平成 15 年度　学士論文

アドホックネットワークにおける
ディスジョイントな
マルチパスルーティングプロトコル

研究指導　　甲藤 二郎　助教授

早稲田大学 理工学部

電子・情報通信学科

G00G078-3

谷山　健太

指導教授印	受付印

平成 16 年 2 月 9 日

目次

第1章 序論

第2章 研究背景

第3章 提案手法

第4章 シミュレーション

第5章 まとめ

参考文献

謝辞

第１章　序論

１．１　モバイルコンピューティングとユビキタス化社会

近年、携帯電話や無線 LAN による通信などで、移動中、あるいは外出先でコンピュータを利用する姿が増えてきた。ノートパソコンや携帯端末の高性能化・高機能化や、携帯電話や PHS によるデータ通信の高速化、無線 LAN アクセスポイントの増加に伴い、モバイルコンピューティングが盛んになってきたのである。こういった携帯端末、コンピュータネットワークの発展により、いつでもどこでも 情報を発信、受信、検索したりすることができる、ユビキタス化社会に近づいていく。

１．２　アドホックネットワーク

従来の端末の通信の方式では､いったん基地局や無線LANのアクセスポイントを介して、他の端末のとの通信をする。しかし、そのような通信では、イベント会場のような人が多く集まる場所や緊急災害時などでインフラが利用できない、また基地局が近くになく通信ができない、といった一時性が要求される時に、役に立たなくなってしまう。このとき、従来の基地局とそれを結ぶ通信網によって構成されるような通信でなく、ノートパソコンや PDA、携帯電話といった個人の所有するノードが、お互いに自由にワイヤレスで直接的に、同時に複数のノードと通信できるための技術が、アドホックネットワーク(Mobile Ad Hoc Network : MANET)である。

アドホックネットワークでは、一般のコンピュータで無線 LAN 接続に使われる IEEE802.11x といったプロトコルを使い、通信したいノード同士で直接に電波が届かなくても、多数の端末を経由しながら相互に接続、通信するマルチホップ型の通信形態をとる。このため、局所的に、基地局やアクセスポイントが不要な、安価なネットワークを構築することができる。

アドホックネットワークはその特徴として、各ノードが無線通信なので自由に動き回ることができ、ネットワークトポロジは絶えずランダムに変わっていくという点や、有線通信に比べて余裕のない通信帯域である点、ノードはバッテリーによって駆動していることが多いことからリソースを考慮しなければならない点などがあることから、様々な課題について考えなければならない。

１．３　ルーティングプロトコル

アドホックネットワークでは各ノードがデータ通信しつつ頻繁に動き回ることから、通信経路を探索するためのアドホックネットワーク専用のルーティングプロトコルが必要とされる。

ルーティングプロトコルには大別して 2 通りの特徴があり、Reactive 型と Proactive 型(On-demand 型)とある。Reactive 型は送信元ノードから受信先ノードに対して、通信要求があるときだけ動作し、受信先までの経路表を更新する。Proactive 型は Reactive 型とは違って常に動作し、あらかじめ経路表を作成しておく。Proactive 型の方が、通信をすぐにできるという点で優れているが、帯域や電力のような資源が限られた無線環境下においては、Reactive 型の方がよく用いられる。

IETF の MANET Working Group によって提案されている各種ルーティングプロトコルの代表的なものとしては、DSR(Dynamic Source Routing Protocol)、TBRPF(Topology Dissemination Based on Reverse-Path Forwarding)、Internet-Draft から RFC 化された AODV(Ad Hoc On-Demand Distance Vector)、OLSR(Optimized Link State Routing Protocol)といったプロトコルが挙げられる。

また、昨今のデータの大容量化やコンテンツ配信に伴い、パケットの到着率や伝送効率を高めるために、ノード間の通信経路を複数作成し保持するマルチパス方式をとるルーティングプロトコルもさまざまな箇所で提案されてきている。

１．４　研究目的

アドホックネットワークの研究の進展に伴い、さまざまなマルチパスルーティングプロトコルが研究、提案されてきた。その中でもディスジョイントなパスを作るというのがテーマになっているものが多く見られる。ディスジョイントなパスというのは複数の経路を保持していた場合に、その経路同士が重複していない経路のことをさす。ディスジョイントなパスを利用することにより、経路が無効になる際に複数の経路が一度に無効になるのを軽減する効果や、使い方によっては負荷を分散させるといったことも考えられる。

本研究ではディスジョイントパスの作成法について提案して、ルーティングプロトコルである DSR を拡張し、また、DSR における Packet salvaging を改良して、その性能の向上させることを目的とする。

１．５　本論文の構成

第１章である本章では、研究背景・研究目的を述べる。
第２章では、アドホックネットワークにおけるルーティングプロトコルの動作、及びその問題点、また従来研究とその問題点について述べる。
第３章では提案手法の目的、およびその動作方法を述べる。
第４章ではシミュレーションにより提案手法を評価する。
第５章では本論分の総括を行なう。

第２章　研究背景

２．１　The Dynamic Source Routing (DSR)

２．１．１ 概要

The Dynamic Source Routing protocol(DSR)[1]はマルチホップワイヤレスアドホックネットワークにおいて機能するように作られた、簡明かつ効果的なルーティングプロトコルである。DSR ではネットワークのインフラや管理をすることなしに、全て自動的にネットワークを構築･設定することができる。DSR では”Route Discovery”と”Route Maintenance”の二つの主機能を持ち、アドホックネットワークにおける任意の宛先へのルートを探索したり維持したりする。プロトコルは全て On-Demand に動作し、現在使っているルートを変える必要がある場合にのみ働くので、DSR のルーティングパケットによる負荷を抑えることができる。また、宛先への多数のルートを持っており、送信元はそのルートを選択・制御することができる。

２．１．２　Route Discovery

送信元 S が宛先 D への通信を始めたいときに､送信元 S はデータパケットのヘッダーに、宛先への転送経路を示すホップ列である「ソースルート」情報をのせる。そして、送信元 S は宛先 D へのルートを自分の「ルートキャッシュ」内に持っているかどうか判別する。もし宛先 D へのルートをもっているならば、そのルートを使ってデータパケットを送る。もしなければ、Route Discovery によって直ちに新しいルートの探索がなされる。
まず、ルートの探索のために、送信元 S は Route Request パケット(以下、RREQ)をブロードキャストし、S の通信範囲内にあるノードは全てこのメッセージを受け取る。この RREQ は宛先 D のアドレス、送信元 S のアドレス、ID ナンバーを含んでいる。また、RREQ は、自分が転送されてきた中間ノードのアドレスリストも含んでいる。このアドレスリストは Route Discovery 開始時には空である。

この RREQ が中間ノード(例えば図２．１．１のノード b)に到達した場合､そのノードは、RREQ の ID と宛先のアドレスを自身の「リクエストテーブル」に記録し、また、自分が宛先 D へのルートを持っていないかルートキャッシュをチェックする。もし持っていなけれ

ばそのノードは自分のアドレスを RREQ に記録し、RREQ を転送(再ブロードキャスト)する。このとき、RREQ の転送数を制限するためとルートのループ回避のため、各ノードは、

- リクエストテーブルと対比し、RREQ 内の ID と宛先アドレスが同一の RREQ を以前に受け取っていない　or
- RREQ 内に自分のアドレスが含まれていない

場合にのみ、RREQ を転送する。図２．１．１の場合、ノード c において S-b-と RREQ が転送されてきたとする。その後、S-a-という RREQ を受信した場合、ノード c では、ID と宛先アドレスが同一の RREQ を受信したということで、S-a-の RREQ を廃棄する。

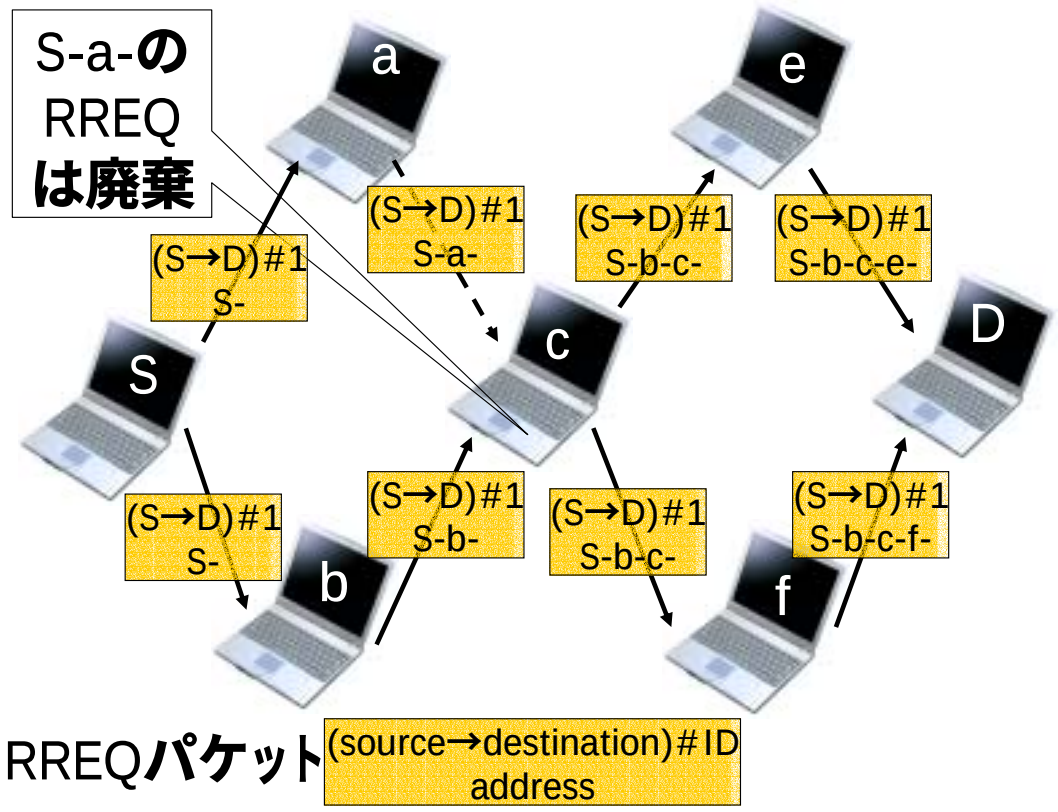


図２．１．１　RREQ

RREQ の到着時に、そのノードが宛先 D への経路をルートキャッシュ内に保持していた場合、あるいはそのノード自身が宛先 D であった場合に、Route Reply パケット(以下、RREP)が返される。このノードが宛先 D であった場合、RREQ 内にあるアドレスリストを RREP にコピーして、送信元 S まで転送する。このノードが中間ノードであった場合には、RREP には RREQ 内にあるアドレスリストに、自身のルートキャッシュ内にあるルートのアドレスリストを加えて RREP を生成し、送信元 S まで転送する。図２．１．２．では宛先 D が 2 つの RREQ を受信したため、2 経路に対して、RREP が返されている。宛先ノー

ドでは RREQ を複数受信した場合、複数の RREP を返信する。

また、RREP を転送するノードは各ノードの持つルートキャッシュに RREP のルートをキャッシュする。

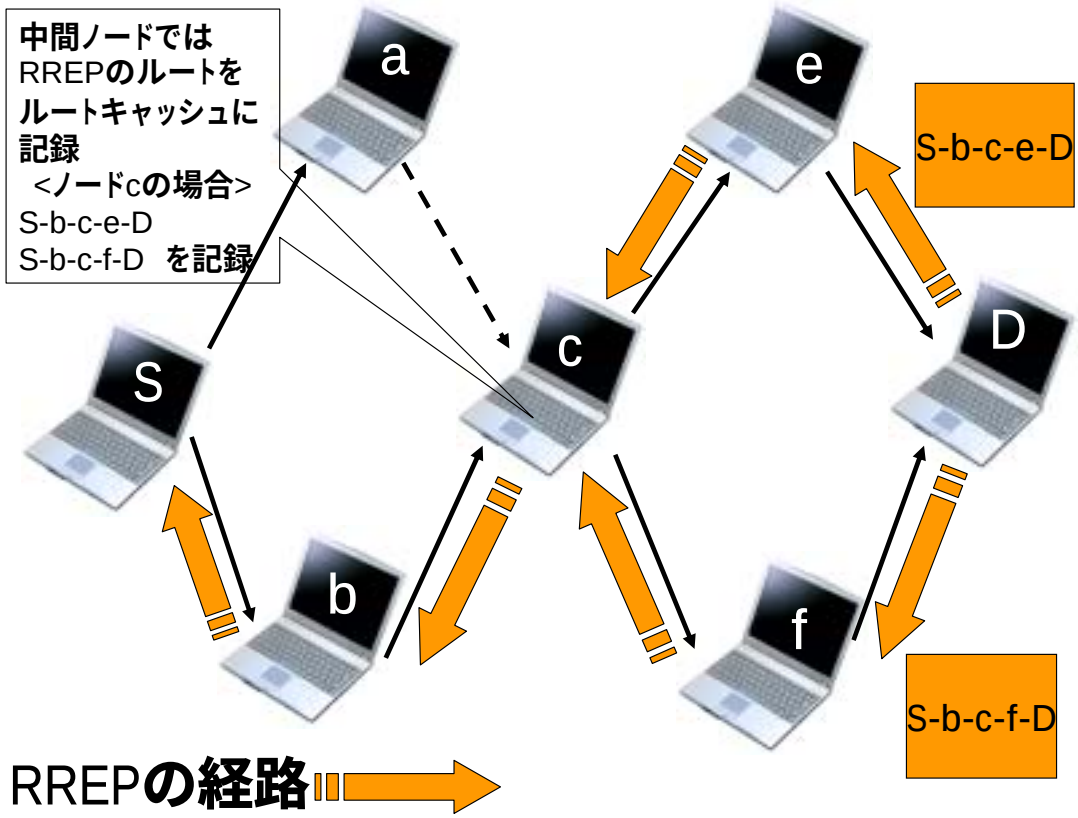


図２．１．２．RREP

Route Discovery を開始する際に、送信元ノード S はデータパケットを「Send バッファ」と呼ばれるバッファに保持する。Send バッファには、宛先までのソースルートがまだないために送信することができないパケットがバッファリングされる。Send バッファにおけるパケットはタイムアウト時間後に廃棄される。

パケットが Send バッファ内にある場合には、送信元 S はパケットの宛先に対して Route Discovery プロセスを続けるが、限られたワイヤレス通信範囲やネットワーク内におけるノードの移動によって宛先ノード D と通信できない可能性がある。このため、送信元ノード S はその Route Discovery 頻度を制限することで RREQ の増加によるネットワーク負荷の増大を防ぐ。

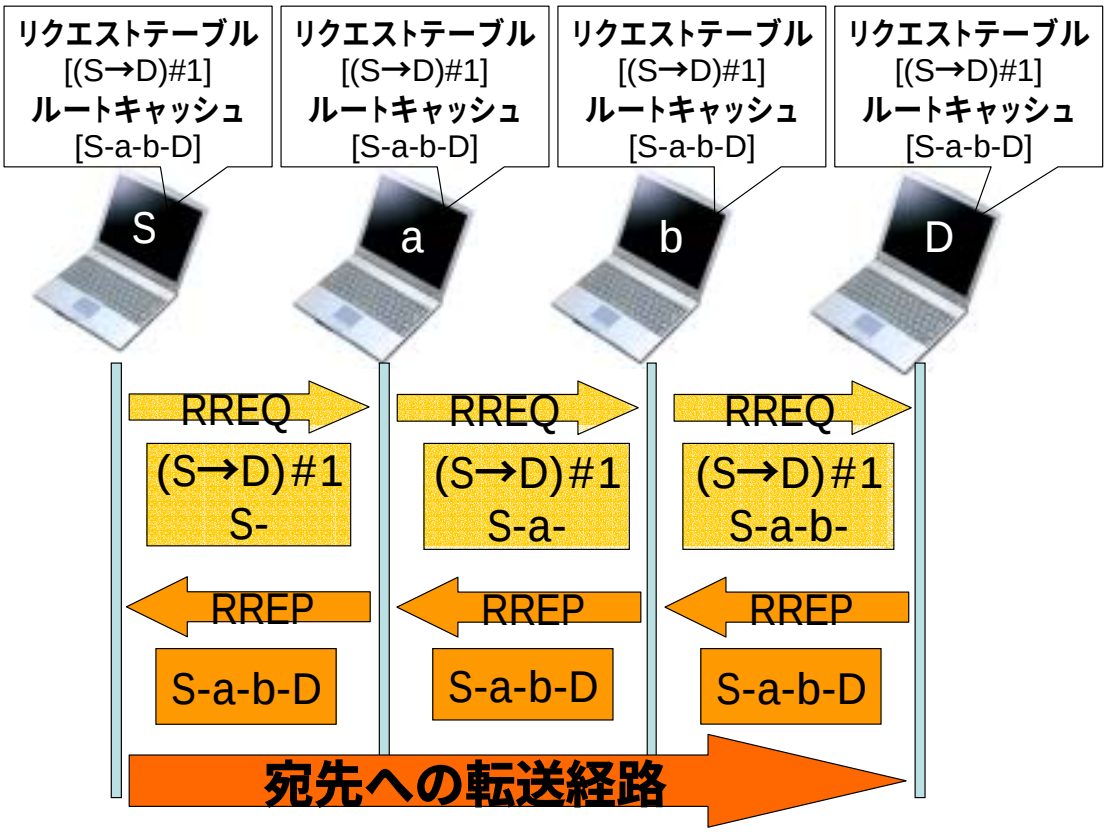


図２．１．３　Route Discovery 概略図

２．１．３　Route Maintenance

ソースルートを使ってパケットの転送を行なうとき、次ホップへの経路が使えるかどうか確認しなければならない。

経路が使えるかの確認は

1. リンク層 Acknowledgement を用いる方法
2. passive acknowledement を用いる方法
3. Acknowledgement Request オプションを用いる方法

の３つがある。

IEEE 802.11 のような無線 LAN を使っている場合には、１のリンク層 ack が使用され、そうでない場合は 2 の方法が利用される。2 の passive acknowledgement とは、次のノードがさらに次のノードへ送信したパケットを盗み聞きする方法のことである。1，2 ともに利用できない場合は、DSR の Acknowledge Request オプションにより確認応答がなされる。

ack が届かなかった場合、パケットを送信したノードは次ホップへのリンクがブレークしたとし、そのリンクをルートキャッシュから削除するとともに、Route Error(以下、RERR)をそのリンクを使ってパケットを送っていた各ノードへ送信する。例えば、図２．１．４で S-a-b-c-D で通信をしていて、もし b が c からの ack を受信できなかった場合、b は S へ RERR を返し、RERR を受信したノードはこのブレークしたリンクをキャッシュ内から削除する。上位層が TCP のようなプロトコルであればデータパケットの再送がなされる。宛先 D への再送や、次のパケットの送信には、もし送信元 S がそのルートキャッシュに別の有効なルート（例えば前に行なわれた Route Discovery による RREP によるものや、他のパケットの盗み聞きによって得られたもの）があれば、即座にその新しいルートをつかって通信が始まる（図２．１．５）。もし別ルートがなければ、新しく Route Discovery が始まる。

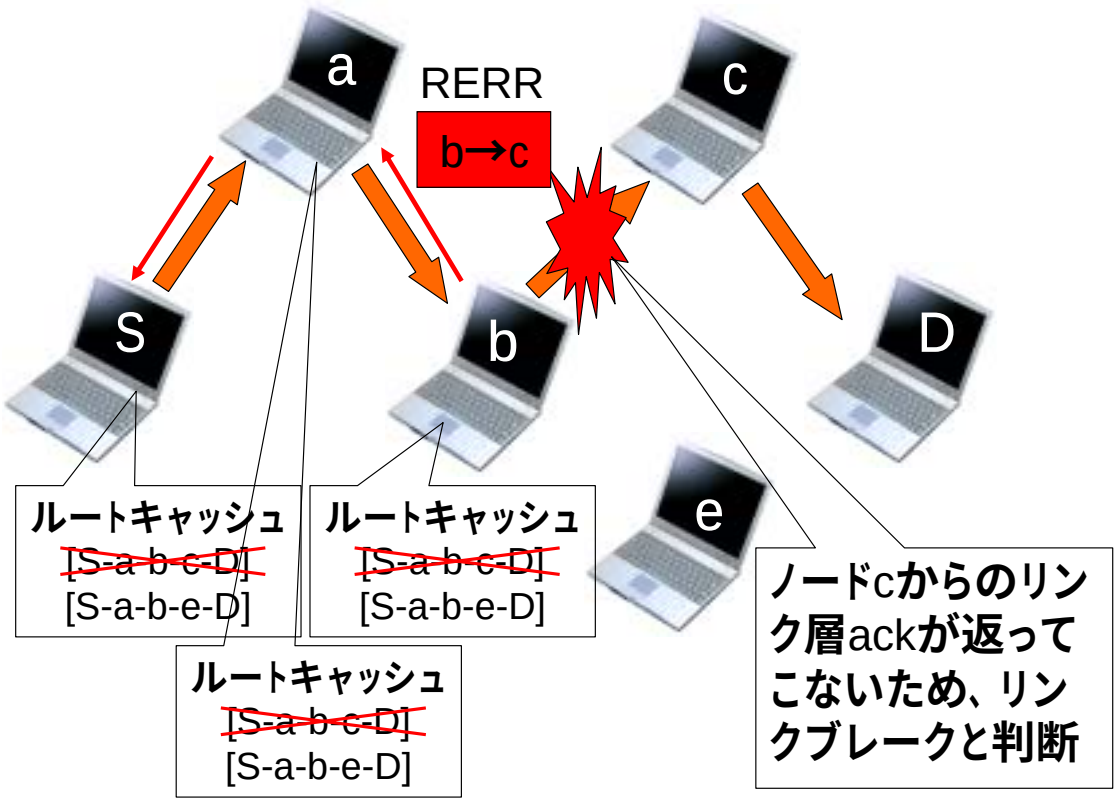


図２．１．５　リンクブレークによる RERR

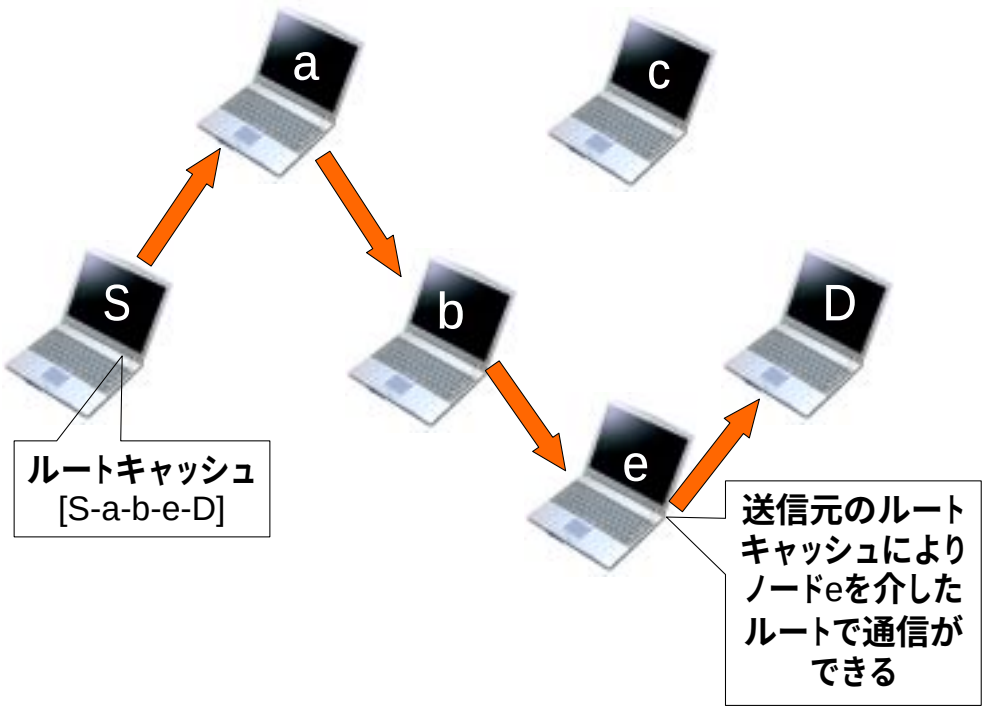


図２．１．５　RERR 後の復帰

２．１．４　Route Discovery プロセスのオプション

(1)盗み聞きしたルート情報のキャッシュ

各ノードは利用可能なルート情報を通信中のパケットから盗み聞きしたときに、そのルート情報を自身のルートキャッシュに加える。ルート情報の有用性はリンクの方向性特徴により決定され、キャッシュされる。

1. 単方向リンク
2. 単方向リンクになりうるが、このリンクは持続性がなくほとんどの場合双方向
3. 双方向リンク

というリンクがあった場合に、1 は forward 方向のみでキャッシュ、2, 3 は forward, reverse 両方向でキャッシュされる。ただし RREP のときのみ、まだこのルートを通して通信がされていない場合にはキャッシュされない。

(2)キャッシュによる RREQ への応答

RREQ を受信した中間ノードは、その RREQ の宛先 D へのルートをルートキャッシュ内に保持していないか探す。もし見つかった場合、ノードは RREQ を転送せずに、送信元 S

へ RREP を返す。RREP には、このノードまでの RREQ のアドレスリストと、自身のルートキャッシュ内の宛先 D へのルートを連結してソースルート情報を乗せる。
しかし、この方法を用いて RREP を送信する前に、そのノードは重複したルートを持たないようルートを確立しなければならない。例えば、図２．１．６．で S-a-b-c- + b-c-D = S-a-b-c-D というようにである。

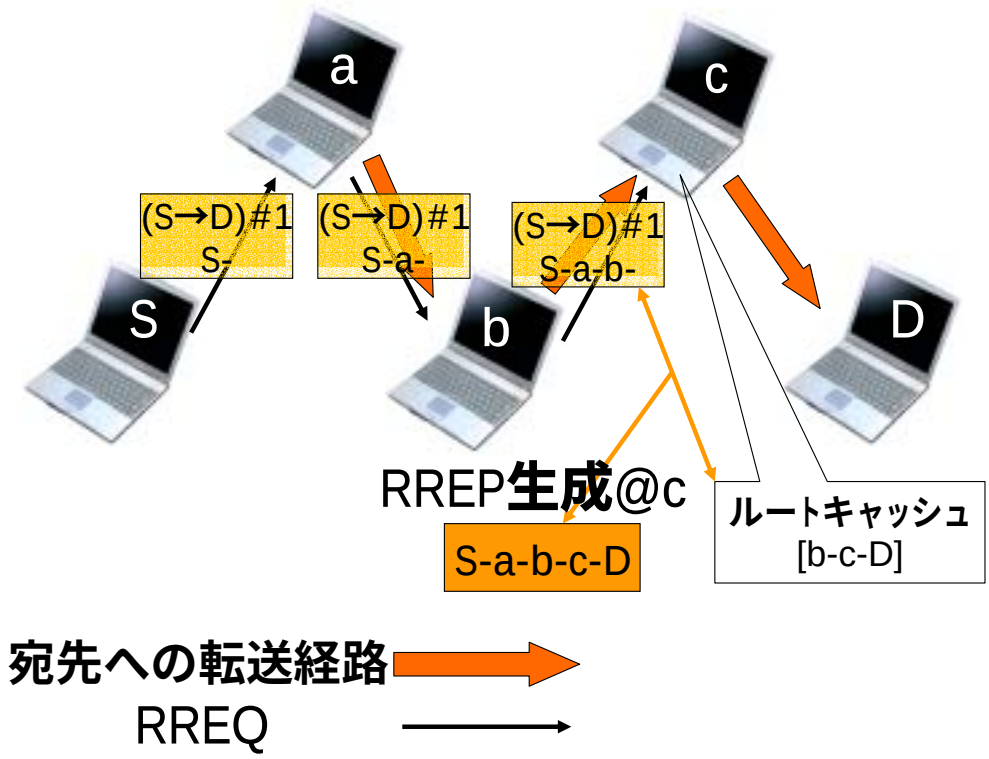


図２．１．６　中間ノードによる RREP

(3)RREQ のホップリミット

各 RREQ メッセージはホップリミットを持っており、中間ノードによる RREQ の転送回数の制限がなされる。このホップリミットは RREQ パケットの IP ヘッダの TTL フィールドを使っている。RREQ が転送されるにつれて、リミットがデクリメントされ、宛先 D へ届く前にリミットがゼロに達するとその RREQ は廃棄される。この RREQ ホップリミットは Route Discovery における RREQ の拡大をコントロールしている。

また、このホップリミットの概念により、Route Discovery の最初に、non-propagating RREQ を投げることができる。まず、送信元 S は RREQ をブロードキャストする際に、ホップリミットが１のものを送信し、この RREQ を受信したノードは再ブロードキャストを行なわない。この RREQ を non-propagating RREQ と呼ばれる。この RREQ により、宛先 D が送信元 S の近隣ノードであったり、近隣ノードが宛先 D へのルートをキャッシュし

たりしていた場合に、即座にルートを確立することができる。もし RREP が即座に受信できなかった場合には、ホップリミット 16 の RREQ をブロードキャストする。

２．１．５　Route Maintenance プロセスのオプション

(1) Packet salvaging

中間ノードが Route Maintenance を通じて、パケットを送ろうとしているノードがルートのブレークを検知したとき、もしそのノードが他のルートをルートキャッシュ内に見つけた場合は、パケットを捨てずに salvage する。パケットを salvage するときにはノードはパケット上のソースルートをルートキャッシュからのルートに置き換え、パケットはこのソースルートに従って転送されていく。例えば図２．１．７．でノード b が宛先 D への別ルートである c-e-D を持っていたとすれば、パケットのソースルートを c-e-D に置き換え、パケットを廃棄せずに salvage する。

パケットを salvage する際に、salvage された回数を記録し、ひとつのパケットが何度も salvage されるのを防ぐ。そうしないと、パケットがループに陥り、違うノードでソースルートが置き換えられ続けることとなってしまう。

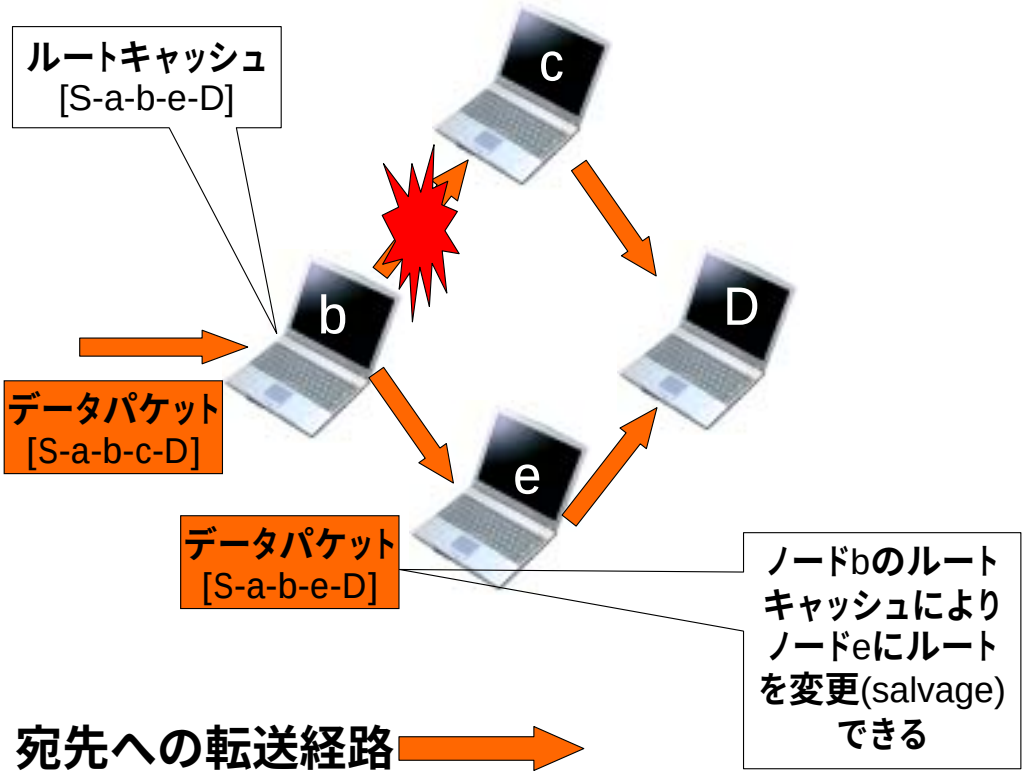


図２．１．７ Packet salvaging

(2)ルートの自動短縮

ソースルートは、中間ノードが冗長で必要でなくなった場合には自動的に短縮される。このルート自動短縮機能は２．１．３で述べた passive acknowledgement に似ている。あるノードが投げられているパケットのソースルートを盗み聞きした場合、そのパケットの次ホップがそのノード自身でなく、かつソースルートの次ホップより後ろに自分のアドレスがあれば、そのソースルート上にある自身より前の中間ノードは必要ないことになる。例えば図２．１．８でノード c が a から b へ送信しているデータパケットを盗み聞きしたとし、S-a-b-c-D のソースルートであったとする。この場合、ノード D は gratuitous RREP(以下、G-RREP)を送信元 S へ返す。この G-RREP は送信元 S から盗み聞きされたパケットの送信ノード(a)までのルートと G-RREP を送信したノード(c)から宛先 D までのルートを連結したものが記載される。この例では S-a-c-D を得る。

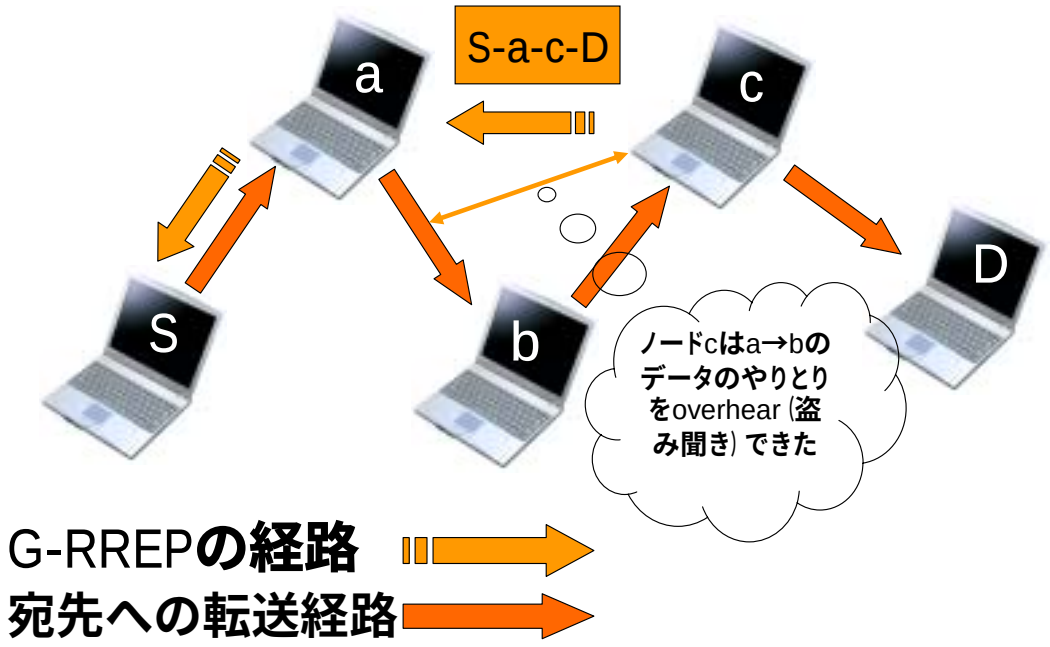


図２．１．８．　G-RREP によるルート短縮

(3)RERR メッセージの拡張送信

送信元 S が RERR を受信した際、ノード S はこの RERR を次の RREQ にピギーバックして送信する。これにより、ノード S の周りのノードは無効になったルートを自身のキャッシュから削除することができるようになる。

先の図２．１．５で、ノード S が b-c 間がブレークしたとする RRER をノード b から受信する。そして、S は自身のキャッシュからこのリンクを削除し、ルートキャッシュ内に別ルートがもしなければ新しい Route Discovery をはじめる。このときの RREQ パケットには、ノード S は RERR のコピーをピギーバックし、RERR が他のノードに広がっていく。そして、各ノードが RREP を返す際には、ブレークしたリンクを含まないルートを返すことになる。

２．１．６．　DSR の問題点

(1)古くなったルートのキャッシュ

アドホックネットワークでは通信中にノードの移動などによるリンクブレークが多発する。

２．１節中で述べたように、Route Discovery, Route Maintenance によって再接続するが、その際にルートキャッシュが問題を起こす。DSR ではソースルートを積極的にキャッシュする特徴があるが、このために、使われないルートを多くキャッシュしてしまう。また、ルートに明示的な期限を与える機能がないために、たくさんのキャッシュがあっても、その多くが時間の経つうちに無効になってしまい、このルートを使おうとすると、通信できないという場合も考えられる。この場合を考える。

もし、現在使用中のルートが切断されたとき、送信ノードはルートキャッシュ内でそのルートが使えるか知らないので、キャッシュ内のルートを使用してデータパケットを送信する。だが、ルートを使用できない場合、Route Maintenance に移り、RERR を受けた送信元が再びルートキャッシュを検索、といった動作が繰り返され、大きな遅延を生み出してしまうこととなる。

(2)Packet salvaging によるループ遅延

２．１．５(1)で述べたように、パケットが中間ノードで経路を失った場合、キャッシュ内に別ルートがあれば、Packet salvaging によって、そのルートで置き換えられる。ただし、この規定だと、図２．１．９のようにノード b でルートが置き換えられた場合に、a-b 間でループが起きる。これにより、パケットに膨大な遅延が発生する。Packet salvaging では 15 回までパケットが salvage されるのを認めて、「回数を規定すれば、パケットが永遠に salvage され続けることはない」としているが、リンク間でのループの規定はしていない。

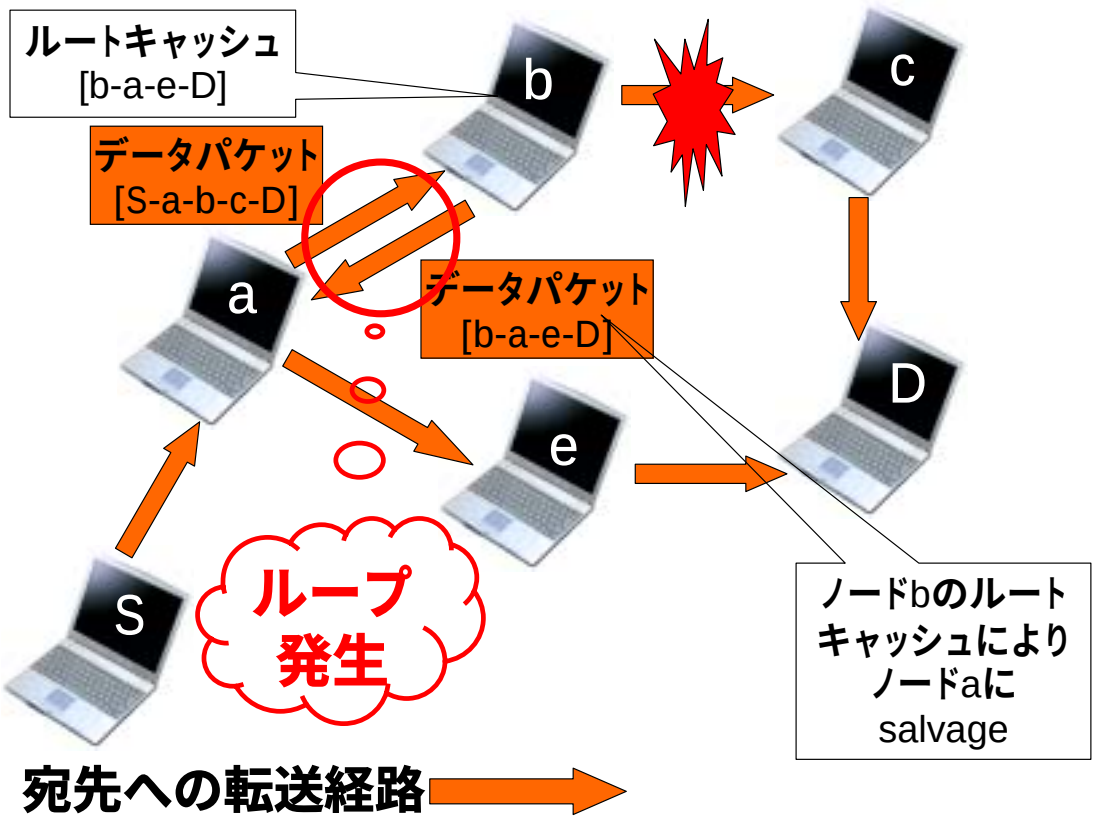


図２．１．９　Packet salvaging によるループ

２．２　Split Multipath Routing(SMR) with Maximally Disjoint Path in Ad Hoc Networks

Split Multipath Routing(以下、SMR)[4]は DSR を元に作られた On-Demand 型のルーティングプロトコルであり、最大限にディスジョイントなパス(重複した経路を持たないパス)を作る。多数のルートを保持するが、そのうちひとつのルートは最短遅延のパスであり、またそれらのルートは必ずしも同じ長さではない。データのトラヒックは、輻輳の回避とネットワークリソースの効果的な利用をするために多数のルートに分けて投げるという特徴を持つ。なお、本節では「ルート」と「パス」を同意の語として扱う。

２．２．１　Route Discovery

SMR は On-Demand 型であるので、送信元ノードが通信を始めるときにルートがない場合にのみルートを探索する。ルートの探索には RREQ/RREP が使われ、RREQ はいくつかの異なるルートを通って、重複して宛先に届き、宛先では多数のディスジョイントなルートからルートを選び、RREP を返す。

(1)RREQ 送信

SMR の目的は最大限にディスジョイントなパスを作ることにある。ディスジョイントなパスを作ることによって輻輳の回避とネットワークリソースの効率的な利用を目指している。このため、宛先はルートを選ぶことができるよう、多くの利用可能なルートを知らなければならない。

また、中間ノードがたとえ宛先までの経路情報を持っていたとしても、その中間ノードが RREP を返すことはできないようにしている。もし中間ノードが DSR や AODV のように応答してしまうと、RREQ が宛先に届く数が十分でなくなり、中間ノードのキャッシュから作られたルート情報かどうかわからないので、最大限にディスジョイントな多数のルートを作るのが困難になってしまう。

送信元がデータパケットを送信するときにルート情報を持っていない場合、RREQ が送られるわけだが、このパケットは送信元のアドレスと ID ナンバーを持っている。宛先以外のノードが重複していない RREQ を受け取ったときにはアドレスを付加して再ブロードキャストする。だが、重複した RREQ を落とすことでは、ジョイントな経路しか作ることができない。図２．２．２(a)は送信元 S から宛先 D への RREQ の転送経路であり、図２．２．２(b)は利用可能なルートを表している。この図から考えると、5 つのルートが一本の

リンクを共有していることがわかる。

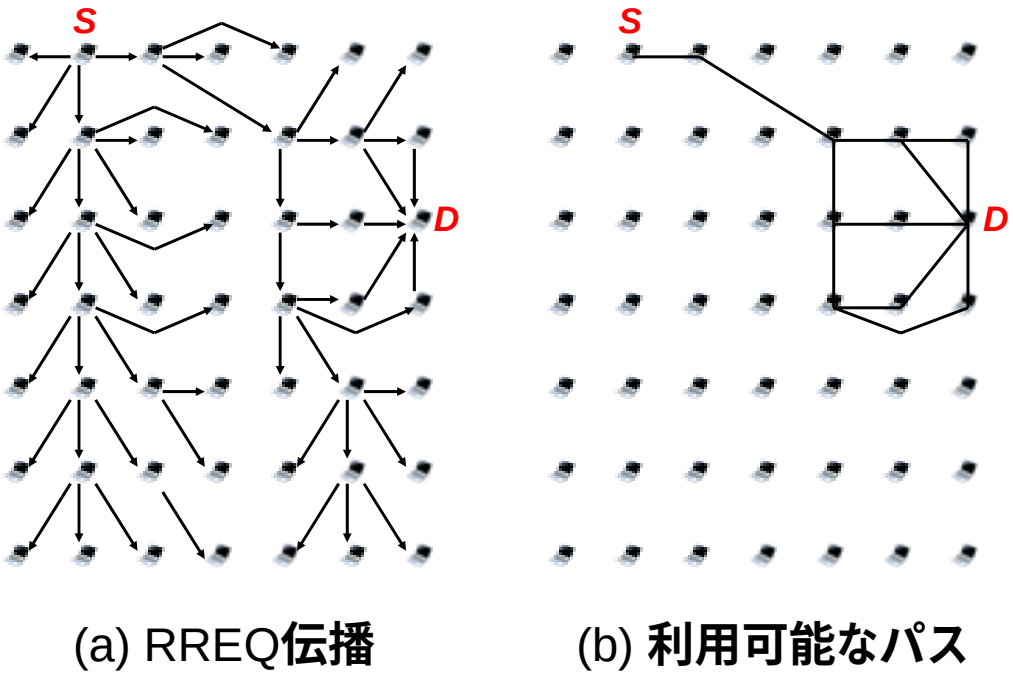


図２．２．１　重複した(ジョイントな)多数の経路

SMR では、このルートのジョイントの問題を避けるため、RREQ の投げ方を変えている。中間ノードでは重複した RREQ を受信しても、先に受信した RREQ よりもホップ数が少なければ、その重複した RREQ を転送する、という投げ方をする。図２．２．２(b)ではこの RREQ によって作られたパスを示している。この図からもわかるように、より多くのディスジョイントなパスを選択できることがわかる。

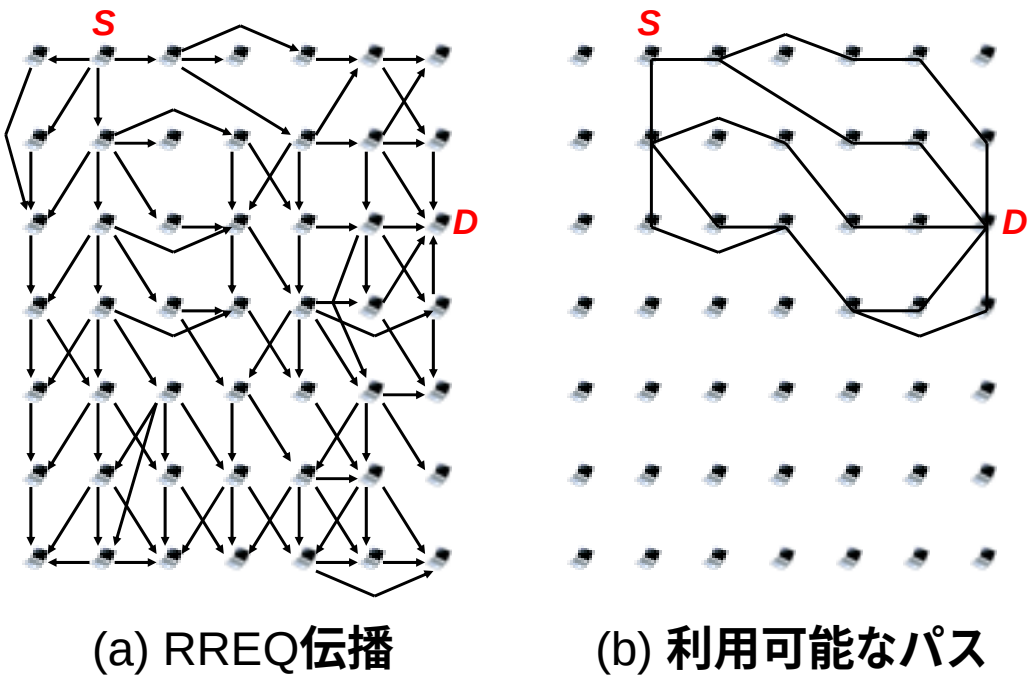


図２．２．２　最大限にディスジョイントな多数の経路

(2)ルート選択法

ルートの選択は、宛先ノードが 2 つの最大限にディスジョイントなルートを選択する。2 つ以上選ぶこともできるが、演繹的にルートの数を２つに制限している。2 つのルートのうち片方は最も遅延の短いルートであり、このルートは宛先ノードが最初に受信した RREQ によるものである。これにより、On-Demand 型ルーティングプロトコルについてまわるルート探索による遅延を最小限に抑えることができる。最初の RREQ を受信すると、宛先ノードはその RREQ のパスを記録し、そして RREP を送信する。その後、宛先ノードは RREQ を受信するために一定時間待つ。その後、すでに RREP が返されたルートと比較し、その中から最大限にディスジョイントなルートを選択する。最大限にディスジョイントなルートが２つ以上あれば、最小ホップ数のルートを 1 本選ぶものとする。もし最小ホップ数のルートが複数あれば、その中で最も早く RREQ が到着したものを採用する。このようにして決まった第２経路に対して RREP を返す。セッションとなる 2 つのルートは必ずしも同ホップ数である必要性はない。

SMR では中間ノードはそのキャッシュからルートを返さないので、送信元ノードのみが宛先ノードへのルート情報を保持する。

２．２．２　Route Maintenance

ルートのリンクはモビリティや輻輳、パケットの衝突などによって切れやすい。ブレークしたルートを即座にリカバーするのは重要である。SMR ではノードがルートの次ホップへデータパケットを転送し損ねた場合に、RERR を返す。送信元ノードは RERR パケットを受け取ると、即座にブレークしたリンクをキャッシュから削除するが、2 つのセッションのうち、片方が無効になった場合でも送信元ノードはデータパケットの通信を続ける。
送信元が片方のルートでまだセッションを保ち続けているとき、2 つのポリシーでルートの再探索をする。

1. セッションがどちらか 1 本でも切れたら、新しくセッションを張りなおす or
2. 全てのセッションが切れたら、その時点でセッションを張りなおす

２．２．３　2 つのポリシーによる性能比較

この 2 つのポリシーと DSR の三者に関して、パケット到着率、パケットドロップ数、ルーティング負荷、ホップ数、エンド間遅延について評価がなされていたが、DSR より概ね良い値を示していた。トラヒックを複数の経路に分けることによってネットワークの負荷を分散したことによる効果であると述べている。

2 つのポリシーを比べると、2 の方が良好であった。これは、1 のように経路切断後ルート探索に移ることによって、ルーティング負荷が大きくなり、結果性能を落とすということである。

２．２．４　SMR の問題点

SMR はディスジョイントなルートを作るために RREQ を中間ノードで廃棄する数を少なくしている。これはネットワーク上に多くのブロードキャストパケットが流れることになり、ネットワーク負荷の増大を意味している。

また、ディスジョイントなルートを選び、その中の最小ホップのものを選ぶため、結果としてルートの長さが長くなってしまうことがあり、これは遅延の増大を招く。

２．３．マルチパスルーティングの性能評価

この研究[6]ではマルチパスの評価に当たり、DSR をマルチパス化した際にどういった経路の選び方をするとルート探索回数が減るかということについて言及している。DSR は元々多数のルートを保持し、最初のパスが切れた際に別経路を使うということはするが、その経路の管理をしないために多くのルートを保持しすぎるといったことで、その多くの経路を活かすことができていない。このため、DSR を用いて、ディスジョイントな経路を使った２つのマルチパス案を評価している。

２．３．１　DSR のマルチパス拡張

２つのマルチパス案をプロトコル１・２としている。

(1)プロトコル１

宛先ノードは最初に自身に届いた RREQ に対しては普通に RREP を返信する。その際にそのルートを覚えておき、宛先ノードは後に到着した RREQ のうち、ディスジョイントになっているものだけ選び出し、RREP を返信する。ディスジョイントにするのはひとつのルートが無効になる際に、他のルートが影響を受けず無効になりにくいからである。これは暗示的に RREP の返信数を抑えている。送信元では最初のルートが切れた際には別のルートを使い、全てのルートが切れたときにルート探索を行なう。(図２．３．１(a))

(2)プロトコル２

プロトコル１では送信元ノードのみが別経路を保持している。この場合、最初のルートがルート中間で切れた場合に RERR パケットが送信元に返され、別経路による通信が始まる。これでは、すでに通信路中にあるデータパケットは廃棄されてしまう。これを避けるため、全ての中間ノードがディスジョイントなルートを宛先に対して持つようにする(図２．３．１(b))。これを行なうためには、宛先ノードは最初のルート上にある中間ノード全てにディスジョイントなルートである RREP を送信することとする。そのような中間ノードに常にディスジョイントな別経路を返すことができるとは限らないが、この研究では可能性を抜きにして、分析を行なっている。

図２．３．１(b)によりこの別経路の利用法を説明すると、まずリンク L(i)が切れたとすると、ノード n(i)は L(i)-L(k)間のルートを P(i)に置き換える。P(i)が切れると N(i-1)まで RERR を返し、ルートを P(i-1)に置き換える。これを繰り返し、RERR が送信元に変えると、新たに経路探索がおこなわれる。

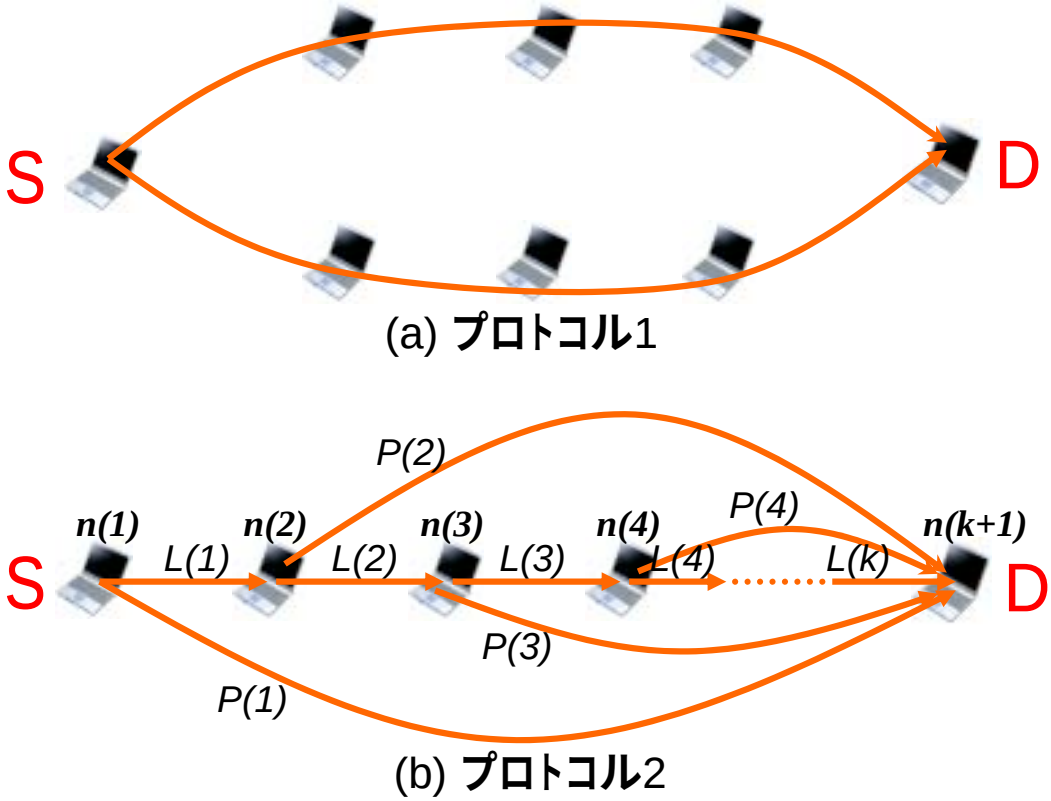


図２．３．１　DSR のマルチパス化

２．３．２　性能評価

理論値評価で経路探索回数はプロトコル１・２ともにシングルパスよりも減っていた。また、プロトコル２のシミュレーション評価でルーティング負荷などにおいてもシングルパスよりマルチパスの方が勝っていることが述べられていたが、遅延の面ではマルチパスが劣るという結果が出ていた。これはマルチパス化して別経路を保持した際に、その別経路が長くなると、マルチパスの効果が薄れ、またリンクも切れやすくなるという結果である。また、別経路の保持数に関しても１つか２つ程度がよく、多数保持しても、その別経路による経路探索回数減少の恩恵は収束してきてしまうという結果が出ていた。

２．４　IEEE802.11 方式無線 LAN

２．４．１　CSMA/CA

MAC 層では２つの異なるアクセス方法を定めている。DCF(the Distributed Coordination Function)と PCF(the Point Coordination Function)である。アドホックネットワークで使われるのは DCF のほうである。

基本的なアクセス機能である DCF は CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance)の機能である。CSMA は Ethernet で使われている CSMA/CD(CSMA with Collision Detection)としてよく知られている。

無線 LAN では衝突を検出できないため、CA 機能によって衝突を回避する。各端末はキャリアセンスによって通信路の状態を確認する。チャネルの状態がアイドルであればただちに信号を送信し，ビジーあればアイドルになるまで送信を延期する。

送信元では、キャリアをセンスした際にアイドル状態であれば、DIFS(DCF Inter Frame Space)の時間待ち、そのときアイドルであればデータを送信する。逆にビジー状態であればチャネルがアイドルになるまで待ち、アイドルになってから DIFS 時間後にバックオフに入る。

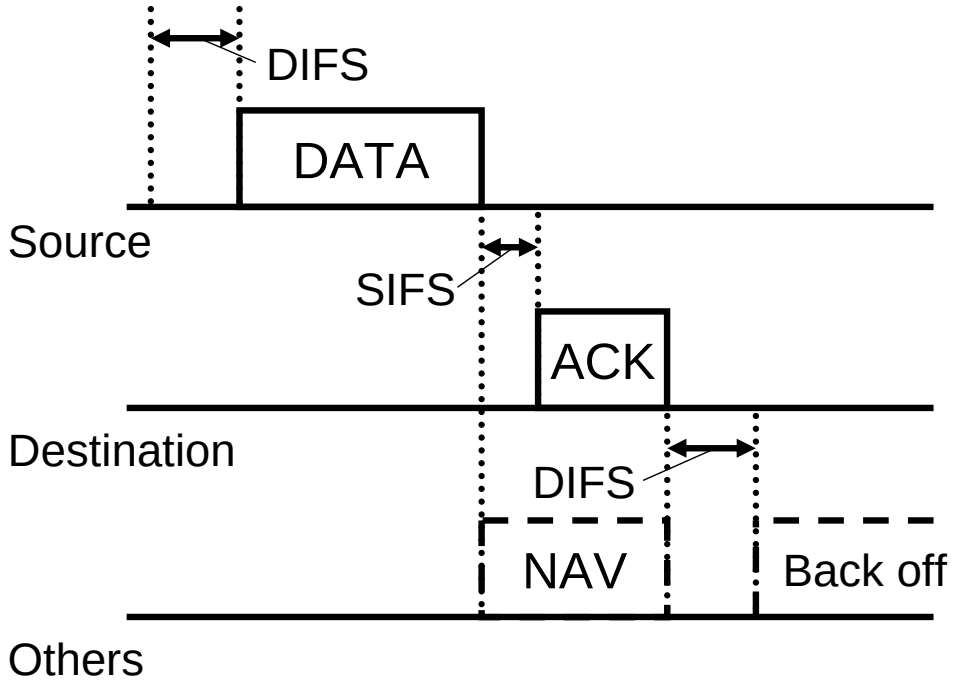


図２．４．１　各ノードにおける送信フェーズの制御

このバックオフ待ち時間は最小限の時間にランダムな長さの待ち時間を加えたもので、直前の通信があってから一定時間後に複数の端末が一斉に送信する事態を防止している。受信先とは違う端末がデータパケットを受信した場合、送信されたパケットが自分宛でないと確認すると、NAV(Network Allocation Vector)をセットし、現在送信中のパケットの送信完了後に ACK が返るまでの時間待つ。

データの受信先は受信完了後、データの CRC(the Cyclic Redundancy Check)をチェックし、 SIFS(Short Inter Frame Space)時間後、送信側に ACK を返す。ACK の受信は衝突のなかったことを示し、送信元が受信しなかった場合は、ACK を受信するまで待つかデータを再送するかする。

２．４．２　隠れ端末問題

CSMA では図２．４．２で示されるような隠れ端末問題については避けることができない。これは、ノード A、B、C があった場合に、A は C と、B は C とそれぞれ通信できる範囲にあり、A と B が通信できない範囲にあるとする。A から B へのデータ送信中に A を感知できない B から C へデータを送信してしまうと、パケットが衝突してしまう問題である。

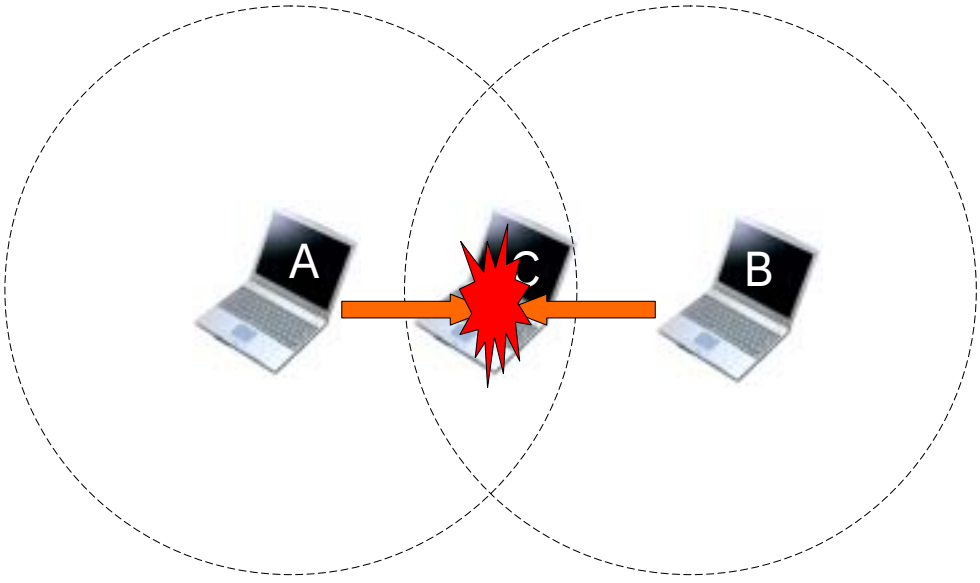


図２．４．２　隠れ端末問題

隠れ端末問題を解決するために、IEEE802.11 には標準で仮想キャリアセンス機能である RTS(Request to Send)と CTS(Clear to Send)がある。RTS は、ACK がやりとりされるまでの時間情報をもっている。RTS を受信ノードが受信して、データを受信可能であれば CTS を送信する。CTS も RTS 同様の時間情報を持っており、CTS を受信した送信ノードはデータを送信する。RTS and/or CTS を受信したデータの受信先でない他のノードは NAV をセットし、送受信が終わるまで何もせずに待つ。こうしてパケット衝突を回避する。

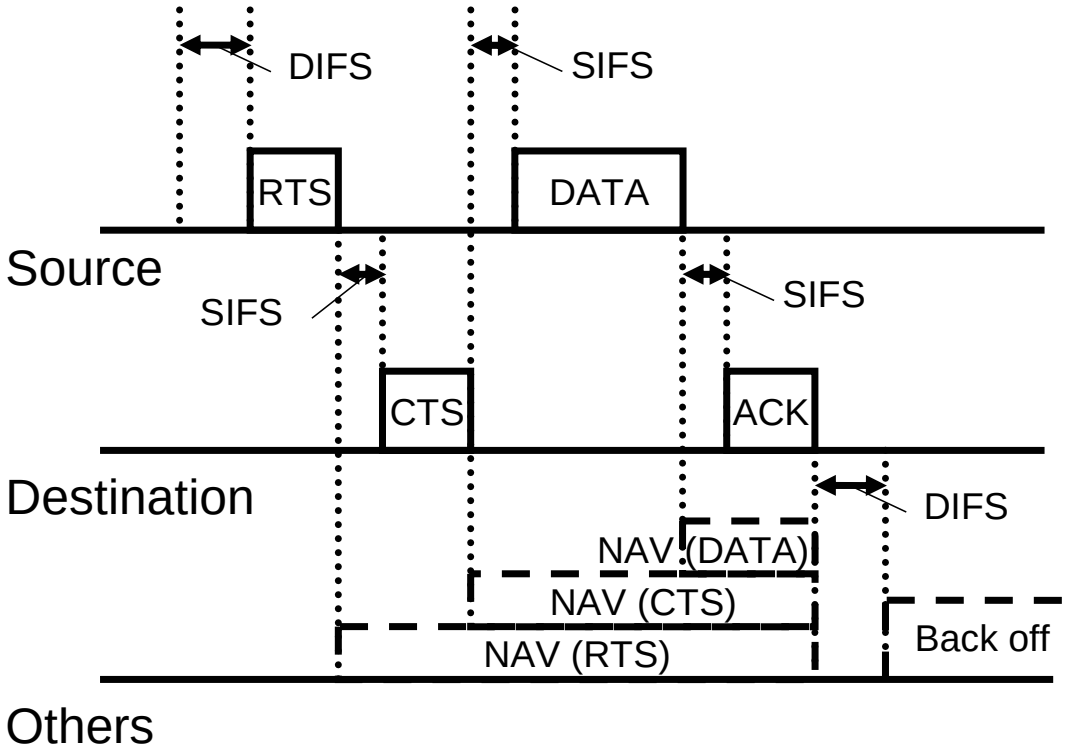
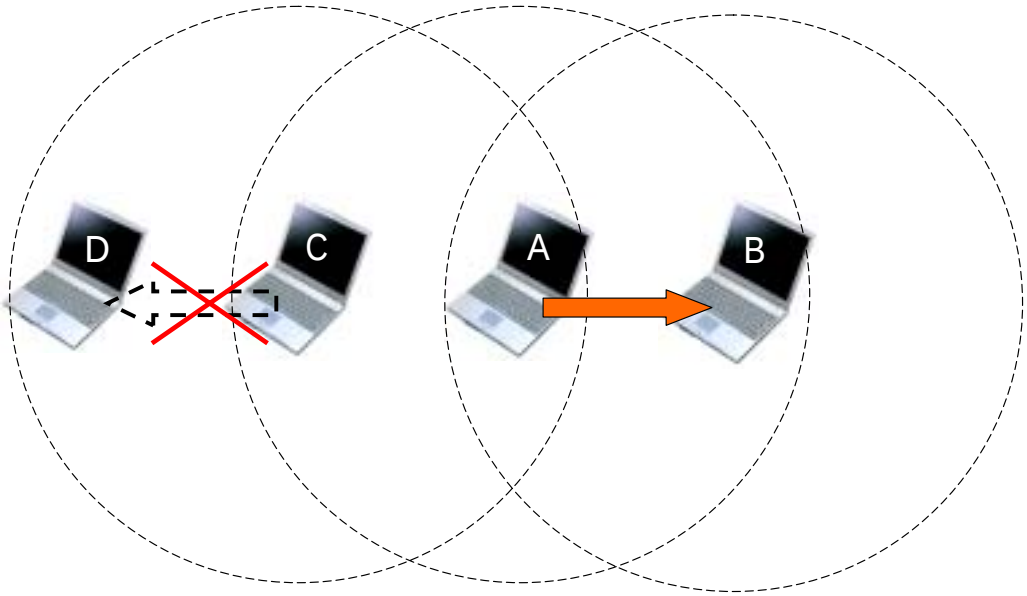


図２．４．３　RTS/CTS 交換による衝突回避

２．４．３　さらされ端末問題

隠れ端末問題のほかに、さらされ端末問題(exposed node problem)がある。上記の RTS/CTS 交換によって隠れ端末問題は解決されるが、さらされ端末問題は解決されない。さらされ端末とは、図２．３．４．で示すように、送信ノードの通信を感知できる範囲に入っているが、受信先ノードの範囲外というノードである。これにより、ノード A が B に通信を始めると、C は本来ノード B での衝突の心配のない、他のノード D への通信を控え

てしまうこととなり、ネットワーク全体では利用可能な帯域が不十分になる。この問題はアドホックネットワークにおいて深刻な問題であり、特にユニキャストパケットが増えてくると、RTS/CTS ハンドシェイクのために周りのノードが通信できない状態になってルート失効やスループットの低下の原因になることが多い。



２．３．４　さらされ端末問題

第 3 章　提案手法

３．１　提案手法の目的

提案手法では、DSR を拡張し、ソースルーティングを用いたディスジョイントなパスのルーティングプロトコルについて提案する。２．１．１で述べた Packet salvaging におけるループ回避と、ルーティング負荷の増大を伴わない最小ホップ数ディスジョイント経路の作成アルゴリズムについて提案し、エンド間遅延の減少等を目的とする。

３．２　提案手法

３．２．１　Packet salvaging のループ回避

２．１．６で述べたように、Packet salvaging をすることによってパケットドロップを抑えることができるが、その代わりループが発生する危険性があり、このループが遅延を増大させる要因となる。Packet salvaging のループを回避することでエンド間の遅延特性等を改善することができる。

Packet salvaging ではただ単に中間ノードのルートキャッシュから別ルートに置き換える、といった手法がとられていた。ソースルート上には各ノードのアドレスが載っており、どのノードを通るかがパケットを投げる時点で判別できるので、これを利用し、以下のように変更する(図３．２．１)。

① 中間ノードでは、ルートキャッシュ内からパケットの宛先へのソースルートを検索する。
② ルートが見つかった場合、パケットがその中間ノードに到達するまでのルートと、Packet salvaging によって置き換えられるルートのアドレス列を比較し、経路が逆戻りしていないかどうか(ループしていないかどうか)確認する。
③ 確認してループがなければ、そのルートをパケットに格納し、検索を続けて、もしホップ数が少ないルートが他にあれば、そのルートを採用する。

以上の手順を行なうことにより、Packet salvaging によるループを回避することができる。

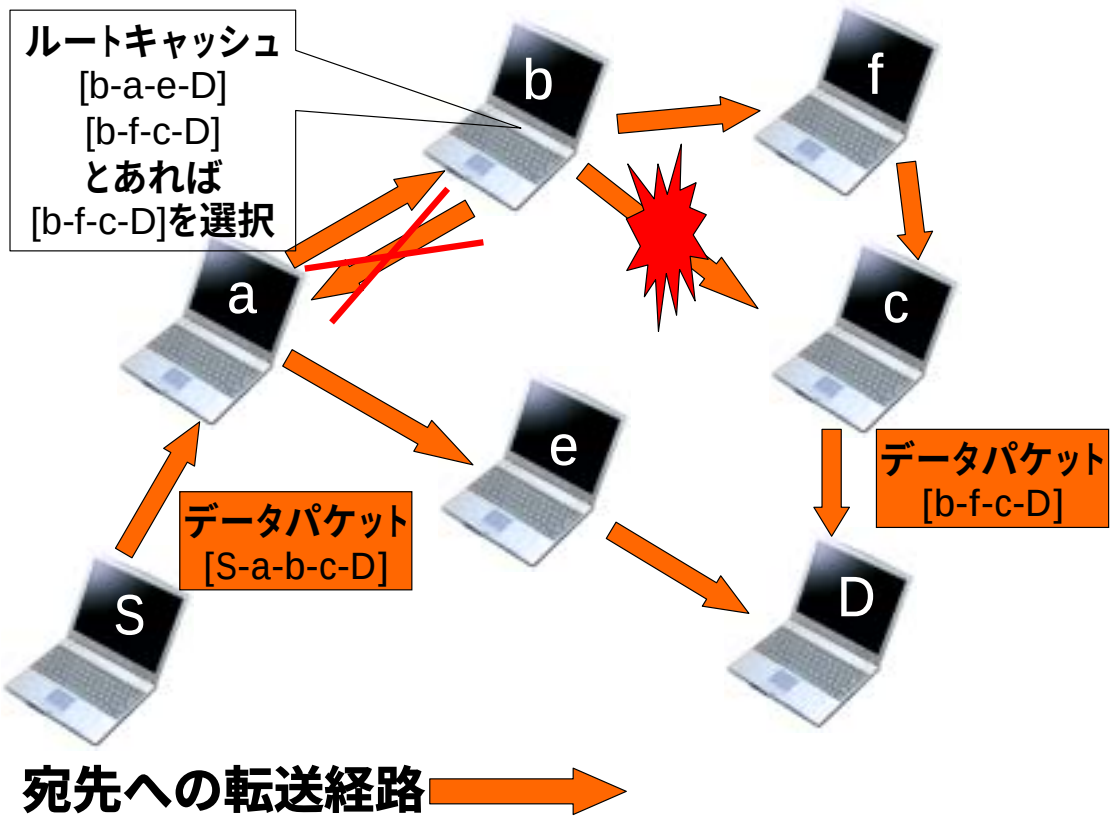


図３．２．１　ループフリーな Packet Salvaging

３．２．２　ディスジョイントな経路作成アルゴリズム

SMR ではデータパケット送信の負荷を分散させるためにディスジョイントな経路を作成していたが、その経路作成のためにネットワーク内に多くの RREQ パケットを流し、ルーティングの負荷を増していた。これはネットワーク内のセッション数などが増えるにしたがって、由々しき事態となってくる。そこで、DSR と同数以下のルーティング負荷によってディスジョイントな経路を作成する経路作成アルゴリズムを提案する。

まず、一般の RREQ の投げ方で経路を作成すると２．２．１で述べたように、図３．２．２のような経路が作成される。これは

「RREQ の転送数を制限するためとルートのループ回避のため、各ノードは、

- リクエストテーブルと対比し、RREQ 内の ID と宛先アドレスが同一の RREQ を以前に受け取っていない　or
- RREQ 内に自分のアドレスが含まれていない

場合にのみ、RREQ を転送する。」
という制限によって、RREQ によって作られたルートが分岐後に再合流する際に破棄されるため、宛先ノードにのみ分岐が合流するという図３．２．２の形のルートが形成されるのである。

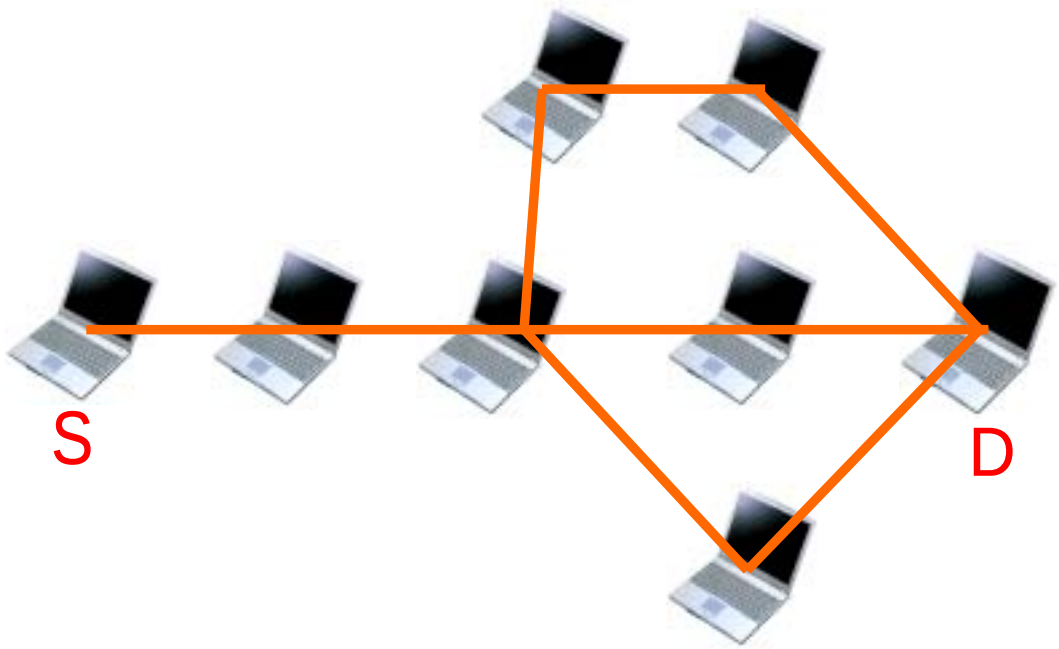


図３．２．２　ルートの概形図

だが、実際には、図２．２．１(a)に点線矢印を加えた図３．２．３からわかるように、構築された経路上の中間ノードでも「RREQ は届いているが廃棄されている状態」であり、送信元から見た Forward 方向の経路(以下、予備経路とよぶ)は保証されている(図３．２．３(b))。SMR でディスジョイントパスとして使われているのはこの予備経路であり、この経路を利用することで DSR の Route Discovery 時と同数以下のルーティングパケット数でディスジョイントな経路を構築することができる。

拡張するのはリクエストテーブル、RREP パケットであり、これらを以下(1)(2)の手順によって、予備経路を利用する。

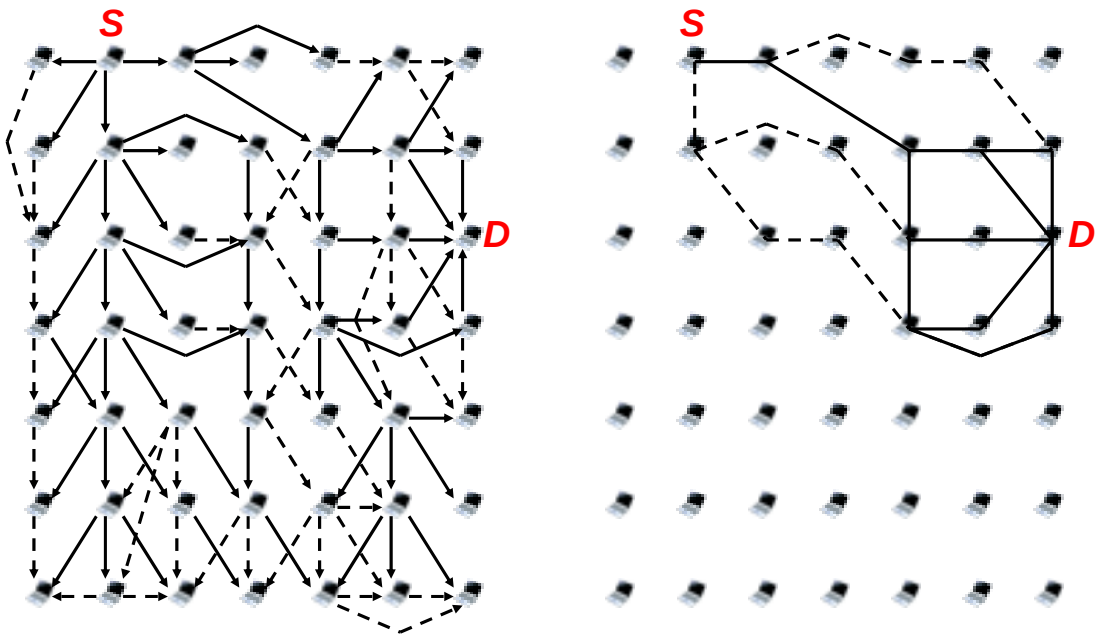


図３．２．３　予備経路情報

(1) RREQ 拡張

リクエストテーブルは RREQ のループ回避のために送信元アドレス、宛先アドレスと ID しか持っていない。このリクエストテーブルを拡張し、ソースルート情報を保持できるようにする。そして、以下の手順でテーブルを更新していく。

① テーブルの RREQ ID がパケットの RREQ ID と等しく(つまり、同じ RREQ)、かつテーブル上での、送信元Ｓからそのノードまでのホップ数よりもパケット上の送信元からそのノードまでのホップ数が同数以下ならば、テーブル上のソースルート情報を書き換える。

② そして、RREQ を廃棄。それに当てはまらない場合は、DSR と同じように廃棄、転送を行なう。

(2) RREP 拡張

RREP にはルートの分岐点を示すフラグを載せる。これにより、RREP を予備経路方向

に流し、ディスジョイントなパスを作成する。手順は以下の通り。

① Destination ノードは最初に届いた RREQ に対し、RREP を返す(図３．２．５における経路①[S-e-d])
② D ノードはある程度 RREQ の到着を待ち、最短ホップの RREQ を選ぶ(図３．２．６における経路②と③で③を選択)
③ そして最初に届いた RREQ と、その最短ホップの RREQ のソースルートのアドレスを上流(Source ノード側)から順に比較。両者が違う場合には、パスの分岐点とわかる。分岐後の１つ目のノードを分岐フラグとして RREP に追加する (図３．２．７におけるノード b が分岐フラグのたつノード)
④ フラグのたったノードより下流(Destination 側)では、リクエストテーブルを参照し、予備ルートがあり、かつその予備ルートは、分岐ノードよりも上流でジョインしている、または、完全にディスジョイントになっている(パケットとテーブルのソースルートを比較して、アドレスの違うノードが分岐フラグのアドレス以前にある)場合に、予備ルートに RREP の転送先を変える。その際に、分岐フラグ情報もまた書き換える。(図３．２．８ではノード c に予備ルートはあるが、分岐ノード e よりも上流でジョインしていないため、予備ルートに転送先を変えない。図３．２．９ではノード b に予備ルートがあり、かつ完全にディスジョイントになっているので、RREP の転送先をノード a に変える)
⑤ もし別ルートがない or 上記④の条件に当てはまらなかった場合には RREP をそのまま返す。(図３．２．１０におけるノード a では普通に S に RREP を転送する)
⑥ 以上④と⑤を繰り返すと、最悪どこかのノードで他経路がなくならない限り、最小ホップ数でディスジョイントになる経路ができる。

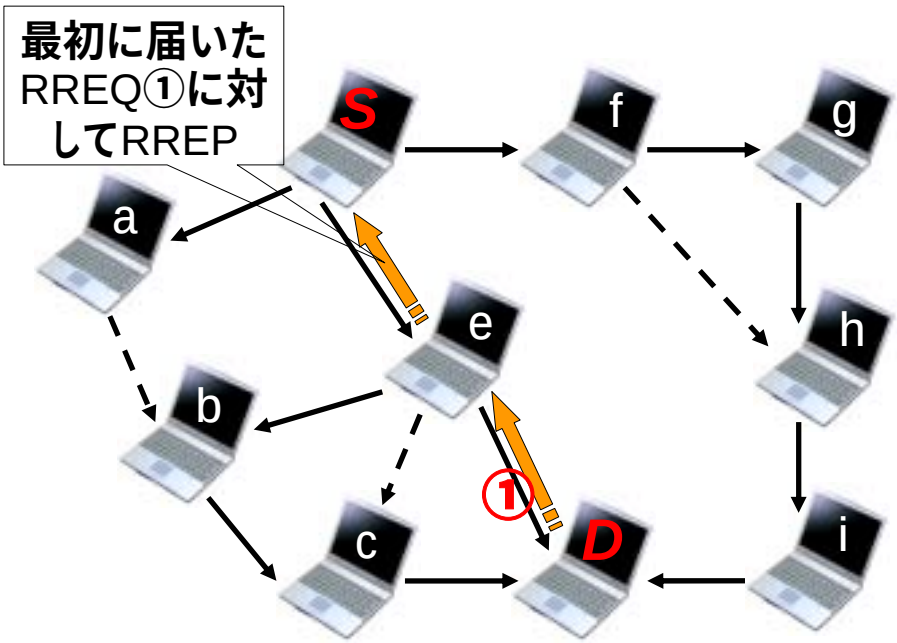


図３．２．４　RREP 転送動作①

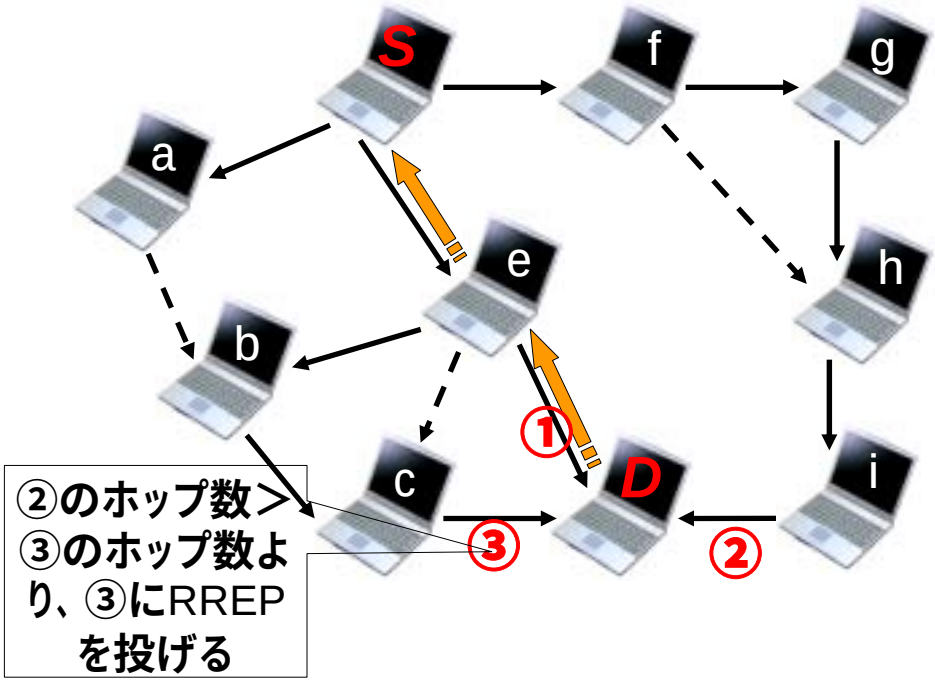


図３．２．６　RREP 転送動作②

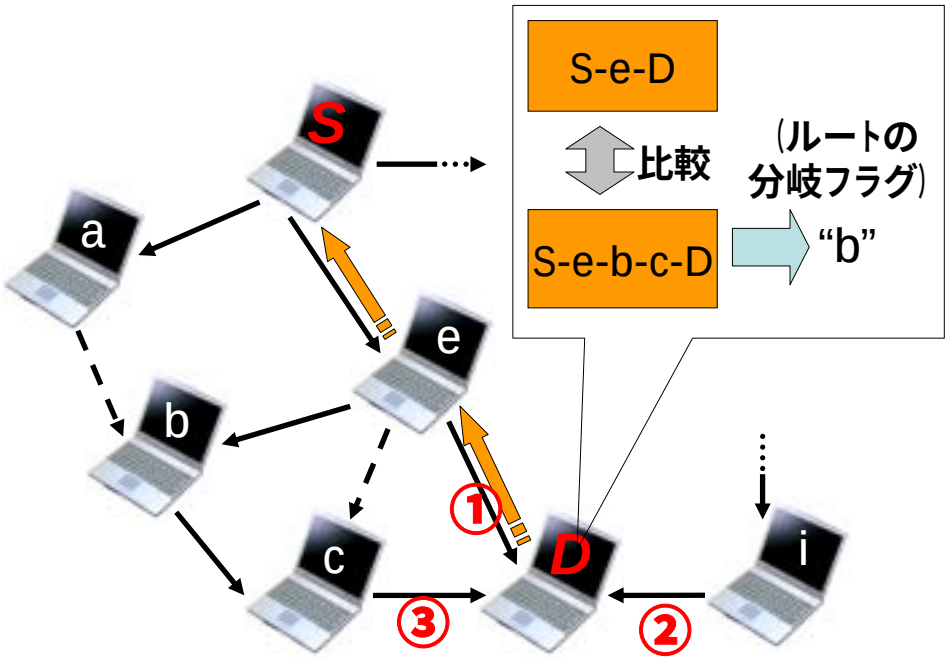


図３．２．７　RREP 転送動作③

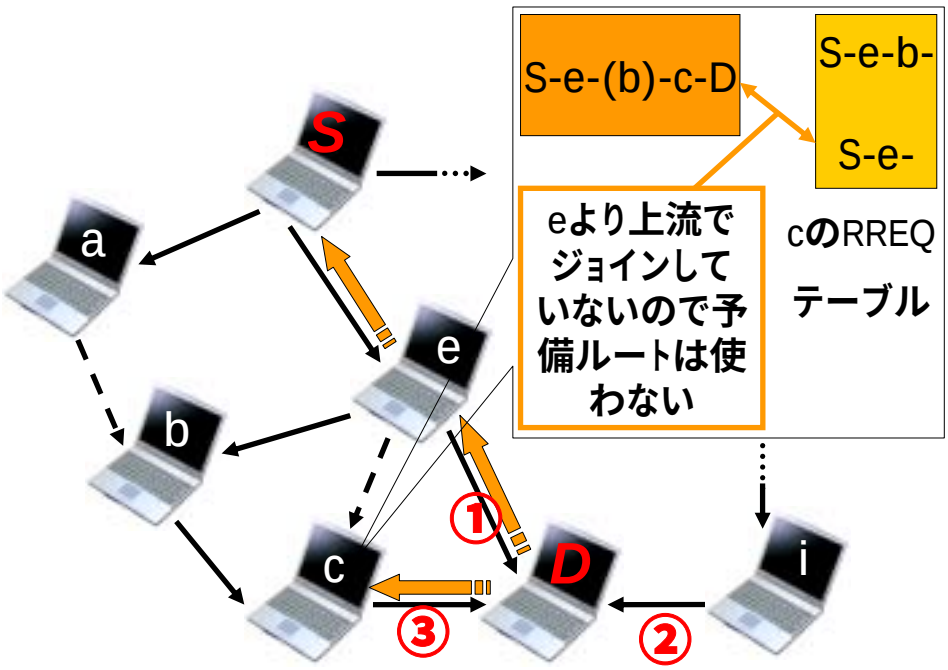


図３．２．８　RREP 転送動作④－(1)

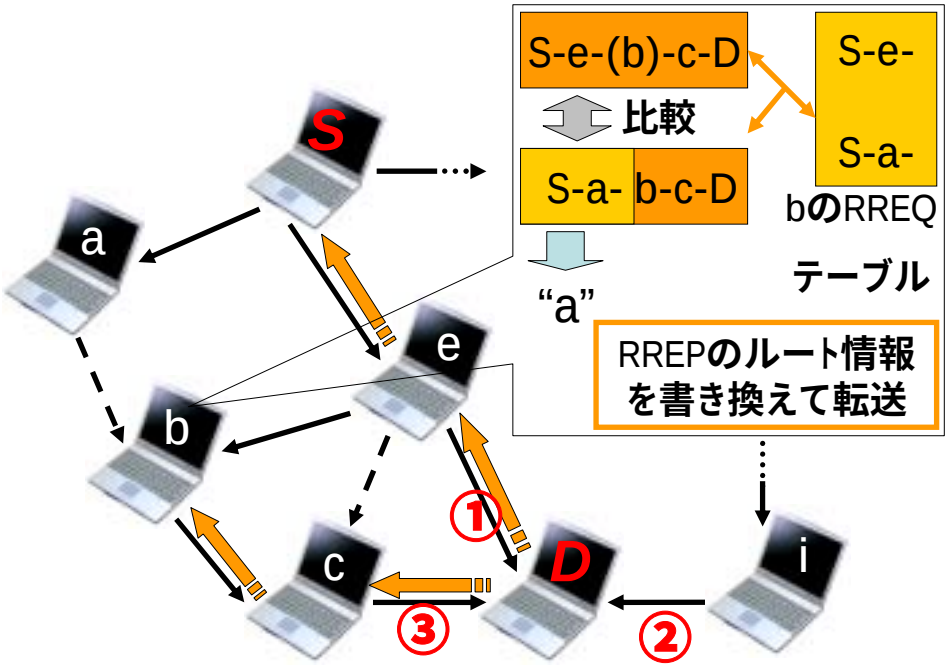


図３．２．９　RREP 転送動作④－(2)

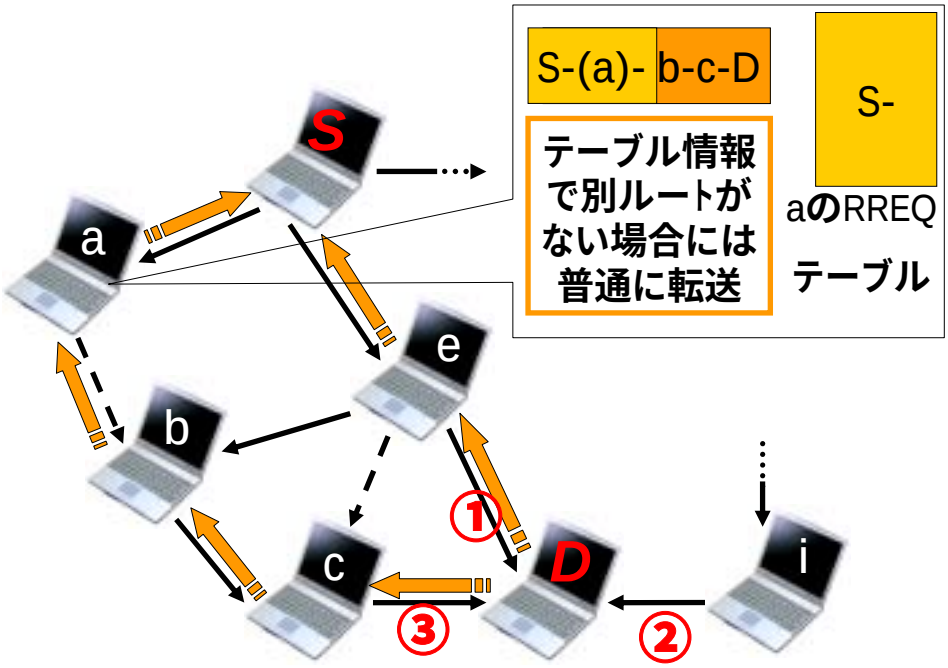


図３．２．１０　RREP 転送動作⑤

感覚的には、ジョイントなパスを引き剥がして、X や Y といった近隣ノードを使って、新たなパスを作るということである(図３．２．１１)。

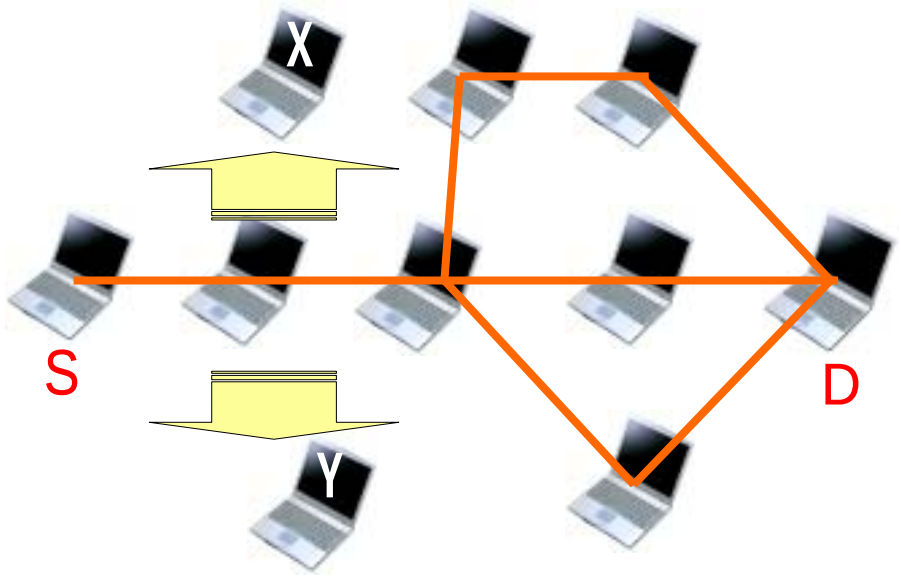


図３．２．１１　提案概念図

このアルゴリズムを使い経路を作成する。RREP パケットの情報量は分岐フラグの分わずかに増えるが、ディスジョイントなパスを作り、複数経路にデータのトラヒックを分散することができ、SMR のように RREQ にともなうオーバーヘッドの影響を受けない。

また、２．３．２で、別経路が長くなると性能が悪くなるという指摘があったが、最小ホップ数経路をディスジョイント化するので、ホップ数が最初に通ったパスより 2 ホップや 3 ホップも多くなるということはない。

３．２．３　その他の動作

今回の DSR 拡張では、SMR とは違い、キャッシュからの RREP 応答も可とした。これは、経路探索時の遅延を減らすほか、例えば、リンクが途中切れてしまったとしても下流(宛先ノード側)ではまだその経路がキャッシュされているため、中間ノード応答を可にし、中間ノードで再接続すれば、再びそのルートが使えるからである。また、このため、必ずしもディスジョイントな経路ばかりを送信元が保持するとは限らないので、トラヒックを分けるディスジョイントなパスは送信元ノードが自身のルートキャッシュ内から選び出すものとする。

第４章　シミュレーション

４．１　シミュレーション環境

提案手法の有効性を示すために、コンピュータによるシミュレーション評価を行なう。評価に当たっては UCB/LBNL/VINT プロジェクトにより作成されたネットワークシミュレーションソフトウェア、Network Simulator 2(以下、ns)[8]を使用した。
ns はスクリプト言語 Tcl/Tk を用いて、ネットワークのトポロジーや負荷状況を設定することができ、また、各プロトコル等のメインプログラムは C++言語で記述されており、ユーザが自由に書き換えることで、あらゆるプロトコルを性能評価することができる。

４．２　ns における提案の実装

ns において提案を実装する際に、既存の DSR のソースコードを用いたが、その変更点を以下に示す。

４．２．１　既存の DSR の修正

ns で実装されている DSR のバックオフアルゴリズムにドラフトと異なる点があったので、修正した。ドラフトでは RREQ を再送して Route Discovery を行なうタイミングは 500msec からバックオフアルゴリズムによって倍になっていき、最高 10sec としているが、ns 上ではこのバックオフが 2sec,8sec,10sec となっていた。このため、ルート探索回数が増えるほど遅延への影響が大きくなってしまっていた。

また、Packet salvaging プロセスにおいて、ドラフトに書かれていない追加機能が記述されていた。これは、パケットを salvage する際に宛先への経路が見つからないと Send バッファにバッファリングし、Route Discovery を行なう、というものであった。これも上記のバックオフアルゴリズムによって管理され、また、Discovery 回数も１度までと制限されているために、宛先の見つからないパケットは長時間バッファリングされ、一度ルートが見つかると、バッファリングされていたパケットをすべて一気に送信するなどしていた。

これらの点を修正した。

４．３　シミュレーション評価

２つの提案手法(Packet salvaging のループ回避・ディスジョイントなルート作成)を実装し、既存の DSR と提案手法それぞれについてランダムトポロジーを用いたシミュレーションを行い、遅延をはじめとしてパケット到着率、ルーティングオーバーヘッド等を評価していく。

４．３．１　シミュレーション条件

ランダムに移動させるときのシミュレーション条件を表４．３．１．に示す。この条件でシミュレーションを行い、５回の試行の平均値を結果として取るものとする。

シミュレーション時間	300(sec)
トランスポートプロトコル	UDP
MAC プロトコル	IEEE802.11
パケットサイズ	512(byte)
送信レート	4(packets/sec)
ノード数	100
通信ノード数	10 組
移動速度	0-20(m/s)
移動範囲	2200(m)×600(m)

表４．３．１．　シミュレーション条件

シミュレーションにおけるモビリティモデルでは、ノード数を 100 とし 2200(m)×600(m)の範囲のフィールド内にノードをランダムに配置する。各ノードはランダムに決定された 0-20(m/s)の速度でさまざまな方向に向かって移動し、到着後に一定の待ち時間(Pause Time)を待った後、また移動する。Pause Time は 0,60,120,240,300(sec)を用意する。Pause Time が短いほど、トポロジの変化が大きく、ハイモビリティのシナリオといえる。

４．３．２　シミュレーション結果と考察１

本節では４．３節において条件を定めた 100 ノードランダムシミュレーションにおける結果を示し、それに対する考察を行なう。

４．３．２．１　シミュレーション結果

(1)平均遅延時間の比較

宛先ノードが受信したデータパケットの平均遅延時間を図４．３．１に示す。

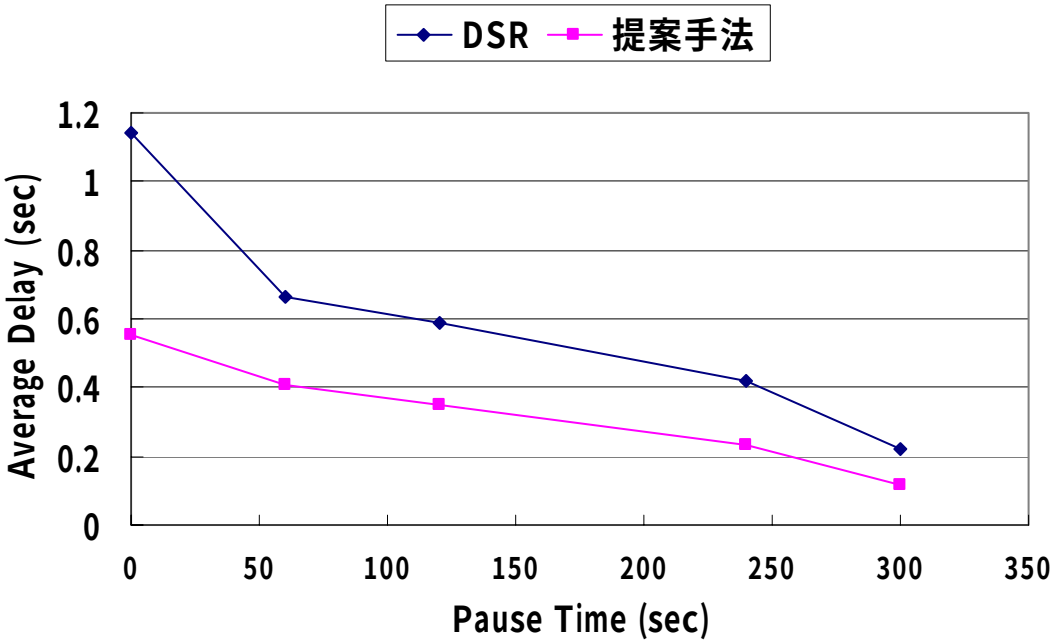


図４．３．１　平均遅延時間

(2)パケット到着率の比較

パケット到着率を図４．３．２に示す。パケット到着率は[受信データパケット数／送信データパケット数]で表される。

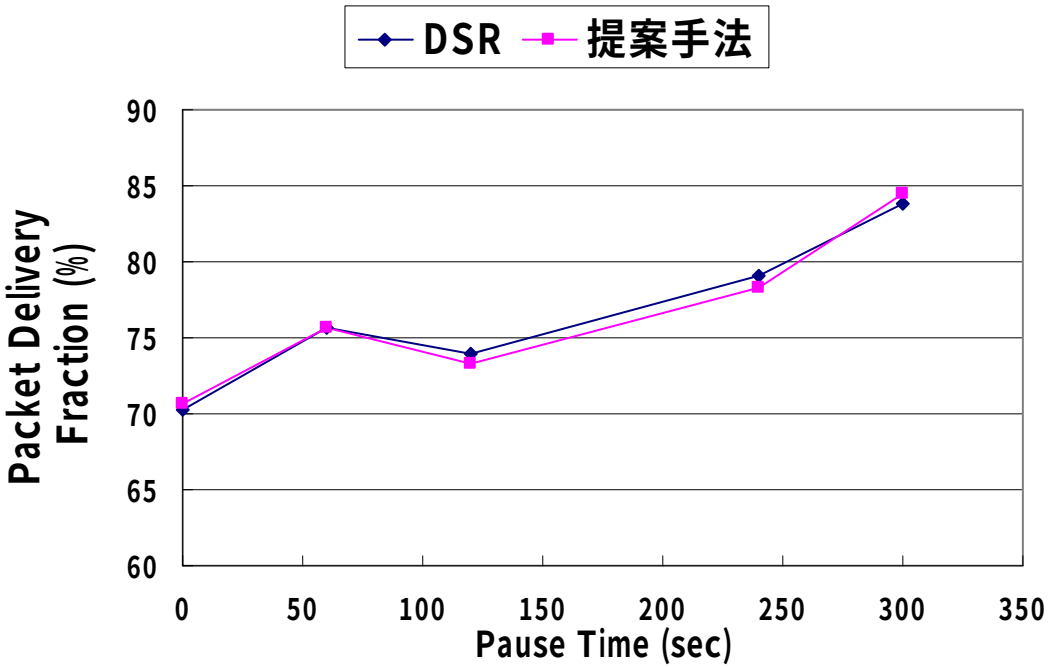


図４．３．２　パケット到着率

(3)経路探索回数の比較

Route Discovery プロセスが行なわれた回数／秒を以下の図４．３．３に示す。

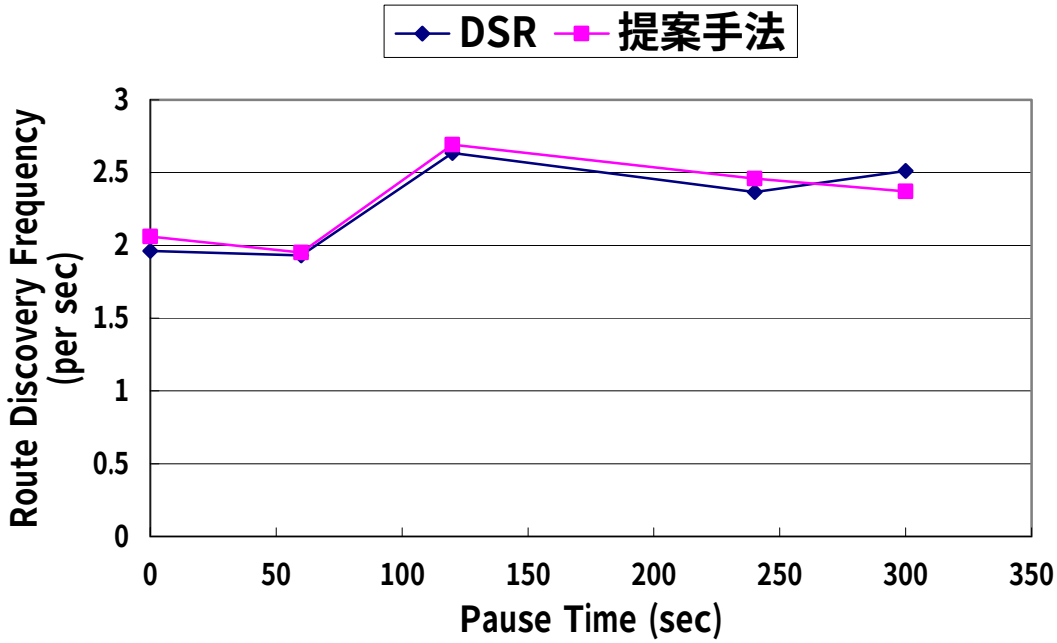


図４．３．３　経路探索回数

(4)平均ホップ数の比較

正常に受信されたデータパケットの平均ホップ数を図４．３．４に示す。

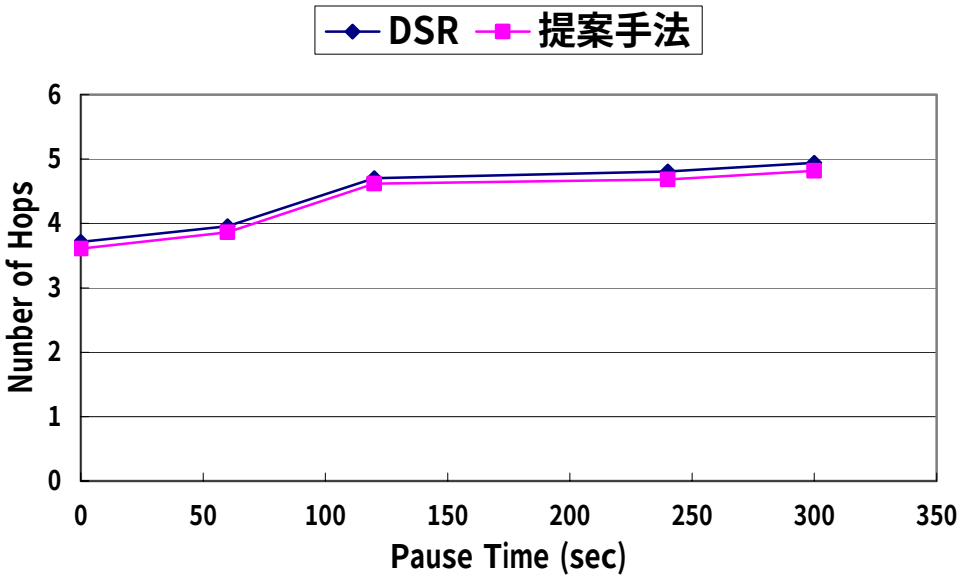


図４．３．４　平均ホップ数

(5)ルーティングパケット数の比較

ルーティング負荷の尺度として、RREQ・RREP・RERR のルーティングパケット数を比較し、図４．３．５～７に示す。

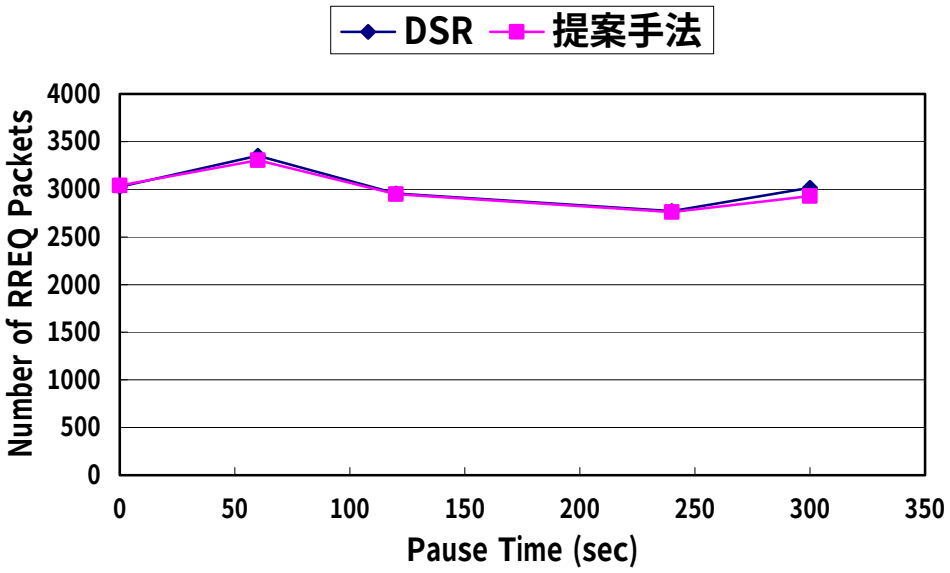


図４．３．５　RREQ 数

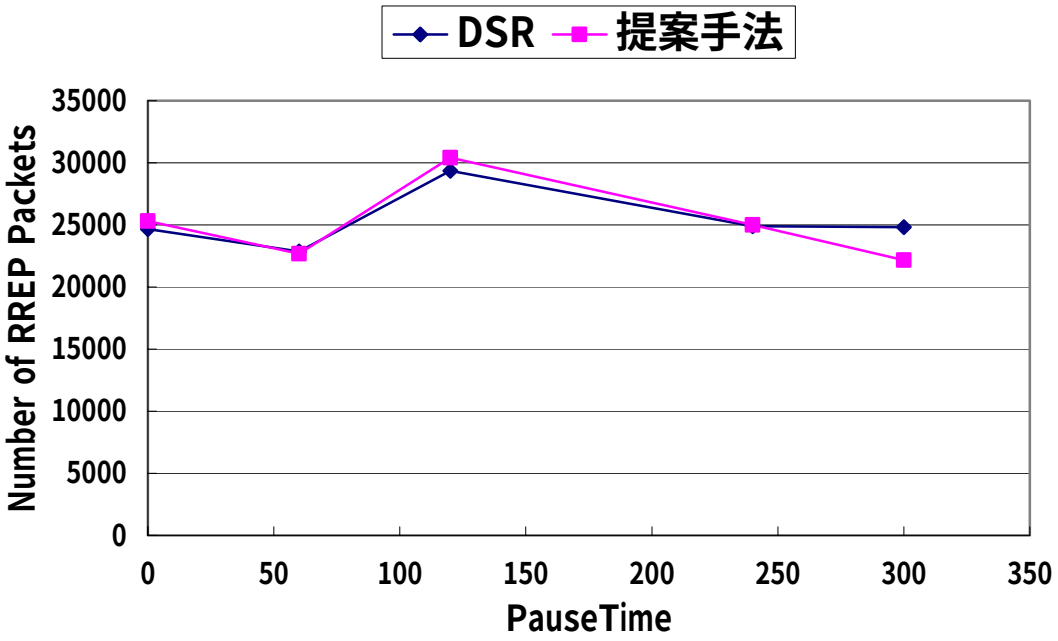


図４．３．６　RREP 数

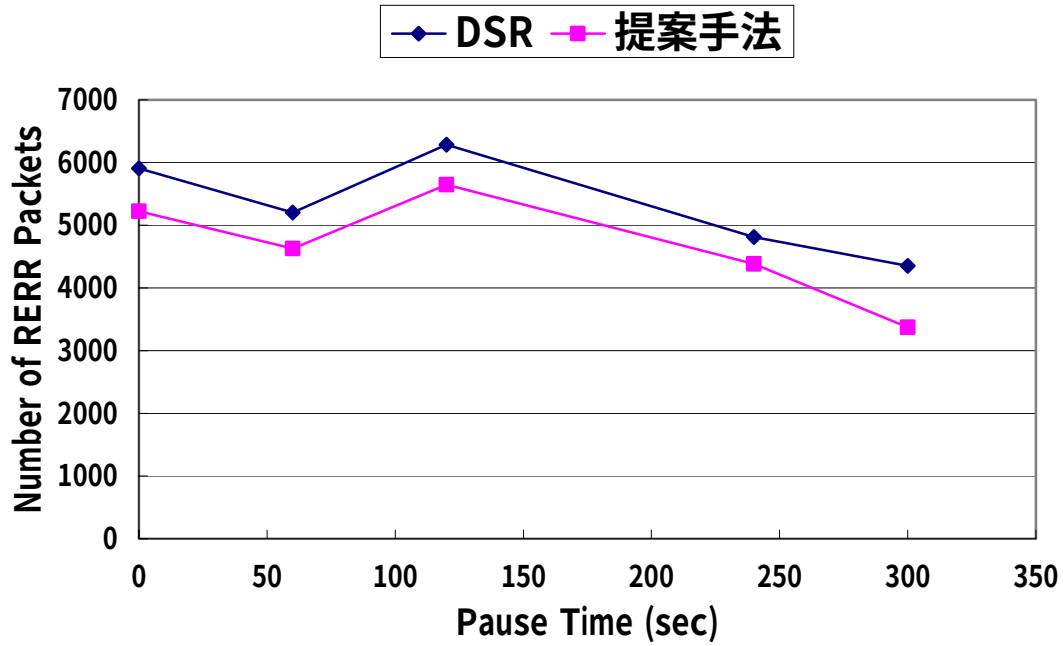


図４．３．７　RERR 数

４．３．２．２　考察

(1)平均遅延時間について(図４．３．１)

遅延の要素としては、Route Discovery 時の待ち時間、MAC 層での RTS/CTS 交換による再送遅延、伝送遅延が挙げられる。

送信元ノードと中間ノードでの平均待ち遅延時間を求めると図４．３．８・９のようになり、中間ノードでの遅延が提案時には極端に少なくなっていることがわかる。これは Packet salvaging によるループ遅延の回避と二経路へ振り分ける負荷分散の効果が現れているものだと考えられる。

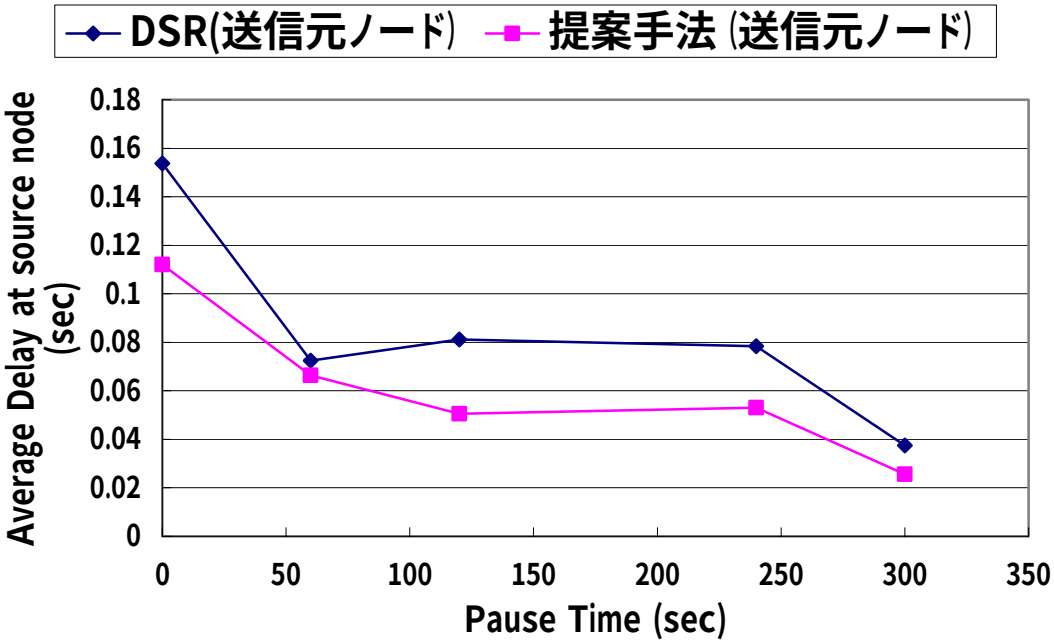


図４．３．８　送信元ノードにおける平均待ち遅延時間

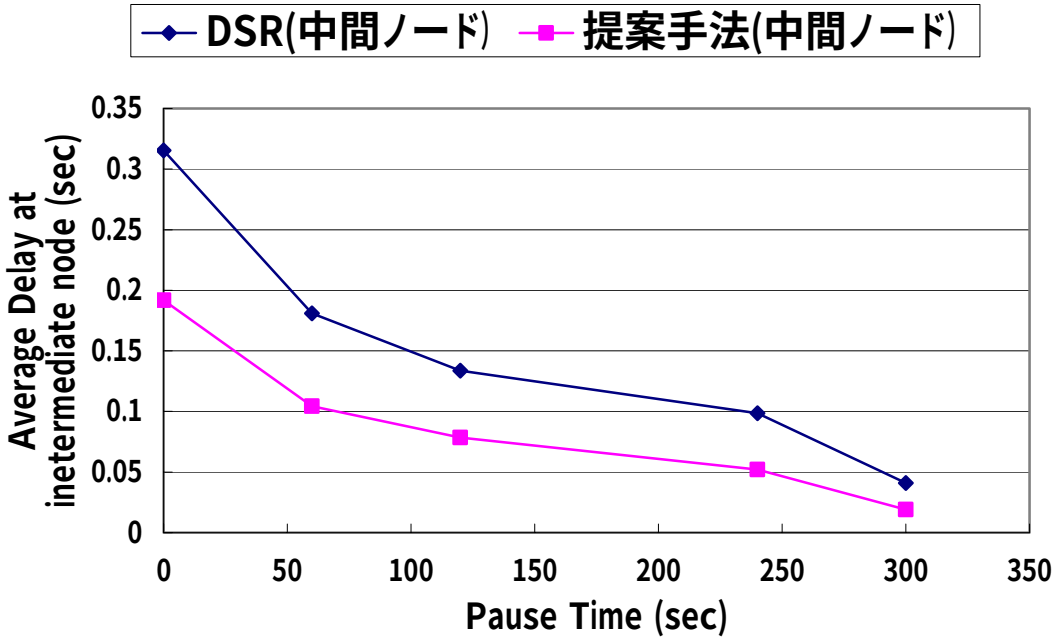


図４．３．９　中間ノードにおける平均待ち遅延時間

(2)パケット到着率について(図４．３．２)

提案・DSR ともにはっきりとした違いは見て取れない。これは、送信元と宛先のエンド間でディスジョイントな経路を複数作成しても、中間ノードでドロップするパケットを salvage できるような経路を作成するなんらかの明示的な機能がなければ、到着率をあげることはできない。

(3)経路探索回数について(図４．３．３)

経路探索回数は若干提案手法の方が多くなっている。これは宛先ノードが RREP を返す際に２個まで、と制限しているために送信元ノードのキャッシュにおける保持ルート数が少なくなるために、経路探索数が増えるからである。ただ、２．１．６でも述べたように、古くなった経路のキャッシュは遅延の原因となる。

(4)平均ホップ数について(図４．３．４)

平均ホップ数は若干提案手法の方が少なくなっている。Packet salvaging によってループが発生した経路を提案手法では利用しないために、到着パケットの平均ホップ数は短くなっている。提案手法で、ディスジョイント経路を利用する上で、ディスジョイントな２つの経路が必ずしも同ホップである必要はない(２．２．６(b)と同様)としているので実際の通信ホップ数は増加していると考えられるが、それでも提案手法の方が短くなっているこ

とがわかる。

(5)ルーティングパケット数について(図４．３．５～７)

RREQ はほぼ同数で推移している。RREP は、提案手法の経路探索回数が多いために Pause Time が短いハイモビリティなシナリオでは数が増えている。Pause Time が 300sec の時は、最初に宛先ノードが返した RREP による経路の生存時間が長くなるために、経路探索回数と RREP の減少が見られると考えられる。RERR 数は提案手法が軒並み少なくなっている。これは保持経路数が少なく、時間経過による無効な経路を多く保持しないということが原因となっている。RREP のようなユニキャストルーティングパケットの減少は２．４で述べたさらされ端末問題の原因ともなる RTS パケットの減少を促す。

４．３．３　シミュレーション結果と考察２

本節では DSR、提案ともにルートキャッシュからの応答機能、Packet salvaging などの機能を除き、送信元－宛先間でのソースルート運用のみした場合の遅延に焦点を絞ってシミュレーションをし、評価する。シミュレーション条件は４．３．１の通り。

４．３．３．１　シミュレーション結果

(1)平均遅延時間

エンド間の平均遅延時間を図４．３．１０に示す。

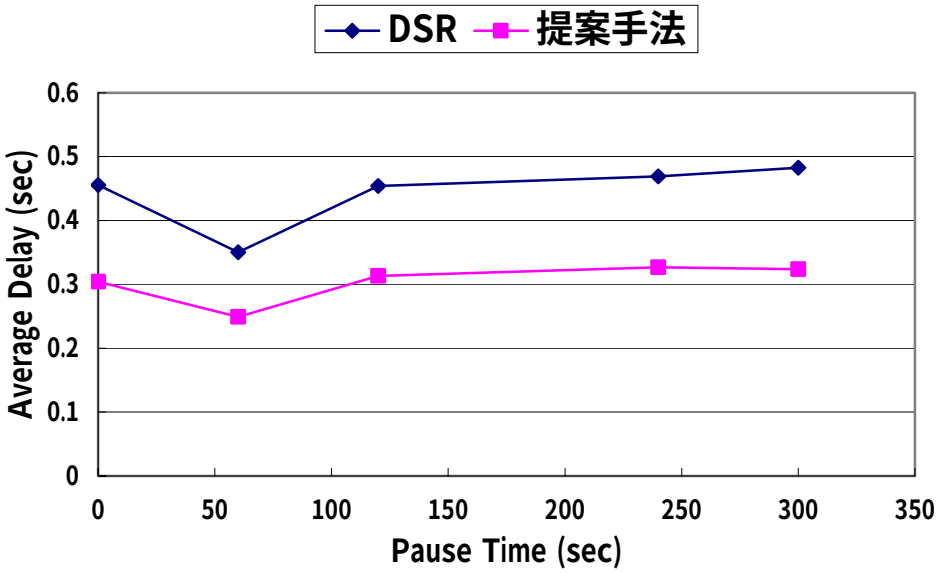


図４．３．１０　平均遅延時間

(2)送信元ノード、中間ノードにおける遅延時間

各ノードにおける平均待ち遅延時間を図４．３．１１・１２に示す。

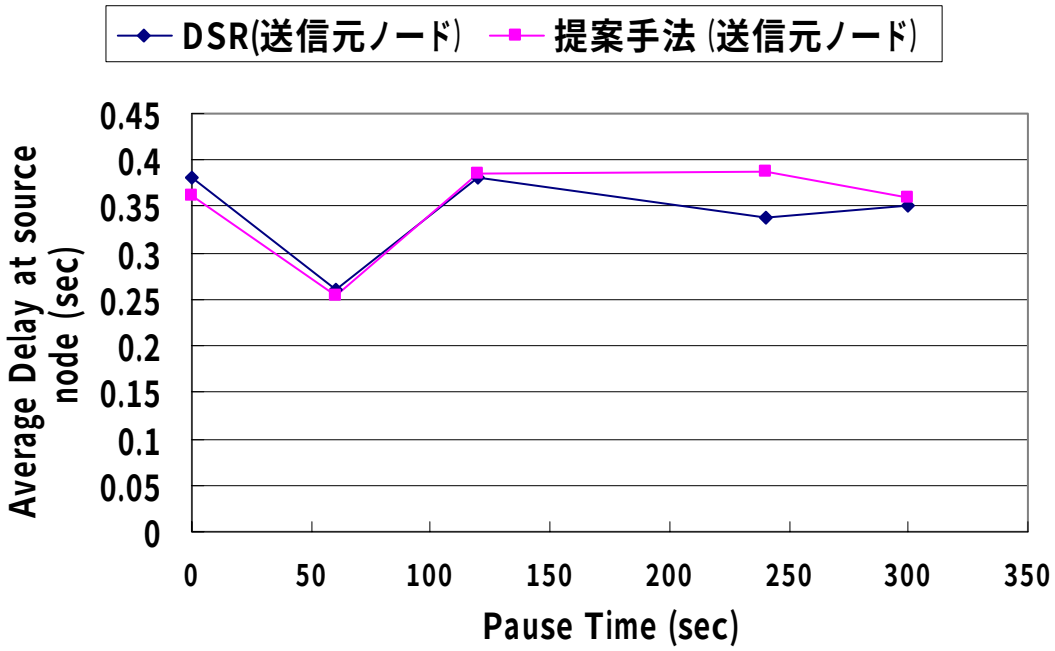


図４．３．１１　送信元ノードにおける平均送信待ち遅延時間

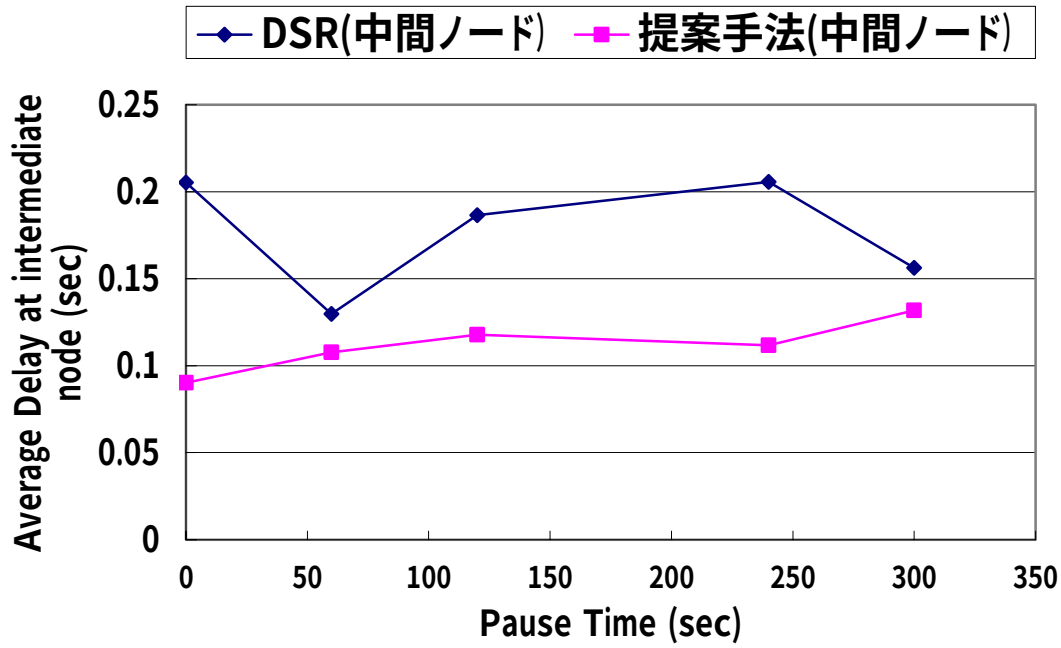


図４．３．１２　中間ノードにおける平均送信待ち遅延時間

４．３．３．２　考察

(1)エンド間平均遅延時間

４．３．２．１(1)とは違い、右肩下がりにならない。これは端末の移動が固定的になるほど部分的にアクセスが集中することになり、MAC 層での再送タイムアウトや IP 層での輻輳を招くことによる。ハンドオフが起こって経路が頻繁に変われば、トラヒックが分離され、輻輳も減るが、今度はハンドオフによるリンクブレークが増える。４．３．２．１では、中間ノードのキャッシュ応答を使うことで宛先へ経路探索が容易になって再接続時間が減少し、また、Packet salvaging によりモビリティの多いときには salvage するパケットに対して遅延が発生し、モビリティの少ないときには遅延がほとんどなくなる、といったことから、右肩下がりのグラフになったと考える。

提案手法において DSR より提案手法のほうが遅延が少なくなっている原因は、負荷分散と、経路保持数の少なさによると考える。以下(2)で述べる。

(2)各ノードにおける平均送信待ち遅延時間

図４．３．１１によるとローモビリティの際には DSR より提案手法のほうが遅延が減り、ハイモビリティの際には遅延が増えている。これはルートの保持数に起因し、本提案ではルートを 2 本しか保持しないのに対し、DSR ではルートをたくさんキャッシュするためにハイモビリティの際にはキャッシュしたルートが無効なことが多く、ローモビリティの際にはルートが有効なことが多いことから、この結果が出るのだと思われる。

次に図４．３．１１によると中間ノードにおける遅延は DSR よりも少なくなっている。これはディスジョイントな経路による負荷の分散が効果を発揮しているものと考える。

第５章　まとめ

５．１　総括

第１章では、研究の背景、目的および本論文の構成について述べた。

第２章では、今回の提案を進めるにあたり、基礎的知識として必要になる DSR について述べ、また SMR を通じてマルチパスルーティングにおけるディスジョイントの経路について述べるとともに、それらのルーティングプロトコルの問題点について指摘した。

第３章では、提案手法の目的と手順を述べた。

第４章では、ns を用いて、提案手法の有効性の評価を行なった。その結果、提案手法の目的としてあげた、Packet salvaging のループ回避、DSR と同等のルーティング負荷によるディスジョイント経路の作成により遅延を減少させることができた。

第５章である本章では、論文の総括および今後の課題について述べる。

５．２　今後の課題

ソースルート情報によるエンド間のディスジョイント経路の作成法を考えてきたが、この概念を使って、Hop by Hop の AODV といったプロトコルに適用するとどうなるかといったことを試していきたい。

また、今回の提案の実験を進めるに当たって、中間ノードにおける予備経路の作成がパケットの到着率に大きく影響することが分かってきたので、その経路の作成について考えたい。また、アドホックネットワークではルーティングプロトコル以外にも TCP やリンク層に関連してさまざまな問題点があるので、それらに関する提案もしていきたい。

參考文獻

[1]D.Johnson,D.Maltz,Y.Hu, ”The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks(DSR)”,Internet-Draft,April 2003

[2]E.Royer,C.Toh, ” A Current Routing Protocols for Ad Hoc Mobile Wireless Networks”,IEEE Personal Communications,1999

[3]C.Perkins,E.Royer,S.Das, ”Performance Comparison of Two On-Demand Routing Protocols”, IEEE Personal Communications,2001

[4] Sung-Ju Lee and Mario Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks", IEEE ICC 2001

[5]H.Lim,K.Xu,M.Gerla, ”TCP Performance over Multipath Routing in Mobile Ad Hoc Networks”, IEEE ICC 2003

[6]A.Nasipuri,R.Castaneda,S.R.Das,”Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks”,Mobile Networks and Applications, vol6, 2001

[7]S.Xu,T.Saadawi, ”Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks？”, IEEE Communications Magazine,2001

[8]The Networks Simulator - ns-2. available at http://www.isi.edu/nsnam/ns/

謝辞

本論文作成にあたり、丁寧なご指導と多くのアドバイスを頂きました甲藤二郎助教授に心から感謝し、御礼を申し上げたいと思います。

また、研究を進めるにあたり、助言や課題を頂いたりネットワーク班の諸先輩方、特に、新宿ラムダックスビルで苦楽をともにしたりした高宗俊輔氏、櫻井祐介氏の mobile 班両先輩に深く感謝いたします。

さらに、ネットワーク班で切磋琢磨した岡田陽平氏、竹部郁夫氏、三好玄氏をはじめ、同学年の大切な仲間である B４の皆様に感謝します。

そして、いつも和気藹々とした雰囲気を作ってくださっていた甲藤研究室の先輩方にお礼を申し上げたいと思います。

最後に、不摂生な生活になりがちな自分の事を心から心配してくれた両親に感謝します。

ありがとうございました。

２００４年2月9日

谷山　健太