УДК 004.724.4

АЛГОРИТМ МАРШРУТИЗАЦИИ СЕМЕЙСТВА Q-ROUTING, ОСНОВАННЫЙ НА ДИНАМИЧЕСКОМ ИЗМЕНЕНИИ КОЭФФИЦИЕНТОВ ОБУЧЕНИЯ ЗА СЧЕТ ОЦЕНКИ СРЕДНЕЙ ЗАДЕРЖКИ В СЕТИ

Ю.А. Шилова,
Пермский национальный исследовательский политехнический университет

Разработан алгоритм маршрутизации Adaptive Rate Full Echo, относящийся к семейству Q-routing, который по итогам имитационного моделирования показал хорошие результаты. Этот алгоритм основан на динамическом изменении коэффициентов обучения за счет оценки средней задержки в сети. Приводится краткий обзор различных протоколов и алгоритмов маршрутизации, дается более подробное описание базового алгоритма Q-routing.

Ключевые слова: *Q-routing, сеть, маршрутизация, алгоритм, коэффициент обучения.*

ВВЕДЕНИЕ

На данный момент мобильные ad hoc сети являются одной из перспективных технологий и по многим прогнозам ожидается, что такие сети в ближайшее время будут занимать все большую долю на рынке. Слово «ad hoc» – латинского происхождения, означает «для данного случая». Предполагается, что сеть ad hoc состоит из множества узлов, свободно перемещающихся относительно друг друга, причем в сети нет единого координирующего центра, отсутствует предварительно выстроенная инфраструктура, поэтому каждый из узлов такой сети одновременно становится и конечным устройством, принимающим и отправляющим пакеты, и устройством, осуществляющим построение возможных маршрутов и передачу данных в этой сети. Сетевая топология и соединения между узлами формируются самими узлами в процессе коммуникации друг с другом. Отсутствие заранее выстроенной инфраструктуры позволяет формировать такие сети для решения различных задач, подбирая наиболее удобную конфигурацию для каждой в отдельности. Самым простым примером таких сетей являются сети смартфонов, выстраивающих взаимодействие между собой для передачи какой-либо информации. В качестве других примеров можно привести сети транспортных средств, мобильные сенсорные сети, сети так называемого интернета вещей (Internet of Things), группы взаимодействующих роботов.

Исходя из этого можно сделать вывод, что сети типа ad hoc могут быть весьма эффективными в условиях непредвиденных изменений внешних условий. Одним из важных способов достижения такой эффективности является маршрутизация, то есть выбор пути передачи пакетов данных. В свою очередь маршрутизация является проблемой для сетей данного типа из-за двух основных свойств, присущих таким сетям. Первое из них заключается в том, что взаиморасположение элементов в сети не является постоянным, а следовательно, влечет периодическое изменение структуры сети в целом. Второе свойство заключается в отсутствии единого координирующего центра, что является достаточно неприятным обстоятельством, так как оно порождает проблемы обслуживания такой сети с учетом интегральных (относящихся ко всей сети) характеристик. В частности, это усложняет задачу маршрутизации.

Одним из интересных направлений в развитии алгоритмов маршрутизации является применение алгоритмов семейства Q-routing [1], использующих принципы обучения с подкреплением [2]. Разработан новый алгоритм данного семейства, названный Adaptive Rate Full Echo, который по итогам имитационного моделирования показал хорошие результаты. В настоящей статье дается краткий обзор различных протоколов и алгоритмов маршрутизации, дается более подробное описание алгоритма Q-routing, а также приводится описание предлагаемого алгоритма Adaptive Rate Full Echo.

КРАТКИЙ ОБЗОР ПРОТОКОЛОВ МАРШРУТИЗАЦИИ

В настоящее время по принципу работы можно выделить три класса протоколов, осуществляющих маршрутизацию в ad hoc-сетях: проактивные, реактивные и гибридные [3].

Проактивные (табличные) протоколы маршрутизации

К данному типу относят протоколы, требующие предварительного построения таблицы маршрутизации, в которую включаются все известные маршруты. В этом случае необходимая информация о топологии сети обрабатывается до начала передачи пакетов, что впоследствии не требует больших дополнительных затрат времени на построение маршрутов во время эксплуатации сети. Естественно, в процессе работы сети необходимо поддерживать достоверные сведения о существующей на данный момент конфигурации, поэтому протоколы в некоторые моменты времени передают по сети служебные сообщения, содержащие информацию обо всех изменениях связей между элементами сети. После получения такого сообщения каждый узел перестраивает все маршруты до других узлов сети, имеющиеся в его таблице маршрутизации, и перезаписывает ее. Впоследствии при передаче пакета какому-либо узлу-получателю, узел-отправитель обращается к необходимой информации, записанной в его таблице маршрутизации.

С одной стороны, проактивные протоколы используют заранее сформированные таблицы маршрутизации, что значительно уменьшает задержку при передаче информации, а с другой – вследствие периодического обмена узлами служебной информацией пропускная способность сети заметно сокращается, к тому же если узлы становятся более подвижными, то еще и загрузка сети резко увеличивается.

К табличным протоколам относятся такие протоколы, как DSDV (Destination Sequenced Distance Vector), OLSR (Optimized Link State routing), FSR (Fisheye State routing).

Реактивные протоколы маршрутизации

Данный тип протоколов – так называемые «протоколы по требованию». Постоянное хранение маршрутной информации не входит в обязанности узлов при исполь-

зовании в сети реактивных протоколов. Когда узлу поступает запрос на предоставление сообщения определенному конечному пункту, узел активизирует механизм построения маршрута, принцип действия которого заключается в следующем: узел-отправитель с помощью рассылки широковещательных запросов отправляет запросы на построение маршрута всем своим соседям, которые в свою очередь пересылают их своим соседям. Таким образом, путем повтора вышеописанных действий другими участниками сети выстраивается маршрут до узла-получателя. Соответственно полученный маршрут записывается в таблицах маршрутизации, чтобы в дальнейшем при передаче пакета с данными тому же узлу-получателю можно было не тратить дополнительное время на построение маршрута, а воспользоваться уже имеющейся информацией. Пропускная способность сети, обслуживание которой основано на использовании реактивных протоколов, больше, чем при использовании проактивных. Однако возникновение запросов на передачу пакетов между узлами без имеющегося маршрута вносит существенную задержку собственно на построение этого маршрута.

Примером реактивных протоколов являются такие, как DSR (Dynamic Source routing), AODV (Ad-hoc On-Demand Distance Vector), TORA (Temporally Ordered routing Algorithm), DYMO (The Dynamic MANET On-demand routing protocol).

Гибридные протоколы маршрутизации

Само название такого типа протоколов говорит об объединении двух вышеописанных механизмов формирования таблиц маршрутизации. В частности, сеть может быть разделена на некоторое количество подсетей, внутри которых взаимодействие между элементами сети выполняется с помощью проактивных протоколов, а реактивные протоколы обслуживают коммуникацию подсетей между собой. Из чего следует вывод, что уменьшаются не только размеры таблиц маршрутизации элементов подсетей, но и объем служебной информации, так как отдельно формируются маршруты между подсетями и отдельно внутри подсетей.

Примерами гибридных протоколов являются протоколы ZRP (Zone routing Protocol), HWMP (Hybrid Wireless Mesh Protocol).

Алгоритм Q-routing

Теперь обратимся к алгоритмам маршрутизации, которые используют в своей основе принципы обучения с подкреплением [2, 4]. Для начала более подробно рассмотрим базовый алгоритм маршрутизации Q-routing.

В статье [1] впервые был предложен алгоритм Q-routing, основанный на принципах обучения с подкреплением (reinforcement learning) [2], в частности на основе так называемого Q-обучения (Q-learning). Так как алгоритм Q-routing является базовым для семейства алгоритмов, к которому мы обращаемся, подробнее остановимся на принципах работы алгоритма Q-routing.

Для того чтобы сформировать собственное представление о конфигурации сети, у каждого узла имеется Q-таблица, структура которой представлена на рис. 1. Как видно из рис. 1, столбцы в таблице соответствуют ближайшим соседям для узла-хозяина таблицы, при этом строки таблицы соответствуют всем известным узлам в сети. На пересечении определенных строки и столбца записывается оценка задержки, возникающей при передаче конечному узлу (определяется строкой) через один из соседних узлов (определяется столбцом). В процессе выполнения алгоритма выбирается один из соседних узлов, имеющихся в таблице узла-отправителя, которому впоследствии и передается пакет. Но необходимо сделать одно важное замечание, что Q-значение в таблице не является точным значением задержки, а лишь ее оценкой, так как получается в ходе обучения. Поэтому выбор маршрута на основе данных Q-таблицы в одном случае может оказаться более, а в другом – менее эффективным.

Q-таблица

Конечный узел	Сосед		
	2	5	7
1	3,25	10,5	8,71
2	7,11	1,32	5,8
3	12,49	15,87	11,58
...	...	...	...
36	9,1	5,42	13,3

1...m, где m – количество соседей

Оценка, показывающая, сколько времени будет передаваться пакет до конечной цели

Конечный узел маршрута пакета

Рис. 1. Структура Q-таблицы узла

Несмотря на это, одним из достоинств данного алгоритма является то, что в процессе изменения сети он подстраивается под новую конфигурацию на протяжении всего времени эксплуатации сети, изменения которой постепенно фиксируются в Q-таблицах для каждого из участников сети. То есть каждый из узлов сети является своего рода самообучающимся агентом, действующим согласно принципам Q-обучения (Q-learning).

Последовательность действий алгоритма маршрутизации Q-routing может быть представлена с точки зрения двух алгоритмов. Первый алгоритм –получение пакета узлом, второй – передача пакета одному из имеющихся соседних узлов. Оба эти алгоритма в контексте шагов представлены в работе [5]. В данной же статье рассмотрим работу алгоритма Q-routing более подробно на определенном примере, совместно описывая и используя оба алгоритма. В качестве топологии сети возьмем типовую структуру сети, представленную на рис. 2, которая использовалась в статье [1], поэтому при моделировании алгоритма и была взята за основу. Кроме того, данный пример является одним из типовых примеров, на основе которого сравниваются между собой различные алгоритмы маршрутизации. Как видно из рис. 2, сеть состоит из 36 узлов, связанных между собой определенным образом, каждый узел обозначается уникальным номером-идентификатором. Условно сеть можно разделить на две части, для связи между которыми имеются два пути: кратчайший, проходящий между узлами с № 14 и 15, и обходной, проложенный между узлом № 0 и узлом № 5. Для более наглядного представления раз-

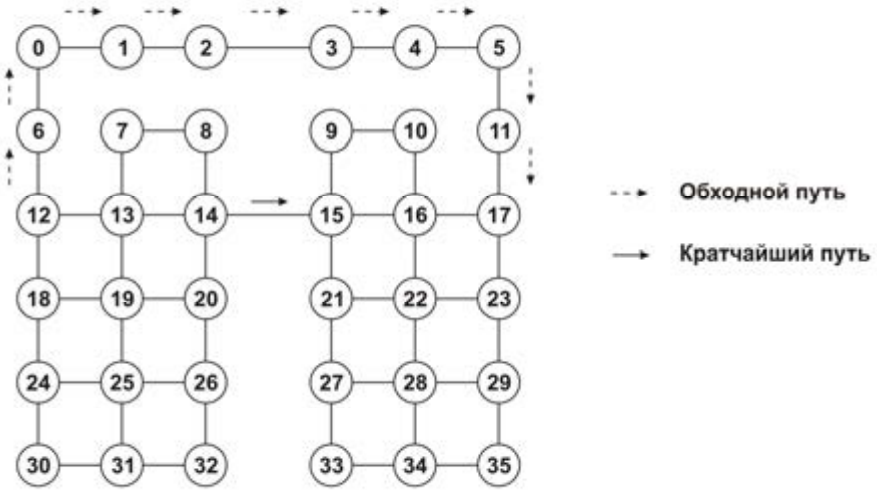

Рис. 2. Типовая структура сети для моделирования

берем выполнение последовательности шагов на простом примере. Единственное, что хотелось бы уточнить, в описываемом примере таблицы маршрутизации на данном этапе уже заполнены некоторыми значениями, которые слегка упрощают последовательность действий в отличие от первоначального запуска сети.

Допустим, узел, желающий передать пакет, имеет свой порядковый номер в сети, например 25. Номер узла, для которого предназначен пакет – узла-получателя – пусть, к примеру, будет равен 28. Узел № 25 имеет четырех непосредственных соседей: это узлы с номерами 19, 24, 26 и 31. Таблица маршрутизации для узла № 25 до выполнения алгоритма представлена в табл. 1.

Таблица 1

Таблица маршрутизации узла № 25 до передачи пакета

Конечный узел	Сосед			
	19	24	26	31
...	...	...	...	...
28	1,35	3,5	10,18	2,04
...	...	...	...	...

Если пакет, который мы должны отправить узлу № 28, стоит первым в очереди на передачу, то обращаемся к таблице маршрутизации узла-отправителя, в нашем случае – узла с порядковым номером 25. В ней выбираем строку, название которой соответствует номеру узла-получателя, то есть для рассматриваемого примера – строку с номером 28. Затем сравниваем друг с другом значения, стоящие на пересечении столбцов с выбранной строкой. Сосед, значение оценки задержки у которого оказывается минимальным из всех возможных, будет использоваться в качестве промежуточного узла для пересылки конечному узлу. В данном примере (табл. 1) – это узел № 19, поэтому узел № 25 пересылает именно ему пакет данных для конечного узла № 28. Узел № 19, получив пакет, анализирует адрес, по которому надо его отправить, и добавляет пакет в выходную очередь для дальнейшей передачи. Узел № 19 повторяет те же действия, что и узел № 25 при отправке пакета, только уже основываясь на собственной таблице маршрутизации. Но перед тем как отправить пакет следующему промежуточному узлу, узел № 19 должен возвратить узлу, приславшему пакет, в нашем примере – это узел № 25, предполагаемую задержку для передачи пакета установленному адресату. Обозначим эту предполагаемую задержку (оценку задержки) как $Q_y(Z, D)$, где Z – номер узла-соседа с минимальным Q-значением для конечного узла D (в нашем примере – узла № 25) в таблице узла Y – узла № 19. Другими словами, у узла № 19 имеется собственная таблица, аналогичная табл. 1, заполненная некоторыми своими значениями и узел № 19 выбирает из строки с номером $D = 28$ минимальное значение $Q_y(Z, D)$, находящееся в столбце, соответствующем узлу-соседу с номером Z. Затем узел № 19 пересылает пакет своему узлу-соседу с номером Z. В то же время узел № 25, получив оценку $Q_y(Z, D)$ от узла № 19, рассчитывает новое значение задержки $Q_x^{est}(Y, D)$ по формуле

$$Q_x^{est}(Y, D) = q + s + Q_y(Z, D), \qquad (1)$$

где q – задержка в текущем узле, s – время передачи между узлами (отправитель-получатель), при моделировании это расстояние было равным единице, Y – номер узла-соседа, которому был передан пакет.

Полученное значение $Q_x^{est}(Y, D)$ обновляет старое значение $Q_x^{old}(Y, D)$, находящееся на пересечении строки D со столбцом Y, согласно следующей формуле

$$Q_x^{new}(Y, D) = Q_x^{old}(Y, D) + \\ +\eta \cdot \left(Q_x^{est}(Y, D) - Q_x^{old}(Y, D)\right), \qquad (2)$$

где η – коэффициент обучения.

Для примера условимся, что $q = 1$, $s = 1$, $Q_y(Z, D) = 2{,}88$, $\eta = 0{,}9$.

Закончив расчет, узел № 25 заменяет значение, находящееся на пересечении строки с номером 28 и столбца, означающего соседа № 19, на величину, полученную в результате расчетов. Соответственно, после перерасчета таблица маршрутизации узла № 25 будет выглядеть следующим образом (табл. 2).

По данным табл. 2 можно сделать вы-

Таблица 2

Таблица маршрутизации узла № 25 после передачи пакета

Конечный узел	Сосед			
	19	24	26	31
…	…	…	…	…
28	4,527	3,5	10,18	2,04
…	…	…	…	…

вод, что если в другой раз у узла № 25 появится необходимость передать пакет в конечный узел № 28, то следующим промежуточным узлом будет выбран уже не узел № 19, а сосед с номером 24, так как теперь он имеет минимальное значение задержки из всех представленных.

Таким образом, повторяя подобные действия для каждого из промежуточных узлов, постепенно пакет данных добирается до конечного узла. С каждой новой пересылкой пакета в тот же узел (в нашем примере – это узел № 28) будет происходить уточнение значений в строке 28 в таблицах тех узлов, через которые этот пакет проходит. За счет таких последовательных уточнений у каждого узла будет уточняться информация о том, какому соседу лучше пересылать пакет с таким-то адресом получателя. Тем самым неявным образом в сети в распределенной форме хранится информация обо всех маршрутах для пакетов разных пар отправитель-получатель. Например, для пакета из узла № 25 в узел № 28 согласно такой распределенной информации может оказаться, что наилучшим маршрутом является маршрут, представленный на рис. 3. При этом с каждой новой пересылкой пакетов информация о маршрутах уточняется. В этом и заключаются принцип самообучения такой сети и принцип работы маршрутизации, основанной на обучении с подкреплением.

Аналогичным образом формируются маршруты для других пакетов, время от времени возникающих в сети.

Допустим, в данный момент загрузка сети пакетами небольшая и есть возможность передавать все пакеты между левой и правой частью сети по кратчайшему пути (см. рис. 2), то есть через канал между узлами № 14 и № 15. Предположим, что загрузка начинает увеличиваться и в какой-то момент пропускной способности канала, через который проходит кратчайший путь, начинает не хватать. На этом коротком пути возникает «затор», и задержки очень сильно возрастают. Именно в этот самый момент начинается вторая волна обучения алгоритма Q-routing. Принцип работы алгоритма не меняется, но так как задержки на кратчайшем пути начинают увеличиваться, а значения задержки в таблицах маршрутизации узлов, через которые проходит обходной путь, оказываются меньше, чем у узлов короткого пути, то сеть начинает переобучаться, постепенно выстраивая маршруты через обходной путь, тем самым уменьшая среднюю задержку при передаче пакетов.

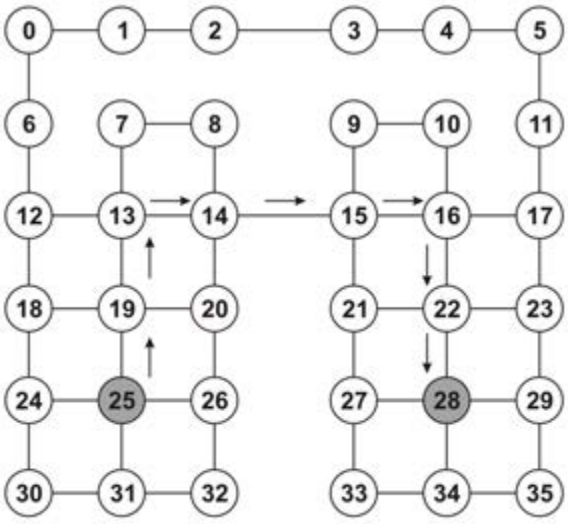

Рис. 3. Построение маршрута между узлами № 25 и № 28

РАЗРАБОТКА ПРОГРАММЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

С целью проведения экспериментальных исследований и сравнения различных алгоритмов маршрутизации была разработана программа имитационного моделирования, причем решено было не использовать известные пакеты моделирования, такие как QualNet Developer 4.5 или Network Simulator 3, а реализовывать алгоритм маршрутизации сразу на C++, также как среду моделирования и модель сети.

Изначально в сети имеется некоторое количество узлов, у каждого узла есть своя таблица значений, поэтому решено было использовать класс Vector, принадлежащий стандартной библиотеке языка C++, для воплощения такой структуры, поскольку он позволяет реализовывать динамически изменяющиеся таблицы, автоматически выделяя необходимую память. В нужные моменты, зная принцип работы данного класса, можно легко добавлять и удалять элементы, менять их местами и производить другие дополнительные действия, если того требует алгоритм, в котором используется данный контейнер.

Формирование таблицы маршрутизации для каждого из появляющихся в сети узлов выполняется единообразно и реализуется с помощью функции «TableNodes QR», которая, в частности, может быть представлена следующим образом:

```
void TableNodesQR (vector<vector<int> > const &Weight_matrix,
vector<vector<vector<double> > >&QTable){
  int i, j, node;
  for (node=0; node< QTable.size(); node++){
    for (i=0; i< QTable[node].size(); i++){
     for  (j=0;  j<Weight_matrix[node].size(); j++){
       if (Weight_matrix[node][j]==1)
         QTable [node][i].push_back(0);  }}}
  for (node=0; node< QTable.size(); node++){
    for (i=0;i<QTable[node][node].size(); i++){
     QTable [node][node][i]=0;  }}}
```

После выполнения функции Table NodesQR, таблицы маршрутизации для каждого из узлов сети выглядят почти одинаково (табл. 3), за исключением количества столбцов, так как количество ближайших соседей у каждого из узлов свое.

Таблица 3

Первоначальный вид таблицы маршрутизации

Конечный узел	Сосед			
	1	2	3	…
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
…	0	0	0	0

Структура сети может быть различной, в зависимости от модели, которую необходимо исследовать, поэтому для создания структуры сети была написана специальная функция void CreateMatrix Nodes (vector<vector<int> > &x, int size).

Интенсивность создания пакетов задается распределением Пуассона $\Pi(\lambda)$ с параметром $\lambda > 0$. Распределение Пуассона – вероятностное распределение дискретного типа, моделирует случайную величину, представляющую собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга. Реализация распределения Пуассона выполнена с помощью алгоритма PEXP (Poisson Exponential) – моделирование распределения $\Pi(\lambda)$ через случайные величины с показательным распределением [6]. Программное описание данного алгоритма достаточно простое и может быть представлено следующей функцией:

```
int IntensPack(double lambda) {
  int QuantInsPack = 0;
  double p, q, alpha;
  p = exp(-lambda);
  q = rand()  /  double ( RAND_MAX );
  while (q >= p)   {
    alpha = rand()  /  double ( RAND_MAX );
    q=q*alpha;
    QuantInsPack = QuantInsPack +1;  }
  return QuantInsPack;}
```

Использование языка C++ для разработки программы имитационного моделирования обосновывается еще и тем, что различные алгоритмы маршрутизации будут кодироваться именно на этом языке, а не на специальном языке той или иной среды моделирования. Такой подход позволит сэкономить время на программирование при реализации библиотеки алгоритмов маршрутизации, которую представляется более удобным разрабатывать именно на языке C++.

СРАВНЕНИЕ Q-ROUTING И SHORTEST PATH

При моделировании обоих алгоритмов, алгоритма Shortest Path и алгоритма Qrouting, удобно ориентироваться на результаты, представленные в статье [1].

Чтобы сравнить возможности работы обоих алгоритмов в разработанной среде моделирования задавались различные параметры загрузки сети, в результате чего получены следующие результаты (табл. 4).

Для наиболее наглядного представления изобразим результаты, представленные в табл. 4, с помощью графика, представленного на рис. 4.

Безусловно, сравнивая алгоритмы Q-routing и Shortest Path между собой, напрашивается не совсем однозначный вывод, так как при малых загрузках наибольшей эффективностью, исходя из графика рис. 4, обладает именно алгоритм Shortest Path. Но это преимущество очень небольшое по сравнению с преимуществом Q-routing на больших загрузках. Другими словами, Q-routing показывает наилучшие результаты с увеличением загрузки, в то время как Shortest Path не справляется с загрузкой сети $\lambda = 2{,}5$.

Однако у алгоритма Q-routing имеется два существенных недостатка, первый из которых заключается в том, что Q-routing не всегда находит кратчайший путь при условиях низкой загрузки сети. Если значение задержки в Q-таблице у длинного пути меньше, чем Q-значение более короткого пути, то алгоритм Q-routing в соответствии с меньшим значением будет посылать пакеты по более длинному пути. Второй недостаток – неспособность алгоритма Q-routing приспособиться к более коротким путям, когда загрузка в сети уменьшается. При большой загрузке дан-

Таблица 4

Сводные значения средней задержки в случае алгоритмов Q-routing и Shortest Path

Алгоритм	Загрузка сети, λ						
	0,5	1	1,5	2	2,5	3	3,5
Q-routing	5,930	6,698	6,938	7,459	9,316	11,497	30,699
Shortest Path	5,483	5,867	6,100	7,151	21,393	7217,410	–

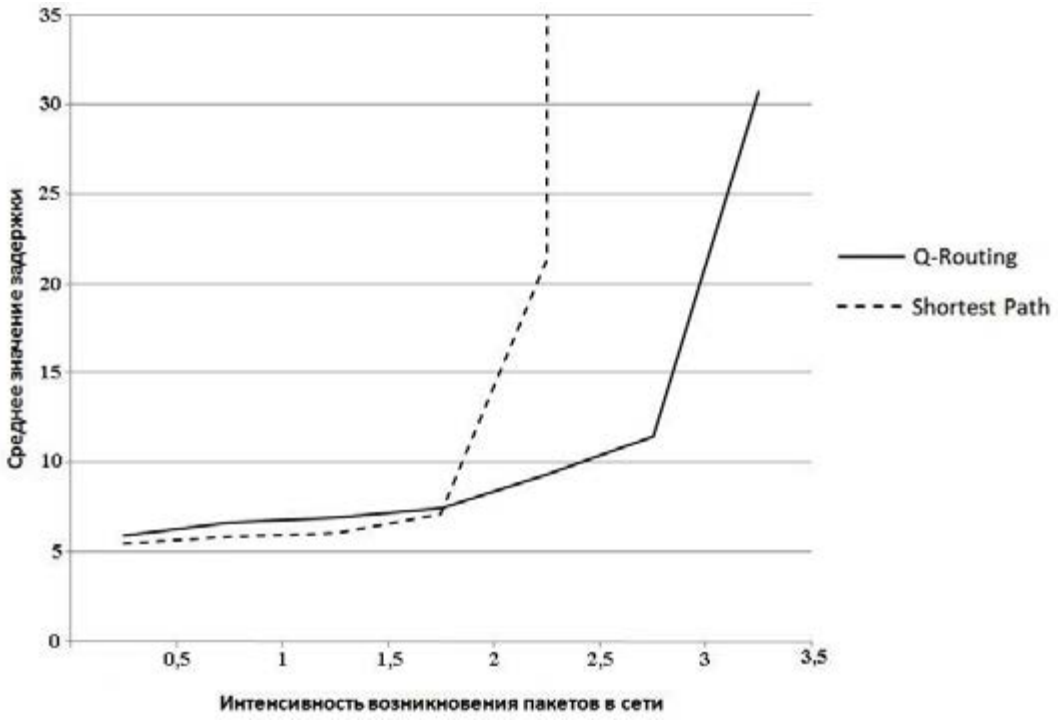

Рис. 4. Сравнение алгоритмов Q-routing и Shortest Path

ный алгоритм начинает искать обходные пути, когда сильно увеличивается время ожидания в очереди по короткому пути, и в какой-то момент выбор более длинного пути становится предпочтительнее. Через некоторый момент времени задержка на коротком пути может уменьшиться, но передача пакетов продолжится по длинному пути до тех пор, пока задержка в Q-таблице не превысит значение, соответствующее короткому пути.

С целью компенсации недостатков базового алгоритмов Q-routing появилось целое семейство алгоритмов маршрутизации, основанных на сходных принципах самообучения. К этому семейству относится сам базовый алгоритм Q-routing, а также такие алгоритмы, как Predictive Q-routing, Dual Reinforcement Q-routing, Policy-Gradient Q-routing, Ant-Based Q-routing и другие. Рассмотрим некоторые из них чуть более подробно.

ДРУГИЕ АЛГОРИТМЫ СЕМЕЙСТВА Q-ROUTING

Predictive Q-routing

Модификация алгоритма Q-routing под названием Predictive (прогнозирующий) Q-routing была разработана S. Choi и D. Yeung, чтобы устранить два вышеизложенных недостатка оригинального алгоритма «Q-routing» [7].

Прогнозирующий алгоритм Q-routing (PQ-routing) помнит самые маленькие Q-значения каждого соседа. Когда Q-значение этих соседей позже увеличивается, алгоритм иногда посылает пакеты данных по кратчайшему пути, таким образом, проводя некое «зондирование обстановки». Целью зондирования является снижение среднего времени доставки пакетов в изменяющейся среде. Если обновленное Q-значение исследуемого пути остается высоким, то PQ-routing ведет себя аналогично алгоритму Q-routing. Если обновленное Q-значение зондируемого пути уменьшается, PQ-routing адаптируется путем выбора более короткого пути. С одной стороны, частое выполнение механизма зондирования увеличивает нагрузку на узлы перегруженных путей, а с другой – редкое зондирование не увеличивает производительность сети, обеспечивая маршрутизацию, не отличающуюся от Q-routing. Поэтому алгоритм PQ-routing пытается уравновесить эти крайности. Таким образом, периодически исследуя пути, которые были бездействующими, PQ-routing может вернуться к более коротким маршрутам после уменьшения нагрузки на сеть.

Dual Reinforcement Q-routing

S. Kumar и R. Miikkulainen предложили алгоритм Dual Reinforcement Q-routing (DRQ-routing) как модификацию к существующему Q-routing [8]. Обновление значений Q-таблицы узла, передающего пакет, происходит по той же схеме, которая лежит в основе алгоритма Q-routing. Отличие DRQ-routing заключается в том, что, помимо обновления Q-значений передающего узла, также обновляются Q-значения в таблице узла, принимающего пакет. Происходит это следующим образом: перед тем как передать пакет, узел-отправитель добавляет к нему информацию о значении задержки в своей Q-таблице. В результате этого узел, принявший пакет, обновляет Q-значение в своей таблице маршрутизации на основании дополнительной информации, полученной из принятого пакета.

Результаты экспериментов показали, что DRQ-routing находит лучшие маршруты в два раза быстрее, чем обычный базовый Q-routing. Это увеличение связано с расширением исследовательской способности DRQ-routing.

Policy-Gradient Q-routing

N. Tao, J. Baxter и L. Weaver предложили использование градиента как улучшение алгоритма Q-routing [9]. Цель данного метода состоит в том, чтобы сбалансировать предлагаемую загрузку и минимизировать время прохождения пакета до места

назначения для всех маршрутов в данной сети. Каждый узел рассматривается как независимый агент, который видит только часть пакетов, перемещающихся через сеть. Обновление значений таблиц происходит в направлении среднего градиента сумм задержек пакетов, дошедших до узла-получателя. Авторы подтверждают, что нереалистично немедленно передать сумму временных задержек ко всем узлам в сети оперативно, поэтому они передаются периодически широковещательно.

Такой метод предназначен для того, чтобы ускорить сходимость маршрутов в стационарной среде маршрутизации, но он не обеспечивает хороших результатов в неустановившихся средах с переменными условиями. Кроме того, широковещательные сообщения увеличивают нагрузку на пропускную способность сети, что плохо сказывается на времени прохождения пакетов до места назначения.

Ant-Based Q-routing

Еще один алгоритм, в основе которого лежит принцип самообучения, был создан D. Subramanian, P. Druschel и J. Chen по аналогии с тем, как ведут себя колонии муравьев, собственно, благодаря чему он и получил свое название [10].

В обычном Q-routing после каждой передачи пакета узлу-отправителю возвращается Q-значение узла-получателя. Отличие Ant-Based Q-routing состоит в том, чтобы отправлять маленькие сообщения, так называемых «муравьев», через сеть. Цель сообщения-муравья состоит в том, чтобы оценить затраты на пересечение проходов между узлами. Сообщение-муравей содержит идентификационные данные узлов-отправителей и узлов-получателей, а также информацию о «стоимости» пути. Узлы, получающие сообщение-муравья, используют информацию о «стоимости», чтобы обновить значения в своей Q-таблице. Есть два вида сообщений-муравьев. Регулярные сообщения, передающиеся через сеть от источника до получателя согласно таблицам маршрутизации, проходят каждый промежуточный узел. Пути, посещаемые регулярными муравьями, сходятся к лучшему пути в сети, создавая некоторое стабильное состояние-решение проблемы маршрутизации. Универсальные сообщения передаются любому соседу с равной вероятностью, тем самым продолжая исследовать сеть, чтобы найти лучшие направления при изменении состояния сети.

Помимо вышеперечисленных алгоритмов маршрутизации семейства Q-routing существует несколько модификаций базового алгоритма Q-routing, более подробное описание которых приводится в статьях [1, 4], а именно Q-routing Full Echo, Q-routing Random Echo и Q-routing ε-greedy Exploration. Поэтому кратко опишем каждую из существующих модификаций, представленных в той статье.

Q-routing Full Echo

Модификация Q-routing Full Echo отличается от простого Q-routing тем, что помимо возвращаемого Q-значения от соседа, которому в результате выполнения алгоритма передается пакет, все остальные соседи также посылают узлу-отправителю минимальные Q-значения из своих таблиц маршрутизации.

Q-routing Random Echo

Отличие модифицированной версии Q-routing Random Echo от Q-routing Full Echo состоит в том, что свои Q-значения возвращают не все соседи, имеющиеся у узла-отправителя, а только часть из них – в зависимости от генерируемого значения случайной величины и заданного параметра R. Этот параметр задается изначально перед запуском алгоритма, и только если случайная величина больше этого параметра, то выбранный сосед отправляет свою задержку.

Q-routing ε-greedy Exploration

В Q-routing ε-greedy Exploration с определенной долей случайности выбирается один из возможных соседних узлов в отличие от Q-routing, в котором всегда выбирается сосед с минимальных временем задержки в Q-таблице.

СРАВНЕНИЕ АЛГОРИТМОВ СЕМЕЙСТВА Q-ROUTING

В результате анализа всех вышеперечисленных алгоритмов, относящихся к семейству самообучающихся алгоритмов маршрутизации, встал вопрос об экспериментальном сравнении этих алгоритмов. Вследствие чего на реализованной программе моделирования, которая упомянута выше, были промоделированы все ранее перечисленные алгоритмы семейства Q-routing при прочих равных условиях. Но прежде чем привести результаты моделирования, для полноты картины стоит сказать, что после анализа имеющихся алгоритмов возникла идея о разработке собственного дополнения базового алгоритма Q-routing. В результате этого был разработан алгоритм Adaptive Rate Full Echo, принцип которого будет описан ниже.

Исходя из рис. 5, можно сделать вывод, что разработанный алгоритм Adaptive Rate Full Echo при небольшой загрузке сети частично уступает некоторым из имеющихся алгоритмов, но с увеличением загрузки явно видно, что полученные результаты превзошли результаты других алгоритмов. Естественно, что в общем случае нельзя сказать, что какой-то определенный алгоритм маршрутизации будет наиболее оптимальным при применении его для всех разновидностей ad hoc-сетей, так как в каждом конкретном случае требования к качеству сервиса в сети будут разными.

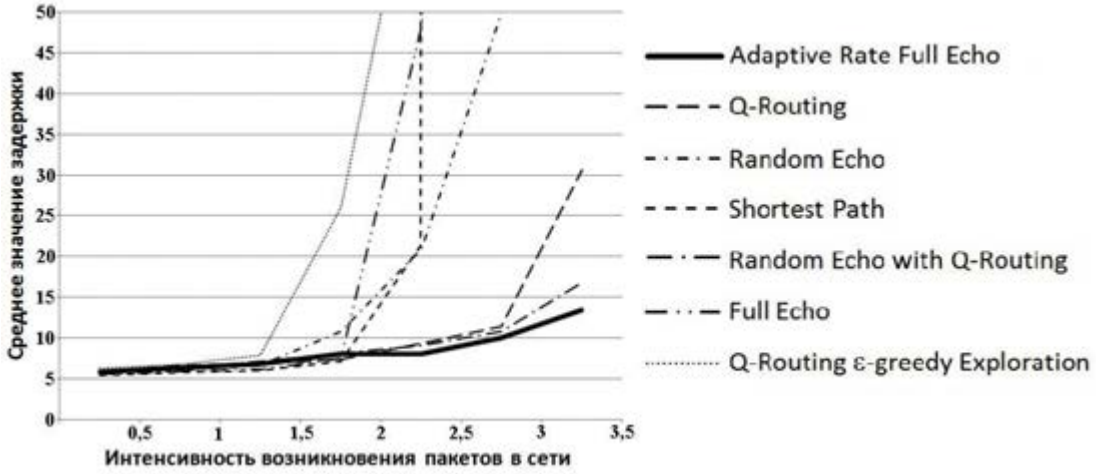

Рис. 5. Пример результатов имитационного моделирования алгоритмов семейства Q-routing

ОПИСАНИЕ АЛГОРИТМА ADAPTIVE RATE FULL ECHO

Теперь более подробно опишем разработанный алгоритм, названный Adaptive Rate Full Echo. Новизна данного решения заключается в дополнении базового алгоритма Q-routing Full Echo тремя новыми составляющими, а именно:

1) разделение коэффициента обучения на два типа;
2) изменение коэффициентов обучения на основе глобальных параметров сети;
3) получение глобальных параметров сети на основе распределенного сбора информации.

Прежде чем рассмотреть в отдельности каждое из трех дополнений, поясним, что за основу алгоритма Adaptive Rate Full Echo взят базовый алгоритм с модификацией Q-routing Full Echo. Таким образом, перерасчет значений в таблице маршрутизации узла, передающего пакет, происходит для всех соседей, независимо от того, является ли сосед следующим промежуточным или конечным узлом, которому передается пакет.

Два вида коэффициентов обучения

В базовом алгоритме Q-routing и в алгоритме Q-routing Full Echo используется один коэффициент обучения, который за-

дается перед началом проведения эксперимента и используется при обновлении значения задержки в столбце соседа, которому был передан пакет. Коэффициент обучения используется в формуле (2) для обновления значений Q-таблицы.

Несмотря на то, что в нашем случае появляется дополнительный второй коэффициент, ограничиваться только двумя коэффициентами необязательно, возможно использовать сразу несколько видов коэффициентов обучения для разных узлов и ячеек таблицы, но рассмотрение этого вопроса оставляется для будущих исследований.

Поясним на примере, каким образом осуществляется раздвоение коэффициентов обучения. Допустим, мы хотим передать пакет из узла № 8. Чтобы не возвращаться к началу статьи, представим на рис. 6 фрагмент сети, необходимый для объяснения работы данного дополнения.

Согласно рис. 6 узел № 8 имеет в ближайших соседях четыре узла – № 2, № 5, № 7 и № 10. Передавать пакет будем в конечный узел № 15. Соответственно, таблицы маршрутизации никоим образом не отличаются от таблиц маршрутизации, имеющихся у каждого узла в базовом алгоритме Q-routing. Поэтому представим, что таблица маршрутизации узла № 8 до передачи пакета кому-либо из соседних узлов выглядит следующим образом (табл. 5).

Как видно из табл. 5, минимальное время задержки для передачи пакета возникает при выборе узла № 2. Поэтому для пересчета значения ячейки, стоящей на пересечении строки с номером 15 и столбца с номером 2, используем коэффициент $\eta = 0{,}9$ согласно рис. 6. Пересчитаем его значение по формулам (1) и (2). Для более простого расчета примем, что задержка в текущем узле q = 1, расстояние между узлами (отправитель-получатель) $s = 1$, минимальное время передачи от узла-соседа $Q_y(Z, D) = 5{,}38$. После перерасчета получим, что $Q_s^{new}(2,15) = 7$. Как видно, пока никаких отличий от базового алгоритма Q-routing не наблюдается. Но все изменяется при вводе η_2 – второго коэффициента обучения для пересчета значений оставшихся соседей, не получивших пакет.

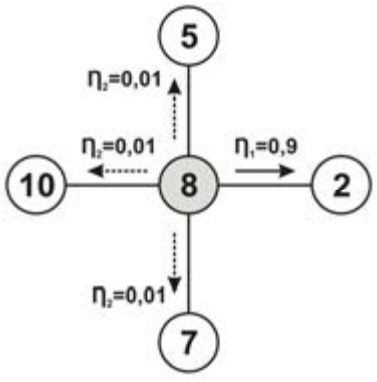

Рис. 6. Два вида коэффициентов обучения

В разработанном алгоритме η_2 используется для вычисления значений возможных задержек всех ближайших соседей, которые не получают пакета, но все равно возвращают оценки узлу-отправителю в соответствии с собственными таблицами маршрутизации согласно принципу, заложенному в базовом алгоритме Q-routing Full Echo.

Для наглядности примем параметр $\eta_2 = 0{,}001$. В результате всех вычислений получаем обновленную строку, соответствующую номеру конечного получателя сообщения, в таблице маршрутизации для узла № 8 после передачи пакета, представленную в табл. 6.

Анализируя табл. 6, можно сделать вывод, что оценки задержек для других соседей меняются мало, так как значение η_2

Таблица 5

Таблица маршрутизации узла № 8 до передачи пакета

Конечный узел	Сосед			
	2	5	7	10
…	…	…	…	…
15	3,58	10,5	8,71	5,04
…	…	…	…	…

Таблица 6

Таблица маршрутизации узла № 8 после передачи пакета

Конечный узел	Сосед			
	2	5	7	10
…	…	…	…	…
15	7	10,497	8,709	5,042
…	…	…	…	…

мало. Это придает стабильность распределенной информации о маршрутах, она становится менее подверженной «автоколебаниям» из-за периодических изменений в соседних узлах. Однако η_2 отличен от нуля, поэтому возможность переобучения (при изменении условий в сети) остается. Таким образом, разделение коэффициентов обучения дает возможности более тонкой настройки процесса обучения.

Изменение коэффициентов обучения на основе глобальных параметров сети

В первоначальной модификации Q-routing Full Echo, как на это было впервые указано в работе [1], из-за того, что узлы постоянно переобучаются, они входят в «автоколебания», так называемые «вредные» осцилляции, представленные на рис. 7, тем самым не позволяя понизить задержку в сети до минимально возможного уровня.

Для борьбы с этими осцилляциями предлагается динамически изменять коэффициент η_2 согласно формуле

$$\eta_2 = \frac{\overline{t}}{\overline{t}_{\max}} \cdot \eta \cdot k, \qquad (3)$$

где $\overline{t}$ – оценка средней задержки в сети, $\overline{t}_{\max}$ – максимально достигнутая средняя задержка в сети, k – коэффициент, обычно меньше единицы, определяющий, во сколько раз параметр η_2 будет отличаться от η.

При этом в разработанном алгоритме Adaptive Rate Full Echo при уменьшении средней задержки в сети коэффициент обучения уменьшается согласно формуле (3) и сеть стабилизируется под новую конфигурацию. Это позволяет зафиксировать текущую конфигурацию с низкими задержками и устранить вредные осцилляции (рис. 8), которые, как выше упоминалось, возникают в базовом алгоритме. На рис. 8 представлен коэффициент обучения с индексом i, что соответствует более общему подходу к модификации алгоритма. В рамках данной статьи и в случае разработанного алгоритма Adaptive Rate Full Echo рассматривается лишь вариант, когда $i = 2$.

Получение глобальных параметров сети на основе распределенного сбора информации

Для того чтобы в каждом узле изменять значение коэффициента η_2 согласно формуле (3), необходимо, чтобы узел формировал оценку текущей средней задержки по всей сети. Но у него нет информации обо всей сети, кроме своих соседей. Поэтому предлагается следующий подход для получения такой оценки.

С определенным периодом времени (этот период может настраиваться) между узлами сети передаются служебные широковещательные сообщения, позволяющие собрать общую глобальную информацию обо всей сети и распространить ее между всеми узлами. С помощью такого механизма собирается информация о таком глобальном параметре. В частности, в случае алгоритма Adaptive Rate Full Echo в качестве такого глобального параметра выступает средняя задержка в сети.

Рис. 7. Возникновение «вредных» осцилляций в Q-routing Full Echo

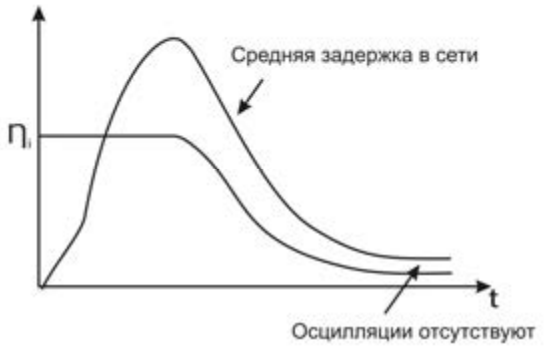

Рис. 8. Отсутствие «вредных» осцилляций в Adaptive Rate Full Echo

Оценку средней задержки в узле предлагается рассчитывать по следующей формуле:

$$\overline{t} = \frac{\sum_{i}^{n} \overline{t_i}}{n}, \qquad (4)$$

где $\overline{t}$ – оценка средней задержки по сети, $\overline{t_i}$ – средняя задержка по узлу № i, n – количество узлов.

Значения средней задержки по узлу № i собираются в данном узле за счет приема вышеуказанных служебных широковещательных сообщений.

Для большей наглядности приведем пример расчета средней задержки для одного конкретного узла. Возьмем за основу таблицу маршрутизации, представленную в табл. 5, и немного изменим ее (см. табл. 7). Пусть в сети имеется не 15, а всего 10 узлов (для того, чтобы таблица была меньше). Пусть значения по столбцам заполнены в ходе процесса обучения. Будем считать среднюю задержку для узла № 8, которому и принадлежит данная таблица. Согласно табл. 7 узел № 8 имеет четырех ближайших соседей – узлы № 2, № 5, № 7 и № 10.

В каждой из 10 строк в табл. 7 выбирается минимальное значение из всех, представленных в столбцах. Затем выбранные 10 значений суммируются и общая сумма делится на количество слагаемых. Таким образом, в нашем примере средняя задержка для узла № 8 равна 5,264. Эта задержка с помощью служебных широковещательных сообщений доставляется до всех остальных узлов в сети, формируя одно из слагаемых числителя в формуле (4). Аналогичным образом другие узлы рассчитывают свои средние задержки и передают их служебными широковещательными сообщениями. Эти сообщения, в конце концов, доходят до узла № 8, тогда он может полностью сформировать числитель формулы (4), а также и знаменатель, который определяется количеством уникальных узлов, приславших узлу № 8 свои служебные широковещательные сообщения. Таким образом, узел № 8 по формуле (4) рассчитывает оценку средней задержки по сети, которую подставляет в формулу (3). Полученное с помощью формулы (3) значение коэффициента η_2 используется для обновления значений Q-таблицы узла № 8.

Таблица 7

Таблица узла № 8 для расчета средней задержки узла

Конечный узел	Сосед			
	2	5	7	10
1	18,119	5,778	18,046	6,947
2	2,966	14,817	8,287	15,126
3	7,534	7,139	7,624	19,288
4	17,462	14,701	20,943	17,010
5	12,585	19,553	17,440	11,182
6	3,789	1,128	14,300	17,876
7	17,799	8,395	7,709	3,985
8	13,879	16,742	2,133	1,158
9	19,023	2,034	8,955	20,776
10	6,760	7,062	2,569	3,633

ЗАКЛЮЧЕНИЕ

По полученным результатам имитационного моделирования (например, см. рис. 5) можно сделать вывод, что разработанный алгоритм Adaptive Rate Full Echo показывает достаточно хорошие результаты по сравнению с другими алгоритмами маршрутизации. Однако следует отметить, что разработанный алгоритм является лучшим далеко не во всех случаях. Выбор того или иного алгоритма маршрутизации во многом зависит от текущих условий сети, а также критерия качества сервиса для данной сети. На данный момент продолжаются исследования семейства алгоритмов, сформированных на основе предложенных дополнений базового алгоритма Q-routing. В частности, помимо разработанного алгоритма Adaptive Rate Full Echo, планируется исследовать другие варианты динамического изменения коэффициентов обучения, а также другие варианты используемых глобальных параметров сети. В любом случае, разработанный алгоритм, а также другие подобные алго-

ритмы семейства Q-routing могут стать значимым дополнением существующего набора алгоритмов маршрутизации, применяемых на практике.

Библиографический список

1. *Boyan J.A., Littman M.L.* Packet routing in dynamically changing networks: A reinforcement learning approach // Advances in neural information processing systems. – 1994. – P. 671–671.
2. *Sutton R., Barto A.G.* Reinforcement Learning: An Introduction: – Cambridge : MIT Press, 1998. – 328 p.
3. *Винокуров В.М., Пуговкин А.В.* Маршрутизация в беспроводных мобильных Ad hoc-сетях // Доклады ТУСУРа, № 2 (22), ч. 1, декабрь 2010, с. 288–292.
4. *Desai R., Patil B.P.* Reinforcement learning for adaptive network routing //Computing for Sustainable Global Development (INDIACom), 2014 International Conference on. – IEEE, 2014. – P. 815–818.
5. *Шилова Ю.А., Кавалеров М.В.* Исследование влияния параметра скорости обучения на результаты работы алгоритма маршрутизации Q-routing // Инновационные технологии: теория, инструменты, практика: Сб. тр. международн. конф. – Пермь, ПНИПУ, 2015. – С. 172–179.
6. *Некруткин В.В.* Моделирование распределений. – [Электрон. данные]. – 90 с.
7. *Choi S., Yeung D.Y.* Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control // Advances in Neural Information Processing Systems. – 1996. – Т. 8. – P. 945–951.
8. *Kumar S., Miikkulainen R.* Dual Reinforcement Q-routing: An On-Line Adaptive routing Algorithm // In Proceedings of the Artificial Neural Networks in Engineering Conference (St. Louis), 1997. – P. 231–238.
9. *Tao N., Baxter J., Weaver L.* A multi-agent, policy-gradient approach to network routing // In Proceedings of the 18th Int. Conf. on Machine Learning. – 2001. – P. 553–560.
10. *Subramanian D., Druschel P., Chen J.* Ants and reinforcement learning: A case study in routing in dynamic networks // IJCAI (2). – 1997. – P. 832–839.

ROUTING ALGORITHM OF THE FAMILY OF Q-ROUTING, BASED ON DYNAMIC CHANGES OF LEARNING RATES ACCORDING TO THE ESTIMATES OF THE AVERAGE DELIVERY TIME

Yu.A. Shilova

Institute of Ecology and Genetics of Microorganisms of the Ural Branch of the RAS

One of the most urgent problems for mobile ad hoc network is the routing problem. A routing algorithm of the family of Q-Routing is developed and called Adaptive Rate Full Echo, and simulation of this algorithm has yielded good results. This algorithm is based on dynamic changes of learning rates according to estimates of the average network delay. The article also provides a brief overview of different routing protocols and algorithms, it provides a more detailed description of the basic algorithm of Q-Routing.

Keywords: Q-routing, network, routing, algorithm, learning rate.

Сведения об авторе

Шилова Юлия Александровна, магистрант группы Телекоммуникации1-13-1м, Пермский национальный исследовательский политехнический университет (ПНИПУ), 614013, г. Пермь, ул. Профессора Поздеева, 7; e-mail: marissaspiritte@mail.ru

Материал поступил в редакцию 25.05.2015 г.