МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. Ломоносова

На правах рукописи

ВДОВИН Павел Максимович

ЖАДНЫЕ АЛГОРИТМЫ И СТРАТЕГИИ ОГРАНИЧЕННОГО ПЕРЕБОРА ДЛЯ ПЛАНИРОВАНИЯ ВЫЧИСЛЕНИЙ В СИСТЕМАХ С ЖЕСТКИМИ ТРЕБОВАНИЯМИ К КАЧЕСТВУ ОБСЛУЖИВАНИЯ

Специальность 05.13.11 –
Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат технических наук,
В.А. Костенко

Москва – 2016

Содержание

Введение 5
1. Описание задач планирования вычислений в распределенных системах реального времени 12
1.1. Задача планирования вычислений в распределенных системах 12
1.2. Задача планирования вычислений в ЦОД с моделью обслуживания IaaS 13
1.2.1. Постановка задачи 14
1.3. Задача построения сетей AFDX 17
1.3.1. Описание сетей AFDX 17
1.3.2. Постановка задачи 20
1.4. Выводы 23
2. Обзор различных подходов к построению алгоритмов решения близких задач 25
2.1. Подходы к решению задачи отображения запросов в ЦОД 26
2.2. Упаковка в контейнеры 29
2.3. Построение маршрутов виртуальных каналов 32
2.4. Подходы к проектированию сетей AFDX 34
2.5. Вычисление оценки длительности передачи сообщений в сетях AFDX 36
2.6. Выводы 37
3. Алгоритм планирования вычислений в ЦОД с моделью обслуживания IaaS 39
3.1. Общая схема алгоритма 39
3.2. Назначение виртуальных машин и storage-элементов 39
3.2.1. Процедура ограниченного перебора 40
3.2.2. Жадные критерии 42
3.3. Построение маршрутов виртуальных каналов 44
3.3.1. Общая схема процедуры построения маршрутов виртуальных каналов 44
3.3.2. Жадные критерии 45
3.3.3. Построение маршрута для одного виртуального канала 45
3.3.4. Процедура ограниченного перебора 46
3.3.5. Процедура репликации 47
3.4. Свойства и теоретическое обоснование алгоритмов 47
3.4.1. Оценка сложности этапов алгоритма 47
3.4.2. Зависимость сложности алгоритма от глубины ограниченного перебора 49
3.4.3. Корректность алгоритма 51

3.4.4. Свойства процедуры ограниченного перебора при решении задачи упаковки в контейнеры 52
3.4.5. Свойства процедуры ограниченного перебора при построении маршрутов виртуальных каналов 54
3.4.6. Достаточные условия оптимальности алгоритма планирования вычислений в ЦОД 55
3.5. Выводы 59
4. Алгоритм построения сетей AFDX 60
4.1. Общая схема алгоритма 60
4.2. Описание процедур, используемых в алгоритме 61
4.2.1. Настройка параметров виртуальных каналов 61
4.2.2. Процедура агрегации 64
4.2.3. Построение маршрутов виртуальных каналов. 68
4.2.4. Процедура ограниченного перебора виртуальных каналов 70
4.2.5. Вычисление максимальной длительности и джиттера передачи сообщений. 70
4.2.6. Процедура переконфигурации виртуального канала. 72
4.3. Свойства алгоритмов и теоретическое обоснование 73
4.3.1. Оценка сложности этапов алгоритма 73
4.3.2. Корректность 77
4.3.3. Свойства процедуры ограниченного перебора при поиске маршрутов виртуальных каналов 78
4.3.4. Оценка точности процедуры построения маршрутов виртуальных каналов 80
4.3.5. Достаточные условия оптимальности алгоритма построения сетей AFDX. 80
4.3.6. Выводы 83
5. Экспериментальное исследование свойств алгоритмов 85
5.1. Методика проведения экспериментального исследования 85
5.2. Экспериментальные исследования для алгоритма планирования вычислений в ЦОД 86
5.2.1. Схема проведения экспериментов 86
5.2.2. Исследование используемых эвристик 87
5.2.3. Исследование точности алгоритма 92
5.2.4. Зависимость от метода выбора физического ресурса 94

5.2.5. Исследование процедуры репликации 96
5.2.6. Выводы 97
5.3. Экспериментальные исследования для задачи построения сетей AFDX 97
5.3.1. Схема проведения экспериментов 97
5.3.2. Эффективность использования процедур алгоритма 98
5.3.3. Исследование процедуры агрегации 100
5.3.4. Исследование работы алгоритма в случае высоких требований к длительности передачи сообщений 102
5.3.5. Выводы 104
6. Инструментальная система построения сетей AFDX 106
6.1. Требования к системе 106
6.2. Описание системы 107
6.2.1. Пользовательский интерфейс 107
6.2.2. Модуль построения виртуальных каналов 108
6.3. Выводы 110
Заключение 111
Литература 113
Приложение А. Корректность процедуры агрегации при построении сетей AFDX 119
Приложение Б. Доказательство утверждения о точности процедуры построения маршрутов в сетях AFDX 121
Приложение В. Доказательство утверждения о достаточных условиях выполнения ограничений на длительность передачи сообщений в сетях AFDX 123
Приложение Г. Доказательство утверждения о достаточных условиях построения виртуальных каналов для всех сообщений в сетях AFDX 126

Введение

В данной работе рассматривается задача планирования вычислений в распределенных системах реального времени с жесткими требованиями к качеству обслуживания. Рассматриваемые системы состоят из физических элементов (вычислительных ресурсов и ресурсов хранения данных) и сети обмена данными, состоящей, в свою очередь, из каналов передачи данных и сетевых элементов (коммутаторов и маршрутизаторов).

Задача планирования вычислений заключается в назначении как можно большего числа запросов на получение ресурсов в распределенной системе. Далее вместо термина «запрос на получение физических ресурсов» будем употреблять термины «виртуальный запрос» или «запрос». Запросы состоят из виртуальных элементов (виртуальных машин и виртуальных систем хранения данных), которые должны быть размещены на физических элементах распределенной системы, и виртуальных каналов, для которых должны быть проложены маршруты передачи данных между размещенными виртуальными элементами в сети обмена данными с учетом заданных ограничений.

Ограничения на размещение виртуальных ресурсов (виртуальных элементов и виртуальных каналов) зависят от конкретной решаемой задачи и используемых стандартов/протоколов. Можно выделить следующие виды ограничений:

- Ограничения на максимальную суммарную емкость размещаемых на физических элементах виртуальных элементов; при этом емкости могут задаваться как некоторой единичной характеристикой, так и набором (вектором) характеристик.
- Ограничения на максимальную пропускную способность каналов передачи данных и сетевых элементов.
- Ограничения на длительность передачи сообщений.
- Технологические ограничения, под которыми понимаются ограничения, обусловленные используемыми стандартами, а также особенностями аппаратных и программных средств системы. Например, ограничения на максимальный джиттер для периодически передаваемых кадров/сообщений (джиттер – разность между реальным временем начала выдачи кадра/сообщения и временем начала выдачи, определенным согласно периоду), ограничения на размер передаваемых кадров.

В данной работе задача планирования вычислений решается для двух классов систем: центры обработки данных (ЦОД) с моделью обслуживания Infrastructure-as-a-Service (IaaS) [1] и бортовые сети на основе стандарта Avionics Full Duplex Ethernet (AFDX) [2].

Модель обслуживания IaaS подразумевает, что планируемыми ресурсами являются виртуальные элементы: виртуальные машины и storage-элементы, и виртуальные каналы, объединяющие эти элементы. Пользователи такой системы могут вместо закупки и установки специального оборудования (серверов и систем хранения данных с прокладкой сетевых соединений между ними) разворачивать ресурсы аналогичной мощности и возможности в ЦОД в кратчайшие сроки.

Основными требованиями, задаваемыми при решении задачи планирования вычислений в ЦОД, являются требования к качеству обслуживания (SLA) [3], которые являются параметрами ограничений на возможные размещения виртуальных ресурсов. В процессе снятия/назначения виртуальных запросов может возникать фрагментация ресурсов, в результате чего новые запросы не могут быть назначены без нарушения SLA. Многие из используемых в настоящее время алгоритмов, такие как жадный алгоритм случайного размещения, используемый в системе Openstack [4], из-за возникающей фрагментации не позволяют эффективно использовать свободные физические ресурсы системы.

Для задачи планирования вычислений в ЦОД планируемыми ресурсами являются вычислительные узлы (серверы), на которых размещаются виртуальные машины; системы хранения данных, на которых размещаются виртуальные системы хранения данных (storage-элементы), и сеть обмена данными, через которую прокладываются маршруты виртуальных каналов.

Бортовые сети на основе стандарта AFDX используются в информационно-управляющих системах реального времени (ИУС РВ) для передачи данных между абонентами [5]. В настоящее время в бортовых системах широко используются системы, построенные на основе архитектуры интегрированной модульной авионики (ИМА) [6], в которых приложения, работающие в рамках ИУС РВ, изолированы друг от друга. Каждому приложению выделяется для работы набор временных интервалов (окон), в которых не могут выполняться задачи других приложений, и память, к которой не имеют доступа задачи других приложений. Стандарт AFDX описывает методы построения и управления сетями обмена данными для систем ИМА. Ограничение пропускной способности виртуальных каналов и длительности передачи сообщений в сетях, построенных согласно данному стандарту, выполняется путем ограничения размера и частоты передачи кадров, на которые делятся сообщения.

При построении сетей AFDX в качестве виртуальных запросов выступают периодически передаваемые сообщения, которые должны передаваться не менее одного раза в период и основным требованием к которым является непревышение максимальной

длительности передачи и максимального времени отклонения между соседними сообщениями. Для обеспечения требований реального времени сообщения передаются через выделенные соединения – виртуальные каналы, для которых фиксирован маршрут передачи и зарезервирована пропускная способность. Зная параметры виртуальных каналов, можно с помощью специальных техник оценить длительность передачи сообщений и обеспечить выполнение требований реального времени.

Цель диссертационной работы. Целью данной работы является разработка и исследование алгоритмов, основанных на сочетании жадных стратегий и ограниченного перебора, для решения задач планирования вычислений в распределенных системах реального времени с жесткими требованиями к качеству обслуживания.

Для достижения поставленной цели необходимо решить следующие задачи:

- Разработать и исследовать алгоритм планирования вычислений в центрах обработки данных с моделью обслуживания Infrustructure-as-a-Service, позволяющий производить назначение максимально возможного числа виртуальных запросов с заданными требованиями к качеству обслуживания.
- Разработать и исследовать алгоритм построения соединений для сетей со статически конфигурируемыми коммутаторами, позволяющий для заданного множества периодически передаваемых сообщений строить виртуальные каналы и маршруты для них с учетом заданных требований реального времени.
- На основе предложенного алгоритма решения задачи построения соединений для сетей, основанных на стандарте Avionics Full Duplex Ethernet, разработать программное инструментальное средство, позволяющее решать поставленную задачу и допускающее настройку на особенности конкретной задачи путем замены отдельных модулей.

Научная новизна. В диссертационной работе разработаны и исследованы алгоритмы планирования вычисления для двух классов систем, а именно для центров обработки данных с моделью обслуживания IaaS и для сетей на базе стандарта AFDX. Алгоритмы основаны на сочетании жадных стратегий и стратегий ограниченного перебора, которые позволяют обеспечить требуемый баланс между временем работы алгоритма и качеством получаемого решения. В алгоритме планирования вычислений в ЦОД используется возможность создавать реплики назначенных виртуальных систем хранения данных для оптимизации получаемого решения.

Для каждого алгоритма обоснована корректность, получена оценка вычислительной сложности. Доказан ряд утверждений о точности отдельных процедур алгоритмов и о достаточных условиях назначения всех запросов.

Практическая ценность. В рамках диссертационной работы на основе алгоритма построения сетей AFDX разработана инструментальная система. Инструментальная система представляет возможность запуска алгоритма с различными параметрами используемых процедур, возможность интерактивного редактирования входных данных и отображения результатов работы алгоритма, а также возможность модульной настройки на особенности конкретной задачи.

Методы исследования. При получении основных результатов диссертации использовались методы математического программирования, теории графов, теории расписаний и математической статистики.

Положения, выносимые на защиту:

1. Алгоритм согласованного отображения множества запросов на физические ресурсы центров обработки данных с моделью обслуживания Infrustructure-as-a-Service, позволяющий выбирать соотношение между вычислительной сложностью алгоритма и подмножеством размещенных запросов. Для алгоритма получено аналитическое выражение для вычислительной сложности в зависимости от глубины ограниченного перебора, обоснована корректность, сформулированы условия, при которых алгоритм размещает все запросы.

2. Алгоритм построения соединений для сетей со статически конфигурируемыми коммутаторами с соблюдением ограничений на качество работы сети. Для алгоритма получено аналитическое выражение для вычислительной сложности в зависимости от глубины ограниченного перебора, обоснована корректность, сформулированы условия, при которых алгоритм строит все соединения.

3. Реализованное на основе предложенного алгоритма построения соединений для сетей на базе стандарта Avionics Full Duplex Ethernet программное инструментальное средство, предоставляющее возможность запуска алгоритма с различными параметрами используемых процедур и возможность модульной настройки на особенности конкретной задачи.

4. Экспериментальный анализ влияния жадный критериев и используемых процедур на качество решения, получаемое разработанными алгоритмами, в результате которого выявлены наилучшие критерии в зависимости от класса исходных данных, показано улучшение качества получаемого решения при использовании процедуры агрегации соединений и процедуры репликации.

Апробация работы. Основные результаты работы докладывались на:

1. VII международной конференции «Исследование Операций (ORM2013)» (Москва, 2013);

2. Международной научной конференции «Управление и виртуализация в современных сетях (Сети 2014: SDN & NFV)» (Москва, 2014);
3. XIII международной научной конференции «Параллельные компьютерные технологии» (PaCT 2015)» (Петрозаводск, 2015);
4. II международной конференции «Инжиниринг & Телекоммуникации (En&T)» (Долгопрудный, 2015).
5. Всероссийской научно-технической конференции «Авиационные системы в XXI веке» (Москва, 2016).

Публикации. Основные результаты по теме диссертации изложены в 9 печатных работах [5, 15-18, 20, 61-63], 7 из которых изданы в журналах, рекомендованных ВАК [5, 15, 17, 18, 20, 61, 62].

Личный вклад автора заключается в выполнении основного объема теоретических и экспериментальных исследований, изложенных в диссертационной работе, включая разработку алгоритмов, построение их программных реализаций, проведение экспериментальных исследований, формулировку и доказательство приведенных утверждений, анализ и оформление результатов в виде публикаций и научных докладов.

В работе [5] Смелянскому Р.Л. и Костенко В.А. принадлежат формулировка проблемы и сравнение описанных подходов, Балашову В.В. принадлежит описание подхода, основанного на Fibre Channel, Шалимову А.В. – описание подхода, основанных на программно-конфигурируемых сетях, Вдовину П.М. принадлежит описание сетей AFDX.

В работе [15] Костенко В.А. принадлежит формулировка проблемы и подход к ее решению. Вдовину П.М., Зотову И.А. и Плакунову А.В. принадлежат реализации и экспериментальные исследования для алгоритмов планирования вычислений в ЦОД, соответственно, с раздельными планировщиками, с единым планировщиком и на основе схемы муравьиных колоний.

В работах [16, 17] Костенко В.А. и Смелянскому Р.Л. принадлежит постановка задачи, Вдовину П.М. – обзорная часть, реализация и экспериментальное исследование алгоритма планирования вычислений в ЦОД с раздельными планировщиками, Зотову И.А. и Плакунову А.В. принадлежат реализации и экспериментальные исследования для, соответственно, алгоритмов планирования вычислений в ЦОД с единым планировщиком и на основе схемы муравьиных колоний.

В работах [18, 20] Костенко В.А. принадлежат постановки задач, Вдовину П.М. – описание алгоритмов и теоретические и экспериментальные исследования их свойств.

Регистрация программы на ЭВМ [62] составлена на основе задачи, сформулированной Костенко В.А., и реализована Вдовиным П.М.

В работе [63] Вдовину П.М. принадлежит описание стандарта AFDX, Костенко В.А. принадлежит описание задач, возникающих при конфигурировании сетей AFDX, Балашову В.В. принадлежит описание методов исследования оценки длительности передачи кадров в сетях AFDX.

Объем и структура работы. Работа состоит из введения, шести глав, заключения и четырех приложений.

В первой главе представлены содержательные и формальные постановки задач планирования вычислений в распределенных системах реального времени с жесткими требованиями к качеству обслуживания. Для задач планирования вычислений в ЦОД и построения сетей AFDX представлены математические модели входных данных, описаны модели получаемых решений и ограничений к ним.

Во второй главе представлен обзор существующих методов решения как для описанных задач, так и для близких подзадач, которые могут возникать в процессе построения распределенных систем с жесткими требованиями к качеству обслуживания. На основе проведенного обзора обоснована необходимость разработки алгоритмов для решения сформулированных задач.

В третьей главе описан алгоритм планирования вычислений в ЦОД с моделью обслуживания IaaS для каждого типа ресурсов. Обоснована корректность описанного алгоритма, получена оценка вычислительной сложности в зависимости от глубины ограниченного перебора, исследованы теоретические свойства, а именно свойства отдельных процедур и достаточные условия назначения всех виртуальных запросов.

В четвертой главе описан алгоритм построения сетей AFDX. Обоснована корректность алгоритма, исследованы его вычислительная сложность, теоретические свойства отдельных процедур и достаточные условия построения виртуальных каналов и маршрутов для них для всех входных сообщений.

В пятой главе описаны проведенные экспериментальные исследования свойств алгоритмов. Для исследования свойств алгоритмов и эффективности их отдельных процедур проводилась серия запусков для случайно сгенерированных данных. На основе полученных результатов исследовались средние полученные значения количества назначаемых запросов и времени работы алгоритма. Кроме того, для обработки результатов экспериментов использовался метод проверки статистических гипотез.

В шестой главе приведено описание инструментальной системы построения сетей AFDX. Сформулированы требования к инструментальной системе, включающие в себя

возможность редактировать входные данные, отображать результаты работы алгоритма, формировать параметры запускаемого алгоритма, настраивать инструментальную систему на особенности конкретной задачи путем замены отдельных модулей. Описана архитектура реализованной инструментальной системы и показано, что она удовлетворяет сформулированным требованиям. Также показано отличие системы от существующих систем автоматического проектирования бортовых систем, включающих в себя возможность проектирования сетей AFDX.

В заключении сформулированы основные результаты диссертационной работы.

В приложении А представлено доказательство корректности используемой в алгоритме построения сетей AFDX процедуры агрегации сообщений, служащей для организации передачи нескольких сообщений в одном виртуальном канале.

В приложении Б приведено доказательство утверждения о точности процедуры построения маршрутов, используемой в алгоритме построении сетей AFDX.

В приложении В представлено доказательство утверждения о достаточных условиях выполнения условий на длительность передачи сообщений при решении задачи построения сетей AFDX. Данное утверждение используется при формулировке достаточных условий построения виртуальных каналов для всех сообщений, доказательство которых приведено в приложении Г.

Полный объем диссертации составляет 127 страниц с 16 рисунками и 13 таблицами. Список литературы содержит 70 наименований.

1. Описание задач планирования вычислений в распределенных системах реального времени

1.1. Задача планирования вычислений в распределенных системах

Распределенная система описывается в виде графа (который может быть как ориентированным, так и неориентированным), в котором вершинами являются физические элементы (вычислительные узлы и системы хранения данных) и сетевые элементы: коммутаторы, маршрутизаторы и оконечные системы, являющиеся интерфейсами между передающими данные абонентами и непосредственно сетью обмена данными. Ребрами (или дугами) графа являются каналы передачи данных.

Каналы передачи данных и сетевые элементы составляют сеть обмена данными. Основной характеристикой каналов передачи данных является пропускная способность передачи данных. Для сетевых и физических элементов задаются характеристики в виде вектора доступных ресурсов.

Входными данными задачи планирования являются виртуальные запросы, для которых заданы ограничения. Виртуальные запросы могут состоять из следующих элементов:

- Виртуальные элементы: виртуальные машины и storage-элементы;
- Коммуникационные каналы между виртуальными элементами, которые могут задаваться следующим образом:
    - в виде виртуальных каналов;
    - в виде периодически передаваемых сообщений, для которых требуется сформировать виртуальные каналы.

Для каждого виртуального элемента определен набор характеристик, который необходим для его разворачивания на физическом ресурсе. Эти характеристики являются параметрами ограничений корректности решений.

Для виртуальных каналов задается (либо формируется в случае формирования виртуальных каналов для заданного набора сообщений) требуемая пропускная способность передачи данных. Для каждого виртуального канала требуется построить маршруты в сети обмена данными. Суммарная требуемая пропускная способность на физических каналах передачи данных, резервируемая виртуальными каналами, не должна превышать максимальную доступную пропускную способность.

Для входных данных задаются требования качества сервиса, которые являются жесткими, то есть недопустимо невыполнение какого-либо из заданных требований. Кроме того, могут задаваться требования на длительность передачи сообщений. Задача заключается, таким образом, в назначении как можно большего числа виртуальных запросов и состоит из следующих подзадач:

- Назначение виртуальных элементов на физические таким образом, чтобы могли быть выполнены заданные требования к качеству обслуживания.
- Формирование виртуальных каналов для периодически передаваемых сообщений между назначенными виртуальными элементами.
- Построение маршрутов для сформированных виртуальных каналов с учетом требований к пропускной способности и длительности передачи сообщений.

В некоторых случаях (например, в задаче построения сетей AFDX) для виртуальных каналов требуется задать определенные характеристики, по которым определяется резервируемая пропускная способность. Кроме того, по виртуальным каналам могут передаваться несколько сообщений. В таких случаях после того, как известно, на какие физические элементы назначены виртуальные элементы, требуется шаг формирования виртуальных каналов с целью определения их параметров.

Наиболее близкими задачами с точки зрения гарантированного выполнения требований к качеству сервиса являются задачи построения расписаний для систем реального времени [7-14]. В этих задачах в качестве критериев SLA задаются ограничения на допустимое время выполнения прикладных программ или процессов программы, которые нарушаться не должны. Решениями таких задач являются расписания, в которых задается порядок запуска программ или процессов. В рассматриваемых задачах решениями являются статические назначения, определяющие привязки виртуальных элементов к физическим и маршруты виртуальных каналов в сети. Описанное отличие делает проблематичным использования алгоритмов построения расписаний для указанных задач.

1.2. Задача планирования вычислений в ЦОД с моделью обслуживания IaaS

В процессе решения задачи планирования вычислений в ЦОД с моделью обслуживания IaaS используется два типа физических элементов: вычислительные узлы и системы хранения данных, на которые назначаются, соответственно, виртуальные машины и storage-элементы. Для каждого из типов ресурсов задается свой вектор характеристик. Например, для вычислительных узлов и виртуальных машин: <количество

ядер процессора>, <тактовая частота работы ядер>, <размер памяти ОЗУ>; для хранилищ данных и storage-элементов: <размер дисковой памяти>, <количество операций считывания в секунду>, <количество операций записи в секунду> и т.п.

Еще одной особенностью задачи является возможность размещения реплик для storage-элементов. В случае, когда основной операцией со storage-элементами является чтение, канал для поддержания консистентности данных не требует большой пропускной способности. В таком случае можно создавать реплики для оптимизации размещения запросов [15].

В работах [16, 17] представлена математическая постановка описанной задачи и проведено сравнение различных подходов к распределению ресурсов в ЦОД. В данной работе рассматривается алгоритм отображения запросов на физические ресурсы для ЦОД с последовательным согласованным планированием для различных типов ресурсов [18] и приводятся результаты исследования его свойств.

1.2.1. Постановка задачи

Модель физических ресурсов ЦОД будем задавать графом $H = (P \cup M \cup K, L)$, где P – множество вычислительных узлов, M – множество систем хранения данных, K – множество сетевых элементов сети обмена ЦОД, L – множество физических каналов передачи данных. На множествах P, M, K и L определены векторные функции, аргументами которых являются соответственно элементы множеств P, M, K или L. Значениями функций выступают характеристики соответствующего вычислительного узла, системы хранения данных, сетевого элемента или канала передачи данных:

- $(ph_1, ph_2, \ldots, ph_{n1}) = fph(p), p \in P$,
- $(mh_1, mh_2, \ldots, mh_{n2}) = fmh(m), m \in M$,
- $(kh_1, kh_2, \ldots, kh_{n3}) = fkh(k), k \in K$,
- $(lh_1, lh_2, \ldots, lh_{n4}) = flh(l), l \in L$.

Предполагается, что для сетевых элементов и каналов передачи данных используются одинаковые наборы характеристики.

Ресурсный запрос будем задавать графом $G = (W \cup S, E)$, где W – множество виртуальных машин, используемых приложениями, S – множество виртуальных систем хранения данных (storage-элементов), E – множество виртуальных каналов передачи данных между виртуальными машинами и storage-элементами запроса. На множествах W, S и E определены векторные функции, аргументы которых – соответствующие элементы

множеств *W, S* или *E*. Значениями функций являются характеристики, запрашиваемые для виртуальной машины виртуальной машины, storage-элемента или виртуального канала:

- $(wg_1, wg_2, \ldots, wg_{n1}) = fwg(w), w \in W$,
- $(sg_1, sg_2, \ldots, sg_{n2}) = fsg(s), s \in S$,
- $(eg_1, eg_2, \ldots, eg_{n4}) = feg(e), e \in E$.

Набор характеристик SLA элемента запроса совпадают с набором характеристик соответствующего ему физического ресурса.

Назначением ресурсного запроса будем называть отображение

$A: G \to H = \{W \to P, S \to M, E \to \{K, L\}\}$.

Множество элементов $\{k_i\}$, $\{l_i\}$, на которые назначается виртуальный канал $e \in E$, формирует маршрут виртуального канала.

Выделим три типа отношений между характеристиками запросов и соответствующими характеристиками физических ресурсов. Обозначим через x_i емкость[1] по некоторой характеристике виртуального элемента i и через y_j – соответствующую ей емкость физического ресурса j. Тогда эти отношения можно записать следующим образом.

1. Недопустимость перегрузки емкости физического ресурса:

$$\sum_{i \in R_j} x_i \le y_j \qquad (1.2.1),$$

здесь R_j – множество запросов, назначенных на выполнение на физическом ресурсе j, x_i – соответствующая характеристика назначенного запроса.

2. Соответствие типа запрашиваемого ресурса типу физического ресурса:

$x_i = y_j$ (1.2.2).

3. Наличие требуемых характеристик у физического ресурса:

$x_i \le y_j$ (1.2.3).

Для каждой характеристики задан используемый тип отношения t_i, $t_i = \{1, 2, 3\}$.

Отображение $A: G \to H = \{W \to P, S \to M, E \to \{K, L\}\}$ *будем называть корректным*, если для всех физических ресурсов для любой характеристики i выполняются отношения t_i.

---

[1] Под емкостью в данном случае понимается значение по данной характеристики физического/виртуального элемента.

Отметим, что на каждый вычислительный узел (на каждую систему хранения данных) может быть назначено такое множество виртуальных машин (виртуальных систем хранения данных), для которого выполняются отношения t_i для всех характеристик.

Остаточным графом доступных ресурсов называется граф H_{res}, для которого переопределены значения функций по характеристикам, которые должны удовлетворять отношению (1.2.1):

$$fph_{res}(p) = fph(p) - \sum_{w \in W_p} fwg(w), \qquad fmh_{res}(m) = fmh(m) - \sum_{s \in S_m} fsg(s),$$

$$fkh_{res}(k) = fkh(k) - \sum_{e \in E_k} feg(e), \; flh_{res}(l) = flh(l) - \sum_{e \in E_l} feg(e) \qquad (1.2.4).$$

Здесь W_p – множество виртуальных машин, назначенных на выполнение на вычислительном узле p, E_l – множество виртуальных каналов, отображенных на физический канал l, E_k – множество виртуальных каналов, проходящих через сетевой элемент k, S_m – множество storage-элементов, размещенных в системе хранения данных m.

В качестве *исходных данных* задачи назначения ресурсных запросов на физические ресурсы заданы:

- множество запросов $Z = \{G_i\}$, поступивших планировщику;
- остаточный граф доступных ресурсов $H_{res} = (P \cup M \cup K, L)$.

Требуется: из множества Z разместить на выполнение в ЦОД максимальное число запросов, таких, что отображение A является корректным.

В случае, когда невозможно построить маршрут к некоторому storage-элементу, возможен вызов процедуры репликации, позволяющей дублировать данные на нескольких физических хранилищах данных. Данная проблема возникает, когда имеются виртуальные хранилища данных с высокой интенсивностью считывания и низкой интенсивностью записи. В этом случае для обеспечения консистентности данных не требуется канал с высокой пропускной способностью, и часть приложений может работать с виртуальной системой хранения данных, а другая – с его копией (или копиями), которая размещается на другой физической системе хранения данных [15, 17].

Репликацией называется отображение $R: H \to H$, дублирующее данные некоторого $s \in S$, назначенного на систему хранения данных $m \in M$, и создающее виртуальный канал поддержки консистентности данных $(m', l_1, k_1, ..., k_{n-1}, l_n, m): k_i \in K, l_i \in L, m' \in M$, m' – система хранения данных с репликой storage-элемента s.

Если storage-элемент является репликацией некоторой виртуальной системы хранения данных s, то в граф запрашиваемых ресурсов G добавляется вершина s'. В граф G также добавляется виртуальный канал между вершинами s и s', пропускная способность

которого определяется исходя из требования обеспечения консистентности репликации и базы данных.

На вход алгоритму назначения запросов на физические ресурсы подаются остаточный граф доступных ресурсов H_{res} и множество ресурсных запросов $\{G_i\}$. Множество $\{G_i\}$ формирует диспетчер задач. В него, кроме вновь поступивших запросов, могут входить и запросы, которые выполняются и для которых допустима миграция. Диспетчер задач также определяет время запуска планировщика.

Выходом алгоритма назначения запросов на физические ресурсы является множество назначений ресурсных запросов на физические ресурсы $\{A_i : G_i \to H, i = \overline{1,n}\}$ и множество репликаций $\{R_i\}, i = 0,1,\ldots$

1.3. Задача построения сетей AFDX

1.3.1. Описание сетей AFDX

Сеть AFDX представляет собой бортовую коммутируемую сеть; такие сети используются в настоящее время наряду с бортовыми авиационными сетями, построенными на основе каналов точка-точка и каналов множественного доступа с централизованным управлением [11, 19]. Преимущество коммутируемых сетей – использование меньшего количества коммутационной техники (в том числе проводов) и базирование на технологиях, распространенных и широко используемых в сетях Интернет.

Стандарт Avionics Full Duplex Ethernet (AFDX) [2] описывает управление бортовыми сетями на основе традиционного стандарта Ethernet 802.3. Согласно стандарту AFDX, сеть состоит из следующих элементов (рисунок 1.1):

- абоненты, передающие сообщения;
- оконечные системы – системы, обеспечивающие интерфейс между абонентами и сетью;
- пакетные коммутаторы, соединяемые каналами передачи данных.

Стек протоколов, используемый при передаче сообщений в бортовых сетях на основе стандарта AFDX, основан на семействе протоколов TCP/IP. На канальном уровне используется протокол Ethernet, при этом на данном уровне выполняется маршрутизация кадров в сети с помощью механизма виртуальных каналов. На сетевом уровне используется протокол IP, однако маршрутизация пакетов на этом уровне не производится, так как данная функция перекладывается на канальный уровень. На

транспортном уровне используется в основном протокол UDP (в некоторых случаях может быть использован протокол TCP).

Рисунок 1.1. Типичная архитектура бортовой сети на основе стандарта AFDX.

Каждый абонент подключен к оконечной системе, причем к одной оконечной системе может быть подключено несколько абонентов. Соблюдение ограничений на время передачи сообщений в сети AFDX достигается за счет выделения гарантированной пропускной способности соединения между каждой парой оконечных систем. Такое соединение может проходить через несколько пакетных коммутаторов и линий передачи данных. В AFDX соединение между оконечными системами называют виртуальным каналом. Передача данных между абонентами осуществляется путем передачи сообщений по виртуальным каналам, маршруты которых в физической сети определены заранее. Для каждого виртуального канала определена только одна оконечная система-отправитель и одна или более оконечная система-получатель. При этом по одному виртуальному каналу могут передаваться сообщения только от одного абонента. Возможность передачи нескольких сообщений по виртуальному каналу может использоваться для получения более оптимальной организации передачи сообщений в сети [20].

Сообщение от абонента попадает на оконечную систему через специально выделенный порт. На оконечной системе-отправителе сообщение разбивается на порции данных (кадры), которым сопоставляются заголовки соответствующих уровней стека TCP/IP – а именно, транспортного (UDP), сетевого (IP), канального. Полученные в результате разбиения кадры выдаются на физический канал передачи данных. В каждый кадр в поле MAC-адреса получателя записывается номер виртуального канала. Этот номер используется коммутаторами AFDX для маршрутизации кадра. Стандартная для Ethernet маршрутизация по MAC-адресу в AFDX не используется. После доставки кадра на оконечную систему-получатель (возможно, одну из нескольких) он буферизуется для последующей сборки сообщения. После прихода последнего кадра собранное сообщение направляется абонентам-получателям.

Надежность передачи данных в сетях AFDX обеспечивают за счёт резервирования. Каждая оконечная система соединена с двумя независимыми идентичными сетями AFDX. Кадры передают в обе сети (в каждой сети кадр следует по идентичным маршрутам), и если в одной из них диагностируется ошибка передачи кадра (например, не совпадает контрольная сумма), то кадр берется из той сети, где не было ошибки. На оконечной системе-получателе происходит проверка целостности кадров, и если кадр уже был принят из другой сети, то кадр-дубликат сбрасывается.

Таблицы маршрутизации коммутатора AFDX настраиваются заранее. Коммутаторы AFDX также выполняют контроль и фильтрацию трафика. При фильтрации трафика проверяется правильность следования кадров по виртуальным каналам, а также их целостность. Контроль трафика обеспечивает заданную для виртуального канала пропускную способность и не позволяет ее превышать. Для контроля пропускной способности канала в сетях AFDX применяется алгоритм текущего ведра с маркерами. Допустимую пропускную способность каждого виртуального канала устанавливают заранее при конфигурировании коммутатора. Таким образом, маршрутизация в сети AFDX задается статически, возможность динамически изменять таблицы маршрутизации стандартом не предусмотрена.

Важным параметром при построении виртуальных каналов является джиттер. При порождении сообщения джиттер описывает разность времени порождения сообщения от начала периода. Задание максимального джиттера при порождении сообщения, таким образом, позволяет описывать случаи, когда сообщения порождаются не строго регулярно.

Джиттер также возникает при выдаче кадров, на которые делятся сообщения, в сеть обмена данными. На виртуальных каналах пропускная способность регулируется с помощью разбиения сообщений на кадры и передачи кадров с заданной частотой. Из-за мультиплексирования кадров разных виртуальных каналов кадры не могут быть переданы строго периодически, и отклонения от начала периода регулируются с помощью задания максимального джиттера виртуального канала. Данный параметр более строго описан в разделе 1.3.2. .

При построении сетей AFDX для заданного множества периодически сообщений требуется построить множество виртуальных каналов с учетом следующих ограничений:

- Непревышение выделенной пропускной способности каналов передачи данных;
- Ограничения на частоту порождения сообщений;
- Ограничения на длительность и джиттер передачи сообщений;

- Ограничения на джиттер при выдачи кадров виртуальных каналов в канал передачи данных.

1.3.2. Постановка задачи

Сеть AFDX будем описывать ориентированным графом $G = (N \cup K, E)$, где N – множество оконечных систем, K – множество коммутаторов, E – множество дуг, соединяющих элементы из множества $N \cup K$. Для каждой дуги $e \in E, e = (k_1, k_2)$ существует парная дуга[2] $e' \in E, e' = (k_2, k_1)$; для каждой дуги e задана пропускная способность R_e, причем $R_e = R_{e'}$.

Для каждой оконечной системы $n \in N$ задано множество абонентов $A_n \subseteq A$ (A – множество всех абонентов), которые к ней подключены, причем каждый абонент подключен только к одной оконечной системе. Каждая оконечная система подключена только к одному каналу передачи данных[3].

Входными данными задачи являются потоки данных между абонентами, которые задаются в виде множества периодически передаваемых сообщений $MSG = \{msg\}$, для каждого из которых задаются следующие параметры:

- $size_{msg}$ (байт) – размер сообщения;
- T_{msg} (мс) – период передачи сообщения;
- J_{msg} (мкс) – максимальный джиттер порождения сообщения внутри периода (интервал времени, в котором может быть порождено сообщение от начала периода);
- $src_{msg} \in A$ – абонент-отправитель сообщения;
- $\{dst_{msg}\} \subset A$ – множество абонентов-получателей сообщения, абоненты-получатели должны быть подключены к оконечным системам, отличным от оконечной системы-отправителя: $dst_{msg} \notin A_n : src_{msg} \in A_n$.

Для каждого сообщения заданы следующие ограничения:

---

[2] Парные дуги соответствуют каналу передачи данных в полнодуплексном режиме.

[3] Согласно стандарту, каждая оконечная система подключена к двум идентичным сетям, и кадры передаются одновременно через обе сети, что позволяет улучшить надежность передачи; в данной работе не рассматриваются аспекты надежности передачи, поэтому исследуется передача данных только через одну сеть, результаты для другой сети аналогичны результатам для первой сети.

- сообщение должно передаваться не менее одного раза в период[4];
- τ_{msg} (мс) – максимальная длительность передачи сообщения с момента выдачи сообщения от абонента оконечной системе-отправителю до момента получения сообщения всеми абонентами-получателями;
- $J*_{msg}$ (мс) – максимальный джиттер передачи сообщения (разность между возможными максимальной и минимальной длительностью передачи сообщения).

Задача заключается в построении множества виртуальных каналов $VL=\{vl\}$ и построении для них маршрутов в сети передачи данных. По одному виртуальному каналу может передаваться несколько сообщений. Сообщения на оконечной системе-отправителе разбиваются на кадры, и в сеть обмена оконечная система передает кадры. Сообщение считается принятым, когда получен последний его кадр. Для каждого виртуального канала должны быть определены следующие параметры:

- LM_{vl} (байт) – максимальный размер кадра, передаваемый по данному виртуальному каналу; согласно стандарту, размер кадра лежит в следующих пределах: $64 \le LM_{vl} \le 1518$ байт, размер заголовка кадра фиксирован и равен c=47 байт, расстояние между кадрами равно gap=12 байт;
- BAG_{vl} (мс) – минимальный промежуток времени между передачей кадров при нулевом джиттере порождения кадров; согласно стандарту, данное значение лежит в промежутке от 1 до 128 мс и является степенью двойки;
- JM_{vl} – максимальный джиттер порождения кадров на оконечной системе-отправителе (разность между реальным временем начала выдачи кадра и временем начала выдачи, определенным согласно периоду, см. рисунок 1.2); данное значение используется при мультиплексировании виртуальных каналов и пояснено далее;
- $S_{vl} \in N$ – оконечная система-отправитель кадров данного виртуального канала;
- $D_{vl} = \{d_{vl}\} \subseteq N$ – множество оконечных систем-получателей кадров данного виртуального канала ($S_{vl} \notin D_{vl}$);
- $Tree_{vl} \subseteq E$ – маршруты передачи кадров в сети, представляет собой дерево, в котором корнем является S_{vl} , а листьями являются все элементы множества D_{vl};
- Множество сообщений $MSG_{vl}=\{msg\}$, передаваемых по данному виртуальному каналу; сообщения должны исходить из одного абонента, подключенного к S_{vl},

[4] Описанный в данной работе алгоритм рассчитан на случай, при котором сообщение порождается один раз в период (с учетом джиттера); если сообщение порождается чаще одного раза в период, то передача всех его кадров не гарантируется, при этом кадры могут сбрасываться на коммутаторах.

абоненты-получатели должны быть подключены к оконечным системам из множества D_{vl}.

На рисунке 1.2 показано, что кадры виртуальных каналов *vl1* и *vl2* не могут быть переданы строго регулярно (с интервалом между кадрами, равным *BAG*). При сдвиге кадров виртуального канала *vl1* относительно BAG_{vl1} на величину, не превышающую JM_{vl1}, кадры могут быть переданы, не нарушая регулярности передачи.

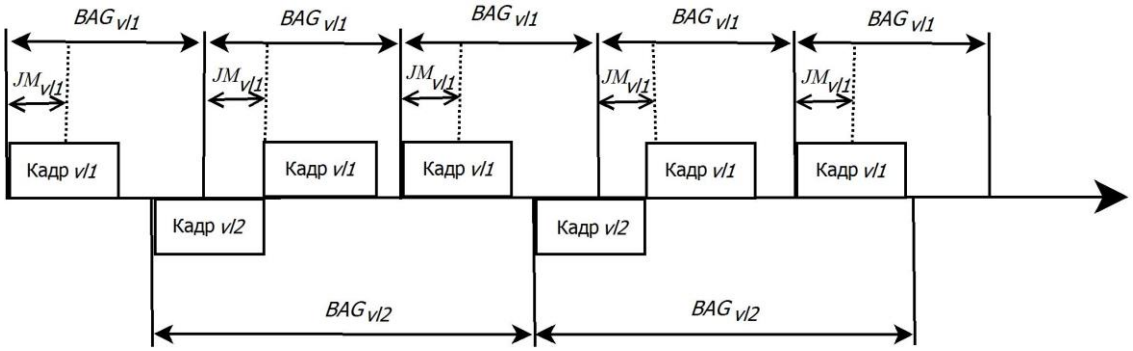

Рисунок 1.2. Мультиплексирование виртуальных каналов с помощью джиттера.

При построении виртуальных каналов и маршрутов для них должны выполняться следующие ограничения:

- Суммарная пропускная способность, зарезервированная под виртуальные каналы, проходящие через физический канал передачи данных *e*, не превосходит его пропускной способности:

$$\forall e \in E : \sum_{vl \in e} \frac{LM_{vl}}{BAG_{vl}} \le R_e \qquad (1.3.1),$$

здесь R_e - пропускная способность физического канала передачи данных *e* .

- Частота передачи кадров каждого виртуального канала не превосходит частоты выдачи кадров в канал:

$$\forall vl \in VL : \sum_{msg \in MSG_{vl}} \left(\left\lceil size_{msg} / (LM_{vl} - c) \right\rceil / T_{msg} \right) \le 1 / BAG_{vl} \qquad (1.3.2).$$

Данное ограничение возникает из того, что все кадры одного сообщения *msg* должны поступить в канал до выдачи следующего сообщения *msg*, то есть за период T_{msg}. Учитывая, что сообщение *msg* делится на количество кадров, равное $\left\lceil size_{msg} / (LM_{vl} - c) \right\rceil$, получаем ограничение (1.3.2).

- Максимальный джиттер на оконечных системах-отправителях не должен превосходить 500 мкс[5]:

$$\forall vl \in VL : JM_{vl} \le 0.5мс \quad (1.3.3)\ .$$

- Максимальная длительность передачи каждого сообщения и максимальный джиттер не превосходят заданных ограничений:

$$\forall msg \in MSG : Dur(msg) \le \tau_{msg}; Jit(msg) \le J*_{msg} \quad (1.3.4),$$

здесь *Dur(msg)* и *Jit(msg)* - процедуры вычисления оценок длительности передачи сообщений и их джиттеров соответственно, данные процедуры описаны далее в работе в разделе 4.2.5. .

1.4. Выводы

Сформулированные задачи планирования вычислений в ЦОД и построения сетей AFDX можно декомпозировать на следующие подзадачи:

1. Назначение виртуальных элементов на физические с учетом заданных требований к качеству сервиса.
2. Формирование виртуальных каналов между назначенными виртуальными элементами.
3. Построение маршрутов для сформированных виртуальных каналов с учетом требований к пропускной способности и длительности передачи сообщений.

Для задачи планирования вычислений в ЦОД актуальными являются 1 и 3 из выделенных подзадач, так как требования к виртуальным каналам являются известными исходя из заданных виртуальных запросов. Для задачи построения сетей AFDX актуальными являются подзадачи 2 и 3, так как отправители и получатели сообщений известны.

Таким образом, общими для двух поставленных задач являются следующие особенности:

- Представление системы в виде графа, состоящего из физических узлов и сети передачи данных;

[5] Данное требование исходит из рекомендаций, описанных в стандарте [2]. Согласно стандарту, вычисленный максимальный джиттер может превышать 500 мкс, но при этом его значение должно быть установлено в 500 мкс. В таком случае реальная пропускная способность виртуальных каналов может не соответствовать заявленной, поэтому в данной работе непревышение вычисленного джиттера 500 мкс ставится как ограничение.

- Необходимость построения маршрутов виртуальных каналов между известными физическими узлами с заданными требованиями на пропускную способность/длительность передачи сообщений;
- Решение задачи должно удовлетворять заданным жестким требованиям к качеству сервиса.

2. Обзор различных подходов к построению алгоритмов решения близких задач

В данной главе описываются подходы к решению не только описанных задач, но и некоторых близких задач и подзадач, на которые декомпозируются сформулированные задачи.

Для задачи планирования вычислений в ЦОД с моделью обслуживания IaaS должны быть учтены следующие особенности.

1. Возможность задания SLA для всех типов ресурсов ЦОД для произвольного набора характеристик. Для этого вычислительные ресурсы, системы хранения данных, сетевые ресурсы должны рассматриваться как планируемые типы ресурсов, характеристики ресурсов должны задаваться в виде векторов характеристик и типов их ограничений 1.2.1-1.2.3. Планирование должно проходить согласовано в смысле соблюдения соглашений о качестве сервиса.

2. Возможность устранения перегрузки входящих в системы хранения данных каналов передачи данных. Данная задача возникает, когда имеются виртуальные системы хранения данных с высокой интенсивностью считывания и низкой интенсивностью записи. Она может быть решена, если допускается репликация виртуальных систем хранения данных. В этом случае для обеспечения консистентности данных не требуется канал с высокой пропускной способностью, и половина приложений может работать с виртуальной системой хранения данных, а другая - с его копией, которая располагается в другой физической системе хранения данных (или несколькими копиями, которые располагаются в других системах хранения данных).

Подходы к решению описанных задач описаны в разделе 2.1.

При решении задачи планирования ресурсов в ЦОД с различными планировщиками ресурсов задача разделяется на подзадачу назначения виртуальных элементов (виртуальных машин и storage-элементов) на физические элементы (вычислительные узлы и системы хранения данных соответственно) и на подзадачу построения маршрутов виртуальных каналов. Первая подзадача соответствует задаче упаковке в контейнеры и описана в разделе 2.2. Задача построения маршрутов виртуальных каналов описана в разделе 2.3.

При решении задачи построения сетей AFDX выделяются следующие требования к решению:

1. Для периодически передаваемых сообщений с возможностью отклонения выдачи сообщения от периода (не более чем на величину J_{msg}) должно производиться построение виртуальных каналов согласовано с учетом ограничений 1.3.1 – 1.3.4.
2. Должна быть возможность проведения агрегации сообщений в один виртуальный канал с целью увеличения количества размещенных сообщений.

Подходы к решению задачи построения сетей AFDX описаны в разделе 2.4, методы построения маршрутов виртуальных каналов – в разделе 2.3.

Одним из этапов решения задачи является оценка длительности передачи сообщений, подходы к решению которой описаны в разделе 2.5.

2.1. Подходы к решению задачи отображения запросов в ЦОД

Существующие алгоритмы решения задачи планирования вычислений в ЦОД с моделью обслуживания IaaS можно условно разделить на алгоритмы с единым планировщиком для всех видов ресурсов [21-27, 64-69] и алгоритмы с раздельными планировщиками [28-34]. Работы [24-29] описывают алгоритмы для решения задачи назначения виртуальных сетей, постановка которой описана ниже. Работы [64-69] посвящены задаче вложения графов, которая используется в основном при решении задач планирования вычислений в параллельных вычислительных комплексах. Краткое описание этой задачи приведено ниже. Работы [30-34] посвящены решению задачи упаковки в контейнеры применительно к задаче планирования вычислений в ЦОД и описаны в разделе 2.2. В работе [35] представлено решение, основанное на предположении, что возможно разделение маршрута виртуального канала на несколько потоков от отправителя к получателю. Такое предположение позволяет эффективнее использовать ресурсы сети обмена данными, используя решение задачи о многопродуктовом потоке [36]. В представленной в данной работе постановке задачи данное предположение не ставится и разделение маршрута не производится.

Задача построения виртуальных сетей схожа с задачей назначения в ЦОД и заключается в следующем:

- Как физическая, так и виртуальная сеть представляются в виде графа, состоящего только из вычислительных узлов, соединенных между собой каналами передачи данных.
- В задачах могут задаваться ресурсные ограничения/требования для физических элементов (производительность) и для каналов передачи данных (пропускная способность).

- Целью задачи является назначение виртуальных сетей на физическую сеть, при котором любой виртуальный элемент может быть назначен на *любой* физический элемент и *никакие два виртуальных элемента одной виртуальной сети не могут быть назначены на один физический*.

Таким образом, в отличие от задачи планирования вычислений в ЦОД, в данной проблеме не рассматриваются сетевые элементы как ресурс. При этом каждый физический элемент может быть использован как сетевой элемент любого виртуального канала. Другим отличием является необходимость назначения на каждый физический элемент не более одного виртуального элемента.

Задача вложения графов [64-69] возникает при вложении параллельных программ на вычислительную многопроцессорную среду. Программа представляется в виде графа, для которого требуется построить отображение в заданный граф вычислительной среды, минимизирующее оценку времени выполнения программы.

Вложение некоторого графа G в граф H – это отображение, при котором каждой вершине исходного графа G ставится в соответствие вершина графа H, каждому ребру графа G ставится в соответствие цепь в графе H. В некоторых случаях [70] допускается перегрузка вершин и ребер, при возникновении которой через вершину и ребро графа H может проходить более одной цепи. При перегрузке степени 1 вложение графа эквивалентно гомеоморфному отображению.

Основное отличие задачи вложения графов от поставленной задачи назначения виртуальных запросов в ЦОД – отсутствие требований к качеству обслуживания. Любое вложение, возникающее в процессе решения, является корректным, и среди вложений производится поиск оптимального с точки зрения времени выполнения программы.

Возможность применимости исследуемых алгоритмов для решения сформулированной задачи будем определять согласно следующим критериям.

- Вычислительные узлы рассматриваются как планируемый тип ресурса.
- Системы хранения данных рассматриваются как планируемый тип ресурса.
- Сетевые ресурсы: сетевые элементы и каналы передачи данных, рассматриваются как планируемый тип ресурса.
- Алгоритм предоставляет возможность построения маршрутов виртуальных каналов через физическую сеть передачи данных.
- Алгоритм предоставляет возможность строить репликации хранилищ данных с целью оптимизации назначения запросов.
- Для планируемых ресурсов задаются векторы характеристик.

В таблице 2.1 приведено сравнение подходов согласно описанным выше критериям. Как видно из таблицы, ни один из алгоритмов не удовлетворяет всем критериям. Одним из требований, не выполненном ни в одной из перечисленных работ, является возможность строить репликации систем хранения данных. Алгоритм из работы [22] удовлетворяет всем остальным критериям с тем ограничением, что он ориентирован только на ЦОД с древовидной топологией. Остальные рассмотренные алгоритмы не удовлетворяют еще хотя бы одному критерию.

Приведенное исследование поясняет актуальность построения алгоритма сформулированной задачи планирования вычислений в ЦОД.

Таблица 2.1. Сравнительная характеристика алгоритмов

Рассмотренные работы	Вычисл. узлы как планируемый тип ресурсов	Системы хранения данных как планируемый тип ресурсов	Сетевые ресурсы как планируемый тип ресурсов	Возможности		
				построение маршрутов вирт. каналов	репли кация	задание ресурсов в виде вектора характеристик
Optimal Mapping of Virtual Networks with Hidden Hops [28]	-	-	-	+	-	-
Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration [29]	+	-	-	+	-	-
A Virtual Network Mapping Algorithm based on Subgraph Isomorphism [27]	+	-	-	+	-	-
Algorithms for Assigning Substrate Network Resources to Virtualized network	+	-	-	-	-	-

Resources [26]						
Virtual network embedding with coordinated node and link mapping [25]	+	-	-	+	-	-
Virtual Network Embedding Through Topology-Aware Node Ranking [24]	+	-	-	+	-	-
Coupled placement in modern data centers [23]	+	+	+	±[6]	-	-
Server-storage virtualization: integration and load balancing in data centers [22]	+	+	+	±[6]	-	+
Joint VM placement and routing for data center traffic engineering [21]	+	-	+	+	-	+

2.2. Упаковка в контейнеры

Входными данными задачи упаковки в контейнеры являются набор контейнеров с заданной свободной емкостью/вектором емкостей и множество элементов с заданными требуемой емкостью/вектором емкостей. Набор характеристик (каждой характеристике соответствует значение в векторе емкостей контейнера/элемента) у контейнеров и элементов должен совпадать. Упаковка в контейнеры заключается в назначении элементов в наименьшее количество контейнеров, при котором суммарные требуемые емкости элементов не превышают емкостей контейнеров, на которые они назначены. Задача упаковки в контейнеры является NP-трудной [37].

Данная задача решается при назначении виртуальных элементов на физические узлы. Существует ряд подходов как для одномерной, так и для многомерной упаковки. Для одномерной упаковки известны жадные алгоритмы и теоретические результаты исследования точности [37].

[6] Алгоритмы применимы лишь для ЦОД, имеющих иерархическую (древовидную) топологию.

Для исследования точности работы жадных алгоритмов при одномерной упаковке используется понятие асимптотической точности. Асимптотическая точность определяется как отношение в наихудшем случае количества контейнеров, назначаемое процедурой, к количеству контейнеров, получаемому оптимальным алгоритмом, при стремлении количества элементов к бесконечности. Известны следующие основные жадные стратегии упаковки:

- Следующий подходящий (Next Fit, NF) – элемент назначается в текущий выбранный контейнер, если имеется достаточно емкости, иначе производится переход к следующему контейнеру.
- Первый подходящий (First Fit, FF) – для каждого элемента контейнеры рассматриваются в порядке поступления (без сортировки) и выбирается первый контейнер с достаточным количеством ёмкости.
- Лучший подходящий (Best Fit, BF) – элементы рассматриваются в произвольном порядке, на каждом шаге выбирается контейнер с наименьшим достаточным количеством емкости.
- Первый подходящий по убыванию (First Fit Decreasing, FFD) – элементы рассматриваются в порядке убывания требуемых емкостей и назначаются в первый контейнер, куда они помещаются.
- Лучший подходящий по убыванию (Best Fit Decreasing, BFD) – элементы рассматриваются в порядке убывания требуемых емкостей и назначаются в контейнер с наименьшим достаточным количеством емкости.

Согласно известным теоретическим результатам [37], NF имеет асимптотическую сложность 2, FF и BF – асимптотическую сложность 1.7, FFD и BFD – асимптотическую сложность 11/9.

Следует отметить, что стратегии NF, FF и BF подходят для упаковки в реальном времени, т.е. такой упаковки, при которой изначальное множество назначаемых элементов неизвестно и элементы поступают один за другим. Стратегии FFD и BFD могут быть использованы только когда все множество элементов заранее известно.

Некоторые работы исследуют задачу упаковки применительно к виртуальным машинам/storage-элементам ЦОД [30-34]. В данных работах предлагаются различные подходы для обеспечения назначения виртуальных элементов, включающие в себя не только упаковку, но и балансировку ресурсов. Балансировка подразумевает равномерное распределение ресурсов по имеющимся контейнерам. Балансировка может быть актуальна в сформулированной задаче в случае, когда критическим ресурсом является сеть передачи данных. В этом случае целесообразно распределять виртуальные элементы различных

типов более равномерно для предоставления большего количества физических ресурсов сети при построении маршрутов виртуальных каналов.

В таблице 2.2 указаны характеристики алгоритмов и описаны их основные особенности. Алгоритмы из работ [32-34] можно использовать при решении каждой из подзадач упаковки при назначении виртуальных машин и storage-элементов в процессе решения задачи планирования вычислений в ЦОД.

Таблица 2.2. Сравнительная характеристика алгоритмов

Рассмотренные работы	Возможности		Комментарии
	задание ресурсов в виде вектора характеристик	балансировка ресурсов	
Application placement on a cluster of servers [30]	-	-	Представлено несколько алгоритмов одномерной упаковки
Cloud Storage and Online Bin Packing [31]	-	+	Представлен алгоритм сбалансированной одномерной упаковки применительно к задаче назначения storage-элементов
Resource Scheduling in Data Centers using MRIS [32]	+	-	Представлен жадный алгоритм, который выполняет жадную упаковку, основываясь на дефиците ресурса; элементы назначаются в порядке убывания требований к дефицитным ресурсам
Quantifying load imbalance on virtualized enterprise servers [33]	+	+	Представлен алгоритм назначения виртуальных машин с балансировкой ресурсов, при этом вычисляется общая степень разбалансировка по всем ресурсам и в случае ее превышения некоторого порогового значения производится переназначение элементов с целью уменьшения степени

			разбалансировки
On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach [34]	+	+	Представлен алгоритм назначения виртуальных машин с балансировкой ресурсов, при этом, в отличие от алгоритма, представленного в работе [33], степень разбалансировки проверяется по каждому ресурсу

2.3. Построение маршрутов виртуальных каналов

При решении задачи построения маршрутов виртуальных каналов возможны следующие варианты постановок:

- Однодуплексный/полнодуплексный режим при передаче по физическим каналам передачи данных. Соответственно, в зависимости от режима в постановке используется неориентированный/ориентированный граф сети.
- Количество получателей в виртуальном канале: возможна постановка с ровно одним получателем, либо с непустым множеством получателей. В первом случае маршрут представляет собой путь в графе от отправителя к получателю, во втором – дерево, являющееся подграфом исходного графа.

При решении задачи планирования вычислений в ЦОД используется однодуплексный режим и виртуальные каналы с одним получателем. При построении сетей AFDX, соответственно, используется полнодуплексный режим и виртуальные каналы с множеством получателей.

Существующие алгоритмы маршрутизации можно условно разделить на два типа:

1. Алгоритмы поиска маршрутов сразу для всех виртуальных каналов; данные алгоритмы основаны на постановке задачи оптимизации, соответствующей задаче построения маршрутов виртуальных каналов, и решении этой задачи одним из известных методов; недостатком таких алгоритмов является то, что при невозможности построения сразу всех маршрутов алгоритм возвращает неуспех.
2. Алгоритмы последовательного поиска маршрутов виртуальных каналов; для каждого виртуального канала строится маршрут одним из известных алгоритмов, в случае невозможности построения маршрута производится переход к следующему виртуальному каналу. В отличие от предыдущего метода, может быть построено неоптимальное решение, однако возможно построение маршрутов для подмножества виртуальных каналов.

В таблице 2.3 представлены описания алгоритмов в зависимости от постановки (режим передачи данных, количество получателей) и используемого типа алгоритмов.

Таблица 2.3. Сравнительная характеристика алгоритмов построения маршрутов

Рассмотренные работы	режим передачи данных	количество получателей	Комментарии
On path selection for traffic with bandwidth guarantees [38]	однодуплексный	один	Используются алгоритмы Беллмана-Форда и Дайкстры с различными критериями выбора маршрута.
Optimal Mapping of Virtual Networks with Hidden Hops [28], Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration [29]	однодуплексный	один	Используется алгоритм поиска k кратчайших путей в графе для построения маршрутов; алгоритм позволяет использовать два критерия при построении маршрута, например, выбрать самый широкий путь (с наибольшей остаточной пропускной способностью) среди k найденных кратчайших путей.
Optimal design of virtual links in AFDX networks [39]	полнодуплексный	множество	Представлен алгоритм первого типа, который производит назначение виртуальных каналов с помощью формулировки и решения смешанного целочисленного программирования (Mixed Integer Linear Programming (MILP)) построения минимальных деревьев Штейнера для виртуальных каналов.
Multicast routing and its QoS extension: problems, algorithms, and protocols [40]	полнодуплексный	множество	Представлено описание различных алгоритмов как первого, так и второго типа для построения маршрутов с множеством получателей в сетях, представляемых ориентированным

			графом
Explicit multicast routing algorithms for constrained traffic engineering [41]	полнодуплексный	множество	Представлен алгоритм второго типа для поиска дерева групповой передачи данных, который сводится к алгоритмам поиска маршрута для одного получателя.

Среди представленных работ при построения маршрутов в ЦОД применимыми являются все алгоритмы построения маршрутов с одним получателем. Более предпочтительными являются алгоритмы второго типа, так как они предоставляют возможность перестроения маршрутов в случае неуспешного назначения. Таким образом, в задаче планирования вычислений в ЦОД предлагается использовать алгоритмы Дейкстры и Беллмана-Форда [38] для оптимизации по одному критерию оптимальности маршрута, либо алгоритм поиска k кратчайших путей [28] для оптимизации по двум критериям. Использование алгоритма поиска k кратчайших путей при этом увеличит время работы алгоритма, но при этом может привести к назначению большего количества запросов.

При решении задачи построения сетей AFDX можно использовать любой алгоритм построения маршрутов с множеством получателей. Предпочтительными являются алгоритмы второго типа, как и при решении задачи планирования вычислений в ЦОД, поэтому при построении маршрутов можно использовать алгоритм, представленный в работе [41].

2.4. Подходы к проектированию сетей AFDX

В литературе среди работ, посвященных проектированию сетей AFDX, основное внимание уделяется задаче нахождения оценки длительности передачи кадров. Описание работ по данной тематике приведено в следующем разделе.

Алгоритмы, посвященные непосредственно проектированию сетей AFDX, описаны в работах [39, 42-44].

В работе [42] используется возможность задавать статический приоритет виртуальных каналов на портах коммутаторов для оптимизации времени передачи

кадров[7]. При решении задачи оптимизации времени передачи кадров для вычисления оценки длительности передачи используется Network Calculus.

Метод построения виртуальных каналов для приведенной задачи описан в статьях [39, 43, 44]. В работе [39] приведен оптимальный алгоритм проектирования виртуального канала для передачи по нему не более одного сообщения с заданными ограничениями на время выдачи последнего кадра в сеть (то есть время фактической передачи кадра по сети не учитывается). В работе также предложена процедура агрегации и алгоритм построения маршрутов виртуальных каналов на основе решения задачи смешанного целочисленного программирования (Mixed Integer Linear Programming (MILP)). Описанный подход обладает следующими недостатками:

1. Процедура агрегации подразумевает, что при передаче разных сообщений происходит «склейка» их кадров, то есть в одном кадре могут передаваться данные разных сообщений. В сетях AFDX возможность «склейки» может отсутствовать.

2. При решении задачи построения маршрутов ставится задача целочисленного линейного программирования сразу для всех виртуальных каналов, и при неуспешном решении не строится ни одного маршрута.

3. Описанные этапы (проектирование виртуального канала, агрегация и маршрутизация) решаются последовательно и независимо, и при неуспешном построении маршрутов не производится попытка, например, по-другому агрегировать виртуальные каналы.

4. В итоговом построении не учитываются ограничения (1.3.2-1.3.4). Кроме того, в постановке задачи отсутствуют ограничения джиттер порождения сообщения внутри периода J_{msg}.

В работе [43] предложен алгоритм проектирования допустимых конфигураций виртуальных каналов (то есть параметров *BAG* и *LM*) с учетом заданных ограничений на период передачи сообщений, пропускную способность сети и ограничений на максимальный джиттер виртуальных каналов. Описанный подход обладает следующими недостатками:

[7] Согласно стандарту, на выходном порту коммутатора можно статически определить приоритет виртуального канала. Стандарт позволяет задать две очереди высокого и низкого приоритета, и кадры очереди низкого приоритета не могут начать передачу, пока не выданы кадры из очереди высокого приоритета.

1. В постановке задачи отсутствуют ограничения на длительность τ_{msg} и джиттер $J*_{msg}$ передачи сообщения, джиттер порождения сообщения внутри периода J_{msg}.
2. Не ставится ограничение (1.3.4).
3. В постановке задачи отсутствует понятие графа сети, и ограничение (1.3.1) ставится как непревышение пропускной способности сети, т.е. доступная пропускная способность представляется как некоторая величина *B*, и суммарная требуемая пропускная способность виртуальных каналов не должна ее превышать.
4. Предполагается, что сообщения уже распределены по виртуальным каналам.
5. Не представлены алгоритмы построения маршрутов виртуальных каналов.

Работа [44] расширяет результаты работы [43] с помощью алгоритма распределения сообщений по виртуальным каналам. Таким образом, из представленных недостатков убирается пункт 4, при этом остальные недостатки остаются в силе.

Описанные недостатки показывают, что поставленная задача полностью не решается в указанных работах. Отсутствие подходов к решению задачи в других работах показывает, что требуется построение алгоритма ее решения.

Стоит отметить, что подходы, используемые в описанных работах, могут быть использованы для решения подзадач в процессе решения описанной задачи, а именно, при формировании параметров виртуальных каналов и распределении сообщений по виртуальным каналам (агрегации сообщений).

2.5. Вычисление оценки длительности передачи сообщений в сетях AFDX

В литературе известно несколько методов вычисления длительности передачи сообщения как для потоков данных в распределенных системах, так и, в частности, для виртуальных каналов в сетях AFDX. Следует отметить, что задача оценки длительности передачи сообщений для сетей AFDX сводится к задаче оценки длительности передачи кадров. Так как сообщение делится на кадры, которые последовательно выдаются в сеть и следуют по одинаковому маршруту, то длительность доставки сообщения от отправителя до всех получателей вычисляется как разница между временем доставки последнего кадра до всех получателей и временем поступления сообщения от абонента [45]. Таким образом, можно разбить вычисление длительности передачи сообщения на две величины:

1. Задержка между поступлением сообщения от абонента и поступлением последнего кадра в очередь выдачи кадров в канал передачи данных.

2. Длительность передачи кадра от оконечной системы-отправителя до получения кадра всеми получателями.

В литературе основное внимание уделяется второй задаче, а именно, вычислению оценки длительности передачи кадров. Решение первой задачи описано, например, в работе [39], а также в разделе 4 данной работы.

Стоит отметить, что сети AFDX обладают некоторыми особенностями, усложняющими решение задачи оценки времени доставки кадров [46]. Первой особенностью является учет максимального джиттера виртуального канала. При отсутствии джиттера кадры виртуального канала предаются строго регулярно, при пиковая пропускная способность совпадает со средней пропускной способностью виртуального канала. При возникновении джиттера регулярность, вообще говоря, пропадает, и пиковая пропускная способность виртуальных каналов становится выше средней выделенной пропускной способности.

Другой особенностью является сериализация виртуальных каналов. Сериализация заключается в том, что на коммутатор из одного канала передачи данных не могут одновременно приходить кадры разных виртуальных каналов. Кадры могут приходить только последовательно, один за другим. При отсутствии учета сериализации оценка длительности получается сильно завышенной.

В работе [46] представлен обзор методов оценки длительности передачи кадров, используемых в настоящее время. Наиболее эффективными считаются подходы на основе Network Calculus [42, 47, 48] и Trajectory Approach [49, 50] благодаря учету описанных выше особенностей сетей AFDX. Согласно проведенным исследованиям, описанным в [46], Trajectory Approach позволяет получить более точные оценки, чем Network Calculus.

Наличие известных подходов к оценке длительности передачи кадров позволяет использовать один из готовых методов.

2.6. Выводы

Согласно проведенным исследованиям методов, полностью решающих поставленные задачи, выявлено не было. Однако для следующих подзадач можно использовать подходы из других работ:

- упаковка в контейнеры и поиск маршрутов виртуальных каналов для задачи планирования вычислений в ЦОД;
- настройка параметров *BAG, LM* виртуальных каналов, поиска маршрутов виртуальных каналов, вычисление длительности передачи кадров для задачи построения сетей AFDX.

Также следует упомянуть, что некоторые подзадачи (такие как упаковка в контейнеры и построение всех маршрутов виртуальных каналов) являются NP-трудными, что обуславливает необходимость применения эвристических алгоритмов.

При решении задач упаковки в контейнеры и построения маршрутов с учетом требований к качеству обслуживания известными методами может возникать фрагментация физических ресурсов, не позволяющая построить оптимальное назначение. Для устранения фрагментации в данной работе предлагается использоваться процедуру ограниченного перебора. Данная процедура заключается в выполнении перебора для подмножеств контейнеров/виртуальных каналов заданной мощности с целью нахождения более приемлемого по качеству решения. С помощью процедуры ограниченного перебора можно варьировать точность получаемых решений и время, затрачиваемое на их поиск.

3. Алгоритм планирования вычислений в ЦОД с моделью обслуживания IaaS

3.1. Общая схема алгоритма

Алгоритм планирования вычислений в ЦОД с моделью обслуживания IaaS с раздельными планировщиками состоит из трех шагов.

Ш а г 1. Назначение виртуальных машин на вычислительные узлы.

Ш а г 2. Назначение storage-элементов на физические системы хранения данных.

Ш а г 3. Для полученных назначений виртуальных машин и storage-элементов на физические ресурсы построение маршрутов виртуальных каналов в коммутационной сети обмена данными.

В последующих разделах приведено описание используемых процедур и их свойств.

3.2. Назначение виртуальных машин и storage-элементов

При назначении виртуальных машин и storage-элементов производится решение задачи упаковки в контейнеры (см. раздел 2.2.). Предлагаемый алгоритм основан на сочетании жадных стратегий и стратегий ограниченного перебора. Алгоритм осуществляет назначение элементов запросов по жадной схеме и в случае невозможности назначения очередного элемента вызывает процедуру ограниченного перебора. Для данной процедуры задается параметр, ограничивающий глубину перебора. Это позволяет выбирать требуемый баланс между вычислительной сложностью и точностью алгоритма.

Общая схема процедуры назначения элементов запроса (виртуальных машин или storage-элементов) выглядит следующим образом.

Ш а г 1. Из множества ресурсных запросов $\{G_i\}$ выбрать очередной запрос G_i в соответствии с жадным критерием K_G.

Ш а г 2. Выбрать очередной элемент e (виртуальную машину $e \in W, W \in G_i$ или storage-элемент $e \in S, S \in G_i$) в соответствии с жадным критерием K_e.

Ш а г 3. Сформировать множество физических ресурсов Ph ($Ph \subseteq P$ или $Ph \subseteq M$ соответственно), на которые может быть назначен выбранный элемент e, т.е. для которого выполнены отношения корректности отображения в случае назначения запроса e на физический ресурс.

Ш а г 3.1. Если множество Ph пусто, то вызвать процедуру ограниченного перебора. Если процедура возвращает неуспех, то запрос G_i не может быть назначен: удалить ранее

назначенные элементы запроса G_i и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.2.4). Если множество $\{G_i\}$ не пусто, перейти на шаг 1; в противном случае завершить работу процедуры.

Ш а г 3.2. Если множество Ph не пусто, то выбрать физический элемент $ph \in Ph$ для назначения в соответствии с жадным критерием K_{ph} и назначить элемент запроса e на физический элемент ph, и переопределить значения его характеристик в соответствии с функциями (1.2.4). Перейти на шаг 2, если есть нерассмотренные элементы запроса. Иначе перейти на шаг 1, если множество $\{G_i\}$ не пусто; в противном случае завершить работу процедуры.

3.2.1. Процедура ограниченного перебора

Процедура ограниченного перебора вызывается в случае, когда очередной элемент запроса e ($e \in W$ или $e \in S$) не может быть назначен ни на один физический элемент ph ($ph \in P$ или $ph \in M$ соответственно). Принцип работы процедуры ограниченного перебора показан на рисунках 3.1, 3.2. На рисунке 3.1 изображена ситуация, в которой виртуальный элемент с емкостью 0.4 не может быть назначен ни на один из физических элементов, несмотря на то, что суммарной свободной емкости достаточно для его назначения. При переназначении элементов таким образом, как изображено на рисунке 3.2, фрагментация устраняется и виртуальный элемент может быть назначен.

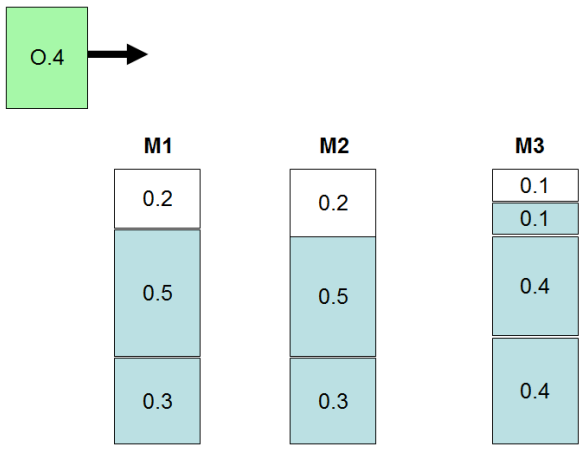

Рисунок 3.1. Пример выполнения процедуры ограниченного перебора.

Ограниченный перебор ограничивается заданным числом m, которое указывает максимальное количество физических узлов, среди которых можно производить переназначение (т.е. при m=2 назначенные элементы могут сниматься и переназначаться одновременно не более чем с двух физических элементов).

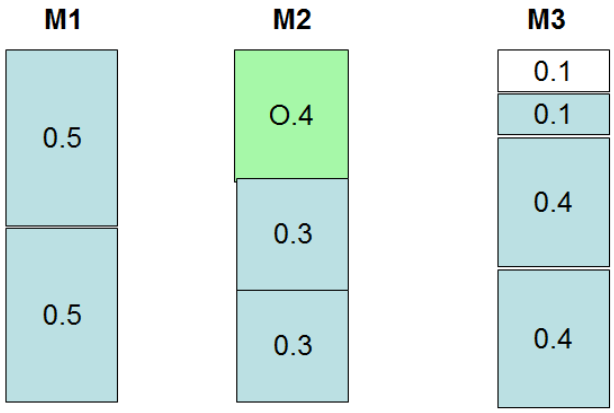

Рисунок 3.2. Переназначение элементов из рисунка 3.1 для назначения элемента.

Общая схема процедуры ограниченного перебора, таким образом, следующая.

Ш а г 1. Для заданного m сформировать все множества $A_m \subseteq A$ из не более чем m физических узлов и перебирать в соответствии с уменьшением суммарного значения, вычисленного согласно жадному критерию K_G.

Ш а г 1.1. Если для некоторого множества A_m суммарная остаточная емкость ресурсов достаточна для назначения текущего элемента, то выполнить шаги 1.1.1 – 1.1.3. Иначе перейти на шаг 2.

Ш а г 1.1.1. Произвести снятие элементов, которые назначены на физические ресурсы из множества A_m.

Ш а г 1.1.2. Произвести попытку переназначения одним из указанных способов:

1) назначить сначала элемент e, согласно жадным стратегиям, на один из физических узлов множества A_m, после чего назначить все виртуальные элементы, снятые с физических элементов множества A_m, используя жадные стратегии;
2) сформировать множество, состоящее из элемента e и из всех виртуальных элементов, снятых с физических элементов множества A_m; произвести назначение сформированного множества согласно жадным стратегиям (см. шаги 2 и 3 общей схемы назначения элементов запроса, без процедуры ограниченного перебора);
3) сформировать множество, состоящее из элемента e и из всех виртуальных элементов, снятых с физических элементов множества A_m; произвести полный перебор возможных назначений элементов из сформированного множества.

Ш а г 1.1.3. В случае неуспешного переназначения вернуть исходные назначения элементов.

Ш а г 2. Если после перебора всех указанных множеств элемент *e* не может быть назначен, вернуть неуспех.

3.2.2. Жадные критерии

Качество работы алгоритма существенно зависит от используемых жадных критериев. Критерии задаются для шага 1 общей схемы процедуры назначения элементов запроса, когда выбирается запрос для назначения (K_G), на 2 шаге, – когда выбирается элемент запроса (K_e), и на шаге 3.2, – когда выбирается физический ресурс, на который назначается элемент (K_{ph}).

Предложенные критерии основаны на функции стоимости *r(e)* назначения элемента *e*. В работе предлагается вычислять стоимость назначения элемента двумя способами:

1) выбор самой загруженной в момент назначения характеристики ресурса и вычисление стоимости по выбранной характеристике;
2) вычисление взвешенной суммы характеристик ресурса согласно их загруженности; идея данного подхода описана в работе [32].

Пусть элемент характеризуется вектором значений требуемых характеристик ресурсов *($r_{e,1}$, $r_{e,2}$, … $r_{e,n}$)*, для которых недопустима перегрузка[8]. Для перового из описанных методов перед каждым назначением определяется самая загруженная характеристика:

$$max\_resource = \arg\max_i \left(\frac{\sum_{G_a} \sum_{e \in E, E \in G_a} r_{e,i}}{\sum_{ph \in Ph} r_{ph,i}}\right).$$

В данной формуле суммирование в числителе производится по всем назначенным запросам G_a. Загрузка ресурса, таким образом, вычисляется как частное между суммарной емкостью назначенных элементов и общей емкостью физических элементов по данной характеристике.

Функция стоимости элемента по самой загруженной характеристике рассчитывается как емкость выбранной характеристики *max_resource*: $r(e) = r_{e,max\_resource}$.

Для определения взвешенной стоимости элемента для каждого ресурса вычисляется дефицит:

---

[8] В данном случае рассматриваются только характеристики, для которых проверяется отношение (1.2.1).

$$d(i) = \frac{\sum_{G} \sum_{e \in E, E \in G} r_{e,i}}{\sum_{ph \in Ph} r_{ph,i}}.$$

Данная величина определяется как частное, в котором числителем является общая требуемая емкость по данной характеристике для всех запросов *G*, знаменатель – общее значение имеющихся физических ресурсов. Чем больше дефицит, тем больше требуется ресурсов данного типа.

Функция стоимости вычисляется следующим образом[9]:

$$r(e) = \sum_{i=1..n} d(i)\, r_{e,i}$$

и равна взвешенной сумме требуемых ресурсов с учетом их дефицита.

Выделяется два варианта критериев: компактное назначение и сбалансированное назначение.

Компактное назначение используется, когда физические ресурсы сети не являются критическим ресурсом, т.е. в случаях, когда отношение стоимостей требуемых сетевых ресурсов к имеющимся сетевым ресурсам меньше заданного порога. В случае компактного назначения предлагаются следующие критерии:

1) K_G: выбирается запрос G с наибольшей суммарной требуемой стоимостью элементов;
2) K_e: среди элементов запроса G выбирается тот элемент e, который имеет наибольшую требуемую стоимость $r_v(e)$;
3) K_{ph}: среди множества физических элементов $ph \in Ph$, которые имеют достаточно ресурсов для назначения выбранного элемента e, выбирается тот, который имеет наименьшую стоимость (вычисляется аналогично стоимости виртуального элемента).

Описанные жадные критерии позволяют производить назначение по стратегии BF (best-fit, [37]). При назначении элементов каждого запроса используется стратегия BFD (best-fit decreased), однако в общем случае виртуальные элементы выбираются не строго в порядке убывания емкости, поэтому используемая стратегия эквивалентна BF. В случае одномерной упаковки данная стратегия достигает асимптотической точности назначения, равной 1.7. Это означает, что в наихудшем случае данный алгоритм назначает элементы в количество физических ресурсов, в 1.7 раз большее, чем оптимальный алгоритм.

[9] В данной формуле значение емкости ресурсов считается нормированным.

Сбалансированное назначение предлагается использовать в случае, когда сетевые ресурсы являются критическими, т.е. когда отношение стоимости требуемых сетевых ресурсов к стоимости имеющихся сетевых ресурсов превосходит некоторый порог.

При сбалансированном назначении первые два критерия (K_G и K_e) совпадают с компактным назначением. Для критерия K_{ph} предлагается среди множества элементов $ph \in Ph$, имеющих достаточно емкости для назначения выбранного элемента *e*, выбрать тот, который имеет *наибольшую* стоимость. Данный подход распределяет виртуальные ресурсы среди всех имеющихся физических элементов, что предоставляет больше физических ресурсов сети при построении маршрутов виртуальных каналов.

3.3. Построение маршрутов виртуальных каналов

Построение маршрутов виртуальных каналов производится после того, как получены назначения виртуальных машин и storage-элементов. При построении могут использоваться алгоритмы Беллмана-Форда, Дейкстры, а также нахождения *k* кратчайших путей в графе. В данной работе предлагается алгоритм, основанный на алгоритме Йена решении задачи нахождения *k* кратчайших путей в графе [51]. В случае неудачи назначения некоторого виртуального канала $e \in E$ вызывается процедура ограниченного перебора, которая заключается в переборе множеств ограниченной мощности из виртуальных каналов с построенным маршрутами с целью их изменения таким образом, чтобы мог быть построен маршрут и для данного виртуального канала. Общая схема процедуры ограниченного перебора виртуальных каналов схожа со схемой ограниченного перебора виртуальных машин и storage-элементов. В случае неуспеха вызывается процедура репликации, которая создает реплику для storage-элемента с целью устранения перегрузки входящих в физическую систему хранения данных каналов обмена. При создании реплики необходимо также проложить канал для поддержания консистентности между storage-элементом и его репликой.

Следует отметить, что процедура репликации в данном алгоритме используется с целью оптимизации назначения виртуальных каналов, т.е. для создания дополнительных возможных путей при назначении запросов.

3.3.1. Общая схема процедуры построения маршрутов виртуальных каналов

Общая схема алгоритма построения маршрутов виртуальных каналов следующая:

Ш а г 1. Выбрать запрос $G \in G_A$ для назначения виртуальных каналов согласно заданному критерию K_G, где G_A - множество запросов, для которых удалось произвести назначение виртуальных машин и storage-элементов.

Ш а г 2. Выбрать виртуальный канал $e \in E$ запроса G согласно заданному критерию K_e.

Ш а г 3. Произвести построение маршрута виртуального канала e с помощью алгоритма нахождения k кратчайших путей и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.2.4).

Ш а г 4. В случае неуспеха произвести процедуру ограниченного перебора.

Ш а г 5. При неуспешном выполнении процедуры ограниченного перебора в случае, когда одним из элементов, которые соединяет виртуальный канал $e = (w, s)$, является storage-элемент $s \in S$, выполнить процедуру репликации. Иначе перейти на шаг 6.

Ш а г 6. В случае неуспеха запрос G не может быть назначен, снять все виртуальные каналы, виртуальные машины и storage-элементы запроса G и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.2.4).

3.3.2. Жадные критерии

На первых двух шагах предлагается выбирать критерии, аналогичные выбранным при назначении виртуальных машин и storage-элементов. Таким образом, предлагается выбирать запрос с наибольшей суммарной стоимостью виртуальных каналов (K_G), после чего последовательно выбирать виртуальный канал с наибольшей стоимостью (K_e). Стоимость определяется аналогично стоимости для вычислительных ресурсов/виртуальных машин и систем хранения данных/storage-элементов. Обычно для каналов передачи данных и сетевых элементов используется только одна характеристика: пропускная способность. Стоимость в данном случае вычисляется как значение данной характеристики.

3.3.3. Построение маршрута для одного виртуального канала

Назначение одного виртуального канала производится с использованием алгоритма Йена [51] поиска k кратчайших путей. Общая схема алгоритма следующая.

Ш а г 1. Для заданного m произвести поиск не более чем m кратчайших путей $p \in P$ минимальной длины (под длиной понимается количество сетевых элементов $k \in K$, входящих в путь $p \in P$). Производится поиск только тех путей, каждый элемент которых

имеет достаточно емкости для назначения данного виртуального канала. Если ни одного пути не найдено, то возвращается неуспех.

Ш а г 2. Среди найденных путей выбрать путь $p \in P$ с наибольшей остаточной суммарной стоимостью среди каналов передачи данных и сетевых элементов.

Описанная процедура позволяет производить поиск с использованием двух критериев: минимальная длина и максимальная остаточная стоимость. Минимальная длина является первичным признаком, так как топологии ЦОД (таких, как «fat tree» [53]) имеют древовидную структуру (с добавлением избыточных ребер), что позволяет выбирать из множества путей одинаковой длины для каждой пары элементов.

Следует отметить, что вместо алгоритма *k* кратчайших путей можно использовать, например, алгоритм Дейкстра, используя в качестве критерия либо длину пути, либо максимальную остаточную пропускную способность. Для последнего варианта при выборе очередной вершины весом можно считать отношение суммарной зарезервированной пропускной способности к общей пропускной способности ребра, ведущего к вершине.

3.3.4. Процедура ограниченного перебора

Процедура ограниченного перебора заключается в выполнении следующих шагов.

Ш а г 1. Для заданного *m* перебирать все множества $A_e \subseteq A$ из не более чем *m* назначенных виртуальных каналов в порядке убывания суммарной стоимости виртуальных каналов. Для каждого множества A_e выполнить шаги 1.1. – 1.3.

Ш а г 1.1. Произвести снятие маршрутов выбранных виртуальных каналов $a \in A_e$.

Ш а г 1.2. Произвести попытку построения маршрута для виртуального канала *e* с помощью алгоритма *k* кратчайших путей; если назначение не удается, то вернуть назначение маршрутов снятых виртуальных каналов $a \in A_e$ и перейти к следующему множеству $A_e \subseteq A$.

Ш а г 1.3. Произвести попытку построения маршрутов для всех снятых виртуальных каналов $a \in A_e$; в случае успешного построения вернуть успех, иначе снять виртуальный канал *e* и вернуть исходные маршруты.

Ш а г 2. Если после перебора всех указанных множеств $A_e \subseteq A$ виртуальный канал *e* не может быть назначен, вернуть неуспех.

3.3.5. Процедура репликации

Процедура репликации возможна в случае, если одним из элементов, которые соединяет виртуальный канал $e=(w,s)$, является storage-элемент $s \in S$, для которого доступна репликация. Репликация считается доступной для storage-элементов с большим количеством чтений и малым количеством записей – для таких storage-элементов не требуется широкого канала для поддержания консистентности данных.

Процедура заключается в поиске системы хранения данных $m \in M$, которое имеет достаточно ресурсов для назначения реплики (реплика требует столько же ресурсов, сколько и данный storage-элемент s), и суммарная длина путей всех виртуальных каналов $e=(w \in W, s)$ (включающих данный storage-элемент s) к данной реплике m минимальна. Кроме того, необходимо провести канал l для поддержания консистентности между репликой и исходным storage-элементом (требуемая пропускная способность канала l определяется типом storage-элемента s). Если проложить канал l для поддержания консистентности не удается, то рассматривается другая система хранения данных s в порядке убывания суммарной длины путей виртуальных каналов.

При успешном выполнении процедуры репликации последующее построение маршрутов виртуальных каналов, включающих данный storage-элемент s, можно производить к данной реплике m. Если не удается найти систему хранения данных $m \in M$ с достаточным количеством ресурсов либо не удается провести канал поддержания консистентности данных l, то процедура возвращает неуспех.

3.4. Свойства и теоретическое обоснование алгоритмов

3.4.1. Оценка сложности этапов алгоритма

Пусть N – число вычислительных узлов, S – число систем хранения данных, W – число сетевых элементов в сети обмена данными. Пусть RV – общее число виртуальных машин всех ресурсных запросов, RS – общее число storage-элементов, RVL - общее число виртуальных каналов. Обозначим через K_{nodes} коэффициент перебора для операции ограниченного перебора виртуальных машин и storage-элементов, K_{links} - коэффициент перебора в операции назначения виртуальных каналов, K_{path} - количество исследуемых кратчайших путей в процедуре нахождения k кратчайших путей.

1. Назначение виртуальных машин на вычислительные узлы, на данном этапе выполняются следующие действия.

1.1. Сортировка виртуальных машин согласно выбранному критерию; сложность данного шага $O(RV \log(RV))$.

1.2. Для каждой из виртуальных машин выбор вычислительного узла, на который производится назначение (если назначение возможно); выполняется в наихудшем случае за $O(RV\ N)$ операций.

1.3. Процедура ограниченного перебора. Количество перебираемых сочетаний вычисляется как $C_N^{K_{nodes}}$ (число сочетаний из N по K_{nodes}). На каждом сочетании происходит попытка переназначения, сложность которого зависит от выбранного способа ограниченного перебора:

1) при использовании жадного алгоритма в процедуре ограниченного перебора сложность переназначения составляет $O(RV(\log(RV) + N))$;
2) при использовании алгоритма полного перебора в процедуре ограниченного перебора сложность в наихудшем случае достигает $O(RV!N)$;

Сложность в наихудшем случае на данном этапе, таким образом, составляет $O((C_{RV}^{K_{nodes}} + 1)RV(\log(RV) + N))$ при использовании жадного алгоритма в процедуре ограниченного перебора и $O(RV(\log(RV) + N) + O(C_{RV}^{K_{nodes}} RV!N))$ – в процедуре ограниченного перебора алгоритма с полным перебором всех возможных отображений.

2. Назначение storage-элементов на системы хранения данных. Данный этап выполняется аналогично предыдущему и имеет такую же сложность, при этом вместо количества виртуальных машин в оценках используется количество storage-элементов.

3. Построение маршрутов виртуальных каналов. Данный этап заключается в следующей последовательности шагов.

3.1. Сортировка виртуальных каналов по выбранному критерию. Сложность данного шага $O(RVL \log(RVL))$.

3.2. Попытка построения маршрута виртуального канала с помощью процедуры построения k кратчайших путей. Данная процедура состоит в использовании алгоритма Дейкстры не более $K_{path}(W+2)$ раз и содержит не более $O(K_{path}(W+2)^2)$ операций (так как виртуальные машины и storage-элементы уже назначены, количество вершин в графе поиска равно $W+2$). Данный шаг, таким образом, оценивается сложностью $O(RVL\ K_{path}(W+2)^3)$.

3.3. Процедура ограниченного перебора глубины K_{links} для виртуальных каналов. Данная операция оценивается аналогично назначению виртуальных машин (отличие состоит в использовании процедуры k кратчайших путей при переназначении). Сложность данного шага, таким образом, равна $O(C_{RVL}^{K_{links}}(K_{links}+1)K_{path}(W+2)^3)$. Здесь $C_{RVL}^{K_{links}}$ - количество перебираемых сочетаний, $K_{links}+1$ - количество переназначаемых виртуальных каналов.

3.4. Процедура репликации. В наихудшем случае процедура репликации производится для каждого из storage-элементов. Сложность поиска системы хранения данных для репликации равна $O(S(N+W+1)^2)$ и заключается в поиске путей минимальной длины к каждому хранилищу данных для графа из $(N+W+1)$ элемента. Сложность поиска пути для канала поддержания консистентности равна $O((W+2)^2)$. Общая сложность, таким образом, составляет $O(RVL(S(N+W+1)^2+(W+2)^2))$.

3.4.2. Зависимость сложности алгоритма от глубины ограниченного перебора

При исследовании сложности алгоритма с нулевой глубиной ограниченного перебора (при значениях $K_{nodes}=0$ и $K_{links}=0$) можно прийти к следующим выводам:

1) сложность алгоритма имеет кубическую зависимость от W, квадратичную зависимость от N и линейную зависимость от S (при отключении операции репликации алгоритм также линейно зависит от N);
2) при увеличении значений RV, RS и RVL сложность алгоритма возрастает как $O(RV\log RV)$, $O(RS\log RS)$ и $O(RVL\log RVL)$ соответственно;
3) алгоритм линейно зависит от параметра K_{path}.

При увеличении значений глубины перебора K_{nodes} и K_{links} сложность алгоритма возрастает как $C_N^{K_{nodes}}(C_S^{K_{nodes}})$ и $C_{RVL}^{K_{links}}$ соответственно. Следует заметить, что при увеличении значения K_{nodes} увеличивается коэффициент в слагаемых $O(C_N^{K_{nodes}}RV(\log(RV)+N))$ и $O(C_S^{K_{nodes}}RS(\log(RS)+S))$, которые возрастают как $RV\log RV$ ($RS\log RS$ соответственно) в зависимости от RV (RS). Изменение сложности операции перебора в зависимости от значений K_{nodes} при использовании жадного алгоритма в процедуре ограниченного перебора показано в таблице 3.1. В случае полного перебора в процедуре ограниченного перебора сложность увеличивается, кроме того, в $(RV-1)!$

($(RS-1)!$ соответственно) раз. Изменение сложности для переназначения с полным перебором показано в таблице 3.2.

Таблица 3.1. Изменение сложности алгоритма в зависимости от глубины перебора при использовании жадного алгоритма в процедуре ограниченного перебора

Глубина перебора K_{nodes}	Коэффициенты увеличения сложности алгоритма в зависимости от	
	N	S
1	N	S
2	$N(N-1)/2$	$S(S-1)/2$
3	$N(N-1)(N-2)/6$	$S(S-1)(S-2)/6$
…	…	…
K_{nodes}	C_N^{Knodes}	C_S^{Knodes}

Таблица 3.2. Изменение сложности алгоритма в зависимости от глубины перебора при использовании полного перебора в процедуре ограниченного перебора

Глубина перебора K_{nodes}	Коэффициенты увеличения сложности алгоритма в зависимости от	
	$N\ u\ RV$	$S\ u\ RS$
1	$(RV-1)!N$	$(RS-1)!S$
2	$(RV-1)!N(N-1)/2$	$(RS-1)!S(S-1)/2$
3	$(RV-1)!N(N-1)(N-2)/6$	$(RS-1)!S(S-1)(S-2)/6$
…	…	…
K_{nodes}	$(RV-1)!C_N^{Knodes}$	$(RS-1)!C_S^{Knodes}$

В то же время при увеличении значения K_{links} меняется коэффициент в слагаемом $O(C_{RVL}^{K_{links}}(K_{links}+1)K_{path}(W+2)^3)$, который кубически зависит от W. Изменение сложности операции ограниченного перебора в зависимости от значений K_{links} приведено в таблице 3.3.

Таблица 3.3. Изменение сложности алгоритма в зависимости от глубины перебора K_{links}

Глубина перебора K_{links}	Коэффициент увеличения сложности алгоритма в зависимости от RVL
1	RVL
2	$RVL(RVL-1)/2$
3	$RVL(RVL-1)(RVL-2)/6$
…	…
N	$C_{RVL}^{K_{links}}$

3.4.3. Корректность алгоритма

Назначение виртуальных запросов в ЦОД является корректным, если по всем характеристикам всех виртуальных элементов и каналов выполнены соответствующие условия (1.2.1) – (1.2.3).

Выполнение указанных условий при выполнении алгоритма поясняется следующим образом:

1. На этапе назначения виртуальных машин/storage-элементов производится назначение виртуального элемента на физический узел только в случае выполнения соответствующих ограничений (1.2.1) – (1.2.3)., иначе вызывается процедура ограниченного перебора.
2. Процедура ограниченного перебора в случае успеха переназначает некоторое множество элементов вместе с назначением текущего элемента таким образом, чтобы все соответствующие ограничения (1.2.1) – (1.2.3) были выполнены для всех переназначаемых элементов и для назначенного текущего элемента.
3. При построении маршрутов виртуальных каналов производится поиск среди только тех путей, каналы передачи данных и коммутаторы которых имеют достаточно пропускной способности. Таким образом, при успешном нахождении маршрута должны выполняться соответствующие ограничения (1.2.1) – (1.2.3).
4. При успешном выполнении процедуры ограниченного перебора при построении маршрута виртуального канала производится построение новых маршрутов для некоторого множества виртуальных каналов таким образом, чтобы выполнялись все ограничения (1.2.1) – (1.2.3) для всех переназначаемых маршрутов и для текущего назначенного маршрута.

5. При успешном выполнении процедуры репликации выполнены следующие условия:
   - Назначается реплика storage-элемента на некоторую систему хранения данных только при выполнения ограничений (1.2.1) – (1.2.3) для всех характеристик.
   - Для виртуального канала поддержания консистентности построен маршрут, при этом каналы передачи данных имеют достаточно емкости для его назначения.
6. При неуспешном назначении некоторого виртуального элемента или неуспешном построении маршрута виртуального канала для некоторого виртуального запроса производится снятие всего запроса.

Таким образом, для всех характеристик всех виртуальных элементов и каналов всех назначенных виртуальных запросов после работы алгоритма выполнены ограничения (1.2.1) – (1.2.3), что подтверждает корректность его работы.

3.4.4. Свойства процедуры ограниченного перебора при решении задачи упаковки в контейнеры

Пусть производится назначение виртуального элемента v с набором требуемых ресурсов $fv(v) = (v_1, v_2, \ldots v_n)$, для которого не хватает свободных ресурсов ни в одном из физических элементов. Процедура ограниченного перебора с глубиной m заключается в переборе множества $A_m=\{p\}$ из не более чем m физических элементов (каждый элемент $p \in P$ характеризуется вектором свободных ресурсов $ph(p) = (p_1, p_2, \ldots, p_n)$) с попыткой переназначения всех виртуальных элементов, назначенных на эти выбранные физические элементы. Переназначение производится таким образом, чтобы можно было назначить эти виртуальные элементы вместе с элементом v.

Описанная процедура обладает следующим свойством:

Свойство 3.1. Если $\exists i : \sum_{p \in P} p_i < v_i$ (то есть сумма всех свободных ресурсов по некоторой характеристике недостаточна для назначения виртуального элемента), то переназначение вернет неуспех.

Свойство 3.2. Если для некоторого виртуального элемента e, назначенного на элемент из множества A_m, верно: $\forall i : \sum_{i=1..n} e_i \geq v_i$ (то есть элемент e «крупнее» v по всем характеристикам), то переназначение элемента e возможно только внутри множества A_m.

Доказательство. Так как v не может быть назначен ни на один физический элемент из множества P, то e не может быть назначен ни на один элемент из множества $P / \{A_m\}$, так как изначально виртуальные элементы снимаются только с физических элементов из множества A_m.

Пример. На рисунках 3.1, 3.2 показан случай, когда переназначение элементов *e* с характеристиками 0.5 возможно только внутри множества A_m, состоящего из первых двух физических элементов.

Следствие 3.1. Если $\forall e \in V(A_m) : (\forall i : \sum_{i=1..n} e_i \geq v_i)$, где V($A_m$) – множество всех назначенных виртуальных элементов на элементы множества A_m), и, кроме того, $\exists i : \sum_{p \in A_m} p_i < v_i$, то переназначение для данного множества A_m неуспешно.

Данное следствие означает, что если все переназначаемые элементы по всем характеристикам требуют не меньше ресурсов, чем элемент *v*, и кроме того, по некоторой характеристике в данном множестве физических элементов не хватает ресурсов для назначения *v*, то переназначение вернет неуспех.

Данное свойство следует из того, что указанные виртуальные элементы могут быть переназначены только внутри множества A_m, при этом по некоторой характеристике элемент *v* не может поместиться ни в один элемент указанного множества.

Использование описанных свойств позволяет производить отсечение перебираемых множеств физических элементов. Для ускорения ограниченного перебора следует выполнить следующие проверки:

1. Проверить, согласно свойству 3.1, что суммарной свободной емкости по всем характеристикам достаточно для назначения элемента.
2. Проверить выполнимость следствия из свойства 3.2.
3. В процессе перебора пытаться переназначить виртуальные элементы, описанные в свойстве 3.2, только внутри множества A_m.

Напомним, что асимптотическая точность процедуры для задачи упаковки в контейнеры определяется как отношение в наихудшем случае количества контейнеров, назначаемое процедурой, к количеству контейнеров, получаемому оптимальным алгоритмом, при стремлении количества элементов к бесконечности.

Утверждение 3.1 (асимптотическая точность процедуры ограниченного перебора для задачи одномерной упаковки). При одномерной упаковки в контейнеры для любого *k* для асимптотической точности *a* жадного алгоритма упаковки с процедурой ограниченного перебора глубины *k* верно:

$$11/9 \leq a \leq 1{,}7 .$$

При этом при k=1 верно: a=1,7. При $k = N$ (где N – количество контейнеров) верно: $a = 11/9$.

Доказательство. При k=1 процедура ограниченного перебора не улучшает результатов работы жадного алгоритма, так как производится перебор элементов внутри одного контейнера (что, очевидно, бессмысленно). Жадный алгоритм при этом работает по схеме Best-Fit, так как элементы помещаются в наилучший подходящий контейнер, при этом сначала выбирается очередной запрос, потом назначаются все его элементы, при этом элементы разных запросов рассматриваются необязательно в убывающем порядке.

При увеличении k точность алгоритма, очевидно, не уменьшается (так как все уже назначенные виртуальные элементы не снимаются).

При назначении $k=N$ во время выполнения процедуры ограниченного перебора снимаются все элементы, после чего они назначаются согласно стратегии Best-Fit-Decreasing, которая имеет асимптотическую точность 11/9.

3.4.5. Свойства процедуры ограниченного перебора при построении маршрутов виртуальных каналов

Пусть производится назначение виртуального канала vl с требуемой пропускной способностью $r(vl)$ (предполагается одномерный случай, с учетом только пропускной способностью каналов передачи данных и виртуальных каналов). Виртуальный канал следует из некоторого элемента e_1 в некоторый элемент e_2.

Пусть попытка провести маршрут с достаточной пропускной способностью от e_1 к e_2 оканчивается неудачей. Обозначим через G^* граф доступных ресурсов, который получается из исходного графа физических ресурсов G следующим образом:

1. $G^* = G$.
2. Из всех значений доступной пропускной способности каналов передачи данных (и коммутаторов, в случае учета пропускной способности на них) вычитается значение $r(vl)$.
3. Те каналы передачи данных (и коммутаторы, вместе со смежными каналами передачи данных), значение полученной пропускной способности которых меньше 0, удаляются из графа G^*.

Так как виртуальный канал не может быть назначен, то элементы e_1 и e_2 находятся в разных компонентах связности графа.

Пусть G' – граф *недоступных* ресурсов, получаемый при вычете G из G^*. Пусть также H^* – граф, включающий в себя все возможных маршруты без циклов (без учета требований пропускной способности) от e_1 к e_2. Такой граф можно построить из графа G за $O((|K| + |E|)^2)$ (K – множество коммутационных элементов, E – множество ребер) с помощью обхода в ширину с запоминанием пройденных вершин и ребер.

Обозначим через *H* пересечение графов *G'* и *H**. В данном графе присутствуют те элементы, через которые *vl* не может быть проложен и которые могут участвовать в прокладке путей.

Утверждение 3.2. Пусть производится переназначение множества виртуальных каналов *V* = {v} и для всех элементов *h* из множества *H* верно: $\sum_{v \in A(h), v \in V} r(h) + r(v) < r(vl)$, где суммирование ведется по всем виртуальным элементам, назначенным на элемент *h*. Тогда переназначение множества *V* вместе с элементом *vl* приведет к неуспеху.

Доказательство.

Так как при снятии элементов ни у одного элемента из *H* не будет достаточно ресурсов для назначения *vl*, то назначить *vl* не удастся.

Следствие 3.2. Среди переназначаемых виртуальных каналов должен быть хотя бы один виртуальный канал, проходящий через множество *H*.

Таким образом, при проведении процедуры ограниченного перебора виртуальных каналов следует перебирать только множества виртуальных каналов, удовлетворяющие условиям утверждения 3.2 и следствия 3.2.

3.4.6. Достаточные условия оптимальности алгоритма планирования вычислений в ЦОД

В данном разделе приведены достаточные условия, при которых алгоритмы работают оптимально. Оптимальность в данном случае подразумевает назначение всех виртуальных запросов в ЦОД.

Пусть *N* – количество физических контейнеров, каждый из которых описывается n характеристиками: $r(e_i) = (r_{i,1}, r_{i,2}, \ldots r_{i,n})$, $i = 1..N$. Пусть производится назначение *M* виртуальных элементов, каждый из которых также описывается *n* характеристиками: $v(v_i) = (v_{i.1}, v_{i.2}, \ldots v_{i.n})$, $i = 1..M$, причем для каждой характеристики известны ограничения сверху: $\forall j : v_{i,j} \le vm_j$.

Утверждение 3.3. (достаточное условие решения задачи упаковки в контейнеры). Пусть значение каждой характеристики ограничено снизу: $\forall j : r_{i,j} \ge rm_j$. Если верно $M \le N \cdot \min_{j \le n} \left\lfloor \frac{rm_j}{vm_j} \right\rfloor$, то алгоритм назначит все виртуальные элементы.

Доказательство. Обозначим $K_j = \left\lfloor \frac{rm_j}{vm_j} \right\rfloor$ - данная характеристика показывает, какое минимальное количество элементов **по данному ресурсу *j*** вместится в любой контейнер.

Тогда минимальное количество элементов, помещающееся в любой контейнер, равно $\min(K_j)$. Учитывая, что число контейнеров N, получаем, что минимальное количество элементов, помещающееся во все контейнеры (причем независимо от используемого жадного алгоритма) равно $N \cdot \min(K_j) = N \cdot \min_{j \le n} \left\lfloor \frac{rm_j}{vm_j} \right\rfloor$.

Замечание 3.1. Несмотря на сильные ограничения, при которых выполняется утверждение, на практике в ЦОД физические контейнеры (а именно, вычислительные узлы и хранилища данных) имеют одинаковые характеристики (используется одинаковое оборудование). Кроме того, предоставляемые виртуальные элементы также могут иметь одинаковые характеристики (либо выделяются различные классы виртуальных элементов, например, виртуальные машины с низкими/высокими требованиями к вычислительным ресурсам/оперативной памяти/дисковой памяти), требуемые значения которых много меньше предоставляемых емкостей [52].

Утверждение 3.4. Пусть входом задачи является ЦОД с древовидной топологией, причем для вычислительных элементов, систем хранения данных и, соответственно, виртуальных машин и storage-элементов выполнены условия утверждения 3.3. Пусть, кроме того, выполнены следующие условия:

- для каждой характеристики физических элементов-контейнеров известны ограничения сверху: $\forall j : r_j \le rn_j$;
- для каждой характеристики виртуальных элементов известны ограничения снизу: $\forall j : v_j \ge vn_j$, причем $\exists j : vn_j > 0$;
- каждый виртуальный элемент присутствует не более чем в k виртуальных каналов;
- каждый виртуальный канал требует пропускную способность, не превышающую RM.

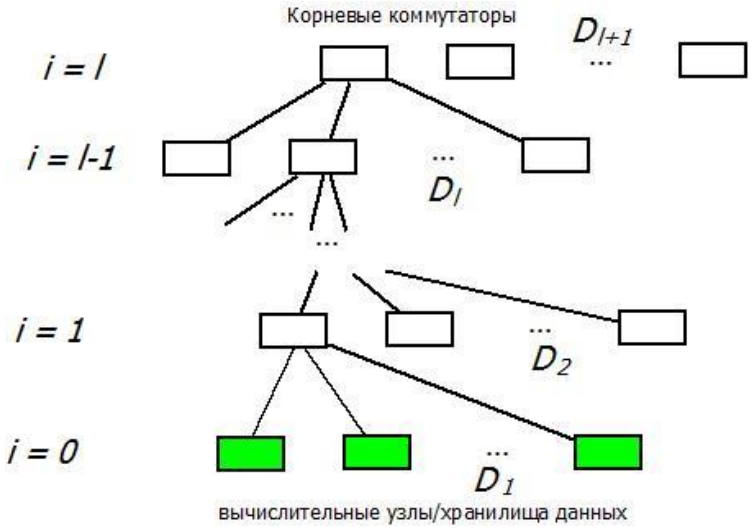

Рисунок 3.3. Представление многоуровневой древовидной топологии.

- физическая топология ЦОД представляет собой древовидную структуру глубины l, обладающую следующими характеристиками (см. рисунок 3.3):
    - каждый канал передачи данных уровня $i \le l$ имеет пропускную способность не ниже R_i;
    - каждый коммутатор уровня $i \le l$ имеет пропускную способность не ниже K_i;
    - каждый уровень i разбивается на кластеры (при этом уровень 0 соответствует вычислительным узлам или системам хранения данных, на остальных уровнях находятся коммутаторы), в кластере находится не более D_{i+1} элементов. При этом коммутаторы уровня i имеют не менее D_i каналов передачи данных[10], идущих к уровню $i - 1$.

Тогда все запросы будут назначены при выполнении следующих условий:

- $R_1 \ge RM \cdot k \cdot \min\limits_{j \le n} \left\lfloor \dfrac{rn_j}{vn_j} \right\rfloor$;
- $K_i \ge D_i \cdot R_i$, $1 < i \le l$;
- $R_{i+1} = K_i$, $1 < i \le l$.

[10] Предполагается, что в «строго древовидной» топологии корневые коммутаторы имеют ровно D_{l+1} исходящих каналов передачи данных, остальные коммутаторы имеют ровно $D_i + 1$ (один из каналов идет на уровень выше). В топологии fat-tree могут добавляться избыточные каналы передачи данных.

Доказательство. При выполнении условий утверждения 3.3 все виртуальные машины и storage-элементы будут назначены. Максимальное количество элементов, которое будет назначено в каждый контейнер, равно $KM = \min_{j \le n} \left\lfloor \frac{rn_j}{vn_j} \right\rfloor$.

В каждом контейнере *e* помещается не более *KM* виртуальных элементов, каждый из которых участвует не более чем в *k* виртуальных каналов. Таким образом, максимальное количество виртуальных каналов, которое может исходить из *e*, равно $k \cdot KM$. При условии, что каждый виртуальный канал имеет пропускную способность, не превосходящую *RM*, достаточно выполнения условия $R_1 \ge RM \cdot k \cdot KM$ (здесь R_1 – пропускная способность каналов передачи данных нижнего уровня), для назначения всех виртуальных каналов на уровне 1.

Так как коммутатор нижнего уровня имеет пропускную способность не ниже K_1 и из него выходит как минимум D_1 каналов передачи данных нижнего уровня, то для назначения всех виртуальных каналов необходимо выполнение условия $K_1 \ge D_1 \cdot R_1$. Т.к. из коммутатора на верхний уровень может выходить только один канал передачи данных, то достаточно $R_2 = K_1 \ge D_1 \cdot R_1$.

Переходя от уровня к уровню в соответствии с описанной выше схемой, достаточно выполнения следующих условий:

- $R_1 \ge RM \cdot k \cdot KM$
- $K_i \ge D_i \cdot R_i$, $R_{i+1} = K_i$, $1 < i \le l$.

Замечание 3.2. В данном утверждении для упрощения предполагается, что все контейнеры имеют одинаковый набор характеристик. Без ограничения общности можно расширить данное утверждение, отдельно описывая ограничения для виртуальных машин и для storage-элементов.

Замечание 3.3. Несмотря на то, что описанные условия кажутся жесткими, можно выделить следующие моменты, которые позволяют использовать описанное утверждение на практике:

- Обычно используются вычислительные узлы и системы хранения данных с одинаковой фиксированной емкостью, при этом выполнены условия: $\forall j : rn_j = rm_j$. Например, в работе [52] описаны типичные характеристики вычислительных узлов и виртуальных машин, используемые в ЦОД фирмы Amazon.

- Для виртуальных машин и storage-элементов обычно выделяется наиболее приоритетная(-ые) характеристика(-и), значение которой ненулевое для всех виртуальных элементов. Например, для виртуальных машин – оперативная память и количество ядер [52], для storage-элементов – требуемый объем памяти. Поэтому условие $\exists j : vn_j > 0$ в большинстве случаев выполняется.
- Требуемая пропускная способность виртуальных каналов в ЦОД обычно на порядок меньше пропускной способности каналов передачи данных [53]. Например, могут предоставляться каналы передачи данных с пропускной способностью 1-10 Гбит/с, в то время как требуемая пропускная способность – 1-100 Мбит/с.
- Количество виртуальных каналов на каждый виртуальный элемент обычно невелико.

Описанные свойства позволяют использовать данное утверждение в реальных ЦОД.

Замечание 3.4. Существует возможность, зная характеристики предоставляемых виртуальных элементов и каналов, прогнозировать требуемые физические ресурсы с помощью указанного утверждения. Для этого достаточно выделить физические ресурсы, для которых будут выполнены указанные условия при наличии прогноза на количество и параметры виртуальных запросов.

3.5. Выводы

В данном разделе предложен новый алгоритм планирования вычислений в ЦОД, основанный на жадных стратегиях и стратегиях ограниченного перебора. Алгоритм позволяет задавать баланс между точностью и вычислительной сложностью путем ограничения количества физических ресурсов, которые участвуют в процедуре ограниченного перебора. Получена оценка вычислительной сложности алгоритма в зависимости от глубины предложенной процедуры ограниченного перебора, обоснована корректность и доказан ряд свойств процедуры ограниченного перебора. Получены оценки точности процедуры для ряда частных случаев. Также получены достаточные условия оптимальной работы алгоритмов, которые могут использованы на практике в реальных ЦОД в процессе их проектирования при ограничениях возможных значений входных данных. Такие ограничения характерны для реальных данных.

4. Алгоритм построения сетей AFDX

Как и алгоритм планирования вычислений в ЦОД с моделью обслуживания IaaS, алгоритм построения сетей AFDX основан на жадных стратегиях и стратегиях ограниченного перебора. Кроме процедуры ограниченного перебора, в алгоритме используется процедура агрегации, целью которой является построение виртуальных каналов, через которые передается более одного сообщения. Данная процедура выполняется в случае неуспешного назначения виртуальных каналов с меньшим числом сообщений.

4.1. Общая схема алгоритма

Общая схема алгоритма следующая:

Шаг 1. Для каждого сообщения выделить виртуальный канал и произвести настройку параметров *LM*, *BAG* и *JM* с учетом ограничения (1.3.2). На данном шаге используется метод, описанный в разделе 4.2.1.

Шаг 2. Для каждого виртуального канала проверить выполнение ограничения (1.3.3). Если ограничение выполняется не для всех виртуальных каналов, произвести процедуру агрегации, описанную в разделе 4.2.2. Входом процедуры являются виртуальные каналы, для которых не выполняется ограничение. Процедура объединяет сообщения, исходящие от одного абонента, в один виртуальный канал (с настройкой новых параметров *LM*, *BAG* и *JM*) таким образом, чтобы выполнялись ограничения (1.3.2) и (1.3.3). Прежние виртуальные каналы этих сообщений удаляются.

Шаг 3. Для каждого виртуального канала в порядке убывания требуемой пропускной способности[11] произвести построение маршрута с учетом ограничения (1.3.1):

Шаг 3.1. Выполнить процедуру построения маршрута, описанную в разделе 4.2.3. Если маршрут найден, перейти к шагу 4.

Шаг 3.2. Выполнить процедуру ограниченного перебора виртуальных каналов, описанную в разделе 4.2.4 и заключающуюся в проведении попыток поиска другого маршрута для уже назначенных виртуальных каналов.

- Если маршрут построен успешно, то перейти к следующему виртуальному каналу.

[11] В данном случае можно использовать другие критерии для определения порядка перебора виртуальных каналов. Например, производить поиск маршрута в порядке убывания критерия $d(vl) = |MSG_{vl}| \cdot bw(vl)$ - пропорционально количеству сообщений и пропускной способности.

- Если маршрут построить не удалось, то в случае, если в виртуальном канале передается только одно сообщение, его назначение считается неуспешным. Иначе из виртуального канала убирается сообщение с наибольшим значением *LM/BAG* и его назначение считается неуспешным. Виртуальный канал переконфигурируется с помощью процедуры агрегации, после чего для данного виртуального канала производится переход на шаг 3.1.

Шаг 4. Для каждого сообщения выполнить шаги 4.1 – 4.3.

Шаг 4.1. Произвести вычисление длительности и джиттера передачи сообщений (см. раздел 4.2.5) и проверить выполнение ограничения (1.3.4). Если ограничения выполняются, перейти к следующему сообщению.

Шаг 4.2. Произвести процедуру переконфигурации виртуального канала по которому передается сообщение (см. раздел 4.2.6), заключающуюся в попытке перенастроить параметры *LM, BAG* и *JM* с учетом оценок длительности передачи, полученных на шаге 4.1. Если после проведения процедуры ограничение (1.3.4) выполняется, перейти к следующему сообщению.

Шаг 4.3. Произвести процедуру агрегации виртуальных каналов, проводя для каждого агрегированного виртуального канала построение маршрута и проверку ограничений (1.3.1) - (1.3.4). Если построение маршрута успешно и все ограничения выполняются, то прежние виртуальные каналы заменяются на новый. Если после выполнения процедуры для исходного сообщения не выполняется ограничение (1.3.4), назначение сообщения считается неуспешным и сообщение удаляется из виртуального канала.

4.2. Описание процедур, используемых в алгоритме

4.2.1. Настройка параметров виртуальных каналов

Настройка параметров виртуальных каналов основана на методе, предложенном в работе [39]. В работе предлагается задавать параметры *LM* и *BAG* с помощью решения задачи минимизации пропускной способности виртуального канала. Для этого пропускная способность представляется как функция $bw_{vl}(n)$, зависящая от количества кадров n, на которое делится сообщение *msg*. В работе [39] задача ставится с ограничением на время передачи сообщения, которое задается в виде:

$$(n-1)BAG \le \delta \qquad (4.2.1),$$

здесь δ - ограничение на время выдачи последнего кадра в канал.

Так как время передачи сообщения – это время ожидания выдачи его последнего кадра δ и длительность передачи последнего кадра через сеть Δ, то верно: $\delta = \tau_{msg} - \Delta$.

Величина Δ может быть вычислена только после построения маршрутов виртуальных каналов, поэтому на данном этапе ее можно задать некоторым числом (например, 1 *мс*), а проверку выполнения ограничений (1.3.4) проводить после построения маршрутов.

Кроме ограничений на длительность передачи сообщения, должны быть также выполнены ограничения (4.2.2) и ограничения на максимальный джиттер J_{msg} порождения сообщения[12]. Для виртуального канала с одним сообщением ограничение (1.3.2) принимает вид:

$$n \cdot BAG \le T_{msg} \qquad (4.2.2).$$

Для формирования ограничения на джиттер порождения сообщения рассмотрим наихудшую ситуацию, при которой некоторое сообщение MSG_1 было выдано через $T_{msg} - J_{msg}$ после предыдущего сообщения MSG_0 (см. рисунок 4.1).

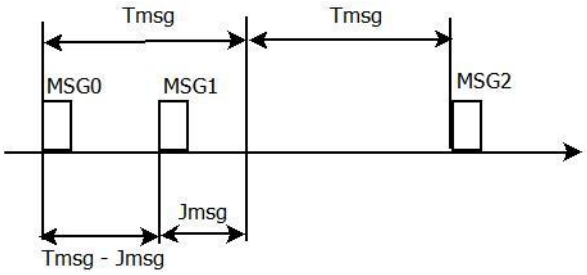

Рисунок 4.1. Возникновение джиттера при порождении сообщения.

Если $n \cdot BAG > T_{msg} - J_{msg}$, то к моменту выдачи MSG_1 не все кадры сообщения MSG_0 поступили в канал. В таком случае выдача последнего кадра сообщения MSG_1 превосходит величину $(n-1) \cdot BAG$ на величину, равную длительности передачи оставшихся кадров сообщения MSG_0, которую можно оценить как $n \cdot BAG - (T_{msg} - J_{msg})$. Исходя из описанных рассуждений, ограничение (4.2.1) принимает следующий вид:

$$\begin{cases} (n-1)BAG \le \delta, \text{ при } n \cdot BAG \le T_{msg} - J_{msg} \\ (2n-1) \cdot BAG - (T_{msg} - J_{msg}) \le \delta, \text{ иначе} \end{cases} \qquad (4.2.3)$$

Таким образом, *LM* и *BAG* формируются с помощью минимизации величины $bw(n)$, вычисляемой как $bw(n) = \frac{LM}{BAG}$, с учетом ограничений (4.2.2) и (4.2.3). Величины *LM* и *BAG* в данной формуле зависят от n и задаются следующим образом:

[12] Указанные ограничения не учитываются в работе [39].

- $LM(n) = \max(64, \left\lceil \frac{size_{msg}}{n} \right\rceil + c)$, так как размер кадра не ниже 64 байт при размере заголовка c,
- $LM(n) = \min(1518, \left\lceil \frac{size_{msg}}{n} \right\rceil + c)$, так как размер кадра не превосходит 1518 байт при размере заголовка c,
- $BAG(n) = 2^{k(n)}, k(n) = 0..7$, где $k(n)$ определяет значение степени двойки BAG.

Согласно исследованиям, проведенным в работе [39], значения функции $bw(n)$ в зависимости от n ведут себя согласно графику, изображенному на рисунке 4.2.

Значения n_i, изображенные на рисунке, вычисляются следующим образом:

$$\begin{cases} n_i = \min(\left\lfloor \frac{\delta}{2^i} \right\rfloor + 1, \left\lfloor \frac{T_{msg}}{2^i} \right\rfloor), \text{ при } n_i \cdot 2^i \le T_{msg} - J_{msg} \\ n_i = \min(\left\lfloor \frac{\delta + T_{msg} - J_{msg}}{2^{i+1}} + 1/2 \right\rfloor, \left\lfloor \frac{T_{msg}}{2^i} \right\rfloor), \text{ иначе.} \end{cases}$$

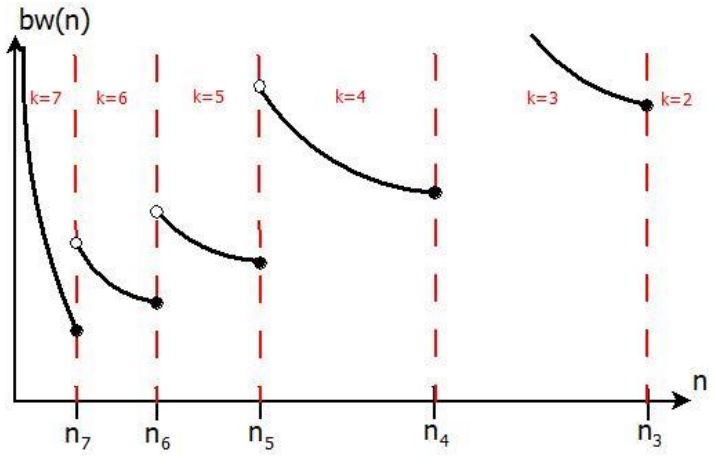

Рисунок 4.2. График зависимости функции $bw(n)$ от n.

n_i соответствует количеству кадров, при которых $k(n)$ принимает граничное значение, то есть при $n > n_i$ для $k=i$ не выполняется одно из ограничений (4.2.2), (4.2.3), причем при $n=n_i$ оба ограничения выполняются. С другой стороны, при фиксированном k график является убывающим.

Согласно данному графику, для выбора оптимального значения n достаточно взять наименьшее значение n_i, при которых выполняются ограничения (4.2.2) и (4.2.3) и ограничения на максимальный размер кадра LM, и вычислить соответствующим образом $LM(n)$ и $BAG(n)$. Полученные значения являются оптимальными с точки зрения минимизации пропускной способности виртуального канала.

После вычисления параметров LM_{vl} и BAG_{vl} для всех виртуальных каналов возможно вычисление максимальных джиттеров порождения кадров JM_{vl}. Данная величина равна времени ожидания отправки кадров со всех виртуальных каналов, исходящих из той же оконечной системы-отправителя *es*:

$$JM_{vl} = \sum_{v \in VL_{es}, v \neq vl} (\frac{LM_v}{R_e} + gap) \quad (4.2.4),$$

где VL_{es} – множество виртуальных каналов, исходящих из *es*; R_e – пропускная способность физического канала передачи данных, исходящего из *es*; *gap* – временной промежуток между кадрами.

4.2.2. Процедура агрегации

Процедура агрегации заключается в построении виртуального канала, по которому передаются несколько сообщений, исходящих от одного абонента. Процедура может вызываться в следующих случаях:

1. При невыполнении ограничения (1.3.3) для некоторого виртуального канала.
2. При невыполнении ограничения (1.3.4) для некоторого сообщения и неуспешной процедуре переконфигурации.

4.2.2.1. Общая схема процедуры агрегации

Пусть процедура проводится для виртуального канала *vl*, исходящего из оконечной системы *es*. Процедура агрегации выполняется по следующей схеме:

Шаг 1. Для каждого абонента *src*, подключенного к *es*, сформировать множество виртуальных каналов VL_{src}, состоящее из виртуальных каналов с сообщениями, исходящими из *src*.

Шаг 2. Если каждое из множеств VL_{src} содержит не более одного элемента или для всех пар виртуальных каналов попытка агрегации уже проведена, вернуть неуспех.

Шаг 3. Выбрать нерассмотренную пару виртуальных каналов vl_1 и vl_2 среди множеств VL_{src}, таких что $vl_1, vl_2 \in VL_{src}$ и значение $r(vl_1) \cdot r(vl_2)$ максимально среди всех подобных пар. *r(vl)* – критерий, который предлагается задавать одним из следующих способов:

- $r(vl) = bw(vl) / |MSG_{vl}|, bw(vl) = LM_{vl} / BAG_{vl}$ - критерий прямо пропорционален требуемой пропускной способности и обратно пропорционален количеству сообщений в виртуальном канале;
- $r(vl) = (\alpha \cdot bw(vl) + \beta \cdot \min_{msg \in MSG_{vl}} \tau_{msg}) / |MSG_{vl}|, \alpha + \beta = 1$ - кроме резервируемой пропускной способности в данном критерии добавляется требование длительности

доставки сообщения; так как агрегация увеличивает длительность доставки сообщения, целесообразно выбирать сообщения с большим значением требуемой длительности передачи.

Шаг 4. Провести агрегацию элементов vl_1 и vl_2 согласно алгоритму, описанному в разделе 4.2.2.2.

Шаг 5. Если требуемая пропускная способность агрегированного виртуального канала больше суммарной требуемой пропускной способности каналов vl_1 и vl_2, восстановить исходные виртуальные каналы[13] и перейти на шаг 2.

Шаг 7. Если хотя бы для одного виртуального канала vl_1 или vl_2 уже построен маршрут, провести попытку построения маршрута агрегированного виртуального канала. При неуспешном построении восстановить исходные виртуальные каналы и перейти на шаг 2.

Шаг 8. Убрать vl_1 и vl_2 из VL_{src} и вставить в VL_{src} полученный виртуальный канал. Если нарушаемые свойства для vl все еще не выполняются, перейти на шаг 2. Иначе завершить процедуру.

4.2.2.2. Агрегация выбранного множества сообщений

Агрегация двух виртуальных каналов, выполняемая на шаге 4 алгоритма из раздела 4.2.2.1, в целом схожа с методом настройки параметров виртуального канала, описанного в разделе 4.2.1. Отличие заключается в том, что в одном виртуальном канале формируются кадры нескольких сообщений. Пусть дано множество потоков сообщений $MSG_{src} = \{msg\}$, исходящих из одного абонента src, для которых требуется построить виртуальный канал. Максимальное количество кадров в очереди виртуального канала равно суммарному количеству кадров:

$$N = \sum_{msg \in MSG_{src}} \left\lceil size_{msg} / (LM_{msg} - c) \right\rceil .$$

Как и в случае с одним сообщением, ставится задача минимизации пропускной способности, которая задается функцией от N:

$$bw(N) = LM(N) / 2^{k(N)} .$$

[13] Требуемая пропускная способность агрегированного виртуального канала может быть как больше, так и меньше суммы пропускных способностей исходных виртуальных каналов. Алгоритм основан на жадном предположении, что при уменьшении пропускной способности достигнут положительный результат.

При этом должны быть выполнены ограничения, аналогичные (4.2.2) и (4.2.3). Отличием является то, что из ограничений T_{msg}, δ_{msg} выбирается наименьшее, так как эти ограничения должны выполняться для всех сообщений виртуального канала:

$$N \cdot BAG \le \min_{msg} T_{msg} \quad (4.2.5),$$

$$\begin{cases} (N-1)BAG \le \min\limits_{msg} \delta_{msg}, \text{ при } N \cdot BAG \le \min\limits_{msg} (T_{msg} - J_{msg}) \\ (2N-1) \cdot BAG \le \min\limits_{msg} (\delta_{msg} + T_{msg} - J_{msg}), \text{ иначе} \end{cases} \quad (4.2.6).$$

Кроме указанных ограничений, должны выполняться ограничения, предъявляемые стандартом Ethernet, т.к. кадры в сети AFDX формируются согласно этому стандарту. Размер кадра должен быть ограничен и равен 1518 байт, т.е. без учета заголовка полезный размер составляет 1518 – c байт. Т.к. каждый полезный размер кадра не превышает этого размера, для общего числа кадров верно следующее ограничение:

$$N \ge \sum_{msg} \left\lceil \frac{size_{msg}}{1518 - c} \right\rceil \quad (4.2.7).$$

Оптимальное значение N определяется аналогично случаю с одним сообщением, путем выбора наименьшего из значений N_i, для которых верны ограничения (4.2.5), (4.2.6) и (4.2.7):

$$\begin{cases} N_i = \min\left(\left\lfloor \frac{\min\limits_{msg} \delta_{msg}}{2^i} \right\rfloor + 1, \left\lfloor \frac{\min\limits_{msg} T_{msg}}{2^i} \right\rfloor \right), \text{ при } N_i \cdot 2^i \le \min\limits_{msg} (T_{msg} - J_{msg}) \\ N_i = \min\left(\left\lfloor \frac{\min\limits_{msg} (\delta + T_{msg} - J_{msg})}{2^{i+1}} + 1/2 \right\rfloor, \left\lfloor \frac{\min\limits_{msg} T_{msg}}{2^i} \right\rfloor \right), \text{ иначе.} \\ N_i \ge \sum\limits_{msg} \left\lceil \frac{size_{msg}}{1518 - c} \right\rceil \end{cases}$$

По выбранному значению N_i определяется значение $k(N_i) = i$. Основным вопросом является вычисление функции $LM(N)$, которая представляет собой максимальный размер кадра, при котором суммарное количество кадров равно N.

Следует отметить, что точного значения данная функция может и не иметь. Например, если имеется два сообщения одинакового размера и N=3, то невозможно определить максимальное значение кадра, так как оба сообщения разобьются на одинаковое количество кадров и, соответственно, количество кадров чётно.

Предлагаемым решением является такое значение $LM(N)$, при котором сообщения разделяются на максимальное количество кадров, не превышающее N. Так как конфигурация позволяет передавать N кадров с промежутком BAG между кадрами, то за

то же время конфигурация гарантирует передачу любого количества кадров, не превышающего *N* (при фиксированных *BAG* и *LM*). При таком вычислении *LM(N)* может резервироваться больше пропускной способности, чем необходимо виртуальному каналу, однако значение зарезервированной пропускной способности удовлетворяет заданным ограничениям.

Значение *LM*(*N*) предлагается вычислять итеративно, начиная со значения $LM(N_0) = \max_{msg}(size_{msg} + c), N_0 = |MSG_{src}|$. Данное значение определяет размер кадра при условии, что количество кадров равно количеству сообщений. В этом случае каждое сообщение должно попасть ровно в один кадр (ограничения на размер кадра в данном случае не учитываются) и размер кадра равен максимальному размеру сообщения.

На каждом последующем шаге производится увеличение общего количества кадров на 1 при условии, что каждое сообщение разбито на кадры одинакового размера. В реальности сообщения не всегда разбиваются на равные части, и общее количество кадров может быть меньше получаемого, поэтому данное условие является жадным. Для разбиения выбирается сообщение, у которого размер кадра на предыдущей итерации максимален.

Описанный алгоритм вычисления *LM*(*N*) заключается в выполнения следующих шагов:

Шаг 1. $F_{msg}^{N_0} = 1, \forall msg \in MSG_{src}$ – данная величина показывает, на сколько кадров может быть разбито данное сообщение *msg* на данной итерации.

Шаг 2. Для $k = N_0+1 \ldots N$ выполнить шаги 2.1, 2.2:

Шаг 2.1. $l = \arg\max_{msg}(c + size_{msg} / F_{msg}^{k-1})$ – данной величине присваивается сообщение, которое на предыдущем шаге разбивалось на кадр максимального размера; если таких сообщений несколько, то берется любое;

Шаг 2.2. $F_{msg}^{k} = F_{msg}^{k-1}, \forall msg \in MSG_{src} / \{l\}$, $F_{l}^{k} = F_{l}^{k-1} + 1$ – количество кадров увеличивается только для выбранного сообщения *l*;

Шаг 3. $LM(N) = \max_{msg}(c + size_{msg} / F_{msg}^{N})$ – функция возвращает максимальный размер кадра с учетом получившихся разбиения сообщений на кадры.

Следует отметить, что во многих случаях выполнено: *f*(*k*) = *f*(*k–1*). Например, в описанном случае для двух одинаковых кадров размера *size* верны следующие выкладки:

- *f*(2) = (*size* + *c*);
- *f*(3) = (*size* + *c*) – одно сообщение разбили на два одинаковых кадра, второе имеет исходный размер;

- $f(4) = (size/2 + c)$ – оба сообщения разбили на два одинаковых кадра.

Значение *LM* для агрегированного виртуального канала определяется, таким образом, путем вычисления функции $LM(N_i)$.

Вычисление джиттера для агрегированного виртуального канала производится согласно формуле (4.2.4).

В приложении А поясняется корректность описанной процедуры агрегации.

4.2.3. Построение маршрутов виртуальных каналов.

Маршрут виртуального канала представляет собой дерево, являющееся подграфом исходного графа сети *G*, корнем дерева является оконечная система-отправитель, листьями – оконечные системы-получатели кадров виртуального канала. Построение маршрута происходит с учетом ограничений на пропускную способность каналов передачи данных.

Для построения маршрута используется алгоритм поиска дерева групповой передачи, основанный на алгоритме, описанном в работе [41]. Схема построения маршрута виртуального канала *vl* заключается в выполнении следующих шагов:

Шаг 1. Преобразовать граф *G*, убрав из него все дуги $e \in E$, для которых остаточная пропускная способность $R_e - \sum_{v \in VL: e \in Tree_v} bw(v)$ меньше, чем требуемая пропускная способность $bw(vl)$. Полученный граф *G'* состоит только из тех дуг, через которые может быть проложен маршрут для данного виртуального канала.

Шаг 2. Если в графе *G'* между оконечной системой-отправителем $S_{vl} \in N$ и хотя бы одной из оконечных систем-получателей $d_{vl} \in D_{vl} \subset N$ не существует пути, то алгоритм возвращает неуспех.

Шаг 3. Запустить алгоритм нахождения кратчайшего пути в графе G' от оконечной системы-отправителя S_{vl} согласно заданному критерию $l(n), n \in (N \cup K)$. Остановить алгоритм после нахождения кратчайшего пути $path(d_{vl})$ до одной из оконечных систем-получателей из множества D_{vl}, до которых еще не найден путь.

На данном шаге предлагается использовать алгоритм Дейкстры либо алгоритм нахождения *k* кратчайших путей в графе [51]. В случае использования алгоритма Дейкстры поиск маршрута происходит быстрее, однако поиск производится только по одному критерию. Предлагается использовать один из следующих критериев для алгоритма Дейкстры:

- $l(n) = | path(S_{vl}, n) |$ – поиск пути минимальной длины; каждой дуге в процессе работы алгоритма Дейкстры присваивается вес, равный 1.
- $l(n) = \sum_{e \in path(S_{vl}, n)} \frac{\varepsilon + \sum_{v \in VL: e \in Tree_v} bw(v)}{R_e}$ – поиск пути «максимальной ширины».

Второй критерий предложен в работе [41] и заключается в том, что каждой дуге сопоставляется вес, равный занятой ширине канала и выражающейся в виде отношения зарезервированной пропускной способности к максимальной пропускной способности канала. Величина $\varepsilon > 0$ задается для сравнения весов дуг в случае, если через дугу не проходит ни один виртуальный канал. Данный критерий является более предпочтительным, так как учитывает загруженность каналов передачи данных.

В случае использования алгоритма *k* кратчайших путей для некоторого заранее заданного *k* производится поиск не более *k* маршрутов по одному из критериев, указанному выше, среди которых выбирается наилучший маршрут согласно описанному далее критерию. В качестве этого критерия предлагается использовать эвристику, которая оценивает длительность передачи кадра. Получить надежную оценку на данном этапе невозможно, так как длительность передачи кадра зависит от маршрутов других виртуальных каналов (в результате ожидания передачи кадров других виртуальных каналов в очередях коммутаторов), которые еще могут быть не построены. Поэтому на данном шаге предлагается эвристика, оценивающая стоимость пути следующим образом:

$$\cos t(path, vl) = \sum_{e \in path(S_{vl}, n)} LM_{vl} / R_e + \sum_{v \in path(S_{vl}, n), v \neq vl} LM_{vl} / \max_{e \in path}(R_e).$$

В данной формуле стоимость складывается из длительности передачи кадра текущего виртуального канала по всем каналам передачи данных данного пути и из оценки ожидания передачи кадров других виртуальных каналов. Эта оценка складывается из ожидания по одному кадру каждого виртуального канала, маршрут которого пересекается с текущим рассматриваемым путем. Среди найденных кратчайших путей выбирается путь *path* с наименьшим значением описанной функции *cost*(*path*,*vl*).

Шаг 4. Если найдены пути до всех получателей, то вернуть множество маршрутов $path(d_{vl})$ до всех оконечных систем-получателей. Объединение построенных путей $Tree_{vl} = \bigcup_{d_{vl} \in D_{vl}} path(d_{vl})$ составляет маршрут виртуального канала. Иначе запомнить пути до найденных получателей, присвоить всем дугам данных путей вес, равный 0 (для любого из критериев), и перейти на шаг 3. Дуги, которым присвоен вес 0, соответствуют элементам дерева маршрута виртуального канала, для которых уже построен путь,

поэтому на последующих этапах построения маршрута учитывать веса данных дуг не нужно.

4.2.4. Процедура ограниченного перебора виртуальных каналов

Процедура ограниченного перебора схожа с процедурой, описанной в разделе 3.3.4. , и заключается в выполнении следующих шагов:

Шаг 1. Для каждого множества назначенных виртуальных каналов мощности не больше заданного значения выполнить шаги 1.1 – 1.7:

Шаг 1.1. Снять все маршруты виртуальных каналов из выбранного множества.

Шаг 1.2. Выполнить алгоритм построения маршрута для данного канала *vl* с помощью алгоритма, описанного в разделе 4.2.3.

Шаг 1.3. Если маршрут виртуальный канал не может быть построен, то перейти на шаг 1.7.

Шаг 1.4. Выполнить алгоритм построения маршрута для всех снятых виртуальных каналов.

Шаг 1.5. Если хотя бы один из виртуальных каналов не может быть назначен, снять виртуальный канал *vl*, перейти на шаг 1.7.

Шаг 1.6. Выполнить назначения виртуальных каналов согласно найденным маршрутам и вернут успех.

Шаг 1.7. Восстановить изначальные назначения маршрутов виртуальных каналов.

Шаг 2. Если после ограниченного перебора маршрут виртуального канала не может быть построен, вернуть неуспех.

4.2.5. Вычисление максимальной длительности и джиттера передачи сообщений.

Максимальная длительность передачи сообщения *msg*, передаваемого по виртуальному каналу *vl*, вычисляется следующим образом:

$Dur(msg) = \mu + \delta + \Delta$, где

- μ – константа, описывающее время, требуемое на разбиение сообщение на кадры и сборку сообщения из кадров (и, возможно, прохождение сообщения через некоторые обязательные этапы обработки, занимающие не более чем некоторое известное постоянное время);

- δ – максимальное время, вычисляемое с момента постановки кадров сообщения в очередь виртуального канала до момента выдачи последнего кадра из этой очереди;
- Δ – максимальная длительность передачи последнего кадра.

Величина δ, как было показано в разделах 4.2.1 и 4.2.2, вычисляется как $\delta = BAG_{vl}(\sum_{msg \in MSG_{vl}} \lceil size_{msg} / (LM_{vl} - c) \rceil - 1)$.

Величина Δ может быть вычислена с помощью одного из известных методов оценки длительности передачи кадров [45-50]. Данная величина может быть вычислена после построения маршрутов виртуальных каналов.

Максимальный джиттер передачи сообщения вычисляется как разница между максимальной и минимальной длительностью передачи сообщения:

$Jit(msg) = Dur(msg) - Dur_{\min}(msg)$.

Минимальная длительность передачи сообщения вычисляется следующим образом:

$Dur_{\min}(msg) = \mu + \delta_{\min} + \Delta_{\min}$, где

- $\delta_{\min}$ – минимальное время, вычисляемое с момента постановки кадров сообщения в очередь виртуального канала до момента выдачи последнего кадра из этой очереди;
- $\Delta_{\min}$ – минимальная длительность передачи последнего кадра.

Величина $\delta_{\min}$ вычисляется как время ожидания сообщения в случае, если на момент поступления кадров сообщения в очередь виртуального канала данная очередь пуста. Данная величина, таким образом, вычисляется как $\delta_{\min} = BAG_{vl}(\lceil size_{msg} / (LM_{vl} - c) \rceil - 1)$.

Величина $\Delta_{\min}$ вычисляется как длительность передачи кадров в случае, когда при прохождении через коммутаторы кадр поступает в пустую очередь выходного порта коммутатора и сразу передается дальше. Данная величина, таким образом, вычисляется следующим образом:

$$\Delta_{\min} = \max_{path \in Tree_{vl}} (\sum_{e \in path} \frac{LM_{vl}}{R_e} + \sum_{k \in path} t_k).$$

Длительность передачи в данном случае вычисляется как время передачи по каналам, равное $\frac{LM_{vl}}{R_e}$ для каждого канала *e*, и время задержки кадра в коммутаторах, равное некоторой фиксированной величине t_k, соответствующей длительности обработки кадра в коммутаторе *k*.

4.2.6. Процедура переконфигурации виртуального канала.

Процедура переконфигурации выполняется в случае, когда не выполняется ограничение (1.3.4) для некоторого виртуального канала *vl*. Процедура заключается в попытке заново сконфигурировать параметры LM_{vl} и BAG_{vl} на основе вычисленной длительности передачи кадра. В разделах 4.2.1 и 4.2.2 задавалась некоторая начальная оценка величины Δ. Пусть данная оценка равна Δ_0 (например, $\Delta_0 = 1$ *мс*). После построения маршрутов виртуальных каналов данная величина уже может быть оценена точнее, как указано в разделе 4.2.5. С помощью новой оценки можно заново сконфигурировать параметры виртуального канала с помощью метода, указанного в разделах 4.2.1 или 4.2.2.2 (в зависимости от количества сообщений, передаваемых через виртуальный канал).

Общая схема процедуры переконфигурации следующая:

Шаг 1. Вычислить необходимую максимальную длительность кадра Δ.

- В случае нарушения длительности передачи сообщения данная величина вычисляется как $\Delta_0 + (Dur(msg) - \tau_{msg})$, то есть требуемая длительность передачи увеличивается на величину, на которую она нарушается;
- В случае нарушения максимального джиттера передачи сообщения данная величина вычисляется как $\Delta_0 + (Jit(msg) - J^*_{msg})$, то есть требуемая длительность передачи увеличивается на величину, на которую нарушается максимальный джиттер;

Шаг 2. Параметры виртуального канала пересчитываются с помощью метода, описанного в разделе 4.2.1 или 4.2.2.2 с новой оценкой длительности передачи кадра. Если метод не может получить конфигурацию, удовлетворяющую новым требованиям, вернуть неуспех и отказать в назначении данного сообщения[14].

Шаг 3. Проверить, выполняются ли ограничение (1.3.3) для виртуального канала на оконечной системе-отправителе. Если данные требования не выполняются, отказать в назначении данного сообщения.

[14] В случае, когда в виртуальном канале передаются кадры нескольких сообщений, неуспех возвращается для сообщения, у которого нарушалось ограничение (1.3.4). Для остальных сообщений необходимо проверить выполнимость требований и провести процедуру переконфигурации в случае их невыполнимости.

Шаг 4. Проверить, выполняются ли ограничение (1.4.1) на маршруте передачи виртуального канала. Если ограничение выполняется, перейти на шаг 5, иначе выполнить шаги 4.1 – 4.2:

Шаг 4.1. Отменить маршрут виртуального канала и выполнить построение маршрута согласно алгоритму из раздела 4.2.3.

Шаг 4.2. Если процедура построения маршрута возвращает неуспех, отказать в назначении данного сообщения.

Шаг 5. Пересчитать максимальные длительности и джиттеры сообщений, для которых данные требования уже были проверены и выполнялись. Если хотя бы для одного из них требование нарушается, отказать в назначении данного сообщения.

Шаг 6. Пересчитать максимальную длительность и джиттер данного сообщения. Если требования снова не выполняются, присвоить $\Delta_0 = \Delta$ и перейти на шаг 1. Данный шаг может выполняться заданное ограниченное число итераций.

4.3. Свойства алгоритмов и теоретическое обоснование

4.3.1. Оценка сложности этапов алгоритма

Пусть в сети передается множество сообщений *MSG*, $M=|MSG|$ – количество сообщений, каждое сообщение $msg \in MSG$ имеет размер $size_{msg}$. Пусть задача ставится для сети AFDX с *K* коммутаторами, *E* оконечными системами, и пропускная способность каналов передачи данных не превосходит *R*. Пусть также от каждого абонента исходит не более M_a сообщений.

Вычислительная сложность алгоритмов на каждом из этапов вычисляется следующим образом:

1) Настройка параметров виртуальных каналов.

На данном этапе для каждого сообщения строится ровно один виртуальный канал и настраиваются его параметры: *BAG*, *LM*, *JM*. В процессе настройки *BAG* и *LM* используется O(1) операций, в процессе вычисления *JM* – сумма из не более чем O(*M*) элементов. Таким образом, вычислительная сложность данного этапа O(M^2).

2) Агрегация виртуальных каналов.

Сначала производится сортировка виртуальных каналов в порядке убывания их стоимости для последующего перебора пар виртуальных каналов. Так как рассматриваются только виртуальные каналы с сообщениями, исходящими из одного абонента, то сложность для данного этапа не превосходит $O(M_a \log M_a)$. В наихудшем случае при проведении агрегации перебираются все пары виртуальных каналов,

исходящих их одного абонента. Максимальное количество попыток агрегировать виртуальные каналы (т.е. выполнить процедуру из раздела 4.2.2.2) равно $C^2_{M_a} = M_a(M_a - 1)/2 = O(M_a{}^2)$.

При каждом вызове процедуры выполняется не более O(M_a) действий для определения количества кадров, на которое делятся сообщения (т.к. сообщений в виртуальном канале не больше M_a и необходимо найти минимальные значения по их параметрам). После того, как известно количество кадров N, выполняется жадная процедура для вычисления параметра LM виртуального канала. Данная процедура выполняет цикл из не более чем N этапов, на каждом из которых выполняется не более O(M_a) операций. Учитывая, что максимальное значение N равно $N_{\max} = \sum_{msg \in MSG} \lceil size_{msg} / 64 \rceil$ (так как минимальный размер кадра 64 байта), получаем вычислительную сложность процедуры агрегации $O(M_a{}^3 N_{\max})$.

Учитывая, что сложность данного этапа вызывается не более O(1) раза для каждого виртуального канала (один раз при нарушении свойства (1.3.3) и один раз при нарушении свойства (1.3.4) и неуспешном выполнении процедуры переконфигурации), общая вычислительная сложность данного этапа в наихудшем случае составляет $O(N_{\max} M \cdot M_a{}^4 \log M_a)$.

Замечание. Данная процедура при, например, параметрах M_a = 100 имеет сложность в наихудшем случае, сопоставимую с числом 10^{11} (при выполнении для всех виртуальных каналов). Однако следующие факты поясняют, что вычислительная сложность в среднем является на порядки ниже, чем сложность в наихудшем случае:

- процедура вычисляется только не для каждого виртуального канала, а только в случаях нарушения ограничений (1.3.3) и (1.3.4) на определенных этапах алгоритма;
- процедура прерывает свое выполнение в случае нахождения пары виртуальных каналов, пропускная способность которой меньше суммы пропускных способностей исходных виртуальных каналов;
- после каждого успешного завершения процедуры количество виртуальных каналов уменьшается, поэтому при последующих вызовах процедуры выполняется меньше действий.

Сложность успешного выполнения процедуры для первой пары виртуальных каналов сложность в наихудшем случае составляет $O(N_{\max} M_a{}^3 \log M_a)$ и составляет порядка 10^7 при M_a = 100 (при небольшой N_{max}), что уже не является большим числом для

современных процессоров. Предполагая, что процедура будет каждый раз завершаться после перебора нескольких виртуальных каналов, получаем приемлемую вычислительную сложность.

3) Построение маршрутов виртуальных каналов.

Первым этапом данной процедуры является сортировка виртуальных каналов согласно их стоимости. Стоимость данного этапа не превосходит $O(M \log M)$.

Для каждого виртуального канала вызывается алгоритм Дейкстры или алгоритм k кратчайших путей не более E раз (количество получателей виртуального канала не больше количества оконечных систем). В случае использования алгоритма Дейкстры сложность $O((K+E)^2)$. В случае использования алгоритма k кратчайших путей для некоторого наперед заданного k производится запуск алгоритма Дейкстры не более $O(kE)$ раз. Общая сложность $O(k\,E(K+E)^2)$.

Общая сложность процедуры, таким образом, не превосходит $O(M \log M\ E(K+E)^2)$ в случае алгоритма Дейкстры, и , $O(M \log M\ kE^2\ (K+E)^2)$ в случае использования алгоритма k кратчайших путей.

4) Процедура ограниченного перебора виртуальных каналов.

Пусть глубина перебора равна m. Количество перебираемых подмножеств не превосходит C_M^m. Сложность на каждом этапе $O((m+1)E(K+E)^2)$ (алгоритм из пункта 3 вызывается не более $m+1$ раз). Процедура ограниченного перебора вызывается не более $O(M)$ раз.

Общая сложность данной процедуры в наихудшем случае, таким образом, $O(MC_M^m(m+1)E(K+E)^2)$ (или $O(MC_M^m(m+1)kE^2(K+E)^2)$ в случае использования алгоритма k кратчайших путей).

5) Вычисление максимальной длительности и джиттера передачи сообщений.

Вычислительная сложность на данном этапе зависит от выбранного метода вычисления оценки длительности передачи кадров. Пусть вычислительная сложность метода является значением функции $C(M, K, E, R)$. Предполагается, что сложность зависит[15] только от максимального количества виртуальных каналов M, коммутаторов K, оконечных систем E и пропускной способности каналов передачи данных R.

[15] Вычислительная сложность метода может быть более сложной функцией и зависеть от большего числа параметров, однако без ограничения общности в данном случае предлагается представлять функцию в данном виде.

При вычислении длительности передачи и джиттера производятся вычисления параметров δ, Δ, $\delta_{\min}$ и $\Delta_{\min}$.

Вычисление параметров δ и $\delta_{\min}$ для всех виртуальных каналов содержит не более чем O(M) сложений и умножений. Вычисление $\Delta_{\min}$ содержит не более $O(M \cdot K \cdot E)$ операций (O(K) сложений для каждой из веток дерева маршрута передачи виртуального канала, количество которых O(E) для каждого виртуального канала).

Общая сложность процедуры для всех виртуальных каналов, таким образом, равна $O(M(K \cdot E + C(M, K, E, R)))$.

6) Процедура переконфигурации виртуальных каналов.

Пусть для каждого виртуального канала количество выполнений переконфигурации ограничено числом l. При каждой попытке переконфигурации выполняются следующие действия:

- вычисление длительности кадра, вычислительная сложность $C(M, K, E, R)$;
- вычисление параметров виртуального канала, сложность в наихудшем случае $O(M_a{}^2 N_{\max})$ (при выполнении процедуры из раздела 4.2.2.2);
- проверка выполнения ограничений на пропускную способность каналов передачи данных, сложность данного этапа O(K);
- перестроение маршрута в случае невыполнения ограничений, сложность O((K+E)^2);
- перевычисление длительностей и джиттеров сообщений, сложность в наихудшем случае $O(M(K \cdot E + C(M, K, E, R)))$.

Сложность в наихудшем случае для данной процедуры, таким образом, равна $O(l \cdot M \cdot (M_a N_{\max} + (K+E)^2 + M \cdot K \cdot E + M \cdot C(M, K, E, R)))$.

Замечание. Сложность в среднем для данной процедуры является ниже сложности в наихудшем согласно следующим свойствам:

- процедура выполняется только для виртуальных каналов, для сообщений которых не выполняется одно из ограничений (1.3.4);
- процедура выполняется до получения первой успешной конфигурации;
- перестроение маршрута виртуального канала производится только в случае невыполнения ограничений на пропускную способность после определения новых параметров виртуального канала;
- перевычисление длительностей и джиттеров производится только для тех сообщений, для которых эта проверка уже была выполнена.

4.3.2. Корректность

Решение задачи построения сетей AFDX является корректным, если для всех сообщений/виртуальных каналов выполнены ограничения (1.3.1) – (1.3.4). Кроме того, должны учитываться джиттер порождения сообщения J_{msg} и ограничения, налагаемые стандартом, такие как ограничения размера кадров, наличие маршрута от каждого отправителя до каждого из получателей и т.п.

Корректность описанного алгоритма построения сетей AFDX поясняется следующими выкладками:

1. На первом шаге для каждого сообщения строится ровно один виртуальный канал и для него настраиваются параметры *LM*, *BAG* и *JM*. Настройка производится с учетом ограничений (1.3.2). Кроме того, на данном этапе учитывается параметр J_{msg} и параметр δ, соответствующий времени выдачи последнего кадра с момента порождения сообщения. Корректность построенного решения (т.е. гарантия того, что при данных параметрах будут выполняться ограничения (1.3.2) и ограничения на время выдачи последнего кадра) пояснена в работе [39].
2. При невыполнении ограничений (1.3.3) на джиттер виртуального канала производится процедура агрегации. Агрегация производит построение виртуальных каналов с учетом ограничений (1.3.2), J_{msg} и δ. Корректность процедуры агрегации в плане соответствия стандарту пояснена в приложении А. При невозможности составить виртуальный канал с учетом требований (1.3.2) и (1.3.3) сообщение не назначается.
3. Для каждого виртуального канала производится построение виртуального канала с учетом ограничений (1.3.1). При невозможности построения маршрута вызывается процедура ограниченного перебора.
4. При успешном выполнении процедуры ограниченного перебора производится перестроение маршрутов некоторых виртуальных каналов и текущего виртуального канала таким образом, чтобы для всех маршрутов выполнялись ограничения (1.3.1). При неуспешном выполнении процедуры производится отказ в назначении одного из сообщений (вместе в виртуальным каналом, если он включал в себя только одно сообщение) и переход на этап выполнения процедуры агрегации (см. пункт 2).
5. Для всех виртуальных каналов, для которых уже выполнены ограничения (1.3.1) – (1.3.3), производится вычисление оценки длительности и джиттера передачи сообщения. Вычисление длительности является корректным, если корректным является выбранный метод вычисления оценки длительности передачи кадра.

Оценка при этом может быть завышенной, однако при выполнении ограничения (1.3.4) для завышенной оценки ограничение выполняется и для реального значения длительности.

6. При невыполнении ограничения (1.3.4) производится процедура переконфигурации, которая повторяет отдельные этапы предыдущих этапов алгоритма, корректность которых уже пояснена.
7. Если после ограниченного числа повторений процедуры ограничения (1.3.4) не выполняются, производится попытка агрегации сообщений с проверкой всех требований (1.3.1) – (1.3.4). При неуспешном выполнении процедуры производится отказ в назначении сообщения.

После выполнения шагов алгоритма остаются только те виртуальные каналы с для них построенными маршрутами и те сообщения, для каждого из которых выполнены ограничения (1.3.1) – (1.3.4), учтены параметры J_{msg} и ограничения, налагаемые стандартом. Построенное решение, таким образом, является корректным.

4.3.3. Свойства процедуры ограниченного перебора при поиске маршрутов виртуальных каналов

Пусть производится назначение виртуального канала *vl* с требуемой пропускной способностью *r*(*vl*). Виртуальный канал следует из некоторого элемента e_1 в некоторый элемент e_2 (или в некоторое множество элементов, все упомянутые ниже рассуждения относятся к одному элементу-получателю и верны также для множества получателей путем разделения виртуального канала на несколько «подканалов» к разным получателям).

Пусть попытка провести маршрут с достаточной пропускной способностью от e_1 к e_2 оканчивается неудачей. Как и в разделе 3.4.5. , обозначим через *G** граф доступных ресурсов, *G'* – граф недоступных ресурсов, получаемый при вычете *G* из *G**, *H** – граф, включающий в себя все возможных маршруты без циклов (без учета требований пропускной способности) от e_1 к e_2, *H* – пересечение графов *G'* и *H**.

Для виртуального канала *vl* верны утверждение 3.2 и следствие 3.2 из раздела 3.4.5. . Кроме того, верно следующее утверждение.

Утверждение 4.1. Если для поставленной задачи верно, что из всех оконечных систем выходит не более одного канала передачи данных, то процедуру ограниченного перебора виртуальных каналов достаточно проводить среди тех назначенных виртуальных каналов, которые проходят через хотя бы один элемент из графа *H** (граф, содержащий все доступные маршруты без циклов для *vl*).

Доказательство.

Очевидно, что некоторый виртуальный канал vl_1, не проходящий через граф H^*, при переназначении не сможет увеличить свободную пропускную способность элементов графа H и, таким образом, не позволит назначить элемент vl.

Предположим, что возможна ситуация, при которой переназначение виртуального канала vl_1 позволит переназначить некоторый другой канал vl_2, проходящий через граф H (что может привести к назначению vl). Пусть виртуальный канал vl_2 соединяет элементы e_3 и e_4. Так как возможно переназначение виртуального канала vl_2 с исходного маршрута $path_1$ на маршрут $path_2$ (после указанного переназначения vl_1), то существует цикл в графе G. Фактически происходит переназначение между некоторыми частями маршрутов $subpath_1 \subset path_1$ на $subpath_2 \subset path_2$, причем $subpath_1$ и $subpath_2$ имеют одинаковые конечные вершины (см. рисунок 4.3). Так каналы передачи данных, соответствующие этим маршрутам, формируют цикл, то эти каналы передачи данных входят в H^*. По предположению, переназначение маршрута $subpath_1$ на маршрут $subpath_2$ невозможно без переназначения vl_1, из чего следует, что изначальный маршрут vl_1 должен содержать в себе элементы из пути $subpath_2$, и, следовательно, из множества H^*. Данное противоречие завершает доказательство.

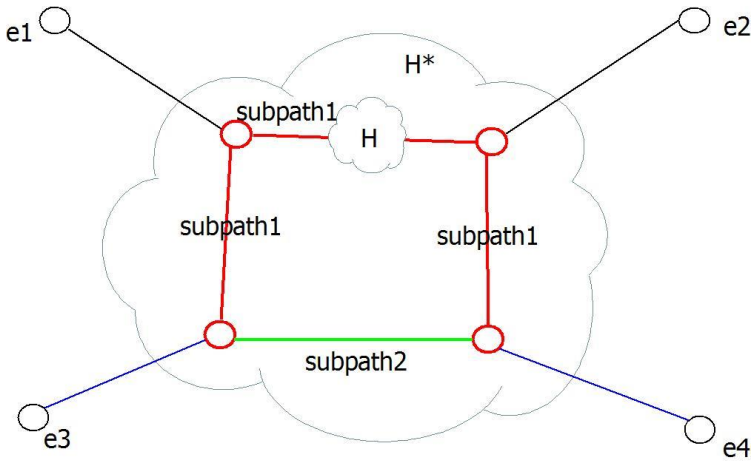

Рисунок 4.3. К доказательству утверждения 4.1.

Условия данного утверждения позволяют использовать его при построении маршрутов в сетях AFDX. Таким образом, можно сократить пространство перебора с помощью следующих проверок:

1. В случае, когда оконечные системы имеют не более одного смежного канала передачи данных, перебирать только множества виртуальных каналов, проходящие через элементы графа H^*.

2. Проверить, согласно следствию 3.2, что хотя бы один виртуальный канал входит во множество *H*.
3. Проверить выполнимость условий утверждения 3.2.
4. Если все три пункта выполнены, провести попытку переназначения.

4.3.4. Оценка точности процедуры построения маршрутов виртуальных каналов

Утверждение 4.2. Пусть все каналы передачи данных имеют равную пропускную способность *R*, а пропускная способность, требуемая для построения маршрута виртуального канала, не превосходит $R_{\max}<R$. Пусть при построении маршрутов число каналов передачи данных, зарезервированная пропускная способность которых превысила $R - R_{\max}$, равно *l*. Тогда отношение суммарной требуемой пропускной способности маршрутов, построенных жадным алгоритмом, к суммарной пропускной способности при построении маршрутов оптимальным алгоритмом, не меньше $\frac{R - R_{\max}}{R + (l-1) * R_{\max}}$.

Доказательство данного утверждения приведено в Приложении Б.

Пример. В сетях AFDX согласно стандарту могут использоваться каналы передачи данных с пропускной способностью 100 мбит/с = 12500 байт/мс. При этом требуемая пропускная способность виртуальных каналов в силе ограничений на размер кадра и на значение BAG не превосходит 1518 байт/мс.

Значение *l*, указанное в теореме, означает количество «узких каналов» сети, то есть каналов передачи данных с высокой загрузкой. Если, например, в сети имеется только один такой канал, то $\frac{R_a}{R_o}\frac{R - R_{\max}}{R + (l-1)R_{\max}} = \frac{12500 - 1518}{12500} > 0{,}87$.

4.3.5. Достаточные условия оптимальности алгоритма построения сетей AFDX

Утверждение 4.3 (достаточные условия выполнения ограничений на длительность передачи сообщений). Пусть для некоторой сети AFDX, по которой передается множество сообщений *MSG* по множеству виртуальных каналов *VL*, верны следующие условия:

- по виртуальному каналу *vl* передается множество сообщений MSG_{vl};
- расстояние без циклов между любыми двумя оконечными системами не превышает *k* коммутаторов;

- для любого сообщения[16] *msg* верно $J_{msg}= 0$;
- все коммутаторы сети имеют стратегию буферизацию FIFO на выходных портах;
- τ_k - наибольшее время обработки кадра в коммутаторе перед постановкой в очередь.

Тогда, если для каждого сообщения *msg*, передающегося по виртуальному каналу *vl*, верно:

$$(\sum_{m\in MSG_{vl}}(\lceil size_m/(LM_{vl}-c)\rceil-1)BAG_{vl})+(k+1)LM_{vl}/R+k\cdot\tau_k+2k\sum_{v\in VL}(LM_v/R+gap)\le\tau_{msg},$$

то для всех сообщений будет выполняться ограничение на длительность передачи.

Доказательство данного утверждения приведено в приложении В.

Утверждение 4.4 (достаточные условия построения виртуальных каналов для всех сообщений). Пусть для некоторой сети AFDX производится проектирование виртуальных каналов для множества сообщений $MSG=\{msg\}$, причем выполнены следующие условия:

- Все каналы передачи данных имеют пропускную способность не ниже *R*;
- Для каждого сообщения *msg* определены следующие параметры:
  - $n=\min\limits_{i=0..7}(\left\lfloor\frac{\tau_{msg}-\Delta_0}{2^i}\right\rfloor+1,\left\lfloor\frac{T_{msg}}{2^i}\right\rfloor)$ (где Δ_0 - начальное приближение длительности передачи кадра), при которых выполнены ограничения:
    - $(n-1)BAG\le\tau_{msg}-\Delta_0$
    - $n\cdot BAG\le T_{msg}$
  - $k(n)=\arg\min\limits_{i=0..7}(\left\lfloor\frac{\tau_{msg}-\Delta_0}{2^i}\right\rfloor+1,\left\lfloor\frac{T_{msg}}{2^i}\right\rfloor)$;
  - $LM(n)=\max(64,\left\lceil\frac{size_{msg}}{n}\right\rceil+c)$, где c = 47 байт;
  - $BAG(n)=2^{k(n)}$.
- Для указанных параметров выполнены следующие ограничения:
  - $\sum\limits_{msg\in MSG_{es}}(\frac{LM(n)}{R}+gap)\le 500мкс$, здесь суммирование ведется по всем сообщениям, исходящим из оконечной системы *es*.

[16] В данном утверждении для упрощения рассматривается случай, когда не задано ограничение на джиттер порождения сообщения. Утверждение может быть расширено с учетом этого ограничения.

- $\sum_{msg \in MSG} \frac{LM(n)}{BAG(n)} \le R$

- Выполнены ограничения утверждения 4.3 (с учетом формирования по одному виртуальному каналу для каждого сообщения).

Тогда алгоритм спроектирует виртуальные каналы для всех сообщений.

Доказательство данного утверждения приведено в приложении Г.

Замечание. При выполнении условий утверждения 4.4 в сети производится построение по одному виртуальному каналу для каждого сообщения. Алгоритм позволяет производить построения и в некоторых случаях нарушения указанных свойств, производя, например, построение виртуальных каналов с передачей более одного сообщения в некоторых из них с помощью процедуры агрегации. Данная особенность в утверждении не рассматривается, и область действия утверждения, таким образом, ограничена небольшими сетями с не небольшим количеством передаваемых сообщений.

Пример. Пусть задана сеть AFDX, изображенная на рисунке 4.2, со следующими входными данными:

- Все коммутаторы работают по стратегии FIFO;
- Из каждой оконечной системы передаются следующие сообщения:
  - По 50 «легких» сообщений размера 30 байт, с периодом выдачи 100 мс, с ограничением на длительность передачи 100 мс.
  - По 2 «тяжелых» сообщения размера 2000 байт, с периодом выдачи 10 мс, с ограничением на длительность передачи 100 мс.
- R = 12500 байт/мс (100 Мбит/с);
- Заданы параметры Δ_0 = 1 мс, gap = 0, τ_k = 32 мкс.

Для указанной сети будут выполнены все ограничения утверждения 4.4:

- Для «легких» сообщений проектируются виртуальные каналы с параметрами LM = 64 байт, BAG = 64 мс; для «тяжелых» сообщений LM = 1047 байт, BAG = 4 мс.
- Для всех проектируемых виртуальных каналов максимальный джиттер вычисляется как $JM_{vl} = \sum_{v \in VL_{es}, v \neq vl} (\frac{LM_v}{R} + gap) \le 50 \cdot \frac{64}{12500} + 2 \cdot \frac{1047}{12500} < 0.425$ - таким образом, выполнены ограничения на максимальный джиттер.
- Общая зарезервированная пропускная способность: $\sum_{vl \in VL} \frac{LM_{vl}}{BAG_{vl}} = 50 \cdot 8 \cdot \frac{64}{64} + 2 \cdot 8 \cdot \frac{1047}{4} = 4588 < 12500$ - выполнены ограничения на пропускную способность.

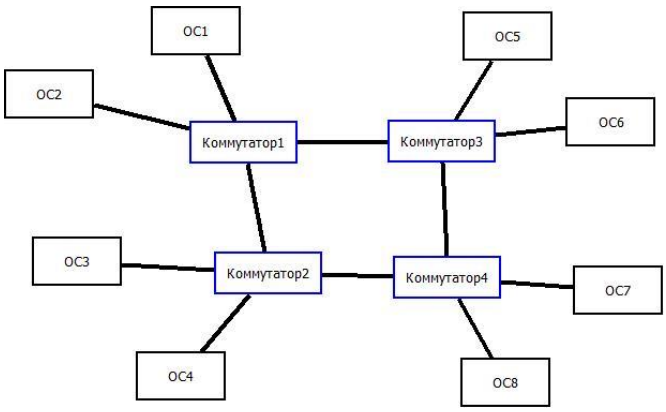

Рисунок 4.4. Сеть AFDX для примера.

Длительность передачи сообщений оценивается следующим образом:

- Максимальное время ожидания других кадров в буферах коммутаторов оценивается как

$$B \le 2k\sum_{v\in VL}(LM_v/R+gap) = 2\cdot 4\cdot(50\cdot 8\cdot\frac{64}{12500}+2\cdot 8\cdot\frac{1047}{12500}) < 28\text{ мс}$$

- Для «легких» сообщений:

$$\delta+\Delta \le (n-1)BAG+(k+1)LM_{vl}/R+k\cdot\tau_k+2k\sum_{v\in VL}(LM_v/R+gap) < 0\cdot 64+5\cdot 64/12500+4\cdot 0$$

(с учетом того, что $n = 1$, $k = 4$).

- Для «тяжелых» сообщений:

$$\delta+\Delta \le (n-1)BAG+(k+1)LM_{vl}/R+k\cdot\tau_k+2k\sum_{v\in VL}(LM_v/R+gap) < 1\cdot 4+5\cdot 64/12500+4\cdot 0.0$$

Таким образом, для указанных входных данных алгоритм успешно спроектирует виртуальные каналы для всех сообщений.

4.3.6. Выводы

В разделе 4 предложен новый алгоритм построения сетей AFDX, позволяющий по заданному множеству периодически передаваемых сообщения строить виртуальные каналы и маршруты для них с учетом ограничений реального времени. Алгоритм позволяет задавать баланс между точностью и вычислительной сложностью путем ограничения мощности перебираемых множеств виртуальных каналов в процедуре ограниченного перебора, а также задания количества итераций в процедуре переконфигурации виртуальных каналов. Получена верхняя оценка вычислительной сложности алгоритма, обоснована корректность и доказан ряд свойств предложенной процедуры ограниченного перебора. Получены оценки точности процедуры построения маршрутов виртуальных каналов для ряда частных случаев. Получены достаточные

условия, при которых производится построение виртуальных каналов и маршрутов с учетом всех ограничений для всех сообщений.

5. Экспериментальное исследование свойств алгоритмов

5.1. Методика проведения экспериментального исследования

Экспериментальные исследования проведены для сравнения предложенных жадных критериев и для выявления влияния отдельных процедур на качество и вычислительную сложность получаемого решения.

Для исследования ряда свойств производится постановка статистических гипотез [54, 55] и их проверка после серии запусков.

Пусть производится серия экспериментов, в которых на вход задаче подается случайный набор входных данных (входные данные генерируются случайным образом) и после работы алгоритма измеряется значение некоторой наблюдаемой величины X. Данное значение можно считать случайной величиной; некоторое высказывание, несущее предположение о свойстве данной величины, называют статистической гипотезой.

Постановку и проверку статистической гипотезы будем проводить в несколько этапов [55]:

1. Располагая выборочными значениями X_1, … X_n наблюдаемой величины X, формулируется нулевая (основная) гипотеза H_0 и альтернативная гипотеза H_1.
2. Задается уровень значимости α, который представляет собой ошибку первого рода, то есть вероятность того, что принимается альтернативная гипотеза в то время, как верна основная гипотеза. Если наблюдаемое распределение наблюдаемых значений соответствует уровню значимости, то принимается основная гипотеза.
3. Формируется критерий φ, являющийся функцией от наблюдаемых значений и подчиняющейся некоторому закону распределения, по которому можно судить о степени расхождения наблюдаемых величин от основной гипотезы.
4. Выделяется критическая область значений критерия φ, при попадании в которую принимается альтернативная гипотеза; критическая область выбирается таким образом, чтобы вероятность попадания в нее была равна уровню значимости α.
5. Производится вычисление критерия φ согласно полученным наблюдаемым величинам. Если значение попадает в критическую область, принимается

альтернативная гипотеза, иначе принимается сформулированная основная гипотеза.

Так как проводимые экспериментальные запуски являются независимыми, то среднее арифметическое значений X_1, … X_n при достаточно больших n приближается к математическому ожиданию величины X. Кроме того, используя центральную предельную теорему, среднее арифметическое аппроксимируется нормальным распределением:

$$\frac{\sum X_i - \mu n}{\sigma\sqrt{n}} \to N(0,1),$$

здесь μ – математическое ожидание, σ – дисперсия случайно величины X.

В представленных экспериментах ставится статистическая гипотеза о значении случайно величины $X = X_0$, выбираются стандартные значения, например, $n = 100$, $\alpha = 0.05$, и предлагается критерий $\varphi = \frac{\sum X_i - X_0 n}{\sigma\sqrt{n}}$ при $\sigma^2 = \frac{\sum (X_i - X_0)^2}{n-1}$ (см. [55]).

Для нормального распределения $N(0,1)$ по значениям n и α можно определить критическую область. Например, при $n = 100$ и $\alpha = 0.05$ для выбранного критерия критическая область составляет область значений $(-\infty,-2) \cup (2,\infty)$. Если при полученных наблюдениях значение критерия φ не входит в указанную область, то гипотеза о значении величины принимается.

5.2. Экспериментальные исследования для алгоритма планирования вычислений в ЦОД

5.2.1. Схема проведения экспериментов

Экспериментальные исследования проводились для выявления следующих свойств алгоритма планирования вычислений в ЦОД.

1. Зависимость качества получаемых назначений и вычислительной сложности алгоритма от выбранного способа расчета стоимости элемента: вычисление с выбором критической характеристики или взвешенная сумма по всем критериям (см. разд. 3.2.2.). Кроме того, проводилось сравнение указанных методов со стратегией случайного назначения элемента.
2. Исследование точности разработанного алгоритма и сравнение со стратегиями случайного назначения и «первый подходящий» на совместимых входных данных, т.е. данных, для которых гарантируется существование оптимального

решения при известных значениях загрузок ресурсов; кроме того, исследовалась эффективность алгоритма в зависимости от глубины ограниченного перебора.

3. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от выбора метода назначения: компактное или сбалансированное назначение (см. разд. 3.2.2.).
4. Эффективность процедуры репликации.

Для каждого из исследуемых свойств проводилась серия запусков с измерением следующих параметров:

- количество назначенных запросов;
- загрузка по ресурсным характеристикам – вычисляется как отношение суммарной используемой емкости к общей емкости по данной характеристике;
- время работы алгоритма[17];

Для ряда свойств ставились и проверялись статистические гипотезы о математическом ожидании некоторой величины. В данном случае для сравнения методов ставилась гипотеза о значении разности количества назначенных запросов либо разности средней/максимальной загрузки по характеристике для того/иного метода.

5.2.2. Исследование используемых эвристик

Для данного исследования проводилась серия запусков для случайно сгенерированного ЦОД, состоящего только из вычислительных узлов, со следующими характеристиками:

1) количество вычислительных узлов – 1000;
2) количество запросов – 100, в каждом запросе в среднем по 100 виртуальных машин;
3) количество характеристик виртуальных машин – 4: <число ядер>, <частота>, <объем оперативной памяти>, <объем дисковой памяти>; для каждой характеристики должно выполняться отношение недопустимости перегрузки физического ресурса (ограничение 1.2.1);
4) тесты формировались таким образом, что возможно назначение всех запросов, т.е. генерировались совместимые данные;

[17] Время измерялось на ЭВМ со следующими характеристиками: процессор Core i7 с тактовой частотой 1900 МГц, 4 Гб оперативной памяти.

5) общая загрузка ресурсов (отношение общей требуемой емкости виртуальных элементов к емкости физических элементов по данной характеристике) задавалась двумя способами:
   - полная загрузка: загрузка по всем характеристикам при назначении всех запросов близка к 100%;
   - неравномерная загрузка: загрузка формировалась следующим образом: <число ядер> – 10%, <частота> – 40%, <объем оперативной памяти> – 70%, <объем дисковой памяти> – 100%.

Для каждой комбинации "способ вычисления стоимости элемента – загрузка ресурсов" проводилось 100 запусков. При этом процедура ограниченного перебора не использовалась.

Таблица 5.1. Результаты исследований метода вычисления стоимости при полной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов, %	Средняя загрузка элементов[18], %	Максимальная загрузка по характеристикам, %	Среднее время работы алгоритмов, с
Случайное назначение	84.04	79.6	79.9	24
с выделением критической характеристики	87.86	88	95.5	23
Взвешенная сумма с учетом дефицита ресурсов	97.36	97.4	97.7	34

Усредненные результаты проведенных исследований в зависимости от выбранного метода вычисления стоимости при полной загрузке по характеристикам показаны в таблице 5.1. Результаты при неравномерной загрузке – в таблице 5.2.

Для сравнения методов также ставилась статистическая гипотеза вида $X_i - X_j = m$, где X_i – количество назначенных запросов, полученных при запуске алгоритма i-ым методом.

[18] Средняя загрузка вычисляется как среднее арифметическое из загрузок по каждой из характеристик. Данная величина используется для сравнения, насколько плотно производится упаковка.

Гипотезы такого вида проверяются для проверки предположения, что один метод работает лучше другого. Если после проведения экспериментов подтверждается гипотеза, что $m>0$, то также будем считать, что на проведенных экспериментах статистически подтверждено превосходство одного метода над другим. Гипотезы проверялись согласно методу, описанному в разделе 5.1. , с параметрами $n=100$ и $\alpha=0.05$.

Пусть F_r, F_d и F_m – количество назначенных запросов для, соответственно, случайного назначения, назначения с вычислением стоимости по критической характеристике и назначения с вычислением взвешенной суммы при полной загрузке, P_r, P_d и P_m – соответствующие величины при частичной загрузке. Для проведенных экспериментов были подтверждены следующие статистические гипотезы:

- $F_d - F_r = 4$;
- $F_m - F_r = 13$;
- $F_m - F_d = 10$;
- $P_d - P_r = 4$;
- $P_m - P_r = 5$;
- $P_m - P_d = 1$;

Кроме того, при сравнении максимальных загрузок по характеристике L_d и L_m (алгоритмы со стоимостью по критической характеристике и взвешенной стоимостью соответственно) при частичной загрузке была подтверждена гипотеза $L_d - L_m = 1\%$ (при вычислении загрузки производилось приведение к целому числу процентов).

Таблица 5.2. Результаты исследований метода вычисления стоимости при неравномерной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов, %	Средняя загрузка элементов, %	Максимальная загрузка по характеристикам, %	Среднее время работы алгоритмов, с
Случайное назначение	93.37	52	88.2	47
с выделением критической характеристики	97.79	56.5	99	46
Взвешенная сумма с учетом дефицита ресурсов	99	56.4	98	81

Полученные результаты позволяют сделать выводы, что оба указанных метода вычисления показывают лучшие результаты, чем метод со случайным назначением элементов. Кроме того, при полной загрузке лучшие результаты показывает метод с вычислением взвешенной суммы. При неполной загрузке разница между количеством назначенных запросов минимальна, при этом большую загрузку (также незначительно) показывает назначение по критической ресурсной характеристике.

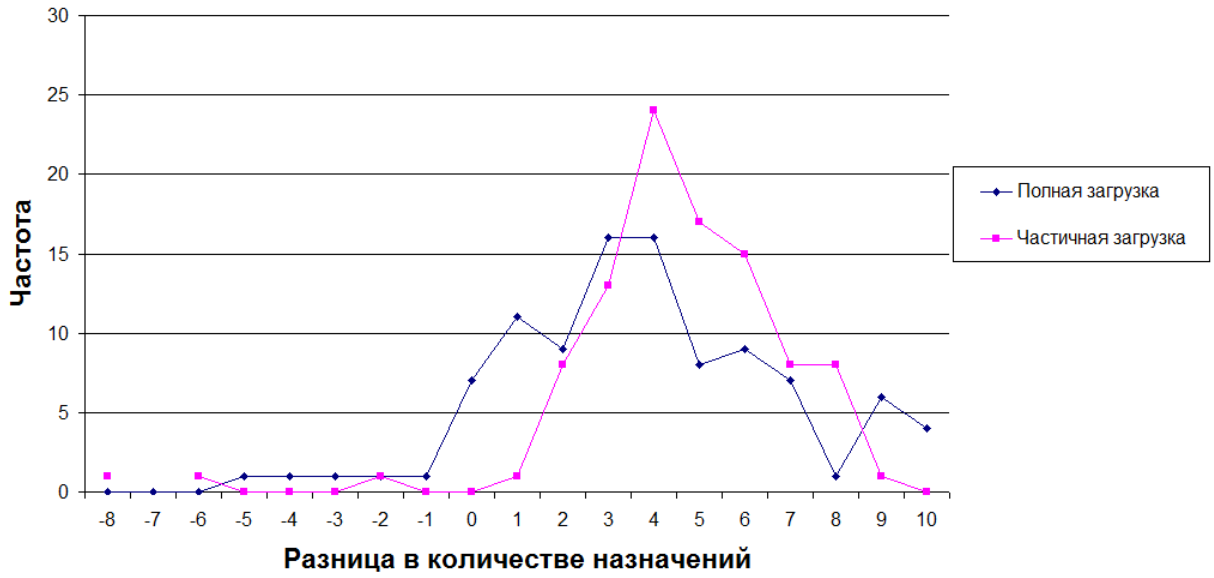

Рисунок 5.1. Частота значений разницы количества назначений при сравнении случайного назначения и назначения по критической характеристике при n = 100.

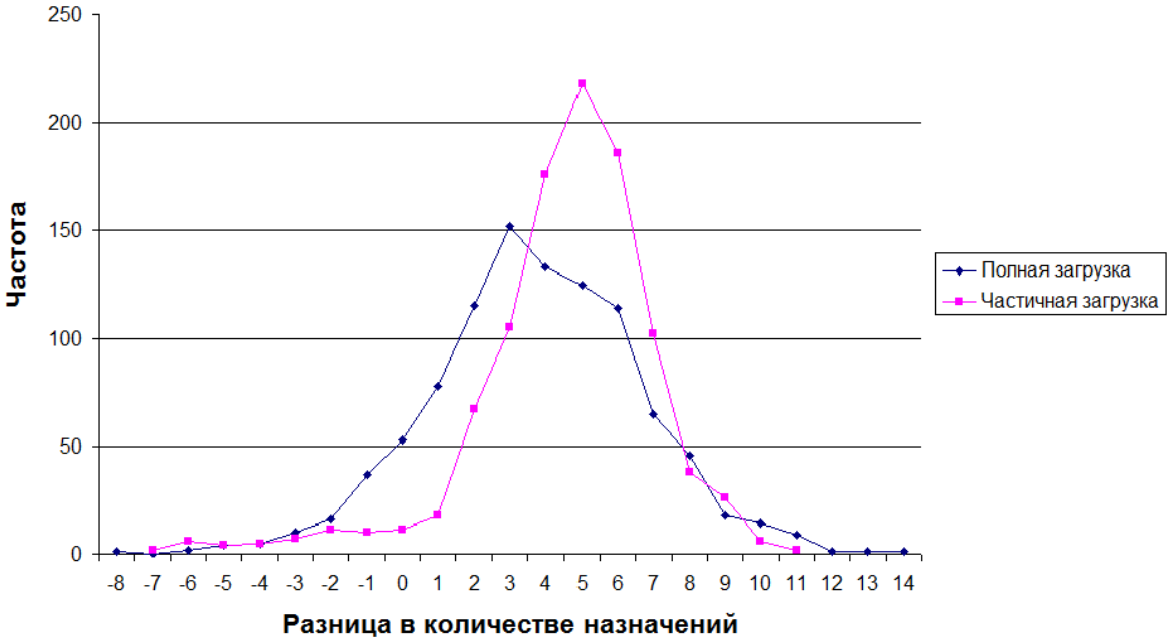

Рисунок 5.2. Частота значений разницы количества назначений при сравнении случайного назначения и назначения по критической характеристике при n = 1000.

На рисунках 5.1 и 5.2 показаны графики, на которых изображено количество экспериментов, на которых достигалась та или иная разность числа назначенных запросов при попарном сравнении методов случайного назначения и назначения по критической характеристике при n = 100 и n = 1000. Как видно из графиков, при увеличении n получаемые распределения приближаются к нормальному.

Приведенные исследования позволяют сделать следующие выводы:

1. Предложенные подходы на проведенных тестах показывают лучшие по загрузке и количеству назначенных запросов результаты, при этом их среднее время работы алгоритмов имеет тот же порядок, что и стратегия случайного назначения. Эффективность предложенных подходов также подтверждается статистическими гипотезами.
2. При полной загрузке элементов, исходя из полученных средних показателей, эффективнее работает подход с вычислением стоимости как взвешенной суммы;
3. При неполной загрузке благодаря выделению одного критического ресурса подход с выделением критической характеристики в среднем получает более компактные решения, т.е. решения, в которых максимальная по характеристикам загрузка больше, чем в других методах, даже несмотря на меньшее в среднем число назначенных запросов.

5.2.3. Исследование точности алгоритма

Для проверки точности предложенного алгоритма и его процедур генерировались совместимые тестовые наборы, для каждого из которых производился запуск того или иного алгоритма. Совместимый тестовый набор гарантирует существование решения, при котором назначаются все запросы. Тестовые наборы составляются независимо случайным образом с заданными параметрами загрузок по ресурсам.

Исследования проводились для сравнения следующих алгоритмов:

- Случайное назначение элементов (запросы и виртуальные элементы выбираются в случайном порядке, среди доступных физических элементов для назначения виртуального элемента выбирается произвольный). Для построения маршрутов использовался алгоритм Дейкстры без ограниченного перебора.
- Стратегия назначения «первый подходящий» (при выборе доступного физического ресурса для назначения виртуального выбирается первый наилучший). Для построения маршрутов использовался алгоритм Дейкстры без ограниченного перебора.
- Описанный алгоритм планирования вычисления в ЦОД без использования процедуры ограниченного перебора.
- Описанный алгоритм планирования вычисления в ЦОД с использованием процедуры ограниченного перебора глубины 1.
- Описанный алгоритм планирования вычисления в ЦОД с использованием процедуры ограниченного перебора глубины 2.

Для проверки эффективности того или иного метода входные данные разделены на два класса со следующими характеристиками:

- использовались ЦОД с топологией fattree [53] с тремя уровнями коммутационных элементов;
- всего в ЦОД имелось 100 вычислительных узлов и 100 хранилищ данных, для каждого из которых задавалась одна характеристика;
- назначение проводилось для 100 запросов, в каждом из которых – в среднем 15 виртуальных каналов;
- средняя загрузка вычислительных узлов и хранилищ данных фиксирована и равна для первого класса данных 99%, для второго класса данных – 50%;
- средняя загрузка сетевых ресурсов ЦОД варьировалась для первого класса данных равна 50%, для второго – 90%.

Усредненные результаты сравнения количества назначенных запросов и времени работы алгоритмов для первого и второго класса данных представлены в таблице 5.3.

Таблица 5.3. Результаты исследования алгоритмов на совместимых данных

Алгоритм	Точность: среднее количество назначенных запросов, %		Среднее время работы алгоритмов, с
	Первый класс данных	Второй класс данных	
Случайное назначение	94.3	56	2
Стратегия первый подходящий	93.7	17.4	1.5
Алгоритм планирования вычислений без ограниченного перебора	96	56	3
Алгоритм планирования вычислений с ограниченным перебором глубины 1	96.85	56	3.5
Алгоритм планирования вычислений с ограниченным перебором глубины 2	96.85	56	5

Пусть P_r, P_f, P_0, P_1 и P_2 – процент назначенных запросов для, соответственно, случайного назначения, назначения с стратегией первый подходящий и назначения алгоритмом с ограниченным перебором глубины 0, 1 и 2 соответственно для первого класса данных. Пусть N_r, N_f, N_0, N_1 и N_2 – соответствующие величины второго класса данных. При проверке статистических гипотез о разности количества назначаемых запросов для каждого из классов данных можно сделать выводы, что подтверждаются следующие гипотезы (проверяемые согласно методу, описанному в разделе 5.1 с параметрами $n = 100$ и $\alpha = 0.05$):

1. Алгоритм планирования в ЦОД для первого класса данных назначает больше запросов, чем алгоритмы со случайным назначением и со стратегией «первый подходящий»:

- $P_0 - P_r = 2$;
- $P_1 - P_r = 3$;

- $P_0 - P_f = 2$;
- $P_1 - P_f = 3$;

2. Алгоритм планирования в ЦОД для второго класса данных назначает больше запросов, чем алгоритмы со стратегией «первый подходящий» и столько же запросов, сколько и алгоритм со случайным назначением:

- $N_0 - N_r = 0$;
- $N_1 - N_r = 0$;
- $N_0 - N_f = 38$;
- $N_1 - N_f = 38$;

3. При увеличении глубины перебора с 0 на 1 для первого класса данный алгоритм получает незначительное увеличение количества назначенных запросов, в остальных случаях улучшения точности не наблюдалось:

- $P_1 - P_0 = 1$;
- $P_\mathrm{i} - P_\mathrm{j} = 0$ при $\mathrm{i,j} \in \{0, 1, 2\}$, $(\mathrm{i.j}) \neq (0,1)$;
- $N_\mathrm{i} - N_\mathrm{j} = 0$ при $\mathrm{i,j} \in \{0, 1, 2\}$;

Приведенные исследования позволяют сделать следующие выводы:

1. Предложенный алгоритм в целом показывает лучшую точность, чем алгоритмы со стратегиями случайного назначения и «первый подходящий».
2. Процедура ограниченного перебора позволяет незначительно улучшить точность только в случае, когда критическими ресурсами являются виртуальные машины/storage-элементы.
3. Алгоритм показывает большую точность в случае, когда критическими ресурсами являются виртуальные машины/storage-элементы, чем в случае, когда критическими ресурсами является сеть обмена данными. Это можно объяснить тем, что при назначении виртуальных машин и систем хранения данных алгоритм не учитывает остаточную пропускную способность сетевых элементов, что приводит к неоптимальному назначению виртуальных каналов, если критическими ресурсами является сеть обмена данными.
4. Все алгоритмы показывают сопоставимое время работы.

5.2.4. Зависимость от метода выбора физического ресурса

В данном разделе исследуется критерий выбора физических элементов при назначении виртуальных элементов:

- компактное назначение: выбирается физический элемент с наименьшей стоимостью;

- сбалансированное назначение: выбирается элемент с наибольшей стоимостью.

Компактное назначение предлагается использовать, когда физические ресурсы сети не являются критическим ресурсом, т.е. в случаях, когда отношение стоимостей требуемых сетевых ресурсов к имеющимся сетевым ресурсам меньше заданного порога. В противном случае предлагается применить сбалансированное назначение.

Для проверки указанных предположений производились запуски тестовых наборов, описанных в предыдущем разделе, для двух методов назначения. При этом проверялись следующие статистические гипотезы, проверяемые согласно методу, описанному в разделе 5.1, с параметрами $n = 100$ и $\alpha = 0.05$:

- для первого класса данных $X_c - X_b > 0$, где X_c и X_b – количество назначенных запросов, полученных, соответственно, компактным и сбалансированным назначением;
- для второго класса данных, соответственно, $X_c - X_b < 0$, где X_c и X_b.

Полученные средние значения указаны в таблице 5.4.

В результате проведенных исследований были подтверждены описанные выше статистические гипотезы:

- $X_c - X_b = 15$ для первого класса данных;
- $X_b - X_c = 27$ для второго класса данных;

Таблица 5.4. Результаты сравнения методов назначения

Метод назначения	Точность: среднее количество назначенных запросов, %		Среднее время работы алгоритмов, с
	Первый класс данных	Второй класс данных	
Компактное назначение	96	29	2
Сбалансированное назначение	81	56	3

Указанные результаты позволяют сделать вывод, что компактное назначение показывает лучшие результаты, когда критическими ресурсами являются виртуальные машины/storage-элементы, в то время как сбалансированное назначение более эффективно в случае, когда критическими ресурсами является сеть обмена данными.

5.2.5. Исследование процедуры репликации

Для исследования эффективности процедуры репликации генерировались тестовые наборы, в которых в каждом запросе к одному storage-элементу прокладывалось несколько виртуальных каналов от виртуальных машин. Предполагается, что к данным storage-элементах производится большое количество чтений и малое количество записей, поэтому можно создавать реплики этих storage-элементов с каналом поддержания консистентности данных, не требующем большой пропускной способности.

Для исследования эффективности производились запуски алгоритма с включением и без включения процедуры репликации для следующих входных данных задачи:

- использовались ЦОД с топологией fattree с тремя уровнями коммутационных элементов;
- всего в ЦОД имелось 60 вычислительных узлов и 60 хранилищ данных;
- назначение проводилось для 100 запросов, в каждом из которых – в среднем 10 виртуальных каналов;
- средняя загрузка вычислительных узлов и хранилищ данных фиксирована и равна 75%;
- средняя загрузка сетевых ресурсов ЦОД варьировалась от 30 до 100%.

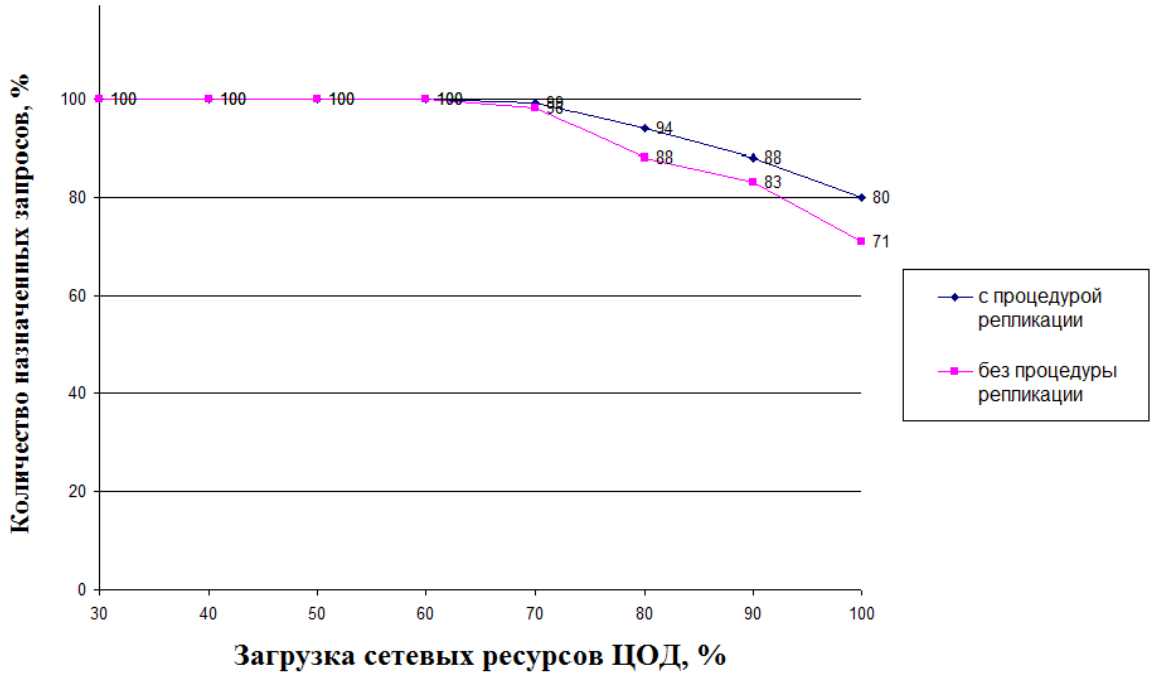

Рисунок 5.3. Исследование эффективности процедуры репликации в зависимости от загрузки сетевых ресурсов.

График количества назначенных запросов изображен на рисунке 5.3. Исходя из полученных результатов можно сделать вывод, что использование процедуры репликации

позволяет увеличить эффективность назначения, когда загрузка сетевых ресурсов превосходит 60%.

5.2.6. Выводы

Проведенные экспериментальные исследования на основе полученных усредненных результатов и подтвержденных статистических гипотез позволяют сделать несколько выводов:

- Используемые в алгоритме жадные критерии назначения позволяют получать лучшие результаты, чем стратегии случайного назначения и «первый подходящий».
- Процедура ограниченного перебора позволяет в некоторых случаях получить небольшой прирост в количестве назначаемых запросов.
- В случае, когда критическими ресурсами является сеть обмена данными, лучшие результаты показывает сбалансированное назначение; если критическими ресурсами являются виртуальные машины/storage-элементы, больше запросов позволяет назначить компактное назначение.
- Процедура репликации улучшить количество назначаемых запросов при большой загрузке сети передачи данных в случаях наличия большого количества виртуальных каналов, идущих к storage-элементам с большим количеством операций чтения и малым количеством операций записи.

5.3. Экспериментальные исследования для задачи построения сетей AFDX

5.3.1. Схема проведения экспериментов

Так как бортовые сети являются сетями небольшого размера, для экспериментальных исследований была выбрана фиксированная сеть AFDX, изображенная на рисунке 5.4.

Проводились исследования следующих свойств алгоритма:

- Эффективность использования процедур алгоритма: процедуры агрегации, ограниченного перебора и переконфигурации.
- Эффективность процедуры агрегации: зависимость количества назначенных сообщений и виртуальных каналов от количества исходных сообщений.
- Эффективность работы алгоритма в случае высоких требований к длительности передачи сообщений.

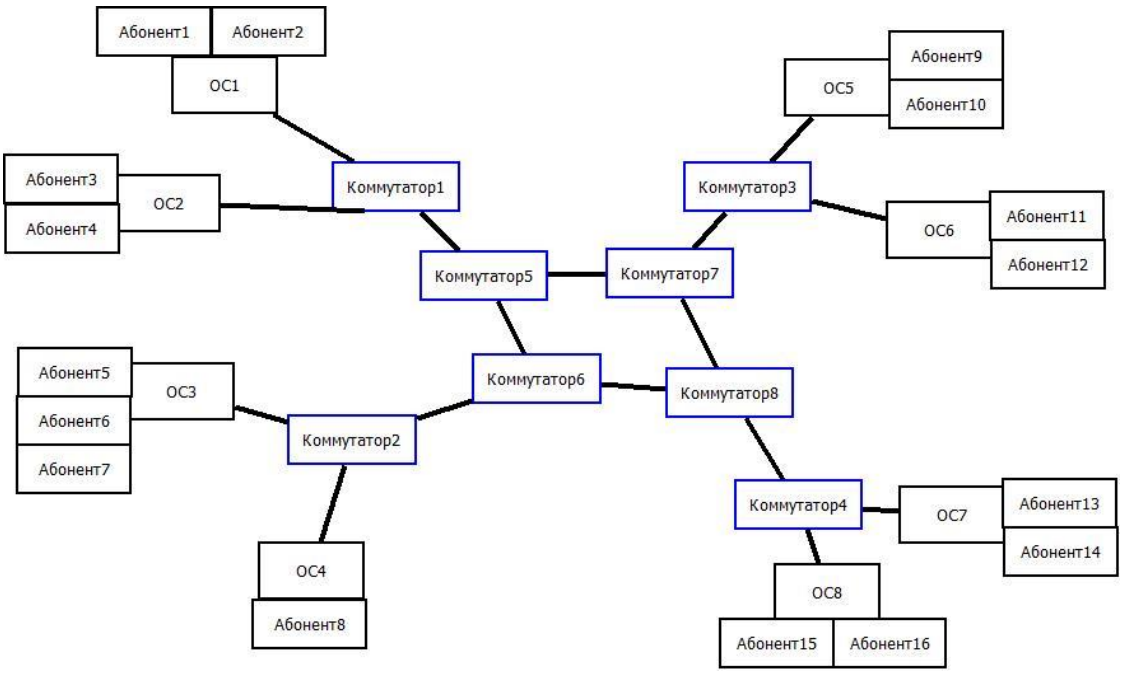

Рисунок 5.4. Сеть AFDX, используемая при проведении экспериментальных исследований.

Для проведения исследований выделялось три класса автоматически генерируемых наборов сообщений:

1. Большое количество сообщений с низкими требованиями к длительности передачи. Генерировалось 2000 сообщений с размером от 1 до 1000 байт, передаваемые с периодом от 1 до 1000 с и с ограничением длительности передачи от 0.1 до 100 с.

2. Сообщения большого размера. Генерировалось 100 сообщений размером от 1кб до 1мб с периодом от 1 до 10 с и с ограничением длительности передачи от 1 до 10 с.

3. Сообщения с высокими требованиями к длительности передачи. Генерировалось 100 сообщений размером от 0 до 1000 байт с периодом от 100 до 1000 мс и с ограничением длительности передачи от 1 до 10 мс.

Для каждого исследуемого свойства проводился запуск на 100 случайно сгенерированных наборов сообщений, после чего результаты усреднялись. Кроме того, ставились и проверялись статистические гипотезы об эффективности работы той или иной процедуры, выражающиеся в виде $E_i - D_i = k > 0$, где E_i - количество назначаемых запросов алгоритмов при включенной процедуре i, D_i – без включения процедуры.

5.3.2. Эффективность использования процедур алгоритма

В данном разделе описано экспериментальное исследование работы алгоритма на всех трех классах данных при следующих вариантах запуска:

- с использованием всех процедур, кроме процедуры агрегации, при этом использовалась процедура ограниченного перебора глубины 2;
- с использованием всех процедур, кроме процедуры переконфигурации виртуального канала, при этом использовалась процедура ограниченного перебора глубины 2;
- с использованием всех процедур, кроме процедуры ограниченного перебора;
- с процедурой ограниченного перебора глубины 1 (с использованием всех остальных процедур);
- с процедурой ограниченного перебора глубины 2 (с использованием всех остальных процедур);

Результаты исследования, а именно средний процент назначенных сообщений и количество сгенерированных виртуальных каналов для указанных выше вариантов запуска для 1-ого, 2-ого и 3-его классов наборов сообщений указаны в таблице 5.5, соответствующее среднее время работы алгоритма – в таблице 5.6.

Из полученных результатов можно сделать следующие выводы:

- Процедура агрегации позволяет существенно улучшить результат. Это можно пояснить тем, что ограничение 1.3.3 нарушается на оконечных системах во многих случаях, и обойти это ограничение позволяет именно процедура агрегации.

Таблица 5.5. Исследования свойств алгоритма на различных вариантах запуска

Свойства запускаемого алгоритма	Средний процент назначенных сообщений (%)			Среднее количество виртуальных каналов		
	1-ый класс данных	2-ой класс данных	3-ий класс данных	1-ый класс данных	2-ой класс данных	3-ий класс данных
Без процедуры агрегации	23	54	75	358	54	75
Без процедуры переконфигурации	100	99	75	459	48	72
Без процедуры ограниченного перебора	100	99	81	459	48	71
С ограниченным перебором глубины 1	100	99	81	459	48	71
С ограниченным перебором глубины 2	100	99	81	459	48	71

Таблица 5.6. Среднее время работы алгоритма на различных вариантах запуска

Свойства запускаемого алгоритма	Среднее время работы алгоритма (с)		
	1-ый класс данных	2-ой класс данных	3-ий класс данных
Без процедуры агрегации	2	0.05	1
Без процедуры переконфигурации	83	0.2	1
Без процедуры ограниченного перебора	85	0.2	10
С ограниченным перебором глубины 1	85	0.2	10
С ограниченным перебором глубины 2	85	0.2	10

- Процедура переконфигурации позволяет улучшить результаты для третьего класса данных. Более подробное исследование для данного класса данных приведено в разделе 5.3.4.
- Процедура ограниченного перебора не позволила улучшить результаты для проводимых тестов. Это можно пояснить тем, что в используемой топологии для каждого пути имеется мало альтернатив для построения и перестроения маршрутов.
- Время работы алгоритмов увеличивается при использовании процедуры ограниченного перебора только для третьего класса данных. Именно в этом случае происходит нарушение ограничений на пропускную способность и запуск процедуры.

5.3.3. Исследование процедуры агрегации

Как было показано в предыдущем разделе, процедура агрегации позволяет существенно улучшить количество назначаемых сообщений. В данном разделе проводится исследование эффективности процедуры агрегации и количества построенных виртуальных каналов в зависимости от исходного количества сообщений. Для этого проводилась серия запусков для первого класса данных (большое количество сообщений с низкими требованиями к длительности передачи) с исходным количеством сообщений, варьирующимся от 100 до 3000. Для исследования эффективности процедуры агрегации измерялось количество назначаемых сообщений с использованием/без использования

процедуры агрегации. Результаты проведенных исследований изображены на рисунках 5.5 и 5.6.

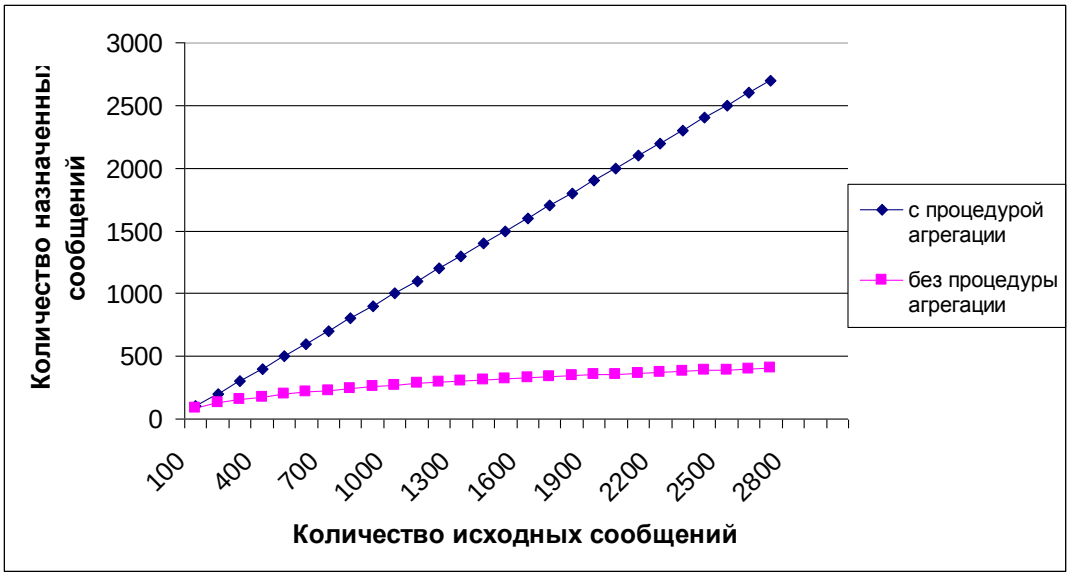

Рисунок 5.5. Эффективность процедуры агрегации.

Исходя из полученных результатов, можно сделать выводы, что процедура агрегации виртуальных каналов позволяет существенно улучшить результат. Для каждого класса данных эта процедура позволила значительно увеличить среднее число назначаемых сообщений. Также процедура агрегации позволяет сократить число виртуальных каналов. При назначении 2000 сообщений, например, число виртуальных каналов более чем в два раза меньше количества сообщений. Также, исходя из графика на рисунке 5.6, можно отметить, что начиная с некоторого количества сообщений (в данном случае около 1600), процедура агрегирует сообщения все в меньшее число виртуальных каналов (график начинает убывать). Это можно пояснить тем, что при большем количестве сообщений требования нарушаются все чаще, что требует большей степени агрегации сообщений в виртуальных каналах.

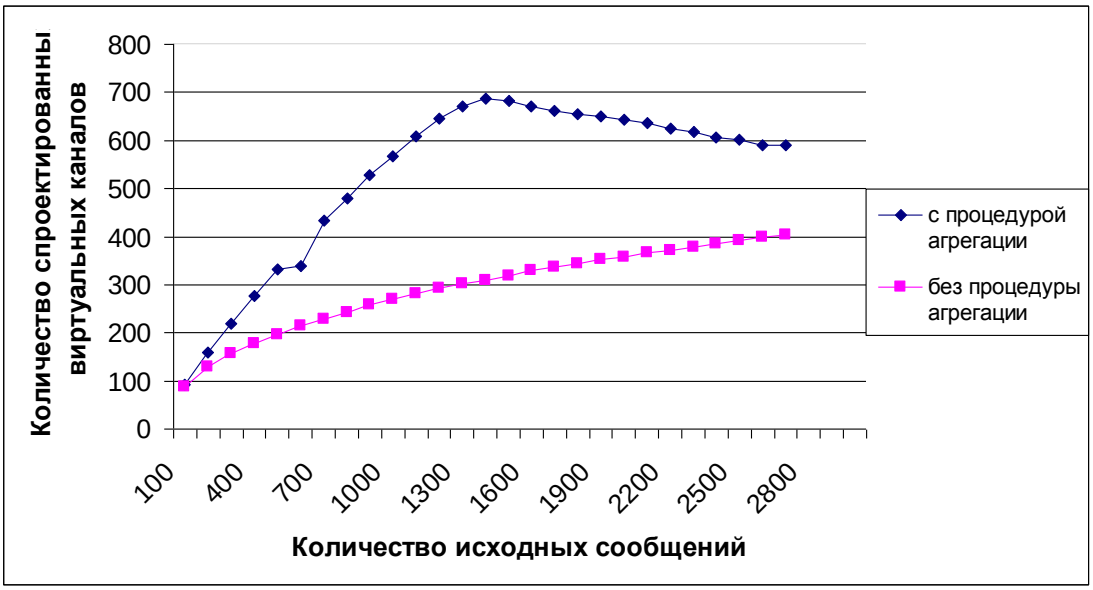

Рисунок 5.6. Количество спроектированных виртуальных каналов.

Кроме указанного исследования была сформулирована и проверена статистическая гипотеза согласно методу, описанному в разделе 5.1, с параметрами $n = 100$ и $\alpha = 0.05$ вида $E_a - D_a = m > 0$, где E_a и D_a – соответственно, количество назначенных сообщения с использованием и без использования процедуры агрегации. Данная гипотеза проверялась для 2000 сообщений. В результате проверки гипотеза подтвердилась при $m = 1642$. Данное значение подтверждает значительное улучшение при использовании процедуры.

5.3.4. Исследование работы алгоритма в случае высоких требований к длительности передачи сообщений

Согласно результатам, полученным в разделе 5.3.2, наиболее интересным для исследования является третий класс данных. В данном разделе описаны дополнительные исследования, проведенные в случае высоких требований к длительности передачи сообщений. Для исследования эффективности различных вариантов запуска алгоритма проводились сравнения следующих вариантов запуска алгоритма:

- С использованием алгоритмов построения маршрутов виртуальных каналов, основанных на алгоритме Дейкстры, либо на алгоритме поиска k кратчайших путей в графе. При этом использовалась процедура ограниченного перебора глубины 2.
- Без использования процедуры переконфигурации в случае невыполнения ограничений на длительность передачи сообщений, либо без проведения процедуры агрегации в процессе процедуры переконфигурации, либо с полностью

работающей процедурой переконфигурации. При этом использовалась процедура ограниченного перебора глубины 2.

- При количестве итераций процедуры конфигурации, равном 1, либо равном 5 (см. Шаг 6 алгоритма переконфигурации в разделе 4.2.6.). При этом использовалась процедура ограниченного перебора глубины 2.

Среднее количество назначенных сообщений и время работы алгоритмов для сравнения указанных свойств приведено в таблице 5.7.

Таблица 5.7. Сравнение свойств алгоритма при высоких требованиях к длительности передачи сообщений

Свойства запускаемого алгоритма	Средний процент назначаемых сообщений (%)	Среднее время работы алгоритма (с)
С алгоритмом Дейкстры	80	10
С алгоритмом *k* кратчайших путей	81	10
Без процедуры переконфигурации	75	0.3
Без процедуры агрегации в процедуре переконфигурации	81	3.5
С полной процедурой переконфигурации	81	10
С 1 итерацией процедуры переконфигурации	81	10
С 5 итерациями процедуры переконфигурации	81	10

Кроме указанных средних значений, были подтверждены следующие статистические гипотезы с параметрами $n = 100$ и $\alpha = 0.05$:

- При сравнении количества назначений при использовании алгоритма Дейкстры E_d и алгоритма *k* кратчайших путей E_k:

  $E_k - E_d = ½ > 0$ (при этом не подтверждалась гипотеза $E_k - E_d = 0$);

- При сравнении количества назначений без использования процедуры конфигурации E_w, без использования процедуры агрегации в процедуре переконфигурации E_a и с полным запуском процедуры E_r подтвердились следующие гипотезы:
  - $E_a - E_w = 6$;
  - $E_r - E_w = 6$;
  - $E_r - E_a = 0.2 > 0$ (при этом не подтверждалась гипотеза $E_r - E_a = 0$);
- При сравнении количества итераций (E_1 - одна итерация, E_5 – 5 итераций):

$E_5 - E_1 = 0$.

Таким образом, в случае высоких требований к длительности передачи сообщений можно сделать следующие выводы:

- Алгоритм *k* кратчайших путей позволяет получить незначительное преимущество перед алгоритмом Дейкстры.
- Процедура переконфигурации позволяет улучшить эффективность алгоритма, при этом достаточно одной итерации процедуры. Это можно объяснить тем, что уже на первой итерации оценки длительности передачи кадра может быть достаточно для переконфигурации виртуального канала, и дальнейшее уточнение оценки является незначительным и не позволяет улучшить результат.
- Процедура агрегации позволяет получить незначительное улучшение в количестве назначаемых сообщений (в среднем на 0.2 % для поставленных экспериментов согласно подтвержденной статистической гипотезе).
- Больше всего временных затрат приходится на процедуру агрегации, в то же время использование процедуры переконфигурации также существенно увеличивает длительность работы алгоритма, однако и позволяет существенно улучшить результаты.

5.3.5. Выводы

По проведенным экспериментальным исследованиям алгоритма построения сетей AFDX можно сделать следующие выводы:

- Наибольшего улучшения эффективности работы алгоритма позволяет достичь процедура агрегации, запускаемая при нарушении ограничения на максимальный джиттер виртуального канала. Кроме того, процедура позволяет уменьшить количество виртуальных каналов, и их количество уменьшается тем сильней, чем больше сообщений подается на вход алгоритму.
- Процедура ограниченного перебора не показала улучшения эффективности назначения сообщений для проведенных экспериментальных исследований. Причиной может быть малое количество альтернатив при построении маршрутов в исследуемой топологии сети AFDX.
- В случае высоких требований к длительности передачи сообщений эффективность может быть значительно повышена в результате использования процедуры переконфигурации. Также эффективность незначительно повышается за счет использования алгоритма *k* кратчайших путей вместо алгоритма Дейкстры и использования процедуры агрегации в процедуре конфигурации. При этом в

проведенных тестовых запусках было достаточно одной итерации процедуры агрегации.

6. Инструментальная система построения сетей AFDX

В ходе построения сетей AFDX возникают следующие задачи:

- Построение топологии сети.
- Построение конфигурации сети, включающее в себя построение виртуальных каналов и маршрутов к ним по заданному множеству сообщений.
- Моделирование, тестирование и верификация построенной конфигурации сети.
- Мониторинг работы сети при вводе системы в эксплуатацию.

Открытые и коммерческие системы [56-60] для автоматизации проектирования бортовых систем, включающие в себя проектирование сетей AFDX, ориентированы на сети с заданной конфигурацией, предварительно построенной вручную. Такие системы позволяют проводить моделирование, тестирование и верификацию конфигурации бортовых систем, а также мониторинг работы реальной бортовой системы. В работах [56-58] описаны инструментальные средства, которые по детальному описанию характеристик бортовых систем на определенном языке моделирования (AADL [56], UPPAAL [57], SPIN [58]) позволяют проверять систему на корректность в плане выполнения заданных ограничений, в том числе, например, ограничений на длительность передачи сообщений в сетях AFDX. Коммерческие системы (EC Comp GmbH [59], MHZ Solutions [60]), исходя из описаний, также предоставляют средства для верификации и тестирования и позволяют проводить мониторинг работы реальной системы.

В отличие от указанных систем поддержки проектирования бортовых систем, реализованное инструментальное средство позволяет по набору входных данных и заданным ограничениям производить построение корректных конфигураций. Кроме того, инструментальная система позволяет задавать конфигурации вручную, а также изменять автоматически спроектированные конфигурации с возможностью их проверки на выполнение ограничений реального времени. Следует отметить, что система строит корректные конфигурации, для которых проводить верификацию не нужно. Схожих инструментальных систем автоматического построения корректных конфигураций для сетей AFDX в открытом доступе найдено не было.

6.1. Требования к системе

На основе описанного алгоритма построения сетей AFDX разработана инструментальная система [61, 61] согласно следующим требованиям:

- Система должна предоставлять графический интерфейс для задания и редактирования входных данных, а именно графа и параметров сети, сообщений и их свойств согласно модели входных данных;
- Система должна позволять запускать алгоритм построения сети AFDX согласно заданным входным данным с заданными параметрами работы алгоритма;
- Система должна иметь графический интерфейс для отображения результатов алгоритма, а именно для отображения параметров и маршрутов построенных виртуальных каналов;
- Система должна позволять задавать виртуальные каналы вручную, а также изменять вручную параметры спроектированных виртуальных каналов;
- Реализованный алгоритм должен иметь модульную структуру и позволять заменять используемые процедуры, а также запускать алгоритм без использования отдельных процедур, таких как процедура ограниченного перебора, процедура агрегации и процедура переконфигурации виртуальных каналов.

6.2. Описание системы

Реализованная инструментальная система состоит из двух основных модулей: модуля редактирования (пользовательский интерфейс) и модуля построения виртуальных каналов.

6.2.1. Пользовательский интерфейс

Модуль редактирования реализует интерфейс для конечного пользователя для задания входных данных и отображения результатов алгоритма. Данный модуль реализован на языке python версии 2.7 с использованием библиотеки PYQT4 и состоит из следующих подмодулей:

- Подмодуль редактирования сети AFDX – позволяет интерактивно задавать и отображать граф сети и задавать различные параметры, такие как пропускная способность каналов передачи данных, стратегии буферизации на выходных портах коммутаторов.
- Подмодуль редактирования сообщений – позволяет задавать множество передаваемых сообщений и их свойства: *size*, T, τ, J, отправителя и получателей сообщения. Также можно указывать существующий виртуальный канал для передачи.
- Подмодуль редактирования виртуальных каналов – позволяет задавать вручную, редактировать и отображать существующие виртуальные каналы и их свойства:

оконечную систему-отправителя, множество оконечных систем-получателей, *BAG*, *LM*, маршрут передачи кадров в сети. Маршрут передачи также отображается на графе сети AFDX.

- Подмодуль внешнего интерфейса – используется для передачи заданных пользователем входных данных модулю построения виртуальных каналов и получения результатов алгоритма. Для передачи данных генерируется файл в формате xml с описанием сети и входных данных. После отработки алгоритма файл с описанием построенных виртуальных каналов в формате xml передается обратно и после обработки отображается с помощью подмодуля редактирования виртуальных каналов. В данном подмодуле также можно запускать отдельные процедуры алгоритма, такие как проверка заданной конфигурации на корректность (в плане проверки выполнения ограничений (1.3.1) - (1.3.4)), вычисление длительности передачи кадров для заданных вручную виртуальных каналов.

Интерфейс описанного модуля изображен на рисунке 6.1.

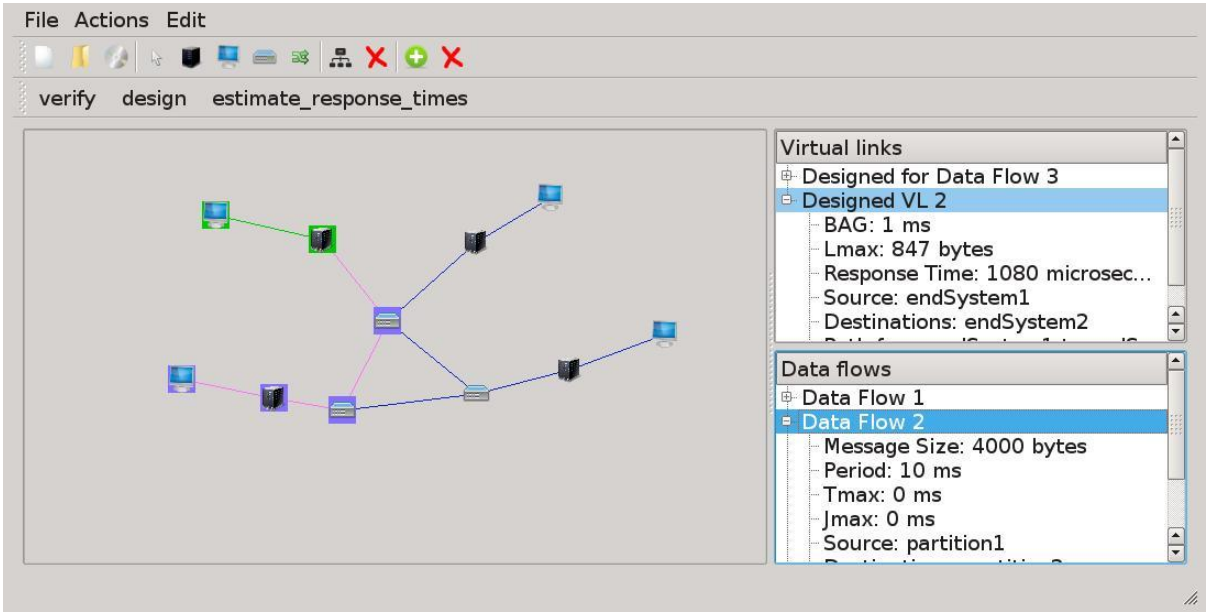

Рисунок 6.1. Пользовательский интерфейс разработанной инструментальной системы.

6.2.2. Модуль построения виртуальных каналов

Модуль построения виртуальных каналов реализует описанную в работе общую схему алгоритма построения сетей AFDX на основе ряда подключаемых процедур. При реализации модуля использовался язык программирования C++ с использованием библиотек STL, QT4. Кроме того, использовалась утилита cmake для обеспечения

возможности запуска инструментальной системы независимо от используемой операционной системы.

В основе модуля лежит подмодуль представления модели входных данных задачи. Модель входных данных представляет собой набор сущностей, соответствующих математической модели из постановки задачи, таких как сообщение, виртуальный канал, оконечная система, коммутатор и др.

В модуле реализован ряд процедур, которые используют представление модели задачи для задания входных данных.

- Процедура конфигурации параметров виртуального канала. Данная процедура предназначена для определения параметров *BAG*, *LM* и *JM* для виртуального канала, по которому передается только одно сообщение.
- Процедура агрегации виртуальных каналов.
- Процедура построения маршрутов виртуальных каналов.
- Процедура ограниченного перебора, вызываемая в случае, когда невозможно построить маршрут виртуального канала.
- Процедура вычисления оценки длительности передачи кадров и сообщений. Система позволяет реализовывать и подключать любой из известных методов оценки длительности передачи кадров, для чего необходимо адаптировать модель представления входных данных под модель, используемую тем или иным средством вычисления оценки.
- Процедура переконфигурации виртуальных каналов, выполняющаяся в случае нарушения ограничений на длительность передачи кадров.

Кроме описанных процедур, в модуле имеется внешний интерфейс, который позволяет принимать на вход файлы в формате xml с описанием входных данных и преобразовывать их во внутреннее представление и возвращать файлы с описанием построенных виртуальных каналов.

Запуск алгоритма производится из командной строки путем задания пути к файлу с описанием входных данных задачи и задания дополнительных параметров. С помощью параметров командной строки можно как включать/отключать использование определенных процедур в процессе решения задачи (таких как процедура агрегации, процедура ограниченного перебора, процедура переконфигурации), так и задавать другие параметры, такие как глубина ограниченного перебора или количество итераций процедуры переконфигурации. Кроме того, существует возможность запускать отдельные этапы алгоритма, такие как проверка входных данных на корректность в плане

выполнения ограничений (1.3.1) – (1.3.4) или вычисление длительности передачи кадров для заданных виртуальных каналов.

6.3. Выводы

Описанная инструментальная система построения сетей AFDX обладает следующими свойствами:

- Предоставляет графический интерфейс для задания и редактирования входных данных; графический интерфейс представляет собой систему интерактивного редактирования графа сети AFDX с возможностью задания всех необходимых параметров;
- Позволяет запускать алгоритм построения сети AFDX для заданных входных данных и значений параметров алгоритма; запуск алгоритма производится путем сохранения входных данных задачи в файле формата xml и передаче этого файла на вход модулю построения виртуальных каналов с указанием необходимых параметров запускаемых процедур;
- Имеет графический интерфейс для отображения результатов работы алгоритма; данные отображаются непосредственно после получения результата от модуля построения виртуальных каналов;
- Позволяет задавать виртуальные каналы вручную, а также изменять вручную значения параметров спроектированных виртуальных каналов, данное свойство поддерживается за счет возможности интерактивного редактирования сети одновременно с возможностью отображения результатов работы алгоритма;
- Алгоритм имеет модульную структуру и позволяет заменять используемые процедуры с помощью механизма наследования, а также запускать алгоритм без использования отдельных процедур. Данное свойство позволяет производить настройку инструментальной системы на особенности конкретной задачи путем настройки отдельных модулей. Также данное свойство активно использовалось при проведении экспериментальных исследований свойств алгоритма.

Описанные свойства показывают, что реализованная инструментальная система удовлетворяет сформулированным требованиям к инструментальной системе построения сетей AFDX.

Заключение

Основные результаты диссертационной работы заключаются в следующем:

1. Построен алгоритм согласованного отображения множества запросов на физические ресурсы центров обработки данных, позволяющий выбирать соотношение между вычислительной сложностью алгоритма и подмножеством размещенных запросов. Получено аналитическое выражение для вычислительной сложности алгоритма в зависимости от глубины ограниченного перебора, обоснована корректность алгоритма, сформулированы условия, при которых алгоритм размещает все запросы.
2. Разработан алгоритм построения соединений для сетей со статически конфигурируемыми коммутаторами с соблюдением ограничений на качество работы сети. Получено аналитическое выражение для вычислительной сложности алгоритма в зависимости от глубины ограниченного перебора, обоснована корректность алгоритма, сформулированы условия, при которых алгоритм строит все соединения.
3. На основе предложенного алгоритма построения соединений для сетей, основанных на стандарте Avionics Full Duplex Ethernet, разработано программное инструментальное средство, предоставляющее возможность запуска алгоритма с различными параметрами используемых процедур и возможность модульной настройки на особенности конкретной задачи.
4. Для построенных алгоритмов проведен экспериментальный анализ влияния жадный критериев и используемых процедур на качество решения. Выявлены наилучшие критерии в зависимости от класса исходных данных, показано улучшение качества получаемого решения при использовании процедуры агрегации соединений и процедуры репликации.

Дальнейшие исследования могут быть связаны со следующими направлениями:

1. Задача планирования вычислений в центрах обработки данных с моделью обслуживания Infrustructure-as-a-Service может быть расширена путем введения необходимости построения планов миграций виртуальных машин с одного вычислительного узла на другой. Миграция позволяет существенно уменьшить возникающую фрагментацию ресурсов физических элементов, однако является дорогостоящей операцией в связи с необходимостью передачи большого количества данных через сеть обмена данными. Построение планов миграций с

учетом ограничений на их максимальную длительность является одним из направлений возможных дальнейших исследований.

2. Задача планирования вычислений в центрах обработки данных с моделью обслуживания Infrustructure-as-a-Service также может быть изменена путем введения расширенной модели физических систем хранения данных. Системы хранения данных в настоящее время представляют собой распределенные системы, которые могут включать в себя специальные контроллеры, которые предоставляют виртуализированный доступ к данным, распределенным между хранилищами различных типов. От типа зависит максимальный объем хранимых данных и скорость чтения/записи данных в хранилище. Задача оптимизации распределения данных по хранилищам в таких системах с учетом заданным требованиям к качеству обслуживания может быть актуальна для оптимизации работы всего ЦОД.
3. Задача организации виртуальных каналов в сетях на основе стандарта Avionics Full Duplex Ethernet может быть расширена путем введения более общей модели передаваемых через сеть сообщений. Вместо введения периода и джиттера для сообщений в современных системах реального времени могут задаваться директивные интервалы, в течение которых может быть выдан очередной экземпляр сообщения, и интервалы, к течение которых сообщение должно быть доставлено до абонентов-получателей. Введение директивных интервалов позволит расширить класс решаемых задач, возникающих при проектировании бортовых систем.

Литература

1. Amies A, Sluiman H., Tong Q.G., et al. Developing and Hosting Applications on the Cloud. Boston: IBM Press, 2012.
2. Aircraft Data Network. Part 7. Avionics Full Duplex Switched Ethernet (AFDX) Network // Aeronautical Radio, Inc. 2012.
3. Wustenhoff E. Service level agreement in the data center. Sun BluePrints, Sun Microsystems Professional Series, 2002.
4. Pepple K. Deploying OpenStack. Sebastopol: O'Reilly, 2011.
5. V. Balashov, V. Kostenko, P. Vdovin, R. Smeliansky, A. Shalimov. An Analysis of Approaches to Onboard Networks Design // Proceedings of the International Science and Technology Conference Modern Networking Technologies (MoNeTec), IEEE. Moscow, Russia, MAKS Press, 2014. P. 20–24.
6. Alena R. L. et al. Communications for integrated modular avionics //Aerospace Conference, 2007 IEEE. IEEE, 2007. P. 1-18.
7. Гончар Д.Р., Фуругян М.Г. Алгоритмы составления многопроцессорного расписания для неоднородного множества работ с директивными интервалами и произвольными процессорами // Системы управления и информационные технологии. 2013. Т. 3, № 1. С. 204-208.
8. Гончар Д.Р., Фуругян М.Г. Алгоритмы планирования вычислений в многопроцессорных системах с неоднородным множеством работ и директивными интервалами // Системы управления и информационные технологии. 2011. № 3. С.8-12.
9. Потехин П. А., Емельянов Д. М., Топорков В. В. Формирование и планирование пакетов заданий в распределенных вычислительных средах // Вестн. ЮУрГУ. Сер. Выч. матем. информ. 2015. Т. 4. №. 2. С. 44-57.
10. Топорков В. В., Бобченков А. В., Емельянов Д. М., Целищев А. С. Методы и эвристики планирования в распределенных вычислениях с неотчуждаемыми ресурсами // Вестн. ЮУрГУ. Сер. Выч. матем. информ. 2014. Т. 3. №. 2. С. 43-62.
11. Костенко В.А., Плакунов А.В. Алгоритм построения одноприборных расписаний, основанный на схеме муравьиных колоний // Изв. РАН. ТиСУ. 2013. № 6. С.87-96; Kostenko V.A., Plakunov A.V. An Algorithm for Constructing Single Machine Schedules Based on Ant Colony Approach // J. of Computer and Systems Sciences Intern. 2013. V.52. № 6. P. 928-937.

12. Костенко В.А. Алгоритмы построения расписаний для вычислительных систем реального времени, допускающие использование имитационных моделей // Программирование. 2013. №5. С. 53-71.
13. Костенко В.А., Шестов П.Е. Жадный алгоритм совместного планирования вычислений и обменов в системах реального времени // Изв. РАН. ТиСУ. 2012. № 5. С.35-49; Kostenko V.A., Shestov P.E. A Greedy Algorithm for Combined Scheduling of Computations and Data Exchanges in Real_Time Systems // J. of Computer and Systems Sciences Intern. 2012. V.51. № 5. P.648-662.
14. Зорин Д.А., Костенко В.А. Алгоритм синтеза архитектуры вычислительной системы реального времени с учетом требований к надежности // Изв. РАН. ТиСУ. 2012. № 3. С.76–83; Zorin D.A., Kostenko V.A. Algorithm for Synthesis of Real-Time Systems under Reliability Constraints // J. of Computer and Systems Sciences Intern. 2012. V.51. № 3. P. 410-417.
15. Vdovin, P. M., Zotov, I. A., Kostenko, V. A., Plakunov, A. V. Congestion Elimination on Data Storages Network Interfaces in Datacenters // Parallel Computing Technologies. – Springer International Publishing, 2015. – P. 298-303.
16. Вдовин П.М, Зотов И.А., Костенко В.А., Плакунов А.В., Смелянский Р.Л. Задача распределения ресурсов центров обработки данных и подходы к ее решению // VII Московская междунар. конф. по исследованию операций (ORM2013). М.: ВЦ РАН, 2013. Т.2. С.30-32.
17. Вдовин П.М, Зотов И.А, Костенко В.А., Плакунов А.В., Смелянский Р.Л. Сравнение различных подходов к распределению ресурсов в центрах обработки данных // Изв. РАН. ТиСУ. 2014. № 4. С. 71-83; Vdovin P.M., Zotov I.A., Kostenko V.A., Plakunov A.V., Smelyansky R.L. Comparing Various Approaches to Resource Allocating in Data Centers // J. of Computer and Systems Sciences Intern. 2014. V.53. № 4. P. 689-701.
18. Вдовин П.М, Костенко В.А., Алгоритм распределения ресурсов в центрах обработки данных с раздельными планировщиками для различных типов ресурсов // Изв. РАН. ТиСУ. 2014. № 6, С.80-93; Vdovin P.M., Kostenko V.A. Algorithm for Resource Allocation in Data Centers with Independent Schedulers for Different Types of Resources // J. of Computer and Systems Sciences Intern. 2014. V. 53. № 6. pp. 854–866.
19. Костенко В.А., Гурьянов Е.С. Алгоритм построения расписаний обменов по шине с централизованным управлением и исследование его эффективности // Программирование. 2005. №6. С.67-76; V. A. Kostenko and E. S. Gury'anov An Algorithm for Scheduling Exchanges over a Bus with Centralized Control and an Analysis of Its Efficiency // Programming and Computer Software. 2005. V. 31. N. 6. P. 340–346.

20. Вдовин П.М., Костенко В.А. Исследование эффективности процедуры агрегации виртуальных каналов при построении бортовых коммутируемых сетей // Вестник Московского университета. Серия 15: Выч. мат. и киб. 2015. № 4. С. 32-40; Vdovin P.M., Kostenko V.A. Studying the Effectiveness of an Aggregation Procedure for Virtual Links in Constructing Onboard Switched Networks // Moscow University Computational Mathematics and Cybernetics. 2015. V.39. N. 4. P. 184-192.
21. Jiang J. W. et al. Joint VM placement and routing for data center traffic engineering // INFOCOM, 2012 Proceedings IEEE. IEEE. 2012. P. 2876-2880.
22. Singh A., Korupolu M., Mohapatra D. Server-storage virtualization: integration and load balancing in data centers // Proceedings of the 2008 ACM/IEEE conference on Supercomputing. IEEE Press, 2008. P. 53.
23. Korupolu M., Singh A., Bamba B. Coupled placement in modern data centers //Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE, 2009. P. 1-12.
24. Cheng X. et al. Virtual network embedding through topology-aware node ranking // ACM SIGCOMM Computer Communication Review. 2011. T. 41. N. 2. P. 38-47.
25. Chowdhury N. M. M. K., Rahman M. R., Boutaba R. Virtual network embedding with coordinated node and link mapping // INFOCOM 2009, IEEE. IEEE, 2009. P. 783-791.
26. Zhu Y., Ammar M. H. Algorithms for Assigning Substrate Network Resources to Virtual Network Components // INFOCOM. 2006. P. 1-12.
27. J. Lischka, H. Karl, A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection // Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, 2009.
28. Botero J.F., Hesselbach X., Fischer A. et al Optimal Mapping of Virtual Networks with Hidden Hops // Telecommunication Systems. 2012. V.51. №4. P.273-282.
29. Yu M. et al. Rethinking virtual network embedding: substrate support for path splitting and migration // ACM SIGCOMM Computer Communication Review. 2008. T. 38. N. 2. P. 17-29.
30. Urgaonkar B., Rosenberg A. L., Shenoy P. Application placement on a cluster of servers // International Journal of Foundations of Computer Science. 2007. T. 18. N. 5. P. 1023-1041.
31. Doina Bein, Wolfgang Bein, Swathi Venigella Cloud Storage and Online Bin Packing // Intelligent Distributed Computing V, 2012.
32. S. Nagendram, J.Vijaya Lakshmi, D.Venkata Narashimha Rao , Ch.Naga Jyothi Efficient Resource Scheduling in Data Centers using MRIS // Indian Journal of Computer Science and Engineering. 2011. V. 2. N. 5. P. 764-769.

33. Arzuaga E., Kaeli D. R. Quantifying load imbalance on virtualized enterprise servers // Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering. ACM, 2010. P. 235-242.
34. Mishra M., Sahoo A. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach // Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011. P. 275-282.
35. W. Szeto, Y. Iraqi and R. Boutaba A Multi-Commodity Flow Based Approach to Virtual Network Resource Allocation // School of Computer Science, University of Waterloo, 2003.
36. R.K. Ahuja, T.L. Magnanti, J.B. Orli. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, 1993.
37. Coffman E.G., Garey M.R., Johnson D.S. Approximation algorithms for bin packing: A survey // Approximation algorithms for NP-hard problems. Boston: PWS Publishing Co., 1996. P. 46-93.
38. Ma Q., Steenkiste P. On path selection for traffic with bandwidth guarantees // Network protocols, 1997. proceedings., 1997 international conference on. IEEE, 1997. P. 191-202.
39. Al Sheikh A. et al. Optimal design of virtual links in AFDX networks // Real-Time Systems. 2013. T. 49. N. 3. P. 308-336.
40. Wang B., Hou J. C. Multicast routing and its QoS extension: problems, algorithms, and protocols // Network, IEEE. 2000. T. 14. N. 1. P. 22-36.
41. Seok Y. et al. Explicit multicast routing algorithms for constrained traffic engineering // Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on. IEEE, 2002. P. 455-461.
42. Frances F., Fraboul C., Grieu J. Using network calculus to optimize the AFDX network. 2006.
43. An D. et al. A feasible configuration of AFDX networks for real-time flows in avionics systems. 2013.
44. Cha Y. J., Kim K. I. Heuristic Algorithm for Virtual Link Configuration in AFDX Networks. 2014.
45. Gutiérrez J. J., Palencia J. C., Harbour M. G. Response time analysis in AFDX networks with sub-virtual links and prioritized switches // XV Jornadas de Tiempo Real. 2012.
46. Scharbarg J. L., Fraboul C. Methods and tools for the temporal analysis of avionic networks // New trends in technologies: control, management, computational intelligence and network systems. 2010. P. 413-438.
47. Guangyu H. et al. Towards Tightening Worst Case End-to-End Delay Estimates for AFDX Network. 2011.

48. Boyer M., Fraboul C. Tightening end to end delay upper bound for afdx network calculus with rate latency fifo servers using network calculus // Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on. IEEE, 2008. P. 11-20.
49. Bauer H., Scharbarg J. L., Fraboul C. Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network // Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on. IEEE, 2009. P. 1-8.
50. Bauer H., Scharbarg J. L., Fraboul C. Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources // Real-time systems. 2012. T. 48. N. 1. P. 101-133.
51. Yen J. Y. Finding the k shortest loopless paths in a network // Management Science. 1971. T. 17. N. 11. P. 712-716.
52. Hermenier, Fabien, Sophie Demassey, and Xavier Lorca. Bin repacking scheduling in virtualized datacenters // Principles and Practice of Constraint Programming–CP. 2011. P.27-41.
53. M. Al-Fares, A. Loukissas, and A. A. Vahdat Commodity data center network architecture // ACM SIGCOMM Comput. Commun. 2008. T.38. N. 4. P. 63–74.
54. Гмурман В. Е. Теория вероятностей и математическая статистика. М.: Высшая школа, 2003. 479 с.
55. Калинина В. Н., Панкин В. Ф. Математическая статистика. М.: Высшая школа, 1998. 336 с.
56. Буздалов, Д. В., Зеленов, С. В., Корныхин, Е. В. И др. Инструментальные средства проектирования систем интегрированной модульной авионики // Труды ИСП РАН. 2014. Т. 26. № 1. С. 201-230.
57. Anand M. et al. Formal modeling and analysis of the AFDX frame management design // Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. Ninth IEEE International Symposium on. IEEE, 2006. P. 7.
58. Saha I., Roy S. A finite state modeling of AFDX frame management using spin // Formal Methods: Applications and Technology. Springer Berlin Heidelberg, 2007. P. 227-243.
59. EC Comp GmbH: [сайт]. URL: http://eccomp.de/en/afdxsuite-en.html (дата обращения: 06.01.2016).
60. MHZ Solutions: [сайт]. URL: http:// http://www.mhz-avionics.com (дата обращения: 06.01.2016).
61. Вдовин П.М. Инструментальная система проектирования сетей AFDX // ТРУДЫ МФТИ. 2015. Т. 7. № 2. С.131 – 136.

62. Вдовин П.М., Костенко В.А. Свидетельство о государственной регистрации программы для ЭВМ №2015661760 «Программа построения виртуальных каналов в коммутируемых сетях» от 09.11.2015.
63. Вдовин П.М., Балашов В.В., Костенко В.А. Управление AFDX-сетями и задачи их конфигурирования // Программные системы и инструменты. Тематический сборник No 13. М.: Изд-во факультета ВМиК МГУ. – 2013. – С. 30-47.
64. Bokhari S. H. On the mapping problem // IEEE Transactions on Computers. 1981. T. 100. N. 3. P. 207-214.
65. Ucar B. et al. Task assignment in heterogeneous computing systems // Journal of parallel and Distributed Computing. 2006. T. 66. N. 1. P. 32-46.
66. Ercal F., Ramanujam J., Sadayappan P. Task allocation onto a hypercube by recursive mincut bipartitioning // Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues-Volume 1. ACM, 1988. P. 210-221.
67. Jose A. An approach to mapping parallel programs on hypercube multiprocessors //Parallel and Distributed Processing, 1999. PDP'99. Proceedings of the Seventh Euromicro Workshop on. IEEE, 1999. P. 221-225.
68. Kafil M., Ahmad I. Optimal task assignment in heterogeneous computing systems // Heterogeneous Computing Workshop, 1997. Proceedings, Sixth. IEEE, 1997. P. 135-146.
69. Курносов М. Г. Алгоритмы вложения параллельных программ в иерархические распределённые вычислительные системы // Вестник СибГУТИ. 2009. №. 2. С. 6.
70. Ложкин С. А., Мин Л. Д. О некоторых оптимальных вложениях двоичных и троичных деревьев в плоские прямоугольные решетки // Вестник Московского университета. Серия 15: Вычислительная математика и кибернетика. 1995. № 4. С. 49–55.

Приложение А. Корректность процедуры агрегации при построении сетей AFDX

Процедура агрегации используется для организации передачи нескольких сообщений в одном виртуальном канале для сетей AFDX и описана в разделе 4.2.2.2. . В данном разделе доказывается корректность указанной процедуры.

Будем говорить, что виртуальный канал *vl* с параметрами LM_{vl}, BAG_{vl} и JM_{vl}, по которому передается множество сообщений $MSG=\{msg\}$, является корректным, если для него выполнены следующие свойства:

1. Ограничения, налагаемые исходной задачей:

- Частота передачи кадров каждого виртуального канала не превосходит частоты выдачи кадров в канал:

$$\sum_{msg \in MSG_{vl}} \left(\left\lceil size_{msg} / (LM_{vl} - c) \right\rceil / T_{msg} \right) \le 1 / BAG_{vl} \qquad \text{(A.1)}$$

2. Ограничения, налагаемые стандартом Ethernet и ограничивающие максимальный размер кадра:

$$LM_{vl} \le 1518 \qquad \text{(A.2).}$$

Остальные ограничения: на пропускную способность каналов передачи данных, на максимальный джиттер кадров и на длительность передачи сообщений проверяются после проведения процедуры агрегации, поэтому не рассматриваются в данном определении.

Утверждение. Виртуальный канал, сформированный согласно процедурой агрегации, является корректным (в смысле выполнения ограничений (А.1) – (А.2)).

Доказательство. Ограничения (А.1) выполняются при построении виртуального канала, т.к. значение *N* максимального количества кадров, на которые делится сообщение, выбирается с учетом заданных ограничений (см. ограничения (4.2.5)-(4.2.6) в описании алгоритма).

Осталось доказать, что $LM_{vl} \le 1518$. В процедуре агрегации LM_{vl} вычисляется в зависимости от *N* согласно жадной процедуре, описанной в разделе 4.2.2.2. . Значение $LM_{vl}(N)$ возвращает максимальный размер кадра с учетом получившихся разбиения сообщений на кадры:

$$LM(N) = \max_{msg} \left(c + \left\lceil size_{msg} / F_{msg}^{N} \right\rceil \right)$$

Соответственно, чтобы доказать, что $LM_{vl}(N) \le 1471$, надо показать, что

$\forall msg \in MSG : \left(c + \left\lceil size_{msg} / F_{msg}^{N} \right\rceil \right) \le 1518$, или (при условии, что c=47 байт)

$\forall msg \in MSG : \left\lceil size_{msg} / F_{msg}^{N} \right\rceil \le 1471,$

где F_{msg}^{N} - количество кадров, на которое разбивается сообщение *msg* при условии разбиения на равные части. Указанную формулу можно переформулировать следующим образом:

$$\forall msg \in MSG : size_{msg} / F_{msg}^{N} \le \left| size_{msg} / F_{msg}^{N} \right| \le 1471, \text{ или}$$

$$\forall msg \in MSG : F_{msg}^{N} \ge size_{msg} / 1471 \qquad \text{(A.3)}$$

Так как в описанном алгоритме на первой итерации принимается $\forall msg : F_{msg}^{N_0} = 1$, $\sum_{msg} F_{msg}^{N_0} = N_0$ и при этом алгоритм выполняется N-N_0 шагов, на каждом из которых ровно одно из значений F_{msg}^{k} увеличивается на 1, то верно:

$$\sum_{msg} F_{msg}^{N} = N .$$

Суммируя неравенство (A.3) по всем *msg*, получаем:

$N = \sum_{msg} F_{msg}^{N} \ge \sum_{msg} \frac{size_{msg}}{1471}$, что верно согласно выполнению условия (4.2.7).

Приложение Б. Доказательство утверждения о точности процедуры построения маршрутов в сетях AFDX

Утверждение 4.2. Пусть все каналы передачи данных имеют равную пропускную способность R, а пропускная способность, требуемая для построения маршрута виртуального канала, не превосходит $R_{\max}<R$. Пусть при построении маршрутов число каналов передачи данных, зарезервированная пропускная способность которых превысила $R - R_{\max}$, равно l. Тогда отношение суммарной требуемой пропускной способности маршрутов, построенных жадным алгоритмом, к суммарной пропускной способности при построении маршрутов оптимальным алгоритмом, не меньше $\frac{R - R_{\max}}{R + (l-1) * R_{\max}}$.

Доказательство. Пусть жадным алгоритмом *не* удалось построить маршруты для множества виртуальных каналов VL, при этом суммарная пропускная способность для виртуальных каналов с построенными маршрутами равно R_a. Для каждого из виртуальных каналов vl с требуемой пропускной способность R_{vl}, для которых не построен маршрут, существует хотя бы один канал передачи данных со свободной пропускной способностью, меньшей R_{vl}:

$$\forall vl \in VL : \exists link : R_{link} > R - R_{vl} \geq R - R_{\max},$$

где R_{link} – зарезервированная пропускная способность канала передачи данных $link$. В данной формуле используется тот факт, что $R_{vl} \leq R_{\max}$.

Пусть множество $Link$ – множество каналов передачи данных, зарезервированная пропускная способность которых превысила R-$R_{\max}$. По условию утверждения мощность $Link$ равно l.

Так как каждый из виртуальных каналов, для которых маршрут не построен, в оптимальном случае должен пройти через хотя бы один виртуальный канал из множества $Link$, то суммарная пропускная способность R_o виртуальных каналов, назначенных оптимальным алгоритмом, превосходит R_a не больше, чем на величину l*$R_{\max}$. Для отношения $\frac{R_a}{R_o}$, таким образом, верны следующие выкладки:

$$\frac{R_a}{R_o} \geq \frac{R_a}{R_a + l * R_{\max}} = 1 - \frac{l * R_{\max}}{R_a + l * R_{\max}}.$$

Так как для всех каналов передачи данных $link$ из множества $Link$ верно: $R_{link} > R - R_{\max}$, то построен маршрут для хотя бы одного виртуального канала, причем

суммарная пропускная способность виртуальных каналов с построенным маршрутом больше, чем $R - R_{\max}$:

$$R_a > R - R_{\max} .$$

Из полученной оценки следует:

$$\frac{R_a}{R_o} \geq 1 - \frac{l * R_{\max}}{R_a + l * R_{\max}} > 1 - \frac{l * R_{\max}}{R - R_{\max} + l * R_{\max}} = \frac{R - R_{\max}}{R + (l-1)R_{\max}} .$$

Приложение В. Доказательство утверждения о достаточных условиях выполнения ограничений на длительность передачи сообщений в сетях AFDX

Утверждение 4.3 (достаточные условия выполнения ограничений на длительность передачи сообщений). Пусть для некоторой сети AFDX, по которой передается множество сообщений *MSG* по множеству виртуальных каналов *VL*, верны следующие условия:

- по виртуальному каналу *vl* передается множество сообщений MSG_{vl};
- расстояние без циклов между любыми двумя оконечными системами не превышает *k* коммутаторов;
- для любого сообщения[19] *msg* верно $J_{msg} = 0$;
- все коммутаторы сети имеют стратегию буферизацию FIFO на выходных портах;
- τ_k – наибольшее время обработки кадра в коммутаторе перед постановкой в очередь.

Тогда, если для каждого сообщения *msg*, передающегося по виртуальному каналу *vl*, верно:

$$\left(\sum_{m \in MSG_{vl}} \left(\left\lceil size_m / (LM_{vl} - c) \right\rceil - 1\right) BAG_{vl}\right) + (k+1) LM_{vl} / R + k \cdot \tau_k + 2k \sum_{v \in VL} (LM_v / R + gap) \le \tau_{msg},$$

то для всех сообщений будет выполняться ограничение на длительность передачи.

Доказательство. Пусть сообщение *msg* передается по виртуальному каналу *vl*. Передача сообщения заключается во времени ожидании выдачи его последнего кадра δ и передачи этого кадра по сети Δ. Для выполнения ограничения на длительность передачи необходимо, таким образом, выполнение следующего неравенства:

$$\delta + \Delta \le \tau_{msg}.$$

Время ожидания выдачи последнего кадра определяется следующим образом:

$$\delta = (N-1) BAG_{vl},$$

здесь N – максимальное количество кадров в очереди выдачи кадров данного виртуального канала, которое определяется как

$$N = \sum_{msg} \left\lceil size_{msg} / (LM_{vl} - c) \right\rceil,$$

[19] В данном утверждении для упрощения рассматривается случай, когда не задано ограничение на джиттер порождения сообщения. Утверждение может быть расширено с учетом этого ограничения.

здесь c – размер заголовка кадра, суммирование ведется по всем сообщениям, передающимся по виртуальному каналу *vl*.

Оценка длительности передачи кадра Δ складывается из длительности передачи кадра по каналам передачи данных L и времени ожидания в коммутаторах, складывающегося из некоторого фиксированного времени обработки кадра D и времени ожидания в очереди на выходных портах коммутаторов B:

$$\Delta \le L + D + B.$$

Длительность передачи по каналам равна количеству каналов на время передачи кадра по ним:

$$L \le (k+1) \cdot LM_{vl} / R.$$

Обработка кадра в каждом коммутаторе не превосходит некоторой фиксированной величины τ_k:

$$D \le k \cdot \tau_k .$$

Для оценки величины B учитывается тот факт, что при работе алгоритма суммарная зарезервированная пропускная способность виртуальных каналов на каждом из каналов передачи данных не превосходит пропускной способности этого канала. При выполнении данного факта очередь на любом выходном порту коммутатора никогда не будет превосходить величины $2\sum_{vl} LM_{vl} / R$, где суммирование ведется по всем виртуальным каналам, проходящим через данный порт. Данная формула поясняется тем, что в наихудшем случае в один момент времени на вход может прибыть не более двух кадров каждого виртуального канала[20] (при возникновении джиттера пиковая пропускная способность виртуальных каналов может превосходить R, поэтому в данном случае берется 2 кадра). После этого очередь будет уменьшаться быстрее, чем увеличиваться (иначе нарушается ограничение на пропускную способность канала).

Таким образом, величина B не превосходит следующего значения:

$$B \le 2k \sum_{v \in VL} (LM_v / R + gap),$$

где сумма берется по всем построенным виртуальным каналам.

---

[20] Данное предположение позволяет получить очень грубую оценку, т.к., например, с одного физического канала передачи данных не может поступить одновременно более одного кадра, т.к. кадры следуют друг за другом. Для получения более точных оценок в алгоритме используются специальные методы, и предполагается, что получаемая оценка длительности передачи кадров, более точная, чем используемая в этом доказательстве.

Суммируя описанные величины, получаем:

$$\delta + \Delta \le (N-1)BAG + (k+1)LM_{vl} / R + k \cdot \tau_k + 2k \sum_{v \in VL} (LM_v / R + gap).$$

Таким образом, если правая часть неравенства не превосходит τ_{msg}, то ограничения на длительность передачи сообщения выполняются.

Приложение Г. Доказательство утверждения о достаточных условиях построения виртуальных каналов для всех сообщений в сетях AFDX

Утверждение 4.4 (достаточные условия построения виртуальных каналов для всех сообщений). Пусть для некоторой сети AFDX производится проектирование виртуальных каналов для множества сообщений $MSG=\{msg\}$, причем выполнены следующие условия:

- Все каналы передачи данных имеют пропускную способность не ниже R;
- Для каждого сообщения msg определены следующие параметры:
  - $n = \min\limits_{i=0..7}\left(\left\lfloor \frac{\tau_{msg} - \Delta_0}{2^i} \right\rfloor + 1, \left\lfloor \frac{T_{msg}}{2^i} \right\rfloor\right)$ (где Δ_0 – начальное приближение длительности передачи кадра), при которых выполнены ограничения:
    - $(n-1)BAG \le \tau_{msg} - \Delta_0$
    - $n \cdot BAG \le T_{msg}$
  - $k(n) = \arg\min\limits_{i=0..7}\left(\left\lfloor \frac{\tau_{msg} - \Delta_0}{2^i} \right\rfloor + 1, \left\lfloor \frac{T_{msg}}{2^i} \right\rfloor\right)$;
  - $LM(n) = \max\left(64, \left\lceil \frac{size_{msg}}{n} \right\rceil + c\right)$, где $c = 47$ байт;
  - $BAG(n) = 2^{k(n)}$.
- Для указанных параметров выполнены следующие ограничения:
  - $\sum\limits_{msg \in MSG_{es}} \left(\frac{LM(n)}{R} + gap\right) \le 500 мкс$, здесь суммирование ведется по всем сообщениям, исходящим из оконечной системы es.
  - $\sum\limits_{msg \in MSG} \frac{LM(n)}{BAG(n)} \le R$
  - Выполнены ограничения утверждения 4.3 (с учетом формирования по одному виртуальному каналу для каждого сообщения).

Тогда алгоритм спроектирует виртуальные каналы для всех сообщений.

Доказательство. На первом этапе алгоритма для каждого сообщения msg создается один виртуальный канал со следующими параметрами:

- $LM(n) = \max(64, \left\lceil \frac{size_{msg}}{n} \right\rceil + c)$;
- $BAG(n) = 2^{k(n)}, k(n) = 0..7$,

при которых выполнены ограничения:

- $(n-1)BAG \le \tau_{msg} - \Delta_0$;
- $n \cdot BAG \le T_{msg}$,

здесь $n = \min(\left\lfloor \frac{\tau_{msg} - \Delta_0}{2^i} \right\rfloor + 1, \left\lfloor \frac{T_{msg}}{2^i} \right\rfloor)$, *k(n)* равно числу *i*, на котором достигается минимум при вычислении *n*, Δ_0 является начальным приближением длительности вычисления передачи кадров (например, 1 мс). Все спроектированные виртуальные каналы удовлетворяют ограничению на период передачи сообщений.

На следующем этапе для каждого виртуального канала вычисляется джиттер по следующей формуле:

$$JM_{vl} = \sum_{v \in VL_{es}, v \neq vl} (\frac{LM_v}{R} + gap),$$

здесь VL_{es} – виртуальные каланы, исходящие из оконечной системы *es*.

Таким образом, при выполнении условия $\sum_{v \in VL_{es}, v \neq vl} (\frac{LM_v}{R} + gap) \le 500 мкс$ для каждого виртуального канала выполняются ограничения на максимальный джиттер.

Далее производится построение маршрутов виртуальных каналов. При выполнении условия $\sum_{vl \in VL} \frac{LM_{vl}}{BAG_{vl}} \le R$ даже в случае прохождения всех виртуальных каналов по одному маршруту суммарная зарезервированная пропускная способность будет меньше пропускной способности каналов передачи данных, и маршруты для всех виртуальных каналов будут построены.

На последнем этапе производится вычисление длительности передачи сообщений и проверка ограничений на длительность. При выполнении утверждения 4.3 данные ограничения будут выполнены.

Таким образом, при выполнении указанных условий алгоритм успешно построит виртуальные каналы для всех сообщений с учетом заданных ограничений.