ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ/COMPUTING SYSTEMS AND THEIR ELEMENTS

DOI: https://doi.org/10.60797/IRJ.2025.155.8

ОЦЕНКА АЛГОРИТМИЧЕСКОЙ СЛОЖНОСТИ МОДИФИЦИРОВАННОГО АЛГОРИТМА ПОИСКА ДАННЫХ В КОНТЕНТ-ОРИЕНТИРОВАННЫХ СЕТЯХ

Научная статья

Дегаев М.Н.[1, *]
[1] ORCID : 0009-0006-8551-145X;
[1] Поволжский государственный технологический университет, Йошкар-Ола, Российская Федерация

* Корреспондирующий автор (worksome[at]ya.ru)

Аннотация

Одним из перспективных подходов для снижения нагрузки на существующую сетевую инфраструктуру является использование сетей, ориентированных на данные (Information-Centric Networking, ICN), таких как Named Data Networking (NDN). В данной статье предлагается модификация структуры таблицы пересылаемой информации (FIB) путём добавления новой служебной таблицы. Также рассмотрен модифицированный алгоритм поиска контента в сети NDN, что позволяет уменьшить время поиска информации и снизить вычислительную сложность алгоритма. Представленный в статье алгоритм использует хэш-таблицы и фильтры Блюма для оптимизации процесса поиска, что обеспечивает более высокую производительность по сравнению с традиционными методами.

Ключевые слова: ICN, NDN, Forwarding Information Table, Pending Interest Table, Content Store, маршрутизатор контента, фильтр Блюма, хэш-таблица.

EVALUATION OF THE ALGORITHMIC COMPLEXITY OF A MODIFIED ALGORITHM FOR DATA SEARCH IN CONTENT-BASED NETWORKS

Research article

Degaev M.N.[1, *]
[1] ORCID : 0009-0006-8551-145X;
[1] Volga State University of Technology, Yoshkar-Ola, Russian Federation

* Corresponding author (worksome[at]ya.ru)

Abstract

One of the promising approaches to reduce the load on the existing network infrastructure is the use of Information-Centric Networking (ICN) such as Named Data Networking (NDN). This article proposes a modification of the forwarding information base (FIB) structure by adding a new service table. A modified algorithm for content retrieval in NDN network is also discussed, which reduces the information retrieval time and reduces the computational complexity of the algorithm. The algorithm presented in the paper uses hash tables and Blum filters to optimise the search process, which provides better performance than traditional methods.

Keywords: ICN, NDN, Forwarding Information Table, Forwarding Information Table, Pending Interest Table, Content Store, content router, Blum filter, hash table.

Введение

Доминирующим видом телекоммуникационного трафика в последнее время стал трафик данных. По данным журнала ComNews Research [1], на долю пакетного трафика приходится свыше 75% полосы пропускания, которая используется в мире для бизнес-сервисов. С каждым годом объём данных, передаваемых через интернет, увеличивается. Это связано с ростом количества пользователей, увеличением популярности видеоконтента, развитием интернета вещей (Inetrnet of Things — IoT) и других информационных технологий [2]. Увеличение трафика создаёт нагрузку на существующую сетевую инфраструктуру, что требует более эффективных решений для управления и передачи данных.

В связи с тем, что объёмы данных растут очень быстро, впервые в истории развития средств связи потребовалось создание совершенно новых сетей с колоссальной пропускной способностью, использующих технологию коммутации пакетов [3]. Стремительное развитие Интернет-приложений и появление новых оптических средств связи также влияют на концепцию развития современных сетей передачи данных. Одним из таких перспективных решений являются сети, ориентированные на данные (ICN — Information-Centric Networking) [4].

Переход к сетям ICN предлагает использование нового подхода к представлению информации в сети и к процессу её поиска, где основное внимание уделяется самим данным, а не месту их хранения. Это позволяет более эффективно управлять трафиком и кэшированием, улучшая доставку данных и снижая нагрузку на сеть.

Поскольку данные могут быть кэшированы и доставлены из различных точек сети, ICN снижает нагрузку на исходные серверы, что особенно важно для популярного контента. Это также уменьшает необходимость в дорогостоящих масштабируемых серверных решениях.

Целью данной работы является повышение эффективности методов и алгоритмов поиска данных в хеш-таблицах с помощью индексирования элементов данных.

Для достижения обозначенной цели в работе поставлены и решены следующие **задачи**:

1. Анализ современного состояния и тенденций развития контент-ориентированных сетей с целью определения способов повышения их производительности.

2. Анализ известных алгоритмов поиска данных в кэше маршрутизатора контента.

3. Исследование и разработка модифицированных структур кэшируемых данных, основанных на дополнительных хеш-таблицах.

4. Исследование и разработка алгоритмов ускоренного поиска данных в кэше маршрутизатора контента и выполняемых с его помощью операций обработки кэшируемых данных (чтения/записи и вытеснения данных).

5. Оценка трудоемкости исходных и модифицированных алгоритмов обработки кэшируемых данных.

Именованные сети передачи данных (Named Data Networking NDN) и контент-ориентированные сети (Content-Centric Networking CCN) — наиболее известные архитектуры ICN. В сетях NDN используются служебные таблицы в маршрутизаторе контента (рис.1) — таблица FIB (Forwarding Information Base — Таблица пересылаемой информации (контента)), PIT (Pending Interest Table — Таблица запросов интереса) и хранилище контента CS (Content Store — Хранилище контента) [5].

Переадресация пакетов в маршрутизаторе NDN-сети предполагает сопоставление префиксов имен с использованием самого длинного префикса. В отличие от IP-адресов фиксированной длины, имена могут иметь переменную длину и могут быть довольно длинными — более 100 байт [6]. Более того, размер таблицы FIB в NDN может быть намного больше, чем таблица IP-переадресации [7]. Требования к памяти, узкие места ввода-вывода и периодические обновления записей в таблице FIB являются основными проблемами при разработке алгоритмов поиска информации в таблице FIB [8].

Чтобы запросить объекты данных, пользователи в сети NDN отправляют пакет интереса на маршрутизатор контента МК, который будет передаваться между узлами МК поэтапно. В МК в примере на рис. 1 имена существующего контента сопоставлены с выходными интерфейсами. Пользователь отправляет пакет интереса для контента с именем /aueb.gr/ai/new.htm. МК А получает пакет и проверяет таблицу FIB, куда переслать запрос, которым является МК С. Затем пакет интереса пересылается в МК С. В таблице PIT в МК А хранится название запроса и информация о том, откуда он был получен на случай, если данные вернутся. В МК С выполняется тот же процесс, но этот FIB имеет выходной интерфейс, адресованный Публикатору 1 (сервер, на котором находится исходный файл/контент). Таким образом, пакет интереса пересылается Публикатору 1. Пакет интереса теперь игнорируется, и в МК С возвращается сообщение данных (4). МК С проверяет свою таблицу PIT, чтобы определить, куда отправить сообщение данных, исходя из того, откуда поступил запрос. Как только сообщение с данными отправлено, таблица CS обновляется, чтобы сохранить данные на случай, если они будут запрошены снова. МК А продолжает этот процесс, и пользователь получает запрошенные данные.

В данной статье предлагается модификация структуры исходной таблицы FIB и алгоритма поиска контента в этой таблице с целью уменьшения времени поиска информации в сети NDN, а также приводится оценка алгоритмической сложности базового и модифицированного алгоритма поиска контента в сети NDN.

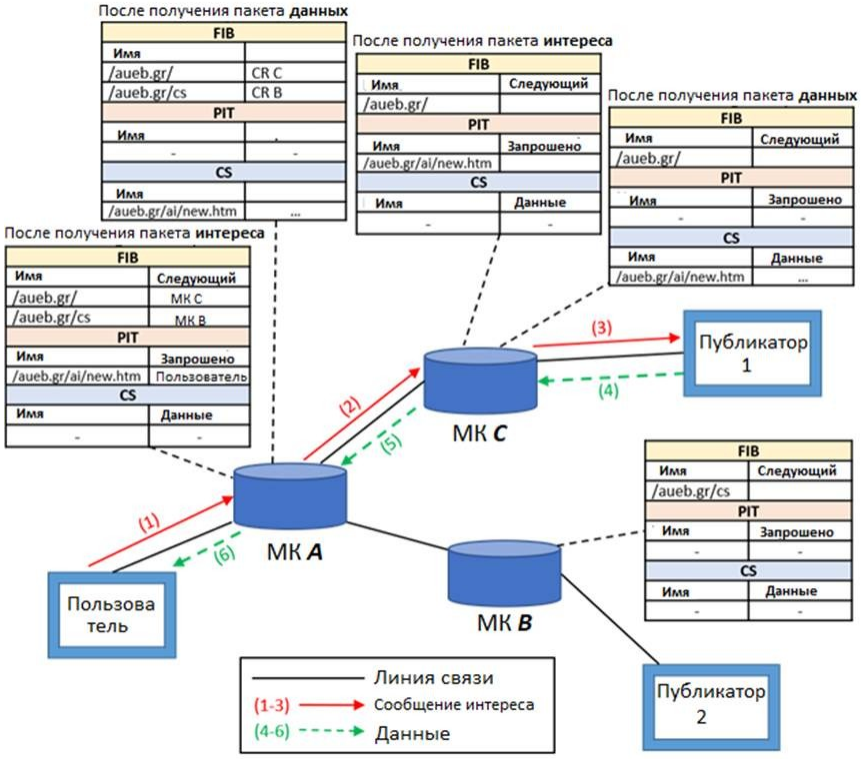

Рисунок 1 - Архитектура сети NDN

DOI: https://doi.org/10.60797/IRJ.2025.155.8.1

Примечание: МК – маршрутизатор контента

Основными алгоритмами обработки данных в маршрутизаторе контента являются алгоритмы поиска контента и генерации (вставки) хэш-значений в таблице FIB.

На рис. 2 представлена граф-схема базового обобщённого алгоритма поиска контента в сети NDN.

Для анализа сложности данного алгоритма выделены основные этапы и показана сложность этапов, связанных с непосредственной обработкой одного пакета данных на входящем интерфейсе маршрутизатора контента:

1. Отправка клиентом пакета интереса.

Просмотр таблицы FIB в ОЗУ маршрутизатора

Проверка на наличие записи с полным совпадением запроса — O(m), где *m* — количество записей в таблице FIB.

Поиск в таблице FIB записи с максимальным совпадением префикса — O(m).

Пересылка пакета интереса хосту из записи в таблице FIB.

Поиск запрошенного контента в памяти маршрутизатора (если контент уже загружен).

Формирование пакета данных для отправки.

Отправка пакета данных клиенту.

Максимальная сложность алгоритма связана с поиском в таблице FIB, что составляет **O(2m)** для случая, когда искомая запись соответствует неполному совпадению с префиксным именем исходного запроса и требуется повторный просмотр записей таблицы.

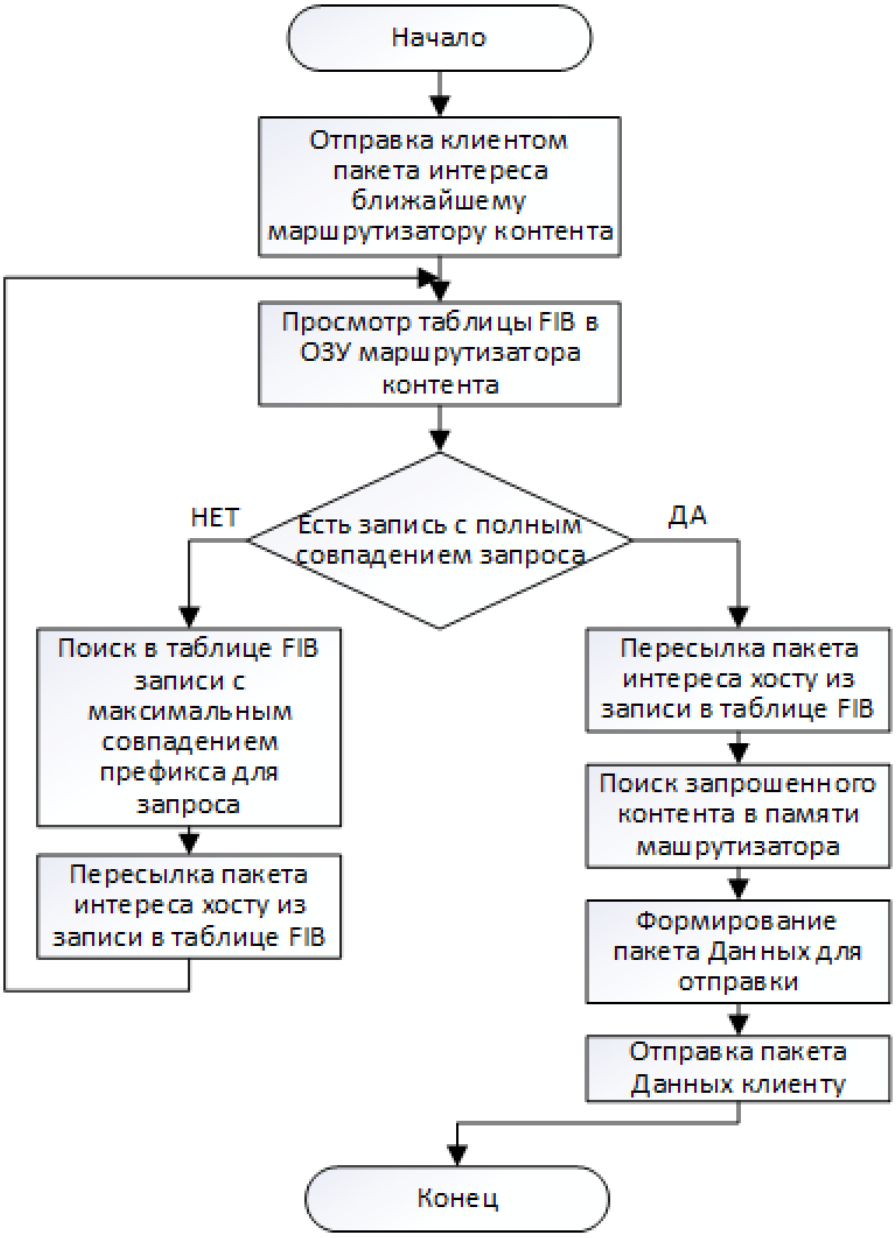

Рисунок 2 - Общий алгоритм поиска контента в сети NDN

DOI: https://doi.org/10.60797/IRJ.2025.155.8.2

В свою очередь этап поиска данных в таблице FIB также можно представить в виде отдельного алгоритма, граф-схема которого представлена на рис. 3. Данный алгоритм для анализа его сложности целесообразно разбить на шесть основных этапов:

1. Получение пакета интереса от клиента.
2. Просмотр таблицы FIB в ОЗУ маршрутизатора для доступа.
3. Полнотекстовой поиск в таблице FIB записи с максимальным совпадением префикса запроса длиной K- O(m·K), где *m* — количество записей в таблице FIB, а *K* — длина префикса в битах.
4. Проверка наличия записи с длиной префикса в запросе длиной K.
5. Поиск в таблице FIB записи с длиной префикса K−1 - O(m·(K−1)).
6. Пересылка пакета интереса хосту из записи в таблице FIB.

Основная сложность связана с полнотекстовым поиском в таблице FIB, который зависит от количества записей и длины префикса. Таким образом, общую сложность алгоритма можно представить, как **O(m·K)**, где *m* — количество записей в таблице FIB, а *K* — максимальная длина префикса.

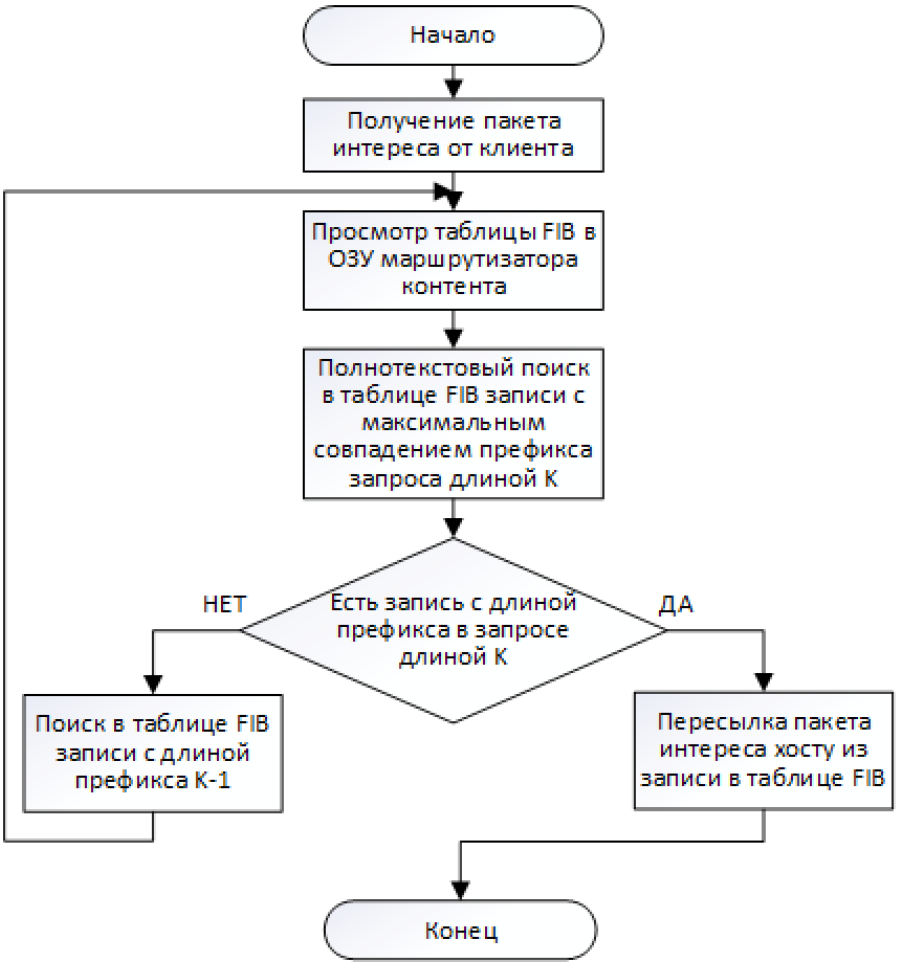

Рисунок 3 - Алгоритм поиска контента в таблице FIB

DOI: https://doi.org/10.60797/IRJ.2025.155.8.3

Для реализации предложена новая структура таблицы FIB, представленная на рис. 4, которая ускорит процесс поиска контента в кэше маршрутизатора контента МК. Как показано на рис. 4, самый длинный префикс имени в указанном наборе данных равен 16. Предлагаемый алгоритм обрабатывает префиксы имен соответствующей длины от 1-й хэш-таблицы до 16-й хэш-таблицы, а 17-я хэш-таблица является дополнительной хэш-таблицей, используемой для вставки префикса имени в случае переполнения хэш-блока. Каждая хэш-таблица содержит несколько хэш-блоков (N), а также фильтр Блюма для хранения префиксов объектов/контента соответствующей длины [10].

Каждый хэш-блок содержит несколько значений хэша и два параметра, один из которых указывает, заполнена ли ячейка хэша, а другой — коллизии при вставке в ячейку хэша. Если хэш-блок заполнен, алгоритм вставляет префикс имени в тот же номер хэш-блока дополнительной хэш-таблицы. Кроме того, параметр «Количество коллизий» означает максимальное количество обращений к дополнительной хэш-таблице, а запись хэша хранит только основную информацию о префиксе имени. Описанные выше хэш-структуры могут обеспечить эффективные операции поиска имен и сбалансировать нагрузки на все хэш-таблицы, так что даже при переполнении некоторых хэш-блоков дополнительная хэш-таблица может компенсировать негативные последствия такие, как возникновение коллизий при вычислении хэш-функций. Например, операция вставки описана в алгоритме на рис. 5.

Поскольку существует только одна дополнительная хэш-таблица для хранения значений конфликта хэшей, установим количество записей в хранилище в 50 раз больше, чем количество префиксов имен, чтобы гарантировать, что коллизии хэшей не приведет к переполнению хэш-таблицы. Это означает, что произведение чисел хэш-блоков и номеров хэш-записей в 50 раз превышает количество префиксов имен в каждой хэш-таблице. Алгоритм использует хэш-функцию murmur [7] — это некриптографическая хэш-функция, подходящая для общего поиска на основе хэша, которая обладает высокой сбалансированностью и низким уровнем коллизий при обработке большого массива данных.

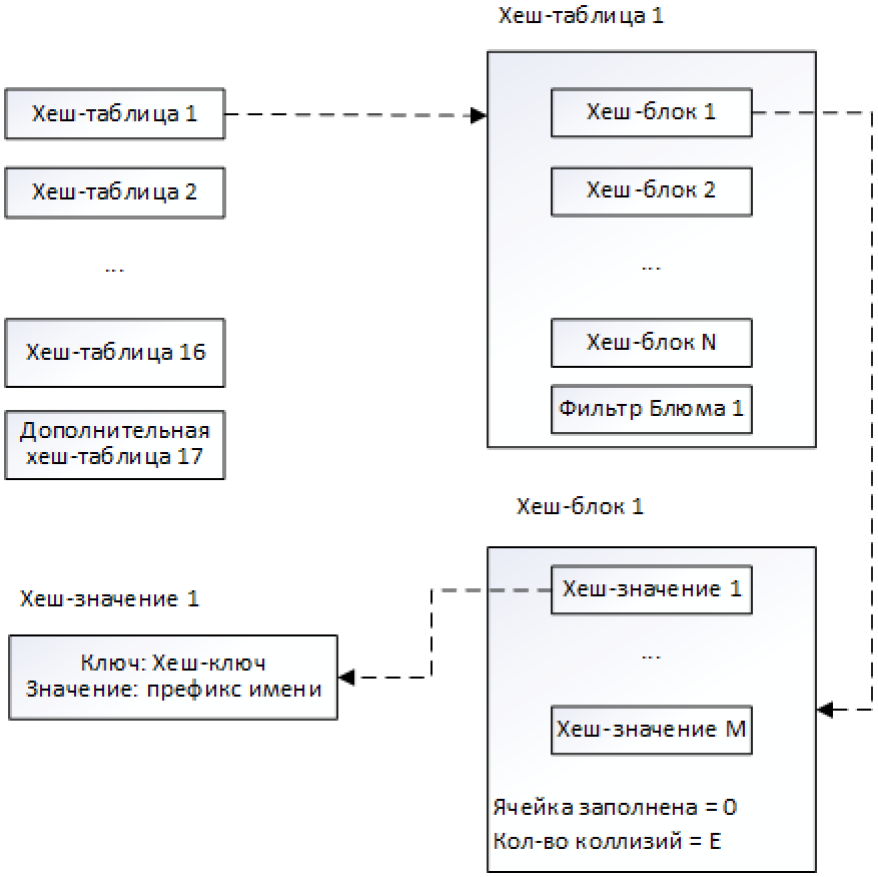

Рисунок 4 - Модифицированная структура таблицы FIB
DOI: https://doi.org/10.60797/IRJ.2025.155.8.4

На рис. 5,6 представлены модифицированные алгоритмы по генерации (вставки) хеш-значений и поиска в модифицированной таблице FIB [12].

Для анализа сложности алгоритма на рис.5. рассмотрим каждый шаг:

Получение пакета интереса от клиента.

Извлечение полного имени контента.

Выбор первых *n* бит в имени контента — O(n), где *n* — количество бит.

Загрузка каждого символа имени — O(p), где *p* — длина имени.

Проверка наличия символа (бита) на входе — O(1) для каждого символа.

Преобразование с помощью хэш-функции Murmur — O(1) для каждого символа.

Запись значения в хэш-таблицу — O(1) для каждого символа.

Основная часть алгоритма заключается в обработке каждого символа имени, что составляет O(p). Хэш-функция и запись в таблицу выполняются за постоянное время для каждого символа. Таким образом, общая сложность алгоритма будет **O(p)**, где p — длина имени.

Для анализа сложности алгоритма, представленного рис.6, рассмотрим каждый шаг:

1. Получение пакета интереса от клиента.
2. Извлечение полного имени контента.
3. Выбор первых *n* бит в имени контента — O(n), где *n* — количество бит.
4. Пропуск префиксов имени через фильтр Блюма — O(k), где *k* — количество хэш-функций в фильтре Блюма.
5. Проверка совпадения с поисковым запросом — O(1) для каждого символа.
6. Преобразование с помощью хэш-функции Murmur.

Поиск значений в хэш-таблице — O(1) для каждого символа.

Поиск пересечений результатов фильтра Блюма в хэш-таблице — O(1) для каждого символа.

Пересылка пакета интереса.

Основные операции, влияющие на сложность, включают пропуск префиксов через фильтр Блюма и выбор первых *n* бит. Таким образом, общая сложность алгоритма будет **O(n+k)**, где *n* — количество бит для формирования префикса, а *k* — количество хэш-функций в фильтре Блюма.

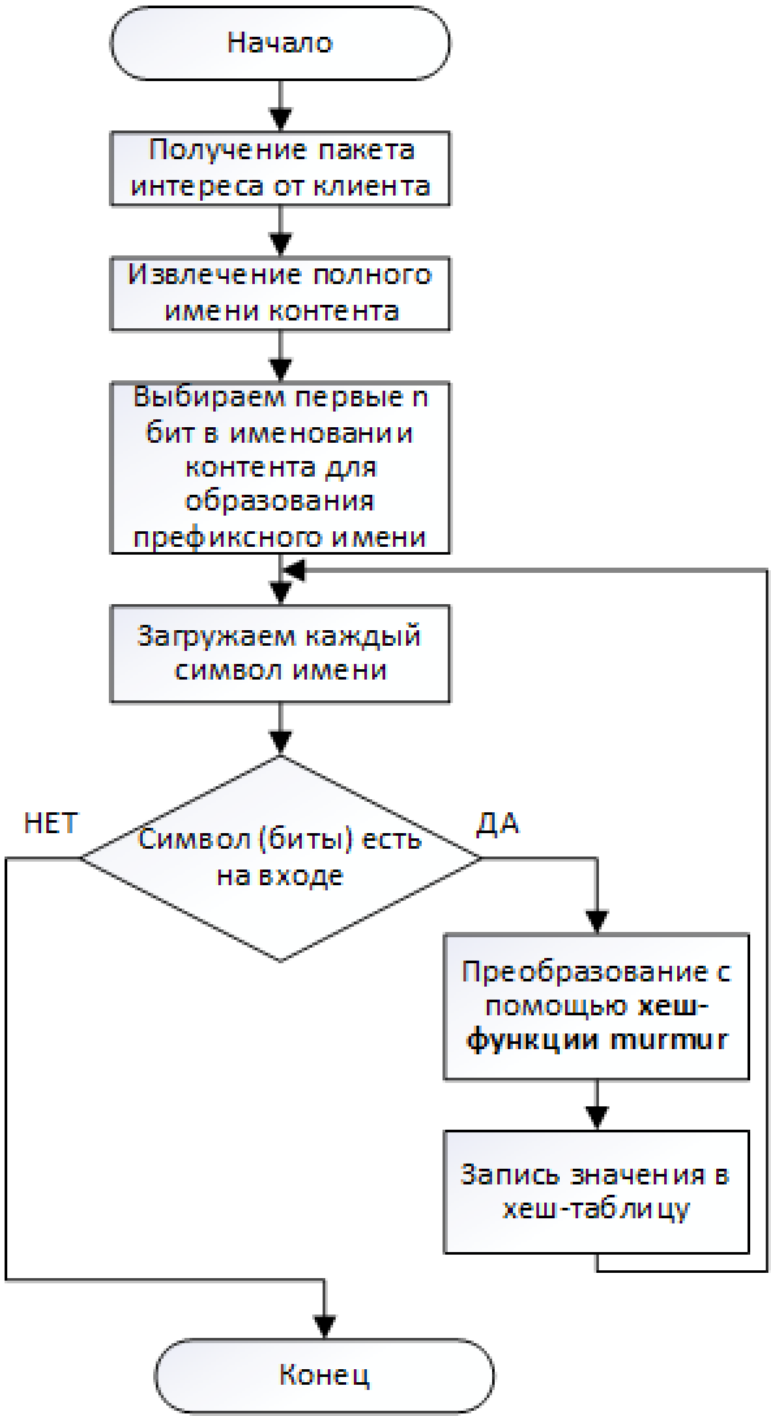

Рисунок 5 - Модифицированный алгоритм генерации (вставки) хэш-значений в таблице FIB

DOI: https://doi.org/10.60797/IRJ.2025.155.8.5

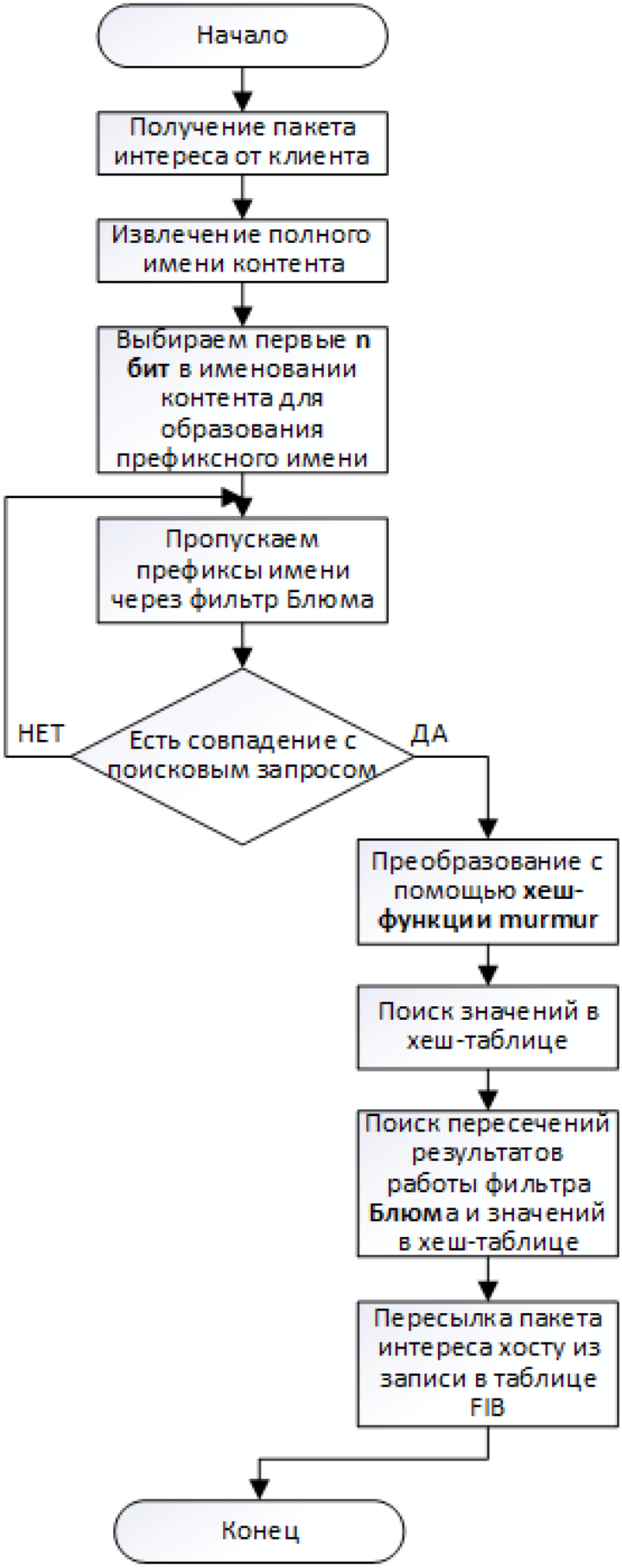

Рисунок 6 - Модифицированный алгоритм поиска контента в таблице FIB

DOI: https://doi.org/10.60797/IRJ.2025.155.8.6

Результаты исследований

Рассмотрим таблицу FIB, которая хранит $m=10^6$ префиксных имен, и оценим сложность алгоритмов поиска контента, генерации (вставки) хэш-значений в таблице FIB для $n=6$ количество бит для формирования префикса, $k=3$ количество хэш-функций в фильтре Блюма, $p=32$ — длина имени, а $K=32$ — максимальная длина префикса.

Таблица 1 - Сравнительная сложность алгоритмов

DOI: https://doi.org/10.60797/IRJ.2025.155.8.7

Сложность	Традиционный алгоритм		Модифицированный алгоритм	
	алгоритм поиска контента в сети NDN	алгоритм поиска контента в таблице FIB	алгоритм поиска контента в таблице FIB	алгоритм генерации (вставки) хэш-значений в таблице FIB
	O(10^6)	O($10^6 \cdot 32$)	O(32)	O(6+3)

Согласно табл. 1, можно сделать вывод о меньшей (на несколько порядков) вычислительной сложности представленного алгоритма, что ускорит процесс поиска контента в таблице FIB и сети NDN целиком, снизит вычислительные затраты, число обращений к памяти (ОЗУ) маршрутизатора контента.

Если значения хэша находятся в диапазоне от 0 до N для общего числа K, вероятность P различных значений хэша определяется как уравнение (1), а уравнение (2) является производным от уравнения (1). Основываясь на уравнениях (1) и (2), в уравнении (3) можно рассчитать частоту конфликтов (коэффициент конфликтности) хэшей R. Процесс вывода уравнения должен гарантировать, что значение $k^2/2N$ будет меньше 0,1.

$$P = \prod_{i=1}^{k-1} \frac{N-i}{N} \quad (1)$$

$$P \approx e^{\frac{-k(k-1)}{2N}} \quad (2)$$

$$R = 1 - P \approx \frac{k^2}{2N} \quad (3)$$

Для любого целевого имени T длиной L символов затраты времени на поиск C на поиск могут быть рассчитаны в соответствии с уравнением (1).

$$c = a_1 \sum_{i=L}^{1} c_{hash\ lookup} + a_2 \sum_{i=L}^{1} c_{bloom\ lookup} + a_3 \sum_{i=L}^{1} c_{string\ match} \quad (4)$$

где a_1 — весовой коэффициенты для поиска с в хеш-таблице;

a_2 — весовой коэффициенты для поиска с помощью фильтра Блюма;

a_3 — весовые коэффициенты для полнотекстового поиска;

C — затраты времени на поиск (мс);

Это означает, что для определения целевого имени время поиска складывается из трех составных частей. Поскольку в существующем алгоритме поиска со структурами хэш-таблиц затраты времени включают первую и последнюю части, a_2 равно 0. Для алгоритмов, основанных на хэш-таблицах, a_2 и a_3 равны 0. Для алгоритмов, основанных на фильтре Блюма, значения a_1 и a_3 равны 0. В предлагаемом алгоритме значение a_2 меньше, чем в других алгоритмах, основанных на фильтре Блюма, с поддержкой префиксов функций. Например, алгоритмам, основанным на фильтре Блюма, требуется четыре раза повторить полный поиск по фильтру Блюма, в то время как предлагаемому алгоритму это требуется только один раз. Наконец, алгоритмам, основанным на хэш-таблицах, требуется четыре раза выполнить полный поиск по хэшу, что значительно дороже. Основываясь на анализе производительности реализованного алгоритма и сравнении, этот предложенный алгоритм, основанный на префиксе функции, обладает отличной производительностью поиска по имени NDN.

Для проведения экспериментов был выделен сервер,состоящийизодногопроцессораInteli5-6400 и 16 ГБ оперативной памяти, работающий под управлением OCUbuntu 22.04 и коллекции компиляторов GNU. На рис. 7 представлена зависимость времени поиска от количества целевых имен при разных значениях весовых коэффициентов для формулы 4. Из графиков видно, что наилучшее время поиска при a3=0, т.е. когда не используется полнотекстовый поиск, а используется поиск в хеш-таблице и фильтры Блюма — 15 мс (при количестве имен префиксов равным $5*10^6$). При увеличении количества имен используемых префиксов уменьшение времени поиска еще более заметен.

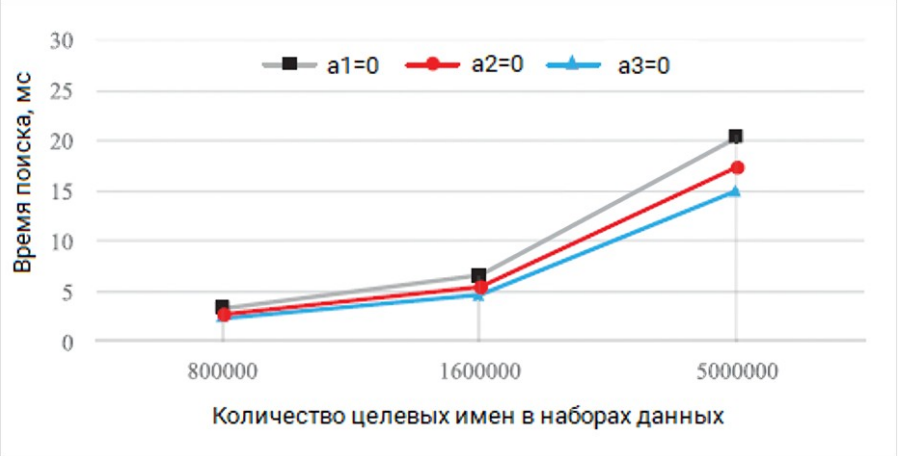

Рисунок 7 - Зависимость времени поиска от количества целевых имен
DOI: https://doi.org/10.60797/IRJ.2025.155.8.8

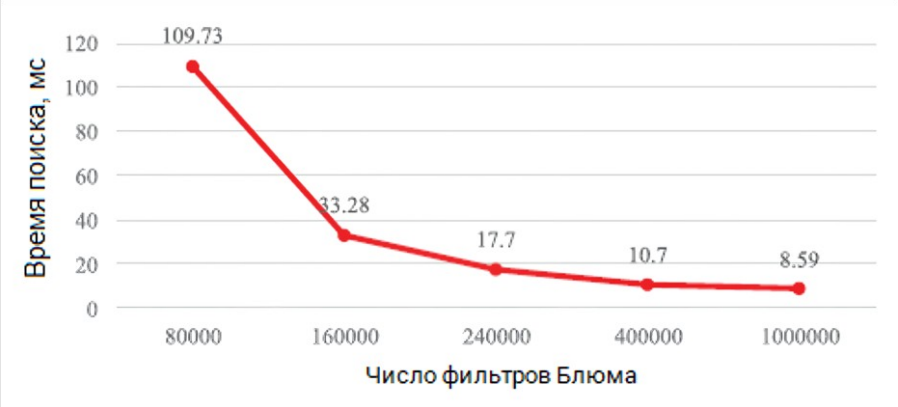

Рисунок 8 - Зависимость времени поиска от количества используемых фильтров Блюма
DOI: https://doi.org/10.60797/IRJ.2025.155.8.9

Для хранения 800 000 префиксов имен в наборе правил требуется 107 МБ памяти, в то время как дополнительные фильтры Блюма занимают всего 78,13 Кбайт в предлагаемом алгоритме, обеспечивая 160000 номеров массива фильтров Блюма, как показано на рис. 8. Принимая во внимание его эффективность, дополнительное потребление памяти не является критической проблемой. В приведенных выше экспериментах предложенный алгоритм, основанный на префиксах функций, может повысить эффективность поиска имени NDN, сохраняя при этом тот же уровень производительности в обычных ситуациях. Основываясь на результатах экспериментов, можно сделать вывод, что эффективность предложенного подхода значительно выше традиционного метода.

Обсуждение

1. В статье предложен комбинированный метод решения задачи поиска и хранения неструктурированных данных большого объема в кэше маршрутизатора контента для ICN-сетей с использованием алгоритмов индексирования и хеширования, отличающийся от существующих уменьшенным временем поиска (латентностью).

2. Предложена модифицированная структура служебных таблиц маршрутизатора контента в ICN-сетях, основанная на дополнительных хеш-таблицах хранилища контента, отличающаяся от существующей архитектуры индексных таблиц более быстрым выполнением операции поиска данных в $\frac{n}{\alpha}$ раз (n — общее число хранимых записей в индексной таблице, α — среднее количество элементов в цепочке коллизий хеш-таблицы).

3. Предложен модифицированный алгоритм поиска с применением информации, хранимой в дополнительных хеш-таблицах, отличающийся на порядок меньшей трудоемкостью (количеством выполняемых операций последовательного перебора при поиске данных) операций последовательного перебора при поиске данных, по сравнению с исходным алгоритмом.

В качестве будущих направлений исследований необходимо изучить применимость фильтров Блюма с проработкой false-positive случаев, а также рассмотреть возможность распараллеливания алгоритмов вставки и поиска контента для многоядерных процессоров.

Заключение

Предложенный в данной статье алгоритм, основанный на использование префиксов объектов и фильтров Блюма, повышает скорость поиска данных в сети NDN, сохраняя при этом тот же уровень производительности вне

зависимости от размерности служебных таблиц, размера кэша на маршрутизаторе контента, длины префиксных слов. Поскольку процессы поиска по фильтру Блюма наименее затратны по вычислительной сложности, чем процессы поиска по хэшу и сопоставления строк, это сокращает время, затрачиваемое на поиск имени в NDN.

Конфликт интересов

Не указан.

Рецензия

Кремлева Э.Ш., Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань Российская Федерация
DOI: https://doi.org/10.60797/IRJ.2025.155.8.10

Conflict of Interest

None declared.

Review

Kremleva E.S., Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan Russian Federation
DOI: https://doi.org/10.60797/IRJ.2025.155.8.10

Список литературы / References

1. Магистральные сети связи России 2024 // ComNews. — URL: https://www.comnews.ru/content/236040/2024-11-29/2024-w48/1180/magistralnye-seti-svyazi-rossii-2024 (дата обращения: 20.01.2025).

2. Интернет вещей, IoT, M2M (мировой рынок) // TAdviser. — URL: https://www.tadviser.ru/index.php/Статья:Интернет_вещей,_IoT,_M2M_(мировой_рынок) (дата обращения: 20.01.2025).

3. Васяева Н.С. Исследование способов коммутации пакетов для коммутаторов Cisco / Н.С. Васяева, М.Н. Дегаев // Инженерные кадры — будущее инновационной экономики России: Материалы VI Всероссийской студенческой конференции. — Йошкар-Ола: ПГТУ, 2021. — С. 28–35.

4. Васяева Н.С. Особенности кэширования и маршрутизации данных в контент-ориентированных сетях / Н.С. Васяева, М.Н. Дегаев // Инженерные кадры — будущее инновационной экономики России: Материалы VIII Всероссийской студенческой конференции. — Йошкар-Ола: ПГТУ, 2022. — С. 404–409.

5. Васяева Н.С. Анализ сетевых архитектур, ориентированных на данные / Н.С. Васяева, М.Н. Дегаев // International Journal of Advanced Studies in Computer Engineering. — 2022. — № 1. — С. 112–124.

6. Taniguchi K. Poster: A Method for Designing High-speed Software NDN Routers / K. Taniguchi, J. Takemasa, Yu. Koizumi [et al.] // Proceedings of ACM ICN. ACM. — 2016. — P. 203–204.

7. Thomas Ya. Object-oriented Packet Caching for ICN / Ya. Thomas, G. Xylomenos, Ch. Tsilopoulos [et al.] // Proceedings of ACM ICN. ACM. — 2015. — P. 89–98.

8. Васяева Н.С. Особенности обработки пакетов в эталонном маршрутизаторе контента для ndn-сетей / Н.С. Васяева, М.Н. Дегаев // Труды Поволжского государственного технологического университета. Сер.: Технологическая. — Йошкар-Ола: ПГТУ, 2024. — Вып. 12. — С. 23–30.

9. Hash Functions all the way down // Aras' Blog. — URL: https://aras-p.info/blog/2016/08/02/Hash-Functions-all-the-way-down (accessed: 20.01.2025).

10. Yi W. NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom fillters / W. Yi, P. Tian, M. Zhian [et al.] // Proceedings of IEEE INFOCOM MiniConference. IEEE. — 2013. — P. 95–99.

11. Chaudhary P. PeNCache: Popularity based cooperative caching in Named Data Networks / P. Chaudhary, N. Hubballi // Computer Networks. — 2025. — Vol. 257. — P. 110995.

12. Дегаев М.Н. Программа для вставки имен на основе префиксов объектов в именованных сетях данных (Named Data Networking, NDN) / М.Н. Дегаев. — Рег. № 2025616238, 13.03.2025.

Список литературы на английском языке / References in English

1. Magistral'nye seti svjazi Rossii 2024 [Backbone communication networks of Russia 2024] // ComNews. — URL: https://www.comnews.ru/content/236040/2024-11-29/2024-w48/1180/magistralnye-seti-svyazi-rossii-2024 (accessed: 20.01.2025). [in Russian]

2. Internet veshhej, IoT, M2M (mirovoj rynok) [Internet of Things, IoT, M2M (global market)] // TAdviser. — URL: https://www.tadviser.ru/index.php/Stat'ja:Internet_veshhej,_IoT,_M2M_(mirovoj_rynok) (accessed: 20.01.2025). [in Russian]

3. Vasjaeva N.C. Issledovanie sposobov kommutacii paketov dlja kommutatorov Cisco [Study of packet switching methods for Cisco switches] / N.S. Vasjaeva, M.N. Degaev // Inzhenernye kadry — budushhee innovacionnoj jekonomiki Rossii: Materialy VI Vserossijskoj studencheskoj konferencii [Engineering Personnel — the Future of Innovative Economy of Russia: Proceedings of the VI All-Russian Student Conference]. — Yoshkar-Ola: PSTU, 2021. — P. 28–35. [in Russian]

4. Vasjaeva N.C. Osobennosti kjeshirovanija i marshrutizacii dannyh v kontent-orientirovannyh setjah [Features of caching and data routing in content-oriented networks] / N.S. Vasjaeva, M.N. Degaev // Inzhenernye kadry — budushhee innovacionnoj jekonomiki Rossii: Materialy VIII Vserossijskoj studencheskoj konferencii [Engineering Personnel — the Future of Innovative Economy of Russia: Proceedings of the VIII All-Russian Student Conference]. — Yoshkar-Ola: PSTU, 2022. — P. 404–409. [in Russian]

5. Vasjaeva N.C. Analiz setevyh arhitektur, orientirovannyh na dannye [Analysing data-centric network architectures] / N.S. Vasjaeva, M.N. Degaev // International Journal of Advanced Studies in Computer Engineering. — 2022. — № 1. — P. 112–124. [in Russian]

6. Taniguchi K. Poster: A Method for Designing High-speed Software NDN Routers / K. Taniguchi, J. Takemasa, Yu. Koizumi [et al.] // Proceedings of ACM ICN. ACM. — 2016. — P. 203–204.

7. Thomas Ya. Object-oriented Packet Caching for ICN / Ya. Thomas, G. Xylomenos, Ch. Tsilopoulos [et al.] // Proceedings of ACM ICN. ACM. — 2015. — P. 89–98.

8. Vasjaeva N.C. Osobennosti obrabotki paketov v jetalonnom marshrutizatore kontenta dlja ndn-setej [Features of packet processing in the reference content router for ndn-networks] / N.S. Vasjaeva, M.N. Degaev // Trudy Povolzhskogo gosudarstvennogo tehnologicheskogo universiteta. Ser.: Tehnologicheskaja [Proceedings of Volga Region State Technological University. Ser.: Technological]. — Yoshkar-Ola: PSTU, 2024. — Iss. 12. — P. 23–30. [in Russian]

9. Hash Functions all the way down // Aras' Blog. — URL: https://aras-p.info/blog/2016/08/02/Hash-Functions-all-the-way-down (accessed: 20.01.2025).

10. Yi W. NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom fillters / W. Yi, P. Tian, M. Zhian [et al.] // Proceedings of IEEE INFOCOM MiniConference. IEEE. — 2013. — P. 95–99.

11. Chaudhary P. PeNCache: Popularity based cooperative caching in Named Data Networks / P. Chaudhary, N. Hubballi // Computer Networks. — 2025. — Vol. 257. — P. 110995.

12. Degaev M.N. Programma dlja vstavki imen na osnove prefiksov ob#ektov v imenovannyh setjah dannyh (Named Data Networking, NDN) [Program for inserting names based on object prefixes in Named Data Networking (NDN)] / M.N. Degaev. — Reg. № 2025616238, 13.03.2025. [in Russian]