Vol.2015-MBL-74 No.55
Vol.2015-UBI-45 No.55
2015/3/3

PiggyCode改善方式の提案と実機評価

野村 拓矢[1,a)]　明田 修平[1]　大月 勇人[1]　毛利 公一[1]　瀧本 栄二[1]

概要：無線マルチホップネットワークにおける TCP 通信は，DATA パケットと ACK パケットの双方向通信となるため，隠れ端末問題やフロー内干渉の影響が大きい．既存研究に，パケットの中継時に DATA パケットと ACK パケットに対しネットワークコーディングを適用し総送信回数を削減する PiggyCode がある．これまで，中継時に DATA パケットを待機させることで符号化率を向上させる手法を提案し，シミュレーション評価によってその有効性を検証してきた．本稿では，当該手法を実機実装し，実環境での評価を行った結果について述べる．さらに，実機実験から得られた知見に基づく性能改善方式を提案する．

キーワード：ネットワークコーディング，無線マルチホップネットワーク，TCP

A Study of Implementation of PiggyCode and an Improvement Method

Abstract: TCP communications on wireless multi-hop networks is bi-directional communications using DATA packets and ACK packets. These properties increase the hidden node problem and the flow within the interference. PiggyCode, which is an existing study focused on bi-directivitiy of DATA packets and ACK packets, reduces the number of transmissions by applying Network Coding to the DATA packets and ACK packets when relaying these packets. We had verifed the proposed method to improve coding rate by inserting wait time, and had proved on a simulator. In this paper, we prove the validity of the proposed method by the result of evaluations in real environment. Furthermore, we propose a performance improving method according to the findings obtained from real machine experiments.

Keywords: Network Coding, Wireless Multi-hop Networks, TCP

1. はじめに

無線デバイスや携帯端末の発展と普及に伴い，複数の無線端末で構成される無線マルチホップネットワークが注目されている．このネットワークの典型的な利用形態として，インターネットアクセスがある．インターネット上での通信の多くは，信頼性の高い TCP を使用する．TCP は，DATA パケットに対し ACK パケットを返送することで信頼性の高い通信を実現する．しかし，無線環境での ACK パケットの返信は，半二重リンクである無線チャネルの特性上，スループット低下の要因となる．

この課題に対して PiggyCode[1] は，TCP における DATA パケットと ACK パケットの双方向性に着目し，中継端末において各パケットにネットワークコーディング [2] を適用することで，中継時のパケット送信回数を削減する．これにより，TCP スループットや RTT を改善する．しかし，PiggyCode は，MAC 層のインタフェースキュー内で送信待ち状態にあるパケットを符号化対象としているため，輻輳に近い状態にならないと符号化の効果が得られない．この問題を解決する手法として，待機時間挿入手法 [3], [4] がある．待機時間挿入手法は，DATA パケット送信時に待機時間を挿入することで，符号化されるパケットの割合（以下，符号化率）を向上させる手法である．胡ら [3] は，待機時間挿入手法を PiggyCode へ適用し，その有効性をシミュレーション評価によって明らかにしている．本稿では，待機時間挿入手法を適用した PiggyCode を Delayed-PiggyCode（Insert Delay - PiggyCode）と表記する．

待機時間の挿入は，符号化率を向上させるが，待機時間内に符号化対象パケットが到着しない場合，挿入した待機時間だけ RTT が増大する．胡ら [3] は，この問題についても言及し，待機時間の挿入による RTT の増加を抑制する

[1] 立命館大学
Ritsumeikan University
[a)] tnomura@asl.cs.ritsumei.ac.jp

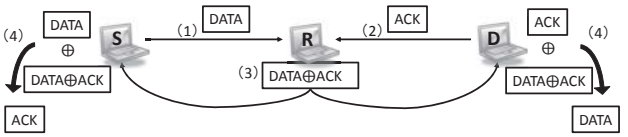

図 1 PiggyCode の基本動作

ため，環境に適応した待機時間の設定が必要であるとしている．しかし，実環境においては，MAC 層以下の挙動や TCP オプションの有無などにより，符号化に必要な待機時間は常に変動し，適応的に待機時間を挿入することは難しい．特に，一般に広く用いられている遅延 ACK オプション [5] は，返信する ACK パケット数を削減するオプションである．遅延 ACK オプション有効時は，DATA パケットと ACK パケットの個数が偏る．また，TCP の輻輳制御やトポロジ構成においても，一定時間に受信する TCP パケットの種別に偏りが生じる．

そこで，本稿では，キュー制御によってこの課題を解決する．提案するキュー制御方式は，DATA パケットを連続で受信する際に待機用キュー長を増加させ，ACK 受信時や待機中のパケットを送信する時にキュー長を減少させる．これにより，遅延 ACK オプションで発生する ACK パケット数の減少と，受信パケット種別の偏りに対応することができる．

以下，本稿では，2 章で PiggyCode および待機時間挿入手法について述べ，3 章で提案手法について述べる．4 章で実機実装についてを述べ，5 章で実環境における既存手法と提案手法との比較評価を述べる．最後に 6 章でまとめる．

2. PiggyCode と待機時間挿入手法

2.1 PiggyCode

2.1.1 PiggyCode の概要

PiggyCode[1] は，TCP 通信に Network Coding（NC）[2] を適用した研究である．NC は，送信および中継端末においてパケットを符号化し，受信端末において復号することでパケットの伝送効率を向上させる技術である．PiggyCode は，同一フロー内の DATA パケットと ACK パケットの双方向性に着目し，これらのパケットに対して符号化を行うことでパケット送信回数を削減する．パケット送信回数の削減により，スループットや RTT（Round Trip Time）を改善している．文献 [1] では，PiggyCode の実現のため，ネットワーク層と MAC 層との間にパケットの符号化および復号を行うための NC 層を追加している．NC 層は，符号化モジュール，復号モジュール，復号に使用するパケットを一時的に保存する復号バッファで構成されている．

PiggyCode の基本的な動作を図 1 に，説明を以下に示す．

(1) 送信端末 S は DATA パケットを送信し，複製を保持する．
(2) 受信端末 D は ACK パケットを送信し，複製を保持する．
(3) 中継端末 R は受信した DATA パケットと ACK パケットを用いて符号化を行い，DATA⊕ACK パケットを生成し，ブロードキャストまたはプロミスキャスモードを用いて送信端末 S と受信端末 D へ送信する．
(4) 送信端末 S と受信端末 D が受信した DATA⊕ACK パケットと送信時に保持していた DATA パケットか ACK パケットを用いて復号を行い，ACK パケット，DATA パケットをそれぞれ生成し，受信処理へ渡す．

2.1.2 PiggyCode の課題

PiggyCode は，インタフェースキュー内で待機している TCP パケットを符号化の対象としている．そのため，PiggyCode には，以下の 2 つの課題が存在する．

- 低送信レート時は，中継端末のインタフェースキューに TCP パケットが溜まりにくいため符号化が行われにくい．
- 高送信レート時は，TCP の輻輳ウィンドウ制御により，DATA パケットの送信間隔が一定にならない．そのため，中継端末が受信するパケットの種別が偏り，符号化を効率的に行うことができない．

この 2 つの課題から，PiggyCode は常に NC の効果を最大限に活用できない．NC の効果を高めるためには，送信レートに影響されずに DATA パケットと ACK パケットを効率的に符号化することが必要である．

2.2 待機時間挿入手法

2.2.1 待機時間挿入手法の概要

先行研究 [3] では，2.1.2 項で述べた PiggyCode の課題を待機時間挿入手法により改善している．待機時間挿入手法は，パケット中継時に待機時間を挿入することで符号化率を改善する手法である．具体的には，インタフェースキューへ挿入する前に符号化対象となるパケットが存在しない場合に，当該パケットを待機させることで符号化率を向上させる．これにより，送信レートにかかわらず，より多くのパケットを符号化することが可能となる．

NS-2 を用いたシミュレーション評価では，2〜4 ホップのチェーントポロジにおいて，スループット，RTT，および ACK パケットの送信回数の計測が行われた．Delayed-PiggyCode は，通常の TCP に比べ，平均受信スループットは約 10%向上し，RTT は最大 70%削減した．また，符号化率は最大 99%であった．文献 [1] とほぼ同等の評価環境における PiggyCode の平均受信スループット向上率は 5%であったことから，待機時間挿入手法は PiggyCode の性能改善に有効であることが示された．また，待機時間の設定は，低送信レート時は待機時間を 10ms に，高送信レート時は待機時間を 30ms にすることが最適であり，かつ，

送信レートおよびホップ数に基づいて待機時間を動的に挿入する手法が有効であると結論づけている．

2.2.2 待機時間挿入手法の課題

待機時間の挿入は符号化率を改善するが，待機時間経過後に符号化を行うことなく送信する場合は，挿入した待機時間分だけ RTT が増加する．そのため，待機時間を挿入する場合は，符号化される可能性が高い場合にのみ限定することが望ましい．しかし，実環境においては，以下のように符号化できないケースが存在する．

- 遅延 ACK オプション [5] により ACK パケットが削減されるため，DATA パケットを待機させても符号化が行えない．
- 輻輳ウィンドウ制御およびトポロジ構成により，TCP パケットの種別に偏りが生じるため，符号化に必要な待機時間が変動する．

前者のケースは，遅延 ACK オプション有効時に発生する．通常，遅延 ACK オプションはデフォルトで有効となっている．遅延 ACK オプションは，ACK パケットの返信を遅延させ，複数の DATA パケットに対する ACK パケットを 1 つにまとめて返信することで TCP 通信を効率化する．なお遅延 ACK オプションの最大遅延時間は 500ms となっているが，最大セグメントサイズより大きなデータを受信した場合，その次の DATA パケットに対して ACK パケットを必ず返信するように規定されている．パケットは最大セグメントサイズに分割されるため，DATA パケットを 2 パケット受信した際に ACK パケットを 1 つ返信する．したがって，送信する DATA パケットのうち約半数の DATA パケットは，待機させても符号化できないことになる

後者のケースは，TCP の輻輳制御によって DATA パケットの送信間隔が一定にならないために発生する．輻輳制御では，輻輳ウィンドウと呼ばれる連続して送信可能なパケット数を調節することで輻輳を抑制している．通信時は，送信端末は輻輳ウィンドウに基づいて DATA パケットを連続して送信し，それを受信した端末は同様に対応する ACK パケットを連続して返信する．したがって，中継端末において，DATA パケットと ACK パケットの種別に偏りが生じる．さらに，送信端末に近いほど DATA パケットを連続して受信しやすく，受信端末に近いほど ACK パケットを連続して受信しやすいため，トポロジ構成によって偏りの傾向が異なる．

これらのケースにおいて，待機時間挿入手法は，冗長な待機時間を挿入してしまう．したがって，適切に待機時間を制御する必要がある．

3. パケット種別の偏りを考慮したキュー制御方式

Delayed-PiggyCode は，すべての DATA パケットに対して待機時間を挿入する．そのため，遅延 ACK オプションのように総 DATA パケット数と総 ACK パケット数との差が大きい場合，待機時間内に符号化できない事象が発生する．また，輻輳制御等による受信パケット種別の時間的偏りも，同様の事象を引き起こす．その結果，符号化されなかった DATA パケットに付与された待機時間はその効果をもたらさず，逆に RTT が増加することになる．これらの事象の解決策として，パケット単位で待機時間を調整することが考えられる．しかし，ACK パケットがどのタイミングで到着するかは，上述の遅延 ACK オプションの有無，輻輳制御のウィンドウサイズだけでなく，中継端末と送受信端末との位置関係や通信リンクの状態などにも依存するため，適切な待機時間の設定が困難である．

そこで，本稿では，キュー制御によってこの課題を解決する．提案するキュー制御方式は，以下のヒューリスティックスに従ってキュー長を調整する．

- 遅延 ACK オプションが有効な環境において，DATA パケットのペイロードサイズが最大セグメントサイズであると仮定すると，受信端末は DATA パケットを 2 つ受信するごとに 1 つの ACK パケットを送信する．
- したがって，輻輳制御等により DATA パケットが連続しても，受信するであろう ACK パケットはその半数である．

提案方式は，上記に基づき，初期状態の待機用キューの最大サイズを 0 パケット分とする．DATA パケット転送時は，あらかじめ設定された待機時間を設定し，待機用キューに挿入する．連続して DATA パケットを受信した際は，先に待機していた DATA パケットを押し出す形で後続の DATA パケットを待機用キューに挿入する．押し出された先行 DATA パケットは，待機時間の満了を待たずに符号化されないまま送信される．待機用キューの最大サイズは，DATA パケットを連続して 2 つ受信し，待機用キューから押し出された DATA パケットを送信するたびに 1 パケット分ずつ増加させる．したがって，連続して受信した DATA パケットのうち，後ろ半分に相当する DATA パケットが待機用キューで待機し，前半の DATA パケットを押し出される形で待機時間を満了することなく送信される．これにより，遅延 ACK オプションで発生する ACK パケット数の減少と，受信パケット種別の偏りに対応することができる．

なお，待機用キューの最大サイズは，ACK パケットを受信して符号化が行われた場合と待機時間が満了し送信される場合に 1 パケット分ずつ減少する．連続して ACK パケットを受信する場合は，待機用キューに複数の DATA パケットが待機している．連続する ACK パケットが返信される場合，その 2 倍以上の DATA パケットを送信している．したがって，少なくとも ACK パケット数分の DATA パケットは必ず待機している．

提案方式の具体的な動作例を図 2 に，動作を以下に示す．

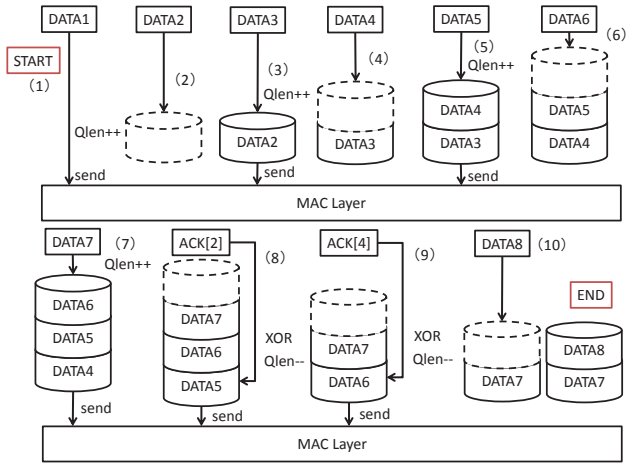

図 2 提案手法の動作概要

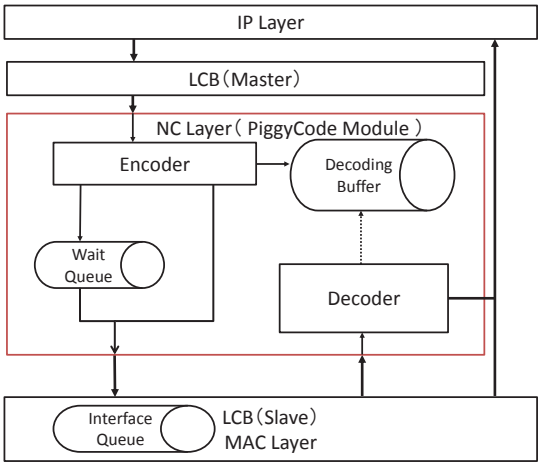

図 3 PiggyCode Module の実装

(1) 通信を開始した際の待機用キューの最大サイズ (Qlen) は 0 である．DATA1 を待機時間を挿入せずにそのまま送信し，Qlen を増加させる．
(2) DATA2 を受信し，キューに挿入する．
(3) DATA3 を受信し，DATA2 を押し出し送信し，DATA3 をキューに挿入後，Qlen を増加させる．
(4) DATA4 を受信し，キューに挿入する．
(5) DATA5 を受信し，DATA3 を押し出し送信し，DATA5 をキューに挿入後，Qlen を増加させる．
(6) DATA6 を受信し，キューに挿入する．
(7) DATA7 を受信し，DATA4 を押し出し送信して，DATA6 をキューに挿入後，Qlen を増加させる．
(8) ACK[2] を受信し，‘[]’ 内の数値は，対応する DATA パケットを示す．キューの先頭にある DATA5 と ACK[2] を符号化して送信し，Qlen を減少させる．
(9) ACK[4] を受信し，キューの先頭にある DATA6 と ACK[4] を符号化して送信し，Qlen を減少させる．
(10) DATA8 を受信し，キューに挿入する．

このように，連続して DATA パケットを受信する際に，符号化に必要なパケット数だけキューに待機させることができる．

4. 実機実装

Delayed-PiggyCode および提案手法を Linux へ実装した．以下，本章では，実機への実装について述べる．

4.1 Delayed-PiggyCode の実機実装

Delayed-PiggyCode を Debian7(LinuxKernel 3.2) へ実装した．実装の簡易化のため，PiggyCode Module を LCB-Driver(Linux Channel Bonding - Driver) の拡張として実装している．LCB は，仮想インタフェース (Master) と実インタフェース (Slave) を登録することが可能である．

PiggyCode Module の実装を図 3 に示す．PiggyCode

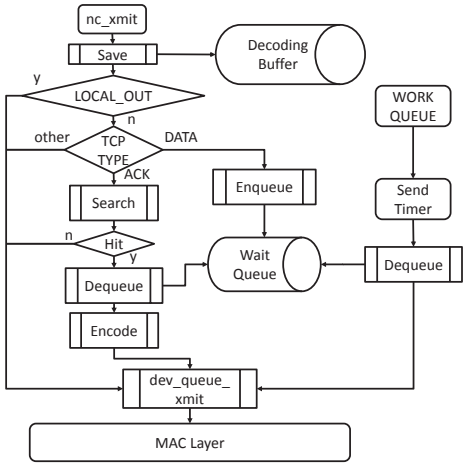

図 4 送信ルーチン (nc_xmit) のフローチャート

Module は，IP-MAC 間で NC（Network Coding）層として動作する．NC 層は，符号化を行う Encoder，復号を行う Decoder，送信パケットを待機させる Wait Queue，デコード用パケットを保存しておく Decoding Buffer で構成する．PiggyCode は，IP ヘッダ以降を符号化するため，NC ヘッダを MAC ヘッダと IP ヘッダの間に挿入する．NC のヘッダは，符号化フラグ，符号化した TCP パケットの各シーケンス番号およびチェックサム，フロー識別用の ID を含む．

送信ルーチン (nc_xmit) のフローチャートを図 4 に示す．送信時は，LCB 内の Master から Slave へのパケット送信処理を nx_xmit 関数に書き換えることで NC 層を実現する．nc_xmit 関数は，TCP パケット以外は何もせず送信する．TCP パケットの場合，NC ヘッダを付与したあと，自端末から送信するパケットの場合は符号化を行わず送信する．中継するパケットの場合は，TCP パケットの種別に応じて処理する．SYN および FIN パケットの場合は，フロー情報の保存・削除を行い，フロー ID の初期値を NC ヘッダに設定して送信する．DATA パケットの場合，Wait Queue に挿入する．カーネルのソフトウェア割り込み機構

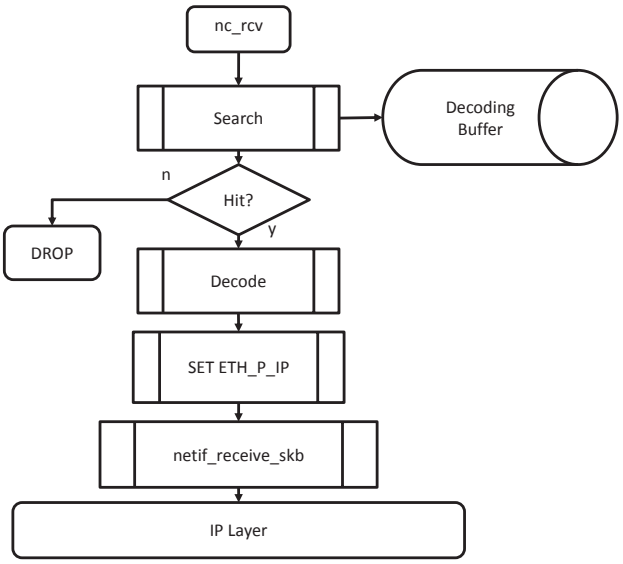

図 5 受信ルーチン (nc_rcv) のフローチャート

である Work Queue を利用して Wait Queue を一定間隔で探索し，待機時間が経過したパケットを送信する．ACK パケットの場合，符号化を試みる．符号化は，送信する ACK パケットと Work Queue 内で待機中の DATA パケットを用いて行い，符号化したのち送信する．

受信ルーチン (nc_rcv) のフローチャートを図 5 に示す．受信時は，dev_add_pack 関数によりカーネルへイーサネットのプロトコルタイプとして ETH_P_NC を登録することで NC 層を実現する．MAC ヘッダのプロトコルタイプが ETH_P_NC であるパケットを受信すると，共通の受信インタフェース (netif_receive_skb 関数) を経由して nc_rcv 関数が実行される．nc_rcv 関数は，符号化パケットの場合，NC ヘッダ内の各シーケンス番号およびチェックサムを用いて Decoding Buffer を探索し，復号を試みる．復号後，NC ヘッダを除去し，MAC ヘッダのプロトコルタイプを ETH_P_IP に書き換え，再度 netif_receive_skb 関数を用いて IP へ送信する．符号化パケットでない場合は，復号後と同様の挙動をとる．

4.2 提案方式の追加実装

提案方式を適用するため，Wait Queue Controller を PiggyCode Module に追加し，そこに提案方式を実装する．Wait Queue Controller を追加した PiggyCode Module を図 6 に示す．Wait Queue Controller は，以下の要素を持つ．

force_send
: 待機時間にかかわらず強制的に送信するパケット数を格納する．DATA パケットを Wait Queue へ挿入する際，force_send の値だけ Wait Queue から強制的に待機中の DATA パケットを送信する．ただし，Wait Queue に DATA パケットが存在しない場合，挿入しようとした DATA パケットをそのまま送信する．DATA パケットの送信後，force_send の値は，送信した数だけデクリメントする．

max_wq_len
: 最大 Wait Queue 長を格納する．最大 Wait Queue 長を超える数のパケットが挿入されようとした時，キューの先頭のパケットを強制的に送信する．

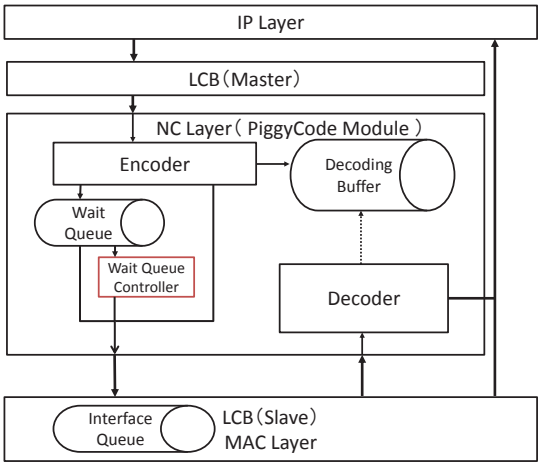

図 6 Wait Queue Controller の追加

提案手法は，2 つ DATA パケットを受信するごとに Wait Queue Controller の force_send をインクリメントすることで実現する．

5. 評価

5.1 実機評価環境

提案手法の有効性を確認するため，実環境において評価を行った．評価環境を表 1 に示す．ネットワークトポロジは，2〜3 ホップのチェーントポロジとした．挿入する待機時間は，1，3，5，10，20，50,100ms と変更して評価した．比較対象は，Standard TCP，PiggyCode（待機時間を挿入しない場合の Delayed-PiggyCode)，Delayed-PiggyCode とする．評価項目として，平均グッドプット，平均 RTT，全端末の送信パケット数の総計（総送信パケット数）の 3 つを計測した．総送信パケット数は，符号化率の指標となる．なお，PiggyCode，Delayed-PiggyCode，提案手法については，NC レイヤで評価項目を計測した．Standard TCP は，NC ヘッダを利用することが出来ないため，リンクレイヤでパケットをキャプチャする機構である tshark を用いて各評価項目を計測した．ただし，TCP の RTT は計測するレイヤが異なるため，単純に比較できない．NC 層は，実際にはリンクレイヤの上位レイヤで動作するため，Standard TCP の RTT については正確な値を比較することができない．したがって，RTT の評価項目については，Standard TCP の値を除外する．

表 1　実機評価環境

パラメータ	設定値
実装環境	Debian7（Linux Kernel 3.2）
通信規格	IEEE802.11g（54Mbps）
ホップ数	2，3
トポロジ	チェーン
アプリケーション	FTP
パケットサイズ	1448
パケット数	10000 パケット
TCP	TCP/cubic

5.2　評価結果

5.2.1　2 ホップにおける評価結果

グッドプットの結果を図 7，全端末における総送信パケット数の結果を図 8，平均 RTT の結果を図 9 に示す．

グッドプットについては，Standard TCP と比較して提案手法および Delayed-PiggyCode は最大 6%向上した．低遅延時はグッドプットにあまり差が見られなかったが，高遅延時は，Delayed-PiggyCode が冗長な待機時間の影響を受け，グッドプットが低下しているのに対し，提案手法は，最大グッドプットに近い値を取り続けている．これは，提案手法により冗長な待機時間を削減しており，その影響を受けないためである．

平均 RTT については，PiggyCode に比べ提案手法は最大約 7%の削減となった．平均 RTT は，グッドプットとほぼ同様の傾向であった．特に，高遅延時において，Delayed-PiggyCode が大きく RTT を増加しているが，提案手法はこの影響を受けていない．これは，グッドプットと同様に冗長な待機時間を削減したことによる．

総送信パケット数については，Standard TCP と比較して提案手法は最大 16%，Delayed-PiggyCode は最大 15%削減した．提案手法は，遅延 ACK オプションを考慮し，待機する DATA パケットの数を変動させているため，効率的に符号化が行えた．低遅延時において性能が低いのは，十分な待機時間が挿入されず，ACK パケットが返信される前にタイムアウトしてしまっていると考えられる．また，10ms 以降において，総送信パケット数の削減率が頭打ちとなっている．このため，2 ホップにおける本手法適用時は，符号化のための待機時間は 10ms で十分であると考えられる．

5.2.2　3 ホップにおける評価結果

グッドプットの結果を図 10，全端末における総送信パケット数の結果を図 11，平均 RTT の結果を図 12 に示す．

グッドプットについては，Standard TCP と比較して提案手法は最大 5%向上，Delayed-PiggyCode は最大 4%向上した．高遅延時は，Delayed-PiggyCode が冗長な待機時間の影響を受け，グッドプットが低下しているのに対し，提案手法は向上している．これは，提案手法により冗長な待機時間を削減しているためである．

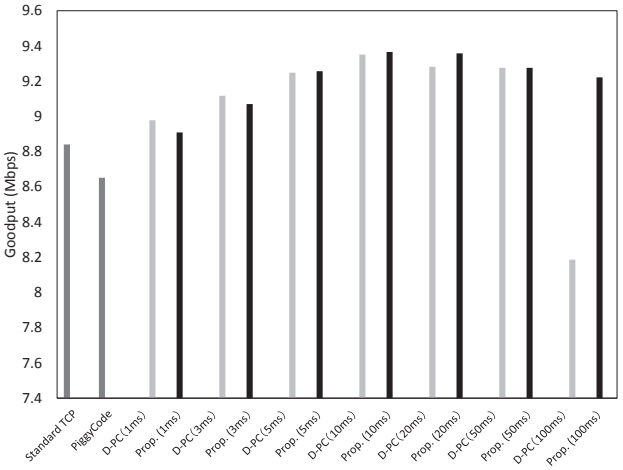

図 7　グッドプット（2 ホップ）

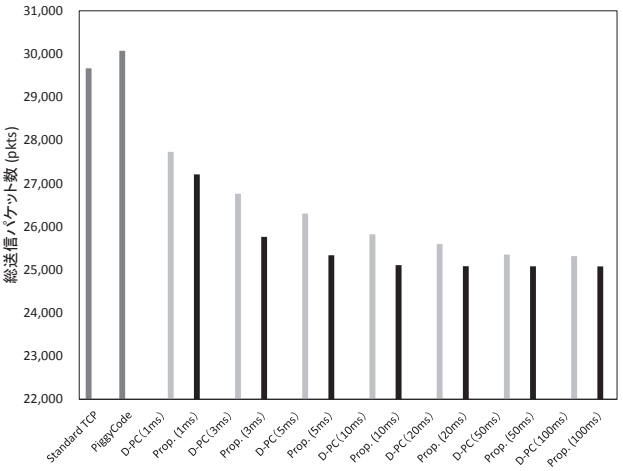

図 8　総送信パケット数（2 ホップ）

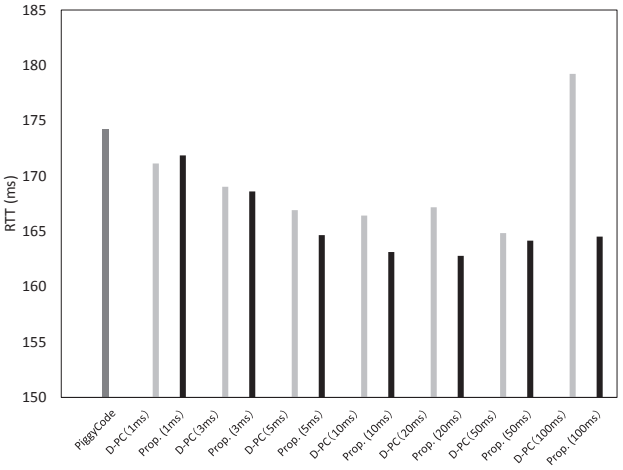

図 9　平均 RTT（2 ホップ）

平均 RTT については，PiggyCode に比べ提案手法は最大 13%の削減，Delayed-PiggyCode は平均 RTT は，最大 6%の削減となった．特に，高遅延時において，Delayed-PiggyCode が大きく RTT を増加しているが，提案手法はこの影響を受けていない．これは，グッドプットと同様に

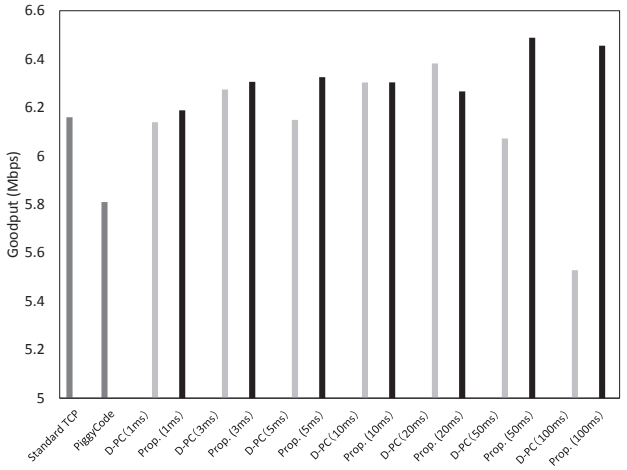

図 10 グッドプット（3 ホップ）

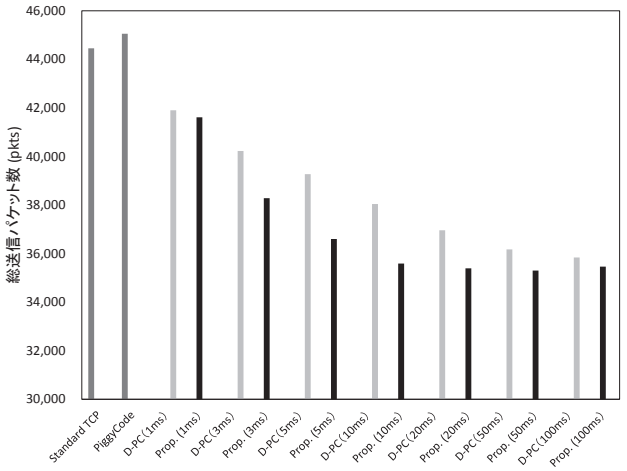

図 11 総送信パケット数（3 ホップ）

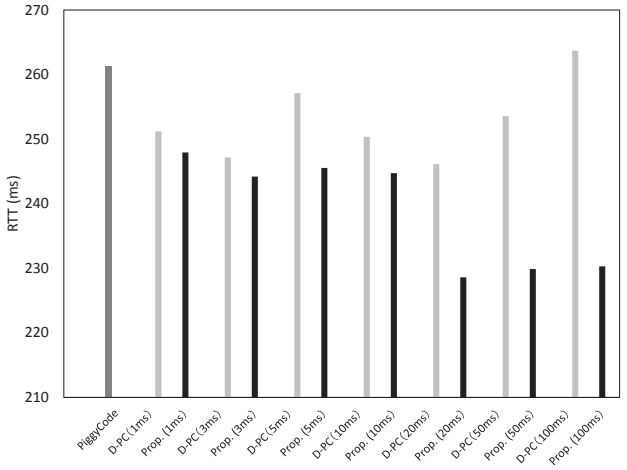

図 12 平均 RTT（3 ホップ）

冗長な待機時間を削減したことによる．

総送信パケット数については，Standard TCP と比較して提案手法は最大 21%，Delayed-PiggyCode は最大 20%削減した．3 ホップの場合においても，総送信パケット数の削減率は 10ms で頭打ちとなり以降は変動しなかった．

5.3 考察

2 ホップ，3 ホップともに，提案手法は，Delayed-PiggyCode よりも総送信パケット数を削減した．これは，提案手法が遅延 ACK オプションを考慮し，適切に DATA パケットを待機させるためである．これにより，DATA パケットと ACK パケットの双方向通信を円滑に行いつつ，効率的に符号化を行えた．総送信パケット数の削減により帯域が削減された結果，グッドプットも向上している．また，平均 RTT についても削減している．これは，総送信パケット数の削減と冗長な DATA パケットの待機時間を削減したことによるものである．

3 ホップは，2 ホップよりも冗長な待機時間の影響が大きく表れている．3 ホップにおいては，中継端末を 2 台経由するため，待機時間を 2 回挿入する．また，DATA パケットと ACK パケットの偏りが発生することにより，Delayed-PiggyCode はさらに符号化率が低下する．しかし，提案手法では，冗長な待機時間を削減しつつ，符号化に必要な DATA パケットを待機させるため，マルチホップ環境における悪影響を受けることなく符号化の効果を得ることが可能である．

提案手法の総送信パケット数は，10ms の場合に削減数が頭打ちとなっている．そこで，Wait Queue へ挿入する時刻と符号化やタイムアウト，提案手法によって Wait Queue からとり出される時刻を記録し，実際に Wait Queue で待機する時間を統計した．待機時間を挿入しない場合は提案手法を適用した場合，100ms 待機する場合においても，待機時間 0〜1ms で取り出されるパケットが約 50%であり，待機時間 0〜10ms 間で取り出されるパケット数の合計は，総待機パケットのうち約 99%であった．したがって，提案手法を適用する場合は，符号化のために挿入する時間は 10ms で十分であると考えられる．

全てのケースにおいて，PiggyCode の性能が低下する原因は，中継端末のインタフェースキューにパケットが溜まらないことにある．これにより，符号化が全く行われないため，NC ヘッダの付与によるパケット長増加の影響が現れていると考えられる．なお，2 ホップ時は，符号化回数が平均 10 パケット程度，3 ホップ時は符号化回数が平均 60 パケット程度となった．

今回実装した手法は，遅延 ACK オプションにより削減される ACK パケットを約半数と見積もっている．HTTP 通信のようなデータサイズが小さなパケットを頻繁に送受信する場合は，削減される ACK パケット数は異なってくる．ただし，遅延 ACK オプションにより削減される ACK パケットは，転送するデータサイズを基に導出可能のため，提案手法のキュー制御方式のパラメータを適切な値に変更することで対応できる．

6. おわりに

本稿では，PiggyCode への待機時間挿入手法におけるパケット種別の偏りを考慮した待機用キュー制御方式について述べた．一般的な TCP 通信は，遅延 ACK オプションが有効となっているため，DATA パケットと ACK パケットの個数に偏りがある．そのため，符号化に必要な待機時間は常に変動し，適応的に待機時間を挿入することは困難である．そこで，DATA パケットを待機させる待機用キューを，DATA パケットを連続で受信する際にキュー長を増加させ，ACK パケット受信時や待機中のパケット送信時にキュー長を減少させる．これにより，遅延 ACK オプションにより発生する受信パケット種別の偏りに対応することで，不要な DATA パケットの待機を抑制する．

提案方式を Linux に実装し実環境において評価した結果，総送信パケット数を最大 21%削減し，これによりグッドプットを最大 6%向上できることを確認した．

参考文献

[1] Scalia, L., Soldo, F. and Gerla, M.: PiggyCode: a MAC layer network coding scheme to improve TCP performance over wireless networks, *Proc. of IEEE GLOBECOM'07*, pp. 3672–3677 (2007).
[2] Ahlswede, R., Cai, N., Li, S.-Y. and Yeung, R.: Network Information Flow, *IEEE Transactions on Information Theory*, Vol. 46, No. 4, pp. 1204–1216 (2000).
[3] 胡　懐穎，瀧本栄二，毛利公一：無線マルチホップネットワークにおける TCP 通信へのネットワークコーディングの適用，電子情報通信学会技術研究報告. IN, 情報ネットワーク， Vol. 112, No. 352, pp. 45–50 (2012).
[4] Huang, Y., Ghaderi, M., Towsley, D. and Gong, W.: TCP performance in coded wireless mesh networks, *Proc. of IEEE SECON'08*, pp. 179–187 (2008).
[5] Braden, R.: Requirements for Internet Hosts – Communication Layers, IETF RFC2018 (1989).