УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2018 Том 18 № 3 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2018 Vol. 18 No 3 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.77

МОДЕЛИРОВАНИЕ ETHERNET СЕТЕЙ В СРЕДЕ OMNET++ INET FRAMEWORK

С.П. Хабаров[a]

[a] Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, Санкт-Петербург, 194021, Российская Федерация
Адрес для переписки: Serg.Habarov@mail.ru

Информация о статье
Поступила в редакцию 13.03.18, принята к печати 15.04.18
doi: 10.17586/2226-1494-2018-18-3-462-472
Язык статьи – русский

Ссылка для цитирования: Хабаров С.П. Моделирование Ethernet сетей в среде OMNeT++ INET framework // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 3. С. 462–472. doi: 10.17586/2226-1494-2018-18-3-462-472

Аннотация

Предмет исследования. Рассмотрен один из возможных подходов к исследованию Ethernet сетей путем их имитационного моделирования в среде OMNeT++ с использованием фреймворка INET. Показана возможность исследования сетевых моделей как в режиме пошагового выполнения, так и в режиме получения обобщенных статистических характеристик их работы. **Метод.** При изложении предлагаемого подхода использован метод последовательного усложнения сетевых моделей с учетом особенностей существующих типовых протоколов и сетевых реализаций. Показана простота реализации предлагаемого подхода, которая базируется на имеющемся в составе фреймворка INET широком наборе уже готовых компонентов для описания различных элементов сети. Это позволяет, используя только графический редактор и встроенный язык описания сети NED, формировать достаточно сложные сетевые модели. Дано краткое описание основных компонентов фреймворка INET, позволяющих в режиме графического редактирования формировать структуру исследуемой сети. **Основные результаты.** Представлена методика построения имитационных моделей сетей с достаточно подробным описанием основных этапов моделирования. Рассмотрены простейшие приемы по запуску модели сети в работу и анализу полученных результатов. Показана возможность создания в проектах фреймворка INET пользовательских составных модулей, которые, являясь готовыми компонентами, могут использоваться в моделях более сложных сетевых структур. На базе сформированного подхода были построены модели как базовых, так и смешанных Ethernet топологий, что позволило провести достаточно подробный анализ каждой из этих топологий. **Практическая значимость.** Описанный инструментарий позволяет проводить исследования по обоснованию проектных решений построения распределенных компьютерных сетей, а также при создании средств поддержки автоматизированного проектирования инфокоммуникационных систем и сетей.

Ключевые слова
компьютерные сети, моделирование, топологии, Ethernet, протоколы, OMNeT++, INET Framework

MODELING OF ETHERNET NETWORKS IN OMNET ++ INET FRAMEWORK MEDIUM

S.P. Khabarov[a]

[a] St. Petersburg State Forest Technical University, Saint Petersburg, 194021, Russian Federation
Corresponding author: Serg.Habarov@mail.ru

Article info
Received 13.03.18, accepted 15.04.18
doi: 10.17586/2226-1494-2018-18-3-462-472
Article in Russian

For citation: Khabarov S.P. Modeling of Ethernet networks in OMNeT ++ INET framework medium. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2018, vol. 18, no. 3, pp. 462–472 (in Russian). doi: 10.17586/2226-1494-2018-18-3-462-472

Abstract

Subject of Research. The paper considers one of the possible approaches to the study of Ethernet networks by simulating them in the OMNeT ++ environment with the use of the INET framework. The possibility of studying network models is shown in both the step-by-step execution mode and the mode of obtaining generalized statistical characteristics of their operation. **Method.** When presenting the proposed approach, the method of sequential complication of network models was used, taking into account the features of existing standard protocols and network implementations. The simplicity of the proposed approach is shown, which is based on a wide range of ready-made components for the description of various network elements within the INET framework. This fact gives the possibility to form quite complex network models by

means of only the graphic editor and the built-in NED network description language. A brief description of INET framework main components is given that makes it possible to form the structure of the network under research in the graphical editing mode. **Main Result.** The paper presents technique for creation of network simulation models with a sufficiently detailed description of the main modeling stages. The simplest methods for launching the network model into work and analyzing the results are considered. The work shows the possibility of creating custom component modules in the INET framework projects, which, being ready-made components, can be used in models of more complex network structures. Based on the generated approach, models were built of both basic and mixed Ethernet topologies that gave the possibility to carry out a sufficiently detailed analysis for each of these topologies. **Practical Relevance**. The described toolkit makes it possible to carry out research on the justification of design solutions for design of distributed computer networks, as well as for creation of tools supporting the automated design of infocommunication systems and networks.

Keywords
computer networks, modeling, topologies, Ethernet, protocols, OMNeT++, INET Framework

Введение

Современные информационные системы различного прикладного назначения [1, 2] имеют распределенную структуру, используя в своем составе как различные вычислительные устройства [3], так и широкий набор протоколов [4–6]. При этом отдельные узлы этих систем связываются между собой различными каналами передачи данных, образуют достаточно сложную сетевую структуру. На начальных этапах проектирования особую роль играет моделирование таких систем для анализа качества их функционирования и обоснованности выбора соответствующего оборудования.

В настоящее время, наряду с хорошо разработанными методами аналитического моделирования [7–10], широко применяют и имитационное моделирование [11–14], которое позволяет проводить исследования и получать практически те же результаты, что и на реальном оборудовании. Помимо явной экономии, такой подход позволяет экспериментировать без построения реальной сети, что является достаточно трудоемким и длительным процессом. На сегодняшний день известно много систем моделирования, и есть из чего выбирать. К ним предъявляются достаточно требования – это детальная реализация протоколов всех уровней, возможность подключения собственных модулей, легкое и оперативное изменение параметров моделирования, платформенная независимость, развитый графический интерфейс, а также доступность продукта и его цена.

Исходя из отмеченных требований, наиболее популярными и востребованными в настоящее время средами имитационного моделирования являются The Network Simulator (NS-2, NS-3), OPNET Modeler Suite (OPNET), а также OMNeT++ (Objective Modular Network Testbed in C++). Наличие в среде OMNeT++ развитого графического интерфейса, как для построения моделей, так и для анализа полученных результатов, возможность работы на платформах с разными операционными системами, свободное распространение, а также достаточно подробная документация делают этот программный продукт наиболее привлекательным, особенно если речь идет об его использовании в учебном процессе.

Постановка задачи и объект исследования

В данной работе рассмотрен подход к построению и исследованию моделей сетей в среде OMNeT++ с использованием фреймворка INET [15, 16], который обладает широким набором уже готовых компонентов для описания различных элементов сети. Это позволяет, используя не язык C++, а только графический редактор и язык описания сети NED (Network Description), формировать сложные модели различного класса инфокоммуникационных сетей, и проводить их имитационное моделирование [17, 18]. На первом этапе ограничимся рассмотрением простейших Ethernet сетей как наиболее простых и хорошо знакомых, уделив основное внимание подходу к их описанию и моделированию в среде INET OMNeT++. Выбор этой среды для имитационного моделирования обусловлен тем, что фреймворк INET поддерживает широкий набор различных инфокоммуникационных технологий, протоколов и стандартов: IEEE 802.3, IEEE 802.11, IEEE 802.15.4, IPv4, IPv6, ARP, TCP, UDP, AODV и ряд других. В его состав включены модули визуализатора, которые могут отображать различные препятствия, маршруты движения, обнаруженные сетевые подключения и сетевые маршруты, текущие передачи и приемы, радиосигналы, статистику и многое другое. Этот фреймворк поддерживает 10 Мбит/с Ethernet, Fast Ethernet, Gigabit Ethernet, Fast Gigabit Ethernet и модели полного дуплекса. В состав его типовых компонентов входят модель уровня управления доступом к среде (EtherMAC), модель уровня управления логическим каналом (EtherLLC), модели концентратора (EtherHub) и коммутатора (EtherSwitch).

Как основу для моделей простейших Ethernet сетей можно использовать компонент EtherHost, который представляет собой пример узла сети с внедренным в него сетевым адаптером Ethernet (рис. 1). Компонент EtherHost не использует протоколы высокого уровня и напрямую генерирует трафик Ethernet. Он объединяет в себе компоненты EtherLLC и EtherMAC, а также простое клиент-серверное приложение. В нем EtherAppCli – простой генератор трафика, который периодически отправляет в канал сообщения типа EtherAppReq, длина которых может быть задана, его параметрами являются DestAddress, startTime, waitType, reqLength, respLength.

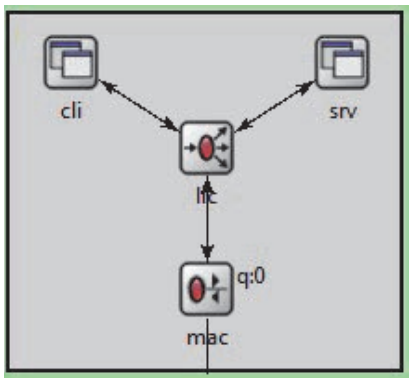

Рис. 1. Внутренняя структура INET-компонента EtherHost

Серверный компонент модели EtherAppSrv отвечает сообщением EtherAppResp запрашиваемой длины. Если ответ не вписывается в один фрагмент Ethernet, то клиент получает данные в нескольких фрагментах. Оба приложения собирают статистику в виде sentPkBytes, rcvdPkBytes, endToEndDelay и подключены непосредственно к EtherLLC.

Среда модулей MAC описывается интерфейсом модуля IEtherMAC. Каждый MAC-модуль имеет порты для подключения к физическому уровню (phys$i и phys$o), а также порты для подключения к верхнему уровню (upperLayerIn и upperLayerOut). Компоненты класса Channel используются для описания основных параметров и характеристик соединения узлов сети по физической среде передачи данных. В INET определены такие типы, как IdealChannel, DelayChannel, DatarateChannel и другие. Они отличаются набором параметров, которые определяют характер прохождения сообщений по каждому из них. В частности, DatarateChannel, как и DelayChannel, позволяет определить задержку распространения сообщения (Delay), но, в отличие от него, позволяет задать еще и ряд других параметров – например, скорость передачи данных по каналу (Datarate), которая должна быть указана в битах в секунду или кратных единицах (bps , kbps , Mbps , Gbps).

После установки среды OMNeT++ и фреймворка INET необходимо подготовить рабочее пространство (workspace) для формирования модели сети – создать новый проект и подключить к нему INET. Любой проект может содержать несколько папок, в каждой из них могут храниться несколько различных моделей сети.

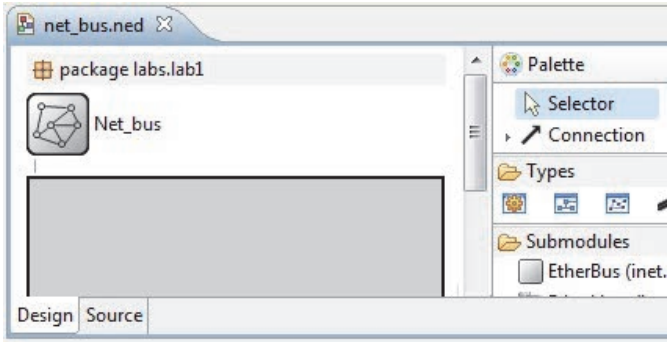

Рис. 2. Окно графического редактора NED-файлов с двумя вкладками

Первым этапом построения любой модели сети является создание файла ее описания (Network Description File, NED). Для этого следует выделить нужную папку, вызвать всплывающее меню и выбрать New → Network Description File (NED), в поле File name ввести произвольное имя сети и нажать Next. На следующем этапе формирования NED-файла имеет смысл выбрать опцию NED file with one item и нажать Finish. Откроется окно редактора NED-файлов с двумя вкладками (рис. 2), что позволяет создавать и редактировать файлы описания моделей сетей, используя либо графическую (Design), либо текстовую (Source) формы.

Модель Ethernet топологии типа «звезда»

Модель Ethernet топологии типа «звезда» может быть построена на базе компонента EtherHub, который описывает работу концентратора. В нем все сообщения, поступающие на входной порт, регенерируются и транслируются на все его другие порты. Сообщения могут быть любыми и определяются только своей длиной. Соединения, подключенные к концентратору, должны иметь одинаковую скорость передачи данных, а длина кабелей должна отражаться в задержках соединений. Модуль концентратора

собирает статистические данные: pkBytes – длину пакетов (вектор), messages/sec – количество пакетов в секунду (скаляр).

Для примера рассмотрим сеть, состоящую всего из четырех узлов, подключенных к одному концентратору. Создав NED-файл с именем, например net_star.ned, можно приступить к формированию новой модели сети. Для этого надо войти в режим Design и присвоить модели сети имя, например, net_hub. Затем во вкладке Submodules нужно найти компонент etherHub и перенести его в графическое поле модели сети. Аналогично следует поступить с компонентом etherHost, присвоив ему имя Host1.

Далее, используя компонент Channel, можно определить собственный тип каналов связи, который будет использован в разрабатываемой модели сети. С этой целью надо из вкладки Types компонент Channel переместить в окно редактора в область между пиктограммой модели сети (net_hub) и собственно схемой этой сети, после чего, войдя в окно его свойств, присвоить ему имя, например, HostToHub, и выбрать тип EtherLink. В каналах этого типа задержка определяется на основании длины подключающего кабеля.

После того как был сформирован собственный тип канала HostToHub, появляется возможность найти его во вкладке Palette, выделить мышкой и соединить с его помощью на графическом поле модели сети узлы Host1 и etherHub.

Для окончательного формирования модели следует одновременно выделить узел Host1 и канал связи, выполнить их копирование, а затем вставить еще три раза, изменяя имена узлов. Получиться модель, аналогичная представленной на рис. 3.

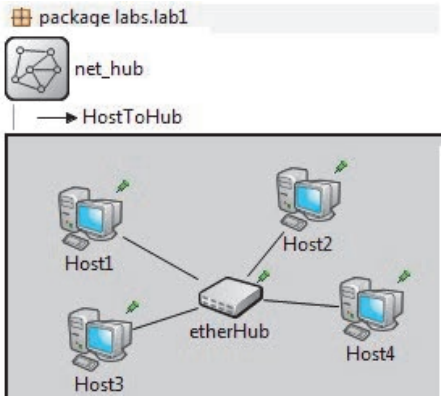

Рис. 3. Модель Ethernet сети на базе концентратора (net_star.ned)

По окончании графического формирования модели сети можно во вкладке NED Parameters познакомиться с набором параметров, которые полностью описывают текущее состояние как отдельных элементов, так и всей модели сети в целом. Сменив в редакторе NED-файлов вкладку Design на Source, можно ознакомиться с текстовым (исходным) кодом описания модели сети. Этот код среда моделирования генерирует автоматически на основе сформированной графической структуры.

В общем случае параметры отдельных компонентов и всей модели в целом могут получать свои значения несколькими способами: из кода NED, из файла начальной инициализации (*.ini) и даже в интерактивном режиме при запуске задачи. Для запуска модели сети необходим файл начальной инициализации. Он должен сообщить системе, какую сеть надо моделировать, какие параметры следует передать в модель сети, а также, если требуется, определить режимы работы генераторов случайных чисел. Исходя из этого, следующим шагом на пути реализации модели сети должно являться создание ini-файла, например, omnetpp_star.ini, один из вариантов которого может иметь следующий вид:

```
[General]
  sim-time-limit = 100s
  **.vector-recording = false
[Config Star_Hub_1]
  # Пример сети из 4-х ПК, подключенных к концентратору
  network = net_hub
  **.channel.datarate = 100Mbps
  **.Host1.cli.destAddress = ""
  **.cli.destAddress = "Host1"
  # Равномерное распределение в интервале от 0.95 до 1.1 секунд
  **.cli.sendInterval = uniform(0.95s,1.1s)
```

Запуск задачи на выполнение позволит исследовать работу этой Ethernet топологии при случайном потоке запросов от узлов Host2, Host3 и Host4 на узел Host1 и получении ответов от этого узла. Изменяя параметры закона распределения времени между запросами, можно выяснить, как меняется общая за-

грузка этого сегмента сети, работающего на каналах 100-мегабитного Ethernet. Особый интерес будет представлять случай, когда периодичность запросов является практически детерминированной в течение всех 100 с модельного времени. Учесть этот аспект в модели сети можно путем добавления в файл начальной инициализации omnetpp_star.ini еще одной секции:

```
[Config Star_Hub_2]
  network = net_hub
  **.channel.datarate = 100Mbps
  **.Host1.cli.destAddress = ""
  **.cli.destAddress = "Host1"
  **.cli.sendInterval = 1s
```

При запуске этого файла на выполнение появляется возможность выбора одного из двух описанных в нем режимов моделирования. При выборе второго режима можно убедиться в возникновении коллизий в концентраторе (рис. 4), а стало быть, и в зависании этой Ethernet топологии.

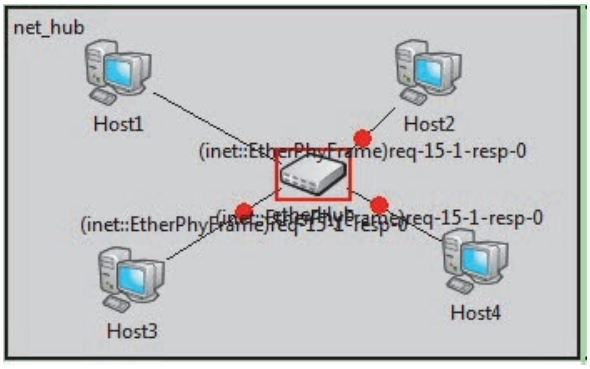

Рис. 4. Коллизии в сети на базе концентратора

В частности, имеет смысл обратить внимание на тот факт, что за 100 с прогона модели клиентским приложением Host2 (net_hub.Host2.cli) было послано 100 пакетов (sentPk:count) общим объемом 10000 Б (sentPk:count), а серверное приложение этого хоста (net_hub.Host2.svr) ничего не получило в ответ. При этом видно, что на MAC-уровне этого хоста было зарегистрировано 99 коллизий (рис. 5).

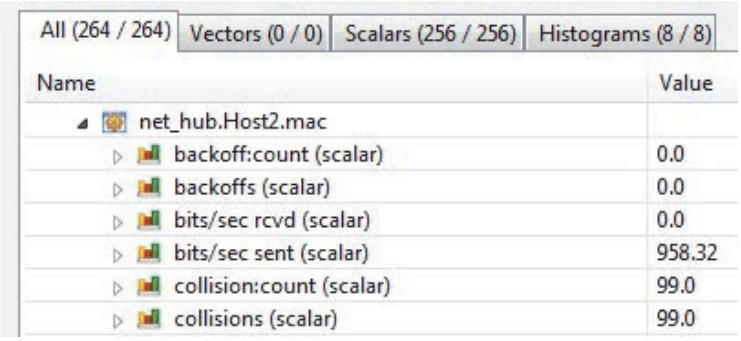

Рис. 5. Статистические данные результатов прогона модели

Для более подробного исследования Ethernet топологии типа «звезда» на базе концентратора можно провести еще ряд прогонов модели сети с измененными параметрами, в частности, по скорости работы канала, длине передаваемых сообщений и т.д.

Коммутаторы, в отличие от концентраторов, работающих на физическом уровне, работают на уровне канала передачи данных и маршрутизируют кадры данных между подключенными соединениями. В то время как концентратор повторяет кадры данных на каждой подключенной линии, возможно, вызывая столкновения, коммутаторы помогают сегментировать сеть. В современных гигабитных локальных сетях каждый узел подключается к коммутатору с помощью дуплексных линий, поэтому никаких столкновений не происходит. Используя среду OMNeT++, можно достаточно просто провести сравнительный анализ этой топологии с рассмотренной ранее. Для этого добавим в имеющуюся модель сегмент Ethernet на базе коммутатора, используя определенный во фреймворке INET компонент EtherSwitch. Для построения новой модели наиболее целесообразной представляется следующая последовательность действий.

– Находясь в режиме Design редактора файла net_star.ned, выделить всю область сети net_hub, скопировать ее, а затем скопированные данные вставить в графическом редакторе чуть ниже области сети net_hub. Новой сети присвоить имя, например, net_switch.

– Выделить компонент etherHub модели net_switch, перейти в окно его свойств и изменить его имя на Switch, а тип на EtherSwitch (inet.node.ethernet).

Для запуска в работу новой модели с параметрами, аналогичными предыдущему примеру, в файл omnetpp_star.ini следует добавить новую секцию, аналогичную секции [Config Star_Hub_2], где параметру network должно быть присвоено имя новой сети:

```
[Config Star_Switch]
# Пример сети из 4-х ПК, подключенных к коммутатору
  network = net_switch
  **.channel.datarate = 100Mbps
  **.Host1.cli.destAddress = ""
  **.cli.destAddress = "Host1"
  **.cli.sendInterval = 1s
```

Запустив модель, можно отслеживать работу как сети в целом, так и каждого узла, компонента и даже отдельного элемента или устройства (рис. 6). Так, например, можно исследовать работу коммутатора, который в этом примере имеет 4 порта для подключения Ethernet каналов eth[i], коммутационное устройство (relayUnit), таблицу MAC-маршрутизации и таблицу интерфейсов.

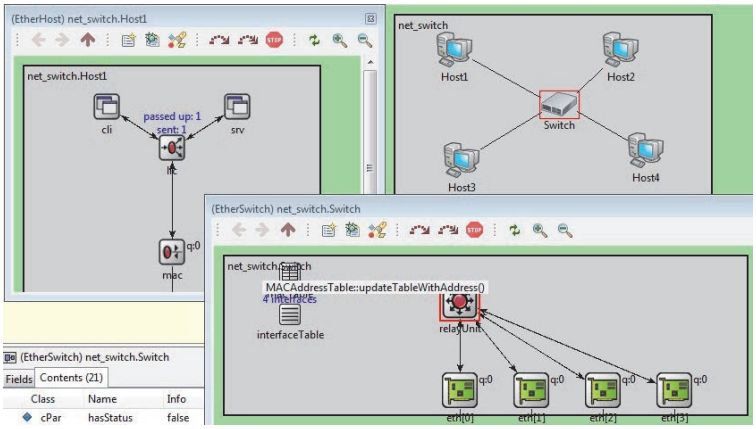

Рис. 6. Окна моделей узла Host1, коммутатора и сети в целом

Следует отметить, что в графической среде выполнения некоторые эффекты анимации позволяют лучше следить за процессом моделирования. Так, при передаче пакетов цвет канала становится желтым, а при обнаружении коллизий – красным. При отключении сетевого адаптера пиктограмма модуля MAC становится неактивной. Пиктограммы сетевых адаптеров окрашиваются в соответствии с состоянием MAC-модуля: желтый – при передаче, синий – при приеме, красный – при обнаружении коллизий, белый характеризует состояние отсрочки, а серый – состояние паузы.

Закончив прогон модели в течение 100 с модельного времени, можно провести анализ статистических данных этого прогона и сравнить их с результатами предыдущего примера, откуда видно, что Ethernet топология на базе коммутатора при тех же исходных параметрах работы сети полностью исключает коллизии (рис. 7).

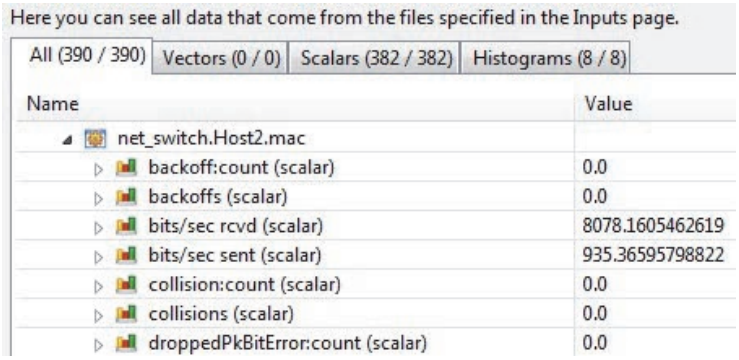

Рис. 7. Статистические данные работы узла Host2 в процессе прогона модели

Добиться этого результата позволяет коммутатор, структурная организация которого принципиально отличается от концентраторов. Запустив модель в режиме пошагового выполнения, можно познакомиться с тем, как на первых шагах работы сети идет процесс заполнения изначально пустой таблицы

MAC-маршрутизации коммутатора и как она используется при дальнейшей работе коммутатора в составе модели сети (рис. 8).

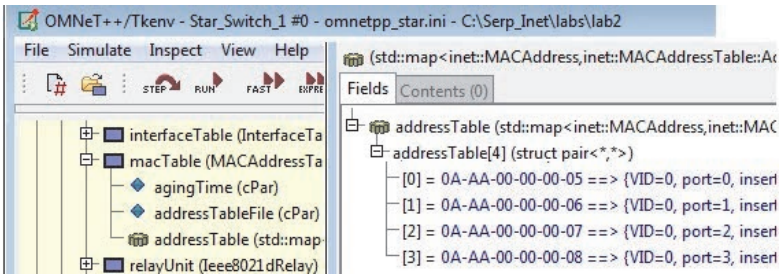

Рис. 8. Процесс заполнения таблицы MAC-маршрутизации коммутатора

Модель сети смешанной Ethernet топологии

В общем случае редко можно встретить случай, когда какая-либо, даже простейшая, локальная вычислительная сеть (ЛВС) описывалась бы какой-то одной топологией. Как правило, в любой промышленный или офисной ЛВС присутствует сегменты всех Ethernet топологий. При этом может использоваться произвольный набор коммутационных устройств: общая шина на базе коаксиального кабеля, широковещательные концентраторы и различные коммутаторы.

Подход к построению таких моделей будет рассмотрен на примере ЛВС, которая имеет два сегмента, подключенных к общей шине. К этой же шине подключен еще один компьютер и сетевой принтер. Первый сегмент – это компьютеры, связанные каналами связи 100 Мбит/с с коммутатором. Второй сегмент организован на концентраторе, к которому подключены компьютеры, один из которых является сервером данной ЛВС.

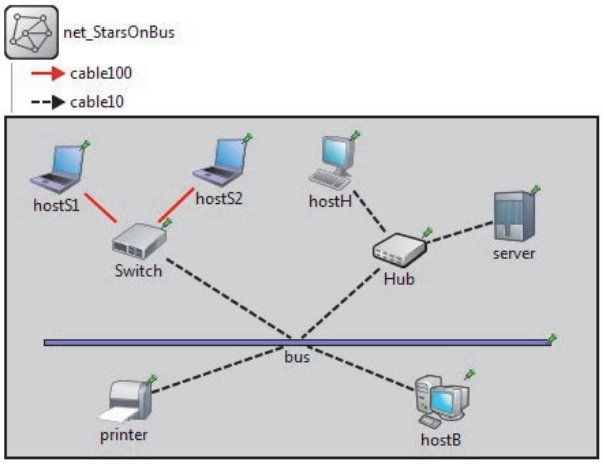

Рис. 9. Пример модели сети смешанной Ethernet топологии

Общая шина работает на скорости 10 Мбит/с, а к ней подключен Hub второго сегмента. Он, как и любой концентратор, может работать только с соединениями, у которых скорость передачи одинакова на всех портах. В связи с этим второй сегмент может быть построен только на 10 Мбит/с каналах связи. Учитывая это, для описания каналов связи потребуется использовать два разных компонента: DatarateChannel и Eth100M. Допуская, что все оконечные узлы сети (сервер, компьютеры (ПК) и принтер) описываются компонентом etherHost, модель сети будет иметь вид, представленный на рис. 9. Для описания этой сети можно использовать тот же файл net_star.ned, добавив в него еще одну, уже третью, сеть с именем, например, net_StarOnBus. Оставляя в стороне статистический анализ модели сети, рассмотрим только один детерминированный пример, иллюстрирующий ее работу.

Пусть требуется регулярно, например, один раз в секунду, с компьютера hostS1 посылать к серверу ЛВС запрос на получения нужной информации. Например, это стандартный запрос к базе данных, который имеет длину 160 Б. Сервер, выполнив поиск в базе данных, возвращает узлу, например, 2048 Б информации. Затраты времени на работу сервера не учитываются. Исходя из этих посылок, в файл начальной инициализации работы модели сети omnetpp_star.ini надо добавить еще одну секцию, которая может иметь следующий вид:

```
[Config Star_On_Bus]
  # Две звезды, ПК и принтер на одной шине
  network = net_StarsOnBus
```

```
**.channel.datarate = 10Mbps
**.hostS1.cli.destAddress = "server"
**.hostS1.cli.sendInterval = 1s
**.cli.reqLength = 160B
**.cli.respLength = 2KiB
```

Запустив этот файл на выполнение, имеет смысл выполнить пошаговую трассировку, по крайней мере, первых трех/четырех обращений клиента к серверу, чтобы познакомиться с характером работы модели и последовательностью перемещения пакетов как по отдельным узлам сети, так и между разными сегментами этой сети (рис. 10).

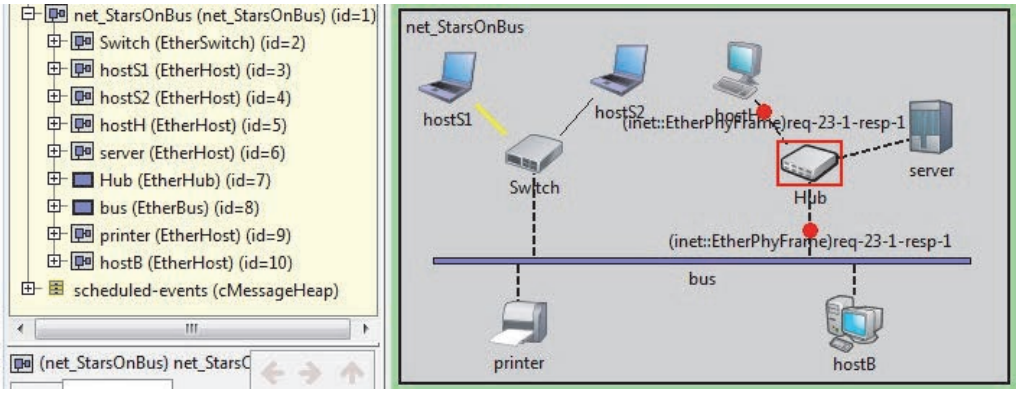

Рис. 10. Процесс функционирования модели сети смешанной топологии

Особо следует обратить внимание на результат трассировки. Он в общих чертах иллюстрирует работу сети по выполнению конкретного запроса к серверу, который начинается с поиска пути до конкретного адресата.

Event#	Time	Src/Dest	Name	Info
#10	1	hostS1 --> Switch	req-23-1	inet::EtherAppReq:160 bytes
#15	1.000015169999	Switch --> hostS2	req-23-1	inet::EtherAppReq:160 bytes
#16	1.000015169999	Switch --> bus	req-23-1	inet::EtherAppReq:160 bytes
#18	1.000015194999	bus --> Hub	req-23-1	inet::EtherAppReq:160 bytes
#19	1.000015194999	Hub --> hostH	req-23-1	inet::EtherAppReq:160 bytes
#19	1.000015194999	Hub --> server	req-23-1	inet::EtherAppReq:160 bytes
#23	1.000015219999	bus --> printer	req-23-1	inet::EtherAppReq:160 bytes
#25	1.000015244999	bus --> hostB	req-23-1	inet::EtherAppReq:160 bytes
#46	1.000175994999	server --> Hub	req-23-1-resp-0	inet::EtherAppResp:1497 bytes
#47	1.000175994999	Hub --> hostH	req-23-1-resp-0	inet::EtherAppResp:1497 bytes
#47	1.000175994999	Hub --> bus	req-23-1-resp-0	inet::EtherAppResp:1497 bytes
#51	1.000176019999	bus --> Switch	req-23-1-resp-0	inet::EtherAppResp:1497 bytes

Рис. 11. Начальный этап трассировки работы модели сети

Налицо разница между первым и последующими запросами, когда коммутатор уже заполнит свою таблицу MAC-адресации. Кроме этого, работая с Ethernet топологиями, надо точно представлять, что размер фрейма не может быть более 1500 Б. Именно поэтому, как видно из результатов трассировки (рис. 11), ответ сервера разбит на два фрейма длиной 1497 Б и 551 Б, что, естественно, будет увеличивать накладные расходы на сеть.

Использование составных модулей в моделях Ethernet сетей

Рассматривая подходы к построению моделей сетей и их исследованию методом имитационного моделирования, необходимо отметить достаточно большие возможности фреймворка INET OMNeT++ для построения моделей сложных Ethernet топологий. Такие модели могут содержать не только типовые компоненты, но и разработанные пользователем модули. Для иллюстрации этой возможности рассмотрим достаточно простой и не очень строгий пример модели сети, в которой к общей шине подключено N сегментов на базе коммутаторов, а каждый из сегментов содержит в своем составе K узлов. Для реализации поставленной задачи надо выполнить следующую последовательность действий.

Создать новый ned-файл, из вкладки Types перетащить в графический редактор новый составной модуль (Compound Module) и присвоить ему имя, например, SwitchStar. На сформированный составной модуль вставить компоненты EtherHost, EtherSwitch и связать их каналом Eth100M. При этом компонент EtherHost определить как векторный и задать для него новое имя, например, pc (рис. 12).

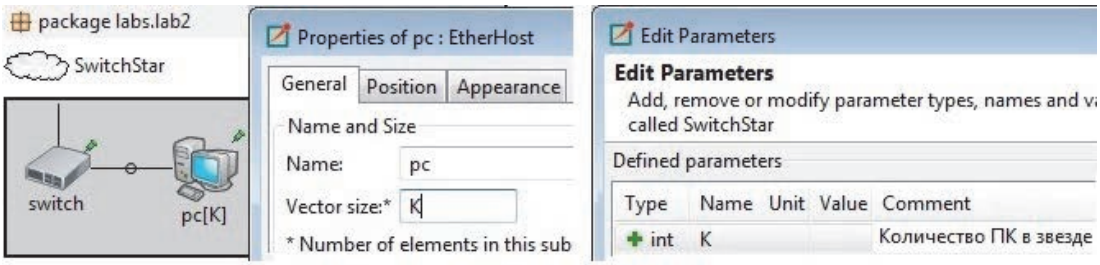

Рис. 12. Задание свойств EtherHost и добавление параметра для числа узлов в сегменте

Затем, выделив весь составной модуль, вызвать всплывающее меню, перейти в режим Parameters и в описание модуля добавить целочисленный параметр с именем *K*, который будет определять количество узлов в сегменте. Закончив формирование собственного составного модуля в графическом режиме, надо в редакторе NED-файла перейти в режим Source и несколько изменить автоматически сформированный код описания сети:

```
module SwitchStar {
  @display("bgb=350,100;i=old/cloud");
  int K;              // Количество ПК в звезде
  gates:
    inout ethg;
  submodules:
    switch: EtherSwitch { @display("p=68,50");  }
    pc[K]: EtherHost    { @display("p=268,50"); }
  connections:
    for i=0..K-1 { switch.ethg++ <--> Eth100M <--> pc[i].ethg; }
    switch.ethg++ <--> ethg;
}
```

После формирования пользовательского составного модуля SwitchStar можно перейти к созданию новой сети, присвоив ей имя, например, StarsOnBus. В этой сети вновь добавленные компоненты EtherHost и SwitchStar подключаются к EtherBus с помощью канала DatarateChannel. Для примера можно пока ограничиться только одной «звездой» с *K* узлами (рис. 13).

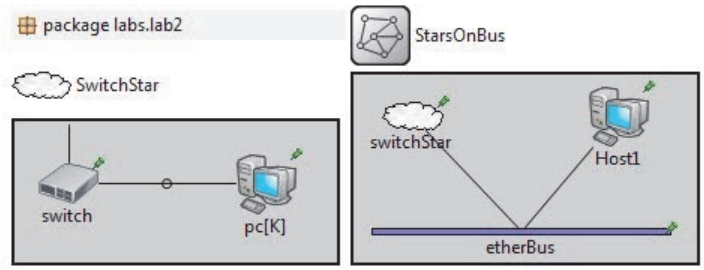

Рис. 13. Модель сети с пользовательским составным модулем

Перейдя в режим Source, следует проверить, что гейт (порт) модуля SwitchStar подключен к гейту (порту) шины etherBus, как и гейт (порт) компонента Host1.

```
connections:
  Host1.ethg <--> DatarateChannel <--> etherBus.ethg++;
  switchStar.ethg <--> DatarateChannel <--> etherBus.ethg++;
```

Затем для описания параметров режима имитационного моделирования надо сформировать файл начальной инициализации, один из вариантов которого может иметь следующий вид:

```
[General]
  network = StarsOnBus
  **.channel.datarate = 10Mbps
  **.Host1.cli.destAddress = ""
  **.cli.destAddress = "Host1"
  *.switchStar.pc[0].cli.reqLength = 333B
  *.switchStar.pc[1].cli.reqLength = 666B
```

При запуске ini-файла в работу появится запрос на задание количества узлов, входящих в состав сегмента на базе коммутатора. Это дает возможность использовать одну и ту же модель для моделирования структур различной сложности и исследовать загрузку сети в зависимости от количества подключенных к ней узлов. Данная модель позволяет не только исследовать обобщенные статистические харак-

теристики работы основной сети, но и получать информацию о работе и загрузке любого из элементов этой сети (рис. 14).

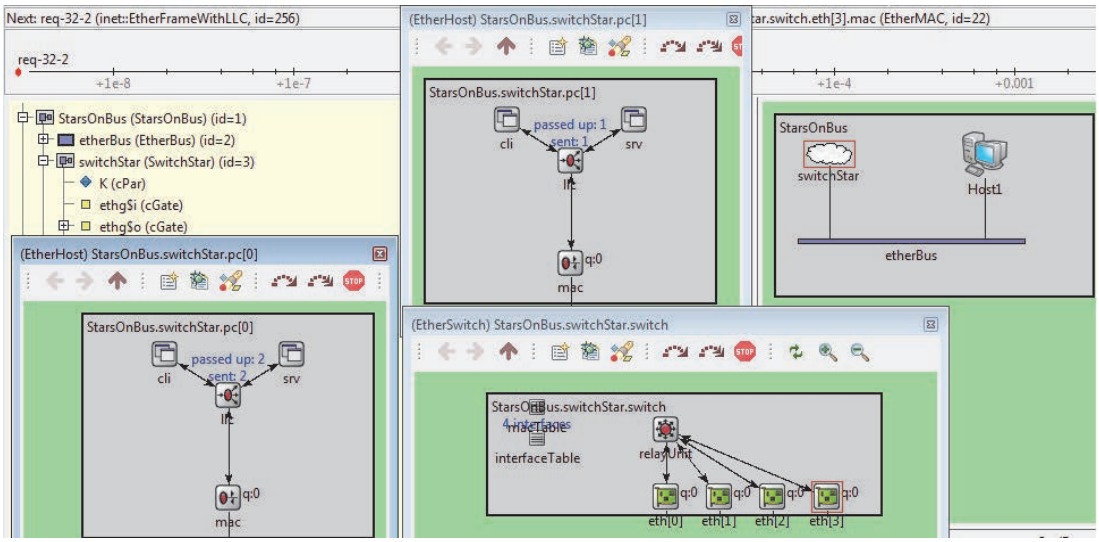

Рис. 14. Структура основных компонентов исследуемой сети

Анализ полученных результатов позволит определить наиболее загруженные сегменты сети, а также те ее элементы, которые оказывают особое влияние на производительность работы сети в целом. Это позволяет наметить пути для оптимизации структуры сети или изменить информационную нагрузку на каждый из ее узлов.

Заключение

Проведенные исследования позволили сформировать один из возможных подходов к имитационному моделированию сетей в среде фреймворка INET OMNeT++ и провести анализ ряда сетевых топологий. Показана простота его реализации, которая базируется на наличии в среде фреймворка INET широкого набора уже готовых компонентов для описания различных элементов сети, что позволяет, используя только графический редактор и встроенный язык описания сети NED, формировать достаточно сложные сетевые модели.

На базе сформированного подхода были построены модели как базовых, так и смешанных Ethernet топологий, что позволило провести достаточно подробный анализ каждой из этих топологий, выяснить сложности, возникающие в сегментах Ethernet топологий типа звезда на базе концентраторов при детерминированных потоках заявок, и убедиться в том, что при тех же законах распределения времени между запросами к узлам сети топология на базе коммутаторов свободна от возникновения коллизий.

Показана возможность создания в проектах фреймворка INET пользовательских составных модулей, которые, являясь готовыми и отлаженными компонентами, могут использоваться в моделях более сложных сетевых структур, что позволяет выполнять декомпозицию общей сети и отдельно исследовать ее фрагменты.

Использование фреймворка INET OMNeT++ и предложенного подхода дает возможность достаточно просто выполнять моделирование не только Ethernet сетей, но и различных беспроводных сетей [19], позволяя исследовать как различные структуры самоорганизующихся Ad Hoc сетей, так и различные варианты построения сенсорных сетей.

Литература

1. Амбросовский В.М., Баглюк Ю.В., Слипченко А.С., Хабаров С.П. Интегрированные системы управления техническими средствами корабля // Морской вестник. 2014. № 2 (50). С. 63–65.
2. Хабаров С.П., Заяц А.М. Использование технологии websocket в клиент-серверных экспертных системах // В кн.: Леса России: политика, промышленность, наука, образование. 2017. С. 278–280.
3. Хабаров С.П. Организация гетерогенных ЛВС с терминальным доступом между ее узлами // Известия СПбЛТА. 2016. № 216. С. 267–280. doi: 10.21266/2079-4304.2016.216.267-280
4. Хабаров С.П. Взаимодействие узлов сети по протоколу WebSocket // В сб. Информационные системы и технологии:

References

1. Ambrosovskii V.M., Baglyuk Yu.V., Slipchenko A.S., Khabarov S.P. Integrated control systems for ship technical means. *Morskoi Vestnik*, 2014, no. 2, pp. 63–65. (in Russian)
2. Khabarov S.P., Zayats A.M. Using websocket technology in client-server expert systems. In *Forests of Russia: Politics, Industry, Science, Education*, 2017, pp. 278–280. (in Russian)
3. Khabarov S.P. Organization of heterogeneous LAN with terminal access between its nodes. *Izvestia SPbFTU*, 2016, no. 216, pp. 267–280. (in Russian) doi: 10.21266/2079-4304.2016.216.267-280
4. Khabarov S.P. Interaction of network nodes using the WebSocket protocol. In *Information Systems and Technologies: Theory and Practice*, 2017, no. 9, pp. 109–119. (in Russian)

теория и практика. 2017. № 9. С. 109–119.
5. Хабаров С.П. Использование утилиты websocketd для удаленного выполнения программ // В сб. Информационные системы и технологии: теория и практика. 2017. № 9. С. 90–104.
6. Хабаров С.П., Колмогорцев Е.Л. Отказоустойчивый протокол надежной доставки данных // Известия СПбЛТА. 2005. № 174. С. 143–153.
7. Богатырев В.А., Богатырев С.В. Резервированное обслуживание в кластерах с уничтожением неактуальных запросов // Вестник компьютерных и информационных технологий. 2017. № 1. С. 21–28. doi: 10.14489/vkit.2017.01.pp.021-028
8. Bogatyrev V.A., Vinokurova M.S. Control and safety of operation of duplicated computer systems // Communications in Computer and Information Science. 2017. V. 700. P. 331–342. doi: 10.1007/978-3-319-66836-9_28
9. Arustamov S.A., Bogatyrev V.A., Polyakov V.I. Back up data transmission in real-time duplicated computer systems // Advances in Intelligent Systems and Computing. 2016. V. 451. P. 103–109. doi: 10.1007/978-3-319-33816-3_11
10. Богатырев В.А., Кармановский Н.С., Попцова Н.А., Паршутина С.А., Воронина Д.А., Богатырев С.В. Имитационная модель поддержки проектирования инфокоммуникационных резервированных систем // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 5(105). С. 831–838. doi: 10.17586/2226-1494-2016-16-5-831-838
11. Богатырев В.А., Богатырев С.В. Надежность мультикластерных систем с перераспределением потоков запросов // Изв. вузов. Приборостроение. 2017. Т. 60. № 2. С. 171–177. doi: 10.17586/0021-3454-2017-60-2-171-177
12. Bogatyrev V.A., Parshutina S.A. Redundant distribution of requests through the network by transferring them over multiple paths // Communications in Computer and Information Science. 2016. V. 601. P. 199–207. doi: 10.1007/978-3-319-30843-2_21
13. Parshutina S.A., Bogatyrev V.A. Models to support design of highly reliable distributed computer systems with redundant processes of data transmission and handling // Proc. Int. Conf. on Quality Management, Transport and Information Security, Information Technologies. St. Petersburg, Russia, 2017. P. 96–99. doi: 10.1109/ITMQIS.2017.8085772
14. Bogatyrev V.A., Parshutina S.A., Poptcova N.A., Bogatyrev A.V. Efficiency of redundant service with destruction of expired and irrelevant request copies in real-time clusters // Communications in Computer and Information Science. 2016. V. 678. P. 337–348. doi: 10.1007/978-3-319-51917-3_30
15. OMNeT++. Discrete Event Simulator [Электронный ресурс]. URL: https://omnetpp.org/ (дата обращения: 10.03.2018).
16. INET Framework for OMNeT++. Manual [Электронный ресурс]. 2016. URL: https://omnetpp.org/doc/inet/api-current/inet-manual-draft.pdf (дата обращения: 10.03.2018).
17. Котенко И.В., Уланов А.В. Многоагентное моделирование распределенных атак «отказ в обслуживании» и механизмов защиты от них // Труды СПИИРАН. 2006. Т. 1. № 3. С. 105–125.
18. Ткачёв М.С. Моделирование распределения потоков трафика с учетом перегрузок каналов сети MPLS-TE DIFFServ // Вестник КРСУ. 2016. Т. 16. № 1. С. 110–112.
19. Думов М.И., Хабаров С.П. Использование OMNET++ для моделирования беспроводных Wi-Fi сетей // В сб. Информационные системы и технологии: теория и практика. 2018. № 10. С. 40–48.

Авторы

Хабаров Сергей Петрович – кандидат технических наук, доцент, доцент, Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, Санкт-Петербург, 194021, Российская Федерация, ORCID ID: 0000-0003-1337-0150, Serg.Habarov@mail.ru

5. Khabarov S.P. Using websocketd utility for remote program execution. In *Information Systems and Technologies: Theory and Practice*, 2017, no. 9, pp. 90–104. (in Russian)
6. Khabarov S.P., Kolmogortsev E.L. Failover secure data delivery protocol. *Izvestia SPbFTU*, 2005, no. 174, pp. 143–153. (in Russian)
7. Bogatyrev V.A., Bogatyrev S.V. Redundant service clusters with the destruction of irrelevant queries. *Herald of Computer and Information Technologies*, 2017, no. 1, pp. 21–28. (In Russian) doi: 10.14489/vkit.2017.01.pp.021-028
8. Bogatyrev V.A., Vinokurova M.S. Control and safety of operation of duplicated computer systems. *Communications in Computer and Information Science*, 2017, vol. 700, pp. 331–342. doi: 10.1007/978-3-319-66836-9_28
9. Arustamov S.A., Bogatyrev V.A., Polyakov V.I. Back up data transmission in real-time duplicated computer systems. *Advances in Intelligent Systems and Computing*, 2016, vol. 451, pp. 103–109. doi: 10.1007/978-3-319-33816-3_11
10. Bogatyrev V.A., Karmanovsky N.S., Poptcova N.A., Parshutin S.A., Voronina D.A., Bogatyrev S.V. Simulation model for design support of infocomm redundant systems. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 5, pp. 831–838. (In Russian) doi: 10.17586/2226-1494-2016-16-5-831-838
11. Bogatyrev V.A., Bogatyrev S.V. Reliability of multi-cluster systems with redistribution of the flow of requests. *Journal of Instrument Engineering*, 2017, vol. 60, no. 2, pp. 171–177. (In Russian) doi: 10.17586/0021-3454-2017-60-2-171-177
12. Bogatyrev V.A., Parshutina S.A. Redundant distribution of requests through the network by transferring them over multiple paths. *Communications in Computer and Information Science*, 2016, vol. 601, pp. 199–207. doi: 10.1007/978-3-319-30843-2_21
13. Parshutina S.A., Bogatyrev V.A. Models to support design of highly reliable distributed computer systems with redundant processes of data transmission and handling. *Proc. Int. Conf. on Quality Management, Transport and Information Security, Information Technologies*. St. Petersburg, Russia, 2017, pp. 96–99. doi: 10.1109/ITMQIS.2017.8085772
14. Bogatyrev V.A., Parshutina S.A., Poptcova N.A., Bogatyrev A.V. Efficiency of redundant service with destruction of expired and irrelevant request copies in real-time clusters. *Communications in Computer and Information Science*, 2016, vol. 678, pp. 337–348. doi: 10.1007/978-3-319-51917-3_30
15. *OMNeT++. Discrete Event Simulator*. URL: https://omnetpp.org/ (accessed: 10.03.2018).
16. *INET Framework for OMNeT++. Manual*. 2016. URL: https://omnetpp.org/doc/inet/api-current/inet-manual-draft.pdf (accessed: 10.03.2018).
17. Kotenko I.V., Ulanov A.V. Multi-agent modeling of distributed "Denial of Service" attacks and defense mechanisms. *SPIIRAS Proceedings*, 2016, vol. 1, no. 3, pp. 105–125. (in Russian)
18. Tkachev M.S. Simulation of distribution the traffic flows with account overload channels of MPLS-TE DIFFServ. *Vestnik KRSU*, 2016, vol. 16, no. 1, pp. 110–112. (in Russian)
19. Dumov M.I., Khabarov S.P. Using OMNET++ for wireless Wi-Fi networks simulation. In *Information Systems and Technologies: Theory and Practice*, 2018, no. 10, pp. 40–48. (in Russian)

Authors

Sergey P. Khabarov – PhD, Associate Professor, Associate Professor, St. Petersburg State Forest Technical University, Saint Petersburg, 194021, Russian Federation, ORCID ID: 0000-0003-1337-0150, Serg.Habarov@mail.ru