Применение муравьиных алгоритмов для решения задачи маршрутизации в компьютерных сетях

Ладыженский Ю.В., Мирецкая В.А., Мирецкий А.В.
Кафедра ПМиИ ДонНТУ
ly@cs.dgtu.donetsk.ua
vmiretskaya@gmail.com
a.miretsky@gmail.com

Abstract

Ladyzhensky Y.V., Miretskaya V.A., Miretsky A.V., Usage of ant-based algorithms for solving routing problem in the computer networks. *The ant-based approach for solving routing problems in the computer networks is described in this paper.*

Введение

Оригинальный эвристический подход Ant System к решению комбинаторных оптимизационных задач, применяющий идеи коллективного поиска оптимальных путей муравьями, предложен в [1]. Алгоритм Ant system часто модифицировался для улучшения его производительности и применения к другим оптимизационным задачам (раскраска графов, маршрутизация автомобильного транспорта и др.) ([2], [3], [4], [5], [6]). Способ использования совместного поведения муравьев в алгоритмах комбинаторной оптимизации на примере задачи о коммивояжере представлен в [7] и [8]. Для решения задачи маршрутизации в телекоммуникационных сетях в [9] предложен алгоритм ABC. Эффективный алгоритм маршрутизации AntNet, который завоевал большую популярность, предложен в работе [10].

За последние 10 лет разработано несколько модификаций алгоритма AntNet, которые позволяют применить его в различных компьютерных сетях, включая мобильные Ad Hoc сети и сети на кристалле.

В предлагаемой статье представлен анализ применения муравьиных алгоритмов для решения задачи маршрутизации в компьютерных сетях. Для исследования муравьиных алгоритмов маршрутизации разработана система моделирования сетей DynNetSim.

Мета-эвристика Ant Colony Optimization

Алгоритмы, использующие муравьиные колонии для поиска оптимальных решений, обобщены в мета-эвристику ACO (Ant Colony Optimization – оптимизация муравьиной колонией) [11]. Мета-эвристика

ACO применяется к задачам дискретной оптимизации, которые имеют следующие параметры:

- $C = \{c_1, c_2, ..., c_{Nc}\}$ – конечное множество элементов;
- $L = \{l_{c_i c_j} | (c_i, c_j) \in \tilde{C}\}, |L| \le N_C^2$ – множество возможных соединений/переходов между элементами $\tilde{C}$, где $\tilde{C}$ – подмножество декартова произведения $C \times C$;
- $G = (C, L)$ – граф, представляющий задачу дискретной оптимизации (см. рис. 1);
- $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$ – функция стоимости соединения, связанная со всеми $l_{c_i c_j} \in L$, и в общем случае, зависящая от времени t;
- $\Omega \equiv \Omega(C, L, t)$ – конечное множество ограничений, определенных над элементами множеств C и L;
- $s = \langle c_i, c_j, ..., c_k, ... \rangle$ – последовательность элементов C, которая называется состоянием задачи. Если S – множество всех возможных последовательностей, то $\tilde{S} \subseteq S$ – множество всех последовательностей, которые возможны при ограничениях $\Omega(C, L, t)$;
- пусть даны два состояния s_1 и s_2 (см. рис. 1), для которых установлено отношение смежности: s_2 называется смежным с s_1, если $s_1, s_2 \in S$ s_1, и s_2 может быть достигнуто из состояния s_1 за один логический шаг перемещения по графу. Т.е., если c_1 последний элемент в последовательности, определяющей s_1, то должен существовать такой элемент $c_2 \in C$, что $l_{c_1 c_2} \in L$ и $s_2 \equiv \langle s_1, c_2 \rangle$. Состояния, смежные для s, образуют окрестность N_s;
- ψ – решение, если $\psi \in \tilde{S}$ и удовлетворяет всем ограничениям задачи;
- $J_\psi(L, t)$ – стоимость, соответствующая каждому решению ψ. $J_\psi(L, t)$ есть функция всех стоимостей $J_{c_i c_j}$ всех соединений, принадлежащих решению ψ;

Решения оптимизационной задачи могут быть выражены в терминах возможных путей на графе G. ACO алгоритмы предназначены для поиска путей (последовательностей) с минимальной стоимостью, допустимых при ограничениях Ω.

ACO алгоритмы используют колонию муравьев для коллективного решения оптимизационной задачи на графе G. Информация, собранная муравьями в процессе поиска, сохраняется в феромонных следах τ_{ij}, соответствующих соединениям l_{ij}. Феромонные следы кодируют всю память о процессе поиска решения муравьями.

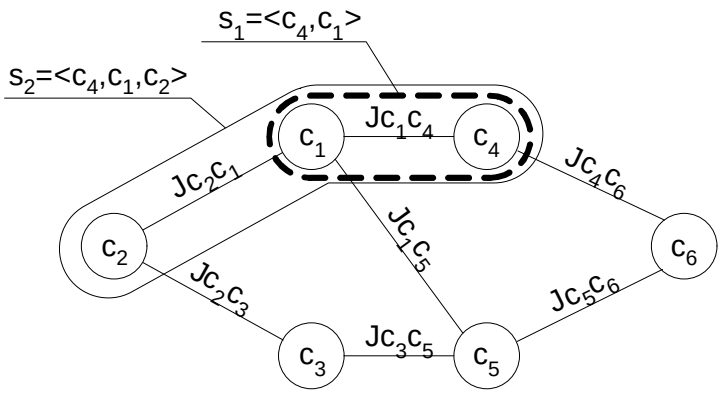

Рисунок 1 – Граф задачи дискретной оптимизации

Муравьи колонии имеют следующие свойства:

- муравей ищет подходящие решения с минимальными стоимостями $\hat{J}_\psi = \min_\psi J_\psi(L,t)$;
- муравей k имеет память M_k, которую он может использовать для сохранения пути, по которому он продвигался. Содержимое памяти может использоваться для построения подходящих решений, для оценки найденных решений и для обратного перемещения по маршруту;
- для муравья k может быть определено начальное состояние s^k_{start} и одно или более конечных состояний s^k_{end};
- муравьи начинают движение из начального состояния и переходят в подходящие смежные состояния, постепенно выстраивая решение. Процедура построения решения останавливается, если хотя бы для одного муравья k достигается хотя бы одно конечное состояние s^k_{end};
- муравей k в состоянии $s_r = \langle s_{r-1}, i \rangle$ может двигаться к любому элементу j в его подходящей окрестности N_i^k, определенной как $N_i^k = \{ j \mid (j \in N_i) \wedge (\langle s_r, j \rangle \in \tilde{S}) \}$. Направление перехода выбирается по вероятностному правилу;
- вероятностное правило для муравья в элементе i есть функция от:
  1) значений, сохраненных в локальной структуре данных $A_i = \lfloor a_{ij} \rfloor$ элемента, называемой таблицей маршрутизации муравьев, и вычисляемой на основе информации о феромонных следах в окрестности элемента i по некоторым эвристическим правилам;
  2) информации в собственной памяти муравьев, где накапливаются данные об элементах, посещенных муравьями;
  3) других ограничений и условий задачи;

- при движении от элемента *i* к элементу *j*, муравей может обновлять феромонный след τ_{ij} на дуге (i, j) по некоторому правилу. Этот процесс называется прямым феромонным обновлением;
- как только решение построено, т.е. получена последовательность ψ, муравей может пройти обратно по тому же пути и обновить феромонный след на дугах, по которым проходит. Это называется обратным феромонным обновлением;
- как только муравей построил решение и вернулся в исходный элемент, он погибает, освобождая все выделенные для него вычислительные ресурсы.

При поиске подходящих решений, муравьи используют только собственную информацию и локальную информацию элементов. Обмен информацией между муравьями происходит косвенно через переменные τ_{ij}, хранящие информацию о феромонных следах.

Алгоритм ABC

В [9] применен метод муравьиной колонии для решения задачи балансировки нагрузки в телекоммуникационных сетях.

Телекоммуникационная сеть представляется в виде графа. Коммутационные станции отображаются в узлы графа, линии связи между станциями – ребра графа.

Каждый узел содержит феромонную таблицу. Таблица имеет *N-1* колонок и N_k строк, N – количество узлов в сети, N_k – количество узлов смежных с узлом *k*.

Периодически из каждого узла запускаются муравьи, имеющие случайные узлы-приемники. Муравьи движутся, выбирая промежуточные узлы в соответствии с феромонной таблицей. Если *j* – узел-приемник, то каждое $P_{i,j}$ в феромонной таблице – это вероятность перехода муравья в соседний узел *i*. При попадании муравья в очередной промежуточный узел, в узле происходит обновление феромонной таблицы:

$$\begin{cases} P'_{\text{n,src}} = \dfrac{P_{n,src} + \Delta P}{1 + \Delta P} \\ P'_{\text{i,src}} = \dfrac{P_{i,src}}{1 + \Delta P}, \qquad i = 1..N_k, i \neq n \end{cases} \tag{1}$$

где n – узел, из которого пришел муравей; *src* – узел-источник, сгенерировавший муравья; $P_{n,src}$, $P'_{n,src}$ – старое и новое значения ячейки таблицы; ΔP – увеличение феромона.

При достижении узла-приемника, муравей уничтожается.

Муравей имеет возраст, равный длине пути, т.е. количеству пройденных узлов. Муравьи задерживаются в узлах, переполненных

вызовами. Время задержки *delay* тем больше, чем выше степень загрузки узла [12]:

$$delay = \left[80 \cdot e^{-0.075 \cdot cap}\right],$$

где *cap* – свободная (резервная) емкость узла.

Возраст муравья *age* влияет на ΔP [12]:

$$\Delta P = \frac{0.08}{age} + 0.005,$$

Муравьи, которые выбирают короткий и менее загруженный путь, в большей степени влияют на вероятность выбора этого маршрута следующими муравьями и пакетами с вызовами, чем муравьи, выбирающие худшие по длине и загрузке пути. Это связано с тем, что первые муравьи раньше достигают узел-приемник и имеют меньший возраст, а следовательно, большее значение ΔP. Новые вызовы будут идти по маршрутам с меньшими длиной и степенью загруженности. Это приведет к балансировке нагрузки за счет разгрузки узлов на загруженных маршрутах.

Сравнение алгоритма ABC с другим распределенным алгоритмом Mobile Software Agents [13] показало снижение вероятности отказа вызовов в 2-3 раза по сравнению с Mobile Software Agents.

Алгоритм AntNet

Алгоритм AntNet [10] позволяет избежать перегрузок и появления точек насыщения (hot-spot) в сети, что является одной из основных проблем маршрутизации.

AntNet использует два множества однородных мобильных агентов. К первому множеству относятся агенты, идущие вперед по сети от узла-источника к узлу-приемнику (F-муравьи, forward ants). Ко второму множеству относятся агенты, идущие в обратном направлении (B-муравьи, backward ants). Агенты каждого множества имеют одинаковую структуру, но по-разному ведут себя в сети. Агенты общаются косвенным путем, записывая и читая информацию в каждом узле k сети в таблицу маршрутизации T_k и структуру локальной модели трафика M_k.

Таблица T_k организована также как и в дистанционно-векторных алгоритмах [14]. Кроме этого для каждого возможного узла d и для каждого соседнего узла n, таблица T_k хранит вероятность P_{nd} выбора узла n при условии, что d – узел-приемник:

$$\sum_{n \in N_k} P_{nd} = 1, \quad d \in [1, N], \quad N_k = \{\text{Соседи узла } k\} \qquad (2)$$

Структура $M_k(\mu_d, \sigma_d^2, W_d)$ – это массив, определяющий локальную параметрическую статистическую модель распределения трафика в сети с точки зрения узла k. Модель является адаптивной и описывается для

каждого узла d математическим ожиданием μ_d и дисперсией σ_d^2 времени обхода сети мобильными агентами, и массивом скользящего окна наблюдений W_d. Когда B-муравей приходит в узел k, время достижения узла d из узла k $o_{k\to d}$ помещается в массив W_d.

Для расчета характеристик в алгоритме используется экспоненциальная модель:

$$\begin{aligned} \mu_d &:= \mu_d + \eta(o_{k\to d} - \mu_d) \\ \sigma_d^2 &:= \sigma_d^2 + \eta((o_{k\to d} - \mu_d)^2 - \sigma_d^2) \end{aligned} \tag{3}$$

где η – весовой коэффициент.

Массив скользящего окна наблюдений W_d используется для расчета значения W_{bestd} – экспериментальной оценки нижней границы времени достижения узла d из текущего узла k.

$\{T_k\}$ и $\{M_k\}$ образуют локальную память узлов. Модель M_k содержит абсолютные оценки расстояния/времени ко всем узлам, таблица маршрутизации T_k содержит относительные вероятностные оценки полезности перехода для каждой пары «линия связи – узел-приемник».

Через регулярные интервалы времени Δt из каждого узла сети src запускается мобильный агент (F-муравей) $F_{src\to d}$, направленный к узлу-приемнику d. Целью F-муравья является исследование состояний загрузки сети и определение наиболее подходящего пути к узлу d. F-муравьи перемещаются по тем же очередям, что и пакеты данных. Это сделано для того, чтобы агент испытал на себе те же самые нагрузки сети, что и пакет с данными.

Муравьи адаптируют свои перемещения к изменяющемуся распределению трафика данных. Узлы-приемники для F-муравья выбираются в зависимости от модели трафика: если $f_{src,d}$ – величина потока передачи данных (в битах или количестве пакетов) от узла s к узлу d, то вероятность создания в узле src F-муравья с узлом-приемником d:

$$p_d = \frac{f_{src,d}}{\sum_{d'=1}^{N} f_{src,d'}} \tag{4}$$

Во время движения к узлу-приемнику идентификатор каждого пройденного узла k и время, прошедшее с момента запуска F-муравья до момента достижения им k-го узла, записываются в стековую память $S_{src\to d}$.

В каждом узле k F-муравей выбирает промежуточный узел n из множества тех узлов, которые он еще не посещал. Если же агент посещал все узлы, смежные с узлом k, то по всем смежным узлам выбирается узел n с вероятностью $P`_{nd}$:

$$P'_{nd} = \frac{P_{nd} + \alpha \cdot l_n}{1 + \alpha(N_k - 1)} \tag{5}$$

где P_{nd} – элемент таблицы маршрутизации; α – весовой коэффициент; l_n учитывает длину в битах очереди q_n к линии связи между узлами k и n:

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}} \quad (6)$$

В [10] рекомендуется использовать $\alpha \in [0.2;0.5]$.

Если в стеке F-муравья обнаружен цикл, т.е. муравей вернулся в уже пройденный узел, то информация об узлах цикла выталкивается из стека.

$F_{src \to d}$ муравей, достигший узла d, уничтожается. Вместо него создается B-муравей $B_{d \to src}$, которому передается вся информация от F-муравья. B-муравей двигается по пути F-муравья, но в противоположном направлении. B-муравьи используют приоритетные очереди, чтобы как можно быстрее распространить между узлами информацию, собранную F-муравьями.

Когда B-муравей достигает узла k из соседнего узла f, в узле k происходит обновление модели трафика M_k и элементов таблицы маршрутизации T_k, соответствующих узлу-приемнику d. M_k модифицируется по формуле (3).

В таблице T_k увеличиваются вероятности $P_{fd'}$ (т.е. вероятности выбора соседнего узла f, если узел-приемник d') и уменьшаются другие вероятности $P_{nd'}$ путем нормирования:

$$P_{fd'} := P_{fd'} + r \cdot (1 - P_{fd'}) \quad (7)$$

$$P_{nd'} := P_{nd'} - r \cdot P_{nd'}, \quad n \in N_k, n \neq f \quad (8)$$

где $r \in (0,1]$ – коэффициент стабилизации, аналог феромонов. В [10] коэффициент $r \equiv r(T, M_k)$ есть функция от времени движения муравья и от локальной модели трафика.

Использование в алгоритме вероятностных маршрутных таблиц увеличивает производительность на 30-40% [10].

Применение AntNet к мобильным Ad Hoc сетям

Мобильные Ad Hoc сети состоят из автономных самоорганизующихся в сети устройств. В таких сетях нет инфраструктуры, они формируются динамически через взаимодействие некоторого множества независимых узлов. Топология мобильной сети постоянно изменяется, и ранее эффективные маршруты могут стать бесполезными или вообще недоступными. Поэтому маршрутная информация в таких сетях должна обновляться чаще [15].

Для маршрутизации в Ad hoc сетях предложены различные гибридные алгоритмы на основе AntNet. Гибридный протокол маршрутизации Ant-AODV [16, 17] использует возможности AntNet и

AODV (Ad-hoc on-Demand Distance Vector) [26]. Эта модификация обеспечивает высокую возможность подключения и позволяет уменьшить поиск маршрута муравьями до начала соединения.

В [18] предложен гибридный многопутевой алгоритм AntHocNet для сетей MANETs. Алгоритм AntHocNet использует три множества муравьев: реактивные RF-муравьи (reactive forward ants), проактивные PF-муравьи (proactive forward ants) и B-муравьи (backward ants). RF-муравей ищет путь к узлу-приемнику и превращается в B-муравья, когда достигает его. Когда начинается передача данных, запускаются проактивные PF-муравьи, контролирующие качество используемых путей. Такие агенты могут быть отправлены с малой вероятностью по широковещательной рассылке для исследования новых путей.

Путь *Path*, который прошел RF-муравей записывается в стек и используется B-муравьем для возврата в узел источник. B-муравей пошагово вычисляет время $\hat{T}_{Path}$, которое понадобится пакету данных для достижения узла-цели при передвижении по пути *Path*: Это время используется для обновления таблицы маршрутизации.

$$\hat{T}_{Path} = \sum_{i=1}^{n-1} \hat{T}_{i \to i+1},\ i \in Path, \tag{9}$$

где $\hat{T}_{i \to i+1}$ – локальная оценка времени перехода из узла i в узел $i+1$, $\hat{T}_{i \to i+1} = (Q^i_{mac} + 1) \cdot \hat{T}^i_{mac}$; $\hat{T}^i_{mac}$ - среднее время отправки одного пакета; Q^i_{mac} - текущее количество пакетов в очереди для отправки на MAC-уровне. $\hat{T}^i_{mac}$ вычисляется как скользящее среднее время, прошедшее между прибытием пакета на MAC-уровне и окончанием успешной отправки:

$$\hat{T}^i_{mac} = \alpha \hat{T}^i_{mac} + (1 - \alpha) t^i_{mac}, \tag{10}$$

где $\alpha \in [0, 1]$, t^i_{mac} - время, затрачиваемое на отправку пакета из узла i.

В каждом промежуточном узле $i \in Path$ B-муравей настраивает путь по направлению к узлу-приемнику d, создавая или обновляя записи T^i_{nd} таблицы маршрутизации. По прибытии в узел i из соседнего узла n, муравьи создают запись в таблице маршрутизации T^i, помечая n как следующий ретрансляционный участок достижения узла d. Запись содержит значение феромона T^i_{nd}, которое определяет полезность пути в узел d через узел n. Если $\hat{\mathrm{T}}_{i \to d}$ время движения, полученное муравьем, и h это количество ретрансляционных участков, то значение феромона определяется как: $\tau_{id} = ((\hat{\mathrm{T}}_{i \to d} + h\mathrm{T}_{hop})/2)^{-1}$, где T_{hop} - фиксированное значение времени нахождения одного ретрансляционного участка в незагруженном состоянии. Если в T^i уже находится запись T^i_{nd}, то ее значение обновляется с помощью взвешенного среднего $\mathrm{T}^i_{nd} = \gamma \mathrm{T}^i_{nd} + (1 - \gamma)\tau_{id}$, $\gamma \in [0,1]$.

Узлы в AntHocNet маршрутизируют данные стохастически. Если на пути к узлу *d*, узел *k* имеет множество вероятных ретрансляционных участков *n*, то один из них выбирается с вероятностью P_{nd}:

$$P_{nd} = \frac{T_{nd}^2}{\sum_{i \in N_d} T_{id}^2}. \quad (11)$$

PF-муравьи используются для периодического контроля качества путей. Эти агенты перемещаются по следу феромона по тому же пути, что и пакеты данных и могут быть отправлены по широковещательной рассылке. Число широковещательных рассылок не превышает двух. Если F-муравей не находит маршрутной информации на расстоянии двух ретрансляционных участков, он удаляется.

Для улучшения маршрутизации RF-муравьев и PF-муравьев, в AntHocNet используются Hello-пакеты. С их помощью узлы собирают информацию о своих текущих соседях и хранят ее в феромонных таблицах. Если узлом-приемником является соседний узел, то муравей отправляется в него без использования широковещательной рассылки.

Узлы могут обнаруживать отказы линий связи. Потерянная линия считается «важной», если она часто использовалась для отправки данных, или если нет альтернативных путей к соседнему узлу. В этом случае узел должен попытаться восстановить путь.

После отказа, узел отправляет по широковещательной рассылке восстанавливающего муравья (route repair ant), который движется к узлу-цели как RF-муравей. Он использует маршрутную информацию, когда это возможно. В остальных случаях он использует широковещательную рассылку. Узел ожидает возвращения восстанавливающего муравья в течение времени, равного $5 \cdot T_{k \to d}$, где $T_{k \to d}$ – последняя оценка времени достижения узла *d* из узла *k*. Если восстанавливающий муравей не возвращается, то считается, что найти альтернативный путь невозможно, и запись о потерянном пути (узле) удаляется из таблицы маршрутизации.

Показано, что AntHocNet превосходит AODV [26] по степени доставки пакетов и по времени задержки [18], особенно в тяжелых условиях (постоянное движение узлов, большое количество узлов).

Маршрутизация в сетях на кристаллах

Проблема маршрутизации информации возникает и при реализации в сверхбольших интегральных схемах (VLSI) программируемых и реконфигурируемых сетей процессоров [19].

Применение распределенного муравьиного алгоритма для маршрутизации данных в однородной сети на кристалле, описанное в [20], уменьшает скученность пакетов в узлах, возникающую при высокой интенсивности трафика.

Каждый маршрутизатор соединен с обрабатывающим устройством (Process Element) (см. рис. 2).

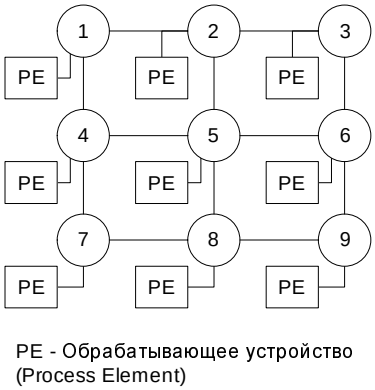

Рисунок 2 –Структура сети на кристалле

Маршрутизатор содержит блоки маршрутизации пакетов с данными, маршрутизации F-муравьев и маршрутизации B-муравьев (см. рис. 3).

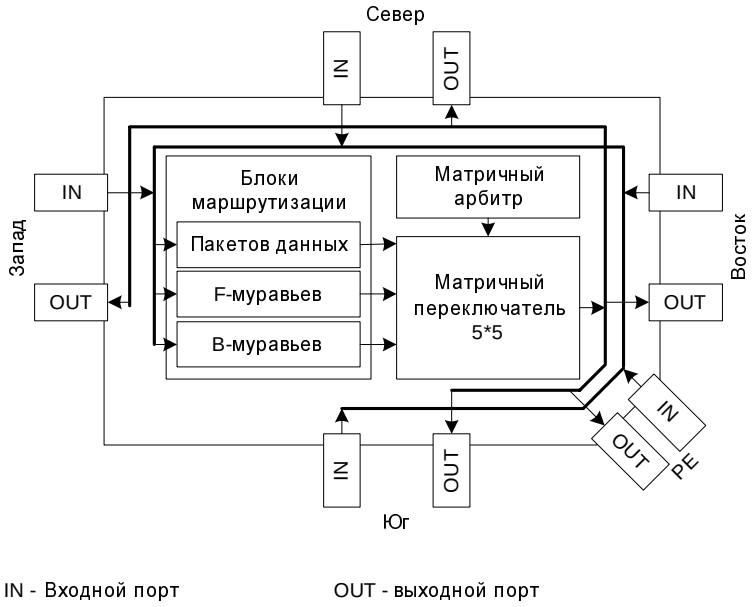

Рисунок 3 – Структура маршрутизатора

Блок маршрутизации пакетов данных использует таблицу маршрутизации размером $4 \cdot (N-1)$,где N – общее количество узлов в сети. Для упрощения аппаратной реализации, в таблице маршрутизации вероятности закодированы целыми кодами от 0 до 4.

На стадии инициализации таблицы маршрутизатор выбирает одно или два направления движения для каждого узла. Если узел-приемник d расположен в одной колонке или строке с текущим узлом, то вероятность выбора такого направления равна 1 (код=4). В остальных случаях вероятность выбора 0.5 (код=2).

Блок маршрутизации F-муравьев определяет выходной порт для F-муравья $F_{src \to d}$. F-муравьи генерируются с интервалом времени Δt, который обратно пропорционален частоте генерации пакетов с данными узловым процессором. Как и в оригинальном алгоритме AntNet, для F-муравья выбирается узел-приемник *d*, в который чаще всего отправляются пакеты данных («популярный узел»).

Выбор промежуточного узла *j* при движении в узел d осуществляется по модифицированной формуле (5), учитывающей специфику сетей на кристаллах:

$$P'(j) = \frac{3 \times P(j) + L_n(j)}{4}, \qquad (12)$$

где *P(j)* – вероятность выбора узла *j* на пути к узлу *d*, полученная из таблицы маршрутизации; $L_n(j)$ – параметр, учитывающий мгновенное состояние очереди управляющих пакетов для соответствующего входного порта в узле *j*. $L_n(j)$ зависит от значения сигнала «буфер полон» (w_full) и определяется по таблице 1, где a и b – узлы, в которые может быть отправлен муравей.

Таблица 1 – Правила вычисления $L_n(a)$ и $L_n(b)$

w_full(a)	w_full(b)	$L_n(a)$	$L_n(b)$
0	0	2	2
0	1	4	0
1	0	0	4
1	1	2	2

Сигнал w_full вычисляется для каждого буфера маршрутизатора. Если количество пустых ячеек в буфере меньше заданного значение, w_full устанавливается равным 1.

Во время движения F-муравей записывает в стековую память номера посещаемых узлов и состояния перегрузки (CS) узлов. Значение CS определяется как сумма четырех Флагов Перегрузки (CF), т.е. CS – целое число, $CS \in [0,4]$. Каждый из флагов соответствует своему направлению в сети.

Когда F-муравей достигает узла-приемника, маршрутизатор узла преобразует его в B-муравья и отправляет в обратном направлении.

Блок маршрутизации B-муравьев реализует задачу B-муравья – обновление таблиц маршрутизации сети. При движении B-муравья каждый узел выталкивает номер узла (ID маршрутизатора) из его памяти, определяя следующий узел для перемещения, обновляет таблицу маршрутизации:

$$P'_{fd} = P_{fd} + \frac{(4 - CS) \times (4 - P_{fd})}{4} \qquad (13)$$

$$P'_{nd} = P_{nd} - \frac{(4 - CS) \times P_{nd}}{4}, для \forall n \neq f \qquad (14)$$

где, f – узел, смежный с k, из которого пришел B-муравей; P_{fd} (P_{nd}) – вероятность выбора линии связи f (n) при маршрутизации из узла k в узел d.

Применение алгоритма AntNet для сетей на кристалле позволяет снизить среднюю задержку пакетов и уменьшить вероятность возникновения точек скученности, по сравнению с другими алгоритмами.

Система моделирования DynNetSim

Муравьиные алгоритмы маршрутизации основаны на вероятностных правилах и используют эвристически задаваемые параметры, поэтому для исследования их свойств применяются методы имитационного моделирования. Для исследования алгоритмов маршрутизации в [21], [22] разработана объектно-ориентированная имитационная система моделирования DynNetSim. Система использует метод дискретно-событийного моделирования, написана на языке Delphi в операционной системой Windows.

Модель сети представлена в виде графа. Вершины графа – это узлы сети, ребра графа – это линии связи. Модель содержит (см. рис. 4): узел сети – предназначен для приема пакетов, хранения их в очередях и отправки пакетов в смежные узлы; маршрутизатор – внутренняя подсистема узла сети для выбора маршрута пакетов; линия связи – модель линии связи, по которой передаются пакеты между узлами сети; генератор трафика – для генерации рабочей нагрузки сети и для моделирования процессов обмена данными между конечными точками сети.

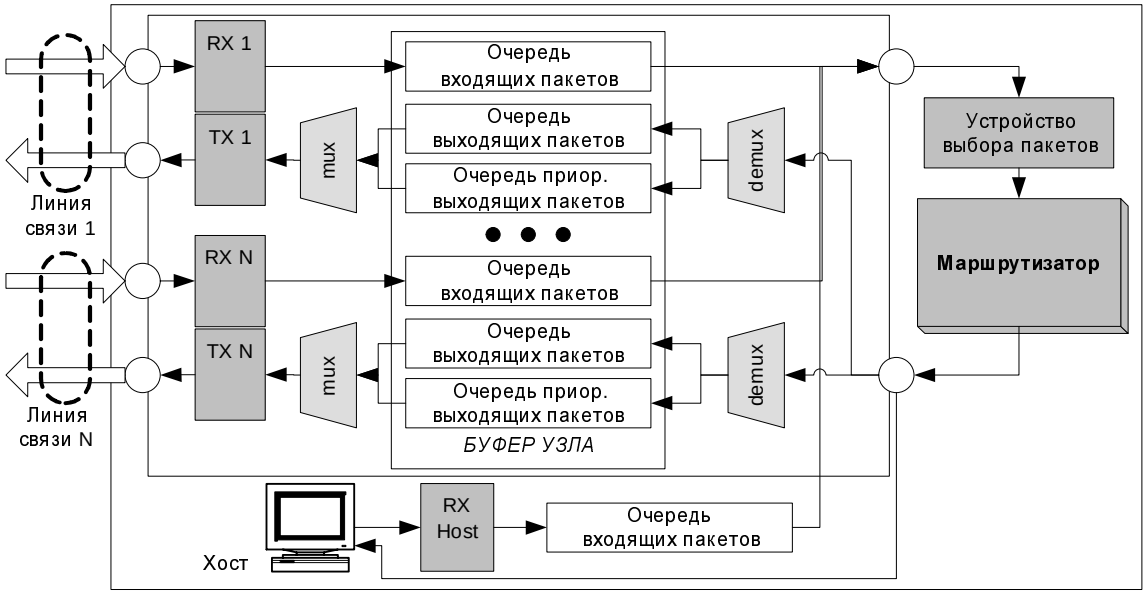

Рисунок 4 – Модель узла сети

Система моделирования реализует: подготовку входных данных для экспериментов (построение сети при помощи встроенного графического редактора, задание параметров объектов сети, планирование исключительных ситуаций в работе сети – обрыв линии связи, резкое повышение поступающего трафика); подключение библиотек с исследуемыми алгоритмами маршрутизации; выполнение моделирования и статистическую обработку результатов; экспорт результатов моделирования в текстовые файлы (TXT и CSV) и графические файлы (BMP). Результаты моделирования – это средние характеристики сети, а также графики их изменения. Система позволяет определить пропускную способность сети, задержку пакетов в узлах сети и потери пакетов.

Система DynNetSim использовалась для исследования поведения алгоритмов AntNet и OSPF [25] в условиях перегрузки и в условиях обрыва линий связи ([23], [24]). Разработан гибридный алгоритм AntNet-M, который объединяет в себе оригинальный алгоритм AntNet и HELLO-протокол алгоритма OSPF. Результаты моделирования показали, что эффективность оригинального алгоритма AntNet после модификации возросла в условиях обрыва линий связи – уменьшилось время настройки таблиц маршрутизации для перенаправления трафика в обход оборванной линии.

Заключение

Эвристический подход ACO для решения задач дискретной оптимизации, использующий муравьиные колонии, нашел широкое применение при решении задачи маршрутизации в различных коммуникационных сетях. Этот подход позволяет снизить вероятность возникновения перегрузок и отказов в обслуживании за счет балансировки нагрузки в сети.

Муравьиные алгоритмы маршрутизации основаны на вероятностных правилах и используют эвристически задаваемые параметры, поэтому для исследования их свойств применяются методы имитационного моделирования. Для исследования алгоритмов маршрутизации построена объектно-ориентированная имитационная система моделирования DynNetSim. Система использует метод дискретно-событийного моделирования, написана на языке Delphi в операционной системой Windows. DynNetSim позволяет определить средние характеристики сети (пропускную способность сети, задержку пакетов в узлах сети и потери пакетов) и графики их изменения.

Разработан новый гибридный алгоритм AntNet-M, объединяющий алгоритм AntNet и HELLO-протокол алгоритма OSPF. AntNet-M позволяет уменьшить время перенастройки таблиц маршрутизации при обнаружении обрывов линий связи в компьютерной сети.

Литература

1. Dorigo M., Maniezzo V., Colorni A., Positive feedback as a search strategy. Tech. rept. 91-016. Dipartimento di Elettronica, Politecnico di Milano, Italy. 1991.
2. Dorigo M., Optimization, Learning and Natural Algorithms (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy. 1992.
3. Dorigo M., Maniezzo V., Colorni. A., The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996. 26(1). P. 29–41.
4. Dorigo M., Gambardella L. M., Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation. 1997. 1(1). P. 53–66.
5. Stützle T., Holger Hoos. Improvements on the Ant System: Introducing the MAX–MIN Ant System. In Proc. Intel. Conf. on Artificial Neural Networks and Genetic Algorithms. Springer Verlag, 1997. P. 245-249.
6. Bullnheimer B., Hartl R. F., Strauss. C., A New Rank-Based Version of the Ant System: A Computational Study. Tech. rept. POM-03/97. Institute of Management Science, University of Vienna, Austria. Accepted for publication in the Central European Journal for Operations Research and Economics. 1997.
7. С.Д. Штовба Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. 2003. №4. С. 70-75.
8. С.Д. Штовба Муравьиные алгоритмы: Теория и применение //Программирование. 2005. №4. С. 1-16.
9. Schoonderwoerd R. Collective Intelligence for Network control. Unpublished Ir.-thesis, Delft University of Technology, Faculty of Technical Informatics. June 1996.
10. Di Caro G., Dorigo M. AntNet: Distributed Stigmergetic Control for Communications Networks: Journal of Artificial Intelligence Research. 1998. №9. P. 317-365.
11. Dorigo M., Di Caro G., Ant Colony Optimization: A New Meta-Heuristic. – Proceedings of the Congress on Evolutionary Computation. IEEE Press: USA. 1999. P. 1470 – 1477.
12. Schoonderwoerd R., Holland O., Bruten J., Rothkranz L. Ant-based load balancing in telecommunication networks. Adaptive Behavior, 1996, 5(2). P. 169–207.
13. Appleby S., Steward S. Mobile software agents for control in telecommunication networks. In BT Technology Journal, Vol.12, No.2. 1994.
14. RFC 2453. RIP Version 2 // http://www.faqs.org/rfcs/rfc2453.html
15. Mobile Ad Hoc Networking. Stefano Basagni, Marco Conti, Silvia Giordani, Ivan Stojmenovic. IEEE Press. 2004. 461p.

16. Marwaha S., Chen K.T., Srinivasan D. Mobile Agents based Routing Protocol for Mobile Ad Hoc Networks // In Proceedings of IEEE Globecorn, 2002.

17. Marwaha S., Chen K.T., Srinivasan D. A Novel Routing Protocol Using Mobile Agents and Reactive Route Discovery For Ad Hoc Wireless Networks // Networks, 2002. ICON 2002. 10th IEEE International Conference on 2002. P. 311-316.

18. Di Caro G., Ducatelle F., Gambardella L.M., AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. Proceedings of PPSN VIII, Birmingham, UK, September 18-22, 2004, Springer-Verlag, Lecture Notes in Computer Science, Vol. 3242.

19. Dally W. J., Towles B. Route packets, not wires: on-chip interconnection Networks in Proc. DAC. June 2001. P. 684-689.

20. Daneshtalab M., Sobhani A., Mottaghi M. D., Kusha A.A., Navabi Z., Fatemi O. Ant Colony Based Routing Architecture for Minimizing Hot Spots in NOCs // Proceedings of the 19th annual symposium on Integrated circuits and systems design. 2006. P. 56-61

21. Ладыженский Ю.В., Мирецкий А.В. МОДЕЛИРОВАНИЕ АЛГОРИТМОВ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей Міжнародної науково-практичної конференції м. Запоріжжя, 13-15 квітня 2006 року / Під заг. ред. Д.М. Пізи. – Запоріжжя: ЗНТУ. 2006. С. 165-167.

22. Ладыженский Ю.В., Мирецкий А.В. ПРОГРАММНАЯ СИСТЕМА ДЛЯ МОДЕЛИРОВАНИЯ АЛГОРИТМОВ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ // ІНТЕРНЕТ-ОСВІТА-НАУКА-2006, п’ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006. Збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця. 2006. С. 372-374.

23. Ладыженский Ю.В., Мирецкая В.А. ИССЛЕДОВАНИЕ ДИНАМИЧЕСКИХ АЛГОРИТМОВ МАРШРУТИЗАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей Міжнародної науково-практичної конференції м. Запоріжжя, 13-15 квітня 2006 року / Під заг. ред. Д.М. Пізи. – Запоріжжя: ЗНТУ. 2006. С. 163-165.

24. Ладыженский Ю.В., Мирецкая В.А. ИССЛЕДОВАНИЕ МУРАВЬИНЫХ АЛГОРИТМОВ МАРШРУТИЗАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ // ІНТЕРНЕТ-ОСВІТА-НАУКА-2006, п’ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006. Збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця. 2006. С. 375-377.

25. RFC2328. OSPF Version 2// http://www.faqs.org/rfcs/rfc2328.html

26. Ad hoc On-Demand Distance Vector (AODV) Routing // http://www.faqs.org/rfcs/rfc3561.html