広島工業大学紀要研究編
第43巻（2009）pp.305-310
論　　文

アドホックネットワークにおけるノード負荷を考慮した高効率ルーティング手法に関する研究

長坂　康史*・金只　圭司**

（平成20年10月31日受理）

A Study on High Effective Routing Method Using Node Loads in Ad-Hoc Networks

Yasushi NAGASAKA and Keiji KANETADA

（Received Oct. 31, 2008）

Abstract

An ad-hoc network technology is focused on realizing the ubiquitous network society. The network, however, has several problems because it is constructed with several independent wireless nodes which might be moved. A lot of routing protocols on such an unstable ad-hoc network have been studied. But most of protocols don't take account of loads of their nodes in spite of its importance. We have proposed the more effective routing algorithm for the ad-hoc network using the node loads. The proposed method uses a CPU usage, a memory usage, and so on, as a definition of the node load, and also has the features of storing a multiple routes for a dynamic routing, transferring the node load information, and selecting the most effective route from the information. The simulation results show that the performance of the proposed method is better than the existing one in case that there are 50 % of high load nodes in the network.

Key words: routing algorithm, ad-hoc networks, AODV

1. はじめに

ユビキタス化が進む昨今では，携帯電話やPDAなどの移動端末の発達が著しい。移動端末におけるネットワークとしては，ハードウェアの小型化やその移動性により配線不要な無線通信技術が用いられる。そのため，携帯端末にはIEEE802.11xやBluetoothなどが標準搭載されているものが多い。これらの無線通信技術も90年代後半から，通信速度が飛躍的に向上している。一般的な無線ネットワークは，基地局を介して構築されるが，基地局なしで自律的なネットワークを形成することも可能である。特に移動端末では後者の利用が注目されている。そのようなネットワークは，アドホックネットワークと呼ばれる。アドホックネットワークは，災害時や車車間通信などの分野での利用が期待されている。

しかし，アドホックネットワークには多くの課題が残されている。アドホックネットワークには，通信ノードの移動によるネットワークトポロジの動的変化や，バッテリやメモリのような移動端末のリソースの制約などの問題がある。また，アドホックネットワークではマルチホップ通信を行うことで，通信範囲外のノードと通信することができるが，多くの2端末間通信で，同じノードを経路とするよ

* 広島工業大学情報学部情報工学科
** ネットワンシステムズ株式会社

うな場合，そのノードに集中的に負荷がかかる。そのため，通信に遅延やパケットロスが生じる可能性が高い。このような問題を解決するためには，ノードの負荷を考慮したルーティング手法が有効であると考えられる。

本研究では，既存のルーティング手法である AODV（Ad hoc On-demand Distance Vector routing）[1] にノード負荷を考慮するように改良を加えた手法を提案する。提案手法は，それぞれの端末が複数経路を保持し，各経路の負荷を定期的に測定する。そして，その負荷情報から，負荷が最小である経路を最適経路とする。このような負荷の高い端末を回避して構築された経路は，遅延やパケットロスを抑え，効率的な通信を行うことができると考えられる。

2. AODV（Ad hoc On-demand Distance Vector）

AODV では，送信ノードの Route Request（RREQ）と宛先ノードの Route Reply（RREP）によって，経路を作成する。また，各経路はシーケンス番号で管理され，RREQ や RREP などの経路制御メッセージにシーケンス番号を付加し，経路制御メッセージ受信時に，経路が最新であるかを調べるために用いられる。

送信元ノードはデータを送信したい相手，すなわち宛先ノードへの経路を知るため，RREQ を通信範囲内の全ノードにブロードキャストする。図 1 に示すように，RREQ は自身の通信範囲内に存在するノードを中継して，送信元ノードから宛先ノードに配送される。RREQ を受信した中継ノードは，双方向通信のために，RREQ を送信してきたノードを送信元ノードへの経路として保持する。この送信元ノードへの経路のことを逆経路という。各ノードが保持する経路は，宛先への中継を行うノードすべての情報を保持するわけではない。宛先へ届けるために送るべき 1 ホップ先のノード，つまりネクストホップのみを保持する。中継ノードは宛先ノードに達するまで，RREQ のブロードキャストを続ける。宛先ノードが，RREQ を受信すると RREP を送信ノードまでユニキャストする。RREP を受信した中継ノードは，RREP を送信してきたノードを宛先ノードへのネクストホップとして保持し，逆経路のネクストホップへユニキャストする。これによって送信元ノードから宛先ノードまでの経路が作成される。

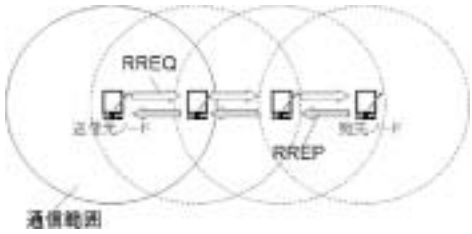

図 1　AODV の経路確立

経路確立後，経路として利用されている各ノードは，Hello を送信する。Hello はノードの存在を通信範囲内（TTL=1）にブロードキャストするものである。使用中の経路のネクストホップからの Hello を受信することで，その経路の有効期限を延長する。Hello が指定した時間内に受信できなければ，リンク切断と判断し，経路を終了する。

3. 提案手法

本研究では，複数の経路から最も中継ノードの負荷が少ない経路を最適経路として選択することで，高効率な通信を行う手法を提案する[2]。また，経路確立後，Hello で近隣ノードと経路情報を交換することで，ネットワークトポロジの動的変化にも対応する。

ノード負荷を考慮した最適経路選択は，前章で述べた AODV の RREQ，RREP そして Hello の 3 つの経路制御メッセージを利用する。各経路制御メッセージパケットに，負荷情報である“コスト”を加えることで，経路作成中と経路作成後にコストを考慮した経路選択が可能となる。各制御メッセージの詳細な動作は，3.2 節，3.3 節，3.4 節で述べる。

図 2 に提案手法による経路確立例を示す。これは，ノード A がノード F への経路を確立した状態である。提案手法では，ネクストホップをリスト化し，複数の経路を保持するため，ノード B は，ノード F への経路のネクストホップとして，ノード C とノード D を保持している。ノード B は，この 2 つのネクストホップのうち，コストが低い方の経路に配送する。また，ノード B は複数経路を保持していることで，どちらかの経路でリンク切断が生じたとき，他方の経路が有効であれば，再度 RREQ を送信することなく，経路を切り替えることができる。従って，提案手法では，経路再探索の時間を短縮することが可能となる。

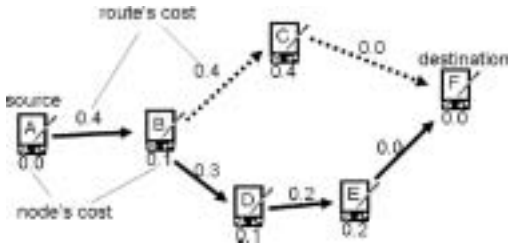

図2　提案手法による経路確立例

コストには，Ncost（ノードコスト）と Rcost（経路コスト）がある。図 2 の各ノードの下に示す数値が Ncost，矢印の上の示しているのが Rcost である。ノードコストは，ノード自体のコストである。Rcost はネクストホップから宛先ノードまでの中継ノードの Ncost の合計である。提案手法では，Rcost が最小となる経路を最適経路とする。例えば，図 2 において，ノード B は Rcost 0.4 の経路と Rcost 0.3 の経路を保持しているため，ノード A から受信したデータは，ノード D に配送する。Ncost と Rcost の算出方法については 3.1 節で述べる。

3.1　コストの算出

提案手法で利用する Ncost と Rcost について述べる。Ncost には，CPU 使用率とメモリ使用率を用いる。Ncost の算出は，次の(1)式で行う。

$$N\cos t = wCPU + (1 - w)\,Mem \qquad \cdots\cdots (1)$$

(1)式において，Ncost はノードコスト，w は CPU 使用率の重み，*CPU* は CPU 使用率，*Mem* はメモリ使用率である。次に，Rcost を定式化すると（2）式のようになる。

$$R\cos t_n = N\cos t_1 + N\cos t_2 + \cdots + N\cos t_{n-1} \qquad \cdots\cdots (2)$$

(2)式において，*n* は宛先ノードから中継ノード数，Rcost は経路コストを表す。Rcost は中継ノードの Ncost の合計である。なお，本研究では，各ノードの性能はすべて同等と仮定した。

3.2　RREQ フェイズ

本節では，RREQ の動作について述べる。RREQ 処理は AODV と同様，通信要求を持つ送信元ノードから開始する。提案手法では，RREQ のフォーマットに Rcost を付け加えている。

送信元ノードは，RREQ を通信範囲内に存在するノードすべてにブロードキャストで送信する。このとき，RREQ に自身のコストである Ncost を RREQ 内の Rcost としてセットする。送信元ノードはこの後，RREP が返って来るまで一定時間待つ。もし，一定時間内に返ってこなければ，再度 RREQ を送信する。このとき，RREQ のシーケンス番号は 1 増加させる。RREQ を受信したノードは，逆経路を参照し，以下の 3 つの条件に沿って経路の更新を行う。

(a) 逆経路を保持していない場合

この場合，ノードは逆経路を作成する。逆経路へのネクストホップリストを作成し，RREQ を送信してきたノードのアドレス，ポート番号，Rcost をリストに挿入する。

(b) 逆経路のシーケンス番号＜ RREQ のシーケンス番号

この場合，現在保持している逆経路は最新の経路ではないので逆経路を更新する必要がる。既に保持している逆経路のネクストホップリストを破棄する。その後，シーケンス番号を 1 増加し，ネクストホップリストを作成し，RREQ を送信してきたノードのアドレス，ポート番号，Rcost をリストに挿入する。

(c) 逆経路のシーケンス番号＝ RREQ のシーケンス番号

AODV では破棄されるが，提案手法では複数経路を保持する必要があるため，シーケンス番号が等しい場合も RREQ を破棄しない。逆経路のネクストホップリストの最後尾に RREQ を送信してきたノードのアドレス，ポート番号，Rcost をリストに挿入する。その後，ネクストホップリストをコストの少ない順にソートする。提案手法では，常にリストの先頭にあるネクストホップがデータ通信に利用される。

次に RREQ の配送処理について述べる。RREQ 受信ノードが RREQ の宛先ノードではなく，(a)，(b)の場合は，処理が終わると RREQ の Rcost に自身の Ncost の値を加え，ホップ数を 1 増加させ，TTL を 1 減少し，再度ブロードキャストする。(c)の場合は，同じ RREQ ID の RREQ をすでに配送したことがあるので，無駄な拡散を防ぐため，再度 RREQ をブロードキャストすることはない。図 3 のように，これらの処理が宛先ノードまで宛先ノードまで繰り返される。宛先ノードに到達すると RREP 処理に移行する。

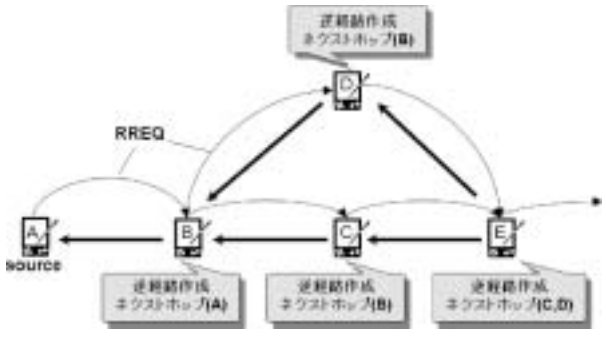

図3　提案手法の RREQ フェイズ

3.3 RREP フェイズ

RREQ を受信したノードが RREP を返す条件は AODV とほぼ同じである。しかし，提案手法では複数経路の保持を許可するため，RREP を複数回返す場合がある。RREQ を受信したノードが，RREP を返す条件は次の 2 通りである。

(a) ノードが RREQ の宛先ノードである場合
(b) ノードが宛先ノードへの経路を保持している場合

どちらの場合も 3.2 節の処理によって逆経路を作成する。次に，RREP の生成し，逆経路に送信する。このとき，(a)の場合は RREP のコスト値に宛先ノードの Ncost を付加する。(b)の場合は，宛先への経路の中で最小の Rcost を RREP に付加する。既に送信元ノードへの RREP を返した後に，別の経路から RREQ を受信した場合には，保持している経路のシーケンス番号は増加せず，RREQ 送信ノードに RREP を返す。このとき，宛先ノードは RREQ 送信ノードを逆経路のネクストホップリストに追加する。

次に RREP の配送処理について述べる。RREP を受信したノードは，RREQ 処理と同様に，宛先への経路を作成する。既に宛先への経路を保持している場合は，以下の 3 つの場合に経路の更新処理を行う。

(a) 経路のシーケンス番号が無効
(b) 経路のシーケンス番号＜ RREP のシーケンス番号
(c) 経路のシーケンス番号＝ RREP のシーケンス番号

(a)，(b)の場合は経路のネクストホップリストを一度削除し，新たに前ホップをネクストホップリストに追加する。(b)の場合は経路エントリを作成することなく，ネクストホップリストの最後尾に RREQ を追加する。RREP の配送は逆経路のネクストホップすべてに行う。RREP 配送時には RREP に自身の Ncost を RREP のコスト値に加えて，配送する。このようにして，RREP が送信元ノードまで配送されれば，送信元ノードと宛先ノード間で双方向通信が可能となる。

3.4 Hello

Hello は，TTL が 1 の RREP をブロードキャストするものである。AODV では自身のノードが通信範囲内に存在していることを，HELLO_INTERVAL[ms]ごとに通知する。提案手法では AODV とは異なり，経路情報の交換に利用する。AODV における HELLO_INTERVAL のデフォルトは，1000 ms となっている。

Hello は，ノードが保持しているすべての有効な経路について行う。Hello によって通知される内容は，ノードが保持している宛先ノードアドレス，その宛先ノードへのネクストホップと Rcost である。このとき，ノードの宛先ノードへのネクストホップには，ネクストホップリストの先頭に存在するノードを指定する。これらを含んだ Hello を，HELLO_INTERVAL[ms]間隔でブロードキャストする。Hello メッセージを受信したノードは次のチェックを行う。

(a) Hello 内の宛先ノードが自身である
(b) Hello 内の宛先ノードへの経路を保持している

(a)の場合は，Hello を破棄する。(b)の場合，経路のネクストホップリストに Hello 送信ノードが既に存在していれば，Rcost の情報と有効期限を更新し，存在していなければ新たに追加する。このとき，Hello のネクストホップが Hello 受信ノードであれば，経路ループが発生するため，その場合は更新を行わない。最後に，Hello 送信ノードをネクストホップとして保持していれば，その経路の有効期限を延長する。

この Hello によって最新の経路のコストが通知され，動的に最適経路を切り替えることが可能となる。

4. シミュレーション

本研究では，提案手法の有効性についてシミュレーションを行い検証した。提案手法の比較対象は，提案手法の基となっている AODV とした。シミュレーションツールには OPNET Modeler を使用した。本章では複数のネットワーク環境においてシミュレーションを行い，その結果と考察を述べる。

実際のノードのコストは 3.1 節の (1)式から算出する必要があるが，シミュレーション上では単純にコスト＝遅延時間とした。例えば，コストが 0.1 の場合，そのノードの遅延時間は 0.1 s となる。

ノードは図 4 のように 10 台配置した。通信は各ノードが 1 s 間に 100 kb のデータをランダムに宛先ノードを選択し，データを送信する。各ノードは図の四角で囲まれた範囲をランダムに移動する。シミュレーションは 100 s 間行った。シミュレーションでは，ネットワーク内に n 台のノードが高負荷状態（遅延が大きい状態）であるときの，到達データ量を AODV と提案手法で比較する。高負荷状態のノードは，コスト 0.1（遅延 0.1 s）の場合と 0.3（遅延 0.3 s）の場合の 2 通りでシミュレーションを行った。

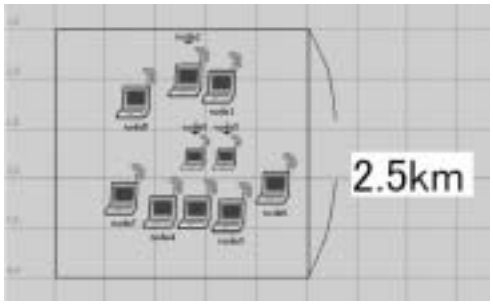

図4　シミュレーションのノード配置

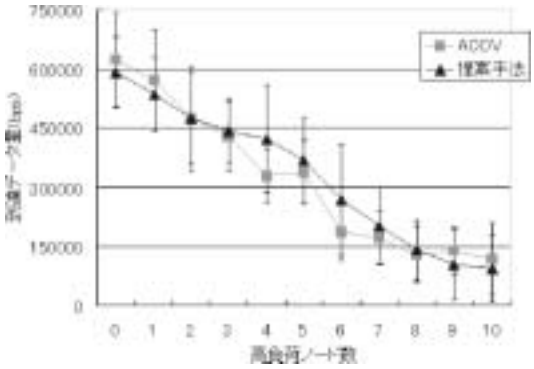

図6　シミュレーション結果（cost = 0.3）

4.1　結果

高負荷ノードのコストが 0.1 のときのシミュレーションの結果を図 5 に示す。グラフは，横軸に高負荷ノード数をとり，縦軸に宛先ノードへの到達データ量［bps］をとる。高負荷ノード数が 0 とは，ネットワーク内に高負荷（コスト 0.1）ノードが存在しない場合である。これを 1 から 10 台まで増加させていった。ノードの移動によるリンク切断やコストによる通信遅延を考慮しなければ，10 台のノードが 1 s 間に 100kb のデータを送信するため，1 Mbps のデータが全宛先ノードに到達する。高負荷ノードは 10 台のうち，node1 から順に選択した。例えば，高負荷ノード数が 3 台のときは，node1，node2，node3 がコストを持つ。

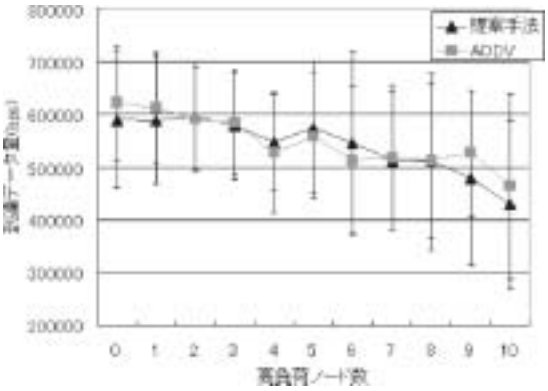

図5　シミュレーション結果（cost = 0.1）

次に高負荷ノードのコストが 0.3 のときのシミュレーション結果を図 6 に示す。先述の場合との違いは高負荷ノードの遅延時間が 0.3 s であるということである。

4.2　考察

図 5 から，コストが 0.1 で高負荷ノードが 4 ～ 6 台の場合において，AODV より提案手法の到達データ量が多い。しかし，誤差を考えるとばらつきが大きくあまり安定しているとはいえない。従って，コスト 0.1 の場合では提案手法と AODV ではほとんど差はないといえる。

図 6 から，コストが 0.3 で高負荷ノードが 3 ～ 8 台の場合において，AODV より提案手法の到達データ量が多い。特に高負荷ノードが 4 ～ 6 台存在する場合には，ばらつきは大きいものの，AODV より有効である可能性が高い。

以上の結果から，AODV において高負荷ノードを経路として選択する確率が高く，提案手法において高負荷ノードを経路とすることを避けることが可能な確率が高い場合に，提案手法が AODV に比べ有効である可能性が高いと考えられる。

5. まとめ

本研究では，アドホックネットワークにおいて各ノードの負荷を考慮することで既存のプロトコルである AODV に改良を加え，新たな手法を提案した。提案手法では，ノードをコストと称し，コストを各経路制御メッセージ内に組み込んだ。これにより，通信ノードはその時点で最もコストの少ない経路選択し効率的にデータを送信することができる。

提案手法をシミュレーションにより評価した。シミュレーションでは，ノードが移動する環境において AODV との比較を行った。その結果，ネットワーク内に高負荷ノードが 10 台中 4 ～ 6 台存在する場合において，提案手法は AODV より有効である可能性が高いことを示した。

提案手法はより多角的な視点からシミュレーション評価されるべきである。また，実際に CPU 使用率とメモリ使用率からコストを算出し，そのコストが実際にどれほどの遅延時間を発生させるのかを知る必要がある。今後これらのことを踏まえ，提案手法の改善を行う必要があると考え

られる。

参 考 文 献

[1] C.Perkins “Ad hoc On-Demand Distance Vector (AODV) Routing” draft-ietf-aodv-13. txt, 2003

[2] 金只圭司，長坂康史“アドホックネットワークにおけるノード負荷を考慮した高効率ルーティング手法の提案”，FIT2007 第 6 回情報科学技術フォーラム，2007