Информационные процессы, Том 25, № 3, 2025, стр. 501–512.

ПЕРЕДАЧА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Оценка времени выполнения алгоритмов на спутнике в сетях 5G NTN [1]

А. Э. Шашин[*,**], **М. В. Суслопаров**[*,**], **А. Н. Красилов**[*,**]

[*] *Национальный исследовательский университет «Высшая школа экономики», Москва*
[**] *Институт проблем передачи информации имени А.А. Харкевича Российской академии наук, Москва*

Поступила в редколлегию 01.07.2025 г. Принята 15.10.2025 г.

Аннотация—Спутниковые системы связи на базе низкоорбитальных спутников приобретают все большую популярность. В 2022 году консорциумом 3GPP был опубликован набор спецификаций Release 17, описывающих расширение технологии сетей пятого поколения (5G) для работы в сценариях неназемных сетей (англ.: 5G Non-Terrestrial Networks, 5G NTN), в том числе с возможностью использования низкоорбитальных спутников для передачи данных. В одном из сценариев развертывания систем 5G NTN спутник является базовой станцией и выполняет все функции, связанные с передачей данных и управлением сетью, в том числе выполняет алгоритмы планирования радиоресурсов и выбора параметров передачи. Заметим, что базовая станция сети 5G NTN обслуживает гораздо больше пользователей (сотни-тысячи), чем базовая станция наземных сетей 5G, что существенно увеличивает время выполнения алгоритмов на спутнике. Для снижения времени выполнения алгоритмов могут использоваться устройства с параллельными вычислениями (например, многоядерные процессоры или графические процессоры). В данной работе разработана модель, которая позволяет оценить время выполнения алгоритма на таких устройствах. Показано, как с помощью модели сконфигурировать вычислительное устройство на спутнике (например, выбрать число ядер), чтобы выполнить заданное ограничение на время выполнения алгоритма.

КЛЮЧЕВЫЕ СЛОВА: 5G NTN, BSP, модель вычислений, планировщик радиоресурсов.
DOI: 10.53921/18195822_2025_25_3_501

1. ВВЕДЕНИЕ

Спутниковые системы связи стремительно развиваются. Среди них стоит особо отметить системы широкополосной связи на базе низкоорбитальных спутников (англ.: Low Earth Orbit, LEO). Такие системы связи имеют широкую область покрытия, обеспечивают высокую пропускную способность и значительно меньшую задержку передачи данных по сравнению с другими спутниковыми системами связи. Коммерческий успех таких систем связи, как Iridium, OneWeb и Starlink, подтверждает актуальность задачи развертывания LEO-систем и повышения их производительности [1, 2].

Ключевыми недостатками коммерческих LEO-систем являются, во-первых, закрытость спецификаций, описывающих протоколы передачи данных, что ограничивает возможность разработки телекоммуникационного оборудования сторонними производителями, и, во-вторых, необходимость использования специального терминального оборудования, в том числе антенн и устройств, обеспечивающих обработку сигналов и интеграцию LEO-систем с технологиями

[1] Исследование выполнено за счет гранта Российского научного фонда № 25-19-00659, https://rscf.ru/project/25-19-00659/

наземной связи Wi-Fi/5G, поддерживаемых конечными абонентами. В связи с этим консорциумом 3GPP в 2022 году был опубликован набор спецификаций Release 17, описывающих расширение технологии 5G для работы в сценариях неназемных сетей (англ.: 5G Non-Terrestrial Networks, 5G NTN), в том числе с использованием спутников для передачи данных. Кроме того, спецификации предусматривают возможность прямого обслуживания абонентских устройств 5G спутниковой системой связи и ее интеграцию с наземными сетями 5G.

Спецификации 3GPP рассматривают две основные архитектуры построения сети 5G NTN. В первой архитектуре базовая станция, реализующая протоколы передачи данных, находится на Земле, а спутник является аналоговым ретранслятором сигналов до абонентских устройств. Во второй архитектуре на спутнике реализуется базовая станция (или часть ее функций), которая напрямую выполняет обслуживание абонентских устройств и декодирование передаваемых ими данных. В рамках данной работы рассматривается вторая архитектура, так как она: (а) обеспечивает меньшую задержку передачи данных по соединению базовая станция-абонент, что существенно влияет на эффективность работы алгоритмов управления параметрами передачи, (б) обеспечивает более высокое отношение сигнал/шум, чем в случае первой архитектуры и, тем самым, повышает скорость передачи данных, (в) позволяет передавать данные напрямую между спутниками, что используется, например, для снижения задержки при передаче данных между абонентами, обслуживаемыми двумя соседними спутниками. Согласно открытым источникам коммерческие LEO-системы также используют вторую архитектуру [3].

В сетях 5G NTN аналогично наземным сетям 5G необходимо решать задачу управления радиоресурсами и параметрами передачи данных. Иными словами, на спутнике должны выполняться ряд алгоритмов, осуществляющих решение этих задач в режиме реального времени. Как показано в ряде работ (см. например, [4]), существующие алгоритмы, разработанные для наземных сетей, не могут напрямую использоваться в сетях 5G NTN по причине их высокой вычислительной сложности. В частности, в отличие от наземных сетей, в которых базовая станция обслуживает десятки-сотни пользователей, спутник с многолучевой антенной обслуживает сотни-тысячи пользователей, что приводит к существенному увеличению времени выполнения алгоритмов.

Перспективным подходом для снижения времени выполнения алгоритмов на спутниках является использование параллельных вычислений на таких вычислительных устройствах, как многоядерные процессоры [5], графические процессоры (видеокарты) [6–8] и программируемые логические интегральные схемы (ПЛИС) [9]. При разработке параллельных алгоритмов, а также при выборе параметров вычислительного устройства (например, числа ядер), устанавливаемого на спутнике, важнейшим вопросом является оценка времени выполнения различных алгоритмов на этом вычислительном устройстве. В частности, при заданном числе пользователей, обслуживаемых спутником, время выполнения алгоритма не должно быть больше, чем период запуска этого алгоритма (т.е. алгоритм должен выполняться в режиме реального времени).

Для решения данной задачи в работе разработана модель, которая позволяет оценить время выполнения алгоритма на устройстве с параллельными вычислениями, установленном на спутнике. В качестве примера в работе детально рассматривается задача оценки времени выполнения алгоритма планирования радиоресурсов (планировщика). Сравнение результатов, полученных с помощью модели и на реальном процессоре, показывает высокую точность построенной модели. С помощью модели проводится анализ конфигурации вычислительного устройства, которое должно быть установлено на спутнике, чтобы обеспечить заданное время выполнения алгоритма.

Дальнейшее изложение работы построено следующим образом. В разделе 2 приводится обзор моделей, используемых для оценки времени выполнения алгоритмов на различных ар-

хитектурах вычислительных устройств. В разделе 3 дано описание предложенной модели для оценки времени выполнения алгоритма на устройствах с параллельными вычислениями. В разделе 4 приведен пример использования модели для оценки времени выполнения алгоритма планирования радиоресурсов. Раздел 5 содержит результаты валидации модели и ее применения для конфигурации вычислительного устройства на спутнике. В разделе 6 представлены выводы и дальнейшие направления исследований.

2. ОБЗОР ЛИТЕРАТУРЫ

В литературе представлено множество моделей вычислительных устройств различных типов, предназначенных для оценки времени выполнения алгоритмов. В частности, авторы [10] предложили модель Roofline, определяющую максимальную производительность устройства по двум основным характеристикам: суммарной вычислительной мощности арифметических устройств и скорости доступа к памяти. Данная модель универсальна и не требует глубокого анализа как архитектуры вычислительного устройства, так и исходного кода программы. Однако в случае выполнения алгоритмов/программ, не обеспечивающих максимальную загрузку устройства (либо со стороны вычислений, либо со стороны обращения к памяти), точность модели Roofline резко снижается [11].

В работе [11] предложено оценивать время выполнения программ на устройствах с параллельными вычислениями через анализ низкоуровневого (ассемблерного) кода этих программ. Модель обеспечивает высокую точность оценки времени выполнения за счет детального анализа времени выполнения всех инструкций и связей между ними. Однако модель существенно зависит от архитектуры исследуемого устройства, для которой необходимо, во-первых, замерить время выполнения машинных инструкций с высокой точностью, и, во-вторых, создать систему обработки и анализа кода, также учитывающую одновременное выполнение инструкций за счет конвейеризации.

В работе [12] предложена модель Work Flow Graph (WFG), позволяющая представить программу в виде взвешенного графа, вершины которой соответствуют выполняемым инструкциям, а ребра – задержкам при выполнении инструкций в различных последовательностях. Недостатком данной модели является сложность ее применения, поскольку необходимо реализовать автоматизированную систему обработки исходного кода, позволяющую построить граф для произвольной программы.

Авторы [13] модели GPUMech предлагают для каждой программы составлять репрезентативный набор программных потоков (англ.: the most representative warp), который определяет время выполнения всей программы. Для работы модели GPUMech требуется запуск программы на целевом вычислительном устройстве, что не всегда возможно при проектировании новых устройств (например, для устройств перед запуском на орбиту).

Имитационные модели многопоточных вычислительных устройств, разработанные в [14,15], позволяют строить модели высокой точности за счет моделирования выполнения отдельных машинных инструкций. Однако получение результатов с помощью таких моделей требует значительного времени и больших объемов вычислительных ресурсов, особенно в случае перебора различных конфигураций вычислительного устройства.

Таким образом, описанные выше модели имеют следующие недостатки: (а) имеют слишком высокий уровень детализации (рассматривают отдельные инструкции), что приводит к необходимости анализа низкоуровневого кода, (б) используют ряд допущений (например, Roofline предполагает работу в режиме насыщения), что приводит к их низкой точности в сценариях, где эти допущения не выполняются. Поэтому в рамках данной работы разработана модель устройства с параллельными вычислениями, которая, с одной стороны, не анализирует исход-

ный код программы, а, с другой стороны, учитывает влияние задержек, приходящихся как на вычисления, так и на обращение к памяти.

3. ОПИСАНИЕ МОДЕЛИ

Для построения модели вычислительного устройства на спутнике используется подход, предложенный в работе [16] для моделирования многопоточных вычислительных систем Bulk-Synchronous Parallel (BSP). На основе данного подхода предлагается моделировать выполнение алгоритма на устройстве в виде последовательности следующих шагов.

1. Загрузка кода алгоритма (инструкций) на вычислительное устройство. Обозначим время загрузки как τ. Данное время в основном зависит от скорости шины и, как правило, много меньше времени выполнения самого алгоритма.
2. Выполнение инструкций, включающих в себя арифметические операции (такие как операция совмещенных суммы и умножения FMADD) и операции доступа к локальной памяти (например, L1-кэш ядра процессора). Обозначим количество тактов процессора, затраченных на выполнение арифметических операций как $Comp$, а количество тактов на обращение к локальной памяти как $Comm_{LM}$.
3. Обмен данных между ядрами процессора через соединения между ними (например, шины между ядрами) с использованием глобальной памяти, общей для всех ядер. Обозначим количество тактов процессора, затраченных на обращение к глобальной памяти, как $Comm_{GM}$.
4. Синхронизация данных (например, барьерная).

Полное время выполнения алгоритма можно оценить следующим образом:

$$T = \tau + \frac{N_{th} \cdot (Comp + Comm_{LM} + Comm_{GM})}{f \cdot C \cdot \lambda}, \tag{1}$$

где N_{th} – число параллельных потоков в алгоритме, f — средняя тактовая частота ядер процессора, C — количество ядер, λ — коэффициент поправки, зависящий от оптимизаций кода компилятором.

Рассмотрим подробнее, как оценить время доступа к памяти. Так как архитектура памяти и время доступа к ней зависит от типа вычислительного устройства, далее будем рассматривать случай использования многоядерного процессора (например, архитектуры ARM). Пусть ld — это количество операций чтения данных из памяти, st — количество операций записи в память, а g — количество тактов процессора для выполнения операции (допустим, что чтение и запись занимают одинаковое количество тактов). Операции чтения и записи в локальной памяти (кэш уровней L1, L2) задаются нижним индексом LM, а операции с глобальной памятью (кэш уровня L3, общий для всех ядер) задаются индексом GM. Также определим количество успешных доступов к локальному и глобальному кэшу как L_{LM} и L_{GM} соответственно (англ.: cache hits). Тогда $Comm_{LM}$ и $Comm_{GM}$ можно оценить следующим образом:

$$Comm_{LM} = (ld_{LM} + st_{LM}) \cdot g_{LM},$$

$$Comm_{GM} = (ld_{GM} + st_{GM}) \cdot g_{GM} + L_{LM} \cdot g_{LM} + L_{GM} \cdot g_{GM}.$$

В приведенной выше модели количество арифметических операций и операций обращения к памяти зависят от конкретного рассматриваемого алгоритма. В разделе 4 будет рассмотрен пример алгоритма планирования радиоресурсов для сетей 5G NTN, и будет показано, как получить функциональную зависимость времени выполнения алгоритма от объема входных данных и набора выполняемых арифметических операций. Подстройка параметров модели (коэффициентов зависимости), в свою очередь, осуществляется на основе экспериментов с реальным вычислительным устройством.

Алгоритм 1 Алгоритм планирования радиоресурсов

Входные данные: $\{B_n\}, \{r_n^{(j)}\}, \{\widehat{r}_n\}$
Выходные данные: $\{A^{(j)}\}, \{\widehat{r}_n\}$

$\forall n : D_n \leftarrow 0$
$\forall j : A^{(j)} \leftarrow NONE$
▷ Шаг 1
for $n = 1$ to N **do**
  for $j = 1$ to R **do**
    $m_n^{(j)} \leftarrow \frac{r_n^{(j)}}{\widehat{r}_n}$
▷ Шаг 2
for $j = 1$ to R **do**
  $\mathbf{S^{(j)}} \leftarrow sortIndex(\{m_n^{(j)}\})$
▷ Шаг 3
for $j = 1$ to R **do**
  $i \leftarrow 0$
  $n^* \leftarrow \mathbf{S^{(j)}}[i]$
  while $(B_{n^*} = 0)$ & $(i < N)$ **do**
    $i \leftarrow i + 1$
    $n^* \leftarrow \mathbf{S^{(j)}}[i]$
  if $i = N$ **then**
    break
  $A^{(j)} \leftarrow n^*$
  if $B_{n^*} > r_{n^*}^{(j)}$ **then**
    $D_{n^*} \leftarrow D_{n^*} + r_{n^*}^{(j)}$
    $B_{n^*} \leftarrow B_{n^*} - r_{n^*}^{(j)}$
  else
    $D_{n^*} \leftarrow D_{n^*} + B_{n^*}$
    $B_{n^*} \leftarrow 0$
▷ Шаг 4
for $n = 1$ to N **do**
  $\widehat{r}_n \leftarrow \beta D_n + (1 - \beta)\widehat{r}_n$
return $\{A^{(j)}\}_{j=1}^{R}, \{\widehat{r}_n\}_{n=1}^{N}$

4. АЛГОРИТМ ПЛАНИРОВАНИЯ РАДИОРЕСУРСОВ

4.1. Описание алгоритма планирования радиоресурсов

В качестве примера алгоритма, для которого может быть получена оценка времени вычисления, в данной работе рассматривается широко известный в литературе пропорционально-честный (англ.: Proportional Fair, PF) алгоритм планирования радиоресурсов на базовой станции.

В сетях 5G частотные ресурсы разделены на R ресурсных блоков (РБ), а время поделено на слоты одинаковой длительности. Пусть в рассматриваемом временном слоте базовая станция обслуживает N пользователей, а пользователь с номером n имеет данные на передачу объемом B_n. На основе измерений канальных условий базовая станция для всех пользователей и РБ определяет объем данных $r_n^{(j)}$, который может быть передан пользователю n в РБ j.

Рассмотрим подробнее алгоритм назначения РБ пользователям (см. алгоритм 1). Алгоритм 1 получает на вход значения B_n, $r_n^{(j)}$, а также средний объем данных $\widehat{r}_n$, передаваемых каждым пользователем n в одном временном слоте. В качестве результата алгоритм возвращает индексы пользователей $A^{(j)}$, назначенных в каждом РБ j (значение $NONE$ соответствует тому, что ни один пользователь не назначен в данном РБ), а также значения $\widehat{r}_n$, обновленные с учетом текущего назначения.

Алгоритм 1 состоит из четырех шагов.

1. На **шаге 1** (строки 4–6) для каждого пользователя n и РБ j рассчитывается метрика качества РБ $m_n^{(j)}$ для каждого пользователя.
2. На **шаге 2** (строки 8–9) для каждого РБ j алгоритм сортирует пользователей по убыванию метрики $m_n^{(j)}$. В результате получаются отсортированные массивы индексов пользователей $\mathbf{S^{(j)}}$.
3. На **шаге 3** (строки 11–25) алгоритм осуществляет назначение пользователей, последовательно рассматривая РБ. Для этого в каждом РБ j с помощью массива $\mathbf{S^{(j)}}$ в данный РБ назначается пользователь n^* с наибольшей метрикой и ненулевым размером буфера. В случае, если объем данных $r_{n^*}^{(j)}$, передаваемых в данном РБ, не превышает размер буфера B_{n^*}, размер буфера уменьшается на величину $r_{n^*}^{(j)}$. В противном случае размер буфера становится равным нулю. Также алгоритм обновляет суммарный объем данных D_{n^*}, который может быть передан с использованием назначенных в данном слоте РБ для данного пользователя. В случае, когда размеры буферов всех пользователей становятся равными нулю, алгоритм переходит к шагу 4.
4. На **шаге 4** (строки 27–28) алгоритм использует метод экспоненциального скользящего среднего с коэффициентом β для обновления значений $\widehat{r}_n$ каждого пользователя n с учетом объема данных D_n, передаваемых в назначенных РБ ($D_n = 0$, если алгоритм не назначил пользователю ни одного РБ).

Далее проведем анализ вычислительной сложности алгоритма для каждого шага с использованием модели, описанной в разделе 3. Полное время выполнения алгоритма равно сумме времен выполнения всех шагов.

4.2. Анализ времени выполнения шага 1

На шаге 1 вычисление каждого значения $m_n^{(j)}$ может осуществляться независимо, поэтому количество параллельных потоков вычислений $N_{th} = R \cdot N$. Время выполнения операции в одном потоке, а также время обращения к локальной и глобальной памяти не зависят от R и N: $Comp = c_1$, $Comm_{LM} = c_2$, $Comm_{GM} = c_3$, где c_i – количество тактов процессора для выполнение операций вычисления и/или обращения к памяти. Согласно (1) общее время выполнения шага 1 в зависимости от размера решаемой задачи (параметров R и N) и параметров рассматриваемого вычислительного устройства (f и C) может быть оценено следующим образом:

$$T_1(N, R, f, C) = \tau_1 + \alpha_1 \frac{RN}{fC}, \tag{2}$$

где $\alpha_1 = (c_1+c_2+c_3)/\lambda$. Коэффициенты τ_1 и α_1 могут быть найдены с помощью аппроксимации функции (2) по измерениям, полученным на реальном процессоре.

Заметим, что глобальная память будет использоваться только в случае, когда объема локальной памяти недостаточно для размещения всех данных, необходимых для выполнения шага 1. Суммарный объем данных, требуемых для выполнения шага 1, можно оценить как:

$$M_1 = R \cdot N \cdot sizeof(r_n^{(j)}) + R \cdot N \cdot sizeof(m_n^{(j)}) + N \cdot sizeof(\widehat{r}_n),$$

где $sizeof(\cdot)$ — объем памяти, необходимый для хранения соответствующей переменной. При $R \gg 1$ (для сетей 5G типичное значение $R \sim 100$):

$$M_1 \approx RN(sizeof(r_n^{(j)}) + sizeof(m_n^{(j)})).$$

Обозначим M_{L2} — размер локальной памяти (L2 кэша) в байтах. Предположим, что все переменные используемые в алгоритме занимают одинаковый объем памяти d в байтах (например,

float в случае значений с плавающей точкой и *int* в случае целых чисел). Тогда глобальная память на шаге 1 будет использоваться при выполнении следующего условия:

$$RN > \frac{M_{L2}}{2d}. \tag{3}$$

В зависимости от выполнения условия (3), изменяется соотношение между временами использования локальной и глобальной памяти ($Comm_{LM}$ и $Comm_{GM}$), поэтому должны быть найдены два разных набора коэффициентов τ_1 и α_1.

4.3. Анализ времени выполнения шага 2

На шаге 2 осуществляется сортировка пользователей по метрике в различных РБ. Заметим, что сортировки массивов индексов, которые относятся к разным РБ, можно осуществлять независимо на разных ядрах вычислительного устройства, т.е. количество потоков вычислений $N_{th} = R$. Количество операций сравнения и обращений к памяти для сортировки массива размера N с помощью метода быстрой сортировки (англ.: quicksort), который рассматривается в данной работе, пропорционально $N \log N$. Таким образом, время выполнения шага 2 можно оценить как:

$$T_2(N, R, f, C) = \tau_2 + \alpha_2 \frac{RN \log N}{fC}, \tag{4}$$

где коэффициенты τ_2 и α_2 могут быть найдены с помощью аппроксимации функции (4) по измерениям, полученным на реальном процессоре.

Аналогично шагу 1, использование глобальной памяти зависит от объема памяти, требуемого для выполнения шага 2. Его можно оценить как:

$$M_2 = R \cdot N \cdot sizeof(n) + R \cdot N \cdot sizeof(m_n^{(j)}) + \gamma N \cdot sizeof(m_n^{(j)}),$$

где γ — коэффициент использования дополнительной памяти для выполнения сортировки. Можно показать, что при $R \gg 1$ условие использования глобальной памяти совпадает с условием (3).

4.4. Анализ времени выполнения шага 3

Так как устройства могут иметь произвольный размер буфера, то назначение РБ на шаге 3 выполняется последовательно, поэтому число потоков вычислений $N_{th} = 1$ и используется одно вычислительное ядро $C = 1$. Процедура назначения пользователя в каждом РБ можно разбить на два этапа. На первом этапе выполняется поиск пользователя с ненулевым размером буфера. Допустим, что среднее количество пользователей с пустым буфером составляет βN (где $\beta < 1$). Тогда количество операций сравнений и обращений к памяти на этом этапе линейно зависит от N. На втором этапе выполняются обновления B_n и D_n, время выполнения этих операций не зависит от N при рассмотрении одного РБ. Таким образом, общее время выполнения шага 3 может оценить как:

$$T_3(N, R, f) = \tau_3 + \alpha_3 \frac{RN}{f}. \tag{5}$$

Использование глобальной памяти зависит от объема памяти, требуемого для выполнения шага 3, который можно оценить как:

$$M_3 = R \cdot N \cdot sizeof(r_n^{(j)}) + R \cdot N \cdot sizeof(n) + N \cdot sizeof(B_n) + N \cdot sizeof(D_n).$$

При $R \gg 1$ условие использования глобальной памяти совпадает с условием (3).

Таблица 1. Параметры моделирования

Параметр	Значение
Несущая частота	3,6 ГГц
Ширина канала	50 МГц
Общее количество РБ R	256
Ширина одного РБ W	180 кГц
Длина слота T_{slot}	1 мс
β	0,01
Длительность эксперимента	10 с
Количество независимых прогонов	5

4.5. Анализ времени выполнения шага 4

На шаге 4 каждое обновление средней скорости пользователя n может выполнятся независимо, поэтому количество потоков $N_{th} = N$. Количество арифметических операций и обращений к памяти в рамках одного потока не зависит от N. Поэтому время выполнения шага 4 можно оценить как:

$$T_4(N, f, C) = \tau_4 + \alpha_4 \frac{N}{fC}. \tag{6}$$

Суммарный объем памяти, используемый алгоритмом, можно оценить как:

$$M_4 = N \cdot sizeof(\widehat{r}_n) + N \cdot sizeof(D_n).$$

Таким образом, условие на использование глобальной памяти может быть записано как:

$$N > \frac{M_{L2}}{2d}. \tag{7}$$

Заметим, что условие (3) является более строгим, чем условие (7), поэтому при больших R и большом объеме локальной памяти ($M_{L2} > 10$ Кбайт) глобальная память будет использоваться только на шагах 1–3.

5. ЧИСЛЕННЫЕ РЕЗУЛЬТАТЫ

5.1. Описание сценария

Для анализа времени выполнения алгоритма планирования радиоресурсов была разработана имитационная модель системы 5G NTN в среде ns-3 [17]. В рассматриваемом сценарии одна базовая станция обслуживает N пользователей в одном луче. Затухание сигнала в канале моделируется в соответствии с рекомендациями 3GPP [18]. Для моделирования Рэлеевских замираний при многолучевом распространении используются модели Extended Pedestrian A (EPA, 3 км/ч) [19]. Основные параметры моделирования приведены в таблице 1.

Для каждого пользователя в нисходящем канале генерируется насыщенный трафик, т.е. в каждом временном слоте есть данные на передачу каждому пользователю. В каждом слоте базовая станция выполняет алгоритм планирования радиоресурсов, описанный в разделе 4.1. Объем данных $r_n^{(j)}$, который может быть передан пользователю n в РБ j, оценивается с помощью формулы Шеннона:

$$r_n^{(j)} = T_{slot} W \log_2(1 + s_n^{(j)}), \tag{8}$$

где T_{slot} — длительность временного слота, W — ширина одного РБ по частоте, $s_n^{(j)}$ — отношение сигнал/шум пользователя n в РБ j.

Таблица 2. Аппроксимации τ для различных шагов алгоритма

Параметр	Условие	Значение, мкс
τ_1	$N \leqslant 250$	0,598
	$N > 250$	0
τ_2	$N \leqslant 250$	0
	$N > 250$	89,01
τ_3	$N \leqslant 250$	2,34
	$N > 250$	3,03
τ_4	$N \leqslant 6{,}4 \cdot 10^4$	0,055

Таблица 3. Аппроксимации α для различных шагов алгоритма

Параметр	Условие	Значение, мкс·Гц
α_1	$N \leqslant 250$	3,64
	$N > 250$	3,99
α_2	$N \leqslant 250$	53,1
	$N > 250$	51,6
α_3	$N \leqslant 250$	0,03
	$N > 250$	0,012
α_4	$N \leqslant 6{,}4 \cdot 10^4$	0,823

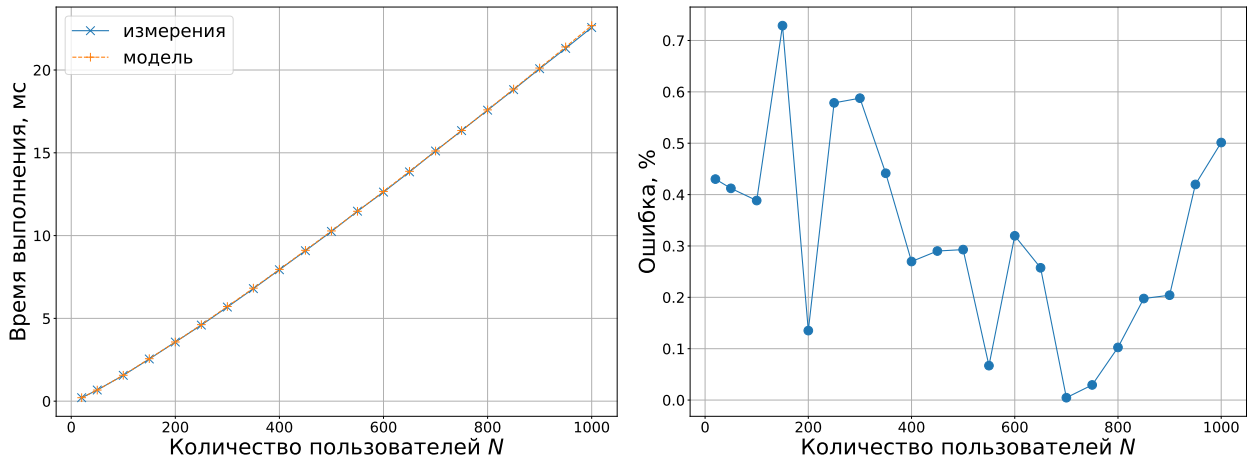
5.2. Валидация модели

В разделе 4 представлены зависимости времени выполнения различных шагов алгоритма от параметров сценария (в данном случае R фиксировано, а N варьируется в ходе эксперимента) и параметров вычислительного устройства (f и C). Для определения коэффициентов в этих зависимостях были проведены эксперименты, в рамках которых измерялось время выполнения каждой функции при варьировании числа пользователей N. Вычисления проводились на одном ядре процессора AMD Ryzen 7 3700X с фиксированной таковой частотой $f = 1800$ МГц. Как показано в разделах 4, параметры функций T_i должны быть разными при разных значениях N. Используя неравенство (3) при заданных параметрах модели и вычислительного устройства ($R = 256$, $M_{L2} = 512$ Кбайт, $d = 4$ байта), получаем, что глобальная память будет использоваться для шагов алгоритма 1–3 при $N > 250$ пользователей. Для шага 4 согласно неравенству (7) при рассматриваемом числе пользователей будет использоваться только локальная память. Измеренные на реальном процессоре значения времени выполнения шагов были аппроксимированы методом наименьших квадратов. Полученные коэффициенты в выражениях (2)–(6) приведены в таблицах 2 и 3.

Далее была проведено сравнение времени выполнения всего алгоритма, полученного с помощью модели, с результатами, полученными на реальном процессоре (см. рис. 1(а)) На рис. 1 (б) приведена зависимость относительной ошибки модели. Результаты показывают, что предложенная модель позволяет оценивать время выполнения алгоритма с высокой точностью (относительная ошибка не превосходит 1%).

5.3. Конфигурация вычислительного устройства

Результаты, приведенные на рис. 1(а), показывают, что при $N > 50$ время выполнения алгоритма на одном ядре превышает длительность временного слота $T_{slot} = 1$ мс. В этом случае алгоритм не может выполняться в реальном времени, так как планирование радиоресурсов необходимо осуществлять в каждом слоте. Для выполнения алгоритма в реальном времени необходимо использовать параллельные вычисления на многоядерном вычислительном



(а) Время выполнения алгоритма (б) Относительная ошибка модели

Рис. 1. Валидация модели выполнения алгоритма планирования радиоресурсов.

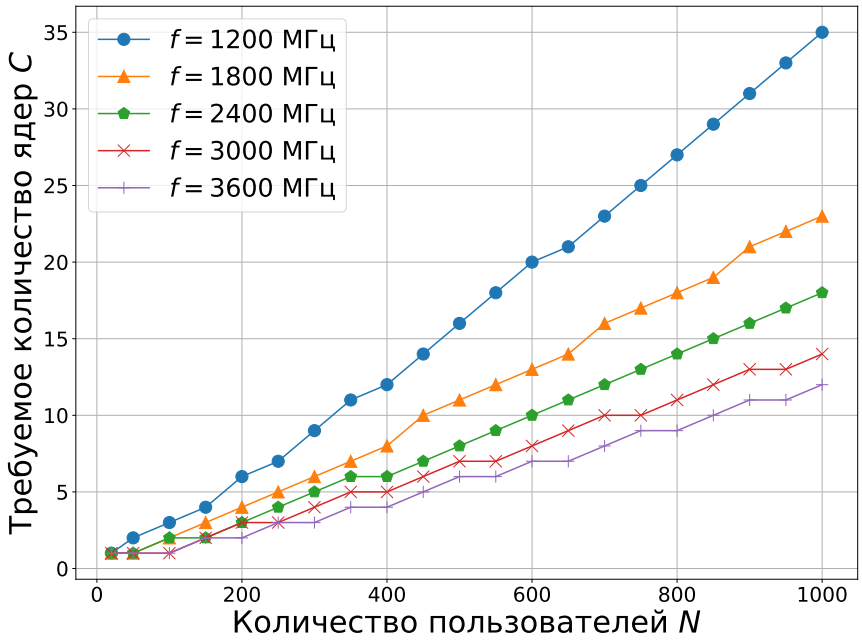


Рис. 2. Число ядер параллельного вычислителя, необходимое для выполнения алгоритма планирования радиоресурсов за время T_{slot}.

устройстве. Используя предложенную модель, можно найти параметры устройства (тактовую частоту f и количество ядер C) в зависимости от обслуживаемого количества пользователей. На рис. 2 представлена зависимость минимального числа ядер C, необходимого для того, чтобы время выполнения алгоритма планирования радиоресурсов не превышало T_{slot}, при разных значениях f. Результаты показывают, что для обслуживания сотен-тысяч пользователей в сетях 5G NTN необходимо использовать параллельное вычислительное устройство с десятками ядер.

6. ЗАКЛЮЧЕНИЕ

В данной работе разработана модель для оценки времени выполнения алгоритмов на устройствах с параллельными вычислениями. Модель была использована для анализа времени выполнения алгоритма планирования радиоресурсов на спутниковой базовой станции в сетях 5G NTN. Сравнение времени выполнения данного алгоритма, полученное с помощью разработанной модели и с помощью измерений на реальном процессоре, показывает высокую точность построенной модели. С помощью разработанной модели был проведен анализ конфигурации вычислительного устройства, необходимого для выполнения алгоритма планирования радиоресурсов в реальном времени. В частности, было показано, что для обслуживания характерного для спутниковых систем связи числа пользователей (сотни-тысячи пользователей) на спутник должно быть установлено вычислительное устройство с десятками ядер.

В дальнейшем планируется использовать разработанную модель для оценки времени выполнения различных алгоритмов, используемых в сетях 5G NTN, таких как алгоритм планирования радиоресурсов для спутника с многолучевой антенной, алгоритм выбора сигнально-кодовой конструкции (например, [20]), алгоритм предсказания качества канала и др.

СПИСОК ЛИТЕРАТУРЫ

1. Моделирование процессов обслуживания абонентов в сети передачи данных на базе космических аппаратов на низкой круговой орбите. I / Маслов А.А., Себекин Г.В., Степанов М.С., Степанов С.Н. и Щурков А.О. // Информационные процессы. — 2024. — Т. 24, № 4. — С. 335–349.
2. Моделирование процессов обслуживания абонентов в сети передачи данных на базе космических аппаратов на низкой круговой орбите. II / Маслов А.А., Себекин Г.В., Степанов М.С., Степанов С.Н. и Щурков А.О. // Информационные процессы. — 2025. — Т. 25, № 2. — С. 151–168.
3. System and method of providing a medium access control scheduler. — 2023. — Oct. 24. — US Patent 11,800,552.
4. On the Path to 6G: Embracing the Next Wave of Low Earth Orbit Satellite Access / Lin Xingqin, Cioni Stefano, Charbit Gilles, Chuberre Nicolas, Hellsten Sven, and Boutillon Jean-Francois // IEEE Communications Magazine. — 2021. — Vol. 59, no. 12. — P. 36–42.
5. Efficient In-Place Hough Transform Algorithm for Arbitrary Image Sizes / Kazimirov D.D., Nikolaev D.P., Rybakova E.O., and Terekhin A.P. // Problems of Information Transmission. — 2024. — Vol. 60, no. 4. — P. 363–391.
6. GPU4S: Embedded GPUs in Space / Kosmidis Leonidas, Lachaize Jérôme, Abella Jaume, Notebaert Olivier, Cazorla Francisco, and Steenari David // 2019 22nd Euromicro Conference on Digital System Design (DSD). — 2019. — P. 399–405.
7. Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space / Bruhn Fredrik, Tsog Nandinbaatar, Kunkel Fabian, Flordal Oskar, and Troxel Ian // CEAS Space Journal. — 2020. — Vol. 12, no. 4. — P. 551–564.
8. GPU@ SAT DevKit: Empowering edge computing development onboard satellites in the space-IoT era / Benelli Gionata, Todaro Giovanni, Monopoli Matteo, Giuffrida Gianluca, Donati Massimiliano, and Fanucci Luca // Electronics. — 2024. — Vol. 13, no. 19. — P. 3928.
9. Wu Keyu, Wu Fengge, Zhao Junsuo. Optimization Strategy of Satellite Signal Processing Software based on Feiteng FT-2000 Computing Board // 2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS) / IEEE. — 2021. — P. 208–215.
10. Williams Samuel, Waterman Andrew, Patterson David. Roofline: an insightful visual performance model for multicore architectures // Commun. ACM. — 2009. — Apr. — Vol. 52, no. 4. — P. 65–76. — Access mode: https://doi.org/10.1145/1498765.1498785.
11. Volkov Vasily. Understanding Latency Hiding on GPUs : PhD dissertation ; University of California, Berkeley. — 2016.

12. An adaptive performance modeling tool for GPU architectures / Baghsorkhi Sara, Delahaye Matthieu, Patel Sanjay, Gropp William, and Hwu Wen-mei // Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. — New York, NY, USA : Association for Computing Machinery. — 2010. — PPoPP '10. — P. 105–114. — Access mode: https://doi.org/10.1145/1693453.1693470.

13. GPUMech: GPU Performance Modeling Technique Based on Interval Analysis / Huang Jen-Cheng, Lee Joo Hwan, Kim Hyesoon, and Lee Hsien-Hsin // 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture. — 2014. — P. 268–279.

14. Analyzing CUDA workloads using a detailed GPU simulator / Bakhoda Ali, Yuan George, Fung Wilson, Wong Henry, and Aamodt Tor // 2009 IEEE International Symposium on Performance Analysis of Systems and Software. — 2009. — P. 163–174.

15. PPT-GPU: Scalable GPU Performance Modeling / Arafa Yehia, Badawy Abdel-Hameed, Chennupati Gopinath, Santhi Nandakishore, and Eidenbenz Stephan // IEEE Computer Architecture Letters. — 2019. — Vol. 18, no. 1. — P. 55–58.

16. Valiant Leslie. A bridging model for parallel computation // Commun. ACM. — 1990. — Aug. — Vol. 33, no. 8. — P. 103–111. — Access mode: https://doi.org/10.1145/79173.79181.

17. The ns-3 Network Simulator, http://www.nsnam.org/.

18. TR 38.811: Study on New Radio (NR) to support non-terrestrial networks; V15.4.0. — 2020. — October.

19. Base Station (BS) Radio Transmission and Reception : Technical specification (TS) : 36.104 / 3rd Generation Partnership Project (3GPP) : 2025.

20. Reducing the Complexity of the Layer Scheduled LDPC Decoder Based on the Information Bottleneck Method / Melnikov I.A., Uglovskii A.Y., Kreshchuk A.A., Kureev A.A., and Khorov E.M. // Problems of Information Transmission. — 2024. — Vol. 60, no. 3. — P. 199–208.

Estimation of algorithm execution time at satellite in 5G NTN systems

A. Shashin, M. Susloparov, and A. Krasilov

Satellite communication systems based on low Earth orbit satellites are becoming increasingly popular. In 2022, 3GPP published Release 17 specifications, which describe the extension of the fifth-generation cellular systems (5G) to support 5G Non-Terrestrial Networks (NTN), including the possibility of using low Earth orbit satellites for data transmission. In one of the considered 5G NTN deployment scenarios, the satellite is a base station that performs all functions related to data transmission and network management, including executing algorithms for radio resource management and selecting transmission parameters. Note that, unlike terrestrial 5G systems, a 5G NTN base station will serve a much higher number of users (from hundreds to thousands), which significantly increases the execution time of algorithms at the satellite. Parallel computing devices (e.g., multi-core processors or graphics processing units) can be used to reduce algorithm execution time. The paper develops a model for estimating algorithm execution time on such devices. It is shown how to use the model to configure a satellite computing device (e.g., select the number of cores) to satisfy a given constraint on the algorithm execution time.

KEYWORDS: 5G NTN, BSP, computational model, scheduler.