ネットワークの負荷分散を行うための評価関数に関する考察

鵜根 弘行 * 池坊 繁屋 † 銭 飛 ‡

* 広島国際学院大学 情報学部 情報デザイン学科
† 広島国際学院大学 情報学部 情報工学科
‡ 関東学院大学 工学部 電気電子情報工学科

本研究では、ネットワークのトラフィックを動的に分散させるための分散型ルーティングアルゴリズムについて考察を行う。筆者らはこれまでにネットワークのトポロジーに関する知識を必要としない、分散型ルーティングアルゴリズムを提案している。このアルゴリズムはネットワークのノード間の転送遅延時間を元に、経路の選択確率を動的に更新する。経路の選択確率はそのまま経路に対するトラフィックの分配比となる。このアルゴリズムはリンクの選択確率の初期値にかかわらず、転送遅延時間の小さい経路に対応する適切な確率を得ることができる。しかし、トラフィックの上流に向かう経路の確率が十分に小さくならないため、配送経路にループが発生してしまう。この問題を克服するため、本研究では転送遅延時間の評価結果が経路選択確率に与える影響について検討を行った。さらにこの結果を元に、ループの発生を抑制する効果の高い転送遅延時間の評価関数について考察する。

A Study of Value Function for Network Load Balancing

Hiroyuki Une* Shigeya Ikebo† Fei Qian‡

*Dept. of Information Design, Faculty of Infomatics, Hiroshima Kokusai Gakuin Univ.
†Dept. of Computer Science, Faculty of Infomatics, Hiroshima Kokusai Gakuin Univ.
‡Dept. of Electrical, Electronic and Information Engineering, Kanto Gakuin Univ.

This research discussed the distributed routing algorithm to divide up the traffic on the network dynamically among specific paths. Formerly the authors proposed the algorithm which fulfills such purpose without the knowledge of network topology. This algorithm updates dynamically the probability of path selection using the transfer delay between the source and the destination nodes. These probabilities are considered as the proportions of the distributed network traffic to each path. This algorithm is able to obtain the appropriate probability to select the path corresponding to short transfer delay even though the initial value of probability is far from the optimal. However some transfer loops are observed on the network because this algorithm can't decrease the probability of the path toward upstream sufficiently. To solve such problem, this research investigated the mechanism how the evaluation of the transfer delay effects on the probability of path selection. Furthermore, the authors considered the value function for the transfer delay which reduces the occurrences of transfer loops on the network.

1 研究背景

近年、マルチメディアデータの配信サービスの増加に伴い、QoS の確保が重要となっている。QoS 制御を行うには転送遅延時間のばらつきを小さくすることが重要な制御要素の一つとして求められている。そのため、輻輳の発生による転送遅延時間の変動を抑制することが課題となる。

輻輳の発生を抑制するための方策の一つに、宛先ノー

ドとの間に存在する複数の経路に対して、転送遅延時間に応じてトラフィックを分配することが挙げられる。著者らの目的は、一つの宛先に対して複数の配送経路を持つルーティングテーブルが得られるアルゴリズムを構築することである。

著者らは学習オートマトンにおける強化学習の手法を利用し、ネットワークトラフィックを複数経路を用いて動的に分散させるための分散型適応ルーティングアルゴリズム *DARLA*（Distributed Adaptive Routing algorithm with Learning Automata）の研究を行っている [1][2][3]。DARLA の前提として、ネットワーク上の各ノードは確率に基づいたルーティングを行うものと仮定する。各ノードが保持するルーティングテーブルには、パケットを転送する際に使用するリンクの選択確率が宛先ごとに記録されている。各ノードはルーティングテーブルに記録された確率を用いて、パケットの転送先を決定する。DARLA では、“*Ants*” と呼ばれるエージェントが収集した情報を元にルーティングテーブルを更新する。

強化学習を用いた同様のルーティングアルゴリズムとして Q-Routing が提案されている [4]。Q-Routing ではノード間の転送時間とノード内部での待ち時間から計算された Q 値を元に経路情報を作成している。

DARLA は単位時間あたりにネットワークに流れるパケット量が非常に大きくなった場合に、Q-Routing と比較して平均転送時間が短くなる。これはパケットを複数の経路に対して分散させる能力が Q-Routing よりも高いことを示している [1][2]。しかし DARLA はルーティングテーブルの安定まで時間がかかるため、配送経路上に一時的なループが多数発生する。そのため転送遅延時間が増大してしまうという問題点が残されている。

配送ループが発生する原因の一つに、配送経路の上流に向かうリンクの選択確率が十分小さくなっていないことが挙げられる。そこで本研究では、ルーティングテーブルの更新時に用いる評価関数に着目し、配送ループを形成するリンクの選択確率を抑制するための手法について考察を行う。

2 提案ルーティングアルゴリズム

2.1 アルゴリズムの概要

DARLA は次の二つの処理から構成される。

1. 経路情報、具体的にはノード間の転送時間の収集
2. ルーティングテーブルの更新

このうち 1 の処理に “*Ants Computing System*” を、2 の処理に強化学習の $L_{R\text{-}P}$ 法を利用している。

DARLA について説明するため、以下の記号を導入する。

$$
\begin{aligned}
N &= \text{（ネットワーク中の全ノード集合）} \\
N_n &= \text{（ノード } n \text{ に接続されているノード集合）} \\
N_n^{\text{known}} &= \text{（ノード } n \text{ から到達可能なノード集合）} \\
L_n &= \{(n,d) |\, d \in N_n\} \\
&\quad \text{（ノード } n \text{ に接続されているリンク集合）} \\
p_{n,d}^l &= \text{（ノード } n \text{ から } d \text{ に送信する際に} \\
&\quad \text{リンク } l \text{ を選択する確率）} \\
P_{n,d} &= \{p_{n,d}^l |\, l \in L_n\} \\
P_n &= \{P_{n,d} |\, d \in N_n^{\text{known}}\} \\
&\quad \text{（ノード } n \text{ の持つルーティングテーブル）} \\
|E| &= \text{（集合 } E \text{ の要素数）}
\end{aligned}
$$

ここでノードとは、特に断らない限りルータを意味する。各ノードは固有の整数をノード ID として付加されている。また N の各要素はノード ID とする。各ノードは以下に示す動作を行う。

- ネットワーク上の全てのノード n はパケットキューを持つ。到着したパケットは FIFO で処理される。
- ノード $n \in N$ はノード $d \in N$ にパケットを送信する際、使用するリンク $l \in L_n$ を確率 $p_{n,d}^l$ で選択する。
- n は得られた経路情報を元に、選択確率 $p_{n,d}^l$ を更新する。

2.2 Ants Computing System

“Ants Computing System”（ACS）は、アリの特徴を模倣したエージェントベースのアルゴリズム [5] で

ある。これは巡回セールスマン問題など、NP 困難・完全問題の準最適解を得るためのヒューリスティックアルゴリズムとして利用されている。

ACS で重要な役割を果たすのが “*Ants*” というエージェントである。Ants は自然界のアリを模倣して作られている。具体的には次の二つの機能を持つ。

- 自分の通った経路上に “**フェロモン**” とよばれる足跡を残す
- 自分の進路を、経路上のフェロモンの量を元にして確率的に決定する

それぞれの Ants が残していくフェロモンの量は、Ants が移動した距離などを元に計算される。具体的には移動距離が短いほどフェロモンの量が多くなる。また、それぞれの Ants は自分の通った経路を “*tabu list*” に記録する。これは移動距離の計算とフェロモンを残す場所の決定に用いられる。

したがって Ants を大量に用意して何度も巡回させると、残されたフェロモンが多い経路と少ない経路ができる。このうちフェロモンの多い経路を選択することで巡回セールスマン問題の準最適解が求められる。

2.3 Ants の役割

DARLA では、Ants を転送遅延時間の収集と各ノードがルーティングテーブルを更新するための情報提供のためのエージェントとして用いる。そのため、以下に挙げる二種類の Ants を導入する。

Forward Ant(FA) 通過したノードと通過時刻の組を記録する

Backward Ant(BA) FA が通過したノードに対して転送遅延時間に関する情報を提供する

Ants は *attr*、*source*、*destination* そして *tabu* という内部変数を持つ。*attr* には FA・BA に対応する整数が格納される。*source* と *destination* には、それぞれパケットの発信ノードと宛先ノードの ID が入る。*tabu* は tabu list を格納するためのスタックである。Ants の構造を図 1 に示す。

```
type Ant =
  record
    attr, source, destination: Integer
    tabu: Stack
  end
```

図 1: Ant の構造

2.3.1 FA の役割

各ノード n は経路情報の収集のため、周期的に FA を送出する。FA の宛先は N_n^{known} からランダムに決定する。N_n^{known} を得る方法については 2.5 節で説明する。FA を受信したノードの動作を図 2 に示す。

FA を受信したノード n は、まず通過したノード n の ID および到着時刻 T_n の組を tabu list に追加する。次に自分自身が Ant の宛先であるかどうかをチェックする。もしそうであれば、ノード n は受信した FA から BA を生成する。この BA は元の FA と同じ内容の tabu list を持つ。そして tabu list に記載されている経路を逆にたどるように BA を送信する。

ノード n は FA の tabu list にループが発生していないかどうかもチェックする。もし tabu list 中にノード n の ID が含まれていればループが発生していることになる。その場合は FA の宛先ノード d への到達時刻を **INFINITY** として tabu list に追加する。**INFINITY** は無限大を表すための十分大きな値とする。そして上の場合と同様、BA を生成・送信する。

上記のどちらの場合にも当てはまらなければ、ノード n は FA の転送先をルーティングテーブル $P_{n,d}$ にしたがって決定し、送信する。

2.3.2 BA の役割

BA は経路情報をノードに提供する役割を持っている。したがって、BA はノードのルーティングテーブルを更新するためのトリガーとなる。BA を受信したノードの動作を図 3 に示す。

BA を受信したノード n は、まず BA が保持している tabu list の内容を以下の通り二つに分ける。

```
Procedure RecvFA(a: Ant)
  T_n ← time()
  PushTab(a, (n, T_n))
  d ← GetDestAnt(a)
  if d = n then {n が宛先ノードの場合 }
    begin
      b ← GenerateBackwardAnt(a)
      {BA を一つ前のノードに送信 }
      t ← DuplicateTabu(a)
      (j, T_j) = Pop(t) {(n, T_n) を除去 }
      (j, T_j) = Pop(t)
      SendBackwardAnt(b, j)
    end
  elseif tabu list にループが存在している then
    begin
      PushTab(a, (d, INFINITY))
      b ← GenerateBackwardAnt(a)
      t ← DuplicateTabu(a)
      {BA を一つ前のノードに送信 }
      (j, T_j) = Pop(t) {(d, INFINITY) を除去 }
      (j, T_j) = Pop(t) {(n, T_n) を除去 }
      (j, T_j) = Pop(t)
      SendBackwardAnt(b, j)
    end
  else
    begin
      if d ∉ N_n^known then
        { ノード d が未知のノードであれば
          ルーティングテーブルを追加する }
        begin
          N_n^known ← N_n^known ∪ d
          foreach l ∈ L_n do
            p_{n,d}^l ← 1.0 / |L_n|
        end
      {FA をルーティングテーブルにしたがって中継 }
      j = SelectNextHop(a, P_{n,d})
      SendForwardAnt(a, j)
    end
```

図 2: FA を受信した場合の動作

- 元の FA が n より以前に通過したノードの経路情報（t）
- 元の FA が n の後にに通過したノードの経路情報（t'）

ノード n は t' に記録されている全ノードへのルーティングテーブルを、手続き “UpdateRoutingTable” で更新する。ルーティングテーブルの更新が終了したら t から次のノードを選択し、BA を送信する。

```
Procedure RecvBA(b: Ant)
  s ← GetSourceAnt(b)
  t ← DuplicateTabu(b)
  { 元の FA が到着した時刻を検索すると同時に
    経路情報をスタック 't に積みあげる }
  t' ← φ
  repeat
    (j, T_j) ← Pop(t)
    Push(t', (j, T_j))
  until j ≠ n
  T_n ← T_j
  (j, T_j) ← Pop(t')  {(n, T_n) を t' から除去 }
  { 元の FA がどのリンクを使って転送されたか調べる }
  (d, T_d) ← Pop(t')
  l ← (n, d)
  Push(t', (d, T_d))
  { ルーティングテーブルの更新 }
  repeat
    (d, T_d) ← Pop(t')
    UpdateRoutingTable(l, d, P_{n,d}, T_n, T_d)
  until t' ≠ φ
  {BA を次のノードに転送する }
  if t ≠ φ then {n ≠ s であれば t ≠ φ が成り立つ }
    begin
      (j, T_j) ← Pop(t)
      SendBackwardAnt(b, j)
    end
```

図 3: BA を受信した場合の動作

2.4 ルーティングテーブルの更新

学習オートマトンは『環境』と呼ばれる制御対象からの報酬を最大限にするように、自らの方策を決定する。ノードを学習オートマトンとして考える場合、リンク選択確率 $p_{n,d}^l$ はオートマトンの方策に対応する。つまりノード n は方策 $p_{n,d}^l$ にしたがってリンク l を出力することになる。また転送遅延時間の評価結果は環境からの報酬に対応する。したがって『学習』とは、評価結果を最大にするようにルーティングテーブル $P_{n,d}$ ($= \{p_{n,d}^l | \, l \in L_n\}$) を更新することである。

オートマトンの学習には二つの尺度が用いられる。一つはリワード（報酬）で、もう一つはペナルティ（罰）である。このうちリワードのみ用いる方法を $L_{R\text{-}I}$ 法で、リワードとペナルティの両方を用いる方法を $L_{R\text{-}P}$ 法と呼ぶ。

DARLA で用いる手続き “UpdateRoutingTable” は学習オートマトンの $L_{R\text{-}P}$ 法を用いてノード n のルー

ティングテーブルを更新する手続きである。UpdateRoutingTable は次に挙げる 5 つの引数を受け取る。

- FA の送信に使用したリンク（l）
- 宛先ノードの ID（d）
- ノード d へのルーティングテーブル（$P_{n,d}$）
- FA がノード d に到着した時刻（T_d）
- FA がこのノードに到着した時刻（T_n）

ルーティングテーブル $P_{n,d}$ の更新式は以下の通りである。

$$\begin{aligned} p^l_{n,d} &\leftarrow p^l_{n,d} - a(1-f(T_d-T_n))\cdot p^l_{n,d} \\ &\quad +\theta f(T_d-T_n)\cdot(1-p^l_{n,d}), \qquad (1) \\ p^{l'}_{n,d} &\leftarrow p^{l'}_{n,d} + a\{1-f(T_d-T_n)\}\cdot\left(\frac{1}{|L_n|-1}-p^{l'}_{n,d}\right) \\ &\quad -\theta f(T_d-T_n)\cdot p^{l'}_{n,d} \quad (l'\in L_n, l'\neq l) \qquad (2) \end{aligned}$$

式 (1) はノード n がリンク l を出力とした場合の学習式に対応する。式 (1) による他のリンク選択確率への影響は式 (2) によって吸収する。式 (1) の右辺第 2 項はペナルティに、右辺第 3 項はリワードに対応する。

関数 $f(t)$ は転送遅延時間を引数とする評価関数であり、リワードおよびペナルティの基準として用いられる。具体的には $f(t)$ はリワードの計算に、$1-f(t)$ をペナルティの計算に使用する。$f(t)$ は $[0,1]$ を値域とする単調減少関数とする。そのため、リワードを最大にする評価結果 ($f(t)=1$) はペナルティを 0 にし、リワードを 0 にする評価結果 ($f(t)=0$) はペナルティを最大にする。式 (1), (2) において、θ はリワード係数、a はペナルティ係数を表す。

学習の効率は評価関数 $f(t)$ によって異なる。評価関数の違いによる影響は 3 節にて説明する。

2.5 初期ルーティングテーブルの構築

DARLA は強化学習の手法を用いているため、ルーティングテーブルの収束まで時間を要する。そのため、アルゴリズムの開始直後でもパケットの送信を可能にするため、初期ルーティングテーブルが必要となる。

初期ルーティングテーブルを得るため、各ノードは “*notify*” あるいは “*update*” メッセージを隣接ノードに送信する [3]。これらのメッセージには各ノードが存在を知っているノード、つまり到達可能なノードの集合 N^{known}_n が記録されている。

ノード n がノード p からこれらのメッセージを受け取った場合、以下の処理を行う。

1. 集合 $S=\{x \mid x\in(\{p\}\cup N^{\text{known}}_p)\wedge x\notin N^{\text{known}}_n\}$ を求める。これはノード n から新たに到達可能となったノード ID のリストである。N^{known}_p はメッセージから抽出する。S が空であれば終了する。
2. S 中の各ノード d について、ルーティングテーブル $P_{n,d}$ を追加する。ルーティングテーブルの初期値は次の通りとする。
$$p^l_{n,d}=\begin{cases} 1.0 & (l=(n,p)) \\ 0.0 & (\text{それ以外}) \end{cases} \qquad (3)$$
3. N^{known}_n に S を追加する。
4. N_n の全ノードに対し、update メッセージを送信する。メッセージには N^{known}_n を記録する。

3 評価関数

本節では最初に、評価関数の値がルーティングテーブルに与える影響についての考察を行う。次にこの考察を元にして評価関数に求められる性質について議論する。

3.1 評価関数とルーティングテーブルの変動

まずルーティングテーブルの更新式である式 (1) に着目する。この式の右辺第 2 項および第 3 項は、FA の送信に使用したリンクの選択確率の変化である。この式において、評価関数 $f(T_d-T_n)$ の値を v に置き換えると、リンク選択確率の変動は a,θ,v をパラメータにとる、確率 p の関数である。これを

$$d_{a,\theta,v}(p)=-a(1-v)p+\theta\cdot v(1-p) \qquad (4)$$

で表す。

式 (4) より、$d_{a,\theta,v}(p)$ のグラフは二つの点 $(0,\ \theta v),(1,\ -a(1-v))$ を結ぶ線分である。図 4 に、パラメータ a,θ をそれぞれ $a=0.003,\theta=0.03$

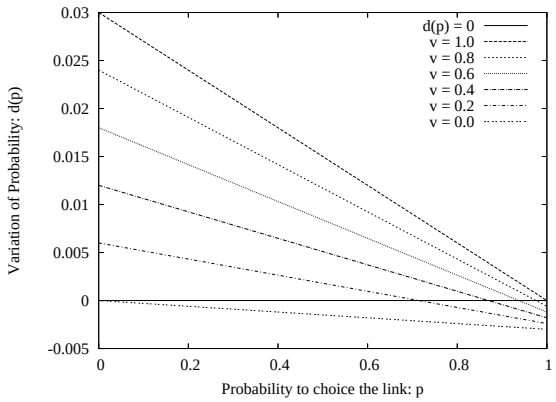

図 4: 評価関数の値 v とリンク選択確率の変動との関係

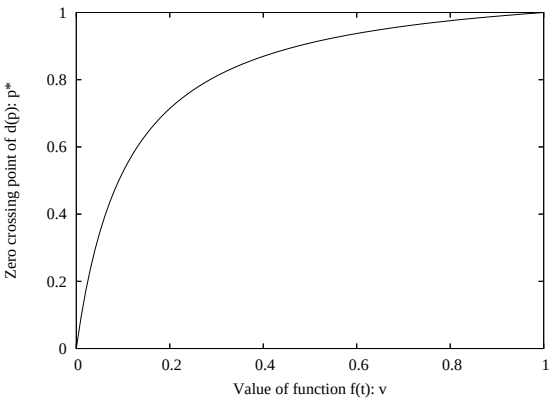

図 5: 関数 $p^*(v)$ のグラフ（$a = 0.003, \theta = 0.03$）

に固定し、評価関数の値 v を変化させた時の関数 $d_{a,\theta,v}(p)$ のグラフを示す。

図 4 より、$d_{a,\theta,v}(p)$ の最大値と最小値は以下の通りである。

最大値 $0.03(v = 1.0, p = 0.0)$

最小値 $-0.003(v = 0.0, p = 1.0)$

これはそれぞれ θ と $-a$ の値に相当する。

リンク選択確率の変動 $d_{a,\theta,v}(p)$ についてさらに掘り下げて考察する。$d_{a,\theta,v}(p) = 0$ の解を p^* とする。p^* は次の式で求められる。

$$p^* = \frac{-\theta v}{(a-\theta)v - a} \tag{5}$$

式 (5) においてパラメータ a, θ は固定であるから、p^* は v の関数と考えることができる。これを $p^*(v)$ で表す。

リンク l を使用した場合に得られる評価値を v_l とすると、図 4 からわかる通り、選択したリンク l の確率 $p^l_{n,d}$ は $p^*(v_l)$ より小さければ増加する。逆に $p^*(v_l)$ より大きければ減少する。つまり $p^l_{n,d}$ は常に $p^*(v_l)$ に向かって変動する。

3.2 評価関数に求められる性質

本節では評価関数 $f(t)$ がどの様な性質を持つべきかについて議論する。まず、図 5 に、$a = 0.003, \theta = 0.03$ とした場合の関数 $p^*(v)$ のグラフを示す。

図 5 を見ると、p^* は $0 \leq v \leq 0.5$ の範囲で急激に増加している。それに対して $0.5 \leq v \leq 1.0$ での増加は緩やかである。

この事実を基に、評価関数の定義について考える。今までの研究 [1][2][3] では以下の評価関数 $f(t)$ を用いた。

$$f(t) = 1 - \beta(t) \tag{6}$$

$$\beta(t) = \begin{cases} 1 & , \text{if } t \geq T_{\text{timeout}} \\ \dfrac{t}{T_{\text{timeout}}} & , \text{otherwise} \end{cases} \tag{7}$$

ここで T_{timeout} はネットワークの規模を基に決められた定数である。式 (1) において $T_d = \textbf{INFINITY}$、つまり FA の tabu list に配送ループを検出した場合、**INFINITY** の定義より $T_d - T_n > T_{\text{timeout}}$ である。したがって $f(t) = 0$ となり、式 (1) のペナルティ項のみが残る。これにより配送ループを発生させるリンクの選択確率が減少する。

上に挙げた評価関数には問題点がある。T_{timeout} は定数であるため、ノード間の距離によってはリンク選択確率に違いが出ないことである。例えば近距離に存在するノード d との転送時間を考える。一般にノード間の距離が短いほど転送時間は小さくなる。T_{timeout} がネットワークの規模から求められた定数であることを考えると、ノード d への転送時間はどの経路を用いても T_{timeout} の半分以下であることは十分ありうる。これは $0.5 \leq f(t) \leq 1.0$ であることを意味する。

しかし図 5 より、$0.5 \leq v \leq 1.0$ では $p^*(v)$ の差があまり見られないことがわかる。したがってノード d に

存在する各経路の転送時間の差がリンクの選択確率に反映されにくくなる。

この問題を解決するためには、評価関数 $f(t)$ に次の性質を加える必要がある。

- 評価関数の計算にノードの距離が反映されること
- 転送時間の差が大きく反映されること

本研究では上の性質を満たす評価関数として以下の関数を用いる。

$$f(t) = 1 - \beta(t) \tag{8}$$

$$\beta(t) = \begin{cases} 1 & , \text{if } t \geq T_{\text{timeout}}(T^m_{n,d}) \\ \left(\dfrac{t}{T_{\text{timeout}}(T^m_{n,d})}\right)^i & , \text{otherwise} \end{cases} \tag{9}$$

$$T_{\text{timeout}}(t) = c \cdot t \tag{10}$$

$T^m_{n,d}$ はノード n から d への平均転送時間を表す。$T_{\text{timeout}}(T^m_{n,d})$ をタイムアウト時間として用いることにより、ノード間の距離を評価関数に反映させることができる。式 (9) では、転送遅延時間 t が $T_{\text{timeout}}(T^m_{n,d})$ に近づいた時の評価関数の値を大きく変化させるために冪乗を用いた。指数 i が大きくなれば評価関数の変化も大きくなるため、i は更新パラメータの一つとなる。

式 (10) はタイムアウト時間の計算式を示している。c はタイムアウト係数を表し、$c > 1$ を満たす定数である。c の値が大きいほど、転送遅延時間の許容範囲が広くなる。

4 関連研究

強化学習の手法を用いた適応型ルーティングアルゴリズムとして Q-Routing が提案されている [4]。このアルゴリズムは Q 学習の手法をルーティングアルゴリズムに応用したものである。

Q-Routing アルゴリズムでは、ルーティングテーブルとして Q **値**（$Q_{n,d}(l)$）を使用する。これはノード n から d にリンク l を使用してパケットを送信した場合の転送時間の推定値である。ノード n が d に送信する際に使用するリンクは

$$Q_{n,d}(l) = \min_{l' \in L_n} Q_{n,d}(l') \tag{11}$$

を満たすリンク l を選択する。

ノード n が d に対して、上で得られたリンク l を使用してパケットを送信すると、l で接続されているノード n' は応答メッセージをノード n に送る。ルーティングテーブルはこの応答メッセージが到着した時に更新される。更新に用いる式を以下に示す。

$$Q^{new}_{n,d}(l) = Q^{old}_{n,d}(l) + \eta(t_{est} - Q^{old}_{n,d}(l)) \tag{12}$$

$$t_{est} = \begin{cases} q_t + \delta_l & (n' = d) \\ q_t + \delta_l + \min\limits_{l' \in L_{n'}} Q^{old}_{n',d}(l') & (n' \neq d) \end{cases} \tag{13}$$

ここで q_t は応答メッセージがノード n に到着してから処理されるまでの時間，つまりパケットキューに格納されていた時間を示す。δ_l はリンク l の伝搬遅延時間である。$\min\limits_{l' \in L_{n'}} Q^{old}_{n',d}(l')$ は応答メッセージに記録されている。η は学習係数（learning rate）といい、転送遅延時間の推定値 t_{est} を Q 値に反映させる割合を意味する。

Q-Routing ではパケットの送信のたびに応答メッセージをやりとりすることになっている [4] が、このまま実装するとネットワークのトラフィックの量が増大する。そのため、上で示した手順はルーティング用のパケットにのみ適用するものとする。

5 評価

本節では DARLA をネットワークシミュレータで評価した結果について議論する。また、Q-Routing アルゴリズムとの結果の比較を通じてアルゴリズムの有効性について検証を行う。

5.1 実験環境

本研究では、ネットワークシミュレータとして ns2[6] を採用した。シミュレーションに使用したネットワークを図 6 に示す。図 6 の各リンクは全て双方向リンクであり、5[Mbit/s] の帯域幅と 10[ms] の伝搬遅延時間を持つ。

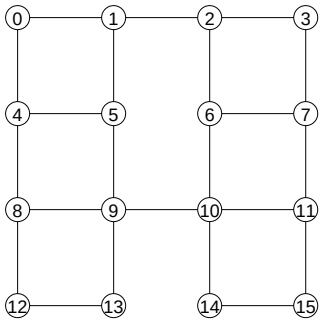

図 6: 実験に使用したネットワーク

5.2 シミュレーションパラメータ

本研究では、DARLA について式 (8) (9) (10) のパラメータ a, θ, c, i を様々に組合せながらシミュレーションを行った。具体的には $i = 1, 2, \ldots, 7$ について、表 1 の (2) から (12) に示すパラメータを組み合わせて行った。

表 1: シミュレーションパラメータの一覧

	θ	a	T_{timeout}	c
(1)	0.03	0.003	70[ms]	—
(2)	0.03	0.003	—	2
(3)	0.03	0.003	—	1.5
(4)	0.1	0.01	—	1.5
(5)	0.03	0.003	—	1.3
(6)	0.1	0.01	—	1.3
(7)	0.03	0.003	—	1.2
(8)	0.1	0.01	—	1.2
(9)	0.03	0.003	—	1.15
(10)	0.1	0.01	—	1.15
(11)	0.03	0.003	—	1.1
(12)	0.1	0.01	—	1.1

表 1 で示したパラメータのうち、(1) はこれまでの研究結果と比較するために式 (6)(7) を使ってルーティングテーブルを更新する。

各シミュレーションとも、トラフィックジェネレータを図 6 のノード 0 とノード 12 に配置した。二つのジェネレータはノード 15 に対し、210 バイトのデータパケットを 2[ms] 間隔で送信する。また、各ノードがルーティングテーブルを更新するために送信する FA の送出間隔を 50[ms] とした。シミュレーション時間は 128[s] とし、ジェネレータはシミュレーション開始から 4[s] 後に起動する。

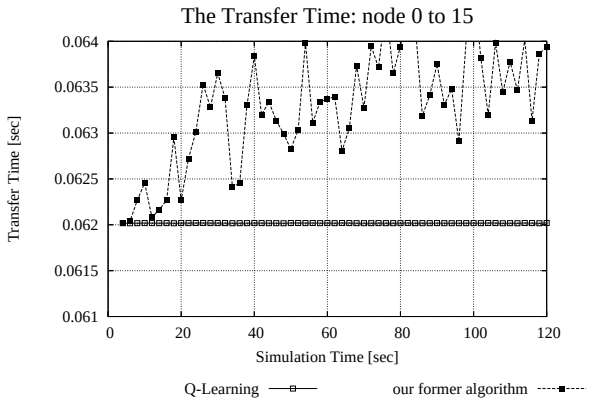

図 7: 従来のアルゴリズムと Q-Routing アルゴリズムを用いた場合の転送時間

Q-Routing アルゴリズムのパラメータ Q-Routing を用いたシミュレーションにおいて、使用したパラメータは以下の通りである。

- 学習係数 $\eta = 0.7$
- ルーティングメッセージを交換する間隔 100[ms]

5.3 転送時間

最初にノード 0 からノード 15 に送信されたパケットの転送時間に対してパラメータ i が与える影響について比較と検討を行う。転送時間の比較は、各シミュレーションにおいてパケットの転送時間を 2[s] ごとに平均を求めてから行った。

シミュレーション結果を図 7, 8, 9 に示す。図 7 は Q-Routing および式 (6)(7) を用いた場合の結果、図 8,9 は表 1 のうち、今回のシミュレーションで最も転送時間が安定していたパラメータ (5),(7) をそれぞれ適用した場合の結果である。

図 7 からわかる通り、Q-Routing アルゴリズムではノード 0 からノード 15 に向かう最短経路を通った場合の転送時間に収束する。それに対して、式 (6)(7) を用いた DARLA の結果はそれより大きい収束時間の周辺で振動している。

図 8, 9 は本研究で提案した評価関数（式 (8)(9)(10)）を用いた場合のグラフである。グラフを見ると、式 (9) のパラメータ i が大きくなるにしたがって Q-Routing を用いた場合の転送時間に収束していくのがわかる。

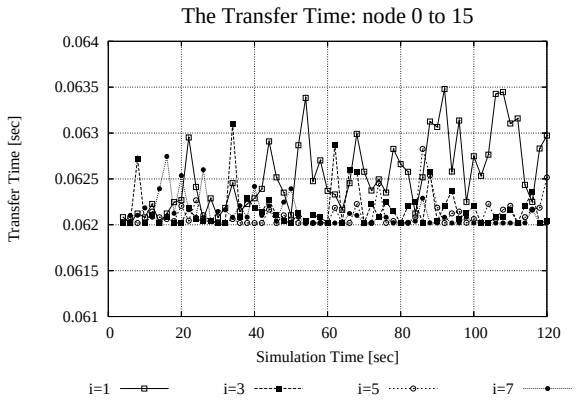

図 8: パラメータ (5) を用いた場合の転送時間

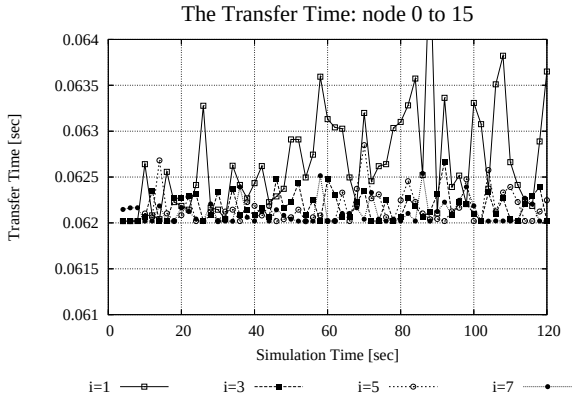

図 9: パラメータ (7) を用いた場合の転送時間

5.4 配送ループの発生頻度

次に配送ループの発生頻度についての検討を行う。表 2, 3 に、ノード 0 からノード 15 にパケットを送信する際に発生する配送ループの頻度をパラメータの組合せごとにまとめた。なお、この表に書かれている番号は表 1 の番号に対応する。

表 2 の番号 (1) は従来の評価関数を用いた場合のループ発生頻度である。トラフィックジェネレータはのパケット生成間隔は 2[ms] であるから、1 回のシミュレーションでノード 0 がノード 15 に向けて発信するパケットは 62000 個となる。従来手法では 4858 回の配送ループが発生している。これは生成されたパケットの 7.8%に当たる数値であり、遅延時間に対する影響はとても大きい。

ここで本研究で提案した評価関数を用いた場合の結果に注目する。パラメータ (2)(3)(4) についてはループ発生を抑制するどころか、発生頻度が大きくなっている。それに対して他のパラメータについては、i の値が大きくなるにつれてループの発生頻度が全体的に抑えられていることがわかる。特にパラメータ (5) と $i = 6$ の組合せでは、ループの発生が 216 回となっている。これは生成されたパケット数の 0.35%であるから、提案した評価関数によりループの発生を十分に抑制できることがわかる。

表 2: パラメータ別配送ループ発生頻度

番号	パラメータ i			
	1	2	3	4
(1)	4858			
(2)	8283	11418	12879	13584
(3)	4244	4674	4192	4978
(4)	8322	6496	5543	5015
(5)	2142	974	601	350
(6)	7756	3727	2160	1670
(7)	2825	1269	524	481
(8)	11609	5574	3371	2238
(9)	2976	1585	1036	847
(10)	14627	7252	4177	3159
(11)	3494	2381	1421	1213
(12)	19145	9924	6293	4395

表 3: パラメータ別配送ループ発生頻度（続き）

番号	パラメータ i		
	5	6	7
(2)	16268	15505	18699
(3)	6059	5645	7025
(4)	6389	7865	8480
(5)	282	216	404
(6)	1265	778	572
(7)	493	327	382
(8)	1357	1147	1084
(9)	566	422	308
(10)	1886	2028	1486
(11)	631	830	722
(12)	4094	2490	2057

5.5 トラフィックの分散

最後に、ネットワークのトラフィックがネットワーク上でどの様に分散しているかについて注目する。表 4 に、ノード 0 からノード 15 へ送信されるパケットが

各ノードを通過した回数をまとめた。ここで提案手法に用いたパラメータは、表 1 のパラメータ (5)（$a = 0.003, \theta = 0.03, c = 1.3$）および $i = 6$ を用いた。

表 4: ノード別パケット通過回数

ノード	提案手法	Q-Routing
0	25	0
1	34526	46026
2	20688	38373
3	5383	13912
4	27557	15969
5	30837	15572
6	15331	24456
7	10151	25157
8	10564	8045
9	41370	23617
10	51887	36823
11	40272	39139
12	17	0
13	22	0
14	21703	22836

DARLA の結果には配送ループが発生しているために単純に数字を比較するわけにはいかないが、Q-Routing と同様、パケットはネットワークの広い範囲を通過している。特にノード 0 に隣接するノード 1, 4 に着目すると、通過するパケットの差は Q-Routing よりも小さい。これよりデータパケットが複数の経路に分散していることがわかる。

6 まとめ

本研究では、分散型ルーティングアルゴリズム DARLA において転送遅延時間の評価関数 $f(t)$ の与える影響について考察した。この結果、リンクの選択確率の収束値は転送遅延時間の評価結果 v より求まる確率 $p^*(v)$ に依存すること、および $p^*(v)$ の変化が $0.0 \leq v \leq 0.5$ と $0.5 \leq v \leq 1.0$ とで大きく異なることがわかった。さらにこの考察結果を元に評価関数 $f(t)$ を組み変えることで、ループの抑制とトラフィックの分散を実現することができた。

現時点で DARLA にはいくつかの未検証点と問題がある。まず学習オートマトンの手法を用いているため、頻繁に FA・BA を送信する必要がある。ネットワークのトラフィックが急激に変化しない限り、$P_{n,d}$ が収束すれば FA の送信頻度を小さくすることができる。FA の送信によるネットワークの負荷を小さくするためには、ネットワークトラフィックの変化を学習開始のトリガとして用いる仕組みが必要となる。また、異なるネットワークトポロジを用いた場合の結果、および TCP のウィンドウ制御など上位層に対する影響について検証する必要がある。

今後の方針として QoS 制御の実現を目指していく。現在のところ DARLA は転送遅延時間を唯一のメトリックとして使用する。そのため帯域制御などを含む QoS 制御に対応できない。したがって QoS 制御を複数の制約を持つ最適化問題としてとらえ直し、評価関数を設計する必要がある。

参考文献

[1] 鵜根, 横山, 池坊, 田中, 銭, “強化学習を利用した分散型ネットワークルーティングアルゴリズム”, **電気学会論文誌** *C*, 122–C(6):922–927, 2002.

[2] Hiroyuki Une, Fei Qian, “Network load balancing algorithm using ants computing”, In *Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology*, pages 428–431, 2003.

[3] Hiroyuki Une, Shigeya Ikebo, Fei Qian, “A traffic shaping algorithm on network using learning automata”, In *Proceedings of the 36th ISCIE International Symposium on Stochastic Systems Theory and Its Applications*, pages 84–89, 2005.

[4] Michael L. Littman, Justin A. Boyan, “A distributed reinforcement learning scheme for network routing”, Technical Report CMU-CS-93-165, Carnegie Mellon, 1993.

[5] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, “The Ant System: Optimization by a colony of cooperating agents”, *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[6] “The network simulator — ns-2”, VINT project, http://www.isi.edu/nsnam/ns/.