Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
«____» _____ 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Методы повышения производительности сети при пиковых нагрузках

09.04.01 Информатика и вычислительная техника

09.04.01.05 Сети ЭВМ и телекоммуникации

Научный руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	Ф. А. Казаков
Выпускник	_____ подпись, дата		А. В. Сопелев
Рецензент	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	Д. А. Кузьмин
Нормоконтролер	_____ подпись, дата		В. И. Иванов

Красноярск 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Методы повышения производительности сети при пиковых нагрузках» содержит 67 страниц текстового документа, 22 иллюстрации, 3 таблицы, 6 формул, 3 приложения, 25 использованных источников.

СЕТИ ПЕРЕДАЧИ ДАННЫХ, МАРШРУТИЗАЦИЯ, ПРОТОКОЛ МАРШРУТИЗАЦИИ OSPF, ПРОТОКОЛ МАРШРУТИЗАЦИИ EIGRP, ПРОПУСКНАЯ СПОСОБНОСТЬ, МЕТРИКА, ЗАГРУЗКА, ТАБЛИЦА МАРШРУТИЗАЦИИ, ОПТИМИЗАЦИЯ, ИНТЕРФЕЙС, МОДИФИКАЦИЯ, МОДЕЛИРОВАНИЕ, OMNET++, ANSAINET.

Цель работы – повышение производительности сети при пиковых нагрузках за счет внесения модификации в протокол маршрутизации EIGRP и моделирование полученных результатов в фреймворке ANSAINET для среды OMNeT++.

Задачи работы:

- рассмотреть наиболее используемые протоколы маршрутизации сетей, в частности – протоколы OSPF и EIGRP;

- определить преимущества и явные недостатки OSPF и EIGRP, не позволяющие эффективно использовать эти протоколы в сетях с большим уровнем трафика, продемонстрировать их на примере моделирования сети в системе имитационного моделирования OMNeT++;

- предложить модификацию протокола EIGRP для перестроения таблиц маршрутизации с учетом динамически изменяемой нагрузки на каналы;

- реализовать предложенные изменения, показать их работоспособность и оценить их параметры моделированием сети в фреймворке ANSAINET для OMNeT++.

В результате выполнения работы был разработан метод повышения производительности сети при пиковых нагрузках, где в качестве протокола маршрутизации используется протокол EIGRP. Метод основан на измерении текущей загрузки на интерфейсах маршрутизаторов и перерасчете маршрутов при падении производительности на них.

В итоге метод был реализован в библиотеке ANSAINET для OMNeT++, работоспособность доработки была показана в ходе моделирования.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Первые компьютерные сети появились в конце 60-х годов прошлого века. Они унаследовали много имеющих пользу свойств от телекоммуникационных телефонных сетей. В то же время компьютерные сети сделали доступными для всех огромнейшие объемы информации, созданные цивилизацией за время своего существования. И эти данные продолжают пополняться со все больше растущей скоростью.

Позже в 80-х были разработаны и утверждены стандартные технологии объединения компьютеров в сеть – Ethernet, ARCNET, Token Ring, Token Bus и другие. Стандартизация сетевых технологий превратила процесс построения локальной сети из решения сложной технической проблемы в рутинную работу.

В конце 80-х годов локальные и глобальные сети имели существенные отличия по протяженности и качеству линий связи, сложности методов передачи данных, скорости обмена данными, разнообразию предоставляемых услуг и масштабируемости. В дальнейшем в результате тесной интеграции LAN (локальные сети), WAN (глобальные сети) и MAN (городские сети) произошло взаимопроникновение соответствующих технологий [1, с. 37].

В крупных сетях со сложной топологией и большим количеством альтернативных маршрутов использование протоколов маршрутизации автоматизирует построение таблиц маршрутизации, а также позволяет отыскать новые маршруты при изменениях сети: отказах или появлении новых линий связи и маршрутизаторов. Хотя протоколы маршрутизации, в отличие от сетевых протоколов (IP и IPX), не являются обязательными, так как таблица маршрутизации может строиться администратором сети вручную, но такие протоколы выполняют очень важную и полезную работу.

Протоколы маршрутизации обеспечивают поиск и фиксацию маршрутов продвижения данных через составную сеть TCP/IP [1, с. 572].

Существуют способы продвижения пакетов в составных сетях, которые не требуют наличия таблиц маршрутизации на маршрутизаторах.

Одним из таких способов передачи пакетов по сети является лавинная маршрутизация, когда каждый маршрутизатор передает пакет всем своим непосредственным соседям, исключая тот, от которого его получил (сильное снижение пропускной способности).

Еще одним видом маршрутизации, не требующим наличия таблиц маршрутизации, является маршрутизация от источника (англ. source routing). Отправитель помещает в пакет информацию о том, какие промежуточные маршрутизаторы должны участвовать в передаче пакета к сети назначения. На основе этой информации каждый маршрутизатор считывает адрес следующего маршрутизатора, и, если он действительно является адресом его непосредственного соседа, передает ему пакет для дальнейшей обработки [1, с. 572]. Сложность в том, как отправитель узнает точный маршрут следования пакета через сеть (маршрут можно задавать либо вручную, либо узел-отправитель делает это автоматически,

но в этом случае ему нужно поддерживать какой-либо протокол маршрутизации, который сообщит ему о топологии и состоянии сети).

Большинство протоколов маршрутизации создают таблицы маршрутизации.

Различают протоколы, выполняющие статическую и адаптивную (динамическую) маршрутизацию.

При статической маршрутизации все записи в таблице статичны (это подразумевает бесконечный срок их жизни). Записи о маршрутах составляются и вводятся в память каждого маршрутизатора вручную администратором сети. При изменении состояния сети администратору необходимо срочно отразить эти изменения в соответствующих таблицах маршрутизации, иначе может произойти их рассогласование, и сеть будет работать некорректно.

При адаптивной маршрутизации все изменения конфигурации сети автоматически отражаются в таблицах маршрутизации благодаря протоколам маршрутизации, которые собирают информацию о топологии связей в сети, что позволяет им оперативно отрабатывать все текущие изменения. В таблицах маршрутизации при адаптивной маршрутизации обычно имеется информация об интервале времени, в течение которого данный маршрут будет оставаться действительным. Это время называют временем жизни (англ. time to live, TTL) маршрута. Если по истечении времени жизни существование маршрута не подтверждается протоколом маршрутизации, то он считается нерабочим, пакеты по нему больше не посылаются.

Применяемые сегодня в IP-сетях протоколы маршрутизации относятся к адаптивным распределенным протоколам, которые, в свою очередь, делятся на две группы [1, с. 573]:

- дистанционно-векторные алгоритмы (Distance Vector Algorithm, DVA);
- алгоритмы состояния связей (Link State Algorithm, LSA).

В дистанционно-векторных алгоритмах каждый маршрутизатор периодически и широковещательно рассылает по сети вектор, компонентами которого являются расстояния (измеренные в той или иной метрике) от данного маршрутизатора до всех известных ему сетей. Пакеты протоколов маршрутизации обычно называют объявлениями о расстояниях, так как с их помощью маршрутизатор объявляет остальным маршрутизаторам известные ему сведения о конфигурации сети [1, с. 574].

Получив от некоторого соседа вектор расстояний (дистанций) до известных тому сетей, маршрутизатор наращивает компоненты вектора на величину расстояния от себя до данного соседа. Кроме того, он дополняет вектор информацией об известных ему самому других сетях, о которых он узнал непосредственно (если они подключены к его портам) или из аналогичных объявлений других маршрутизаторов. Обновленное значение вектора маршрутизатор рассылает своим соседям. В конце концов, каждый маршрутизатор узнает через соседние маршрутизаторы информацию обо всех имеющихся в составной сети сетях и о расстояниях до них [1, с. 574].

Затем он выбирает из нескольких альтернативных маршрутов к каждой сети тот маршрут, который обладает наименьшим значением метрики. Маршрутизатор, передавший информацию о данном маршруте, отмечается в таблице маршрутизации как следующий (англ. next hop).

Дистанционно-векторные алгоритмы хорошо работают только в небольших сетях. В больших сетях они периодически засоряют линии связи интенсивным трафиком, к тому же изменения конфигурации не всегда корректно могут отрабатываться алгоритмом этого типа, так как маршрутизаторы не имеют точного представления о топологии связей в сети, а располагают только косвенной информацией – вектором расстояний [1, с. 574].

Наиболее распространенным протоколом, основанным на дистанционно-векторном алгоритме, является протокол RIP. Однако существуют и другие популярные протоколы, относящиеся к этой группе. Например, усовершенствованный дистанционно-векторный протокол динамической маршрутизации EIGRP, разработанный компанией Cisco и являющийся преемником IGRP. EIGRP является внутренним протоколом шлюзов, пригодным для различных топологий и сред. Согласно [2] в хорошо спроектированной сети EIGRP хорошо масштабируется и позволяет обеспечить малое время согласования при минимальном сетевом трафике. Для расчета кратчайшего пути к конечному адресу используется алгоритм диффузного обновления (англ. Diffusing Update Algorithm, DUAL). К преимуществам алгоритма работы сети EIGRP можно отнести:

- низкое использование сетевых ресурсов в режиме нормальной эксплуатации; только пакеты HELLO передаются в условиях стабильной сети;
- при возникновении изменений по сети передаются только изменения, произошедшие в маршрутной таблице, а не вся таблица целиком; это позволяет уменьшить нагрузку на сеть, создаваемую протоколом маршрутизации;
- малое время конвергенции (или сходимости) в случае изменения в топологии сети (в отдельных случаях сходимость обеспечивается почти мгновенно);
- возможность использования до 5-ти компонентов (минимальная пропускная способность маршрута до конечного адреса, суммарная задержка, надежность маршрута, загрузка маршрута, минимальный путь максимального размера передаваемого блока данных (англ. maximum transmission unit, MTU)) при расчете метрики маршрутизации.

Изначально при вычислении стоимости пути в EIGRP берутся значения минимальной пропускной способности и задержки на всем маршруте. Это поведение можно изменить, добавив в формулу метрики, например, надежность и загрузку канала, но они фиксируются только в момент изменения топологии сети (состояния интерфейса, какого-либо IP-адреса, пропускной способности и (или) задержки)) [3; 4], то есть этот протокол маршрутизации не перестраивается под возникающие в компьютерной сети нагрузки, когда скорость передачи данных сильно падает. Такая мера реализована для снижения уровня служебного трафика в сети и сохранения совместимости векторных метрик IGRP и EIGRP, чтобы была возможность одновременного использования этих протоколов и

(или) легкой миграции с IGRP к EIGRP. У IGRP такая же составная метрика и очень похожая формула ее вычисления, этот протокол периодически анонсирует маршрутную информацию, то есть загрузка и надежность интерфейса регулярно распространяются по всей сети, поэтому эти параметры включены в расчет составной метрики, что может привести к нестабильности работы сети из-за большого количества служебного трафика, вызываемого постоянными изменениями данных показателей [3].

Алгоритмы состояния связей обеспечивают каждый маршрутизатор информацией, достаточной для построения точного графа связей сети. Все маршрутизаторы работают на основании одного и того же графа, что делает процесс маршрутизации более устойчивым к изменениям конфигурации.

Каждый маршрутизатор использует граф сети для нахождения оптимальных по некоторому критерию маршрутов до каждой из сетей, входящих в составную сеть.

Чтобы понять, в каком состоянии находятся линии связи, подключенные к его портам, маршрутизатор периодически обменивается короткими пакетами Hello со своими ближайшими соседями. В отличие от протоколов DVA, которые регулярно передают вектор расстояний, протоколы LSA ограничиваются короткими сообщениями, а передача более объемных сообщений происходит только в тех случаях, когда с помощью сообщений Hello был установлен факт изменения состояния какой-либо связи [1, с. 574].

В результате служебный трафик, создаваемый протоколами LSA, гораздо менее интенсивный, чем у протоколов DVA.

Протоколами, основанными на алгоритме состояния связей, являются протокол IS-IS стека OSI (этот протокол используется также в стеке TCP/IP) и протокол OSPF стека TCP/IP.

В настоящее время для маршрутизации в магистральных IP-сетях часто используется протокол OSPF, в котором для распределения входящих потоков применяется алгоритм Дейкстры. К положительным качествам распределения данных с помощью алгоритма Дейкстры можно отнести его простоту практической реализации.

С другой стороны, благодаря использованию этим алгоритмом статического критерия качества распределения входного потока информации (метрика), алгоритм Дейкстры не адаптирован к работе в сетях с большим уровнем динамически изменяемого трафика. Перегрузки могут повлечь за собой отказ важных с точки зрения передачи данных узлов сети (например таких, в случае выхода из строя которых данные не передадутся адресату).

Протоколы OSPF и EIGRP приняты в работе за базовые, спецификация OSPF описана в документе RFC 2328 [5; 6], EIGRP – в RFC 7868 [4]. В EIGRP есть встроенный механизм учета нагрузки на линию, но этот инструмент недостаточно гибок. Протокол EIGRP можно тщательно настроить за счет использования метрики со множеством составляющих, которые также могут быть изменены под разные требования варьированием значений коэффициентов при них

[3; 4]. Однако пересчет маршрутов происходит только в случае изменения топологии сети, и динамически изменяемая нагрузка на канал не учитывается.

Целью данной работы является повышение производительности сети при пиковых нагрузках за счет внесения модификации в протокол маршрутизации EIGRP и моделирование полученных результатов в фреймворке ANSAINET для среды OMNeT++.

Для достижения указанной цели были поставлены следующие задачи:

- рассмотреть наиболее используемые протоколы маршрутизации сетей, в частности – протоколы OSPF и EIGRP;

- выявить преимущества и явные недостатки OSPF и EIGRP, не позволяющие эффективно использовать эти протоколы в сетях с большим уровнем трафика, продемонстрировать их на примере моделирования сети в системе имитационного моделирования OMNeT++;

- предложить модификацию протокола EIGRP для перестроения таблиц маршрутизации с учетом динамически изменяемой нагрузки на каналы;

- реализовать предложенные изменения, показать их работоспособность и оценить их параметры моделированием сети в фреймворке ANSAINET для OMNeT++.

Работа будет вестись, в основном, экспериментально, разрабатываемый метод основан на измерении текущей загрузки на интерфейсах маршрутизаторов сети и перерасчете таблиц маршрутизации при падении производительности на них. Основным оптимальным подходом к оценке работоспособности сети, ее настройке, а также к проверке новых модификаций является моделирование сети в специализированном программном обеспечении, предназначенном для создания сетевых симуляторов. Одним из таких популярных программных средств является библиотека OMNeT++, которую и предлагается использовать для моделирования.

В работе предполагается использовать понятия и методы теории графов, теории алгоритмов, теории массового обслуживания, теории языков программирования, теории систем и сетей. На этапе анализа должен быть определен критерий эффективности работы сети. При реализации деталей модификации объектно-ориентированный подход является основным.

1 Анализ состояния решаемой проблемы

1.1 Протокол динамической маршрутизации Open Shortest Path First

1.1.1 Обзор протокола

Протокол OSPF (англ. Open Shortest Path First – по самому короткому пути) служит для маршрутизации трафика TCP/IP. Протокол OSPF относится к числу внутренних протоколов маршрутизации (англ. Interior Gateway Protocol, IGP) – это означает, что маршрутная информация распространяется между маршрутизаторами одной автономной системы (англ. Autonomous System, AS). Протокол OSPF работает на основе технологии SPF (или link-state – по состоянию канала) в отличие от алгоритмов Беллмана-Форда, используемых традиционными протоколами маршрутизации TCP/IP [6, с. 3].

Протокол OSPF подготовлен одноименной рабочей группой IETF и предназначен для использования в средах TCP/IP. Протокол включает явную поддержку бесклассовой адресации и установки меток (англ. tagging) при использовании внешней маршрутной информации. OSPF использует аутентификацию и групповую адресацию (англ. IP multicast) при обмене маршрутными сообщениями. Кроме того, при разработке протокола были приложены значительные усилия по ускорению обработки топологических изменений в сети и снижению уровня служебного трафика.

OSPF обеспечивает маршрутизацию пакетов IP исключительно на основе IP-адресов получателей, определенных из заголовка пакетов IP. Пакеты IP маршрутизируются без их изменения, то есть не используется инкапсуляция в какие-то иные пакеты. OSPF является динамическим протоколом маршрутизации, обеспечивающим быстрое обнаружение топологических изменений в AS (например, сбои маршрутизаторов или каналов) и расчет новых беспетлевых (англ. loop-free) маршрутов. Период схождения (англ. convergence) – расчет нового маршрута – достаточно короток и уровень служебного трафика невелик [6, с. 3].

В протоколах на основе состояния каналов каждый маршрутизатор поддерживает базу данных с описанием топологии AS. Эти базы называют базами данных о состоянии каналов (англ. link-state database). Базы данных всех маршрутизаторов одной области идентичны. Каждый элемент базы данных представляет собой локальное состояние отдельного маршрутизатора (например, поддерживаемые интерфейсы или доступные соседи). Маршрутизаторы распространяют информацию о своем локальном состоянии путем лавинной маршрутизации (англ. flooding).

Все маршрутизаторы работают в параллель, используя одинаковый алгоритм. На основе базы каналов каждый маршрутизатор строит дерево кратчайших путей, корнем которого является сам маршрутизатор. Это дерево содержит маршруты ко всем адресатам внутри AS. Маршрутная информация внешнего происхождения представляется как листья дерева.

При наличии нескольких путей равной стоимости к одному адресату трафик поровну распределяется между всеми маршрутами (по каждому из маршрутов пакеты отправляются попеременно). Стоимость маршрута описывается безразмерной метрикой, представляемой в виде одного числа. По умолчанию в этой метрике учитывается пропускная способность каналов связи. Например, для Ethernet значение равно 10, для Fast Ethernet – 1, для канала T-11 – 65, для канала с пропускной способностью 56 Кбит/с – 1785. При наличии высокоскоростных каналов, таких как Gigabit Ethernet или STM-16/64, администратору нужно задать другую шкалу скоростей, назначив единичное расстояние наиболее скоростному каналу [1, с. 584]. Допускается применение других метрик, одна из них учитывает задержки, а другая – надежность передачи пакетов каналами связи. Для каждой из метрик протокол OSPF строит отдельную таблицу маршрутизации. Выбор нужной таблицы происходит в зависимости от значений битов TOS в заголовке пришедшего IP-пакета (англ. Type of Service, предназначен для приоритизации IP-трафика на сетевых маршрутизаторах, чтобы обеспечить высокое качества передачи данных).

OSPF позволяет группировать сети – такие группы называют областями (англ. area). Топология области невидима для остальной части AS. Такое сокрытие (избыточной) информации позволяет существенно снизить уровень служебного трафика. Кроме того, маршрутизация внутри области определяется исключительно внутренней топологией этой области, что обеспечивает защиту областей от использования некорректной маршрутной информации. Понятие области является обобщением подсетей IP.

OSPF обеспечивает возможность гибкой настройки подсетей IP. Каждый маршрут в OSPF распространяется с указанием адресата и маски подсети. Две разных подсети одной сети IP могут иметь различные размеры (то есть разные маски) – для обозначения этого обычно используется термин variable length subnetting (переменный размер подсетей). Пакеты маршрутизируются по пути с наилучшим (то есть самым длинным или более конкретным) соответствием. Маршруты к хостам рассматриваются как пути в подсети с маской из одних единиц (0xffffffff или 255.255.255.255).

Весь обмен информацией OSPF осуществляется с использованием аутентификации. Это означает, что в маршрутизации внутри AS могут участвовать только уполномоченные маршрутизаторы. Могут использоваться различные схемы аутентификации; в частности, допустимо применение различных схем для каждой подсети IP.

Внешние маршрутные данные (то есть маршруты, полученные от протоколов внешних шлюзов EGP (англ. Exterior Gateway Protocol) – например, BGP) анонсируются через AS. Такие данные сохраняются отдельно от данных OSPF о состоянии каналов. Каждый внешний маршрут может также быть помечен анонсирующим его маршрутизатором – это дает возможность передачи дополнительной информации между маршрутизаторами на границе AS [6, с. 4].

1.1.2 Пакеты протокола маршрутизации

Протокол OSPF инкапсулируется в IP, используя идентификатор 89, не требует какой-либо дополнительной фрагментации или сборки пакетов – при возникновении такой необходимости используется обычная фрагментация и сборка IP. Пакеты протокола OSPF имеют такой формат, что большие блоки протокольной информации в общем случае можно легко разделить на более мелкие пакеты. Рекомендуется использовать именно такой вариант, избегая по возможности фрагментирования IP.

Пакеты протокола маршрутизации всегда должны передаваться с нулевым значением поля IP TOS. Если это возможно, пакеты протоколов маршрутизации должны иметь преимущество по сравнению с обычным трафиком данных IP как для приема, так и при передаче [6, с. 14].

Все пакеты протокола OSPF используют однотипные заголовки. Типы пакетов OSPF приведены в таблице 1. Протокол OSPF Hello использует пакеты Hello для организации и поддержки соседских отношений. Пакеты Database Description (описание базы данных) и Link State Request (запрос состояния канала) служат для поддержки отношений смежности. Гарантированный обмен обновлениями OSPF основан на обмене пакетами Link State Update (обновление состояния канала) и Link State Acknowledgment (подтверждение приема обновления).

Таблица 1 – Типы пакетов OSPF

Тип	Название	Назначение
1	Hello	Обнаружение и поддержка соседства
2	Database Description	Резюмирование содержимого БД
3	Link State Request	Загрузка БД
4	Link State Update	Обновление БД
5	Link State Acknowledgment	Подтверждение лавинной рассылки

Каждый пакет Link State Update содержит набор новых анонсов состояния каналов (англ. link-state advertisement, LSA – объявление о состоянии канала) на один интервал (англ. hop) удаленных от пункта генерации анонса. Один пакет Link State Update может содержать анонсы LSA от нескольких маршрутизаторов. Каждая запись LSA помечается идентификатором создавшего анонс маршрутизатора и сопровождается контрольной суммой содержимого. В каждой записи LSA имеется также поле типа; возможные варианты этого поля описаны в таблице 2.

Таблица 2 – Типы анонсов LSA в OSPF

Тип	Имя LSA	Описание LSA
1	Router-LSA	Генерируются всеми маршрутизаторами. Этот тип LSA описывает состояния интерфейсов маршрутизатора в область. Анонс рассылается в лавинном режиме внутри области.
2	Network-LSA	Генерируется выделенным маршрутизатором DR для широковещательных и NBMA-сетей. Этот тип LSA включает список маршрутизаторов, подключенных к сети. Рассылается в лавинном режиме внутри области.
3, 4	Summary-LSA	Генерируется граничными маршрутизаторами областей и рассылается в лавинном режиме в пределах связанной с LSA области. Каждый анонс summary-LSA описывает маршрут к адресату за пределами данной области, но внутри данной AS (междоменный маршрут). Тип 3 summary-LSA описывает маршруты в сети, а тип 4 – к граничным маршрутизаторам AS.
5	AS-external-LSA	Генерируется граничными маршрутизаторами AS и рассылается по всей автономной системе. Каждый анонс AS-external-LSA описывает маршрут к адресатам в другой AS. Принятые по умолчанию маршрутизаторы AS также могут описываться в AS-external-LSA.

Пакеты протокола OSPF (за исключением пакетов Hello) передаются только между смежными маршрутизаторами. Это означает, что все пакеты протокола OSPF проходят только один интервал между маршрутизаторами, за исключением тех ситуаций, когда смежность поддерживается через виртуальные соединения (англ. virtual adjacency). IP-адрес отправителя пакета OSPF является адресом одного из смежных маршрутизаторов, а IP-адрес получателя является адресом второго из смежных маршрутизаторов или групповым IP-адресом [6, с. 15].

1.1.3 Достоинства и недостатки протокола

К основным преимуществам OSPF можно отнести следующее:

- высокая скорость сходимости по сравнению с протоколами, использующими дистанционно-векторную маршрутизацию;
- поддержка сетей с переменным размером (работа с сетевыми масками переменной длины);
- оптимальное использование пропускной способности (низкий уровень служебного трафика) с построением дерева кратчайших путей. Таким образом, данный протокол можно использовать даже в очень больших сетях (высокая масштабируемость).

Есть также некоторые недостатки:

- сложность OSPF, что требует грамотного планирования, более комплексной настройки и администрирования;

- из-за использования алгоритма Дейкстры при построении дерева кратчайших путей протокол не защищает сеть от перегрузок (отслеживание загрузки канала в динамике не происходит). Это требует реализации дополнительных методов по снижению вероятности перегрузки.

1.2 Протокол динамической маршрутизации Enhanced Interior Gateway Routing Protocol

1.2.1 Обзор протоколов IGRP и EIGRP

IGRP (англ. Interior Gateway Routing Protocol) – протокол маршрутизации DVA, используется в сетях TCP/IP и OSI, был спроектирован фирмой Cisco. Первая IP-версия протокола была разработана и успешно внедрена в 1986 году. IGRP рассматривается как IGP-протокол, но он также широко применяется как EGP для междоменной маршрутизации [7; 8].

Информация о расстоянии в IGRP представляется набором сведений, состоящим из доступной полосы пропускания, времени задержки, загрузки и надежности канала связи. Это позволяет точно настраивать характеристики канала для выбора оптимальных маршрутов [7; 8; 9].

EIGRP (англ. Enhanced Interior Gateway Routing Protocol) – это улучшенная версия IGRP, где используется та же технология DVA, и основная дистанционная информация остается прежней. Протокол EIGRP обеспечивает совместимость и полное взаимодействие с маршрутизаторами IGRP. По умолчанию маршруты IGRP имеют более высокий приоритет, чем маршруты его расширенной версии. Свойства сходимости и эффективность работы этого протокола были значительно улучшены. Это позволяет модернизировать архитектуры сети с сохранением средств, вложенных в разработку сети на базе протокола IGRP [7; 8].

Технология конвергенции основана на исследованиях, проводимых компанией SRI International. В EIGRP для получения беспетлевых маршрутов в каждый момент времени их расчета применяется распределенный обновляемый алгоритм DUAL, что позволяет синхронизировать все маршрутизаторы, участвующие в изменении топологии. Маршрутизаторы, на которые не влияют изменения топологии, не участвуют в этом процессе. Время конвергенции по алгоритму DUAL сопоставимо с показателями других существующих протоколов маршрутизации [7; 8].

EIGRP независим от сетевого уровня, позволяя алгоритму DUAL поддерживать другие группы протоколов [7; 8].

Протокол EIGRP состоит из четырех основных компонентов [7; 8; 9]:

а) обнаружение/восстановление соседа (англ. Neighbor Discovery/Recovery);

б) надежный транспортный протокол (англ. Reliable Transport Protocol, RTP);

в) блок конечных состояний алгоритма DUAL (англ. DUAL Finite State Machine);

г) модули, зависимые от протоколов (англ. Protocol Dependent Modules, PDM).

Процесс а) используется маршрутизаторами для динамического изучения других маршрутизаторов, к сетям которых они непосредственно подключены. Маршрутизаторы должны также определять недоступность соседа. Этот процесс обеспечивается периодической отправкой небольших по размеру пакетов HELLO. Сосед функционирует нормально, пока маршрутизатор получает такие пакеты. Когда отношение соседства установлено, роутеры могут обмениваться маршрутной информацией.

Надежный транспортный протокол отвечает за гарантированную, упорядоченную доставку пакетов EIGRP всем соседям. Он поддерживает групповую и одноадресную (англ. unicast) адресации. Одни пакеты EIGRP должны передаваться с большой степенью надежности (например, UPDATE-пакеты, что и указывается в них), а для других это необязательно (например, HELLO-пакеты в Ethernet-сети с мультивещанием и множественным доступом). Поэтому для повышения эффективности надежность предоставляется только в случае необходимости. У RTP есть возможность быстрой передачи multicast-пакетов, когда есть неподтвержденные пакеты. Такие средства гарантируют, что время сходимости будет низким при наличии в сети каналов связи, работающих на разных скоростях передачи данных.

Блок конечных состояний алгоритма DUAL реализует процесс принятия решений для расчетов всех маршрутов. Блок отслеживает все маршруты, объявленные всеми соседями. Дистанционная информация (метрика) используется DUAL для выбора эффективных путей, не содержащих петель (циклов). Алгоритм DUAL выбирает маршруты для добавления в таблицу маршрутизации на основе принципа вероятных последующих элементов (или возможных преемников). Последующий элемент (преемник) – это соседний маршрутизатор, используемый для передачи пакетов, с маршрутом самой низкой стоимости к адресату, который гарантированно не является частью цикла маршрутизации. Когда нет таких элементов, но есть соседи, объявляющие пункт назначения, необходимо произвести пересчет, чтобы определить новый последующий элемент. Время пересчета влияет на общее время конвергенции. Хотя пересчет не требует интенсивного использования процессора, лучше избегать этого без необходимости. При изменении топологии DUAL проверяет наличие вероятных последующих элементов. Если они есть, алгоритм использует все обнаруженные, чтобы избежать ненужных вычислений.

PDM отвечают за обработку требований конкретных протоколов сетевого уровня. Например, модуль IP-EIGRP отвечает за отправку и получение пакетов EIGRP, инкапсулированных в IP-пакеты, за их разбор и уведомление DUAL о получении новой информации. IP-EIGRP обращается к алгоритму DUAL за принятием решений о маршрутизации, результаты которых хранятся в IP-таблице

маршрутизации. IP-EIGRP отвечает за перераспределение маршрутов, обнаруженных другими протоколами IP-маршрутизации.

1.2.2 Пакеты протокола маршрутизации EIGRP

Протокол EIGRP использует пять типов пакетов [4; 7; 8; 10]:

- HELLO/ACK (англ. acknowledgment) – multicast-пакеты, отправляемые для обнаружения/восстановления соседа, не требующие подтверждения о получении. Пакет HELLO, не содержащий данных, также используется как подтверждение. Пакет подтверждения ACK всегда посылается в режиме одиночной отправки и содержит ненулевой номер подтверждения;

- UPDATE. Пакеты обновлений используются для передачи параметров узлов назначения. Когда обнаруживается новый сосед, ему посылаются пакеты UPDATE, чтобы он мог создать свою таблицу топологии. В этом случае пакеты UPDATE посылаются в режиме одиночной отправки. В других случаях, например, при изменении стоимости связи, такие пакеты посылаются в режиме мультиотправки. Обновления всегда передаются с подтверждением;

- QUERY и REPLY. Пакеты запросов (QUERY) и откликов (REPLY) посылаются, когда маршруты назначений переходят в активное состояние. Пакеты QUERY всегда посылаются в multicast-режиме, только если они не отправляются в ответ на полученный запрос. В этом случае запрос посылается в unicast-режиме обратно последующему элементу, создавшему первоначальный запрос. REPLY всегда посылаются в ответ на QUERY, чтобы оповестить запрашивающего о том, что переходить в активное состояние не нужно, так как откликающийся располагает вероятными последующими элементами. Пакеты REPLY посылаются в режиме одиночной отправки запрашивающему. Запросы и ответы передаются с подтверждением;

- REQUEST-пакеты используются для получения специфической (конкретной нужной в данный момент) информации от одного или нескольких соседей, могут передаваться в multicast- или unicast-режимах. Запросы передаются с негарантированной доставкой.

1.2.3 Метрика протокола маршрутизации EIGRP

Метрика EIGRP основана на 5-ти компонентах [2; 4; 10]:

- bandwidth (BW) – наименьшая пропускная способность между узлами источником и назначения;

- delay ($DELAY$) – суммарное время задержки интерфейсов на всем пути;

- reliability (REL) – наихудший показатель надежности на всем пути на основании keepalive-сообщений;

- load ($LOAD$) – наихудший показатель загрузки канала на всем пути на основании уровня передачи пакетов и настроенной пропускной способности на интерфейсе;

- MTU – наименьшее MTU на всем пути. MTU включается в обновления EIGRP, но фактически не используется для подсчета метрики.

По умолчанию для подсчета метрики используются BW и $DELAY$. Остальные критерии не рекомендуется использовать, так как это приведет к частым пересчетам маршрутов.

EIGRP подсчитывает метрику с использованием K-коэффициентов, которые передаются в пакетах HELLO. По умолчанию значения коэффициентов такие:

$$K_1 = K_3 = 1,$$

$$K_2 = K_4 = K_5 = 0$$

Если $K_5 = 0$, то частное при REL определяется как 1 [4].

Общая метрика вычисляется при помощи значений BW и $DELAY$. Используется следующая формула для вычисления значения BW:

$$BW = \frac{1000000}{BW_i} \cdot 256, \quad (1)$$

где BW_i является наименьшей пропускной способностью из всех исходящих интерфейсов по пути в сеть назначения, представленная в килобитах.

Формула для вычисления значения $DELAY$:

$$DELAY = DELAY_i \cdot 256, \quad (2)$$

где $DELAY_i$ является суммой всех задержек, сконфигурированных на исходящих интерфейсах по пути в сеть назначения, в десятках микросекунд. Задержка, показываемая командой `show ip eigrp topology` или `show interface` в Cisco IOS (англ. Internetwork Operating System), указана в микросекундах, соответственно это значение нужно поделить на 10 перед использованием в этой формуле.

EIGRP использует полученные значения при подсчете общей метрики.

При вычислении метрики, когда $K_5 = 0$ (значение по умолчанию), используется формула

$$metric_1 = K_1 BW + \frac{K_2 BW}{256 - LOAD} + K_3 DELAY \quad (3)$$

Если значения коэффициентов K_1, K_2, K_3 равны значениям по умолчанию, то формула превращается в такую:

$$metric_1 = BW + DELAY \quad (4)$$

Если K_5 не равно 0, то дополнительно выполняется следующая операция:

$$metric_2 = metric_1 \cdot \frac{K_5}{REL + K_4} \quad (5)$$

Таким образом, составная метрика EIGRP позволяет тщательно (оптимально) настраивать работу сетей передачи данных.

1.3 Анализ предшествующих работ

Протоколы маршрутизации EIGRP и OSPF не адаптированы к резким увеличениям нагрузки на каналы на длительное время, что также буде продемонстрировано моделированием.

Решение (хотя бы частичное) этой проблемы имеет теоретическое и практическое значение для более эффективного использования ресурсов сети, это позволит уменьшить нагрузку на отдельные узлы сети и увеличить их отказоустойчивость, например за счет смены маршрута, по которому передается информация, на альтернативный, менее загруженный, и в то же время даст дополнительные возможности для дальнейшего анализа и улучшения работы протоколов.

Многие исследователи предлагают свои способы для смягчения последствий возникновения перегрузок в сетях. В ходе анализа имеющихся разработок в этой области было обнаружено, что для EIGRP исследования ведутся не так активно, как, например, для OSPF. Это может быть связано с тем, что протокол маршрутизации EIGRP является проприетарной разработкой фирмы Cisco и был открыт в начале 2013 года в качестве информационного RFC, а не как стандарт [4], это позволяет Cisco Systems, Inc. сохранять контроль над протоколом EIGRP. При этом компания Cisco оставила закрытыми детали функционирования и (или) реализации некоторых функций и особенностей EIGRP для сохранения и защиты опыта клиентов и их инвестиций в развертывание сетей [11; 12]. Этот факт может замедлять (ограничивать) внедрение и использование EIGRP в маршрутизаторы других производителей сетевого оборудования. Анализ некоторых из существующих решений как для EIGRP, так и для OSPF приведен ниже.

В [13] представлено решение задачи поиска кратчайших путей алгоритмом Беллмана-Форда, который положен в основу протоколов маршрутизации RIP, BGP, EBGP, IBGP, IGRP, EIGRP. Модифицированный алгоритм Беллмана-Форда для построения резервных путей использует данные о ребрах, входящих в узлы, его использование в составе протоколов маршрутизации позволяет повысить устойчивость телекоммуникационной системы на величины, пропорциональные топологической сложности и связности ее сети. Например, прирост устойчивости сети по показателю среднесетевая вероятность устойчивости ин-

формационного направления связи составил 11 %. Это достигается заблаговременным формированием сети резервных путей и быстрым переходом на резервные каналы без затрат времени на поиск новых путей.

Есть работы, направленные на повышение информационной безопасности при передаче данных в EIGRP-сетях. Использование параметра риска информационной безопасности в формуле расчета метрики протокола EIGRP для маршрутизации трафика по самым безопасным маршрутам в сети предлагается в [14].

Здесь риск рассчитывается на основе двух составляющих: риск на основе стандарта NIST CVSS и риск, вычисляемый на основе формулы уровня уязвимости узла из теории живучести информационных систем. Это позволяет рассматривать как информационную безопасность маршрутизируемых пакетов, так и структурную целостность сети. Также предлагается модифицированный алгоритм распределения нагрузки между маршрутами, позволяющий разгрузить наиболее эффективный узел маршрутизации, когда на сеть производится DoS-атака.

Результаты исследования [14] показывают, что предлагаемый подход может быть использован для увеличения вероятности предотвращения нарушения информационной безопасности маршрутизируемых пакетов и для обеспечения безопасности наиболее эффективных узлов маршрутизации в сети, которые позволяют эффективно маршрутизировать доверенный трафик, когда сеть находится под DoS-атакой, или в ней отсутствуют критические системные ресурсы.

Авторами статьи [15] было предложено повысить производительность традиционного EIGRP путем добавления некоторых SDN-функций (англ. Software Defined Networking, программно-определяемая или программно-конфигурируемая сеть) в виде модифицированного подхода для улучшения некоторых показателей производительности сети (перегрузка каналов и уровень потерь пакетов), что приводит к увеличению (оптимизации) общей пропускной способности сети. Этот подход приводит к созданию сети, где есть интеллектуальный динамический контроллер-корректор (англ. dynamic supervisory corrector controller), способный обнаруживать точку перегруженности до или в начале ее появления. Контроллер обрабатывает выбранные потоки данных на выбранных маршрутизаторах по всей сети, чтобы предотвратить перегруженность с помощью интеллектуального эвристического алгоритма предотвращения перегрузки и алгоритма маршрутизации, тем самым максимально уменьшая генерацию управляющих сообщений.

В [16] решался вопрос эффективного распределения входных информационных потоков в магистральных IP-сетях при использовании протокола OSPF с целью улучшения робастности алгоритма Дейкстры по отношению к перегрузкам в информационных сетях.

Многочисленные испытания нового алгоритма Дейкстры совместно с алгоритмом робастной коррекции осуществлялись методом Монте-Карло. Моделирование проводилось с предельно высокой сетевой нагрузкой.

В результате был предложен алгоритм, в основу которого заложен алгоритм Дейкстры, существенно снижающий вероятность перегрузок. Это достигается за счет использования в критерии распределения вместо пропускной способности канала его остаточной пропускной способности. Суть предложенного обобщенного алгоритма Дейкстры заключается в следующем. При распределении очередной доли входного потока $Q_r(t)$ (матрица $Q(t)$ определяет многоадресный поток данных всех маршрутизаторов сети) в критерий качества включается информация о занятости канала связи ранее распределенными долями входного потока. Поэтому текущее распределение части потока будет учитывать реальную пропускную способность IP-сети. Моделирование нового алгоритма показало существенное улучшение минимаксного критерия качества (метрики) с ростом разбиений входного потока информации.

Эксперименты по сравнению качества функционирования предложенного в работе [16] алгоритма совместно с алгоритмом робастной коррекции и распределения нагрузки в IP-сетях с помощью метода линейного программирования позволяют сделать вывод о высокой степени эффективности модифицированного алгоритма. К недостатку следует отнести отсутствие достаточной гарантии от сетевых перегрузок. Для разреженных IP-сетей с числом узлов 27 и больше высокую эффективность нового алгоритма по сравнению с алгоритмом линейного программирования показать не удалось.

В [17] авторами статьи был поставлен вопрос о возможности повышения устойчивости связи при маршрутизации (устойчивость относится к одним из основных свойств сетей связи) путем модификации алгоритма Дейкстры, позволяющей одновременно с решением задачи поиска кратчайших путей сформировать резервные пути к узлам сети. Подход этой работы был взят за основу при модификации алгоритма Беллмана-Форда в [13].

Методы: в интересах использования топологической избыточности сети связи модифицируется алгоритм Дейкстры в направлении расширения его функциональности за счет формирования как кратчайших, так и резервных путей. Данное расширение обеспечивается введением дополнительных множеств в расчет, а также новых блоков в тело алгоритма.

В результате была проведена модификация алгоритма Дейкстры, основанная на использовании входящих в узлы ребер для построения резервных путей к узлам. Оценка прироста устойчивости сети связи осуществлена по показателю вероятности устойчивости информационного направления. Рассмотрена работа алгоритма на примере сети, и показано, что его применение дает повышение устойчивости от 5 до 35 % по обоснованному показателю. Повышение устойчивости информационно-телекоммуникационных сетей (ИТКС) по показателю вероятность устойчивости информационного направления происходит за счет одновременного повышения показателей коэффициент готовности информационного направления (благодаря снижению временных задержек восстановления информационного направления связи за счет переключения информационных потоков) и вероятность выживания информационного направления в результате

деструктивно-разрушающих воздействий (достигается введением резервных путей). Прирост эффективности, получаемый при использовании модифицированного алгоритма Дейкстры, возрастает при увеличении топологической сложности сети и при снижении длительностей временных параметров конкретных протоколов маршрутизации.

Предложенная в [17] модификация алгоритма Дейкстры может быть использована для совершенствования (улучшения эффективности) протоколов маршрутизации PNNI и OSPF в целях обеспечения заданного уровня устойчивости сетей ATM и TCP/IP в условиях деструктивно-разрушающих воздействий на сетевые элементы.

В [18] рассматривается вопрос гарантированного распределения нагрузки в сетях под управлением протокола OSPF с помощью применения планируемой адаптивной стратегии маршрутизации, которая опиралась бы на оценку пропускной способности и состояния каналов. Представлены результаты исследований 3-х алгоритмов адаптивной балансировки трафика в сети, которая функционирует по протоколу маршрутизации OSPF, и результаты выявления их недостатков и преимуществ при моделировании.

Каждый из описанных алгоритмов имеет свои особенности в описании и разработке и был протестирован определенным образом. Алгоритм адаптивной балансировки через оценку эффективной пропускной способности (алгоритм № 1) моделировался в среде OPNET Modeler. Алгоритм адаптивной и распределенной балансировки в сети OSPF (алгоритм № 2) был испытан на случайных сетях, генерируемых механизмом, описанным в одном из использованных в работе источников, при различных сценариях трафика. Модифицированный алгоритм адаптивной балансировки в сети OSPF на основе нечеткой логики (алгоритм № 3) тестировался в симуляторе ns-2 на различных уровнях нагрузки в сети.

Для алгоритма № 1 качество сети оценивалась экспериментально путем моделирования с точки зрения средней пропускной способности сети, задержки передачи пакетов и скорости потери пакетов. С целью сравнения традиционный метод маршрутизации OSPF было принят за эталон. Было подтверждено, что адаптивная стратегия маршрутизации принимает лучшие решения, чем при эталонном OSPF. Этот результат был интуитивно ожидаемым в связи с тем, что адаптивный алгоритм маршрутизации может лучше распределить нагрузку трафика среди сетевых соединений. Непосредственным следствием этого факта является высокая пропускная способность сети, сохраняемая даже при высокой интенсивности трафика. Главным недостатком этого алгоритма является большая задержка за счет дополнительной нагрузки сообщениями об обновлении стоимости канала. Но дополнительная задержка, сформированная для сообщений, имеет незначительное влияние по сравнению с введенными альтернативными маршрутами. По темпам потерь пакетов использование адаптивной стратегии маршрутизации становится еще более полезным, когда в сети преобладают потоки с высокой скоростью. Если сравнивать алгоритм № 1 без модификаций с алгоритмами № 2 и № 3, то, несмотря на свою более простую математическую

модель и неэффективность при малых нагрузках на сеть, алгоритм охватывает больше параметров качества обслуживания и гарантирует большее использование сетевых ресурсов (по сравнению с алгоритмом № 2). В алгоритме № 2 определяющей особенностью является использование функций задачи оптимизации (неоптимальной или субоптимальной) и функции статической задачи балансировки нагрузки (оптимальной). Использование этих задач позволяет сформировать критерии и более гибко рассчитать параметры для изменения весовых коэффициентов канала. В алгоритме № 3 используются функции принадлежности, что позволяет получить наиболее оптимальный результат, основанный на предварительной сборке и результатах. В работе предлагается модификация, целью которой является уменьшение количества и частоты изменений стоимости каналов, что позволяет избежать резких колебаний трафика, но модификация не привела к существенным улучшениям показателей качества, однако существенно уменьшила количество изменений коэффициентов стоимости каналов. Также удалось уменьшить количество потерь пакетов на 20 %.

Задача OSPF-маршрутизации относится к NP-сложным задачам. Предложенные в [18] алгоритмы показывают возможность создания таких алгоритмов, которые позволяли бы адаптивно и динамически управлять трафиком в сети и достичь стабильности и конвергентности в наиболее короткий промежуток времени, высокого уровня использования сетевых ресурсов со снижением нагрузки на отдельные каналы и уменьшением потери пакетов.

В [19] поднимается вопрос о повышении эффективности использования протокола OSPF в корпоративных сетях в условиях динамически изменяющихся нагрузок на линиях связи за счет уменьшения трудоемкости расчета таблиц маршрутизации.

Для подтверждения правильности предлагаемого алгоритма на базе протокола OSPF была разработана программа имитационного моделирования процессов маршрутизации в корпоративных сетях. При разработке основное внимание уделялось корректности предлагаемого алгоритма и размерности решаемой задачи (выражается через количество вершин, для которых необходим поиск кратчайшего пути; корпоративная сеть здесь представляется в виде неориентированного взвешенного связного графа).

Для повышения эффективности использования протокола OSPF на его базе был предложен алгоритм парных перестановок маршрутов в корпоративных сетях, позволяющий за счет сбора дополнительной информации учесть возможные изменения конфигурации корпоративной сети и не производить полный пересчет маршрутных таблиц, а также уменьшить трудоемкость построения (расчета) таблиц маршрутизации с величины $O(N^2)$ (при выборе оптимального маршрута по алгоритму Дейкстры), где N – число маршрутизаторов в корпоративной сети, до величины $O(N)$. В ходе имитационного моделирования на разных размерностях задачи (графы, состоящие из 10, 100 и 500 вершин) было показано, что математическое ожидание числа изменений не превышает величины $N/2$, а его максимальное значение не превышает N. На основе этого был сделан вывод, что

предложенный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF является эффективным при поиске оптимальных маршрутов в условиях частичных динамических изменений на линиях связи корпоративной сети за счет использования дополнительной информации о возможных маршрутах замены.

Таким образом, разработанный алгоритм парных перестановок маршрутов на базе протокола OSPF позволяет уменьшить трудоемкость построения таблиц маршрутизации до величины порядка $O(N)$, из-за чего повышается эффективность функционирования корпоративных сетей в условиях динамически изменяющихся нагрузок на линиях связи в этих сетях.

В [20] автор задался вопросом повышения энергоэффективности сетевых устройств при маршрутизации. В частности, рассматривается новая идея энергоэффективного расширения протокола OSPF.

Для экспериментов используется среда моделирования компьютерных сетей OMNeT++.

В итоге предложена новая концепция протокола OSPF, основанная на алгоритме Дейкстры, который пересчитывает кратчайший путь каждый раз, когда обновляется метрика. В OSPF вводится новое значение метрики протокола в зависимости от интенсивности трафика на интерфейсах маршрутизаторов. Это расширение протокола реализовано в OMNeT++. Были произведены тесты для улучшения OSPF-симулятора в среде OMNeT++, в дальнейшем он будет совершенствоваться: неиспользуемые интерфейсы маршрутизаторов могут быть отключены или переключены на разные энергетические состояния (в зависимости от нагрузки на них), благодаря этому энергия может быть сохранена, а компьютерная сеть сможет работать более эффективно.

Работа по расширению модуля OSPF в среде OMNeT++ еще продолжается. Цель предложенного решения состоит в контроле энергетических состояний интерфейсов маршрутизатора. Энергоэффективность (для данной работы) понимается как поддержание процесса балансировки нагрузки для интерфейсов маршрутизатора. С помощью новой функциональности роутеры смогут обмениваться информацией о потреблении энергии на интерфейсах, что позволит изменять их состояния и в конечном итоге сохранить энергию и сократить расходы в больших компьютерных сетях связи.

В то время как работоспособность и эффективность рассмотренных алгоритмов подтверждена, детали их реализации недоступны, методы [13; 14; 17] нигде не реализованы. Представленные в [16; 17; 18; 19] алгоритмы внедрены для использования в отличных от OMNeT++ средах моделирования. Работа [20], реализуемая для OMNeT++, все еще не закончена, акцент здесь сделан на повышение энергоэффективности сети, а не на улучшение работы OSPF при возникновении пиковых нагрузок. В [14] для оптимизации используются параметры, учитывающие риски информационной безопасности. Авторы работы [15] указывают, что при очень высоких уровнях трафика применяемый внутри SDN-контроллера алгоритм не позволяет полностью защитить сеть от потерь пакетов,

здесь также вводится дополнительный управляющий узел, который не всегда реально найти. В [16] отсутствует достаточная гарантия от сетевых перегрузок, для сетей с большим числом узлов не удалось показать высокую эффективность нового алгоритма. Одна из предлагаемых в [18] модификация не привела к значимым улучшениям показателей качества, а разработанный в [19] алгоритм предназначен для функционирования в корпоративных сетях, обычно являющимися относительно небольшими по размеру. Решения, предложенные в [16; 17] актуальны для протоколов маршрутизации на основе алгоритма Дейкстры, а в [18; 19; 20] – конкретно для OSPF.

Как видно, для протокола маршрутизации OSPF есть много разных способов его оптимизации с точки зрения распределения нагрузки на сеть, а EIGRP еще недостаточно исследован. Таким образом, есть необходимость в создании, описании и реализации метода оптимизации протокола маршрутизации EIGRP при работе в сетях с большим уровнем трафика. Вносить модификации было решено именно в EIGRP.

2 Модификация протокола EIGRP для оптимизации его работы в сетях под пиковыми нагрузками

2.1 Выбор критерия эффективности работы сети

Для функционирования сети самым эффективным образом и для ее оптимизации нужно выбрать критерий эффективности работы сети. Наиболее часто используемые критерии можно разделить на несколько групп. Первая из них характеризует производительность работы сети, вторая – надежность, то есть способность сети правильно функционировать в течение длительного времени (достигается, например, слежением за количеством отказов).

Производительность сети измеряется с помощью показателей двух типов – временных, оценивающих задержку, вносимую сетью при выполнении обмена данными, и показателей пропускной способности, отражающих количество информации, переданной сетью в единицу времени [21]. Эти показатели взаимно обратны, зная один из них, можно вычислить другой.

Обычно временной характеристикой производительности сети выбирается время реакции, в общем случае, это интервал времени между возникновением запроса пользователя к какому-либо сетевому сервису и получением ответа на этот запрос. В качестве примера показателя времени реакции можно привести RTT (англ. Round Trip Time) – время между отправкой запроса и получением ответа, замеряемое утилитой ping, которая отправляет запросы ICMP Echo-Request протокола ICMP по указанному адресу и фиксирует поступающие ответы ICMP Echo-Reply. Определив RTT по маршруту и частоту потери пакетов, можно косвенно судить о загруженности на каналах передачи данных и промежуточных устройствах. Также RTT связано с длиной очереди буфера на интерфейсах

устройства, потому что ее увеличение может привести к «заторам» в обработке соединений и, соответственно, к слишком большим значениям RTT.

Значение пропускной способности хорошо отражает качество выполнения сетью ее основной задачи – быстрой передачи информации между устройствами. Определить этот критерий можно множеством вариантов, которые отличаются друг от друга выбранной единицей измерения количества передаваемой информации, характером учитываемых данных – только пользовательские или же пользовательские вместе со служебными, количеством точек измерения передаваемого трафика, способом усреднения результатов на сеть в целом [21].

Измерять передаваемую информацию можно в пакетах (кадрах) или в битах, то есть пропускная способность может быть выражена в пакетах в секунду или в битах в секунду. Первый вариант является более предпочтительным, так как в этом случае можно определить используемый пакетами протокол и их размер (чаще всего подразумевают пакеты протокола Ethernet, как самого распространенного [21]) и отделить пользовательские данные от служебных.

Если пропускная способность измеряется без деления информации на пользовательскую и служебную, то в этом случае нельзя ставить задачу выбора протокола или стека протоколов для данной сети, но, если тип протокола не меняется при настройке сети, можно использовать и критерии, не выделяющие пользовательские данные из общего потока [21].

Пропускная способность может быть измерена между любыми двумя узлами или точками сети. Полную характеристику пропускной способности сети дает набор пропускных способностей, измеренных для различных сочетаний взаимодействующих компьютеров, – матрица трафика узлов сети. Для этого есть специальные средства измерения, которые фиксируют матрицу для каждого узла сети.

Так как в сетях данные на пути до узла назначения обычно проходят через несколько транзитных промежуточных этапов обработки, то в качестве критерия эффективности может рассматриваться пропускная способность отдельного промежуточного элемента сети – отдельного канала, сегмента или коммуникационного устройства [21].

Знание общей пропускной способности не дает полной информации о возможных путях ее повышения, так как из общей цифры нельзя понять, какой из промежуточных этапов обработки пакетов в наибольшей степени тормозит работу сети. Поэтому данные о пропускной способности отдельных элементов сети могут быть полезны для принятия решения о способах ее оптимизации [21].

Для предлагаемого метода повышения производительности сети под нагрузкой в качестве критерия эффективности работы сети выбрана текущая пропускная способность, так как этот показатель хорошо характеризует скорость передачи данных между узлами, фиксация значений параметра является наиболее простым и распространенным процессом. Измерения происходят на каждом EIGRP-маршрутизаторе (на всех интерфейсах) в битах в секунду без разделения трафика.

2.2 Модификация для оптимизации протокола маршрутизации, алгоритм работы

Предлагается следующая модификация в работу протокола EIGRP. После полной инициализации работы EIGRP в сети, когда она сошлась:

- на каждом интерфейсе роутера, который использует EIGRP в качестве протокола маршрутизации, производить расчет текущей пропускной способности (загрузки) за определенный интервал или за определенное количество пакетов в битах в секунду;
- если загрузка на одном из интерфейсов, например, в течение 1 секунды, меньше определенного порога, то необходимо изменить загрузку EIGRP-интерфейса на величину, соответствующую падению пропускной способности;
- при изменении состояния канала вызвать перерасчет маршрутов, в которые он входит, после чего оповестить об этом маршрутизаторы-соседи. В этом случае (то есть при изменении топологии) на каждом из оставшихся маршрутизаторов также должен произойти перерасчет требуемых маршрутов (при необходимости).

Блок-схема алгоритма для одного интерфейса представлена на рисунке 1.

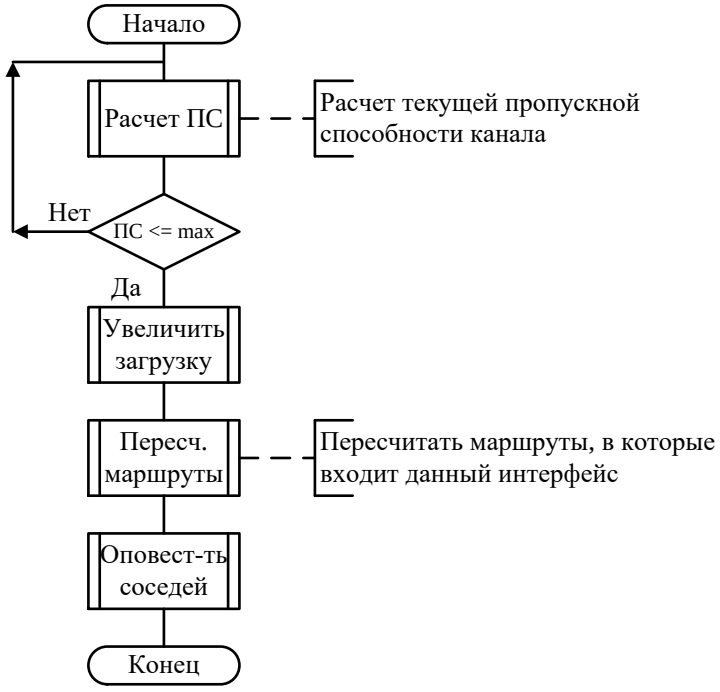

Рисунок 1 – Блок-схема алгоритма слежения за текущей пропускной способностью канала

Максимальный порог текущей пропускной способности, после преувеличения которого должна быть изменена загрузка интерфейса, другие важные для работы алгоритма значения и работа элементов, отвечающих за изменение загрузки и перестроение маршрутов, будут рассмотрены и определены позже.

2.3 Общие сведения об OMNeT++

Для проведения экспериментов в работе используется фреймворк OMNeT++ версии 5.0.

OMNeT++ (англ. Objective Modular Network Testbed in C++) [22] – расширяемая, модульная C++ библиотека моделирования на основе компонентов, предназначенная для создания сетевых симуляторов. Это продукт с открытым исходным кодом, является бесплатным только для академического и некоммерческого использования. Здесь сеть понимается в более широком смысле, включая в себя проводные и беспроводные сети связи, чипы, которые построены по архитектуре сеть-на-чипе (подразумевается, что каждое вычислительное ядро связано непосредственно только с ближайшими ядрами), сети массового обслуживания и так далее. Предметно-ориентированная функциональность (поддержка сенсорных сетей, беспроводных ad-hoc-сетей или беспроводных динамических (самоорганизующихся) сетей, интернет-протоколов, моделирования производительности, фотонных сетей и так далее) обеспечивается модельными фреймворками, разработанными в качестве самостоятельных проектов. OMNeT++ предоставляет IDE на основе Eclipse, графическую среду выполнения, а также множество других инструментов. Есть расширения для моделирования в реальном времени, эмуляции сети, интеграции с базами данных, с SystemC и ряд других функций.

OMNeT++ предоставляет компонентную архитектуру для моделей. Компоненты (модули) запрограммированы в C++, затем собраны в более крупные компоненты и модели с использованием языка высокого уровня (NED). Данный инструмент имеет расширенную поддержку графического интерфейса пользователя, и благодаря своей модульной архитектуре ядро моделирования (и модели) может быть легко встроено в пользовательские приложения.

OMNeT++ работает на Windows, Linux, Mac OS X и других UNIX-подобных системах и состоит из следующих компонентов:

- библиотека ядра моделирования;
- язык описания топологии NED (англ. NEtwork Description);
- IDE на базе платформы Eclipse;
- GUI для выполнения моделирования Tkenv, линкуется в исполняемый файл симуляции;
- интерфейс командной строки для выполнения моделирования (Cmdenv);
- утилиты (инструмент создания makefile и так далее);
- документация, примеры моделей и другое.

OMNeT++ предоставляет эффективные инструменты для пользователя, чтобы описать структуру реальной системы. К главным особенностям можно отнести иерархически вложенные модули, которые являются экземплярами типов модулей, и язык описания топологии NED. Модули связываются сообщениями по каналам, имеют гибкие параметры.

Модель OMNeT++ состоит из следующих частей:

- описания топологий на языке NED (файлы с расширением .ned), которые описывают структуру модуля с параметрами, шлюзами и так далее;

- определения сообщений (.msg файлы). Можно определить различные типы сообщений и добавить поля данных в них. OMNeT++ автоматически переведет определения сообщений в полноценные классы C++;

- исходные коды модуля представляют собой C++ файлы с расширением .h или .cc.

Система моделирования предоставляет такие компоненты, как ядро моделирования, которое управляет моделированием и классовыми библиотеками моделирования, и пользовательские интерфейсы.

Программы моделирования построены из вышеуказанных компонентов. Сначала .msg файлы преобразуются в код на C++ с использованием компилятора сообщений `opp_msgc`. После все исходники на C++ компилируются и линкуются с ядром моделирования и библиотекой пользовательского интерфейса для формирования исполняемого файла моделирования или общей библиотеки. NED-файлы загружаются динамически в исходных текстовых формах при запуске программы моделирования.

После запуска программа считывает все файлы NED, потом – конфигурационный файл (обычно называется `omnetpp.ini`). Этот файл содержит параметры, которые определяют, как выполняется моделирование, значения параметров модели и так далее.

Процесс построения и запуска программ моделирования представлен на рисунке 2.

Выходные данные моделирования записываются в файлы результатов, которые могут быть проанализированы с помощью OMNeT++: выходные файлы векторов (запись данных во время выполнения моделирования), выходные файлы скаляров (итоговые результаты, вычисленные во время моделирования и записанные после завершения моделирования), выходные файлы, определенные пользователем.

OMNeT++ может быть расширен с помощью специальных библиотек моделирования. Наиболее известной и распространенной является INET Framework [23], исходный код которой открыт. INET предоставляет протоколы, агенты и другие модели для исследователей и студентов, работающих с сетями передачи данных. INET особенно полезна при разработке и утверждении новых протоколов, при изучении новых экзотических сценариев.

INET содержит модели для стека протоколов Интернета (TCP, UDP, IPv4, IPv6, OSPF, BGP и так далее), для проводных и беспроводных протоколов канального уровня (Ethernet, PPP, IEEE 802.11 и так далее), для поддержки мобильности, протоколов MANET (Mobile Ad hoc Network), DiffServ, MPLS с LDP и RSVP-TE-сигнализацией, включает несколько моделей приложений, много других протоколов и компонентов.

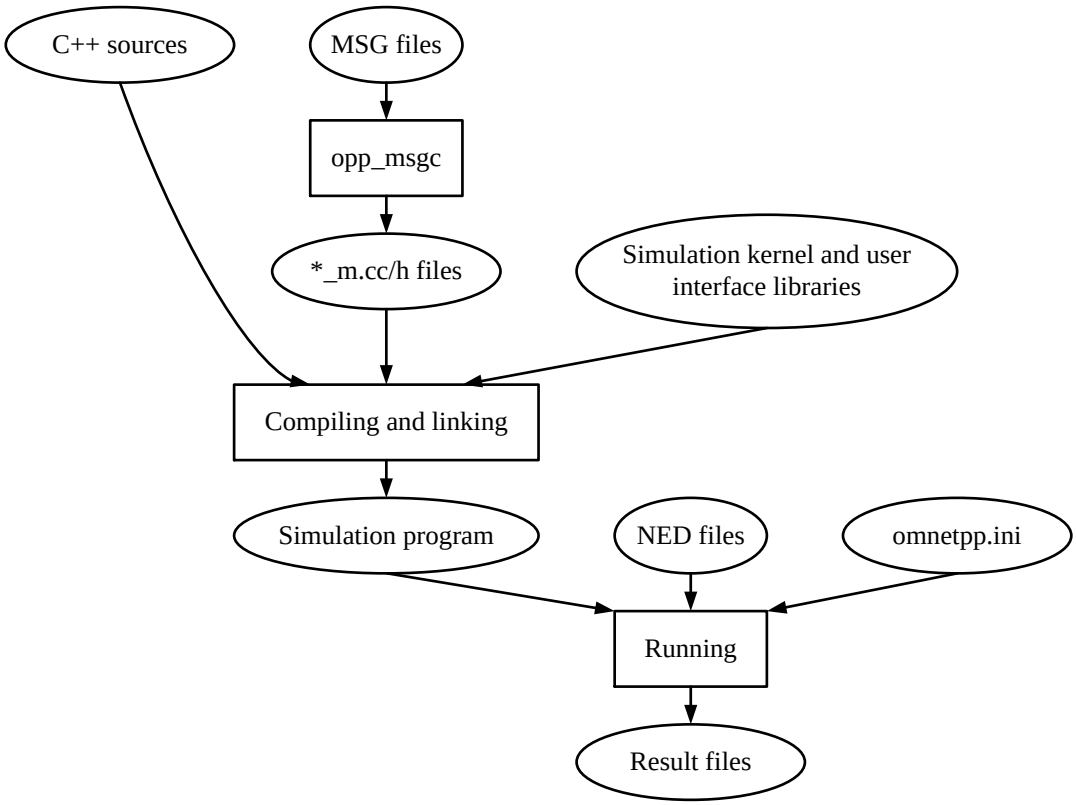

Рисунок 2 – Обзор процесса построения и запуска программ моделирования, созданных в OMNeT++

Существует множество основанных на INET фреймворков моделирования, которые поддерживаются независимыми исследовательскими группами. К числу таких библиотек относится ANSAINET (англ. Automated Network Simulation and Analysis (ANSA)) [24] – проект исследователей и студентов Технологического университета Брно факультета информационных технологий в Чехии. Он посвящен разработке различных имитационных моделей, совместимых с спецификациями RFC или эталонными (базовыми) реализациями (англ. referential implementations), расширяя тем самым проводной сетевой IP-фреймворк INET (поэтому исходный код называется ANSAINET). Впоследствии эти модули и связанные с ними инструменты могли бы позволить производить формальный анализ реальных сетей и их конфигураций. ANSAINET может свободно использоваться в качестве основы для маршрутизации/коммутации для дальнейших исследовательских инициатив, то есть в симуляциях, доказывающих (или опровергающих) определенные аспекты сетевых технологий (например, поиск узких мест и единых точек отказа, ошибок конфигурации, неисправных состояний сети и так далее).

Основные цели проекта ANSAINET:

- разработать точные имитационные модули (совместимые с базовыми реализациями) для протоколов, используемых в традиционных проводных TCP/IP-сетях;

- популяризовать использование симуляторов в качестве средств верификации и валидации при развертывании сетей и разработки протоколов;

- внести вклад и оказать поддержку в сообществе OMNeT ++, интегрируя популярные функции ANSAINET в оригинальный INET.

В ANSAINET поддерживаются следующие протоколы:

- HSRP, VRRPv2, GLBP;
- IS-IS;
- RIPv2 и RIPng, EIGRP, Babel;
- CDP, LLDP;
- STP, TRILL;
- LISP;
- PIM-DM, PIM-SM, IGMPv2 и IGMPv3.

Для моделирования работы компьютерной сети с протоколом EIGRP будет использоваться библиотека ANSAINET версии 3.4.0, установка описана в [24]. Чтобы использовать возможности фреймворка ANSAINET, в свойствах проекта нужно настроить ссылку на него (зайти в Preferences нажатием правой кнопки мыши на проекте, далее – в Project References, выбрать ANSA-ansainet-3.4.0 и нажать OK). Окно настроек для проекта с выбором данной опции представлено на рисунке 3.

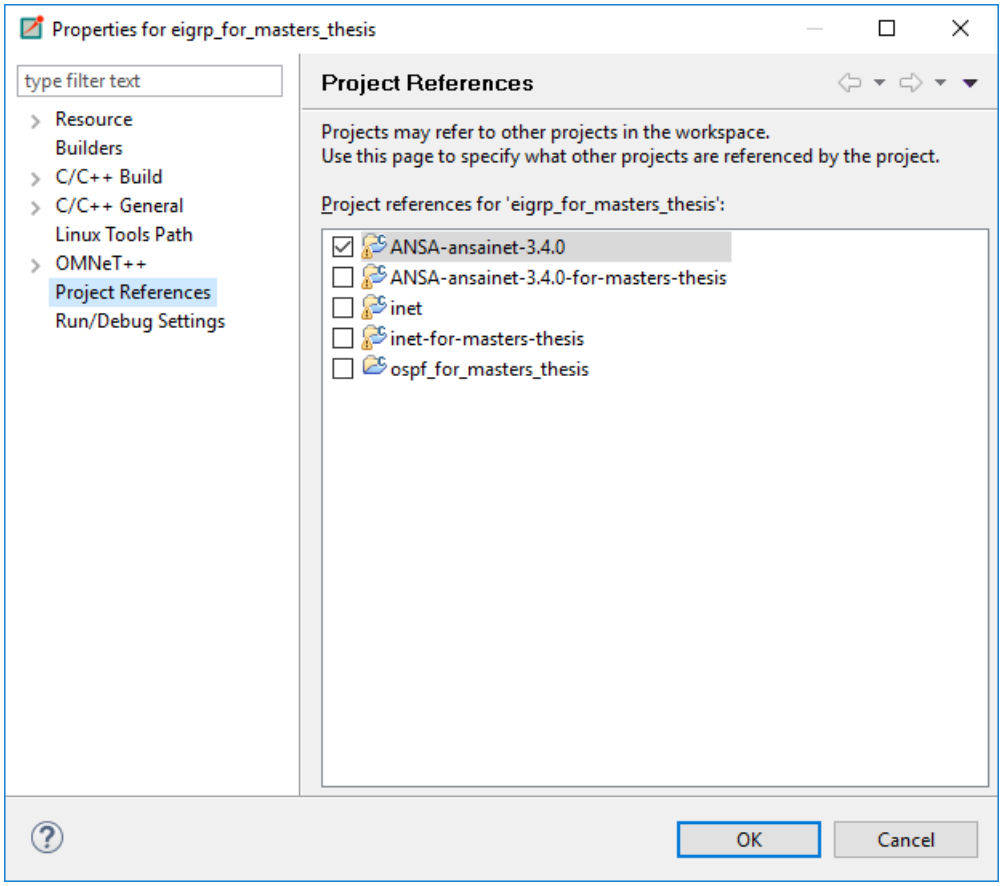

Рисунок 3 – Выбор ANSA-ansainet-3.4.0 для использования в проекте OMNeT++

2.4 Реализация метода в библиотеке ANSAINET для OMNeT++

В ходе реализации предложенного метода были затронуты следующие исходные классы и файлы библиотеки ANSAINET:

- `class INET_API EtherEncap` (файлы `EtherEncap.h` и `EtherEncap.cc`), осуществляет инкапсуляцию и декапсуляцию Ethernet-фреймов;
- `class INET_API IPv4` (файл `IPv4.cc`), который реализует протокол IPv4;
- `class INET_API ANSA_MultiNetworkLayerLowerMultiplexer` (файл `ANSA_MultiNetworkLayerLowerMultiplexer.cc`);
- `class EigrpIpv4Pdm` (файлы `EigrpIpv4Pdm.h` и `EigrpIpv4Pdm.cc`), представляющий собой EIGRP-модуль, зависимый от протокола, для IPv4;
- `class EigrpIpv6Pdm` (файл `EigrpIpv6Pdm.cc`), представляющий собой EIGRP-модуль, зависимый от протокола, для IPv6;
- `class EigrpMetricHelper` (файл `EigrpMetricHelper.cc`) для вычисления метрики EIGRP;
- `class INET_API EigrpDeviceConfigurator` (файл `EigrpDeviceConfigurator.cc`), отвечающий за начальную конфигурацию EIGRP-сети;
- файл `EigrpMessage.msg`, в котором описываются EIGRP-пакеты.

Для того чтобы EIGRP учитывал загрузку интерфейса при вычислении метрики, нужно установить значение коэффициента K2 в 1 в следующих файлах протокола EIGRP:

- `EigrpMessage.msg`:

```
// Struct for K-values
struct EigrpKValues
{
      ...
      uint16_t K2 = 1;
      ...
}
```

- `EigrpIpv4Pdm.cc`:

```
EigrpIpv4Pdm::EigrpIpv4Pdm() : EIGRP_IPV4_MULT(
                                IPv4Address(224, 0, 0, 10))
{
    ...
    kValues.K1 = kValues.K2 = kValues.K3 = 1;
    kValues.K4 = kValues.K5 = kValues.K6 = 0;
    ...
}
```

- `EigrpIpv6Pdm.cc` – аналогично `EigrpIpv4Pdm.cc`;

- EigrpDeviceConfigurator.cc:

```
void EigrpDeviceConfigurator::loadEigrpProcessesConfig(
     cXMLElement *device, IEigrpModule<IPv4Address> *eigrpModule)
{
    ...
    kval.K2 = loadEigrpKValue((*procElem), "k2", "1");
    ...
}
```

и аналогично в функции `void EigrpDeviceConfigurator::loadEigrpProcesses6Config(cXMLElement *device, IEigrpModule<IPv6Address> *eigrpModule)`.

Тогда формула метрики (3) примет такой вид:

$$metric_1 = BW + \frac{BW}{256 - LOAD} + DELAY \quad (6)$$

В файл `ANSA_EIGRP_Router.ned`. описывающего структуру EIGRP-маршрутизатора (рисунок 4), были добавлены переменные `bool isThresholdPassed` и `int ifIndex`, первая из которых служит для предупреждения о том, что пропускная способность сильно упала и нужно предпринять требуемые действия, а вторая – для сохранения номера интерфейса, на котором наблюдается снижение данного параметра, для дальнейшего изменения уровня нагрузки на нем в подмодуле `eigrpIpv4Pdm` модуля `eigrp` типа `EigrpProcessDS` (рисунок 5). Описание роутера в этом файле выглядит так:

```
package ansa.node;

package ansa.node;

import ansa.node.ANSA_Router;

module ANSA_EIGRPRouter extends ANSA_Router
{
    parameters:
        hasEIGRP = true;
        eigrp.configData = configData;

        bool isThresholdPassed = default(false);
        int ifIndex = default(-1);
}
```

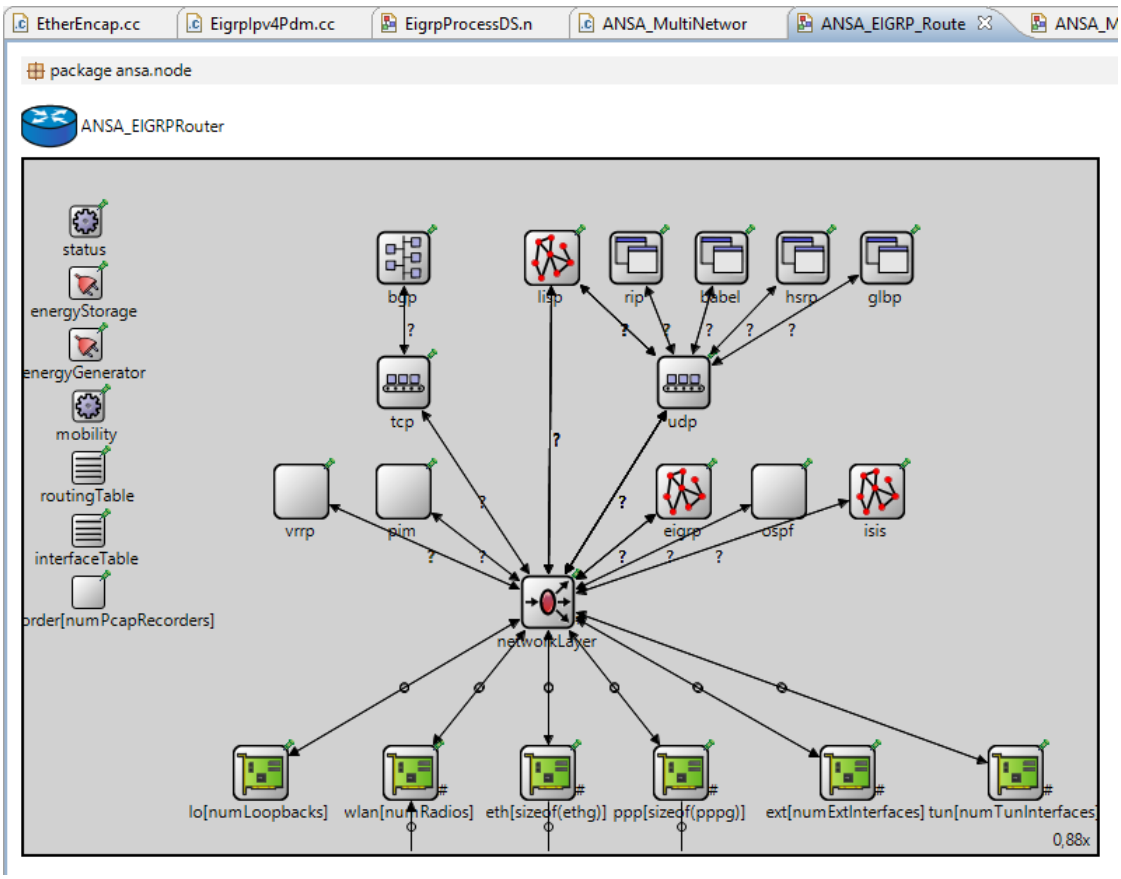

Рисунок 4 – Структура модуля `ANSA_EIGRPRouter`, описывающего структуру маршрутизатора EIGRP в OMNeT++

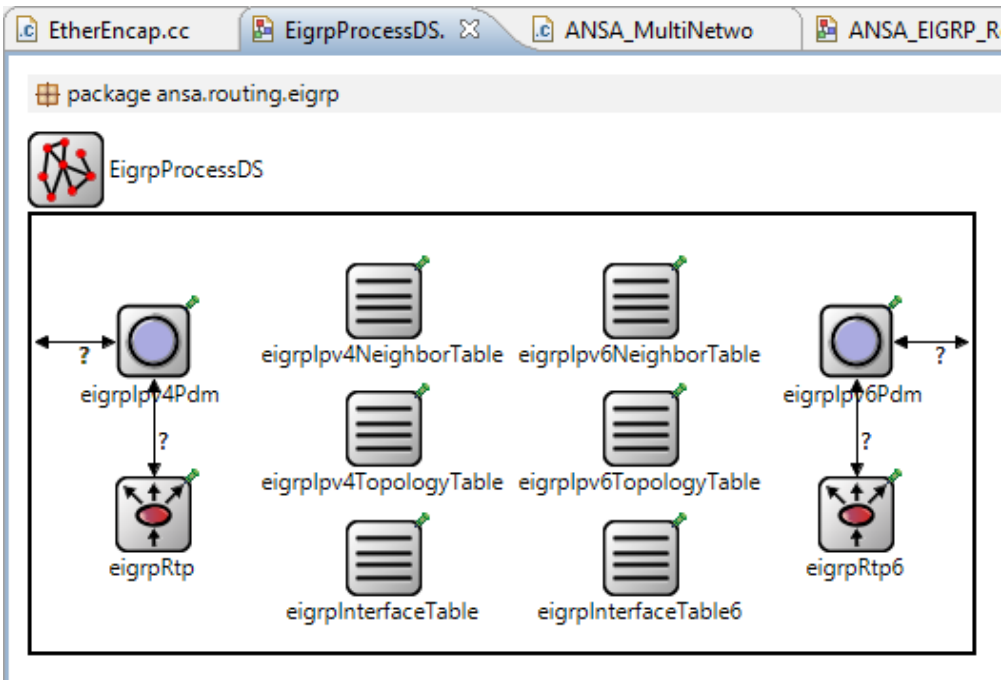

Рисунок 5 – Структура модуля `eigrp` типа `EigrpProcessDS`, описывающего процесс маршрутизации EIGRP

Пропускная способность рассчитывается на уровне Ethernet, значение имеет количество инкапсулированных фреймов. В класс `class INET_API EtherEncap : public cSimpleModule` были введены следующие дополнительные данные (свойства), описание которых приведено в таблице 3:

```
simtime_t startTime; // start time
unsigned int batchSize; // number of packets in a batch
// max length of measurement interval (measurement ends
// if either batchSize or maxInterval is reached, whichever
// is reached first)
simtime_t maxInterval;
simtime_t timestamp; // time when threshold passed
// current measurement interval
simtime_t intvlStartTime;
unsigned long packetsPerIntvl;
unsigned long bitsPerIntvl;
double threshold, // threshold value
       oldThreshold,
       newEigrpIfaceLoad; // new EIGRP interface load
bool shouldChangeLoad;
cModule *ethIface; // EthernetInterface module (network card)
```

Таблица 3 – Описание введенных для измерения пропускной способности переменных

Переменная	Описание переменной
`simtime_t startTime`	Время начала моделирования
`unsigned int batchSize`	Количество пакетов в партии, по пришествию которых начинается новый измерительный интервал
`simtime_t maxInterval`	Максимальная длина измерительного интервала
`simtime_t timestamp`	Время, когда максимальный порог пропускной способности был пройден
`simtime_t intvlStartTime`	Время начала нового интервала
`unsigned long packetsPerIntvl`	Количество пакетов за интервал
`unsigned long bitsPerIntvl`	Количество битов за интервал
`double threshold`	Пороговое значение
`bool shouldChangeLoad`	Нужно или нет увеличить загрузку на интерфейсе
`cModule *ethIface`	Указатель на текущий Ethernet-интерфейс, где происходит расчет пропускной способности

В методе этого класса `void EtherEncap::initialize()` новые данные инициализируются:

```
startTime = 0;
batchSize = 25;
```

```
maxInterval = 1;
timestamp = -1;
intvlStartTime = 0;
packetsPerIntvl = bitsPerIntvl = 0;
threshold = oldThreshold = 0;
newEigrpIfaceLoad = 1;
shouldChangeLoad = false;
ethIface = getParentModule();
WATCH(intvlStartTime);
WATCH(packetsPerIntvl);
WATCH(bitsPerIntvl);
WATCH(timestamp);
WATCH(shouldChangeLoad);
```

Размер «пачки» пакетов `batchSize`, равный 25, по приходу которых значение пропускной способности снова рассчитается, было подобрано в ходе экспериментов при моделировании, оно действительно для модели тестовой сети, приводимой в качестве примера в этой работе. Макрос `WATCH()` позволяет видеть значение переменной, указанной в качестве параметра к нему, в окне инспектора на вкладке `Contents` во время моделирования (рисунок 6).

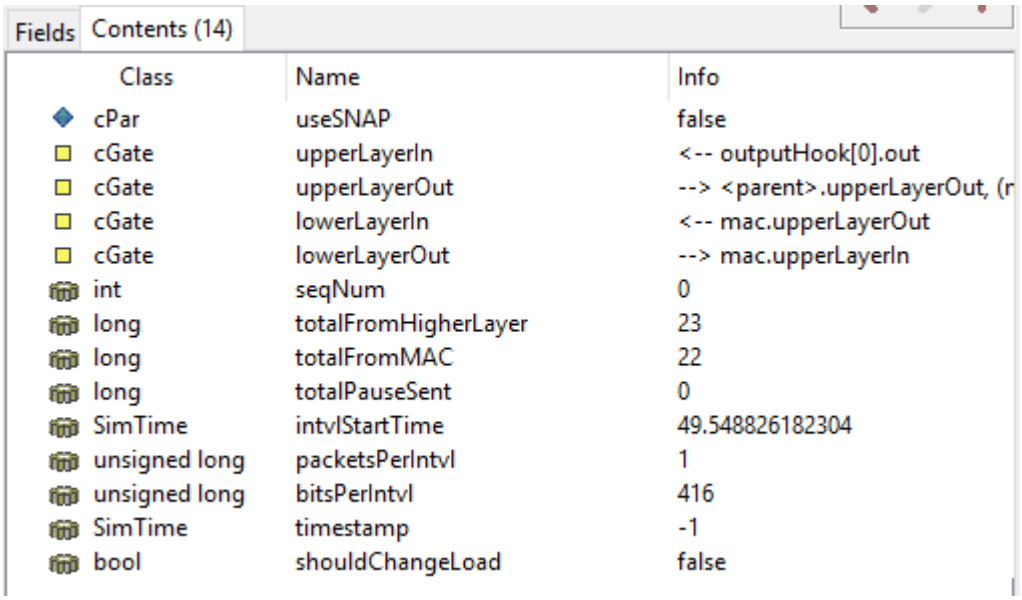

Рисунок 6 – Возможность просмотра значения переменной любого модуля в окне инспектора во время моделирования, обеспечивающаяся макросом `WATCH()`

Вычисление пропускной способности `double bitpersec` происходит в функции `void EtherEncap::processPacketFromHigherLayer(cPacket *msg)`, обрабатывающей пакеты от вышележащих уровней роутера:

```
// measure throughput only on an EIGRP router
if (strcmp(ethIface->getParentModule()->getModuleType()->getName(),
          "ANSA_EIGRPRouter") == 0)
{
    bool isThresholdPassed = ethIface->getParentModule()
                                     ->par("isThresholdPassed");

    simtime_t now = simTime();
    unsigned long bits = msg->getBitLength();

    // packet should be counted to new interval
    if (packetsPerIntvl >= batchSize
        || now - intvlStartTime >= maxInterval)
    {
        simtime_t duration = now - intvlStartTime;


        // record measurements
        double bitpersec = bitsPerIntvl / duration.dbl();
        //double pkpersec = packetsPerIntvl / duration.dbl();

        EV << "bitpersec = " << bitpersec
           << " at t = " << simTime() << "\n";

        // threshold passed, set necessary values
        if (threshold && bitpersec <= threshold
            && !isThresholdPassed)
        {
            timestamp = simTime();
            ethIface->getParentModule()
                    ->par("isThresholdPassed").setBoolValue(true);
            ethIface->getParentModule()
                    ->par("ifIndex").setLongValue(ethIface
                    ->getIndex());
            shouldChangeLoad = true;

            // determining a maximum channel data rate
            cGate *ethIfaceOutGate = ethIface->getParentModule()
                                             ->gate("ethg$o",
                                               ethIface
                                             ->getIndex());
            ASSERT(ethIfaceOutGate);
```

```
            cDatarateChannel *channel =
                check_and_cast<cDatarateChannel*>(
                    ethIfaceOutGate->getChannel());
            double channelDatarate = channel->getDatarate();

            // new EIGRP interface load normalization (feature
            // scaling) from <0-channel data rate> to <1-255>
            newEigrpIfaceLoad = (bitpersec - 0.) /
                                (channelDatarate - 0.) *
                                (255. - 1.) + 1.;
        }
        oldThreshold = bitpersec;
        // 30 %
        threshold = 0.70 * bitpersec;

        // restart counters
        intvlStartTime = now;
        packetsPerIntvl = bitsPerIntvl = 0;
    }
    packetsPerIntvl++;
    bitsPerIntvl += bits;

    // send TCN (Topology Change Notification)
    // (to ANSA_MultiNetworkLayerLowerMultiplexer)
    if (isThresholdPassed && shouldChangeLoad
        && simTime() >= timestamp)
    {
        // (EigrpProcessDS) eigrp
        cModule *eigrp = ethIface->getParentModule()
                                 ->getSubmodule("eigrp");

        char msgname[20];
        sprintf(msgname, "%s-to-%s-TCN", ethIface->getFullName(),
                eigrp->getName());

        // set TCN fields
        TplgyChngNtfctn *tcnMsg = new TplgyChngNtfctn(msgname);
        tcnMsg->setSource(ethIface->getFullName());
        tcnMsg->setDestination(eigrp->getName());
        tcnMsg->setShouldChangeTopology(shouldChangeLoad);
        tcnMsg->setIfIndex(ethIface->getIndex());
        tcnMsg->setNewEigrpIfaceLoad(newEigrpIfaceLoad);

        cGate *outGate = gate("upperLayerOut");
        send(tcnMsg, outGate);

        shouldChangeLoad = false;
    }
}
```

Если пропускная способность упала на 30 % и больше, то происходит фиксация этого момента времени в timestamp, параметр isThresholdPassed роутера устанавливается в значение true, а в ifIndex записывается номер интерфейса, на котором произошло падение производительности. Значение порога threshold = 0.70 * bitpersec; было выбрано таким во время экспериментов, оно соответствует создаваемому падению производительности в передаче данных во время моделирования и одинаково для всех моделей сетей.

Далее, если случилось сильное уменьшение пропускной способности и нужно поменять загрузку интерфейса, формируется сообщение TCN (англ. Topology Change Notification) типа TplgyChngNtfctn, сигнализирующее об изменении топологии сети, которое отправляется вышележащему уровню (модулю) устройства, в данном случае – модулю (ANSA_MultiNetworkLayer) networkLayer, отвечающему за работу протокола IPv4 и некоторых других протоколов сетевого уровня модели OSI. В этом сообщении дублируется номер интерфейса ifIndex и указывается новая загрузка для EIGRP-канала, которая соответствует падению пропускной способности (происходит нормирование получившейся величины пропускной способности bitpersec к диапазону от 1 до 255). Например, если максимальная скорость передачи данных канала равна 100 Мбит/с, и текущая скорость передачи данных – 50 Мбит/с, то новая нагрузка на EIGRP-интерфейс будет равна 127. Для создания и использования такого TCN-сообщения нужно было реализовать его в .msg файле.

Был создан файл TCN.msg:

```
namespace inet;

// Topology Change Notification
message TplgyChngNtfctn
{
    string source;
    string destination;
    bool shouldChangeTopology = false;
    short ifIndex = -1;
    double newEigrpIfaceLoad = 1;
}
```

При последующей компиляции автоматически создаются файлы TCN_m.cc и TCN_m.h с C++ классом class TplgyChngNtfctn : public ::omnetpp::cMessage, который можно использовать в модулях. Эти файлы содержат реализацию созданного класса (в том числе сеттеры и геттеры). Для использования класса нужно подключить сгенерированный заголовочный файл (это было сделано для всех затронутых в работе классов):

```
#include "inet/linklayer/ethernet/TCN_m.h"
```

Модуль сетевого уровня networkLayer EIGRP-маршрутизатора, в свою очередь, состоит из множества других подмодулей (рисунок 7), где тоже нужно обрабатывать любое пришедшее сообщение.

Для возможности корректной переотправки TCN-сообщения в этом модуле изменялись только подмодули lowerMultiplexer и ipv4. В подмодуле upperMultiplexer TCN будет правильно приниматься и отправляться без дополнительных манипуляций с исходным кодом класса этого подмодуля. Для lowerMultiplexer изменению подвергся метод void ANSA_MultiNetworkLayerLowerMultiplexer::handleMessage(cMessage *message) после

```
else if (!strcmp(arrivalGateName, "ifLowerIn")) {
```

Код:

```
// received message is a TCN message,
// send it to IPv4NetworkLayer module
if (dynamic_cast<TplgyChngNtfctn *>(message))
{
    cMessage *msgCopy = message->dup();
    delete message;
    send(msgCopy, "ifUpperOut",
         getProtocolCount() * arrivalGate->getIndex() + 0);
}
else
{
    int res = getProtocolIndex(message);
    ...
```

Когда посланное сообщение пришло к модулю ipv4, происходит его отправка к upperMultiplexer и далее к eigrp. Это осуществляется в функции void IPv4::handleMessage(cMessage *msg) после

```
else    if    (!msg->isSelfMessage()    &&    msg->getArrivalGate()
                ->isName("arpIn")) endService(PK(msg));
```

Код:

```
// received message is a TCN message, send it to
// ANSA_MultiNetworkLayerUpperMultiplexer module
else if (dynamic_cast<TplgyChngNtfctn *>(msg))
{
    cMessage *msgCopy = msg->dup();
    delete msg;
    cGate *outGate = gate("transportOut", 0);
    send(msgCopy, outGate);
}
```

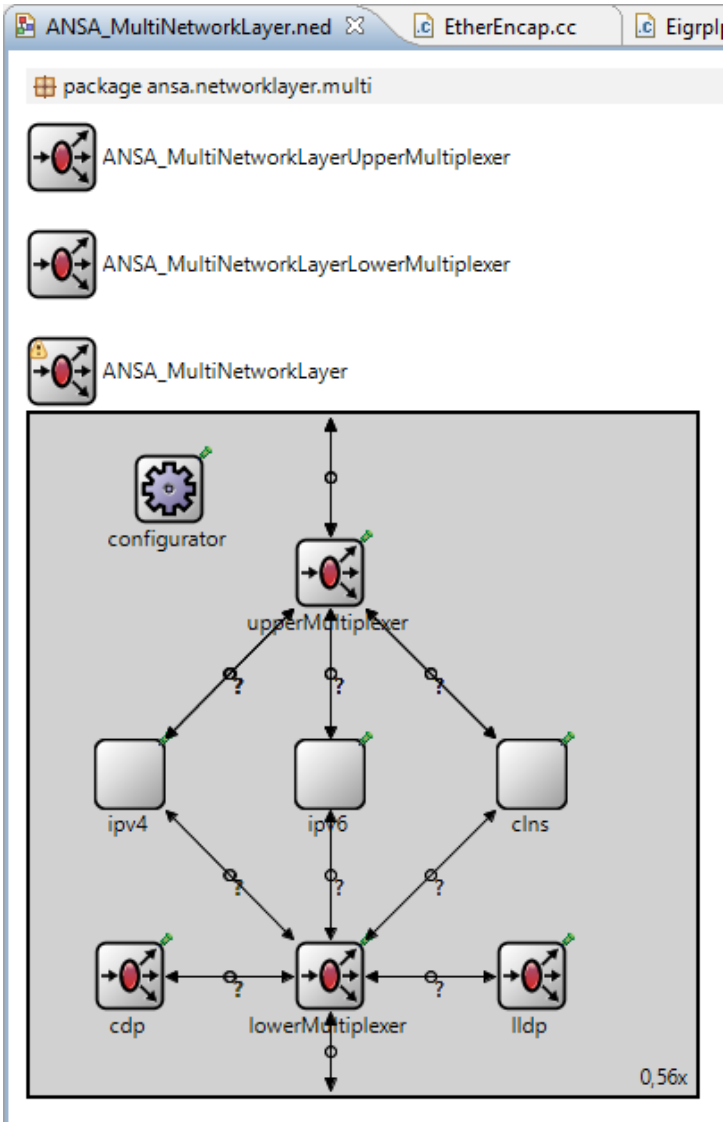

Рисунок 7 – Структура модуля `ANSA_MultiNetworkLayer`, отвечающего за работу различных протоколов сетевого уровня

При принятии модулем, зависимым от протокола, (`EigrpIpv4Pdm`) `eigrpIpv4Pdm` модуля (`EigrpProcess`) `eigrp` TCN-сообщения в функции обработки сообщений `void EigrpIpv4Pdm::handleMessage(cMessage *msg)` начинается перерасчет маршрутов, в которые входит данный интерфейс:

```
// received TCN, take necessary actions
if (dynamic_cast<TplgyChngNtfctn *>(msg))
{
    TplgyChngNtfctn *tcnMsg = check_and_cast<TplgyChngNtfctn *>
                              (msg);
    EV << "Received TCN: (" << tcnMsg->getClassName() << ")"
       << tcnMsg->getName() << "\n";
```

```
        if(tcnMsg->getShouldChangeTopology())
        {
            // try to find the necessary EIGRP interface
            // to change the load on it
            int ifaceId = tcnMsg->getIfIndex();
            EigrpInterface *eigrpIface =
              getInterfaceById(INTERFACEIDS_START + ifaceId);
            if (eigrpIface == NULL)
            {
                EV << "EIGRP interface eth" << ifaceId << "("
                   << INTERFACEIDS_START + ifaceId << ") is not found\n";
                delete msg;
                delete tcnMsg;
                return;
            }
            // get and set a new load
            int newEigrpIfaceLoad = tcnMsg->getNewEigrpIfaceLoad();
            if(newEigrpIfaceLoad != eigrpIface->getLoad())
            {
                eigrpIface->setLoad(newEigrpIfaceLoad);
                if (eigrpIface->isEnabled())
                    processIfaceConfigChange(eigrpIface);
                // else do nothing (interface is not part of EIGRP)
            }
        }
        delete msg;
        delete tcnMsg;
    }
```

Чтобы функция `processIfaceConfigChange(eigrpIface)` отработала правильно, требуется изменить функцию `EigrpWideMetricPar adjustParam(const EigrpWideMetricPar& ifParam, const EigrpWideMetricPar& neighParam)` класса `EigrpMetricHelper`:

```
newMetricPar.load = getMax(ifParam.load, neighParam.delay);
```

заменить на

```
newMetricPar.load = getMax(ifParam.load, neighParam.load);
```

С новыми маршрутами, полученными на основе отражающей действительность загрузки, данные будут передаваться по другому возможному пути, где уровень трафика может быть низким.

3 Моделирование

3.1 Обобщенная структура сети

Недостаток протоколов EIGRP и OSPF при работе в сетях под нагрузками, который описан ранее в диссертации, наглядно можно показать на сетях с общей структурой, приведенной на рисунке 8.

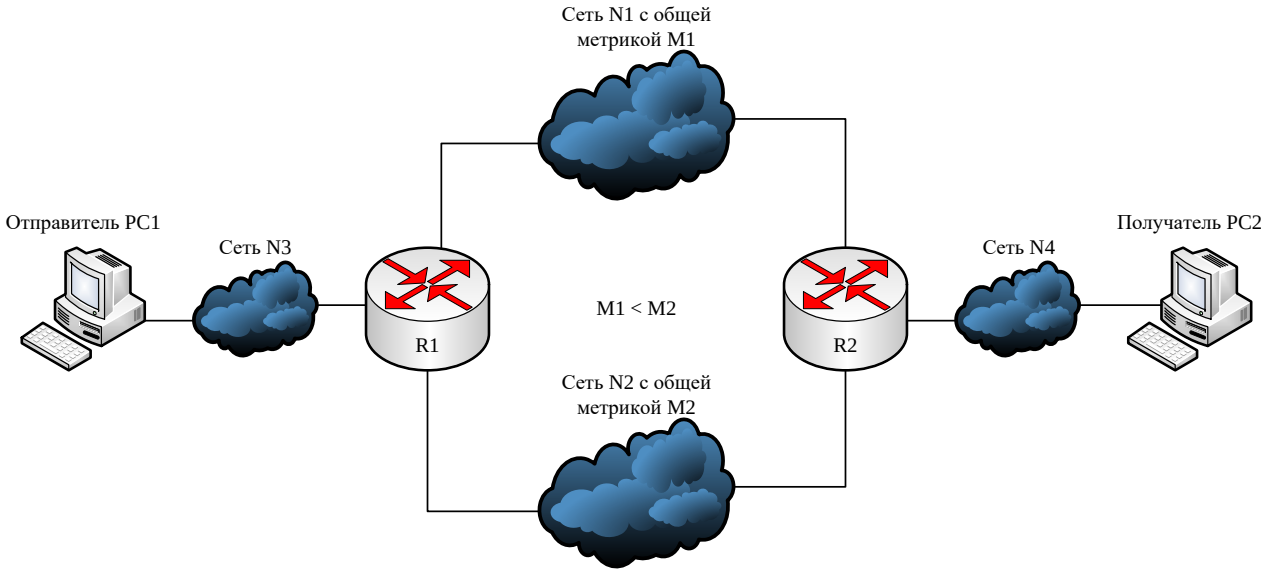

Рисунок 8 – Структурная схема сети для демонстрации работы протоколов EIGRP и OSPF

Здесь информация передается от отправителя PC1 до получателя PC2. При этом инфраструктура сети N3 и N4 не так важна, их может и не быть, в этом случае PC1 напрямую соединен с маршрутизатором R1, а PC2 – с R2. Топология сетей N1 и N2 так же не имеет значения и может быть произвольной.

Основное условие – наличие разветвления на роутере R1, который является маршрутизатором, принимающим решение о том, по какому пути направить трафик. Тогда, если на одном из двух возможных маршрутов на пути трафика через R1 в сторону R2 находится сеть N1 с метрикой M1, а на втором – сеть N2 с метрикой M2, и, например, M1 заведомо лучше (меньше) M2, согласно алгоритмам работы протоколов EIGRP и OSPF пользовательские данные будут идти по пути, проходящему через сеть N1.

При возникновении в сети N1 такого дополнительного трафика, при котором общая нагрузка сильно увеличится, скорость передачи данных упадет, появляется необходимость перенаправить пользовательский трафик через сеть N2. Но в EIGRP и OSPF нет механизмов, обеспечивающих эту возможность.

Для моделирования подойдут любые вариации описанной сети со структурой рисунка 8.

3.2 Моделирование сети с протоколами маршрутизации EIGRP и OSPF при пиковых нагрузках

3.2.1 Описание модели

Для проведения моделирования в OMNeT++ IDE была создана сеть, модель которой представлена на рисунке 9.

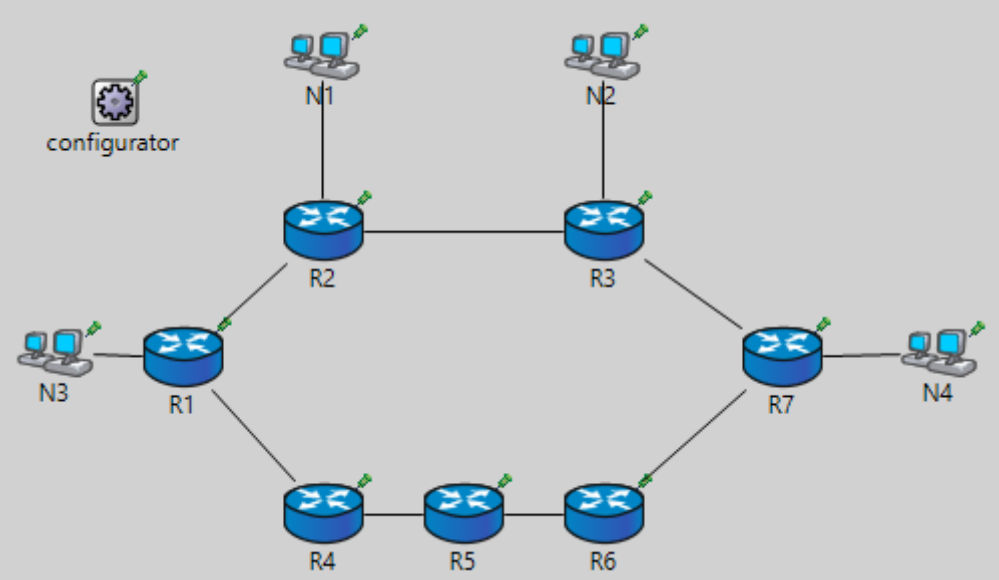

Рисунок 9 – Модель сети, написанная в OMNeT++ IDE

Эта модель описана в файле `EigrpNet.ned` на языке NED, код которого приведен в приложении А.

Конструкция `channel C extends ThruputMeteringChannel {}` в `EigrpNet.ned` описывает тип канала соединения, это позволяет использовать только имя этого типа в секции `connections`, вместо повторения одного и того же определения канала; выбраны следующие параметры соединения:

- задержка соединения равна 0,1 мкс;
- скорость передачи данных канала равна 100 Мбит/с;
- `thruputDisplayFormat = "#N"` позволяет выводить общее количество принятых и отосланных пакетов на интерфейсе в графической оболочке Tkenv во время выполнения моделирования.

Этот тип канала используется для всех соединений (секция `connections`) в приведенной модели, которые, к тому же, являются двухсторонними (символ `<-->`).

Модуль `module EigrpLan {}` определяет локальную сеть с произвольным количеством хостов `h` в ней. В каждой такой сети хаб типа `EtherHub` соединен со всеми хостами `ANSA_Host` и с одним из роутеров общей модели по Ethernet-интерфейсам `ethg`.

Тестовая сеть EigrpNet состоит из семи роутеров R1-R7, каждый из которых имеет тип ANSA_EIGRPRouter и некоторое количество Ethernet-интерфейсов, четырех локальных сетей N1-N4 типа EigrpLan и конфигуратора IPv4NetworkConfigurator, который может автоматически назначать IP-адреса на всех устройствах и сетях. Но из-за ограничений, введенных разработчиками ANSAINET, основные функции IPv4-конфигуратора при настройке сети не используются.

Конфигурация (параметры) создаваемой модели описана в файле omnetpp.ini (приложение Б).

Все EIGRP-маршрутизаторы принадлежат одной области, эта и другие настройки тестовой сети находятся в файле config.xml (приложение В).

На Ethernet-интерфейсах всех устройств сети используется так называемый хук outputHook типа ThruputMeter, позволяющий измерить пропускную способность, что потом будет использовано для визуализации результатов моделирования.

3.2.2 Эксперименты и результаты

Для создания нагрузки на модель сети через файл omnetpp.ini были созданы приложения уровня TCP на некоторых хостах локальных сетей N1-N4, а именно:

- на одном из хостов сети N3 создается приложение, отправляющее 2000000 байт (2 Мбайта) одному из хостов сети N4, на котором инициализируется приложение, отправляющее эти данные обратно. Время начала отправки – 50,10 с.

- аналогичная ситуация и в сетях N1 и N2: хост сети N1 в 50,15 с отправляет хосту сети N2 1000000 байт (1 Мбайт), а тот отправляет их обратно. Такой манипуляцией создается дополнительная нагрузка на маршрутизаторы R2 и R3.

После запуска проекта создается симуляция с настроенной через файл config.xml сетью (рисунок 10). По окончанию моделирования были получен график загрузки интерфейсов eth[0] (соединен с интерфейсом eth[0] роутера R1) и eth[2] (соединен с Ethernet-интерфейсом хоста из сети N1) маршрутизатора R2, который представлен на рисунке 11.

В момент времени, примерно равный 50,15 с, действительно происходит провал в производительности, что обусловлено механизмом работы протокола TCP, который динамически регулирует размер окна (число, определяющее в байтах размер данных, которые отправитель может отправить без получения подтверждения) для уменьшения потерь данных.

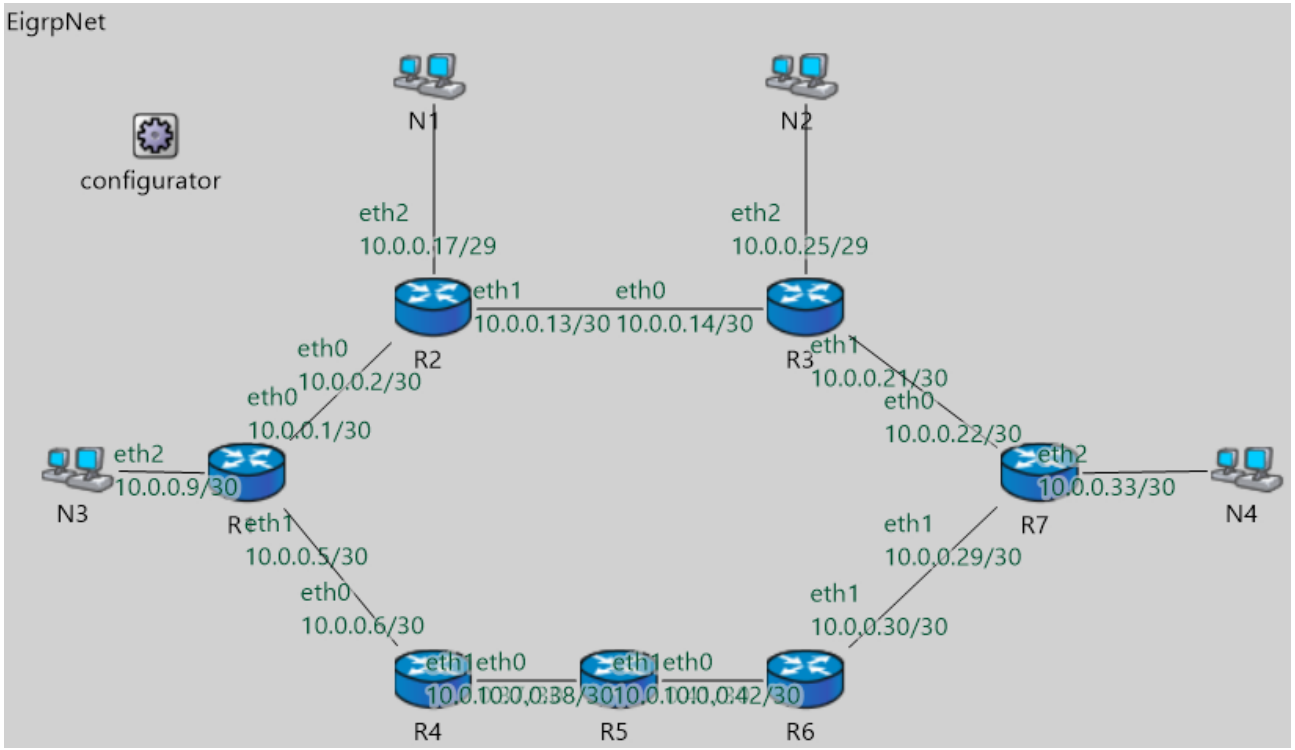

Рисунок 10 – Настройка сети при запуске программы моделирования

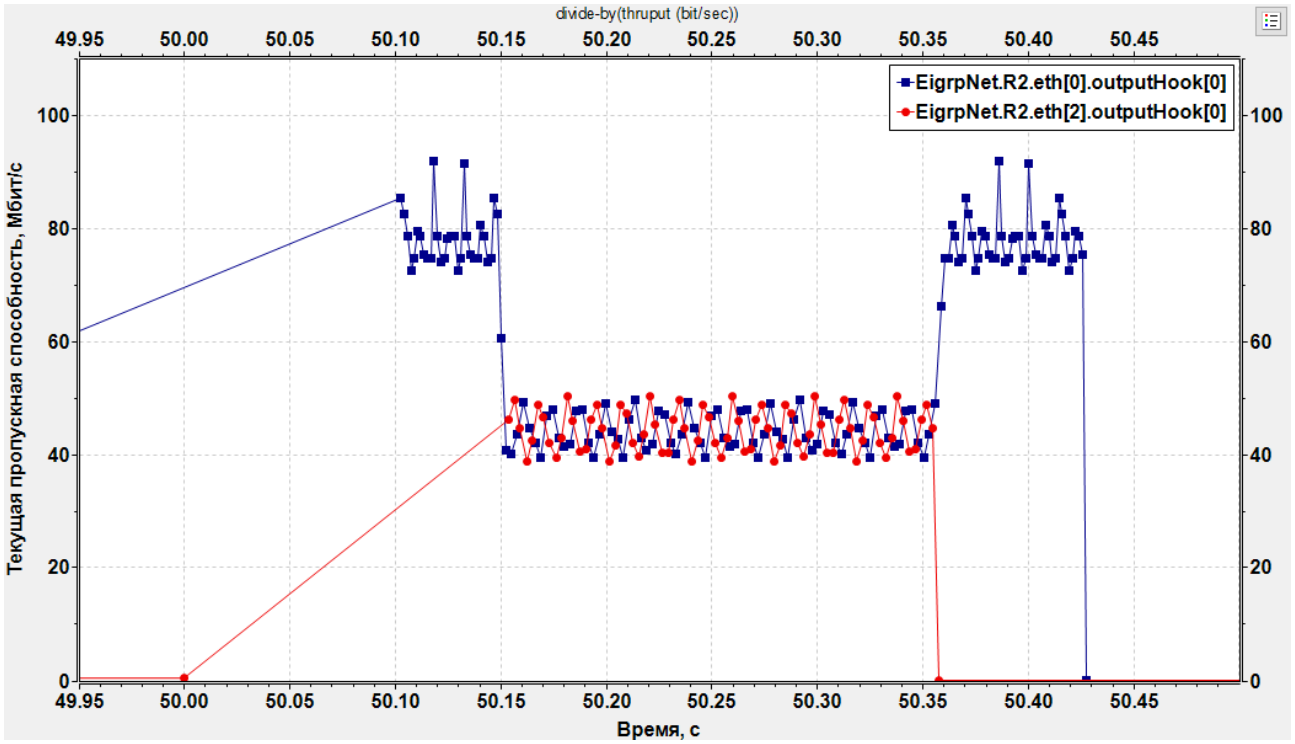

Рисунок 11 – Полученный график изменения пропускной способности каналов маршрутизатора `R2` (интерфейсы `eth[0]` и `eth[2]`) для EIGRP-сети

Для сравнения на рисунке 12 приведен график загрузки интерфейсов `eth[0]` и `eth[2]` роутера `R2` сети со структурой рисунка 10, но работающей на основе протокола OSPF. Изменения в конфигурации модели минимальны и описываться не будут. Как видно, результаты идентичны. Это связано с тем, что ни OSPF, ни EIGRP не учитывают текущую пропускную способность каналов связи для оптимизации нагрузки на сеть (например, путем соответствующего изменения метрики на интерфейсе и последующего перестроения таблицы маршрутизации роутера). Но в EIGRP можно внести изменение в процесс расчета суммарной метрики для искомой сети назначения, поменяв значения коэффициента K_2 с 0 на 1. После этого при расчете стоимости маршрута будет учитываться параметр $LOAD$ – величина нагрузки на линию, вычисляемая динамически как 5-минутное экспоненциально взвешенное среднее, которое обновляется каждые 5 секунд. Однако и в этом случае EIGRP рассматривает загрузку интерфейса только при отправке обновления по какой-то другой причине (например, при отказе канала связи или изменении топологии сети): обновления не анонсируются в сеть каждый раз при изменении нагрузки [3; 4].

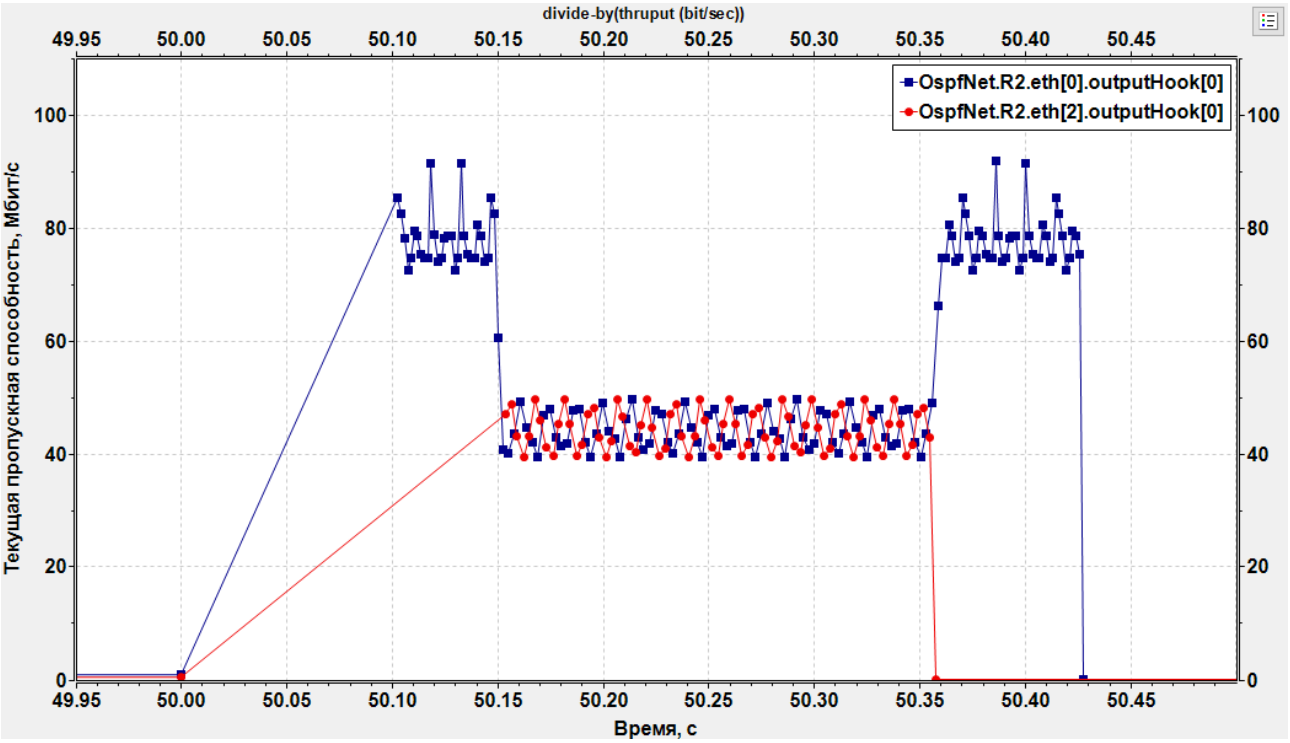

Рисунок 12 – Полученный график изменения пропускной способности каналов маршрутизатора `R2` (интерфейсы `eth[0]` и `eth[2]`) для OSPF-сети

На рисунке 13 показано количество потерянных пакетов на интерфейсах маршрутизатора `R2` для EIGRP – их нет, что соответствует нормальной работе протокола и сети.

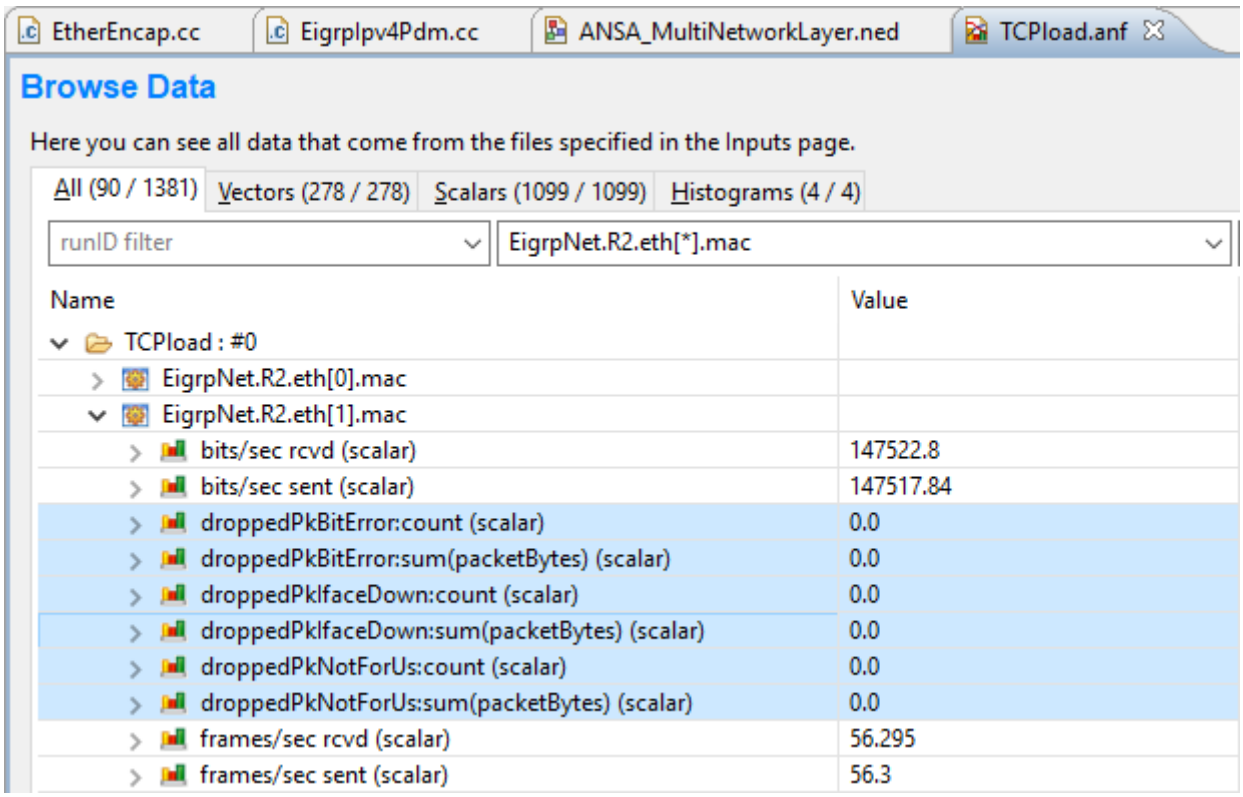

Рисунок 13 – Количество потерянных пакетов на интерфейсах маршрутизатора R2 для EIGRP

3.3 Моделирование сети с оптимизированным протоколом EIGRP при пиковых нагрузках

3.3.1 Описание модели

Моделирование для подтверждения работоспособности разработанного способа оптимизации EIGRP производится на модели сети, описанной ранее.

Значения основных параметров для этой модели:

- `batchSize = 25`;
- `maxInterval = 1`;
- `threshold = 0.70 * bitpersec`.

3.3.2 Эксперименты и результаты

Во время моделирования на интерфейсе `eth[0]` маршрутизатора `R1` (рисунок 14) была зафиксирована перегрузка канала.

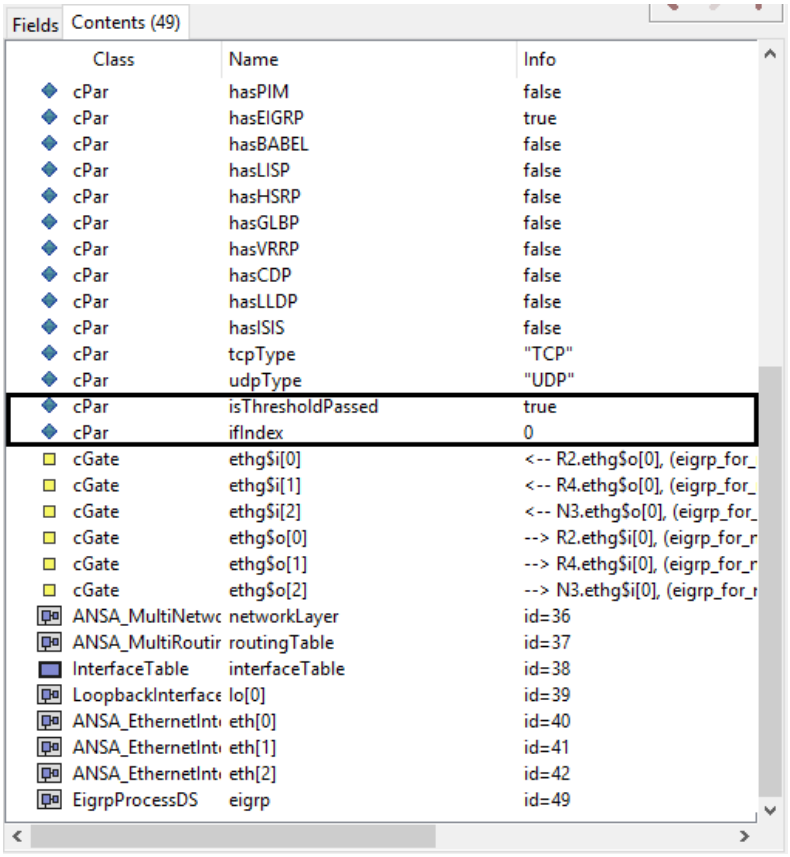

Рисунок 14 – Изменение значений параметров `isThresholdPassed` и `ifIndex` на роутере `R1` при сильном уменьшении пропускной способности

На `R1` в параметр `timestamp` было сохранено время определения падения пропускной способности (рисунок 15), после чего модулем `eigrp` было получено TCN-сообщение (рисунок 16), была изменена загрузка интерфейса `eth[0]`, произошло перестроение таблиц топологии и маршрутизации, соседям были разосланы маршрутные обновления (рисунки 17-19).

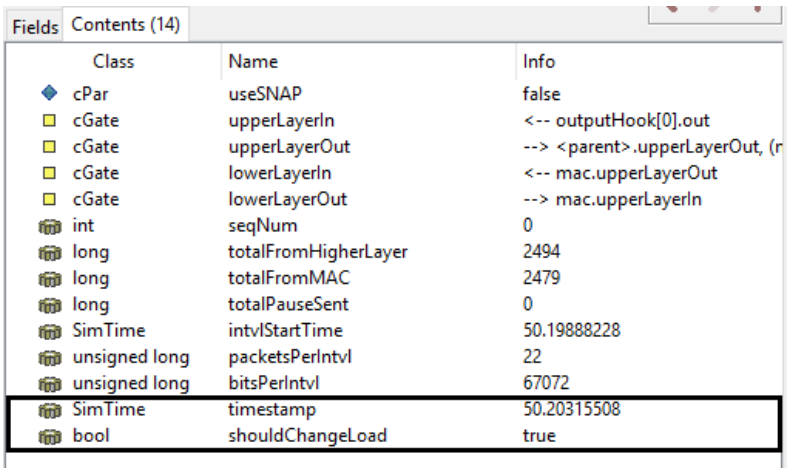

Рисунок 15 – Значения параметров `timestamp` и `shouldChangeLoad` роутера `R1` после определения падения пропускной способности

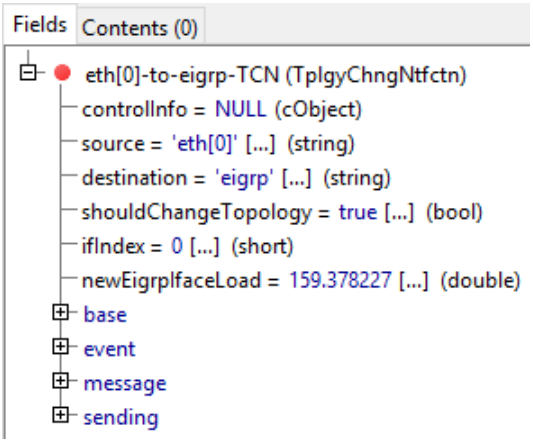

Рисунок 16 – TCN-сообщение, полученное модулем `eigrp` маршрутизатора `R1` после определения падения пропускной способности на интерфейсе `eth[0]`

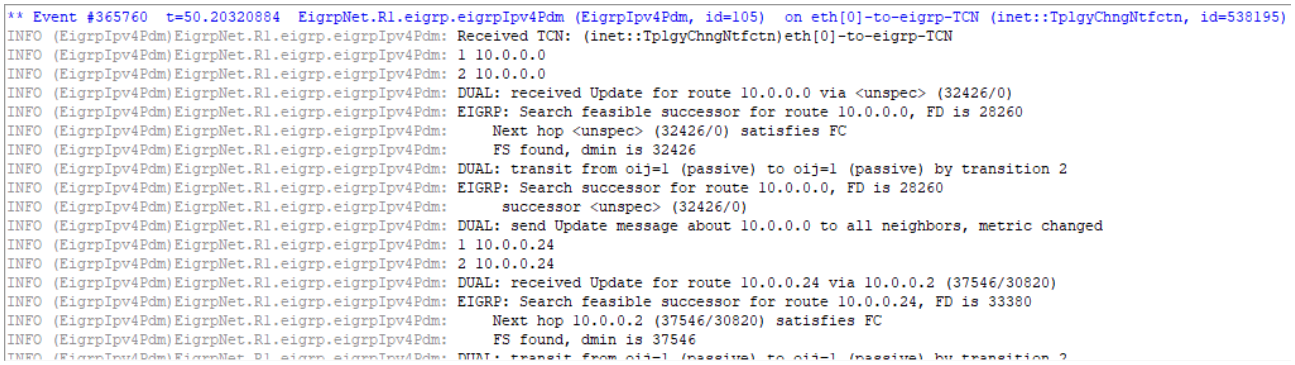

Рисунок 17 – Получение модулем `eigrp` маршрутизатора `R1` TCN-сообщения о падении производительности на интерфейсе `eth[0]`, начало процесса обновления маршрутной информации и рассылки пакетов UPDATE соседям

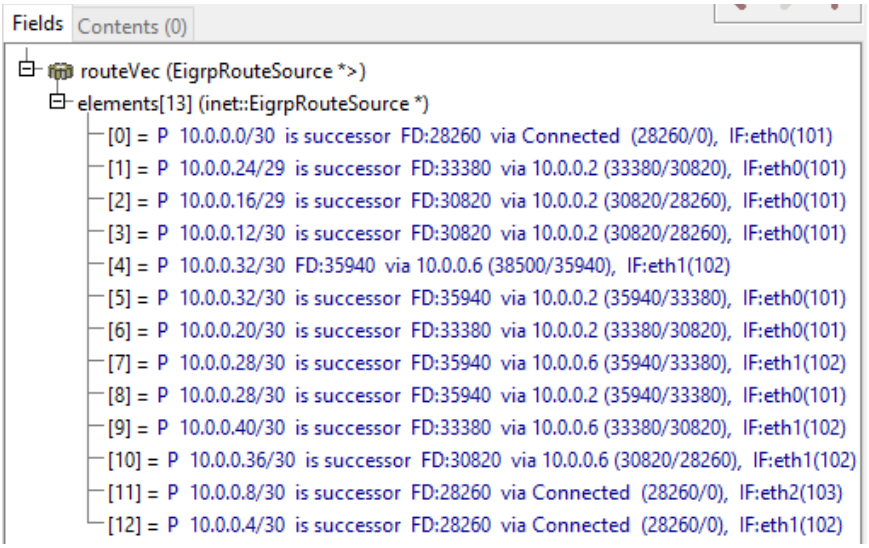

Рисунок 18 – Таблица топологии `R1` со старыми метриками

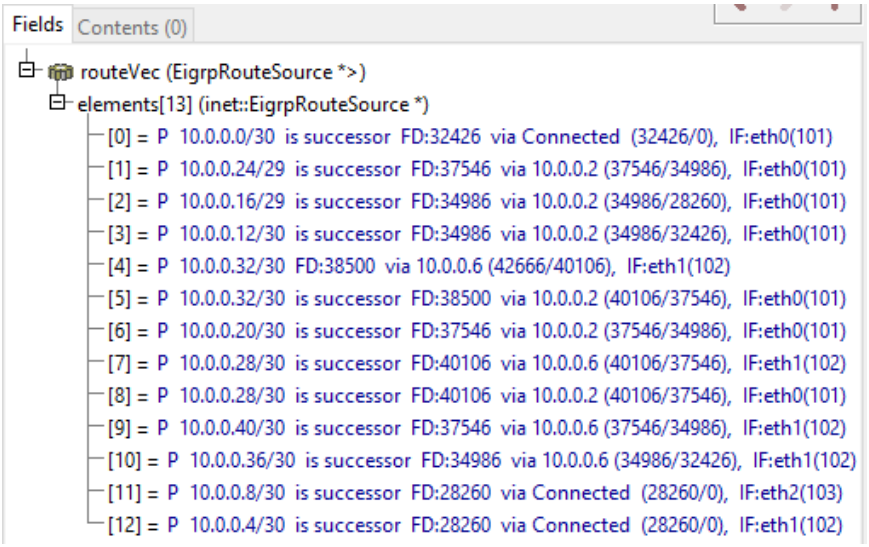

Рисунок 19 – Таблица топологии R1 с новыми метриками после определения перегрузки интерфейса eth[0]

После всего этого данные стали передаваться по новому пути. График изменения пропускной способности на интерфейсах eth[0] и eth[1] маршрутизатора R1 и на интерфейсе eth[2] маршрутизатора R2, полученный после окончания моделирования, представлен на рисунке 20.

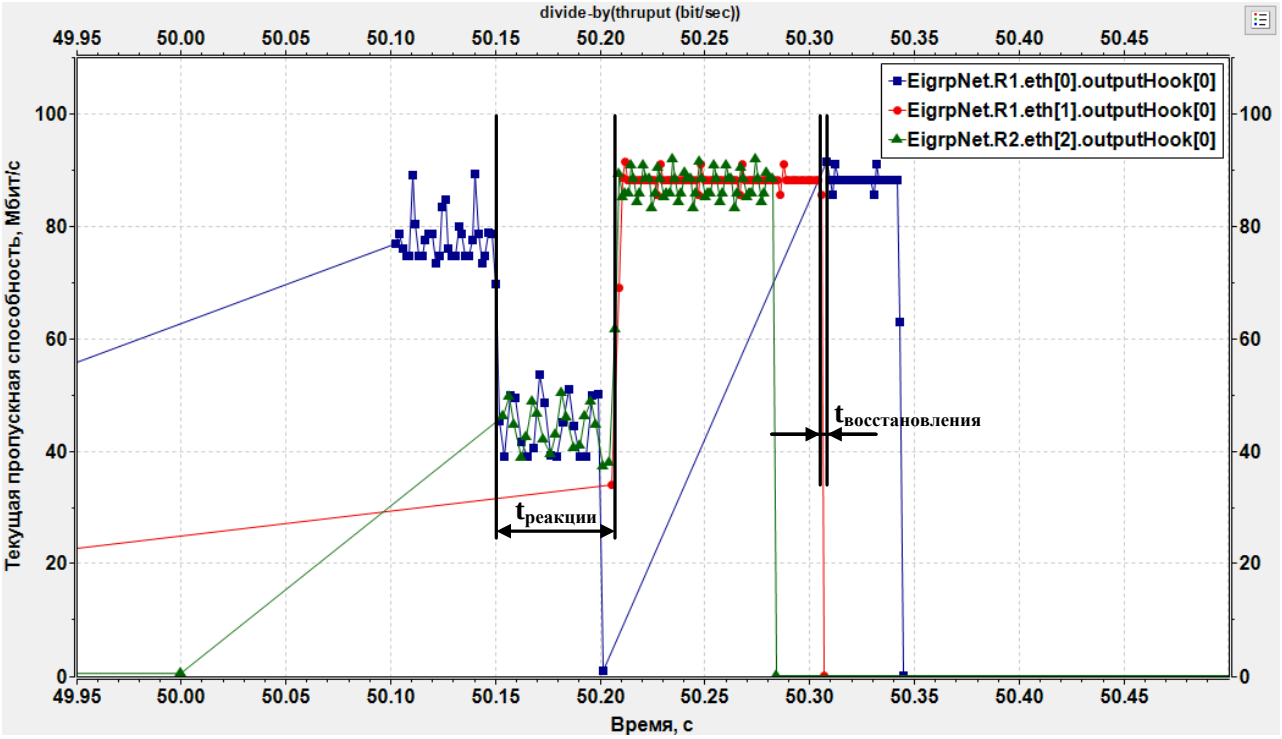

Рисунок 20 – Полученный график изменения пропускной способности интерфейсов маршрутизатора R1 и R2 при моделировании сети с оптимизированным протоколом EIGRP

На этом графике $t_{\text{реакции}}$ – время, потраченное на определение перегрузки интерфейса `eth[0]` маршрутизатора `R1`, а $t_{\text{восстановления}}$ – время, которое потребовалось для восстановления обычной работы EIGRP в тестовой сети.

После 50,15 с пропускная способность на `R1`, `R2` и `R3` сильно упала, на определение чего ушло время, равное $t_{\text{реакции}}$. В этот же промежуток времени произошло обновление маршрутов на `R1` с учетом новой нагрузки на канал `eth[0]`. Далее данные от `N3` к `N4` и обратно стали передаваться по второму более длинному, но уже более выгодному пути, проходящему через интерфейс `eth[1]` роутера `R1`, через `R4`, `R5` и `R6`. При этом пользовательские данные от `N1` к `N2` отправлялись и обрабатывались все это время, то есть сеть продолжала функционировать вне зависимости от возникающих перегрузок и обновлений маршрутной информации в ней. Затем примерно в 50,306 с через $t_{\text{восстановления}}$, когда пользовательский трафик от `N1` к `N2` передался полностью (через роутеры `R2` и `R3`), произошло восстановление сети, и оставшаяся информация от `N3` к `N4` продолжила передаваться по обычному верхнему пути через интерфейс `eth[0]` узла `R1`, через `R2` и `R3`.

Количество потерянных пакетов для интерфейса `eth[0]` роутера `R1` представлен на рисунке 21, а для интерфейса `eth[1]` этого же маршрутизатора – на рисунке 22. Потерь нет, что может говорить о некоторой надежности разработанной модификации для протокола маршрутизации EIGRP.

Таким образом, при проведении моделирования была показана работоспособность разработанной модификации для оптимизации протокола маршрутизации EIGRP.

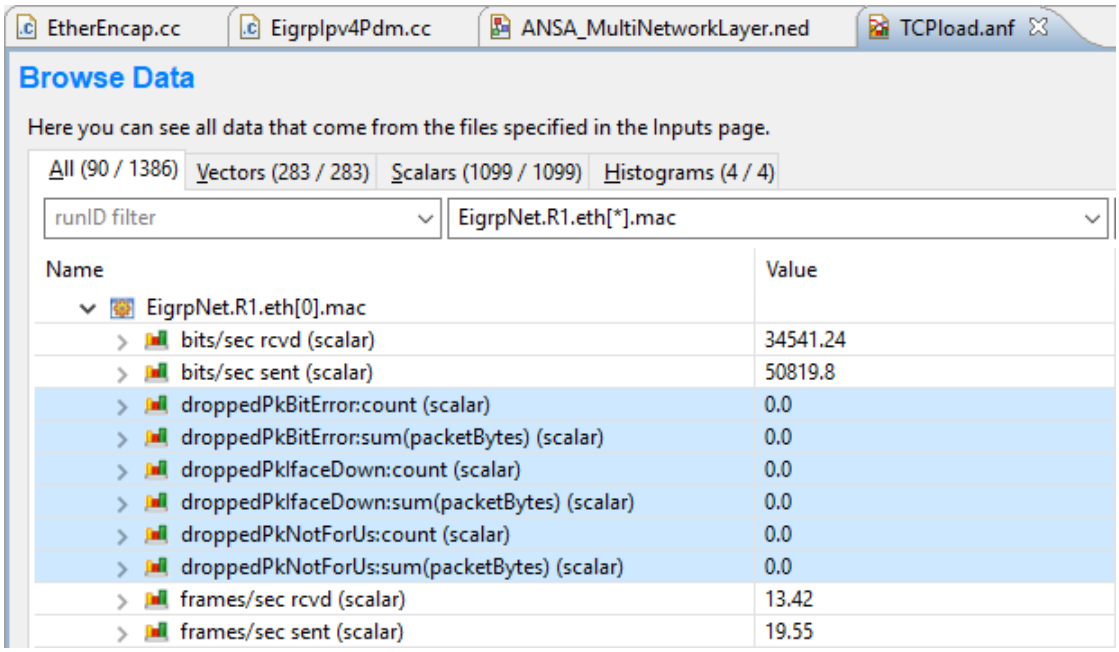

Рисунок 21 – Количество потерянных пакетов на интерфейсе eth[0] маршрутизатора `R1` для модифицированного EIGRP

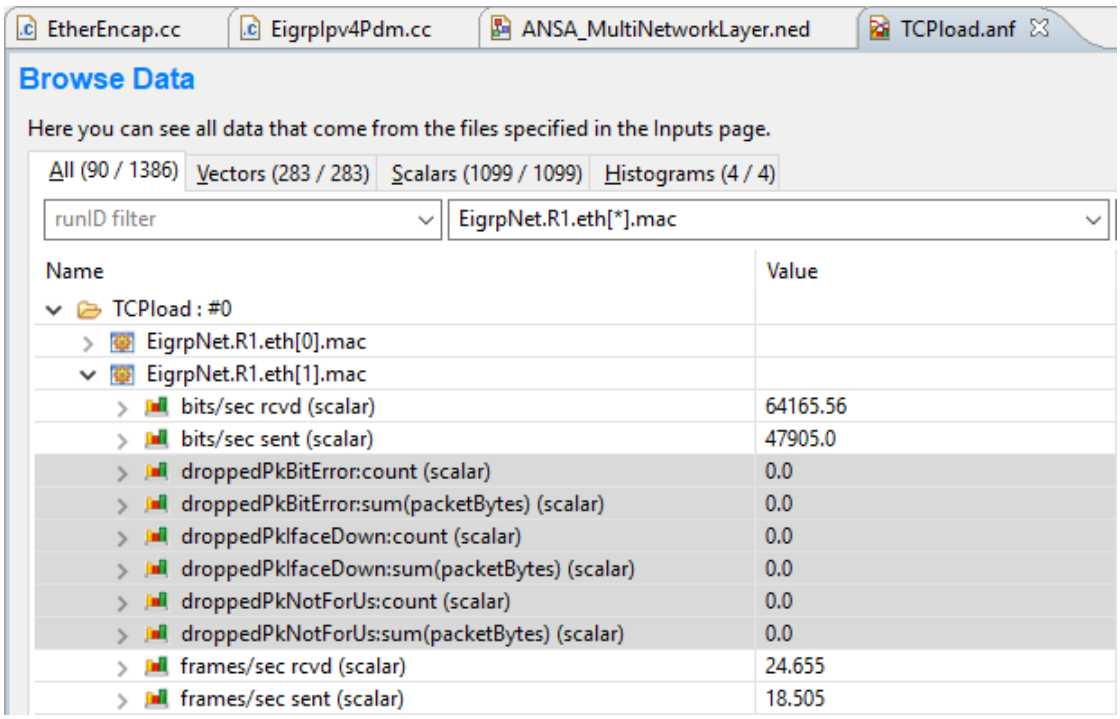

Рисунок 22 – Количество потерянных пакетов на интерфейсе `eth[1]` маршрутизатора `R1` для модифицированного EIGRP

ЗАКЛЮЧЕНИЕ

В результате выполнения работы был разработан метод повышения производительности сети при больших нагрузках, где протокол EIGRP используется в качестве основного протокола маршрутизации. Метод был реализован в библиотеке ANSAINET для OMNeT++. Также были изучены основы маршрутизации, основные протоколы маршрутизации, в том числе EIGRP и OSPF. Были отмечены их преимущества и недостатки. Один из главных недостатков – невозможность протоколов учитывать текущую пропускную способность канала для оптимизации нагрузки на сеть – был показан с помощью моделирования в OMNeT++.

Были проведены эксперименты с модифицированным EIGRP на модели сети, написанной в OMNeT++, и была показана работоспособность доработанной версии протокола.

Впоследствии предложенные наработки могут быть использованы при составлении и реализации оптимального алгоритма необходимости переключения на новые маршруты сетей произвольной топологии для использования не только в средах моделирования, но и в реальных сетях передачи данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В. Г. Олифер, Н. А. Олифер. Изд. 4-е. – Санкт-Петербург : Питер, 2010. – 944 с.

2 Enhanced Interior Gateway Routing Protocol [Электронный ресурс] // Cisco Systems, Inc. – Режим доступа: https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html (дата обращения: 28.05.2018).

3 Pepelnjak, I. EIGRP load and reliability metrics [Электронный ресурс] / I. Pepelnjak // ipSpace.net: Internetworking perspectives by Ivan Pepelnjak. – Режим доступа: http://blog.ipspace.net/2009/06/eigrp-load-and-reliability-metrics.html (дата обращения: 28.05.2018);

4 RFC 7868, Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) [Электронный ресурс] // The Internet Engineering Task Force (IETF). – Режим доступа: https://datatracker.ietf.org/doc/rfc7868/ (дата обращения: 28.05.2018).

5 RFC 2328, OSPF Version 2 [Электронный ресурс] // The Internet Engineering Task Force (IETF). – Режим доступа: https://datatracker.ietf.org/doc/rfc2328/ (дата обращения: 28.05.2018).

6 Перевод RFC 2328 [Электронный ресурс] // Энциклопедия сетевых протоколов. – Режим доступа: http://www.protocols.ru/files/RFC/RFC-2328.pdf (дата обращения: 28.05.2018).

7 Cisco tips: Введение в EIGRP [Электронный ресурс] // ООО «Новаком». – Режим доступа: http://novacom.ru/tech_support/Cisco/eigrp.html (дата обращения: 28.05.2018).

8 Introduction to EIGRP [Электронный ресурс] // Cisco Systems, Inc. – Режим доступа: https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13669-1.html (дата обращения: 28.05.2018).

9 An Introduction to IGRP [Электронный ресурс] // Cisco Systems, Inc. – Режим доступа: https://www.cisco.com/c/en/us/support/docs/ip/interior-gateway-routing-protocol-igrp/26825-5.html (дата обращения: 28.05.2018).

10 Самойленко, Н. EIGRP [Электронный ресурс] / Н. Самойленко // Xgu.ru. – Режим доступа: http://xgu.ru/w/index.php?title=EIGRP&oldid=38608 (дата обращения: 28.05.2018).

11 Enhanced Interior Gateway Routing Protocol (EIGRP) Informational RFC Frequently Asked Questions [Электронный ресурс] // Cisco Systems, Inc. – Режим доступа: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enhanced-interior-gateway-routing-protocol-eigrp/qa_C67-726299.html (дата обращения: 06.06.2018).

12 Burke, A. Why Is Cisco Bothering with «Open» EIGRP? [Электронный ресурс] / A. Burke // Packet Pushers Interactive, LLC. – Режим доступа: http://packetpushers.net/why-is-cisco-bothering-with-open-eigrp/ (дата обращения: 06.06.2018).

13 Макаренко, С. И. Модифицированный алгоритм Беллмана-Форда с формированием кратчайших и резервных путей и его применение для повышения устойчивости телекоммуникационных систем [Электронный ресурс] / С. И. Макаренко, М. Н. Квасов // Научная электронная библиотека eLIBRARY.RU. Режим доступа: https://elibrary.ru/item.asp?id=28911387 (дата обращения: 06.06.2018).

14 Snihurov, A. Improvement of EIGRP Protocol Routing Algorithm with the Consideration of Information Security Risk Parameters [Электронный ресурс] / A. Snihurov, V. Chakrian // Электронный архив открытого доступа Харьковского национального университета радиоэлектроники. – Режим доступа: http://openarchive.nure.ua/bitstream/document/2243/1/SJET38707-714.pdf (дата обращения: 06.06.2018).

15 Improvement of Performance of EIGRP Network by Using a Supervisory Controller with Smart Congestion Avoidance Algorithm [Электронный ресурс] / H. Hasan [и др.] // ResearchGate GmbH. – Режим доступа: https://www.researchgate.net/publication/306925828_Improvement_of_performance_of_EIGRP_network_by_using_a_supervisory_controller_with_smart_congestion_avoidance_algorithm (дата обращения: 07.06.2018).

16 Кузнецов, Н. А. Алгоритм Дейкстры с улучшенной робастностью для управления маршрутизацией в IP-сетях [Электронный ресурс] / Н. А. Кузнецов, В. Н. Фетисов // Общероссийский математический портал Math-Net.Ru. – Режим доступа: http://www.mathnet.ru/at607 (дата обращения: 19.04.2018).

17 Цветков, К. Ю. Формирование резервных путей на основе алгоритма Дейкстры в целях повышения устойчивости информационно-телекоммуникационных сетей [Электронный ресурс] / К. Ю. Цветков, С. И. Макаренко, Р. Л. Михайлов // Научная электронная библиотека eLIBRARY.RU. – Режим доступа: http://elibrary.ru/item.asp?id=21427025 (дата обращения: 19.04.2018).

18 Полторак, В. П. Алгоритмы балансировки в сети OSPF [Электронный ресурс] / В. П. Полторак, А. Ю. Степаненко // Молодой ученый. – 2014. – № 4. – С. 232-242. – Режим доступа: https://moluch.ru/archive/63/pdf/505 (дата обращения: 19.04.2018).

19 Перепелкин, Д. А. Повышение эффективности функционирования корпоративных сетей на базе протокола OSPF [Электронный ресурс] / Д. А. Перепелкин, А. И. Перепелкин // Научная электронная библиотека eLIBRARY.RU. – Режим доступа: http://elibrary.ru/item.asp?id=25834152 (дата обращения: 19.04.2018).

20 Daniluk, K. Energy-Efficient Protocol in OMNeT++ Simulation Environment [Электронный ресурс] / K. Daniluk // ITHEA International Scientific Journals. – Режим доступа: http://foibg.com/ibs_isc/ibs-27/ibs-27-p24.pdf (дата обращения: 19.04.2018).

21 Олифер Н. А. Средства анализа и оптимизации локальных сетей [Электронный ресурс] / Н. А. Олифер, В. Г. Олифер // Центр Информационных Технологий. – 1998. – Режим доступа: http://citforum.ru/nets/optimize/index.shtml (дата обращения: 23.05.2018).

22 OMNeT++ [Электронный ресурс] // OMNeT++ Discrete Event Simulator. – Режим доступа: https://omnetpp.org (дата обращения: 28.05.2018).

23 INET Framework [Электронный ресурс] // INET Framework. – Режим доступа: https://inet.omnetpp.org/ (дата обращения: 28.05.2018).

24 ANSAINET [Электронный ресурс] // ANSA by Brno University of Technology. – Режим доступа: https://ansa.omnetpp.org/ (дата обращения: 28.05.2018).

25 СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2–07–2012 ; введ. 09.01.2014. – Красноярск : ИПК СФУ, 2014. – 60 с.

ПРИЛОЖЕНИЕ А

Содержимое файла `EigrpNet.ned`, описывающего топологию модели тестовой EIGRP-сети

```
package eigrp_for_masters_thesis;

import inet.common.misc.ThruputMeteringChannel;
import inet.linklayer.ethernet.EtherHub;
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.common.scenario.ScenarioManager;
import inet.common.lifecycle.LifecycleController;
import ansa.node.ANSA_EIGRPRouter;
import ansa.node.ANSA_Host;

channel C extends ThruputMeteringChannel
{
    delay = 0.1us;
    datarate = 100Mbps;
    thruputDisplayFormat = "#N";
}

module EigrpLan
{
    parameters:
        int h; // number of hosts on the hub
        @display("i=device/lan");
    gates:
        inout ethg[];
    submodules:
        hub: EtherHub {
            parameters:
                @display("is=s");
        }
        host[h]: ANSA_Host {
            parameters:
                @display("is=s");
        }
    connections:
        for i=0..sizeof(ethg)-1 {
            hub.ethg++ <--> ethg[i];
        }
        for i=0..h-1 {
            hub.ethg++ <--> C <--> host[i].ethg++;
        }
}
```

```
network EigrpNet
{
    @display("bgb=385,240;bgl=2");
    submodules:
        R1: ANSA_EIGRPRouter {
            parameters:
                @display("p=71,137");
            gates:
                ethg[3];
        }
        R2: ANSA_EIGRPRouter {
            parameters:
                @display("p=125,89");
            gates:
                ethg[3];
        }
        R3: ANSA_EIGRPRouter {
            parameters:
                @display("p=233,89");
            gates:
                ethg[3];
        }
        R7: ANSA_EIGRPRouter {
            parameters:
                @display("p=301,137");
            gates:
                ethg[3];
        }
        R4: ANSA_EIGRPRouter {
            parameters:
                @display("p=125,197");
            gates:
                ethg[2];
        }
        R5: ANSA_EIGRPRouter {
            parameters:
                @display("p=179,197");
            gates:
                ethg[2];
        }
        R6: ANSA_EIGRPRouter {
            parameters:
                @display("p=233,197");
            gates:
                ethg[2];
        }
```

```
        N1: EigrpLan {
            parameters:
                h = 2;
                @display("p=124,22");
        }
        N2: EigrpLan {
            parameters:
                h = 2;
                @display("p=232,22");
        }
        N3: EigrpLan {
            parameters:
                h = 1;
                @display("p=22,136");
        }
        N4: EigrpLan {
            parameters:
                h = 1;
                @display("p=361,136");
        }
        configurator: IPv4NetworkConfigurator {
            parameters:
                config = xml("<config></config>");
                assignAddresses = false;
                assignDisjunctSubnetAddresses = false;
                addStaticRoutes = false;
                addDefaultRoutes = false;
                addSubnetRoutes = false;
                optimizeRoutes = false;
                @display("p=44.615387,39.23077");
        }
    connections:
        R1.ethg[0] <--> C <--> R2.ethg[0];
        R2.ethg[1] <--> C <--> R3.ethg[0];
        R3.ethg[1] <--> C <--> R7.ethg[0];

        R1.ethg[1] <--> C <--> R4.ethg[0];
        R4.ethg[1] <--> C <--> R5.ethg[0];
        R5.ethg[1] <--> C <--> R6.ethg[0];
        R6.ethg[1] <--> C <--> R7.ethg[1];

        N1.ethg++ <--> C <--> R2.ethg[2];
        N2.ethg++ <--> C <--> R3.ethg[2];

        N3.ethg++ <--> C <--> R1.ethg[2];
        N4.ethg++ <--> C <--> R7.ethg[2];
}
```

ПРИЛОЖЕНИЕ Б

Содержимое файла `omnetpp.ini`, описывающего конфигурацию (параметры) выполнения модели тестовой EIGRP-сети

```
[General]
network = EigrpNet
#record-eventlog = true
#debug-on-errors = true
#tkenv-plugin-path = ../../../etc/plugins
sim-time-limit = 200s#200s#600s

**.enableIPv6 = false
**.enableCLNS = false

**.*.networkLayer.enableANSAConfig = true
**.R1.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.9']/")
**.R2.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.17']/")
**.R3.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.25']/")
**.R4.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.37']/")
**.R5.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.41']/")
**.R6.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.42']/")
**.R7.configData = xmldoc("config.xml",
                       "Devices/Router[@id='10.0.0.33']/")
**.N1.host[0].configData = xmldoc("config.xml",
                       "Devices/Host[@id='10.0.0.18']/")
**.N1.host[1].configData = xmldoc("config.xml",
                       "Devices/Host[@id='10.0.0.19']/")
**.N2.host[0].configData = xmldoc("config.xml",=
                       "Devices/Host[@id='10.0.0.26']/")
**.N2.host[1].configData = xmldoc("config.xml",
                       "Devices/Host[@id='10.0.0.27']/")
**.N3.host[0].configData = xmldoc("config.xml",
                       "Devices/Host[@id='10.0.0.10']/")
**.N4.host[0].configData = xmldoc("config.xml",
                       "Devices/Host[@id='10.0.0.34']/")

# hookType settings
**.eth[*].numOutputHooks = 1
**.eth[*].outputHook[0].typename = "ThruputMeter" # Nop |
     ThruputMeter | OrdinalBasedDropper | OrdinalBasedDuplicator
```

```
[Config TCPload]
description = "Use of TCP apps to decrease network throughput"
# tcp apps
#
# from N3.host[0] (R1) to N4.host[0] (R7)
#
**.N3.host[0].numTcpApps = 1
**.N3.host[0].tcpApp[0].typename = "TCPSessionApp"
**.N3.host[0].tcpApp[0].sendBytes = 2000000 bytes
**.N3.host[0].tcpApp[0].active = true
**.N3.host[0].tcpApp[0].connectAddress = "N4.host[0]"
**.N3.host[0].tcpApp[0].connectPort = 10021
**.N3.host[0].tcpApp[0].tOpen = 50s
**.N3.host[0].tcpApp[0].tSend = 50.10s
**.N3.host[0].tcpApp[0].tClose = 50s

**.N4.host[0].numTcpApps = 1
**.N4.host[0].tcpApp[0].typename = "TCPEchoApp"
**.N4.host[0].tcpApp[0].localPort = 10021

#
# from N1.host[0] (R2) to N2.host[0] (R3)
#
**.N1.host[0].numTcpApps = 1
**.N1.host[0].tcpApp[0].typename = "TCPSessionApp"
**.N1.host[0].tcpApp[0].sendBytes = 1000000 bytes
**.N1.host[0].tcpApp[0].active = true
**.N1.host[0].tcpApp[0].connectAddress = "N2.host[0]"
**.N1.host[0].tcpApp[0].connectPort = 10021
**.N1.host[0].tcpApp[0].tOpen = 50s
**.N1.host[0].tcpApp[0].tSend = 50.15s
**.N1.host[0].tcpApp[0].tClose = 50s

**.N2.host[0].numTcpApps = 1
**.N2.host[0].tcpApp[0].typename = "TCPEchoApp"
**.N2.host[0].tcpApp[0].localPort = 10021
```

ПРИЛОЖЕНИЕ В

Содержимое файла config.xml, описывающего параметры устройств тестовой EIGRP-сети

```
<Devices>
     <!-- R1 -->
     <Router id="10.0.0.9">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>10.0.0.1</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth1">
                <IPAddress>10.0.0.5</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth2">
                <IPAddress>10.0.0.9</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
        </Interfaces>

        <Routing>
           <EIGRP>
                <ProcessIPv4 asNumber="1">
                     <Networks>
                          <Network>
                               <IPAddress>10.0.0.1</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                               <IPAddress>10.0.0.5</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                               <IPAddress>10.0.0.9</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                     </Networks>
                </ProcessIPv4>
           </EIGRP>
        </Routing>
```

```
    <Routing6>
    </Routing6>

</Router>

<!-- R2 -->
<Router id="10.0.0.17">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>10.0.0.2</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
        <Interface name="eth1">
            <IPAddress>10.0.0.13</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
        <Interface name="eth2">
            <IPAddress>10.0.0.17</IPAddress>
            <Mask>255.255.255.248</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
    </Interfaces>

    <Routing>
       <EIGRP>
            <ProcessIPv4 asNumber="1">
                 <Networks>
                      <Network>
                           <IPAddress>10.0.0.2</IPAddress>
                           <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                      <Network>
                           <IPAddress>10.0.0.13</IPAddress>
                           <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                      <Network>
                           <IPAddress>10.0.0.17</IPAddress>
                           <Wildcard>0.0.0.7</Wildcard>
                      </Network>
                 </Networks>
            </ProcessIPv4>
       </EIGRP>
    </Routing>
```

```
    <Routing6>
    </Routing6>

</Router>

<!-- R3 -->
<Router id="10.0.0.25">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>10.0.0.14</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
        <Interface name="eth1">
            <IPAddress>10.0.0.21</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
        <Interface name="eth2">
            <IPAddress>10.0.0.25</IPAddress>
            <Mask>255.255.255.248</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
    </Interfaces>

    <Routing>
       <EIGRP>
            <ProcessIPv4 asNumber="1">
                 <Networks>
                      <Network>
                           <IPAddress>10.0.0.14</IPAddress>
                            <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                      <Network>
                           <IPAddress>10.0.0.21</IPAddress>
                           <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                      <Network>
                           <IPAddress>10.0.0.25</IPAddress>
                           <Wildcard>0.0.0.7</Wildcard>
                      </Network>
                 </Networks>
            </ProcessIPv4>
       </EIGRP>
    </Routing>
```

```
    <Routing6>
    </Routing6>

</Router>

<!-- R4 -->
<Router id="10.0.0.37">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>10.0.0.6</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
        <Interface name="eth1">
            <IPAddress>10.0.0.37</IPAddress>
            <Mask>255.255.255.252</Mask>
            <EIGRP-IPv4 asNumber='1'>
            </EIGRP-IPv4>
        </Interface>
    </Interfaces>

    <Routing>
       <EIGRP>
            <ProcessIPv4 asNumber="1">
                 <Networks>
                      <Network>
                           <IPAddress>10.0.0.6</IPAddress>
                           <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                      <Network>
                           <IPAddress>10.0.0.37</IPAddress>
                           <Wildcard>0.0.0.3</Wildcard>
                      </Network>
                 </Networks>
            </ProcessIPv4>
       </EIGRP>
    </Routing>

    <Routing6>
    </Routing6>

</Router>

<!-- R5 -->
<Router id="10.0.0.41">
    <Interfaces>
        <Interface name="eth0">
```

```
                <IPAddress>10.0.0.38</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth1">
                <IPAddress>10.0.0.41</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
        </Interfaces>

        <Routing>
           <EIGRP>
                <ProcessIPv4 asNumber="1">
                     <Networks>
                          <Network>
                               <IPAddress>10.0.0.38</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                               <IPAddress>10.0.0.41</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                     </Networks>
                </ProcessIPv4>
           </EIGRP>
        </Routing>

        <Routing6>
        </Routing6>

    </Router>

    <!-- R6 -->
    <Router id="10.0.0.42">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>10.0.0.42</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth1">
                <IPAddress>10.0.0.30</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
```

```
            </Interface>
        </Interfaces>

        <Routing>
           <EIGRP>
                <ProcessIPv4 asNumber="1">
                     <Networks>
                          <Network>
                               <IPAddress>10.0.0.30</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                               <IPAddress>10.0.0.42</IPAddress>
                               <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                     </Networks>
                </ProcessIPv4>
           </EIGRP>
        </Routing>

        <Routing6>
        </Routing6>

    </Router>

    <!-- R7 -->
     <Router id="10.0.0.33">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>10.0.0.22</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth1">
                <IPAddress>10.0.0.29</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
            <Interface name="eth2">
                <IPAddress>10.0.0.33</IPAddress>
                <Mask>255.255.255.252</Mask>
                <EIGRP-IPv4 asNumber='1'>
                </EIGRP-IPv4>
            </Interface>
        </Interfaces>

        <Routing>
```

```
        <EIGRP>
              <ProcessIPv4 asNumber="1">
                    <Networks>
                          <Network>
                                <IPAddress>10.0.0.22</IPAddress>
                                <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                                <IPAddress>10.0.0.29</IPAddress>
                                <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                          <Network>
                                <IPAddress>10.0.0.33</IPAddress>
                                <Wildcard>0.0.0.3</Wildcard>
                          </Network>
                    </Networks>
              </ProcessIPv4>
        </EIGRP>
     </Routing>

     <Routing6>
     </Routing6>

</Router>

        <Host id="10.0.0.10">
        <Interfaces>
              <Interface name="eth0">
                    <IPAddress>10.0.0.10</IPAddress>
                    <Mask>255.255.255.252</Mask>
              </Interface>
        </Interfaces>
        <DefaultRouter>10.0.0.9</DefaultRouter>
 </Host>
        <Host id="10.0.0.18">
        <Interfaces>
              <Interface name="eth0">
                    <IPAddress>10.0.0.18</IPAddress>
                    <Mask>255.255.255.248</Mask>
              </Interface>
        </Interfaces>
        <DefaultRouter>10.0.0.17</DefaultRouter>
 </Host>
        <Host id="10.0.0.19">
        <Interfaces>
              <Interface name="eth0">
                    <IPAddress>10.0.0.19</IPAddress>
                    <Mask>255.255.255.248</Mask>
              </Interface>
```

```
            </Interfaces>
            <DefaultRouter>10.0.0.17</DefaultRouter>
      </Host>
            <Host id="10.0.0.26">
            <Interfaces>
                  <Interface name="eth0">
                        <IPAddress>10.0.0.26</IPAddress>
                        <Mask>255.255.255.248</Mask>
                  </Interface>
            </Interfaces>
            <DefaultRouter>10.0.0.25</DefaultRouter>
      </Host>
            <Host id="10.0.0.27">
            <Interfaces>
                  <Interface name="eth0">
                        <IPAddress>10.0.0.27</IPAddress>
                        <Mask>255.255.255.248</Mask>
                  </Interface>
            </Interfaces>
            <DefaultRouter>10.0.0.25</DefaultRouter>
      </Host>
            <Host id="10.0.0.34">
            <Interfaces>
                  <Interface name="eth0">
                        <IPAddress>10.0.0.34</IPAddress>
                        <Mask>255.255.255.252</Mask>
                  </Interface>
            </Interfaces>
            <DefaultRouter>10.0.0.33</DefaultRouter>
      </Host>
</Devices>
```