколирования и визуализации динамики распространения ВПО.

СПЭ ВПО представляет собой специально сгенерированные скрипты («экземпляр ВПО», «детектор ВПО») для каждого узла NPS кластера. Эти скрипты позволяют моделировать различные этапы распространения ВПО на хостах в моделируемой сети (этап выбора жертвы, этап сканирования, этап заражения, этап передачи тела ВПО, этап вредоносной активности). ЦУР ВПО запускает данные скрипты с определенными параметрами, основываясь на модели распространения конкретного типа ВПО. Тип ВПО фиксирован и может быть изменен пользователем системы NPS в настройках эксперимента.

Подсистема моделирования распространения ВПО позволяет достоверно моделировать все этапы жизненного цикла ВПО. Корректность моделирования распространения ВПО в системе NPS доказана на экспериментах, подробнее в разделах 4.3.1, 4.3.2, 4.4.1.

4.3 Экспериментальное исследование машстабируемости моделей, построенных в СИМ NPS

4.3.1 Эксперимент 1.1: Моделирование распространения сетевого червя CodeRedv2

В разделе 1.1 указывалось, что под понятием домен может пониматься набор сетевых хостов и маршрутизаторов. Далее в этом разделе будем считать, что домен состоит из нескольких хостов. О структуре домена ничего не известно, кроме количественных характеристик, описанных с разделе 2.2.1.

Цель эксперимента продемонстрировать работу СИМ NPS с сетями масштаба ГКС, когда домен представляется более чем одним хостом. Этот случай интересен тем, что внутри домена для моделирования распространения ВПО будут использоваться эпидемические модели, описанные в разделе 2.2.2. Использование данного механизма позволяет моделировать сети большого размера, несмотря на отсутствие подробной информации о структуре сети или нехватки вычислительных ресурсов.

В экспериментальной топологии, представленной на рисунке 4.2 каждая вершина в графе является автономной системой (АС). Приведенная топология построена по данным о связанности АС за июль 2001 года [110]. В текущем эксперименте с позиции СИМ NPS каждая АС является доменом. Распределение уязвимых хостов по доменам представлено в таблице 4.1.

Доказывать адекватность модели функционирования ГКС с возможностью моделирования динамики распространения ВПО будем на примере имитационного моделирования распространения сетевого червя CodeRedv2. По сценарию зараженные хосты будут генерировать соответствующий поток ВПО трафика в другие точки сети; выбор этих точек осуществляется случайно, исходя из алгоритма выбора жертвы сетевым червем CodeRedv2 [111].

Зная начальную популяцию червя CodeRedv2 (пусть зараженные хосты будут в доменах AS50 и AS60), можно определить популяцию зараженных хостов во всей сети через заданный интервал времени. Будем считать, что в модели сети число уязвимых хостов соответствует числу хостов, зараженных червем CodeRedv2 (примерно 359000 хостов). Настроим параметры уязвимости хостов в домене следующим образом:

Таблица 4.1: Распределение хостов по доменам.

Домен	Число уязвимых хостов	Начальная популяция
AS10	42.845	0
AS20	47.354	0
AS30	40.862	0
AS50	38.257	217
AS60	50.768	278
AS70	30.591	0
AS100	52.945	0
AS200	45.846	0
Итого	359.477	495

- уязвимые хосты будут иметь значение компоненты вектора защищенности к сетевому червю CodeRedv2 равной нулю, то есть "полностью" уязвимы;
- неуязвимые хосты будут иметь значение компоненты вектора защищенности к сетевому червю CodeRedv2 равной единице. По сути это означает, что на данных хостах либо отсутствует уязвимое приложений IIS, либо оно не функционирует.

В результате моделирования процесса распространения червя CodeRedv2 в СИМ NPS получаем число зараженных хостов. Помимо общего числа зараженных хостов, СИМ NPS фиксирует динамику изменения данного параметра, что позволяет сравнить моделируемый процесс распространения червя CodeRedv2 с данными о реальной эпидемии червя CodeRedv2. Результаты данного сравнения представлены на рисунке 4.3, 4.4.

Стоит отметить, что из-за особенностей алгоритма выбора жертвы, используемого червем CodeRedv2 (модифицированный случайный поиск), топология сети внутри АС не важна. Напомним, что связи между АС взяты, исходя из данных о реальной связанности АС на момент распространения CodeRedv2 [110].

В результате моделирования распространения сетевого червя CodeRedv2 были заражены все уязвимые хосты. При этом наблюдалась схожая динамика заражения с реальной эпидемией CodeRedv2. Следовательно, эпидемические модели, заложенные в формальную модель сети ГКС, функционируют верно.

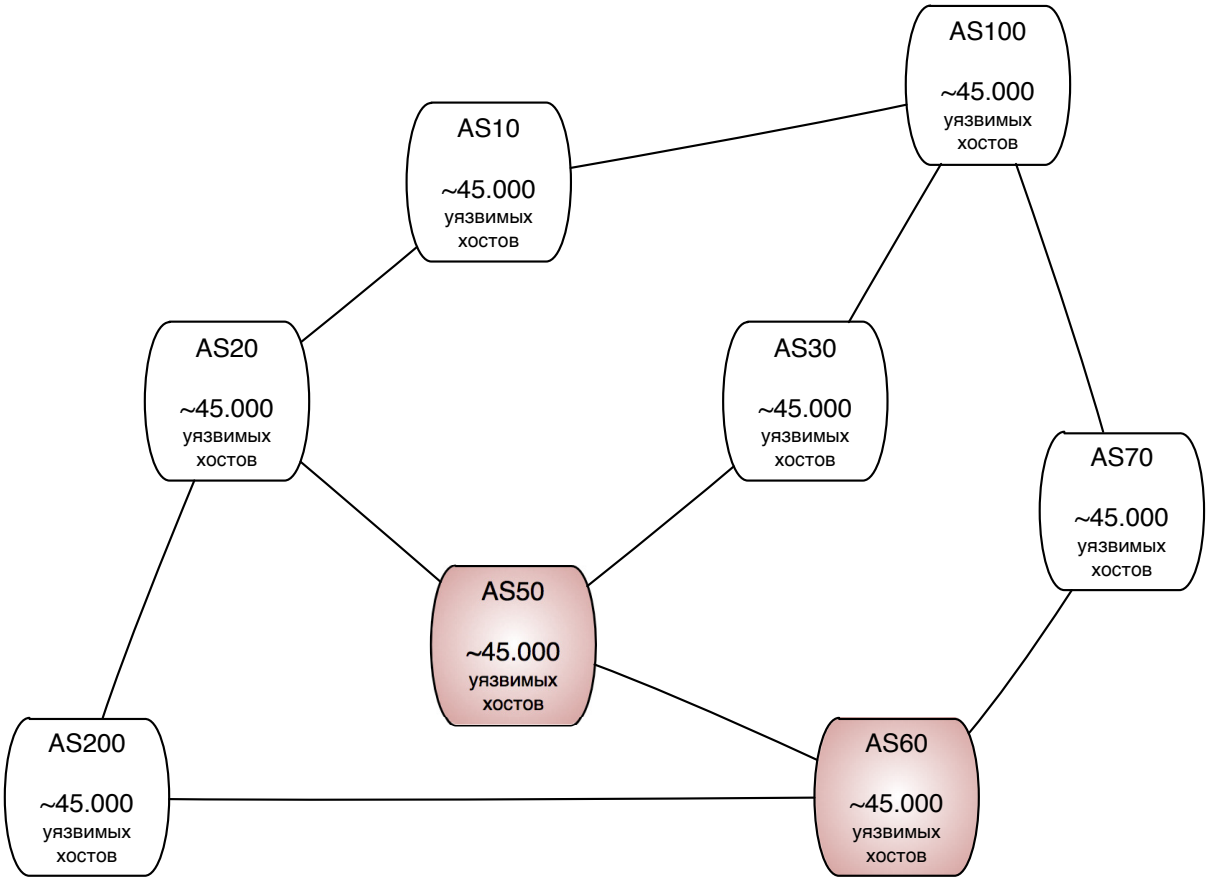

Рисунок 4.2: Экспериментальная топология сети.

Эксперимент проводился на NPS кластере, состоящем из двух вычислителей со следующими характеристиками:

1. 2xXeon E-5620, 18G DDR3 1333 MHz;

2. 2xXeon E-5620, 18G DDR3 1333 MHz.

Средняя загрузка CPU каждого вычислителя не превышала 47%. Невысокая загрузка обусловлена малым числом узлов в топологии экспериментальной сети: 8 узлов (по 4 узла на каждый вычислитель), а также использованием простых эпидемических моделей, которые не предъявляют высоких требований к вычислительным ресурсам.

Проведенный эксперимент наглядно показывает возможности масштабирования математической модели ГКС, создаваемой при помощи СИМ NPS, за счет повышения уровня абстракции узла: с хоста до домена (набора хостов). Стоит отметить, что от уровня абстракции зависит точность имитационного моделирования ГКС, однако в данном эксперименте потеря в точности моделирования сетевого обмена между хостами внутри домена не влияет на успешность результатов, так как сравнение идет

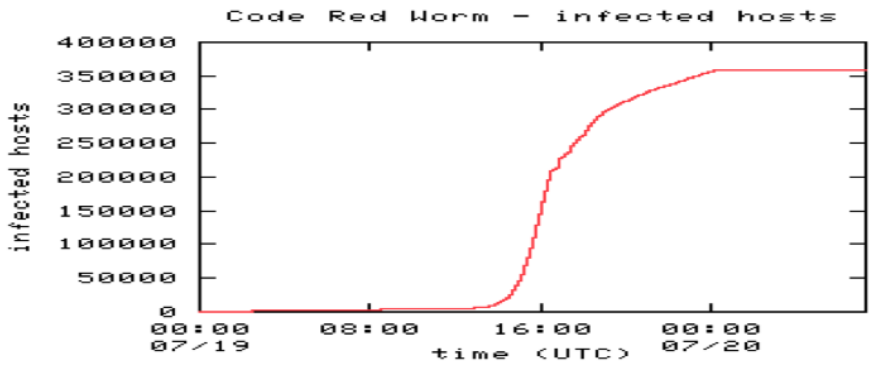

Рисунок 4.3: Реальные данные о распространению червя CodeRedv2.

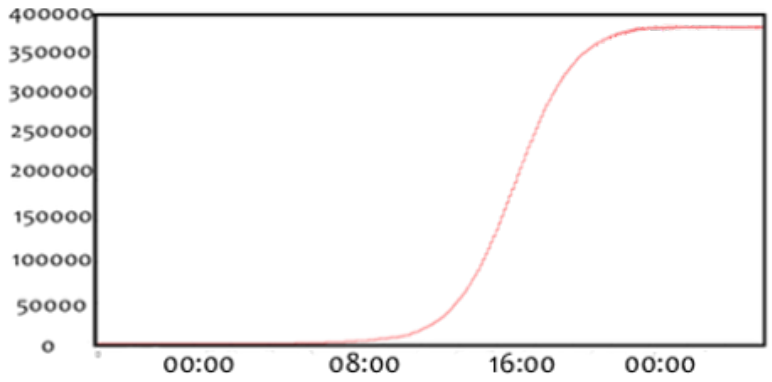

Рисунок 4.4: Результаты моделирования распространения червя CodeRedv2.

с обобщенными данными о реальном распространении сетевого червя CodeRedv2, в которых отсутствует информация о топологии сети. Эксперимент без абстрагирования до уровня доменов представлен в разделе 4.3.2.

Данные результаты были опубликованы в работе [18].

4.3.2 Эксперимент 1.2: Моделирование распространения сетевого червя Sasser

В этом эксперименте будем считать, что домен состоит из одного хоста, далее по тексту будем использовать термин хост вместо термина домен.

Цель эксперимента доказать адекватность модели функционирования ГКС, построенной в СИМ NPS, когда домен представляется единичным хостом. Этот случай интересен тем, что на каждом хосте полностью моделируются все стадии распространения сетевого червя на примере Sasser. Следовательно, моделирование

динамики распространения сетевого червя будет максимально приближенным к реальному распространению Sasser.

Методология эксперимента

Как было сказано в разделе 4.2, за процесс моделирования распространения ВПО отвечает ЦУР ВПО, главными задачами которого являются: определение начальной популяции, и запуск процесса распространения на хосте.

Процесс распространения червя Sasser разделен на несколько шагов. Каждый шаг соответствует этапу жизненного цикла ВПО, который описан в разделе 4.2. Коротко можно описать шаги распространения следующим образом:

1. модуль ЦУР ВПО определяет IP адрес следующей жертвы, руководствуясь алгоритмом распространения моделируемого червя;

2. на хост-жертву устанавливается детектор (скрипт детектора был послан на этапе конфигурации узла NPS кластера модулем СПЭ ВПО);

3. модуль ЦУР ВПО запускает генератор на атакующем хосте (используя скрипт для генерации специализированных пакетов, который был послан на этапе конфигурации узла NPS кластера модулем СПЭ ВПО);

4. в случае успешного получения последовательности сгенерированных сетевых пакетов процессом хоста-жертвы посылается запрос на обновление состояния популяции в модуль ЦМР ВПО;

5. ЦМР ВПО обновляет состояние популяции в моделируемой сети, после чего данный хост будет считаться зараженным и атаковать другие хосты.

Для моделирования процесса распространения червя Sasser необходимо в качестве входной информации предоставить описание алгоритма распространения. Сетевой червь Sasser [112] заражал компьютеры с уязвимой версией ОС Windows XP и Windows 2000. Sasser был впервые обнаружен 30 апреля 2004 года. Червь был назван Sasser из-за названия эксплуатируемого им сетевого приложения. В ходе своего распространения червь использовал ошибку переполнения буфера в компоненте ОС LSASS (Local Security Authority Subsystem Service) [113]. Червь сканировал различные наборы IP адресов и пытался соединиться с хостом жертвы через 445 TCP порт. Анализ исходного кода червя показал также, что в некоторых случаях использовался 139 порт.

Для окончания подготовки к эксперименту необходимо описать шаблон сетевой активности распространения червя Sasser, то есть размер и количество сетевых пакетов, отправляемых экземпляром червя Sasser на определенные порты в процессе распространения. Данные приведены в таблице 4.2 (источник [114]).

Таблица 4.2: Шаблон сетевой активности червя Sasser.

Порт транспортного уровня	Число сетевых пакетов	Потоки трафика
445/TCP	1254	SYN пакеты на порт 445
445/TCP	586	SYN пакеты на порт 445
9996/TCP	586	SYN пакеты на порт 9996
5554/TCP	1	SYN пакеты на порт 5554
1033/TCP	1	SYN пакеты на порт 1033

Результаты эксперимента

В данном разделе представлены результаты моделирования процесса распространения сетевого червя Sasser. В ходе эксперимента была случайным образом создана топология состоящая из 100.000 узлов. Данные узлы были распределены между 50 узлами NPS кластера примерно по 2000 вершин на каждый узел. Узел NPS кластера представлял из себя виртуальную машину на базе ОС Ubuntu 13.04.

Эксперимент проводился на двух серверах, на каждом из которых размещалось до 25 виртуальных машин, со следующими характеристиками:

1. 2xXeon E-5620, 48G DDR3 1066 MHz;
2. 1xXeon E5-2650, 32G DDR3 1600 MHz.

В эксперименте червь Sasser использует случайный алгоритм поиска новой жертвы, следовательно топология незначительно влияет на результаты моделирования. Для доказательства этого утверждения была проведена серия экспериментов со случайными топологиями разных размеров. Количественные характеристики входных данных каждого эксперимента из серии приведены в таблице 4.3. Средняя скорость заражения измеряется в количестве хостов, зараженных за один цикл распространения сетевого червя.

Таблица 4.3: Количественные данные экспериментов с разными по размерам случайными топологиями.

Количество NPS узлов кластера	Число вершин	Средняя скорость заражения
10	7.500	210
30	30.000	850
100	100.000	2690

Из таблицы видно, что средняя скорость заражение растет пропорционально росту размера сети. Это обуславливается, как было сказано ранее, особенностью алгоритма распространения. Далее будем приводить результаты самого крупного эксперимента (100.000 хостов).

Начальная популяция сетевого червя Sasser составляла 5.257 хостов (хосты выбирались случайным образом). Скорость распространения равнялась одной попытке захвата жертвы за один круг жизненного цикла ВПО. Результаты моделирования представлены на графиках (см. рисунки 4.5, 4.6).

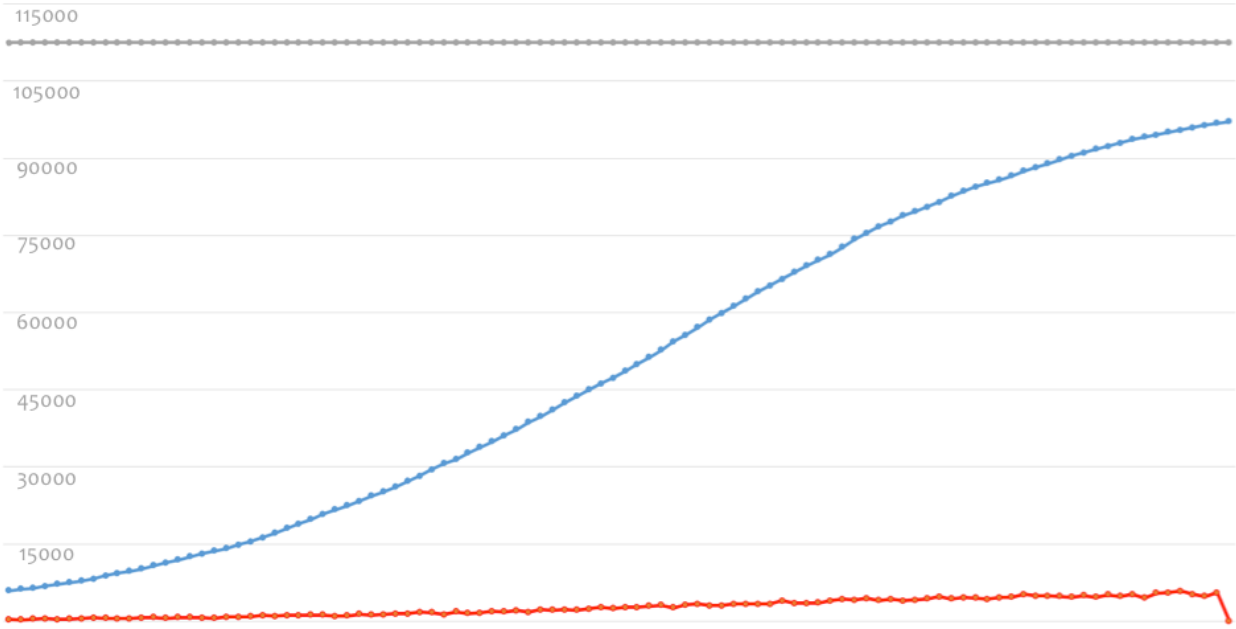

Рисунок 4.5: Результаты моделирования распространения червя Sasser.

«Серый» (верхний) график на рисунке 4.5 представляет общее число уязвимых хостов. Поскольку предполагалось, что все хосты уязвимы к заражению, данное значение является константой. «Синий» (средний) график представляет число

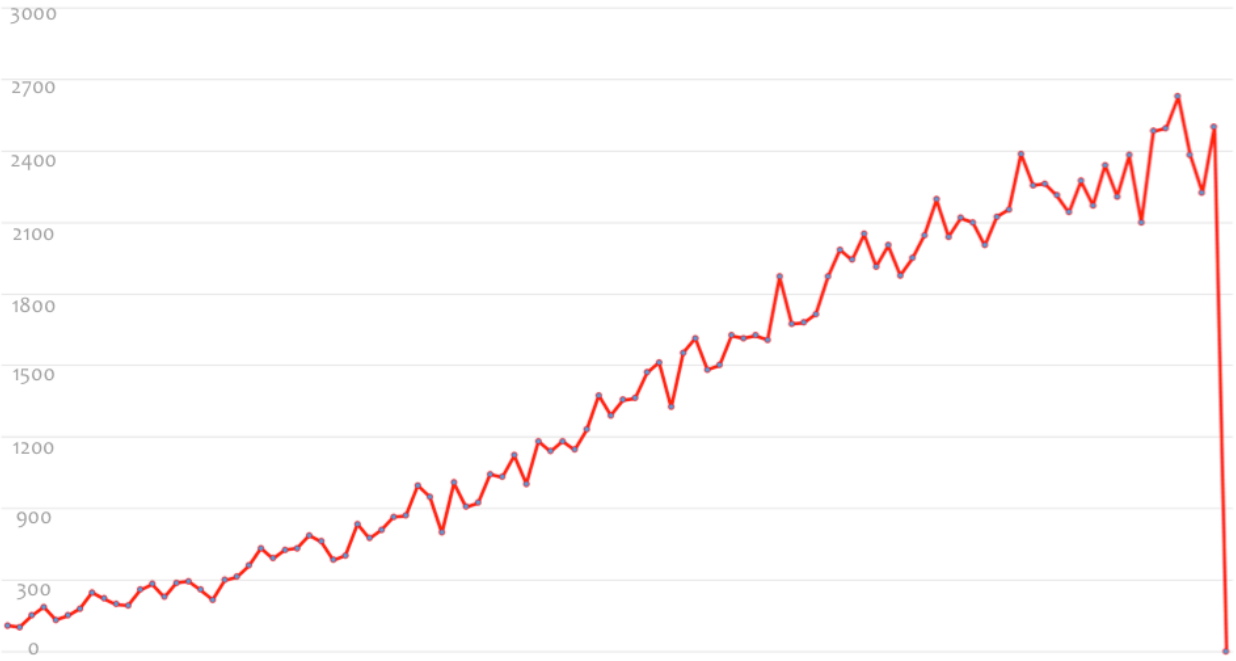

Рисунок 4.6: График скорости заражения новых жертв червем Sasser. По оси абсцисс — время распространения; по оси ординат — средняя скорость заражения.

зараженных хостов в сети. Так как в ходе эксперимента никак не моделировались контрмеры против распространения червя, «синий» график монотонно возрастает. «Красный» (нижний) график представляет число успешных попыток заражения новых жертв в ходе одного шага распространения. Как можно было заметить, число таких попыток медленно, но растет (см. рисунок 4.6).

Полученные результаты были сопоставлены с результатами наблюдения за активностью, связанной с портом 445 в глобальной сети во время эпидемии червя Sasser 4.7. Данная информации была собрана и представлена компанией SWITCH Company [115]. Черная вертикальная линия показывает период, когда компания Microsoft выпустила исправления, которые устраняют уязвимость в сетевом приложении LSASS. «Черный» график — это интерпретация результатов моделирования, описанных выше. Данные о реальной эпидемии сетевого чевря Sasser носят косвенных характер, так как наблюдается только повышение обращений к порту 445. Поэтому корреляцию данных можно только визуально сопоставить на графике.

Стоит отметить, что функциональное тестирование показало, что модуль ЦУР ВПО на каждом этапе корректно передавал тело сетевого червя Sasser (см. таблицу 4.2), а также выбирал адрес следующие жертвы в соответствии с алгоритмом распространения сетевого червя Sasser. Следовательно, каждая итерация жизненного цикла сетевого червя моделируется корректно, что позволяет моделировать динамику

распространения ВПО. Наблюдаемые результаты совпали с ожидаемым, и эксперимент завершился успешно.

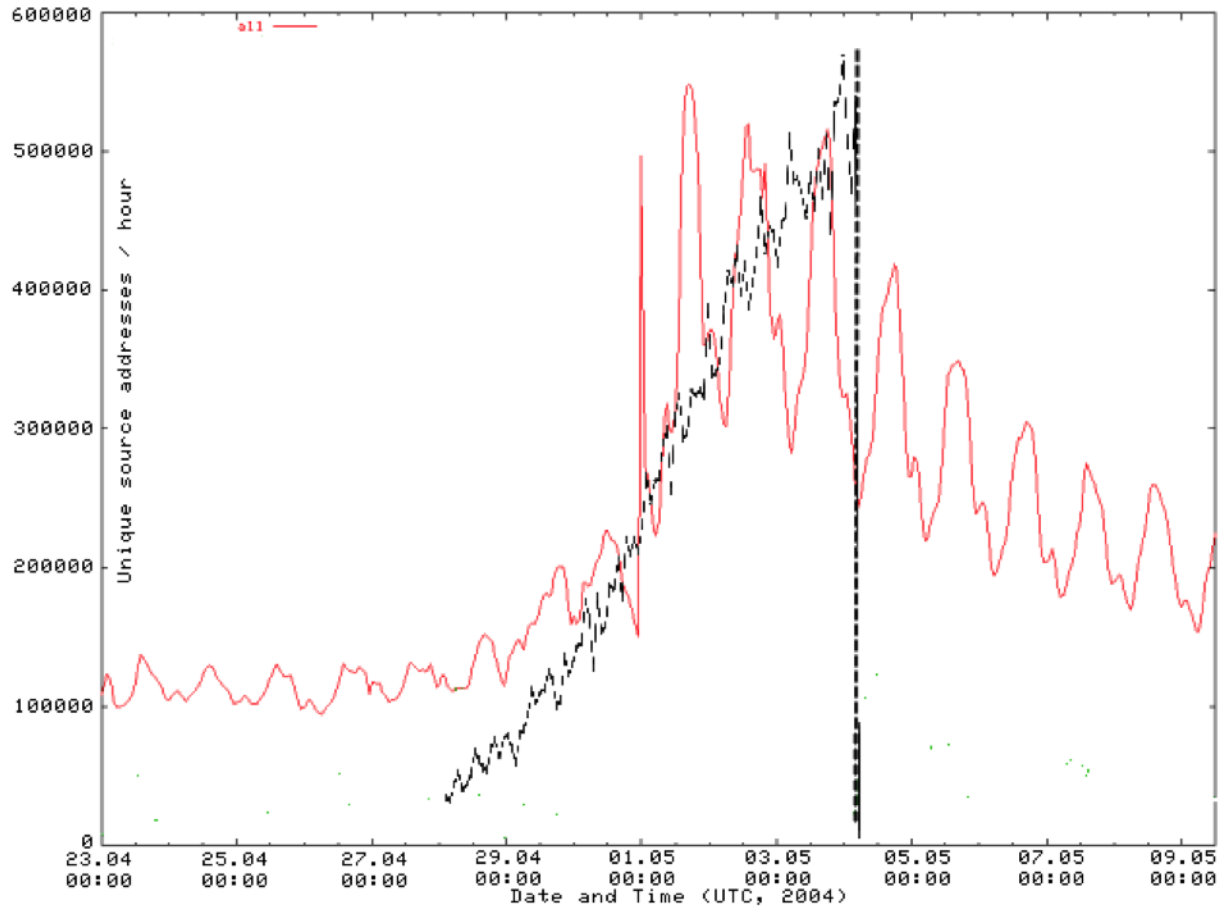

Рисунок 4.7: Сравнение результатов моделирования с реальными данными об эпидемии червя Sasser.

В результате данного эксперимента показана возможность СИМ NPS моделировать все этапы жизненного цикла ВПО. Продемонстрирована возможность работы СИМ NPS с ГКС без применения механизма абстрагирования от уровня хоста до уровня домена.

4.3.3 Выводы

Проведенные эксперименты с моделями распространения ВПО наглядно продемонстрировали возможности СИМ NPS моделировать функционирование сетей машстаба ГКС.

В этих экспериментах был воспроизведен процесс полного жизненного цикла ВПО, что позволяет говорить о СИМ NPS, как об уникальной Hi-Fi системе распространения ВПО.

Стоит отметить, что моделирование распространения ВПО является одним из многих вариантов использования СИМ NPS. Разработанная система может использоваться, например, для исследования корректности функционирования новых протоколов маршрутизации или построения тестовой площадки для новых сетевых приложений.

4.4 Экспериментальное исследование системы управления модельным временем в NPS

В главе 3 была сформулирована проблема несоответствия временных характеристик физических и моделируемых каналов (см. раздел 3.4.2). Докажем на примере эксперимента, что СИМ NPS решает обозначенную проблему.

4.4.1 Эксперимент 2.1: Временная синхронизация между разными узлами NPS кластера

Методология эксперимента

В данном разделе представлен эксперимент, демонстрирующий возможность системы управления модельным временем (СУМВ) синхронизации между частями модели, расположенными на разных узлах NPS кластера. Данный эксперимент доказывает корректность работы со временем в разработанной распределенной системе NPS. Под корректностью работы СУМВ понимается предотвращение потерь сетевого трафика из-за несоответствия временных характеристик модельных каналов с физическими каналами, поверх которых они прокладываются. Ожидаемыми результатами эксперимента являются:

- предупреждение об нарушениях порядка доставки сетевых пакетов на хосты или сбросе сетевых пакетов;
- корректировка задержки на моделируемом канале и корректировка временных характеристик сетевых приложений.

Возьмем топологию, представленную на рисунке 4.8. Данная топология развертывается на NPS кластере, состоящем из двух узлов NPS кластера, соединенных каналом с задержкой равной $dl_{phy} = 0.11$ мс. (см. рисунок 4.2). В соответствии с

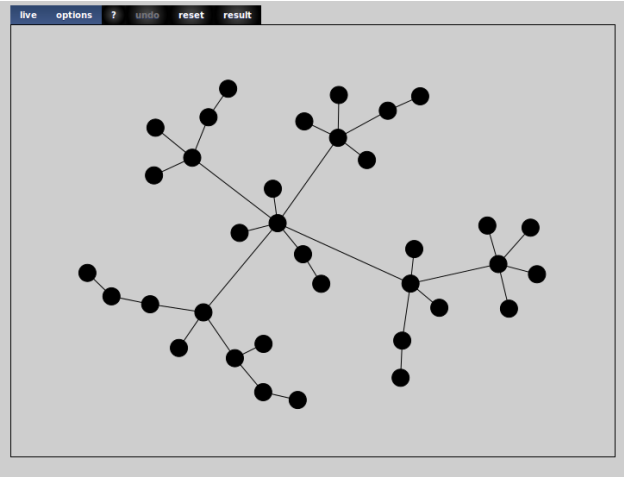

Рисунок 4.8: Граф сети.

процедурой развертывания модели поверх NPS кластера (описанной в разделе 3.5) граф сети будет разделен на две части. Для этого разбиения будут верны следующие утверждения:

- каждая из этих частей запустится на своем вычислителе в NPS кластере;
- как бы не было выполнено разбиение топологии моделируемой сети, один из каналов будет отображен на физический канал, соединяющий два вычислителя в NPS кластере.

На рисунке 4.9 зеленым цветом помечены вершины, развернутые на первом узле NPS кластера, синие — на втором узле NPS кластера. Черная дуга показывает модельный канал, который отобразился на физический канал между узлами в NPS кластере. Предполагается, что значение задержки на модельном канале равно $dl_{log} = 0.018$ мс (см. рисунок 4.1). Важно отметить, что команды «ping» на «loopback» интерфейс и «ping» между модельными хостами внутри NPS кластера показывают одинаковое значение RTT.

Результаты эксперимента

Рисунок 4.1: Задержка модельного канала.

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.104 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.032 ms
```

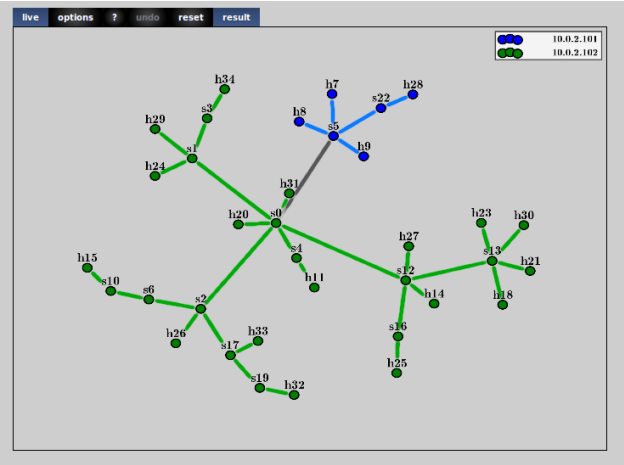

Рисунок 4.9: Размеченный граф сети.

Рисунок 4.2: Задержка физического канала.

```
PING n001 (10.0.2.7) 56(84) bytes of data.
64 bytes from n001 (10.0.2.7): icmp_seq=1 ttl=64 time=0.250 ms
64 bytes from n001 (10.0.2.7): icmp_seq=2 ttl=64 time=0.208 ms
64 bytes from n001 (10.0.2.7): icmp_seq=3 ttl=64 time=0.213 ms
64 bytes from n001 (10.0.2.7): icmp_seq=4 ttl=64 time=0.233 ms
64 bytes from n001 (10.0.2.7): icmp_seq=5 ttl=64 time=0.222 ms
```

Запустим эксперимент, не включая СУМВ. Используя NPS библиотеку сетевых приложений, установим DHCP сервер на заданный хост внутри одной из двух частей топологии сети. Запустим DHCP клиенты на всех остальных хостах в топологии сети. Таймаут на подключение к DHCP серверу установим равным наибольшему значению RTT между хостами. Напомним, что значение задержек модельных каналов задано равным $dl_{log} = 0.018$ мс. для каждого канала.

Наиболее интересным является случай, когда физическая задержка больше модельной $dl_{phy} > dl_{log}$. Тогда в результате моделирования получаем, что:

- в все хосты, которые находятся на одном с DHCP сервером узле NPS кластера, получили новые IP-адреса из подсети 1.2.5.∗ ;
- все хосты, которые находятся на разных с DHCP сервером узлах NPS кластера, остались со старыми IP-адресами из подсети 1.2.3.∗ .

Следовательно, наблюдается явное несоответствие поведения модели де-факто с ожидаемым поведением. Ожидания заключались в получении всеми хостами IP-адресов из подсети 1.2.5.∗, но некоторые хосты так и не смогли подключиться к DHCP серверу (см. рисунок 4.3). Это несоответствие вызвано тем, что задержка физического канала не удовлетворяет задержке модельного канала, определенной пользователем.

Теперь повторим эксперимент с включенной СУМВ. На стадии инициализации модели определяется максимальное отклонение физической задержки от логической — $R_{max} = \frac{dl_{phy}}{dl_{log}} \approx 6.1$ среди всех каналов модели. Далее происходит нормализация всех модельных каналов по найденному отклонению: $R_{max} \cdot dl_{log}$, где dl_{log} — модельная задержка канала. Также нормализуются с использованием R_{max} все временные характеристики сетевых приложений, функционирующих в модели сети (в нашем эксперименте это только таймаут на подключение DHCP сервера). Далее приведены результаты эксперимента (см. рисунок 4.4)

Рисунок 4.3: Неудачное применение dhclient.

```
n001 dhclient: Internet Systems Consortium DHCP Client 4.2.4
n001 dhclient: Copyright 2004-2012 Internet Systems Consortium.
n001 dhclient: All rights reserved.
n001 dhclient: For info, please visit https://www.isc.org/software/dhcp/
n001 dhclient:
n001 dhclient: Listening on LPF/h9-eth0/de:a7:d7:dd:34:bf
n001 dhclient: Sending on LPF/h9-eth0/de:a7:d7:dd:34:bf
n001 dhclient: Sending on Socket/fallback
n001 dhclient: DHCPDISCOVER on h9-eth0 to 255.255.255.255 port 67 interval 2 (xid=0x9454a18)
n001 dhclient: No DHCPOFFERS received.
n001 dhclient: No working leases in persistent database - sleeping.
```

Рисунок 4.4: Успешное применение dhclient.

```
n001 dhclient: Internet Systems Consortium DHCP Client 4.2.4
n001 dhclient: Copyright 2004-2012 Internet Systems Consortium.
n001 dhclient: All rights reserved.
n001 dhclient: For info, please visit https://www.isc.org/software/dhcp/
n001 dhclient:
n001 dhclient: Listening on LPF/h8-eth0/8a:04:6d:e0:45:e6
n001 dhclient: Sending on LPF/h8-eth0/8a:04:6d:e0:45:e6
n001 dhclient: Sending on Socket/fallback
n001 dhclient: DHCPDISCOVER on h8-eth0 to 255.255.255.255 port 67 interval 2 (xid=0x11e23d19)
n001 dhclient: DHCPREQUEST of 1.2.5.96 on h8-eth0 to 255.255.255.255 port 67 (xid=0x11e23d19)
n001 dhclient: DHCPOFFER of 1.2.5.96 from 1.2.3.21
n001 dhclient: DHCPACK of 1.2.5.96 from 1.2.3.21
n001 dhclient: bound to 1.2.5.96 -- renewal in 280 seconds.
```

В итоге отмечаем, что после включения СУМВ поведение модели де-факто соответствует ожидаемому поведению. Данный вывод делается на основании получения всеми хостами в модели сети IP адресов из подсети 1.2.5.$*$ (см. рисунок 4.5). Платой за корректность в работе с модельным временем является увеличение астрономического времени проведения эксперимента. Так время проведения эксперимента увеличилось с 88.35 сек до 187.6 сек.

Рисунок 4.5: Настройки сетевых интерфейсов хостов.

```
h28-eth0  Link encap:Ethernet  HWaddr c6:29:c0:57:16:00
 inet addr:1.2.5.90  Bcast:1.2.255.255  Mask:255.255.0.0
 inet6 addr: fe80::c429:c0ff:fe57:1600/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
 RX packets:412 errors:0 dropped:4 overruns:0 frame:0
 TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:46378 (46.3 KB)  TX bytes:1820 (1.8 KB)
...
...
h20-eth0  Link encap:Ethernet  HWaddr c2:e7:26:13:6e:c0
 inet addr:1.2.5.77  Bcast:1.2.255.255  Mask:255.255.0.0
 inet6 addr: fe80::c0e7:26ff:fe13:6ec0/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
 RX packets:367 errors:0 dropped:6 overruns:0 frame:0
 TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:36766 (36.7 KB)  TX bytes:1820 (1.8 KB)
```

Стоит отметить, что отношение R_{max} — это константа. Следовательно, временные метки всех событий, произошедших в ходе эксперимента, могут быть легко пересчитаны в выходных трассах модели сети.

Представленный эксперимент демонстрирует успешное решение проблемы несоответствия временных характеристик физических и моделируемых каналов.

4.5 Выводы

Экспериментальное исследование показало, что разработанная и реализованная распределенная система имитационного моделирования ГКС полностью удовлетворяет требованиям (точности моделирования процесса сетевого обмена данными и масштабируемости модели сети, создаваемой в СИМ NPS), описанных в разделе 1.1.1.

В ходе экспериментов 4.3.1 4.3.2 было показано, что процесс распространения ВПО в модели практически полностью совпадает с данными о реальном распространении ВПО. Различия между моделью и реальностью минимальны и связанны с неполнотой данных о реальном распространении ВПО (информацией и топологии, либо о количестве уязвимых хостов).

Отдельно был проведен эксперимент 4.4.1 с использованием СУМВ. В ходе этого эксперимента было продемонстрировано успешное решение проблемы рассинхронизации между различными частями модели сети, расположенными на разных вычислителях.

Заключение

Основные результаты работы заключаются в следующем:

1. Построена математическая модель, позволяющая корректно и адекватно моделировать функционирование ГКС. В рамках модели поставлена задача прогнозирования динамики распространения ВПО и доказано, что она имеет решение.

2. Предложен подход к имитационному моделированию сети на основе техник легковесной виртуализации, позволяющий строить имитационные модели сетей большого размера так, что процесс обработки и передачи сетевого трафика воспроизводится аналогично процессам обмена трафиком в реальной сети.

3. Разработана и реализована уникальная распределенная система имитационного моделирования (СИМ) сети, названная Network Prototype Simulator (NPS). В основе СИМ лежит аппарат легковесной виртуализации, который позволяет строить эффективно масштабируемые модели сети с высокой точностью воспроизведения процессов обработки и передачи сетевого трафика.

СИМ NPS обладает широкими перспективами развития. Возможные дальнейшие направления развития NPS заключаются в:

- расширении списка сетевых приложений в NPS библиотеке сетевых приложений;

- создании версии для других сетевых стеков, используемых при соединении NPS узлов кластера, например, это может быть полезно для запуска процесса моделирования на суперкомпьютерных вычислителях с использованием Infiniband-соединений;

- разработке перспективных средств визуализации больших топологий компьютерных сетей на базе графического интерфейса пользователя СИМ NPS.

Предложенный в работе подход к имитационному моделированию функционирования ГКС открывает новые возможности для исследователей и разработчиков новых сетевых решений по тестированию и построению сетевого окружения. СИМ NPS является открытой, уникальной площадкой для тестирования сетевого оборудования и сетевых протоколов, а так же универсальной исследовательской площадкой для изучения различных сетевых активностей в ГКС.

Список рисунков

Список таблиц

Литература

1. Tanachaiwiwat Sapon, Helmy A. Computer worm ecology in encounter-based networks // Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on. 2007. Sept. С. 535–543.

2. Tanachaiwiwat Sapon, Helmy A. Modeling and analysis of worm interactions (war of the worms) // Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on. 2007. Sept. С. 649–658.

3. Measuring and evaluating large-scale CDNs / Cheng Huang, Angela Wang, Jin Li [и др.] // In IMC. 2008.

4. Huang C., Li J., Wang A. [и др.]. Determining Network Delay and CDN Deployment. 2010.

5. Filsls Clarence, Mohapatra Pradosh, Bettink John [и др.]. BGP Prefix Independent Convergence (PIC) Technical Report. 2011.

6. Mao Yun, Saul Lawrence K. Modeling Distances in Large-scale Networks by Matrix Factorization // Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement. IMC '04. New York, NY, USA: ACM, 2004. С. 278–287. URL: http://doi.acm.org/10.1145/1028788.1028827.

7. Fluid Models and Solutions for Large-Scale IP Networks / Yong Liu, Francesco Lo Presti, Vishal Misra [и др.] // In Proc. of ACM SIGMETRICS. 2003. С. 91–101.

8. Liu Xin, Chien Andrew A. Realistic Large-Scale Online Network Simulation. 2004.

9. Simulation of large scale networks using SSF / D.M. Nicol, J. Liu, M. Liljenstam [и др.] // Simulation Conference, 2003. Proceedings of the 2003 Winter. Т. 1. 2003. Dec. С. 650–657 Vol.1.

10. Liu J., Li Yue, He Ying. A large-scale real-time network simulation study using PRIME // Simulation Conference (WSC), Proceedings of the 2009 Winter. 2009. Dec. C. 797–806.

11. Р.Л. Смелянский. Компьютерные сети. В 2 томах. Том 1. Системы передачи данных. Академия, 2011.

12. Rewaskar Sushant, Kaur Jasleen, Smith F. Donelson. A passive state-machine approach for accurate analysis of TCP out-of-sequence segments // ACM Computer Communication Review. 2006. T. 36. c. 2006.

13. Reproducible Network Experiments Using Container-based Emulation / Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar [и др.] // Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies. 2012. C. 253–264.

14. Wang Shie Yuan, Kung H. T. A Simple Methodology for Constructing an Extensible and High-Fidelity TCP/IP Network Simulators. // INFOCOM. 1999. C. 1134–1143.

15. OpenFlow: enabling innovation in campus networks / Nick McKeown, Tom Anderson, Hari Balakrishnan [и др.] // SIGCOMM Comput. Commun. Rev. 2008. T. 38. C. 69–74.

16. Kirkpatrick Keith. Software-defined Networking // Commun. ACM. 2013. T. 56. C. 16–19.

17. Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches / Vinesh Gudla, Saurav Das, Anujit Shastri [и др.] // IEEE/OSA Optical Fiber Communication Conference OFC 2010. 2010.

18. Antonenko Vitaly, Smelyanskiy Ruslan. Global Network Modelling Based on Mininet Approach. // Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. HotSDN '13. New York, NY, USA: ACM, 2013. C. 145–146. URL: http://doi.acm.org/10.1145/2491185.2491211.

19. Антоненко В.А., Смелянский Р.Л. Моделирование вредоносной активности в глобальной компьютерной сети // Программирование. 2013. Т. 1. С. 60–72.

20. Р.Л. Смелянский. Компьютерные сети. В 2 томах. Том 2. Сети ЭВМ. Академия, 2011.

21. QoS Parameters to Network Performance Metrics Mapping for SLA Monitoring / H. J. Lee, M. S. Kim, J. W. Hong [и др.]. 2002.

22. Nicol David M., Liljenstam Michael, Liu Jason. Advanced Concepts in Large-scale Network Simulation // Proceedings of the 37th Conference on Winter Simulation. WSC '05. Winter Simulation Conference, 2005. С. 153–166. URL: http://dl.acm.org/citation.cfm?id=1162708.1162740.

23. Rinse: the real-time immersive network simulation environment for network security exercises / Michael Liljenstam, Jason Liu, David Nicol [и др.] // In Proceedings of the 19th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS). 2005.

24. Hybrid Modeling for Large-scale Worm Propagation Simulations / Eul Gyu Im, Jung Taek Seo, Dong-Soo Kim [и др.] // Proceedings of the 4th IEEE International Conference on Intelligence and Security Informatics. 2006. Т. 6. С. 572–577.

25. Large-scale Virtualization in the Emulab Network Testbed / Mike Hibler, Robert Ricci, Leigh Stoller [и др.] // USENIX 2008 Annual Technical Conference on Annual Technical Conference. ATC'08. Berkeley, CA, USA: USENIX Association, 2008. С. 113–128. URL: http://dl.acm.org/citation.cfm?id=1404014.1404023.

26. Н.П. Бусленко. Моделирование сложных систем. Наука, 1978.

27. Б.Я. Советов. Моделирование систем: учеб. для вузов. Высш. шк., 2001.

28. С.А. Бешенков. Моделирование и формализация: методическое пособие. Лаборатория базовых знаний, 2002.

29. С.В. Назарова. Локальные вычислительные сети. Кн. 3. Организация функционирования, эффективность, оптимизация: справочник. Финансы и статистика, 1995.

30. О.М. Замятина. Моделирование сетей: учебное пособие. Изд-во Томского политехнического университета, 2011.

31. A Markov-based Channel Model Algorithm for Wireless Networks / Almudena Konrad, Ben Y. Zhao, Anthony D. Joseph [и др.] // Wirel. Netw. 2003. Т. 9. С. 189–199.

32. Jagerman David L., Melamed Benjamin, Willinger Walter. Frontiers in Queueing / под ред. Jewgeni H. Dshalalow. Boca Raton, FL, USA: CRC Press, Inc., 1997. С. 271–320. URL: http://dl.acm.org/citation.cfm?id=279251.279263.

33. Mondragón R. J., Arrowsmith D. K., Pitts J. M. Chaotic Maps for Traffic Modelling and Queueing Performance Analysis // Perform. Eval. 2001. Т. 43. С. 223–240.

34. Б.В. Гнеденко. Курс теории вероятностей. Едиториал УРСС, 2005.

35. Analytical studies of superposition GMDP traffic sources for ATM networks / Z. Sun, J. Cosmas, L.G. Cuthbert [и др.] // Computer and Communication Systems, 1990. IEEE TENCON'90., 1990 IEEE Region 10 Conference on. 1990. Sep. С. 772–775 vol.2.

36. Liu Hai, Ansari N., Shi Y.Q. Modeling VBR video traffic by Markov-modulated self-similar processes // Multimedia Signal Processing, 1999 IEEE 3rd Workshop on. 1999. С. 363–368.

37. Ihler Alexander, Hutchins Jon, Smyth Padhraic. Learning to Detect Events with Markov-modulated Poisson Processes // ACM Trans. Knowl. Discov. Data. 2007. Т. 1.

38. М. Л. Федорова Т. М. Леденева. Об исследовании свойства самоподобия трафика мультисервисной сети. // Вестник ВГУ, серия: Системный анализ и информационные технологии. 2010. С. 46–54.

39. Dang Trang Dinh, Sonkoly B., Molnar S. Fractal analysis and modeling of VoIP traffic // Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International. 2004. June. С. 123–130.

40. Прохоров Ю.В., Адян С.И. Математический энциклопедический словарь. Сов. энцикл., 1988.

41. Матвеев В.Ф. Ушаков В.Г. Системы массового обслуживания. Изд-во МГУ, 1984.

42. И.В. Максимей. Имитационное моделирование на ЭВМ. Радио и связь, 1988.

43. Таха Хемди А. Введение в исследование операций. Вильямс, 2007.

44. Nicol David, Goldsby Michael, Johnson Michael. Fluid-based Simulation of Communication Networks using SSF. 1999.

45. Дж. Питерсон. Теория сетей Петри и моделирование систем. Мир, 1984.

46. О. Оре. Графы и их применение. Изд-во МИР, 1965.

47. Меньших ВВ, Петрова ЕВ. ПРИМЕНЕНИЕ МЕТОДОВ ТЕОРИИ АВТОМАТОВ ДЛЯ МОДЕЛИРОВАНИЯ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ // Вестник Воронежского Института Мвд России. 2009. Т. 1.

48. Bochmann Gregor V. Conformance Testing Methodologies and Architectures for OSI Protocols / под ред. Richard Jerry Linn, M. Ümit Uyar. Los Alamitos, CA, USA: IEEE Computer Society Press, 1995. С. 66–77. URL: http://dl.acm.org/citation.cfm?id=202035.202044.

49. С.С. Зайцев. Описание и реализации протоколов сетей ЭВМ. Наука, 1989.

50. Holmevik J.R. Compiling SIMULA: a historical study of technological genesis // Annals of the History of Computing, IEEE. 1994. Т. 16. С. 25–37.

51. Crain R.C., Henriksen J.O. Simulation using GPSS/H // Simulation Conference Proceedings, 1999 Winter. Т. 1. 1999. С. 182–187 vol.1.

52. Linux Bridge online documentation. http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge/.

53. OpenVSwitch homepage. http://openvswitch.org/.

54. Click Router homepage. http://read.cs.ucla.edu/click/click/.

55. NS-3 homepage. http://www.nsnam.org/.

56. Emulab homepage. http://www.emulab.net/.

57. OPNET Modeler homepage. http://www.opnet.com/solutions/network_rd/modeler.html.

58. OMNeT++ Network Simulation Framework homepage. http://www.omnetpp.org/.

59. NS-2 homepage. http://www.isi.edu/nsnam/ns/.

60. Extending Tcl for Dynamic Object-Oriented Programming / David Wetherall, David Wetherall, Christopher J. Lindblad [и др.] // In Proceedings of the Tcl/Tk Workshop. 1995.

61. B. Braden D. Clark J. Crowcroft [и др.]. Recommendations on Queue Management and Congestion Avoidance in the Internet: RFC: 2309: RFC Editor, 1998. April. URL: http://tools.ietf.org/html/rfc2309.

62. McKenney P.E. Stochastic fairness queueing // INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE. 1990. Jun. С. 733–740 vol.2.

63. NetAnimator Homepage. http://netanimator.sourceforge.net.

64. PyViz Homepage. http://www.nsnam.org/wiki/PyViz.

65. Garifullin Maxim, Borshchev Andrei, Popkov Timofei. Using Anylogic and Agent-Based Approach to Model Consumer Market // EUROSIM 2007. 2007.

66. Wei Songjie, Mirkovic J., Swany M. Distributed worm simulation with a realistic Internet model // Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on. June. С. 71–79.

67. Simulating realistic network worm traffic for worm warning system design and testing / Michael Liljenstam, David M. Nicol, Vincent H. Berk [и др.] // Proceedings of the 2003 ACM workshop on Rapid malcode. WORM '03. New York, NY, USA: ACM, 2003. С. 24–33. URL: http://doi.acm.org/10.1145/948187.948193.

68. Cowie J. H. Nicol D. M. Ogielski A. T. Modeling the global internet // Computing in Science and Engg. 1999. Т. 8. С. 42–50.

69. Mininet. An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/.

70. Mininet github repo. https://github.com/mininet/mininet.

71. Lantz Bob, Heller Brandon, McKeown Nick. A network in a laptop: rapid prototyping for software-defined networks // Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. Hotnets-IX. New York, NY, USA: ACM, 2010. С. 19:1–19:6. URL: http://doi.acm.org/10.1145/1868447.1868466.

72. Heller Brandon, Sherwood Rob, McKeown Nick. The Controller Placement Problem // Proceedings of the First Workshop on Hot Topics in Software Defined Networks. HotSDN '12. New York, NY, USA: ACM, 2012. С. 7–12. URL: http://doi.acm.org/10.1145/2342441.2342444.

73. Das Saurav, Parulkar Guru, Mckeown Nick [и др.]. Packet and Circuit Network Convergence with OpenFlow.

74. NOX: towards an operating system for networks / Natasha Gude, Teemu Koponen, Justin Pettit [и др.] // SIGCOMM Comput. Commun. Rev. 2008. Т. 38. С. 105–110.

75. Linux Containers (LXC) HOWTO homepage. http://lxc.teegra.net/.

76. Pan Jianli. A Survey of Network Simulation Tools: Current Status and Future Developments: Tech. Rep.: : 2008. November.

77. Smelyanskiy R. L. On the Theory of Functioning of Distributed Computer Systems // Proc. of the Int. Conf. on Parallel Computing and Control Problems (PACO'2001). 2001. С. 161–182.

78. Ferro Greg. Packets Are the Past, Flows Are the Future. 2012. November. http://www.networkcomputing.com/networking/packets-are-the-past-flows-are-the-future/a/d-id/1233975?

79. Monitoring and Early Warning for Internet Worms / Cliff Changchun Zou, Lixin Gao, Weibo Gong [и др.] // Proceedings of the 10th ACM Conference on Computer and Communications Security. CCS '03. New York, NY, USA: ACM, 2003. С. 190–199. URL: http://doi.acm.org/10.1145/948109.948136.

80. Котенко И. В. Воронцов В. В. Аналитические модели распространения сетевых червей // Труды СПИИРАН. 2007. С. 208–224.

81. Л. Клейнрок. Теория массового обслуживания. Машиностроение, 1979.

82. Филлипс Д. Гарсиа-Диас А. Методы анализа сетей. Мир, 1984.

83. Bonald Thomas, Roberts James. Scheduling Network Traffic // SIGMETRICS Perform. Eval. Rev. 2007. Т. 34. С. 29–35.

84. Колмогоров А. Н. Фомин С. В. Элементы теории функций и функционального анализа, глава II: Метрические и топологические пространства, §5 Топологические пространства, раздел 1: Определение и примеры топологических пространств. Наука, 1981.

85. Колмогоров А. Н. Фомин С. В. Элементы теории функций и функционального анализа, глава IV: Линейные функционалы н линейные операторы, §1 Непрерывные линейные функционалы, раздел 1: Непрерывные линейные функционалы в топологических линейных пространствах. Наука, 1981.

86. Smelyansky R.L., Kazakov Y.P. Organization of synchronization algorithms in distributed simulation. T. 2. 1994. c. 45.

87. Network Prototype Simulator homepage. https://github.com/ARCCN/nps.

88. Paramiko Module main site. http://www.lag.net/paramiko/.

89. Python 2.7 documentation. http://www.python.org/doc/.

90. Python-Scapy Module main site. http://www.secdev.org/projects/scapy/.

91. Droms R. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard). 1997. mar. Updated by RFCs 3396, 4361, 5494.

92. Mills D.L. Network Time Protocol (NTP). RFC 958. 1985. September. Obsoleted by RFCs 1059, 1119, 1305.

93. Mockapetris P. RFC 1034 Domain Names - Concepts and Facilities. 1987.

94. Fielding R., Gettys J., Mogul J. [и др.]. RFC 2616, Hypertext Transfer Protocol – HTTP/1.1. 1999.

95. Bangsow Steffen. Use Cases of Discrete Event Simulation: Appliance and Research. Springer, 2012.

96. Ю. Харитонов В. Сетевые механизмы обеспечения согласованности данных в распределенных системах виртуальной реальности. Ph.D. thesis: Московский Энергетический Институт. 2010.

97. Кельтон В. Лоу А. Имитационное моделирование. Классика CS. ПИТЕР, 2004.

98. Smelyansky R.L. Model of distributed computing system operation with time // Programming and Computer Software. 2013. T. 39. C. 233–241.

99. Intel Corporation. InfiniBand Software Architecture Access Layer High Level Design. 2002.

100. Technologies Mellanox. Mellanox OpenStack and SDN/OpenFlow Solution Reference Architecture. 2013. September.

101. Shavitt Yuval, Zilberman Noa. A Study of Geolocation Databases // CoRR. 2010. T. abs/1005.5674.

102. Network Prototype Simulator installation manual. http://arccn.github.io/NPS/.

103. Zou Cliff Changchun, Gong Weibo, Towsley Don. Code red worm propagation modeling and analysis // Proceedings of the 9th ACM conference on Computer and communications security. CCS '02. New York, NY, USA: ACM, 2002. C. 138–147. URL: http://doi.acm.org/10.1145/586110.586130.

104. Microsoft Security Bulletin MS01-033. http://technet.microsoft.com/en-us/security/bulletin/ms01-033.

105. Inside the Slammer Worm / David Moore, Vern Paxson, Stefan Savage [и др.] // IEEE Security and Privacy. 2003. Т. 1. С. 33–39.

106. Microsoft Security Bulletin MS02-039. http://technet.microsoft.com/en-us/security/bulletin/ms02-039.

107. Aleksandr Matrosov Eugene Rodionov David Harley, Malcho Juraj. Stuxnet Under the Microscope. 2011.

108. Fei Su, Zhaowen Lin, Yan Ma. A survey of internet worm propagation models // Broadband Network Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on. Oct. С. 453–457.

109. Zou Cliff C., Towsley Don, Gong Weibo. On the performance of internet worm scanning strategies // Perform. Eval. 2006. Т. 63. С. 700–723.

110. AS Links Dataset. http://www.caida.org/data/active/skitter_aslinks_dataset.xml.

111. Moore David, Shannon Colleen, claffy k. Code-Red: A Case Study on the Spread and Victims of an Internet Worm // Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment. IMW '02. New York, NY, USA: ACM, 2002. С. 273–284. URL: http://doi.acm.org/10.1145/637201.637244.

112. Sasser Worm Symantec Description. http://www.symantec.com/security_response/writeup.jsp?docid=2004-050116-1831-99.

113. Microsoft Security Bulletin MS04-011. http://technet.microsoft.com/en-us/security/bulletin/ms04-011.

114. Terada Masato Takada Shingo Doi Norihisa. Proposal for the Experimental Environment for Network Worm Infection // Transactions of Information Processing Society of Japan. 2005. T. 46. C. 2014–2024.

115. SWITCH company main site. http://www.switch.ch/.