УДК 51-74

СРАВНЕНИЕ МЕТОДОВ БАЛАНСИРОВКИ НАГРУЗКИ В БЕСПРОВОДНЫХ СЕНСОРНЫХ СЕТЯХ

ВЛАСОВА В.А., ОЛЕЙНИКОВА А.А.

Рассматриваются основные методы балансировки нагрузки в телекоммуникационных сетях. На основе сравнительного анализа и применимости каждого из них для использования в беспроводных сенсорных сетях (БСС) рассчитывается функция полезности и выбирается предпочтительный метод для эффективного применения в автономных сетях с невысокой вычислительной мощностью узлов. Полученный метод был адаптирован для применения в БСС.

Ключевые слова: беспроводная сенсорная сеть, балансировка нагрузки, функция полезности

Key words: wireless sensor network, load balancing, utility function

1. Введение

Балансировка нагрузки в традиционных сетях связи нацелена на оптимизацию использования ресурсов, максимизацию пропускной способности, уменьшение времени отклика и предотвращение перегрузки какого-либо одного ресурса. Благодаря алгоритмам балансировки достигается более высокая пропускная способность и улучшается время отклика в распределенных системах.

Актуальной является задача балансировки трафика в беспроводных сенсорных сетях, так как узлы, находящиеся в окрестности базовых станций, вынуждены ретранслировать информацию от оконечных узлов, что истощает заряд батареи и существенно сокращает «время жизни» сети из-за нарушения связности. Сложность в выборе подходящего алгоритма состоит в том, что специфика БСС накладывает жесткие ограничения на вычислительные мощности и требует высокой энергоэффективности.

Целью данного исследования является выбор метода балансировки с помощью многокритериальной оптимизации.

Задачи исследования:

– обзор и сравнение характеристик существующих методов балансировки нагрузки;

– выбор предпочтительного метода балансировки на основе полученных характеристик с использованием функции полезности.

2. Обзор методов балансировки нагрузки

В Round Robin Scheduling (управление задачами в системах с распределением времени) алгоритм определяет кольцо как очередь и квант фиксированного времени. Каждое задание может быть выполнено только в этот квант времени и в свою очередь. Если задача не может быть завершена в течение одного кванта, она вернется в очередь на ожидание следующего круга. Однако существует проблема определения подходящего кванта времени. Когда квант времени очень большой, то RR алгоритм планирования работает так же, как FCFS Scheduling. А когда квант времени слишком мал, то Round Robin Scheduling известен как Processor Sharing алгоритм.

Max-Min Algorithm. Выясняется сначала минимальное время выполнения задач, выбирается максимальное значение, которое является максимальным временем среди всех задач на всех ресурсах. Далее задача с найденным максимальным временем назначается на выполнение на конкретно выбранный узел. Затем перерасчитывается время выполнения всех заданий на этом узле путем добавления времени выполнения поставленной задачи ко времени выполнения других задач на этом узле. Затем поставленная задача удаляется из списка системы.

Алгоритм Compare and Balance используется для достижения равновесного состояния и управления несбалансированной нагрузкой системы. В этом алгоритме на основе вероятности (номер виртуальной машины, запущенной на текущем хосте и всей облачной системы) текущий хост случайным образом выбирает хост и сравнивает их нагрузку. Если нагрузка текущего хоста больше выбранного он передает дополнительную нагрузку на этот конкретный узел. Затем каждый узел системы выполняет ту же процедуру. Этот алгоритм балансировки нагрузки разработан и реализован для уменьшения времени миграции виртуальных машин. Общая память используется также для этой цели.

Ant Colony Optimization – это распределенный алгоритм. В этом алгоритме информация о ресурсах динамически обновляется при каждом движении муравьев. Множественные колонии муравьев описываются таким образом, что узел посылает цветные колонии по всей сети. Раскрашенные колонии муравьев используются для предотвращения движения муравьев из одного гнезда, следующих одним маршрутом, а также обеспечения их распределения по всем узлам в системе, где каждый муравей действует как мобильный агент, который несет обновленную информацию балансировки нагрузки в следующий узел.

В алгоритме Shortest Response Time First главной идеей является прямая переадресация. В нем каждому процессу назначается приоритет для его запуска. В процессы с равными приоритетами планируется в FCFS порядке. SJF алгоритм является частным случаем общего алгоритма приоритетного планирования. В алгоритме SJF приоритет является обратным по отношению к следующему всплеску процессора (CPU). Это означает, что если всплеск процессора увеличивается, то приоритет понижается. Политика SJF выбирает

задачу с кратчайшим временем обработки. В этом алгоритме короткие задачи выполняются перед длинными задачами. В SJF очень важно знать или оценить время обработки каждого задания, что является главной проблемой SJF.

Active Clustering load balancing Algorithm – это алгоритм самоагрегации, оптимизирует задачи, подключив похожие услуги с использованием локального переприсваивания. Работает на основе группирования похожих узлов. Процесс группирования основан на концепции рефери-узла. Рефери-узел образует соединение между соседями, которое подобно инициализирующему узлу. Затем рефери-узел разрывает соединение между собой и начальным узлом. Далее совокупность процессов снова и снова повторяется. Производительность системы возрастает на основе высокой доступности ресурсов, из-за этого пропускная способность также увеличивается.

ACCLB – это методика балансировки загрузки, основанная на муравьиной колонии и теории сложной сети (ACCLB) в открытых облачных вычислениях. Она использует низкоуровневые и безмасштабные характеристики сложной сети, чтобы добиться лучшего распределения нагрузки. Эта методика позволяет преодолеть неоднородность, является адаптивной к динамичной среде, превосходна по устойчивости к ошибкам и имеет хорошую масштабируемость, следовательно, помогает в улучшении производительности системы.

Join-Idle-Queue – алгоритм балансировки нагрузки для динамически масштабируемых веб-сервисов, обеспечивает масштабную балансировку нагрузки по распределенным отправителям. Сначала вычисляет доступность простаивающих процессоров в каждом отправителе, а затем назначает задания процессорам для уменьшения средней длины очереди в каждом процессоре. При удалении задач балансировки нагрузки из критического пути обработки запроса он эффективно снижает нагрузку системы, не несет никакой коммуникационной нагрузки на вновь прибывшие задачи и не увеличивает фактическое время отклика.

Central queuing – алгоритм работает по принципу динамического распределения. Каждая новая задача прибывает к менеджеру очередей и становится в очередь. Когда запрос для выполнения задачи принимается менеджером очередей, он удаляет первую задачу из очереди и передает ее запрашивающей стороне. Если в очереди нет готовых задач, то запрос буферизируется, пока новая задача не будет доступна. Но в случае записи новой задачи в очередь, пока есть не отвеченные запросы в очереди, первый такой запрос удаляется из очереди, а новая задача ставится перед ней. Когда загрузка процессора падает ниже порогового значения, то локальный менеджер загрузки посылает запрос на новую задачу центральному менеджеру загрузки. Затем центральный менеджер отвечает на запрос, если найдена готовая задача, в противном случае соблюдается очередность запросов до поступления новой задачи.

Connection mechanism – это алгоритм балансировки нагрузки, основан на механизме наименьшего количества соединений, который является частью динамического алгоритма планирования. Он необходим для подсчета количества соединений для каждого сервера и динамической оценки нагрузки. Балансировщик нагрузки записывает количество соединений каждого сервера. Количество соединений увеличивается, когда новое соединение отправляется к серверу, и уменьшается, когда соединение завершается или происходит прерывание соединения.

Least connections – алгоритм минимума соединений, он посылает запросы на сервер, который в настоящее время обслуживает наименьшее количество подключений. Балансировщик нагрузки отслеживает количество соединений серверов и отправляет следующий запрос к серверу с минимумом соединений.

Алгоритм DNS, его суть в том, что на DNS сервер просто добавляется несколько А-записей с разными IP-адресами всех наших серверов, и сервер сам будет в цикличном порядке выдавать эти адреса, т.е. первый запрос получит первый сервер, второй запрос – второй сервер, третий запрос – третий сервер, четвертый запрос – первый сервер.

Он абсолютно не зависит от протоколов высокого уровня; не зависит от нагрузки сервера, благодаря тому, что на всех клиентах, в основном, есть кэширующие DNS сервера, которые позволяют в случае резкого увеличения нагрузки на сервис компенсировать эту проблему.

Алгоритм проксирования. Суть его заключается в том, что в качестве балансировщика применяется так называемый умный прокси, т.е. если балансировщик получает запрос к нашему ресурсу, он анализирует заголовки прикладного уровня. Соответственно он может понимать, запрос к какому ресурсу пришел на наш балансировщик, и направить запрос на тот или иной сервер, на котором этот ресурс содержится. Плюс ко всему, при получении этого запроса балансировщик может добавлять в заголовки HTTP, например, информацию о том, с какого IP пришел клиент, для того, чтобы сервер знал, куда его потом отправлять и с кем он работает. Выполнив запрос, сервер передает его обратно на балансировщик, тот выполняет необходимые манипуляции с но-

выми заголовками либо третьего уровня, либо седьмого уровня и отдает его клиенту.

Алгоритм anycast. Он не требует настройки со стороны клиента, и суть его заключается в том, что из разных географических участков анонсируется один и тот же префикс сети. Таким образом, каждый запрос клиента будет маршрутизироваться на ближайший к нему сервер, который будет его обрабатывать.

Таблица 1. Сравнительный анализ алгоритмов по показателям производительности

Алгоритм / Метрика	Пропускная способность, k_1=0,08	Время миграции, k_2=0,08	Время отклика, k_3=0,06	Отказоустойчивость, k_4=0,15	Затраты, k_5=0,08	Использование ресурсов, k_6=0,2	Масштабируемость, k_7=0,15	Производительность, k_8=0,2
1.Round Robin Sheduling	Да	Нет	Да	Нет	Да	Да	Нет	Да
2.Max-Min	Да	Нет	Да	Нет	Да	Да	Нет	Да
3.Compare and Balance	Да	Да	Нет	Да	Да	Нет	Да	Да
4.Ant Colony Optimization	Да	Нет	Нет	Нет	Да	Да	Да	Да
5.Shortest Response Time First	Нет	Да	Да	Нет	Да	Да	Нет	Да
6.Active Clustering	Нет	Да	Нет	Нет	Да	Да	Нет	Нет
7.ACCLB	Да	Нет	Нет	Нет	Да	Да	Да	Да
8.Join-Idle-Queue	Да	Да	Да	Нет	Да	Нет	Да	Да
9.Central queuing	Нет	Да	Да	Нет	Да	Да	Нет	Да
10.Connection mechanism	Да	Нет	Нет	Да	Да	Да	Да	Нет
11.Least connections	Нет	Нет	Нет	Да	Да	Нет	Нет	Нет
12.DNSбалансировка	Да	Нет	Да	Нет	Нет	Да	Нет	Да
13.Проксирование	Да	Да	Да	Нет	Да	Да	Да	Да
14.Anycast	Нет	Нет	Нет	Да	Да	Да	Нет	Да

Он обеспечивает минимальную задержку при обработке запроса, так как клиент будет обслуживаться на ближайшем к нему сервере не только с географической точки зрения, но и с топологической. Обеспечивается доставка трафика, минуя магистральные каналы связи, соответственно, удешевляется трафик. Данный алгоритм балансировки имеет высокую отказоустойчивость. Если один из серверов выходит из строя, то все запросы к нему будут просто переброшены на ближайший сервер. Легко добавлять и выводить из работы любые сервера – просто перестать анонсировать по BGP, например, подсеть, и все запросы пользователей будут маршрутизироваться на другие сервера.

Эффективность алгоритмов балансировки нагрузки определяется несколькими показателями, которые представлены в табл. 1.

3. Метод выбора предпочтительного варианта

Сложные системы характеризуются многими параметрами, причем чаще всего в одних системах близки к оптимуму одни параметры, в других – другие. В таких условиях непросто оценить системы без специального формализованного правила – критерия. Предложенная методика предусматривает проведение оценки эффективности по произвольному количеству параметров. Оценка эффективности проводится в два этапа. На первом этапе с системой проводятся различные тесты с помощью специально разработанных моделей. Тесты позволяют определить значение отдельных параметров – критериев оценки. Тесты должны проводиться при максимально возможных неблагоприятных условиях функционирования системы. Каждый из них позволяет оценить один параметр системы при заданных условиях работы. На втором этапе оценки работы системы предлагается использовать весовые критерии эффективности. Весовой критерий позволяет учесть значимость частных критериев, полученных на первом этапе оценки. Значение интегрального критерия эффективности определяется выражением

$$K_m = \sum_{i=1}^{N} a_i k_i , \qquad (1)$$

где N – число частных критериев; k_i – их значение для оцениваемой системы; α_i – их весовые коэффициенты. Для устранения компенсации одного указанного критерия другим их значения используются в приведенной до максимума форме.

Лучшая система избирается по максимальному значению критерия:

$$K(s) = \max \sum_{i=1}^{N} a_i k_i . \qquad (2)$$

Для выбора величин весовых коэффициентов применяется два подхода:

1. В качестве коэффициентов используют дробные числа, сумма которых, для каждой системы, равна единице.

2. Используют целые коэффициенты, причем для наименее важного указанного критерия берут

единичный коэффициент, а для других берут коэффициенты, кратные единице.
В любом случае более важному указанному критерию соответствует больший по абсолютному значению коэффициент.

Таблица 2. Расчет функции полезности

Алгоритм/ Метрика	P1	P2	P3	P4	P5	P6	P7	P8	K_i
A1	1	0	1	0	1	0	0	1	0,42
A2	1	0	1	0	1	0	0	1	0,42
A3	1	1	0	1	1	1	1	1	0,94
A4	1	0	0	0	1	0	1	1	0,51
A5	0	1	1	0	1	0	0	1	0,42
A6	0	1	0	0	1	0	0	0	0,16
A7	1	0	0	0	1	0	1	1	0,51
A8	1	1	1	0	1	1	1	1	0,85
A9	0	1	1	0	1	0	0	1	0,42
A10	1	0	0	1	1	0	1	0	0,46
A11	0	0	0	1	1	1	0	0	0,43
A12	1	0	1	0	0	0	0	1	0,34
A13	1	1	1	0	1	0	1	1	0,65
A14	0	0	0	1	1	0	0	1	0,43

В связи с огромным количеством методов их выбор и оценка основных параметров системы становится нетривиальной задачей. Существующая методика оценки по одному критерию выглядит просто, не позволяя в полной мере судить об эффективности работы одной системы по сравнению с другими. Предложенная методика многокритериальной оценки позволяет значительно облегчить выбор системы и ее параметров для применения в конкретных условиях. Методика позволяет дать комплексную оценку работы системы. Она также обладает высокой гибкостью, оставляя выбор значений весовых коэффициентов пользователю, тем самым, давая возможность оценивать систему в зависимости от того, насколько важен каждый из критериев для работы системы в заданных условиях.
В качестве вариантов для балансировки траффика были выбраны методы, условно обозначенные так:
A1 – Round Robin Scheduling,
A2 – Max-Min Algorithm,
A3 – Compare and Balance,
A4 – Ant Colony Optimization,
A5 – Shortest Response Time First,
A6 – Active Clustering load balancing Algorithm,
A7 – ACCLB,
A8 – Join-Idle-Queue,
A9 – Central queuing,
A10 – Connection mechanism,
A11 – Least connections,
A12 – DNS балансировка,
A13 – Проксирование,
A14 – Anycast.

Критериями для выбора стали те параметры, которые эффективны для применения в беспроводных сенсорных сетях, а именно:
P1 – Пропускная способность,
P2 – Время миграции,
P3 – Время отклика,
P4 – Отказоустойчивость,
P5 – Затраты,
P6 – Использование ресурсов,
P7 – Масштабируемость,
P8 – Производительность.
В табл. 1 показан сравнительный анализ алгоритмов по показателям качества с использованием бинарного соответствия каждому критерию.
Для вычисления функции полезности каждого метода каждой метрике присваиваем коэффициент важности, который учитывает специфику БСС, причем их сумма равна 1. Таким метрикам как производительность и использование ресурсов присваивается коэффициент 0,2, так как они являются наиболее важными в автономных сетях. Отказоустойчивости и масштабируемости присваивается 0,15. Пропускная способность, время миграции и затраты получают 0,08, так как менее важные параметры. И время отклика присваивается 0,06, так как задержки не являются критичными в таких системах.
Следующим шагом мы проставляем в числовом виде значения в табл. 1. Значению *да* присваиваем 1, а значению *нет* – 0, в столбце 6 – наоборот.
Функция ценности для каждого метода отдельно рассчитывается по (2) с учетом того, что P_i принимают значения либо 0, либо 1:

$$K_{A_i} \sum_{i=1}^{14} k_i P_1 . \qquad (3)$$

Результаты расчёта показаны в табл. 2. Максимальное значение соответствует алгоритму Compare and Balance. Адаптация этого алгоритма для эффективного использования в БСС представлена далее.

Пусть p – количество узлов; (*V, E*) – граф, соответствующий связям узлов БСС, где *V* = (*1,2, ..., p*) – множество вершин, каждая из которых представляет узел БСС, а E – множество ребер. Предполагается, что граф связан. Две вершины *i* и *j* образуют ребро, если узлы *i* и *j* имеют канал передачи данных. Связь каждого i узла является скаляром l_i, представляющим нагрузку на узел БСС.

Средняя нагрузка узла БСС

$$\bar{l} = \frac{\sum_{i=1}^{p} l_i}{p}. \qquad (4)$$

Каждое ребро (i,j) графа также имеет связанный с ним скаляр δ_{ij}, где δ_{ij} – объем нагрузки, отправляемый от узла i к узлу j. Переменные δ_{ij}, являются направленными, т.е.

$$\delta_{ij} = -\delta_{ij}. \qquad (5)$$

Это означает, что если узел *i* должен отправить сообщение δ_{ij} узлу i, то узел j должен получить тот же объем информации $-\delta_{ij}$.

В действительности рабочая нагрузка будет целочисленным числом. В случае приложений с конечным числом элементов это может быть количество узлов в сети, если в каждый момент времени рабочая нагрузка на каждом узле является бесконечно малым числом.

Граф балансировки нагрузки должен сделать нагрузку на каждом узле равной средней нагрузке, т.е.

$$\sum_{\{j|(i,j)\in E\}} \delta_{ij} = l_i - \bar{l}, \quad i = 1, 2...., p. \qquad (6)$$

Если i>j и $(i, j) \in E$, то *i* будем называть вершиной ребра (*i, j*), а – *j* концом. Исходя из (5), берем δ_{ij}, если *i* является вершиной ребра (i, j), или δ_{ij} следует заменить на $-\delta_{ji}$, если *i* – конец ребра.

Если выполнены $p-1$ уравнения (6), то оставшееся уравнение будет выполнено автоматически. Таким образом, число независимых уравнений не превышает число вершин минус единицу. Число переменных в системе уравнений (6), с другой стороны, равно числу ребер в графе. В графе обычно больше ребер, чем вершин, и в любом случае для связного графа $|E| \geq |V| - 1$, где $|E|$ и $|V|$ – количество ребер и вершины графа (*E,V*) соответственно. Поэтому (6), вероятно, будет иметь бесконечно много решений. Выбираются среди этих решений те, что минимизируют пересылку данных.

Пусть A – матрица, связанная с (6), x – вектор из δ_{ij} и b – правая часть. Предполагая, что евклидова норма движения данных используется, как метрика и что стоимость связи между любыми двумя узлами одинакова.

Минимизируем $\frac{1}{2} x^T x$ при условии $Ax = b$.

Здесь A есть $|V| \times |E|$

$$(A)_{ik} = \begin{cases} 1, \text{если вершина i является вершиной ребра k,} \\ -1, \text{если вершина i является концом ребра k,} \\ 0, \text{в противном случае.} \end{cases} \qquad (7)$$

Применение необходимого условия для ограниченной оптимизации на (7) дает

$$x = A^T \lambda,$$

где λ - вектор множителей Лагранжа. Подставляя обратно в (6), получаем

$$L\lambda = b, \qquad (8)$$

с $L = AA^T$ – матрица размера $|V| \times |V|$.

Для иллюстрации матриц A и L рассмотрим простой граф из трех вершин, связанных линией связи, и пусть (1,2) и (2,3) – первое и второе ребра, тогда

$$A = \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}$$

матрицы, заданной выражением

$$L = AA^T = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}.$$

Таким образом, задача нахождения оптимального графика балансировки нагрузки становится решением линейного уравнения (8). Нетрудно убедиться, что L на самом деле является матрицей Лапласа с размерностью $|V| \times |V|$, определяемой как

$$(L)_{ij} = \begin{cases} -1, \text{если } i \neq j \text{ и ребро } (i, j) \in E, \\ \deg(i), \text{если } i = j, \\ 0, \text{в противном случае.} \end{cases}$$

Здесь deg(i) – степень вершины i в графе.

Как только вектор Лагранжа A решается из (8), то по уравнению (7) из-за специального вида A^T (каждая строка матрицы имеет только два ненулевых значения 1 и -1) величина нагрузки передается от узла i к узлу j, просто $\lambda_i - \lambda_j$, где λ_i и λ_j – множители Лагранжа, связанные с узлами i и j соответственно.

Таким образом, алгоритм балансировки нагрузки можно представить в виде последовательности:
Шаг 1: Нахождение средней рабочей нагрузки и, следовательно, правой части (8).
Шаг 2: Решение $L\lambda = b$ для получения λ.
Шаг 3: Определение количества нагрузки, подлежащей передаче. Количество узлов, которые отправляют данные на узел j, $\lambda_i - \lambda_j$.

4. Вывод

Для анализа методов балансировки нагрузки в телекоммуникационных сетях было выбрано 14 основных методов, которые сравнивались по 8 параметрам, выбранным на основе особенностей БСС. Рассмотрены практические особенности применения метода многокритериальной оптимизации, в результате которого были вычислены функции полезности для каждого метода, что позволило по максимальному значению выбрать предпочтительный метод для использования в БСС.

Практическая значимость полученных результатов заключается в возможности применения методов многокритериальной оптимизации при проектировании БСС, что позволит, с одной стороны, сократить время развертывания сети, а с другой – обеспечить эффективное распределение траффика, что непосредственно связано со сроком службы БСС. В дальнейшем исследовании можно увеличить количество рассматриваемых методов и критериев для сравнения.

Литература: 1. *Иванисенко И.* Методы балансировки с учетом мультифрактальных свойств нагрузки / И. Иванисенко, Л. Кириченко, Т. Радивилова // International Journal "Information Content and Processing". 2015. Т. 2, №4. **2.** *Безрук В.М.* Выбор предпочтительных протоколов маршрутизации узлов беспроводной сенсорно-актуаторной сети / В.М. Безрук, А.Н. Зеленин, В.А. Власова, Ю.В. Скорик, Ю.Н. Колтун // Восточно-Европейский журнал передовых технологий. 2016. №1/9 (79). С. 4-9. **3.** *Gupta R.O.* A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment / R.O. Gupta, T. Champaneria // International Journal of Advanced Research in Computer Science and Software Engineering. 2013. 3(11). P.782-790. **4.** *Rajwinder K., Pawan L.* Load Balancing in Cloud Computing / K. Rajwinder, L. Pawan // Association of Computer Electronics and Electrical Engineers. 2014. P. 374-381.

Тринслитерованный список литературы:

1. *Ivanisenko I.* Metody balansirovki s uchetom mul'tifraktal'nyh svojstv nagruzki / I. Ivanisenko, L. Kirichenko, T. Radivilova // International Journal "Information Content and Processing". 2015. T. 2, №4.
2. *Bezruk V.M.* Vybor predpochtitel'nyh protokolov marshrutizacii uzlov besprovodnoj sensorno-aktuatornoj seti/ V.M. Bezruk, A.N. Zelenin, V.A. Vlasova, Ju.V. Skorik, Ju.N. Koltun // Vostochno-Evropejskij zhurnal peredovyh tehnologij. 2016. №1/9 (79). S. 4-9.
3. *Gupta R.O.* A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment / R.O. Gupta, T. Champaneria // International Journal of Advanced Research in Computer Science and Software Engineering. 2013. 3(11). P. 782-790.
4. *Rajwinder K., Pawan L.* Load Balancing in Cloud Computing / K. Rajwinder, L. Pawan// Association of Computer Electronics and Electrical Engineers. 2014. P. 374-381.

Поступила в редколлегию 11.06.2018
Рецензент: д-р техн. наук, проф. Безрук В.М.

Власова Виктория Александровна, канд. техн. наук, доцент кафедры «Информационно-сетевая инженерия» ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Науки, 14, E–mail: viktoriia.vlasova@nure.ua
Олейникова Анна Александровна, бакалавр, студентка кафедры «Информационно-сетевая инженерия» ХНУРЭ. Адрес: Украина, 61166, Харьков, пр. Науки, 14, E–mail: anna.oleinikova@nure.ua
Vlasova Viktoriya, Candidate of Technical Sciences, Assistant Professor Department of Information and network engineering**,** Kharkiv National University of Radio Electronics. Address: 14 Nauki av., Kharkiv city, Ukraine, 61166**,** E–mail: viktoriia.vlasova@nure.ua
Oleinikova Anna, Bachelor, student, Department of Information and network engineering**,** Kharkiv National University of Radio Electronics. Address: 14 Nauki av., Kharkiv city, Ukraine, 61166**,** E-mail: anna.oleinikova@nure.ua