РЕАЛИЗАЦИЯ СИМУЛЯТОРА ИНФРАСТРУКТУРЫ ДЛЯ МНОГОАДРЕСНОЙ ШИРОКОПОЛОСНОЙ ПЕРЕДАЧИ МУЛЬТИМЕДИЙНОГО ТРАФИКА НА БАЗЕ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

Ушаков Ю.А., Полежаев П.Н., Шухман А.Е., Бахарева Н.Ф.
Оренбургский государственный университет, г. Оренбург

1 Введение

Сегодня IPTV является главным конкурентом обычного и цифрового телевидения. IPTV работает по принципу многоадресной передачи – распространения одинакового контента для ограниченного числа абонентских устройств, подключенных к одной и той же группе вещания (каналу).

Основные трудности развертывания IPTV связаны с маршрутизацией широкополосного мультимедийного трафика, которая в современных сетях осуществляется с помощью протоколов IGMP и PIM. Основными недостатками данных протоколов является сложность настройки, дороговизна устройств, которые их поддерживают, и низкая производительность на коммутаторах уровня доступа.

2 Реализация симулятора

Инфраструктура передачи широкополосного мультимедийного трафика может быть представлена в виде ориентированного графа, вершинами которого являются сервера, коммутаторы и маршрутизаторы, а дугами – сетевые связи между ними. Совокупность всех маршрутов передачи данных от вещателя к абонентам группы образует дерево вещания. Формирование оптимального дерева вещания может быть формализовано в виде решения NP-полной проблемы Штейнера для ориентированного графа сетевой инфраструктуры.

Для изучения работы потоков IPTV в сетях второго и третьего уровня, сравнения с традиционными подходами IGMP и PIM, был создан симулятор, описанный в работе [1]. Симулятор построен на базе системы OMNET++ с открытым исходным кодом [2] и возможностью использовать не просто симулируемые структуры, а интегрироваться с эмуляторами и реальными контроллерами (через PCAP интерфейс сетевой карты).

Интерфейс симулятора выполнен на базе IDE Eclipse с интеграцией с OMNET++ (рисунок 1).

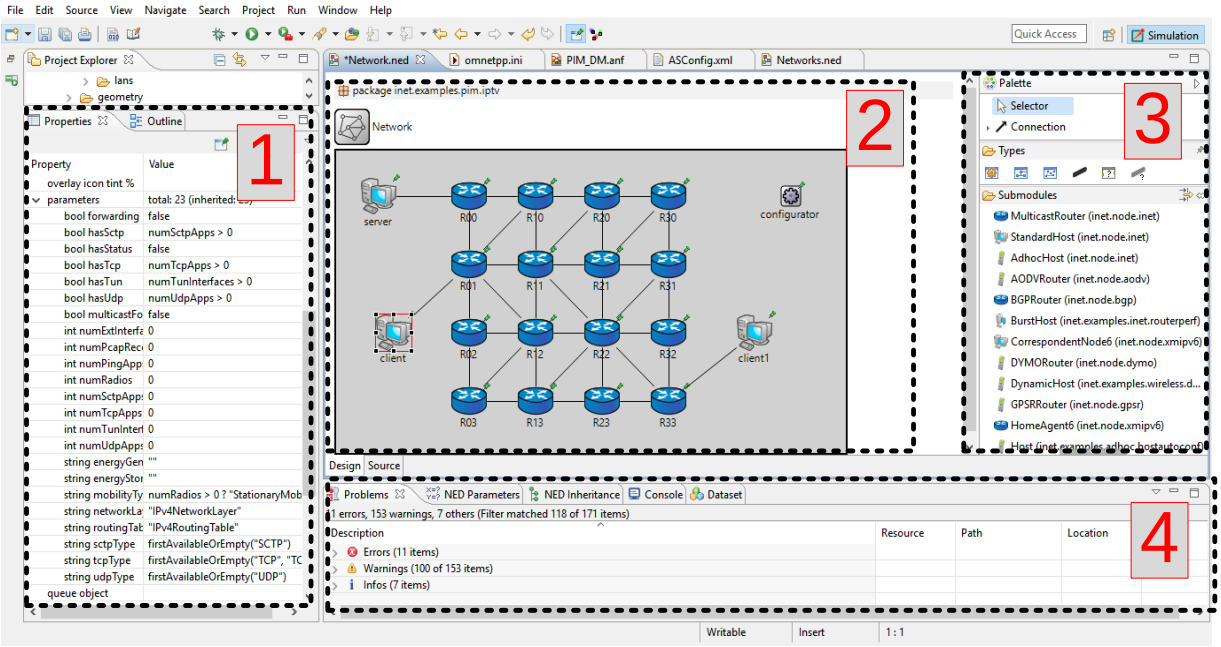

Рисунок 1 – Окно симулятора

Зона 1 предназначена для настройки свойств объектов, например, скорости линии. Зона 2 предназначена для визуального конструирования сети, зона 3 – для добавления новых устройств или связей. В зоне 4 выводятся ошибки и предупреждения от симулятора.

На рисунке 2 показана сеть, на которой исследовались алгоритмы и протоколы многоадресной сети. Топология сети подобрана таким образом, чтобы на ее основе можно было путем отключения интерфейсов симулировать любые топологии, встречающиеся у провайдеров IPTV – звезда, кольцо, кольцо с деревом, вложенные кольца и прочие.

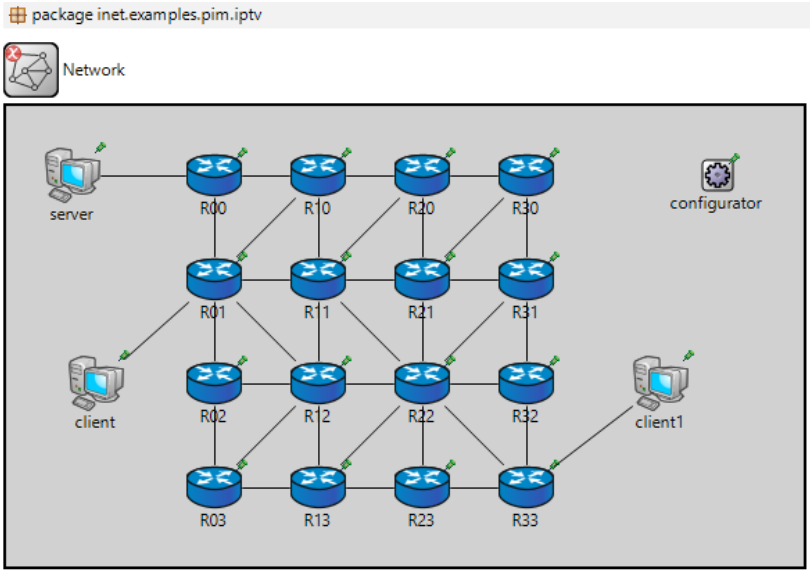

Рисунок 2 – Прототип сети для исследования работы многоадресной рассылки

Также для исследования простых действий каждого протокола был создан модуль визуализации пакетов на основе модуля симуляции OMNET++, дополненный средствами глубокого анализа пакетов, показанный на рисунке 3.

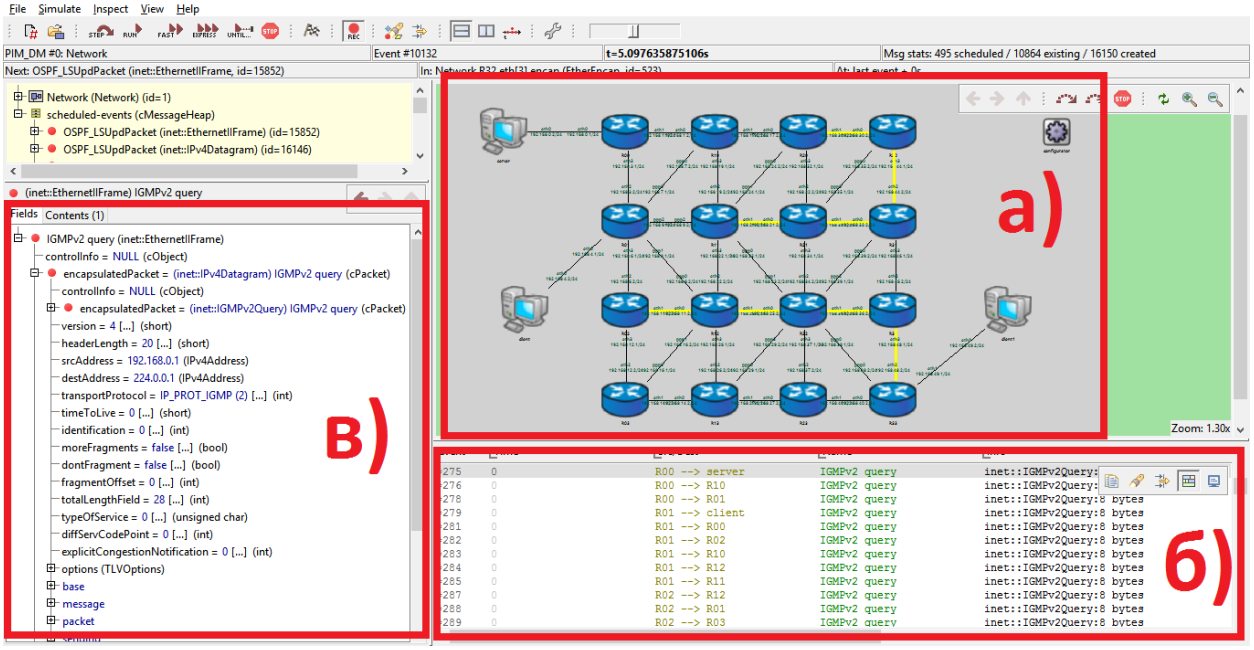

Рисунок 3 – Модуль визуализации пакетов: а) визуализация движения пакетов и нагрузки на линии; б) список пакетов в системе (с фильтром); в) детализация выбранного пакета.

3 Настройка схемы сети симулятора

На схеме отображаются все IP адреса маршрутизаторов и клиентов, пакеты и их содержимое. Для задания параметров используется INI файл и переменные проекта, что делает возможным автоматизацию серии экспериментов с разными входными параметрами. Для задания интенсивности и закона распределения генерации трафика используются параметры приложений, которых может быть несколько на каждом конечном устройстве. Для задания длины пакета используется конструкция:

***.server.udpApp[0].messageLength = uniform(500B, 1400B)*

которая задает на сервере длину UDP пакета случайным образом, распределенным по равномерному распределению в интервале 500-1400 байт. Для задания наиважнейшей характеристики потока – интенсивности и распределения времени между пакетами по экспоненцяальному закону распределения используется конструкция:

***.server.udpApp[0].sendInterval = exponential(30us)*

Для задания начала и конца генерации, адреса, порта, типа приложений используется следующие строки

***.server.udpApp[0].destPort = 5000*

***.server.udpApp[0].destAddresses = "239.0.0.11"*
***.server.udpApp[0].startTime = 20s*
***.server.udpApp[0].stopTime = 100s*

Адрес назначения должен быть такой же, как и на клиентских приложениях. Начало генерации через 20 секунд сделано для исключения начального влияния процессов обмена таблицами маршрутизации на конечный результат [3].

Для того, чтобы управлять маршрутизацией, в каждом маршрутизаторе есть настройки пиритизации интерфейсов в виде метрик OSPF:

```
<Router name="R00" RFC1583Compatible="true">
  <BroadcastInterface ifName="eth0" areaID="0.0.0.0" interfaceOutputCost="1"/>
  <PointToPointInterface ifName="eth1" areaID="0.0.0.0" interfaceOutputCost="1" />
  <PointToPointInterface ifName="eth2" areaID="0.0.0.0" interfaceOutputCost="10" />
  <PointToPointInterface ifName="eth3" areaID="0.0.0.0" interfaceOutputCost="100" />
 </Router>
```

Метрика 1 взята за цену маршрута 10Гбит/с, соответственно 10 и 100 – для 1Гбит/с и 100Мбит/с. При таком способе задания метрики маршрут строится как самый быстрейший по сумме всех каналов следования пакета, как описано в [3].

Для включения OSPF вместо статической маршрутизации необходимо явно это задать:

***.configurator.addStaticRoutes = false*
***.R??.hasOSPF = true*

Для обеспечения корректного симулирования очередей была выбрана очередь DropTail и последующий обработчик WRRQ, как описано в [2]

***.R??.eth[*].queueType = "DropTailQueue"*
***.R??.eth[*].queue.frameCapacity = 256*

Размер очереди 256 пакетов взят из спецификации распространенного маршрутизатора Cisco серии 2900 (2921). Для задания режима Sparse нужно использовать явное указание режима работы и точки RP

***.pimConfig = xml("<config><interface mode=\"sparse\"/></config>")*
***.RP = "192.168.3.1"*

где IP адрес должен совпадать с крайним маршрутизатором, подключенным серверу вещания. Также, в случае Dense маршрутизации явно определяем режим:

***.pimConfig = xml("<config><interface mode=\"dense\"/></config>")*

Для подключения OpenFlow протокола был использован модуль [4], который поддерживает базовую функциональность OpenFlow версии 1.0, которой вполне хватает для реализации маршрутизации. Для работы OpenFlow необходимо добавить в модель поддержку OF и модуля коммутации:

import openflow.nodes.;*
import openflow.utility.SpanningTree;

и настроить контроллер

***.controller.ofa_controller.port = 6633*
***.open_flow_switch*.sendCompletePacket = false*
***.controller.behavior = "Router"*
***.ofa_switch.connectPort = 6633*
***.ofa_switch.connectAddress = "controller"*
***.buffer.capacity = 10*
***.ofa_switch.flow_timeout = 5s*
***.open_flow_switch*.etherMAC[*].promiscuous = true*

где «controller» - адрес контроллера, «Router» - место применения OpenFlow. Маршрутизаторы для работы с OpenFlow тоже нужно изменить, добавив в них коммутатор OpenFlow на вход из Ethernet интерфейсов.

4 Апробация симулятора

В результате прогона симулятора в файл записываются все показатели сети, которые можно затем исследовать как в векторном виде, так и в виде гистограмм и абсолютных чисел (рисунок 4)

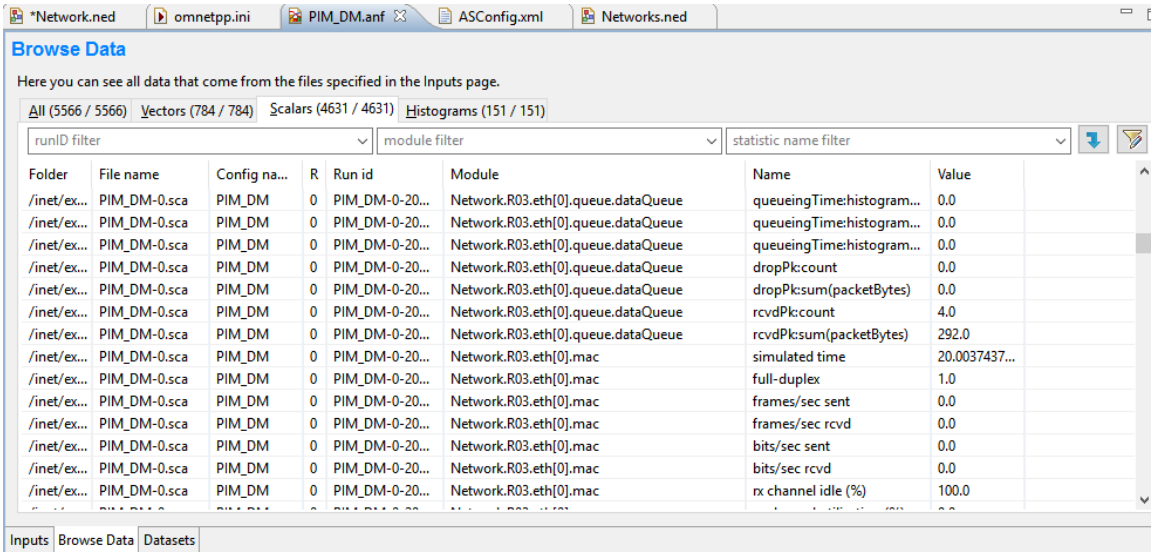

Рисунок –4 Результаты симуляции

Например, для времени ожидания в очереди маршрутизатора R01 и интерфейса eth3 будет выглядеть так:

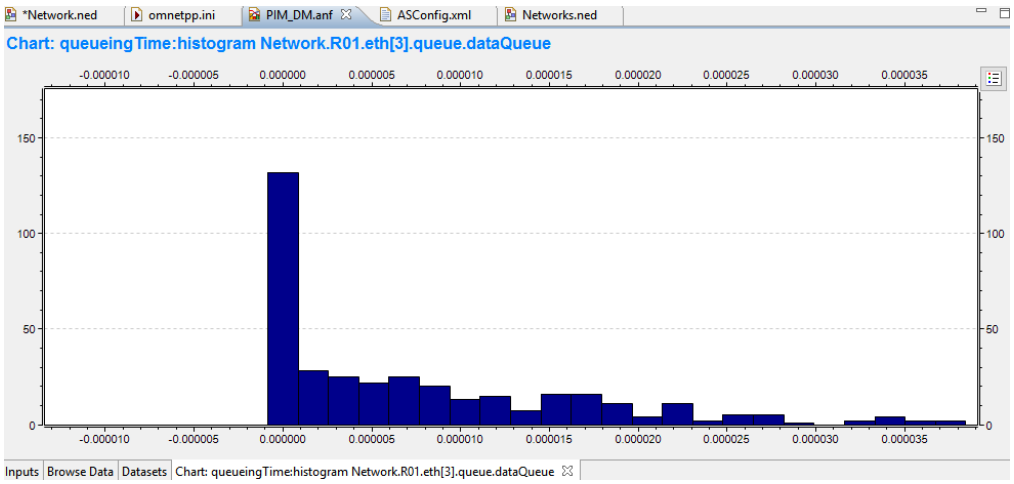

Рисунок 5 – Гистограмма времени ожидания в очереди

В настоящее время провайдер не имеет возможности контролировать деревья мультимедийного трафика на уровне доступа и между L2 коммутаторами. Главный канал может быть перегружен 200–300 IPTV-каналами, и при увеличении числа каналов в HD-разрешении увеличивается и перегрузка.

Рисунок 6 – Тестовый прогон симуляции для базового уровня

Но для сравнительного анализа задержек лучше использовать гистограммы: на рисунке 7 показана гистограмма для ненагруженного режима сети при генерации пакетов по экспоненциальному закону с интенсивностью 1000 пакетов, это около 5 FullHD каналов IPTV или 7-8 Мбит/с.

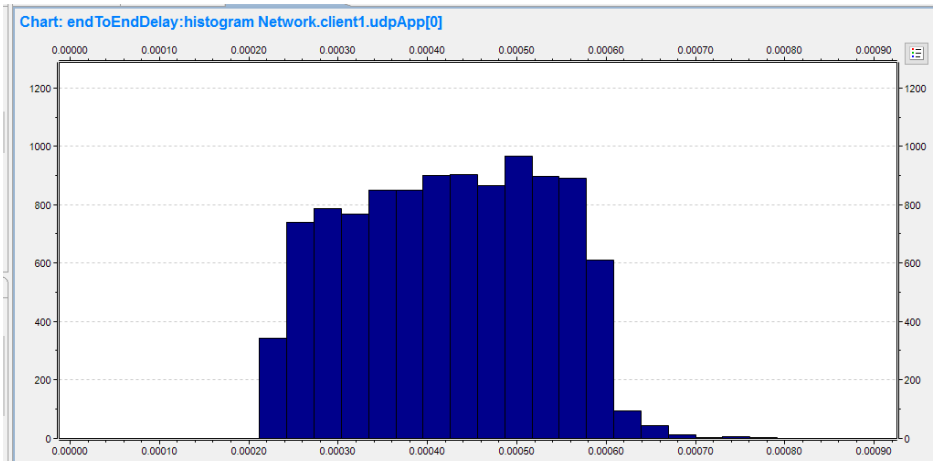

Рисунок 7 - Гистограмма времени отклика

5 Выводы

Разработанный симулятора показан принципиальную работоспособность и гибкость, а также возможность изучать различные аспекты передачи трафика IPTV не вдаваясь в подробности моделирования и не зная основы языков модельных сред. Это позволяет использовать симулятор как в обучении, так и при проектировании новых сетей многоадресной рассылки, в том числе и с использованием других способов маршрутизации и коммутации, например, OpenFlow.

Проект был реализован при финансовой поддержке Российского фонда фундаментальных исследований, проект № 15-07-06071.

Список литературы

1. Ushakov Yu., Polezhaev P., Legashev L., Bolodurina I., Shukhman A., Bakhareva N. Increasing the Efficiency of IPTV by Using Software-Defined Networks // International Conference on Next Generation Wired/Wireless Networking. – Springer International Publishing, 2016. – PP. 550-560.

2. Veselý V., Ryšavý O., Švéda M. Protocol Independent Multicast in OMNeT++. // ICNS 2014: The Tenth International Conference on Networking and Services. – PP. 132-137

3. Ashique M. Simulation-Based Comparative Study of EIGRP and OSPF for Real-Time Applications. Master Thesis Electrical Engineering Thesis no: MEE 10:53 September 2010

4. An OpenFlow Extension for the OMNeT++ INET Framework. Lehrstuhl für Informatik III, Am Hubland. Режим доступа: https://www3.informatik.uni-wuerzburg.de/research/ngn/ofomnet/of_omnet.shtml - 20.12.2016.