2007－DPS－131 (10)
2007／6／7

# アドホックネットワークにおける中継遅延の削減方式

瀧本 栄二[†], 滝沢 泰久[†], 鈴木 龍太郎[†], 小花 貞夫[†]

アドホックネットワークは，ノード間の自律的な通信により，柔軟なネットワークの構築が可能である．そのため，様々な分野での適用が期待されている．アドホックネットワークにおける通信性能の評価指針の1つに，応答性がある．しかし，従来のアドホックネットワークに関する研究では，応答性が重視されていない．例えば，アドホックネットワークの適用アプリケーションである車車間通信等では，高速な応答性が求められる．そこで，本稿では，アドホックネットワークにおける通信方式の1つであるフラッディングに関して，応答性を向上させる手法を提案する．また，提案手法のプロトタイプシステムを用いた性能評価を行った結果についても述べる．

# A Method of Decreasing Delay of Relay in Ad-hoc Networks

Eiji Takimoto[†], Yasuhisa Takizawa[†], Ryutaro Suzuki[†], Sadao Obana[†]

Ad-hoc network can construct flexible networks by a autonomous communications among nodes. Therefore, ad-hoc network is expected to apply in the number of different fields. The responsiveness is one of important issues in ad-hoc network. In the conventional researches, however, it is not enough to consider the responsiveness. For example, the high response is required in the inter-vehicle communication which is one of ad-hoc network applications. Therefore, in this paper, we propose the method for enhancing the responsiveness in the flooding which is a communication method in ad-hoc network. We also describe results of evaluations with a prototype system of the proposed method.

## 1 はじめに

アドホックネットワークは，ネットワークを構成するすべての通信ノードが送信，受信，中継を行う機能を持つため，柔軟かつ動的にネットワークを構築することが可能である．アドホックネットワークにおける通信形態の1つに，フラッディングがある．フラッディングは，アドホックネットワークにおけるブロードキャスト通信であり，ネットワーク全体にパケットを送信するために用いられる．フラッディングでは，あるノードがパケットを送信すると，それを受信したノードがさらに中継を行うことで，ネットワーク全体にパケットが到達する．

アドホックネットワークを用いたアプリケーションの1つに，ITS(高度交通情報システム)の車車間通信 [1] がある．車車間通信では，安全運転支援を目的とした車両情報の配布と収集が行われる．また，対向車の存在や急制動の発生を知らせるための危険情報通知も行われる．これらの通信は，特定の車両間で行われるユニキャスト通信ではなく，周辺一帯に存在する車両全体を対象としたブロードキャスト，すなわちフラッディングである．また，危険情報を速やかに通知する必要があるため，その通信には高い応答性が求められる．

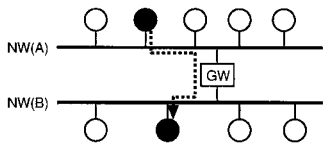


図1 有線ネットワークにおける中継

フラッディングの応答性を向上させるためには，通信時に発生する遅延を削減することが考えられる．マルチホップ通信では，複数のノードによってパケットが中継されるため，送信ノードから離れるほど中継回数が増え，結果として遅延が増加する．したがって，中継による遅延 (以下，中継遅延と記す) を削減することで，ネットワーク全体の通信時間を削減することが可能になり，ひいては高速な応答性の実現につながると考えられる．

中継遅延は，デバイス制御，プロトコル処理，中継処理といったソフトウェアによるものとMAC制御，電波送受信といったハードウェアによるものとに分類することができる．これらの性質は大きく異なるため，一元的に扱うことができない．本稿では，ソフトウェアに起因する中継遅延を対象とする．

有線ネットワークでも，ルータやゲートウェイ等

[†](株)ATR 適応コミュニケーション研究所
Advanced Telecommunications Research Institute International, Adaptive Communication Research Laboratories

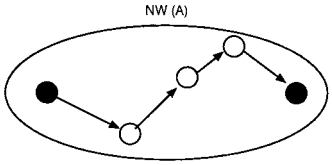


図 2　アドホックネットワークにおける中継

において中継処理が行われる．図 1 に示すように，有線ネットワークで行う中継処理の目的は，ネットワーク間接続である．したがって，図 1 のゲートウェイのように，中継ノードはあらかじめ異なるネットワークに接続された通信デバイスを複数持つ．中継時には，入力と出力がそれぞれ異なる通信デバイスで行われる．

一方，アドホックネットワークにおける中継処理は，図 2 に示すように 1 ホップで通信できないノード間を補完するためのものであり，同一ネットワーク内に対して行われる．したがって，中継ノードが必要とする通信デバイスとそのデバイスドライバは 1 つである．

そこで，我々は，これらの有線ネットワークとアドホックネットワークにおける中継処理と特性の違いに着目し，中継機能をデバイスドライバに付与することにより高速中継を実現する手法を提案する．提案手法では，フラッディングパケットの中継処理を割込み処理時に行う．これにより，従来手法では必要となるプロトコルスタックによる処理と，割込み処理時に行われる遅延処理により発生する遅延時間を削減する．さらに，通信アプリケーションとデバイスドライバの間で直接パケットを授受する機構を設けることにより，中継だけでなく送受信時におけるプロトコルスタック処理も削減している．

以下，本稿では，2 章で従来のフラッディング実装方式とその遅延要因について述べ，3 章で提案手法とそのプロトタイプ実装について述べ，4 章で提案手法の性能評価を行った結果とその考察を述べたあと，5 章で本稿をまとめる．

## 2　従来の実装方式と遅延要因

フラッディングの応答性を向上させるためには，送受信と中継において生じる遅延を削減する必要がある．遅延の要因は様々であり，それらはシステム構成によって異なる．本稿では，プロトタイプシステムを以下のように構成した．

・計算機：PC/AT 互換機 (CPU:PentiumD 3GHz)
・通信デバイス：IEEE802.11b デバイス
・OS：CentOS Linux(kernel2.6.9)
・デバイスドライバ：linux-wlan-ng0.2.3

Linux は，オープンソースであることと，サーバから組込み用途まで多様な使用実績があることから採用した．通信デバイスには，一般的であり入手も容易であることから Prism2.5 チップを用いた IEEE802.11b デバイスを採用した．デバイスドライバは，Linux の Prism2.5 チップ用ドライバである linux-wlan-ng を用いた．フラッディングのアルゴリズムには，実装が容易なピュアフラッディングを採用した．

### 2.1　実装方式

中継処理は，デーモンや受信するアプリケーション自身で行う方式と，カーネル内で処理する方式のどちらかで実装されている．これらの方式は，実装される階層が異なるため，生じる中継遅延の大きさとその要因が異なる．

カーネル内でフラッディングを実現している代表的な実装として，Kernel-AODV[2] がある．Kernel-AODV では，経路要求パケットを UDP でフラッディングしている．パケットの中継は，IP 層で行われる．中継処理を行うためにプロトコルスタックを変更することは，開発コストを上げ，汎用性を低下させる原因になる．そのため，Kernel-AODV では，中継機能を LKM(Loadable Kernel Module) として実装し，Netfileter[4] を用いてプロトコルスタックとは独立した処理を実現している．この方式では，受信したパケットは IP 層で Netfilter によってフックされ，LKM に渡される．中継が行われる場合は，LKM 内で中継処理が行われ，IP 層を経由して送信処理が行われる．同時に，受信パケットは，上位の UDP 層へ渡され最終的にアプリケーションが受信する (図 3 参照)．

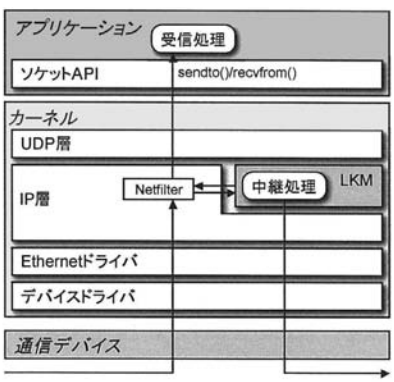


図 3　カーネル内実装方式

より簡易な方式は，アプリケーションで実装することである．この方式では，デーモンのようにフラッディング専用アプリケーションとして実装する方法と，パケットを送受信するアプリケーション自身がフラッディングの制御を行う方式が考えられる．後者は，送受信と中継をアプリケーション内で閉じて行うことができるため，より実装が容易である．

### 2.2 遅延要因

プロトタイプシステムと同じ構成の通信システムにおいて，提案手法を用いない場合に発生する遅延要因を調査した．以下，本項では，既存システムにおける遅延要因とその性格について述べる．

#### 2.2.1 カーネルによる遅延処理

Linux では，カーネルが行う処理を効率的に行うために，通信データの処理を通信イベントから遅延して行う (以降，この遅延を挟むイベント処理を遅延処理と記す)．遅延処理は，要求された処理を即座に行わずカーネル内のキューにためておき，システムコールやタイマ割込みをきっかけに，キュー内のタスクをまとめて処理する方式である．遅延処理自体は，一般的に用いられている手法である．linux-wlan-ng デバイスドライバでは，受信割込みが発生すると次のような流れで処理が行われる．

手順 1. 割込み原因を判別し，受信割込みであれば通信デバイスの受信バッファからデータを読み出し，sk_buff 構造体と呼ばれるパケット管理用構造体に記録する．ここで，遅延処理用機構である tasklet に登録されている受信処理用エントリをアクティブにして割込み処理を終了する．

手順 2 システムコールやタイマ割込みを契機に tasklet により処理を再開する．ここでは，受信パケットの種別 (ARP，IP，ICMP 等) の判別やエラーチェック等を行い，上位層へ処理を引き渡す．

遅延処理によって生じる遅延は，デバイスドライバで tasklet を使用するため，デバイスドライバによって解決する必要がある．

#### 2.2.2 アプリケーション内実装のみの遅延

アプリケーションによるフラッディングでは，中継処理をアプリケーション自身が行う．中継時には，図 4 に示すように，カーネルとアプリケーションの間でコンテキストスイッチが発生する．また，アプリケーションでパケットを扱うためには，アプリケーションとカーネルの間でパケットをコピーする必要がある．したがって，アプリケーションで実装する場合，カーネル内で実装した場合と比べ，コンテキストスイッチとメモリコピーによる遅延が増えることになる．また，カーネル内実装とは異なり，中継パケットが UDP 層を往復することも遅延となると考えられる．

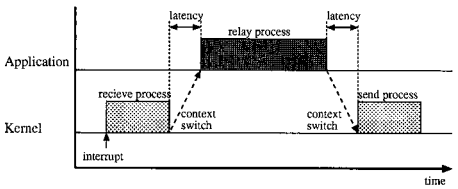


図 4 中継時のコンテキストスイッチ

## 3 提案手法とプロトタイプの実装

高速なフラッディングを実現するためには，前章で述べた遅延処理，アプリケーション特有の要因を削減することが必要である．アプリケーション特有の遅延は，カーネル内に実装することで回避することができる．したがって，遅延処理による遅延を削減することが，重要な課題である．この課題を解決するために，本稿では，次の 2 点を戦略とする手法を提案する．

- アプリケーションの送受信を直接デバイスドライバが処理する．
- 中継処理をデバイスドライバ内で行う．

すなわち，提案手法では，前者の戦略によりフラッディングに関わるすべての処理において遅延処理を行うことなく割込み発生時に即座にデバイスドライバで行い，かつ中継時におけるプロトコルスタック処理を削減する．また，後者により送受信時に行われるプロトコル処理をショートカットすることで，遅延処理以外の遅延要因を削除し高速化を図る．

中継処理の高速化は，有線ネットワークでも行われている．しかし，有線ネットワークでは，中継を行う通信ノードがルータやゲートウェイといった一部のノードに限られている．これは，有線ネットワークにおける中継処理の目的がネットワーク間接続のためである．そのため，中継ノードには，それぞれ異なるネットワークに接続された通信デバイスがあり，ある通信デバイスで受信したパケットは別の通信デバイスを用いて送信される (図 5 参照)．これに対して，アドホックネットワークでは，通信ノード

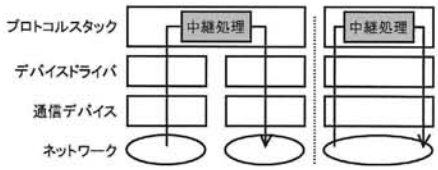


図 5　有線・無線ネットワークの違い

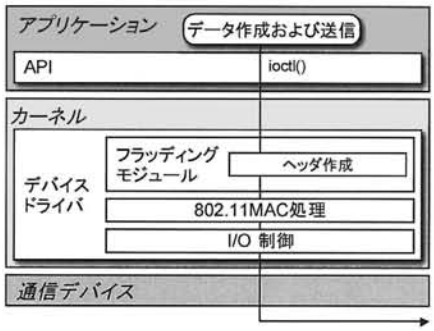


図 6　プロトタイプの構成 (送信)

は 1 つの通信デバイスのみを持つノード構造が一般的であり，すべてのノードが中継機能を持つ必要がある．また，中継処理は，同一ネットワークに対して行われる．

有線ネットワークでは，通信デバイスを複数持つため，中継処理をデバイスドライバ内で閉じて行うことが困難であった．これに対して，アドホックネットワークでは，単一デバイスで中継処理を行うため，中継時に行われるプロトコルスタックの処理や遅延処理による遅延を削減するために，割込み処理時において中継処理を行うことが可能となる．以下，本章では，提案手法の詳細について述べる．

### 3.1　送受信処理

提案手法では，アプリケーションがデバイスドライバに対して直接送受信を要求する．プロトタイプでは，実装を簡素化するため，ioctl システムコールを改良した．具体的には，ioctl システムコールで扱うことができる要求に，フラッディングパケットの送信と受信を加えた．これにより，UDP/IP 処理による遅延を削減した．さらに，ソケット API を使用することなく通信を行うことで，より高速化を実現した (図 6 参照)．

パケットの冗長性を判断するために，図 7 のフラッディング用ヘッダを用いた．このヘッダは，送信時にデバイスドライバによって作成される．プロトタイプでは，ユーザが送受信するデータにもヘッダが含まれるものとした．ヘッダの詳細は，以下に示す通りである．

| ID (8) | SEQ (8) | TTL (8) | LEN (16) | DATA (0-2307Bytes) |
|---|---|---|---|---|

ID(8bits):Host Identify Number
SEQ(8bits):Sequential Number
TTL(8bits):Time To Live
LEN(16bits):Data Length(Bytes)

図 7　フラッディング用ヘッダ

### 3.2　中継処理

プロトタイプでは，フラッディングアルゴリズムにピュアフラッディングを採用した．ピュアフラッディングでは，冗長パケットを中継しないように，受信パケットをキャッシュする．

プロトタイプの構成を図 8 に示す．提案手法では，中継をデバイスドライバが行うため，デバイスドライバ内にキャッシュを作成した．プロトタイプでは，ioctl システムコールによって受信が行われる．そのため，キャッシュと同様に，受信用のキューをデバイスドライバ内に追加した．

## 4　性能評価

提案手法に基づき，実機を用いてプロトタイプシステムを作成し，その性能評価を行った．実装対象となる計算機の性能は，2 章で述べた通りである．通信速度は，IEEE802.11b のブロードキャスト通信であるため 1Mbps となる．また，従来手法であるアプリケーションによる方式とカーネルの IP 層による方式を実装し，それらを比較対象とした．以下，本稿では，アプリケーションによる方式とカーネルによる方式をそれぞれ従来手法 1，従来手法 2

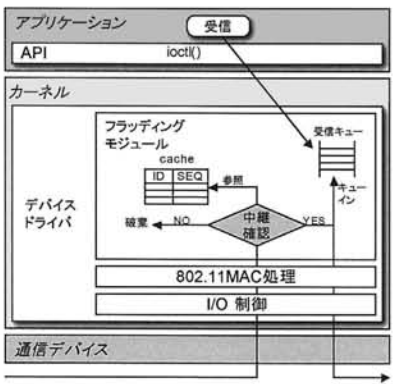


図 8　プロトタイプの構成 (受信と中継)

と記す．

性能評価は，図 9 に示す構成のネットワークを構築し，中継ノード数を必要に応じて変更しながら行った．測定値に関しては，正確な時刻同期が困難であることと電波伝搬の影響を考慮し，10000 回通信を往復して行い，送信ノードにおけるターンアラウンドタイムを 2 で割った値の平均値を採用した．また，パケットサイズは，車車間通信等を想定し，100 バイトとした．

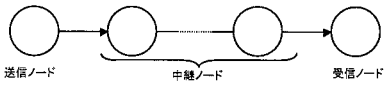


図 9　評価用トポロジ

### 4.1　中継遅延とその詳細

中継遅延は，中継ノードが存在しない 1 ホップ通信と 1 つの中継ノードが存在する 2 ホップ通信の通信時間の差で求めることができる．各手法における通信時間と中継遅延は，図 10 に示す通りであった．図 10 から分かるように，中継処理を行う階層が低いほど中継遅延が小さいという結果が得られた．ただし，図 10 は，ハードウェア処理を含めた遅延時間であるため，ハードウェアによるチャネルアクセスの遅延や電波伝搬による遅延も含んでいる．

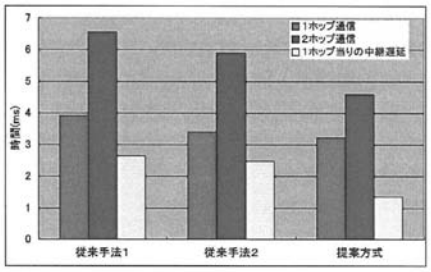


図 10　各手法の通信時間と中継遅延

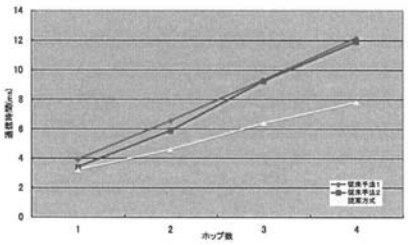


図 11　ホップ数と通信時間の関係

表 1　ソフトウェア処理による中継遅延

| | 従来手法 1 | 従来手法 2 | 提案手法 |
|---|---|---|---|
| 中継遅延 | $0.97ms$ | $0.95ms$ | $0.35ms$ |

図 9 のホップ数を 1〜4 ホップまで増加させた場合の通信時間を計測した結果は，図 11 の通りとなった．トポロジが直線であるため，ホップ数と通信時間は，ほぼ線形に増加している．したがって，提案手法を用いた場合の通信時間は，中継遅延を削減しているため，ホップ数が増加するほどその効果が大きくなることがわかる．

中継ノードにおける中継遅延のうち，ソフトウェア処理における遅延時間は，表 1 の通りであった．この測定では，デバイスドライバの受信割込みハンドラの入口と送信命令書込み直前の 2 箇所において CPU のクロック数を取得し，その差分を用いた．測定の結果，2 つの従来手法の差は，$0.02ms$ と非常に小さかった．しかし，提案手法は，従来手法に比較して約 $0.6ms$ も小さい遅延時間で処理できており，70%を越える改善率を示している．このことから，提案手法の効果が非常に高いといえる．

従来手法 1 と 2 との差は，UDP 層の処理とアプリケーションとカーネル間でのコンテキストスイッチの有無である．従来手法 2 と提案手法との差は，プロトコルスタックと遅延処理の有無である．図 12 に，表 1 の内訳を示す．従来手法 2 では，従来手法 1 に比べ UDP 層とアプリケーション処理がなくなっているが，中継処理が IP 層に移されているため IP 層での処理時間が増加している．提案手法は，遅延処理及び上位プロトコルによる処理をすべて省いているため，デバイスドライバによる遅延時間のみとなっている．また，中継処理をデバイスドライバ内で行うため，その分だけ従来手法と比べてデバイスドライバでの遅延時間が大きくなっている．ただし，従来手法 2 の IP 層における遅延時間の増加分より，提案手法の増加分の方が大きくなっている．これは，提案手法がアプリケーションと直接パケットをやり取りするためのキューを持っており，そのキュー操作によるものと考えられる．

### 4.2　送受信の高速化の効果

提案手法では，通信プロトロコルとして UDP/IP を用いずに通信することで，中継時だけでなく送受信の高速化を図っている．送信時と受信時のソフトウェアにおける処理時間は，表 2 に示す通りであった．

送信処理における効果は小さいが，受信処理にお

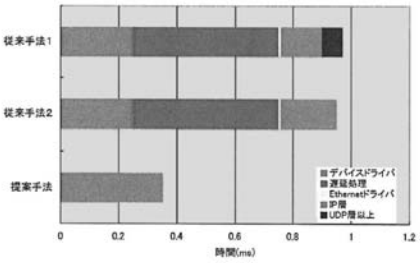


図 12　中継遅延の内訳

表 2　送信時と受信時の処理時間

| | 従来手法 1 | 従来手法 2 | 提案手法 |
|---|---|---|---|
| 送信時 | $0.23ms$ | $0.23ms$ | $0.20ms$ |
| 受信時 | $1.82ms$ | $1.81ms$ | $1.40ms$ |
| 合計 | $2.05ms$ | $2.04ms$ | $1.60ms$ |

ける効果が大きかった．これは，中継処理と同様に，受信時の遅延処理の有無による影響であった．ただし，通信時間においてプロトコルスタック処理が占める割合が図 12 同様小さいため，プロトコルスタックを省いたことによる効果はあまり大きくなかった．

### 4.3　考察

以上の実験と結果から，提案手法によりフラッディングを約 70% 高速化することが可能であることが確認できた．中継遅延の内訳を解析した結果，プロトコルスタックによる処理遅延以上に，遅延処理による遅延が大きな割合を占めていることがわかった．このため，提案手法は，中継処理を実装する階層を下げたことよりも，遅延処理を削減することで高い効果を得ているといえる．

遅延処理を行わない提案手法は，システム全体の性能を低下させる可能性がある．したがって，提案手法を用いる際には，システム構成等を考慮する必要がある．ただし，提案手法は，車車間通信などといった特定用途を主な対象としている．このような用途では，単純なシステム構成を取ることが多いため，システム構成に応じたカスタマイズによりこの問題を回避することができると考えられる．また，遅延処理は，様々な OS に採用されている一般的な手法である．したがって，提案手法は，プロトタイプシステムで用いた Linux 以外でも有効であると考えられる．

今回のプロトタイプシステムでは，駆動クロック数が 3GHz と高速な CPU を用いているため，ソフトウェア処理に占める遅延処理以外の部分の割合が小さいものとなっている．例えば，組込みシステム向けの低速な CPU を用いた場合では，図 12 とは異なりソフトウェア処理に占める遅延処理以外の部分の割合が大きくなる．筆者らが今回使用したプロトタイプよりも性能の低い計算機を使用して行った実験 [3] では，削減された中継遅延のうち，遅延処理による遅延が約 55% に留まっており，それ以外のソフトウェア処理による遅延が大きくなっており今回とは異なる結果となっている．ただし，これは，遅延処理による遅延が小さくなったのではなく，その他の遅延が大きくなったためである．したがって，提案手法は，計算機性能に因らずに遅延処理分の遅延を削減し，それ以外の遅延に関しては計算機性能に依存して遅延時間を削減することが可能である．

## 5　おわりに

本稿では，デバイスドライバにおいてフラッディングを実現する手法の妥当性と，その評価について述べた．提案手法は，従来のカーネル内およびアプリケーション内で中継処理を行う方式に比べ，ソフトウェアが行う中継遅延を約 70%削減することが可能であり，アドホックネットワークの応答性能を向上させることが実証できた．

## 謝辞

本研究は，情報通信研究機構の研究依託「高レスポンスマルチホップ自律無線通信システムの研究開発」により実施した．

## 参考文献

[1] 関　馨:海外における車車間通信の開発-その現状と現状-，自動車研究，Vol. 27, No. 1, pp. 21–26, 日本自動車研究所 (2005).

[2] Chakeres, I. D. and Belding-Royer, E. M.: AODV Routing Protocol Implementation Design, *24th International Conference on Distributed Computing Systems Workshop*, pp. 698–703 (2004).

[3] 瀧本栄二，滝沢泰久，門脇直人，小花貞夫，大久保英嗣：アドホックネットワークにおける中継遅延の削減方式，*FIT2006*, L–075 (2006).

[4] netfilter/iptables project: URL: <http://www.netfilter.org/>.