

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра автоматизации систем вычислительных комплексов

Гудис Александр Вячеславович

**Исследование метода Отелло хеширования для доступа к таблицам классификации в архитектуре сетевого процессорного устройства**

КУРСОВАЯ РАБОТА

**Научный руководитель:**

к.ф-м.н, доцент

Волканов Дмитрий Юрьевич

Москва, 2024

**Аннотация**

В данной работе рассматривается сетевое процессорное устройство — ключевой элемент программируемых коммутаторов, отвечающий за обработку сетевого трафика. В процессе этой обработки каждый пакет данных проходит несколько этапов, одним из которых является классификация. Классификация пакета заключается в поиске соответствующего правила, определяющего, как следует изменить его заголовки и на какой порт направить. Эти правила хранятся в соответствующих таблицах классификации, выбор типа которых зависит от множества факторов. Поиск по таким таблицам является вычислительно трудной задачей, поэтому разработка более эффективных методов её выполнения остаётся актуальной. В работе предлагается использование метода хеширования данных для организации таблиц классификации типа MAC-VLAN. Рассматривается и изучается метод Отелло хеширования для представления такой таблицы. Сравнительные экспериментальные исследования с уже существующими классическими методами хеширования для доступа к таблицам классификации показали, что Отелло хеширование обладает

…………….

высокой эффективностью в рамках поставленной задачи, а также гибкостью настройки, что делает их применимыми в широком спектре сценариев.

Оглавление

[Введение 4](#_Toc39190115)

[1. Постановка задачи 5](#_Toc39190116)

[2. Обзор литературы и существующих решений 6](#_Toc39190117)

[2.1. Подраздел обзора 6](#_Toc39190118)

[2.2. Подраздел обзора 6](#_Toc39190119)

[2.3. Выводы к разделу 2 7](#_Toc39190120)

[3. Исследование и решение задачи 8](#_Toc39190121)

[3.1. Подраздел 8](#_Toc39190122)

[3.2. Подраздел 8](#_Toc39190123)

[3.2.1. Подраздел 3-го уровня 8](#_Toc39190124)

[4. Экспериментальное исследование 10](#_Toc39190125)

[4.1. Подраздел 10](#_Toc39190126)

[Заключение 11](#_Toc39190127)

[Литература 11](#_Toc39190128)

# Введение

В современном мире сетевые технологии играют важнейшую роль в обеспечении коммуникации и передачи данных. Развитие технологий и растущие потребности общества требуют более эффективных решений в области сетевых взаимодействий. Ключевыми устройствами, отвечающими за высокую пропускную способность и минимизацию задержек в компьютерных сетях, являются коммутаторы и маршрутизаторы. Они выполняют обработку сетевых пакетов, определяют их маршруты, управляют приоритетами передачи данных, выявляют и блокируют вредоносный трафик и выполняют другие важные функции.

В настоящее время активно разрабатываются программируемые коммутаторы, отличительной чертой которых является реализация обработки пакетов на уровне программного обеспечения, а не аппаратного. Их центральный компонент — сетевое процессорное устройство (СПУ) [1] — позволяет гибко настраивать логику обработки данных, обеспечивая при этом высокую производительность. Такие коммутаторы являются важной частью программно-конфигурируемых сетей (SDN), но также находят применение и в традиционных сетевых архитектурах.

Использование программной обработки сетевых пакетов в СПУ позволяет поддерживать широкий спектр протоколов и работать на различных уровнях модели OSI. Однако такая гибкость сопровождается увеличенными требованиями к памяти, необходимой для хранения алгоритмов обработки, а также снижением скорости обработки пакетов. Для решения этой проблемы требуется оптимизация структур данных, отвечающих за хранение правил маршрутизации и обработки.

Одним из ключевых методов оптимизации является хеширование данных, которое позволяет каждой хранимой единице назначить уникальный идентификатор, упрощающий и ускоряющий её поиск. Введение хеширования в процесс классификации пакетов может существенно сократить время его выполнения, приближая его к константному, однако это требует дополнительных затрат памяти.

Основная сложность хеширования связана с коллизиями, возникающими, когда разные данные получают одинаковый идентификатор. Чем больше выделенной памяти, тем ниже вероятность таких коллизий. Однако существуют различные методы хеширования, позволяющие минимизировать избыточное использование памяти без значительного увеличения времени поиска. В данной работе рассмотрен и изучен один из таких методов.

# Цель работы

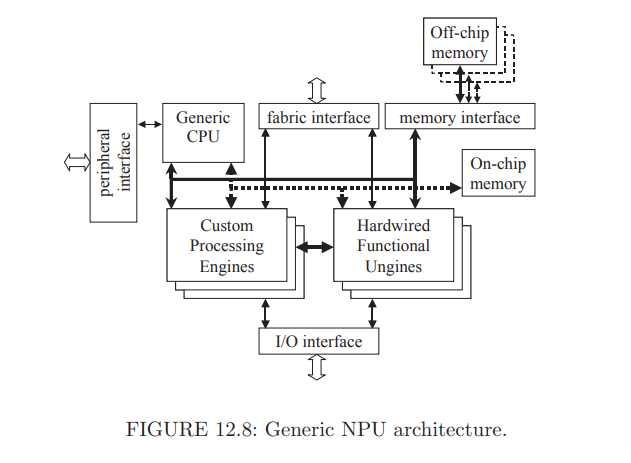
Цель данной работы — изучение метода Отелло хеширования и разработка подхода к представлению таблиц классификации на его основе.

Для достижения этой цели планируется:

1. Провести анализ и изучить теоретическую составляющую данного метода, предложенную авторами в их научной статье [7]
2. Разработать модель представления таблиц классификации с использованием этого метода хеширования;
3. Реализовать предложенную модель на практике;
4. Выполнить экспериментальный и сравнительный с другими классическими методами хеширования анализ эффективности.

# Предметная область

## 2.1 Сетевое процессорное устройство



Картинка из литературы на лето, страница 440

Сетевое процессорное устройство (СПУ) — это специализированный микропроцессор, предназначенный для обработки и управления сетевым трафиком в программируемых коммутаторах. Архитектура СПУ представляет собой баланс между высокой производительностью, характерной для узкоспециализированных решений, и гибкостью настройки, присущей процессорам общего назначения. Благодаря этому СПУ поддерживает широкий спектр сетевых стандартов и протоколов, обеспечивая при этом высокую пропускную способность.

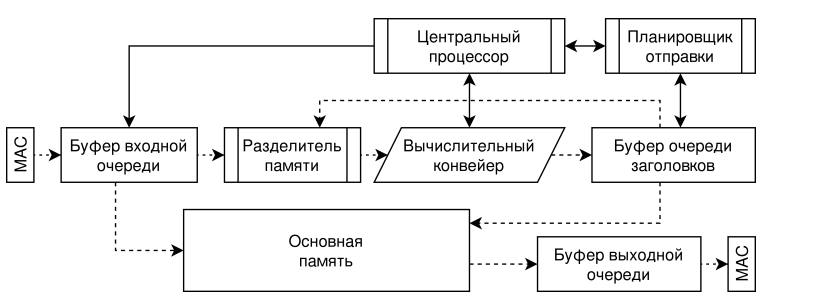


Рис. 1: Пример архитектуры СПУ

Как правило, обработка трафика в СПУ осуществляется с использованием конвейера вычислительных ядер. Такой подход позволяет разбить процесс обработки пакетов на последовательные этапы, что значительно ускоряет обработку данных. Однако задержка на любой из стадий конвейера может повлиять на общую производительность системы. Обработка сетевого пакета в СПУ включает следующие этапы:

1. Прием пакета с входного порта;
2. Извлечение заголовка пакета и временное сохранение его содержимого во внутренней памяти;
3. Классификация пакета, заключающаяся в поиске соответствующего правила в таблице классификации;
4. Модификация заголовка в соответствии с найденным правилом;
5. Объединение измененного заголовка с телом пакета;
6. Передача пакета в систему управления очередями;
7. Отправка пакета на выходной порт согласно определенному правилу.

Третий шаг, классификация пакета - является одним из самых ресурсоемких этапов. Он представляет собой поиск по соответствующей таблице классификации, содержащей правила обработки пакетов.

## 2.2 Классификация пакетов

Таблица классификации представляет собой структуру данных, содержащую информацию о способе обработки пакетов определенного типа. Записи в таблице, называемые правилами, могут быть упорядочены по приоритету. Каждое правило включает в себя несколько полей: ключ, определяющий условия сопоставления, и значение, задающее соответствующие действия.

Ключ правила формируется на основе определенных частей заголовков пакета, таких как исходный и целевой адреса, порты, тип протокола и другие параметры. В зависимости от задачи ключ может быть задан как точное значение (константа) или как шаблон (wild-card), например, префикс IP-адреса. Поиск по таблице классификации может выполняться либо по полному соответствию, либо по частичному, например, по наиболее длинному совпадающему префиксу.

Значение правила играет разную роль: оно может представлять собой константу, используемую при обработке пакета, либо ссылку на исполняемый код, предназначенный для обработки пакетов данного типа. Такой подход особенно распространен в таблицах потоков (Flow table).

В данной работе рассматриваются таблицы классификации типа «MAC-VLAN», работающие на втором (канальном) уровне модели ISO-OSI.

|  |  |
| --- | --- |
| MAC-VLAN table | |
| MAC-VLAN | port |
| xx:xx:xx:xx:xx:xx-xxx  (x – 4 бита) | 1 |
| … | … |
| xx:xx:xx:xx:xx:xx-xxx | n |

Таблица 1: Таблица классификации «MAC-VLAN»

Они позволяют сопоставлять MAC-адреса с виртуальными сетями VLAN, обеспечивая эффективную маршрутизацию трафика.

Правила в таких таблицах имеют следующий формат:  
<{MAC-addr, VLAN-id}, port-mask>, где:

* {MAC-addr, VLAN-id} — ключ, представляющий собой 60-битную строку, полученную путем объединения MAC-адреса и идентификатора VLAN;
* port-mask — значение, представленное 8-битной строкой, указывающей, на какие порты следует направить пакет после обработки.

Поиск в таблице классификации MAC-VLAN осуществляется по точному совпадению ключа.

## 2.3 Хеширование данных

Хеш-таблица [2; 3] представляет собой структуру данных, организованную в виде ассоциативного массива, который обеспечивает эффективное хранение и извлечение пар ключ-значение. Простейший вариант такой таблицы показан на рисунке 2. На этапе инициализации массив заполняется специальными значениями, обозначающими пустые ячейки (NULL).

Основой работы хеш-таблицы являются одна или несколько хеш-функций, преобразующих множество ключей в множество хеш-кодов. Хеш-код представляет собой индекс массива, по которому можно получить доступ к соответствующей записи.

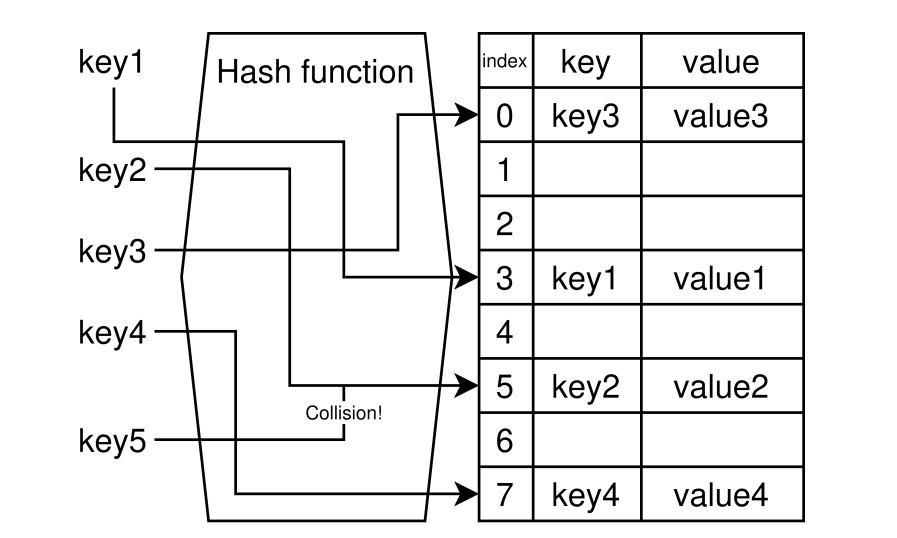


Рис. 2: Пример хеш-таблицы



?????

Для использования в структурах данных хеш-функции должны обладать рядом свойств:

1. Фиксированная длина выходных данных – все хеш-коды находятся в заранее заданном диапазоне;
2. Сюрьективность – каждому ключу соответствует определенный хеш-код, который не обязательно является уникальным;
3. Равномерное распределение – хеш-коды должны быть распределены как можно более равномерно по множеству ключей;
4. Лавинный эффект – небольшое изменение входных данных должно приводить к значительному изменению хеш-кода, что обеспечивает учет каждого бита ключа.

В теории хеш-функции позволяют получать доступ к данным за константное время, не зависящее от объема хранимой информации. Если удается подобрать хеш-функцию, обеспечивающую биективное отображение ключей в хеш-коды, такое хеширование называют идеальным. Однако при расширении множества ключей идеальное хеширование, как правило, теряет свои преимущества.

Одна из ключевых проблем при работе с хеш-таблицами — коллизии, возникающие, когда разные ключи получают одинаковый хеш-код. В таких ситуациях необходимо применять методы разрешения коллизий, чтобы корректно сохранить и извлечь данные. Наиболее распространенный способ — использование цепочек: массив ячеек хранит ссылки на связанные списки, в которых размещаются записи с одинаковым хеш-кодом, однако со временем такая структура данных может деградировать и перестать обладать стандартными преимуществами хеш-таблиц и станет более похожа на классический список.

Для минимизации коллизий важным свойством хеш-функции является равномерность распределения хеш-кодов, что повышает эффективность и отказоустойчивость хеш-таблицы. Однако в некоторых случаях коллизии могут накапливаться до такой степени, что добавление новых записей становится невозможным. В таких ситуациях применяют перехеширование — смену хеш-функции и пересборку структуры.

Дополнительный инструмент для улучшения распределения хеш-кодов — соль (salt), представляющая собой дополнительный константный параметр хеш-функции. Использование соли увеличивает количество возможных вариантов хеширования и повышает вероятность выбора оптимальной функции для текущего множества ключей.

# Постановка задачи

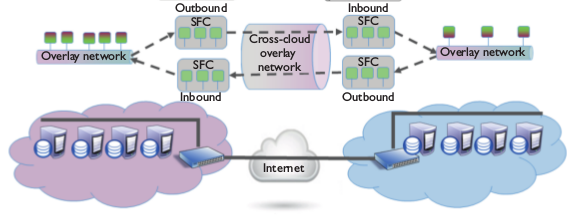
<…>.

# **Обзор литературы и существующих решений**

<…>.

# Подраздел обзора

<…>.



WIM

VIM2

VIM1

*Рисунок 2. Создание цепочки из функций, размещенных в разных доменах виртуальной инфраструктуры*

# Подраздел обзора

<…>

Список:

* элемент;
* элемент;
* элемент..

# Выводы к разделу 2

<…>

# **Исследование и решение задачи**

<…>

# Подраздел

**<…>**

# Подраздел

<…>

<…>

.

… контроллера с коммутаторами будет осуществляться по протоколу OpenFlow 1.3 [13] и с использованием плагина OpenDaylight OVSDB [14] ….

# Подраздел 3-го уровня

<…>

# **Экспериментальное исследование**

**<…>**

# Подраздел

**<…>**

# Заключение

**<…>**

# Литература

[1] Об одном подходе к построению сетевого процессорного устройства / С. О. Беззубцев [и др.] // Моделирование и анализ информационных систем. — 2019. — Т. 26, № 1. — С. 39—62.

[1] Dong, C., Wang, F, Feng, D., 2021. Dxhash: A scalable consistent hash based on the pseudo-random sequence. arXiv preprint arXiv:2107.07930.

[2] Maurer W. D., Lewis T. G. Hash table methods // ACM Computing Surveys (CSUR). — 1975. — Т. 7, № 1. — С. 5—19.

[3] A survey on learning to hash / J. Wang [и др.] // IEEE transactions on pattern analysis and machine intelligence. — 2017. — Т. 40, № 4. — С. 769—790.

[2] Ahmadi, M. and Wong, S., 2007, January. Hashing functions performance in packet classification. In Proceedings of the International Conference on the Latest Advances in Networks (ICLAN’07) (pp. 127-132).

[3] Dharmapurikar, S., Krishnamurthy, P. and Taylor, D.E., 2003, August. Longest prefix matching using bloom filters. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (pp. 201-212).

[4] Charles, D. and Chellapilla, K., 2008, September. Bloomier filters: A second look. In European Symposium on Algorithms (pp. 259-270). Berlin, Heidelberg: Springer Berlin Heidelberg.

[5] Shi, S., Qian, C., 2020. Ludo hashing: Compact, fast, and dynamic keyvalue lookups for practical network systems. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 4(2), pp.1-32.

[6] Pagh, R., Rodler, F.F., 2001, August. Cuckoo hashing. In European Symposium on Algorithms (pp. 121-133). Berlin, Heidelberg: Springer Berlin Heidelberg.

[7] Yu, Y., Belazzougui, D., Qian, C., Zhang, Q., 2018. Memory-efficient and ultra-fast network lookup and forwarding using Othello hashing. IEEE/ACM Transactions on Networking, 26(3), pp.1151-1164.

[8] Александров А.В., Волканов Д.Ю. Модифицированный метод хеширования данных для доступа к таблице классификации MACVLAN в сетевом процессорном устройстве // Программные системы и инструменты. Тематический сборник № 24. Т. 24. – М.: ООО «МАКС Пресс», 2024. – С. 6–16. DOI: 10.29003/m4299.978-5-317-07288-9