情報処理学会論文誌　コンピューティングシステム　Vol. 3　No. 1　25–37 (Mar. 2010)

トラフィック量に適応する非対称マルチリンク Ethernet トランキング

米元大我[†1]　塙　敏博[†1,†2]　三浦信一[†2]
朴　泰祐[†1,†2]　佐藤三久[†1,†2]

本論文では，非対称なマルチリンク Ethernet により，高性能・耐故障性を実現する汎用ネットワークシステム RI2N+および RI2N++を提案する．従来我々が開発してきた RI2N は Gigabit Ethernet を複数本平行結線することにより，高バンド幅化と耐故障性を実現してきた．Linux で標準的に用いられている Linux Channel Bonding とは異なり，RI2N はネットワークに接続するリンク数をノードごとに変えて非対称な構成にすることができ，価格対性能比の良い柔軟な構成が可能である．非対称なマルチリンク Ethernet トランキングでは，各スイッチに接続されるリンク数およびアプリケーションが発生するトラフィックパターンに応じ，適切な割合でパケット振り分けを行わなければならず，これを怠ると，特に TCP のようなウィンドウ制御によって，大きな性能低下を引き起こす．本システムは送受信ノード間で定期的なトラフィック情報交換を行うことにより，ネットワーク構成とアプリケーションの振舞いに動的に適応した最適なトラフィック制御を行う．本機能により，通常はマルチリンク接続されているノードにおいて，リンク故障が発生してリンク数が減少した場合でも，システム全体でのトラフィックバランスを保ち，通信性能を最適化することが可能になる．その結果，RI2N++では最大で約 60%の性能向上を達成した．

Asymmetric Multi-link Ethernet Trunking System with Adaptive Traffic Control

Taiga Yonemoto,[†1] Toshihiro Hanawa,[†1,†2]
Shin'ichi Miura,[†2] Taisuke Boku[†1,†2]
and Mitsuhisa Sato[†1,†2]

In this paper, general purpose networking system RI2N+ and RI2N++, which enable high-throughput and fault-tolerant interconnection using asymmetrical multi-link connection. We have been developing a multi-link binding network system for Ethernet named RI2N for high-throughput and fault-tolerant interconnection with Gigabit Ethernet. RI2N allows asymmetrical multi-link connection for fitting to various cost-effective and flexible system configuration. Such a configuration cannot be supported by Linux Channel Bonding which is widely used in standard Linux distributions. Under asymmetrical multi-link connection, packets must be distributed to multiple links in optimal ratio based on the number of links or traffic pattern. RI2N++ automatically detects the asymmetric network configuration and controls the traffic distribution to multiple links. In basic performance evaluation under high traffic rate, we confirmed that the throughput of network with our proposed scheme is improved up to approximately 60% to that of original RI2N.

1. はじめに

近年，PC クラスタはコストパフォーマンスの高さから HPC（High Performance Computing）において多くの局面で用いられている．そのようなクラスタ間のネットワークはシステム全体の性能を支えるうえで非常に重要である．SAN（System Area Network）と呼ばれる InfiniBand[1] や Myrinet[2] が存在するものの，MPI を利用した並列計算，NFS やリモートログイン，FTP といった従来の UNIX ネットワークサービスにおいて Ethernet は今もなお広く使われている．特に Gigabit Ethernet（GbE）は，NIC やスイッチが安価に入手可能であり，適度な性能を持っていることから，現在最も価格対性能比に優れたネットワークである．TOP500[3] にランクインするシステムのうち，50%を超えるシステムがクラスタ間ネットワークとして Ethernet を利用しており，また研究室程度の単位における小規模な PC クラスタの多くは 24 ポート程度の安価な GbE スイッチで接続されている．

GbE はコストパフォーマンスの高いシステムを実現できる一方で，スループットやレイテンシの絶対的な性能は InfiniBand DDR や QDR に比べて劣っている．この 2 つの問題のうち，レイテンシの問題を解決することは難しいが，複数の GbE リンクを論理的な 1 つのネットワークとして使用することでスループットを改善することは比較的容易である．Linux Channel Bonding[4]（以下 LCB）は今日の標準 Linux ディストリビューションに含まれており[5),6)]，複数の Ethernet リンクの平行結線によりスループットの改善を実現する．LCB の balance-rr mode[4] は 1 ノードあたりのリンク数を増加させることでバンド幅を向上さ

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

せる．また，balance-rr モードを構成するリンクの 1 つが故障したとき，自動的にリンクの選択を行う耐故障性もあわせ持つ．しかし，LCB はネットワークの構成をシステムで利用する全ノードにあらかじめ登録しておかなければならないことや，現在の LCB ではノード数が最大 16 台に制限されること，一般的なネットワークサービスに用いられるメッセージサイズにおいて比較的スループットが低いこと等から，適用範囲や有効性に問題があることが知られている[7]．また，複数パスを用いるプロトコルとして SCTP（Stream Control Transmission Protocol）[8] が存在するが，既存の TCP/IP で書かれたプログラムを修正する必要がある．

我々はこれまで，大規模な HPC クラスタで利用可能なマルチリンク Ethernet によるネットワークシステムとして，RI2N（Redundant Interconnection with Inexpensive Network）を提案してきた[9]．そのうちの実装の 1 つである RI2N/DRV [7),10),11)] では疑似ネットワークドライバとしての実装がされており，完全にユーザ透過であるため，アプリケーションレベルで一般的な Ethernet と互換性があり，プログラムの変更なしに，あらゆる Ethernet サービスやプロトコルに適用できる．システム構成定義としては，使用する NIC を各ノード上で指定するだけでよいため，使用できるノード数に上限はなく，様々なネットワークトラフィックパターンにおいて LCB より高い性能を持つ．

RI2N/DRV はネットワーク全体がすべて完全な平行結線で接続された「対称」なマルチリンク接続を想定してきたが，システムの潜在的な能力として，非対称なマルチリンクネットワークに対しても適用可能であることが分かっている．ここで，「非対称なマルチリンクネットワーク」とは，1 つの論理的なチャンネルを構成するリンク数が各ノードで異なることをいう．たとえば，大きな PC クラスタシステムにおいて性能と費用のバランスを考えた場合，重要なサーバ間は高バンド幅と耐故障性を持たせるためにマルチリンクで接続し，クライアントとの間は費用を抑えるためにシングルリンクで接続を行うという構成が想定される．RI2N/DRV は本来の性質としてこのような非対称ネットワーク通信に対応できるが，非対称なネットワークではトラフィックの不均衡により性能向上が妨げられていることが分かった．そこで我々は RI2N/DRV を改良することで，非対称なネットワークへ適応させ，さらに性能の向上を図った．本論文では，この新システムの設計と実装，および性能評価について述べる．

2. RI2N/DRV の非対称なネットワークへの適用

本章では，RI2N/DRV の非対称なネットワークにおける利用を考える．初めに RI2N/DRV のコンセプトと実装を簡単に説明し，続いて非対称なネットワークで利用する際の問題点を解明する．

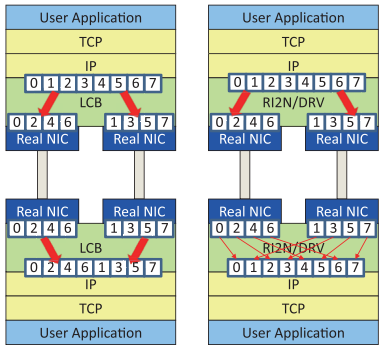


図 1　LCB におけるパケット順序入れ替え発生 (a) と RI2N/DRV におけるパケット順序整合 (b)
Fig. 1　(a) Packet disordering in LCB and (b) its solution in RI2N/DRV.

2.1 RI2N/DRV

RI2N/DRV [7),10),11)] は高バンド幅・耐故障性を実現するために複数の Ethernet NIC をトランクした仮想的なネットワークデバイスである．これはリンクアグリゲーション[12] や，LCB [4] の balance-rr モードと似ている．しかし，LCB において TCP のような高レイヤプロトコルを用いた場合，パケット順序入れ替えが生じ，深刻な性能低下を招く．

LCB の balance-rr モードでは，連続した Ethernet パケットは複数のリンクへラウンドロビンで送信されるため，各 Ethernet リンク上ではパケット順序が不連続になる．受信側のノードにおいても，各 NIC は不連続な Ethernet パケットを受信する．Linux デバイスドライバではハードウェア割込みの回数を減少させるため，NAPI [13] や interrupt coalescing [14] 技術が導入されている．このような状況では，**図 1** (a) に示すように NIC によって起動された割込みハンドラは不連続なパケットを 1 度に取り扱い，上位層である TCP のプロトコルハンドラにそのままの形で渡す．TCP レイヤではそれらのパケットのシーケンス番号が抜けているように見えるため，大量のパケットロスが生じたように観測される．実際にはパケットロスは生じていないにもかかわらず，最終的に多くの ACK パケットが再送要求として送信側へ送り返されることで，これらの不必要な ACK パケットがトラフィックの妨げとなり，全体の性能は大幅に低下する．

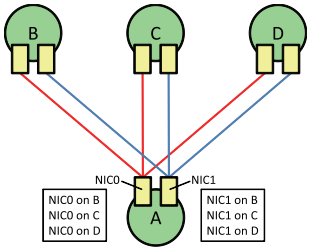


図 2　Node-A 上の通信相手 endpoint 管理情報
Fig. 2　End-points management information in Node-A.

RI2N/DRV はこの問題を次のような方法で解決している（図 1 (b)）．ここで，使用する NIC の数を 2 枚と仮定する．NIC にパケットが到着すると，IP のようなネットワークプロトコルハンドラに代わって，初めに RI2N/DRV ハンドラがパケットを取得する．RI2N/DRV ハンドラでは，一方の NIC からのパケットストリームを受信した後も，他方の NIC のパケットストリームを待って，一定期間パケットを保持する．他方の NIC でパケットが受信されると，元のパケットストリームを再構成するため，パケットをシーケンス番号どおりに並べ替える．このときに物理的なパケットロスがなければ，2 つのパケットストリームから結合された新しいパケットストリームは元のパケットストリームと一致し，このリオーダリングによって上位レイヤでのパケット再送要求を抑制する．本機構を用いることで，多くのトラフィックパターンと一般的なアプリケーションにおいて RI2N/DRV の性能は LCB を上回っていることを確認した．

RI2N/DRV のもう 1 つの特徴は，リンク故障とリンク回復の自動検出である．リンクが故障すると大量のパケットロスを生じるため，できるだけ早い故障検出が必要となる．RI2N/DRV ではノードどうしが通信を行うとき，通信を行うノードの相手とリンクの組合せを “endpoint” と呼ぶ．**図 2** に endpoint の例を示す．ノード A がデュアルリンクのネットワークを通して 3 つのノード B，C，D と通信を行う場合，ノード A は 6 つの通信相手 endpoint を持ち，endpoint ごとにネットワークの統計量を管理する．これにより，多対多通信のように通信相手が複数存在する場合でも問題なく情報の交換が可能となる．ネットワークの故障は，各 endpoint で受信したパケット数を監視することによって検出する．もし同一の相手ノードを指す複数の endpoint 間で，受信パケット数に大きな差が生じ，その差が一定の閾値を超えたなら，RI2N/DRV はパケットの受信が少なかったほうのリンクを故障と判断し，故障したリンクの利用を自動的に停止する．リンク故障後に受信パケット数のカウントを行うことはできないため，リンク回復の検出にはハートビートパケットを用いる．通常，ネットワークの故障からの回復は，ケーブルやスイッチを人手で復旧させることによって行う．したがって，ハートビートパケットの送信間隔は数秒に 1 度といった低い頻度で十分であり，ハートビートパケットが通常のトラフィックに与える影響は無視できる．

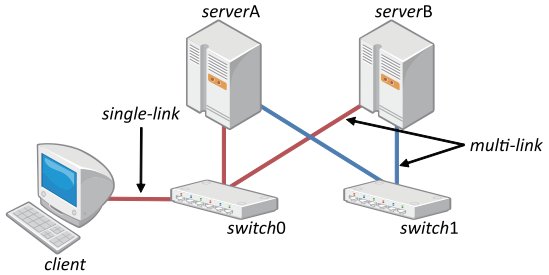
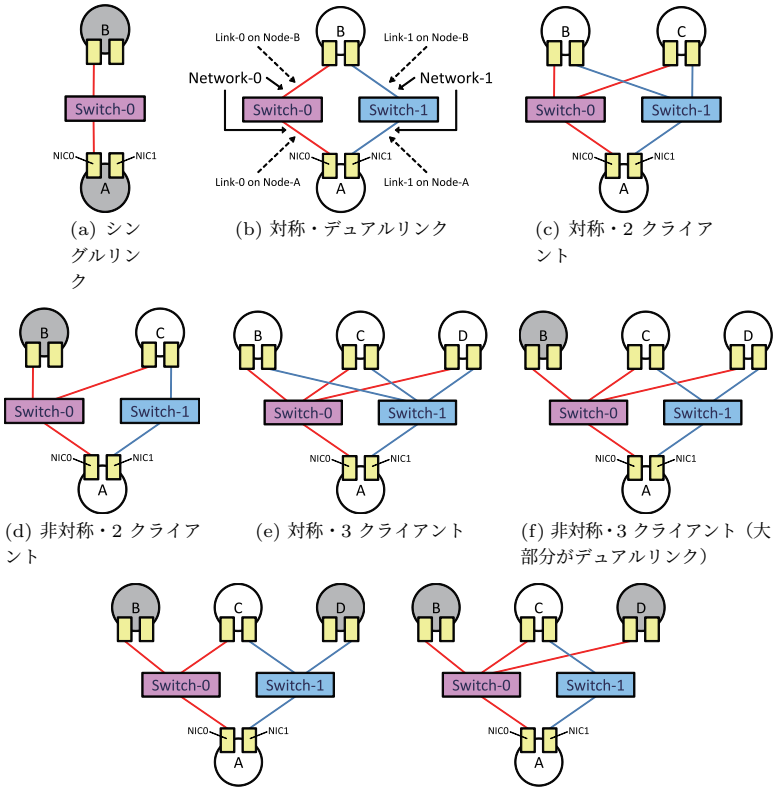


図 3　非対称なネットワーク構造によるマルチリンク Ethernet 接続
Fig. 3　Multi-link Ethernet binding with asymmetric configuration.

RI2N/DRV の実装では，ドライバレベルで Ethernet のすべてのプロトコルに対応するため，いくつかの制御情報が RI2N ヘッダとして Ethernet フレームのヘッダに続いて付け加えられている．加えて，受信側のノードの受信手続き中では，追加された情報を適切に取り除く必要がある．以降，RI2N/DRV を単に RI2N と呼ぶこととする．RI2N/DRV の詳しい説明については文献 7)，10)，11) を参照されたい．

2.2　非対称なネットワーク

本来，LCB と RI2N はすべてのノードにおいて同じリンク数が使われるものとして設計されている．また，近年の CPU とネットワーク性能のバランスを考えると，2 リンクの GbE で最も良いコストパフォーマンスが得られる．しかし，クラスタのノード数が増えてきた場合，全ノードに 2 枚の NIC を差すことや，スイッチ，ケーブルのコストまで含めて考えると，費用面の負担が増加する．クラスタシステムを用途別に複数のパーティションに分けるような場合には，各ノードに必要となる NIC の数を変更することで柔軟でコストパフォーマンスの良いシステム構成が可能となる．このようなネットワーク構成を「非対称なマルチリンクネットワーク」と呼ぶこととし，**図 3** に非対称なネットワーク構成の例を示す．LCB では必ず全ノードが同一のリンク数をとらなければならないため，このようなネットワークを構築することはできない．一方で RI2N には，そのような制限がなく，ノードごとにリンク数が異なるような，あらゆるネットワークを構築することが理論的に可能である．



(a) シングルリンク　(b) 対称・デュアルリンク　(c) 対称・2 クライアント

(d) 非対称・2 クライアント　(e) 対称・3 クライアント　(f) 非対称・3 クライアント（大部分がデュアルリンク）

(g) 非対称・3 クライアント（両スイッチが同数のリンクを持つ）　(h) 非対称・3 クライアント（大部分がシングルリンク）

図 4　様々な対称/非対称構成のマルチリンクネットワーク
Fig. 4　Various symmetric/asymmetric multi-link configurations.

RI2N が非対称なネットワークへ適用できることを確かめるために，**図 4** に示す様々なネットワーク構成で簡単な測定を行った．測定環境を**表 1** に示す．図 4 の各図において，灰色のノードはシングルリンクで，白色のノードはマルチリンクでサーバに接続されている．また，各図において上半分のノードはクライアント，下半分のノードはサーバと位置付ける．以後，Switch-i に接続されるリンクを各ノード上の Link-i と呼び，全ノードの Link-i で構成されるネットワークを Network-i と呼ぶ．

表 1　実験環境
Table 1　Experimental environment.

Item	Specification
CPU	Xeon 5110 Dual-Core 1.6 GHz
Memory	DDR2 2,048 MB
Kernel	2.6.27.24-78.2.53.fc9.x86_64
NIC	Intel PRO/1000PT dual port 1000base-T
Switch	Dell PowerConnect 5324 (24 ports Gigabit Ethernet switch)

表 2　従来の RI2N の性能
Table 2　Performance on the original RI2N.

Topology in Fig. 4	Throughput [MB/s]
(a)	112.2
(b)	223.1
(c)	213.8
(d)	142.9
(e)	211.2
(f)	160.6
(g)	195.7
(h)	120.4

これらの各ネットワーク構成において，クライアントからサーバに対し，単方向連続通信を行う．サーバはデュアルリンクの Ethernet を持つため，理論上パケット受信時の最大スループットはシングルリンクのときに比べ，2 倍となる．**表 2** に結果を示す．表から分かるように，図 4 (b) のような対称ネットワークに比べて非対称ネットワークにおけるスループットが低くなっている．特に図 4 (h) に示す，2 台のクライアントがシングルリンクで接続され，1 台のクライアントがデュアルリンクで接続されているような，非対称性が強い構成では最もスループットが低い結果となった．

この結果は以下のようにして起こると考えられる．たとえば図 4 (d) の場合，Node-B からのトラフィックはすべて Network-0 を通るが，Node-C からのトラフィックは Network-0 と Network-1 に均等に分かれる．この場合，Node-B と Node-C がバースト転送を行うので，Switch-0 の負荷は Switch-1 に比べて増大する．Ethernet スイッチにおいて，同一送信先への大量のトラフィックはパケットロスを生じさせるため，送信側ではパケットロスを減らそうとして輻輳制御が働き，ウィンドウサイズは急速に縮小される．したがって，この場合，Node-B と Node-C は Switch-0 におけるパケットロスに起因して輻輳制御の影響を受ける．Node-B はシングルリンクでのみ接続されているため，輻輳制御の影響を受ける

のは当然である．一方の Node-C は別に Network-1 があるにもかかわらず，Network-0 と Network-1 の両方のネットワークに対してラウンドロビンで同等にパケットを送信してしまうため，Network-1 のトラフィックも，Network-0 でのパケットロスが原因で縮小されたウィンドウサイズに制限されてしまう．

仮に，RI2N がデータリンク層で単一のデバイスとして扱われなければ，このような輻輳制御による問題は生じないだろうが，RI2N を利用する大きな目的の 1 つである，「ユーザ透過な利用」を実現するためには，ユーザがリンク数を意識せずに利用できなければならず，データリンク層では単一のデバイスとしてとらえる必要がある．

ここまでをまとめると，非対称構造を含む様々な構成を持つネットワークに対して RI2N は適用可能であるが，大量のトラフィックを転送する場合において，全体のスループットはネットワークの非対称性が増加するにつれて減少する．したがって，リンクの追加によってバンド幅が増強されているにもかかわらず効果が得られていないことになる．

3. RI2N による動的リンク情報検出

本論文では前章で示した問題を解決するために，RI2N の改良を行う．非対称なネットワークで RI2N が非効率である理由は，混雑しているリンクとスイッチにおいて輻輳制御が働き，空いているリンクを効率的に使えないためであった．ここで図 4 (d) のようなネットワーク構成における解決策を考える．Switch-0 での輻輳制御の影響を小さくするためには，Node-A に接続されている Link-0 と Link-1 を適切なバランスで利用すればよい．これは Node-C の 2 つのリンクが必ずしも同量のパケットを送信しないことを意味する．RI2N の実装においては，Node-C の各リンクはラウンドロビンに従って交互にパケットを送信する．そこで，ネットワーク構成の非対称性に従い，トラフィックのバランスを変える必要がある．

RI2N に改良を加える場合にも，従来の RI2N と同様，ネットワーク全体の構成を記述した複雑な設定ファイルを不要とする．そのため，通信ノード間でリンク情報を交換し合う必要がある．たとえば Node-B と Node-C の間に通信が発生していない場合，Network-0 は Node-B と Node-C に共有され，Network-1 は Node-C に占有されることを Node-A は知っておく必要がある．こういった情報の取得を，RI2N が自動的・動的に行うようにする．この接続情報交換メカニズムを以後，「動的リンク情報検出」と呼ぶ．

接続情報の管理を動的に行う理由は 2 つある．

(1)　システムにネットワーク構成等を記述した設定ファイルの登録が不要で，管理が容易である．

(2)　NIC やケーブル，スイッチに故障が起きた場合，その情報を動的にトラフィック制御へ反映させることができる．

当然，RI2N を用いてもシングルリンク接続のネットワークで故障が起きた場合，トラフィックの回復をすることはできない．冗長なネットワークを設定するかどうかはシステムの目的による．

また，RI2N は非対称ネットワークにおいてもインターネットのような広域における利用は考えず，計算センタの運用系ネットワークや企業の部門等，限られた範囲のネットワークを想定する．非対称なネットワークにおける RI2N の利用例として，NFS のような高スループットが要求されるネットワークサービスがあげられる．ファイルのより高速な転送やネットワークの耐故障性が必要とされる NFS サーバ間はマルチリンクで接続し，クライアントと NFS サーバ間はコスト削減のためにシングルリンクで接続するといった，非対称なネットワーク系においても RI2N がその性能を十分に発揮できれば，コスト性能バランスのとれた効率的なシステム運用が可能となる．一方，MPI のように集団通信がしばしば行われるようなアプリケーションでは計算時間・ネットワークを含め，同期の関係から反応が遅いノードに律速されてしまう．ゆえに，このようなアプリケーションにおいて一定のパフォーマンスを得るためには非対称なネットワークではなく，対称なネットワークでの運用が必須である．

3.1 RI2N+

本節では，動的リンク情報検出に対応した改良版の RI2N の実装について述べる．以後，改良版 RI2N を RI2N+と呼ぶ．

RI2N+では自ノードの endpoint ごとに通信相手の endpoint がいくつの endpoint を持っているかを監視している．たとえば図 4 (d) の場合，Node-A における Network-0 の endpoint は通信相手の Node-B と Node-C の endpoint について ‘2’ という情報をカウンタに持っており，Network-1 に関しては Node-C の endpoint について ‘1’ という情報を持つ．このカウンタの情報は後述するシステムのパラメータ（*HBSPAN*）によって，一定期間保たれる．カウンタの値はハートビートパケットに埋め込まれ，通信相手の endpoint へと送られる．2.1 節で述べたように，ハートビートパケットは比較的低い頻度で定期的に送信される．従来の RI2N のハートビートパケットはリンクの生存情報だけを持っていたが，RI2N+ではこれに加えて，いくつの endpoint が 1 本のリンクを共有しているかという情報を伝える役割も持つ．

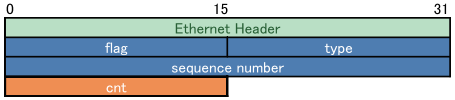


図 5　RI2N+におけるハートビートパケット
Fig. 5　Extension of the heartbeat packet in RI2N+.

本手法では送信側のノードがスイッチの使用状況を判断し，制御を行う．RI2N+は単純なラウンドロビンでパケットを送出する代わりに，自ノードの各 endpoint に重み付けの概念を導入し，この weight 値に従ったパケット割当てを行うことで，全体のトラフィックのバランスを維持する．weight を決定するアルゴリズムには様々なものが考えられるが，ここでは endpoint のカウンタ値に反比例するように weight を決定する，最も簡単なアルゴリズムを採用した．たとえば図 4 (d) の場合，Node-A から Node-C へ伝えられた Link-0 と Link-1 の情報は “2 : 1” であり，それに従って Node-C は “1 : 2” の割合で Link-0 と Link-1 へ送信パケットを割り当てる．このトラフィック制御により，非対称なネットワークにおける全体のトラフィックは平均化される．

図 5 は RI2N+で用いられるハートビートパケットを示す．*sequence number* フィールドには TCP のような上位レイヤとは独立に，RI2N レベルでパケット順序を格納する．RI2N+で新たに付け加えられた *cnt* フィールドには前節で述べた endpoint カウンタ情報を格納する．したがって，各 endpoint の weight はハートビートの間隔（=*HBSPAN*）で更新される．

たとえば図 4 (f) に示すネットワーク構成の場合，Node-B，Node-C，Node-D は Node-A から送られた endpoint カウンタ情報より weight を計算し，**表 3** と**表 4** に示す情報を持つことになる．ノードがパケットを送信する際，RI2N+はこれらの weight に比例して各 NIC にパケットを送出する．この送信割合は次のハートビートパケットが届くまで維持される．

ここで，ハートビートパケットによる endpoint カウンタ情報の非同期更新には小さな問題が生じる．たとえば図 4 (d) のネットワーク構成において，Node-C は Node-A から Network-0 と Network-1 経由で 2 つのハートビートパケットを受け取る．Network-0 と Network-1 は独立しているため，これら 2 つのハートビートパケットは同時に届く保証がない．よって，Node-A の endpoint カウンタ情報は両ハートビートパケットが Node-C へ届き，処理されるまで正しい情報が反映されない．しかし，実際にはハートビートパケットが到達する差は非常に小さく，無視できると考える．

表 3　構成 (f) における Node-B 上の endpoint 管理情報
Table 3　End-point information on Node-B in case (f).

Device	# of links	Weight
Link-0	3	-

表 4　構成 (f) における Node-C と Node-D 上の endpoint 管理情報
Table 4　End-point information on Node-C and D in case (f).

Device	# of links	Weight
Link-0	3	2
Link-1	2	3

3.2　RI2N++

本節では，トラフィック量の観測に基づいてリンク情報検出を実装した改良版 RI2N について述べる．以後，この実装を RI2N++と呼ぶ．

3.1 節で述べた RI2N+は，動的リンク情報検出を接続リンク数の観点から実装したものであった．リンク数によるトラフィックの流量制御では，実際のトラフィック量が各ノードで異なったとしても，リンク数が変わらない限り送信割合は変化しない．

たとえば図 4 (d) のネットワークにおいて，Node-B は Node-A へ ping のような転送量の少ない通信を行い，Node-C は Node-A へバースト転送のような転送量の多い通信を行うことを考えた場合，RI2N+ではリンク数のみを評価するため，リンク数の逆数比から，Node-B が送信するパケットのうち 1/3 は Link-0 から，残りの 2/3 は Link-1 から送出される．このとき，Node-B から送出されるパケット量は非常に少ないため，RI2N+を使用すると，Network-0 の帯域は余ることになる．その結果，ネットワーク全体のスループットとして最大性能が発揮されることは見込めない．

この問題は RI2N+特有の問題であり，従来の RI2N はリンク数等を意識することなく単純ラウンドロビンでパケットを送出していたため，図 4 (d) の通信パターンに限ってはほぼネットワークを利用しきれていたと考えられる．

そこで我々は，RI2N+のようにリンク数だけを流量制御の評価対象とするのではなく，実際にネットワークを流れたトラフィック量を受信ノードで動的に検出し，送信ノードへフィードバックする手法，RI2N++を提案する．

RI2N++では以下のように weight を算出する．まず，

- N_i：Node-i
- $A_{i,j,k}$：N_i が Network-k 経由で N_j へ送信したパケットの到達率
- $S_{i,j,k}$：N_i が Network-k 経由で N_j へ送信したパケット数（N_i による観測値）

- $R_{i,j,k}$：$S_{i,j,k}$ のうち実際に到達したパケット数（N_j による観測値）
- $Q_{j,k}$：N_j へ到着したパケットの総量のうち Network-k から到着したパケット数の割合
- $W_{i,j,k}$：N_i が Network-k で N_j へパケットを送信する際に使用する weight

と定義すると，パケット到着率 $A_{i,j,k}$ は

$$A_{i,j,k} = \frac{R_{i,j,k}}{S_{i,j,k}}$$

のように求められる．また，N_j の Link-k へ届いたパケット数の和 $R_{j,k}$，N_j へ届いた全パケット数 R_j はそれぞれ

$$R_{j,k} = \sum_i R_{i,j,k}$$

$$R_j = \sum_i \sum_k R_{i,j,k}$$

となる．よって，全体のネットワークを見たときに，Network-k が利用されている割合 $Q_{j,k}$ は

$$Q_{j,k} = \frac{R_{j,k}}{R_j}$$

で表される．これらから N_i が N_j へ Netowrk-k 経由で送信するパケット数の最適な weight を求めるには，実際に到達したパケット量 $R_{i,j,k}$ に $Q_{j,k}$ の逆数を掛けることでトラフィックの空いている経路に送信パケットが多く割り振られるようにすればよく，最終的に $W_{i,j,k}$ は

$$W_{i,j,k} = \frac{R_{i,j,k}}{Q_{j,k}} \times A_{i,j,k}$$

と決定する．このとき $A_{i,j,k}$ については，全パケットがロスなしで到着した場合には考慮しなくてよいが，スイッチでロスが発生した場合にはそのスイッチの混雑度を示す尺度となり，weight を減少させる．こうしてパケット数を基に計算された RI2N++の weight は接続リンク数には依存しないため，先ほどの問題は解決できる．また，これらのトラフィック情報は endpoint 情報であるので，2.1 節で述べた endpoint（通信相手＋リンク番号）ごとに管理される．よって，weight の調整は通信相手ごとに独立で行うことができ，RI2N や RI2N+と同様に，多対多通信においても RI2N++は動作する．

図 6 は RI2N++で用いられるハートビートパケットの拡張を示す．ハートビートパケットの前半は RI2N+で用いられたハートビートパケット（図 5）と同様であるが，後半部分が大きく異なる．*arrive* フィールドは $R_{i,j,k}$，*path_total* フィールドは $R_{j,k}$，*arrive_total* フィールドは R_j に相当する．各 NIC の weight はこれらの情報を基に決定されるため，weight は *HBSPAN* の間隔で更新される．ここで各フィールドは 32 bit のサイズを確保してあるが，このサイズは 4 G パケットのカウントができることを意味する．1 パケットが 1,500 バイトのパケットであれば，約 6 T バイトのサイズであるが，それでも *HBSPAN* を非常に長くすればオーバフローを生じる可能性はある．しかし，故障回復検出にも用いられている *HBSPAN* を数千秒に設定することは耐故障性を保つためにも現実的ではないため，GbE を利用する範囲では 32 bit で問題ない．

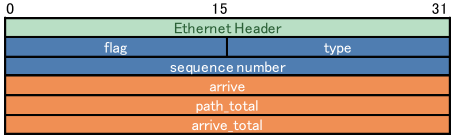


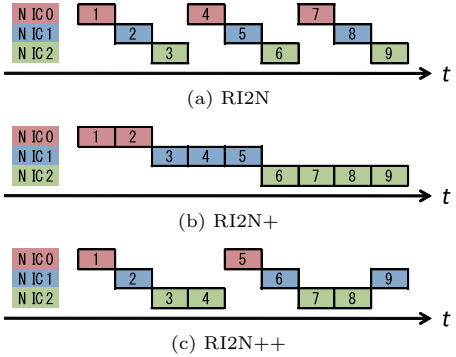
図 6　RI2N++におけるハートビートパケット拡張
Fig. 6　Extension of the heartbeat packet in RI2N++.

RI2N++の weight は，スイッチを共有するリンク数の逆数という単純な値を用いていた RI2N+と異なり，アプリケーションのトラフィック量と *HBSPAN* の設定によっては数万といった大きな値になる．ゆえに，単純に weight を連続送信回数に設定してしまうと問題が生じる．たとえば，weight が Link-0 : Link-1 = 9999 : 10000 といった，大きな値どうしの比となった場合，weight に従ってパケットを連続送信すると，一方の NIC が送信し終わるまで，もう一方の NIC は長期間使用されないことになってしまう．**図 7** に各システムにおいて 3 枚の NIC を利用して 9 パケットを送信するときのパケット送出イメージを示す．

図 7 (a) は RI2N のパケット送出方式を示す．単純ラウンドロビンであるため，Link-0 : Link-1 : Link-2 = 1 : 1 : 1 である．図 7 (b) は RI2N+のパケット送出方式を示す．Link-0 : Link-1 : Link-2 = 2 : 3 : 4 であり，weight がそのまま連続送信回数となっている．図 7 (c) は RI2N++のパケット送出方式を示す．RI2N+と同様に weight の設定は Link-0 : Link-1 : Link-2 = 2 : 3 : 4 であるが，連続送信回数を以下のモデルにより決定する．

- $E_{i,j}$：N_i が N_j との間に持つ endpoint 数
- $C_{i,j,k}$：N_i が Network-k で N_j へパケットを連続送信する回数

$$C_{i,j,k} = \left\lceil \frac{W_{i,j,k}}{min\left\{W_{i,j,0}, W_{i,j,1}, \ldots, W_{i,j,E_{i,j}}\right\}} \right\rceil$$



(a) RI2N

(b) RI2N+

(c) RI2N++

図 7　各システムにおけるパケット送出例
Fig. 7　Packet sending example in each system.

図 7 (c) からも分かるように，NICk は 1 度に $C_{i,j,k}$ を最大回数としてパケットを送出し，weight が新たに更新されるまでの Link-k における総送出回数が $W_{i,j,k}$ を超えると，次の NIC が選択される．

よって，本送信手法により，weight が高い比率になったとしても，パケットの振り分けを，リンク番号という空間的分散において，さらに時間方向でもなるべく均質な分散となるように制御できる．

もちろん，当モデルでは通信ノードが増えた際の集計作業の増大にともなうオーバヘッドの増加は少なからず発生する．しかし，RI2N++におけるトラフィック量の集計作業は通信相手 1 ノードあたり，メモリアクセス回数 16 回，演算回数 8 回で実現されており，通信ノードが数十台から数百台に増えたとしても，この作業にかかる時間はたかだか数 μ 秒程度であり，*HBSPAN* 間隔でしか発生しないため，一般的な UNIX ネットワークサービスへの適用を目指した RI2N++としては問題にならない．なお，ここに示す値は NIC を 2 枚と仮定した場合である．実際にはレジスタが利用できるため，メモリアクセスの回数はより少なくなると考えられる．

4. 性能評価

本章では，RI2N+と RI2N++の性能評価を述べる．LCB は非対称なネットワークをサポートしていないため，ここでは RI2N と RI2N+および RI2N++を様々なネットワーク構成で比較する．本評価で用いる測定環境は表 1 と，評価対象のネットワーク構成も図 4 と同じである．ただし，ここでは *HBSPAN* は 2 秒とする．

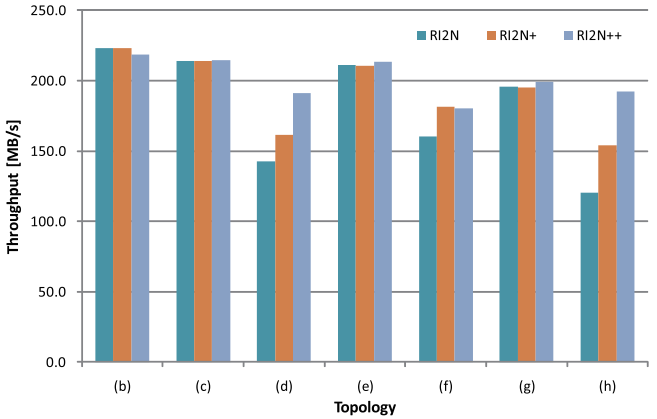


図 8　RI2N+と RI2N++におけるスループットの向上
Fig. 8　Throughput improvement of RI2N+ and RI2N++.

4.1 平均スループット

まず，図 4 に示すネットワーク構成においてクライアントからサーバへ片方向のバースト転送を行い，その平均スループットを比較する．

図 8 に結果を示す．以後，図 4 に示す各ネットワーク構成を単に (a)〜(h) と呼ぶことにする．また，縦軸の “Throughput” は Node-A における全 NIC で観測した 1 分間のスループットの和の平均値を示す．なお，(a) は単にシングルリンクの接続を行ったネットワークであるため省略する．**表 5** には RI2N と RI2N+，RI2N と RI2N++を比較したときの相対性能比を示す．

まず，対称なネットワーク構成である (b)，(c)，(e) は RI2N と比べて RI2N+，RI2N++ともにほとんど変化はなかった．RI2N++の (b) においては約 2%の性能低下が見られるが，それほど大きな低下ではなく，パケットカウントの追加処理等が原因と考えられる．これら 3 つのネットワークは Node-A の endpoint カウントがそれぞれ 2 リンクであり，同じである．したがって，RI2N+においてクライアント上の 2 枚の NIC の weight は同じであり，送信されるパケットは単純ラウンドロビンによって等しく割り当てられる．ここで示された結果は，対称なネットワークにおいて RI2N+が RI2N と同等の性能を持つことを意味しており，RI2N+が RI2N に対する性能的互換性を保つことを示している．また，RI2N++に

表 5　Throughput の相対性能比
Table 5　Relative performance ratio of throughput.

Topology	Ratio [%]		
	RI2N	RI2N+	RI2N++
(b)	100.0	100.0	97.9
(c)	100.0	100.0	100.4
(d)	100.0	113.2	133.9
(e)	100.0	99.6	100.9
(f)	100.0	113.1	112.2
(g)	100.0	99.7	101.6
(h)	100.0	127.8	159.9

おいてもクライアント上の 2 枚の NIC の weight はほぼ等しくなっており，これについては後述する．また，これら 3 つのネットワークにおいて RI2N+と RI2N++の性能差はほぼ無視できる範囲であった．RI2N+と RI2N++の実装の差は，weight を決定するための計算コストのみであるため，性能差がほとんどないという結果は RI2N+と RI2N++の計算オーバヘッドの差がほとんど無視できることを裏付けている．

次に，非対称なネットワークについて検証する．非対称なネットワークである (d)，(f)，(h) において，RI2N+は RI2N に比べてそれぞれ 13.2%，13.1%，27.8%，RI2N++はそれぞれ 33.9%，12.2%，59.9%という性能向上を見せた．ここで注目すべきは，(h) のように非対称性が強く，トラフィックの偏りが激しいネットワークほど，RI2N+および RI2N++の効果が大きいという点である．

(g) は非対称なネットワークに分類されるが，RI2N でも比較的高い性能が得られている．なぜなら (g) はある意味で対称的なネットワークであり，Switch-0 と Switch-1 のトラフィックはほぼ同じだからである．実際に Node-A の Link-0 と Link-1 におけるトラフィック量は Node-B，Node-D から送信されるパケットおよび Node-C のデュアルリンクから送信されるパケットで同量となる．事実，RI2N の性能は対称な場合の (e) と近く，RI2N+，RI2N++は RI2N とほぼ同じ性能となっている．なお，RI2N++の (e) と (g) における性能は微少な改善を見せているが，これは処理のオーバヘッドが少なくなったことを意味するわけではなく，単に測定時のばらつきが要因であると考える．

4.2　weight の比較と変化

これまで見てきたように，RI2N+と RI2N++の weight 決定方法は異なり，その結果，得られるスループットに差が生じることが分かった．そこで，実際に weight にどれぐらいの差があるのかを，先ほどと同じく (b)〜(h) を用いたバースト転送により検証する．

表 6　RI2N+と RI2N++における weight の比較
Table 6　Comparison of weight between RI2N+ and RI2N++.

Topology	Node-B		Node-C		Node-D	
	+	++	+	++	+	++
(b)	1.0	1.0	-	-	-	-
(c)	1.0	1.0	1.0	1.0	-	-
(d)	-	-	2	5.0	-	-
(e)	1.0	1.0	1.0	1.0	1.0	1.0
(f)	-	-	1.5	2.2	1.5	2.2
(g)	-	-	1.0	2.1	-	-
(h)	-	-	3	10.6	-	-

表 6 は 2 枚の NIC を利用するクライアントにおいて，Link-0 の weight を 1 としたときの Link-1 の weight 比をまとめたものである．表 6 ではシステム名 RI2N+を ‘+’，RI2N++を ‘++’ で表す．今回の実験では計測中にリンク数が変化しないため，RI2N+の weight が変わることはないが，パケット量を観測する RI2N++は *HBSPAN* の間隔で weight が更新されるため，定常状態になってからの 1 分間の平均を記す．

4.1 節では，(d) と (h) における RI2N++の性能が RI2N に比べて特に良いことを示したが，それは weight に大きな差が生じていたからであったことがこの結果から分かる．また，(f) においては今回検証した非対称なネットワークにおいて，1 台のクライアントに占有されるスイッチが存在しない唯一の構成であり，どちらかのネットワークをできるだけ多く使った方がよいという考え方はできない．実際，RI2N++は RI2N+に比べて大きな weight を Link-1 に割り当てているが，どちらのアルゴリズムでもほとんど同じ結果となっている．このようなネットワークでは，実際にアプリケーションの振舞いにおいて動的なトラフィックの偏りが生じた場合に RI2N++の有効性が発揮されると考えられ，さらなる検証が必要である．

表 7 にスイッチにおける congestion level を各システムで比較した結果を示す．表 7 では表 6 より求めた，Switch-0 と Switch-1 に入力されるトラフィック量の比が congestion level として表してあるが，これを見ると RI2N，RI2N+，RI2N++の順に “1 : 1” へと近づいていくことが分かる．計算するにあたり，実際には輻輳制御が働くため表 6 の Node-B における ‘1’ と Node-C における ‘1’ が等しいとはいえないが，理想的に衝突がないとすれば ‘1’ が意味するところは同じであるという前提で比較を行った．そうすると，congestion level が “1 : 1” であるということはネットワークのトラフィック平均化にともない性能が改善されたことを意味し，RI2N++が正しく機能していることの証明でもある．(b)，(c)，(e)

表 7　congestion level の比較
Table 7　Comparison of congestion level.

Topology	RI2N	RI2N+	RI2N++
(b)	1 : 1	1 : 1	1 : 1
(c)	1 : 1	1 : 1	1 : 1
(d)	2 : 1	1.5 : 1	1.2 : 1
(e)	1 : 1	1 : 1	1 : 1
(f)	1.5 : 1	1.2 : 1	1 : 1
(g)	1 : 1	1 : 1	0.8 : 1
(h)	3 : 1	2.3 : 1	2.1 : 1

は対称なネットワークなので，“1 : 1” となるのは自明である．(g) における RI2N++ の値は “0.8 : 1” となっているが，トラフィックを平均化するための本来の理想は “1 : 1” である．本現象は Switch-1 で Node-C と Node-D から来るパケットが衝突し，偶然 Node-D の方が Node-C より早く輻輳制御が働いたことにより発生したと考えられ，Node-C は Node-D のペースダウンにより空いた Network-1 を Network-0 より多く使用したことになる．事実，図 8 の結果において (g) 全体のスループットは RI2N++が RI2N と RI2N+の性能に勝っており，RI2N++の調整が成功したことを意味している．(h) においては “2.1 : 1” という “1 : 1” には遠い結果となっており，RI2N や RI2N+に比べると改善はされたものの，まだまだ改良の余地があることを表す．

ここで RI2N++が通信状況に応じて変化を追従することを確認するために，故障やトラフィック量の変化を想定し，時間が経つにつれてトポロジを (e)，(f)，(h) の順に変化させるという実験を行った．Node-B，Node-C，Node-D における weight の変化を，それぞれ**図 9** (a)，(b)，(c) に示す．weight は各ノードにおいて，両リンクの総和が 1 となるように正規化してある．したがって，対称的な構成で両リンクが完全にバランスしているときに “0.5 : 0.5” となる．トポロジの変化は時刻 60 秒で Node-B の Link-1 を切断，続いて時刻 120 秒で Node-D の Link-1 を切断することにより行う．まず，時刻 0–60 秒では Node-B，Node-C，Node-D のいずれのノードも平等な環境であるため，weight に偏りがないことが見てとれる．次に時刻 60–120 秒では Node-B の Link-1 が切断され，Node-C と Node-D が同じ環境となる．そこで，図 9 (b) と図 9 (c) を比較すると，本来ならば weight の偏りが Node-C と Node-D で同じになるはずであるが，Node-C の weight の差に対して，Node-D の weight の差は小さい．これは，Node-C と Node-D が空いたネットワークである Network-1 の帯域を奪い合う段階において，偶然 Node-C が Network-1 の帯域を多く確保したタイミングで Network-0 では Node-B と Node-D における TCP の輻輳制御によってパケットロスが少なくなり，送信されるべきパケット数が固定した結果であると考える．

最後に時刻 120–180 秒では Node-D の Link-1 も切断され，Node-C は Link-1 の weight をより重くしている．これらの結果は，RI2N++が動的変化に対応していることも裏付ける．ただし，(f) のような非対称ネットワークでかつ，すべての Network が混雑している状況においては，トラフィック競合の解消が難しいため，どのようにしても全体のスループットの低下が生じ，RI2N と大差ない結果となりうる．こうした状況はノード数が増えるほど発生しやすくなると予想されるが，weight に偏りが付きにくくなるだけで，RI2N と同等の性能は最低限維持される．しかし，RI2N++は与えられた環境の中で全体のスループットを最大性能にするということも目的の 1 つであるため，全体のスループットが理想値に到達しないことは大きな問題であり，今後の課題となっている．

4.3　スループットの内訳

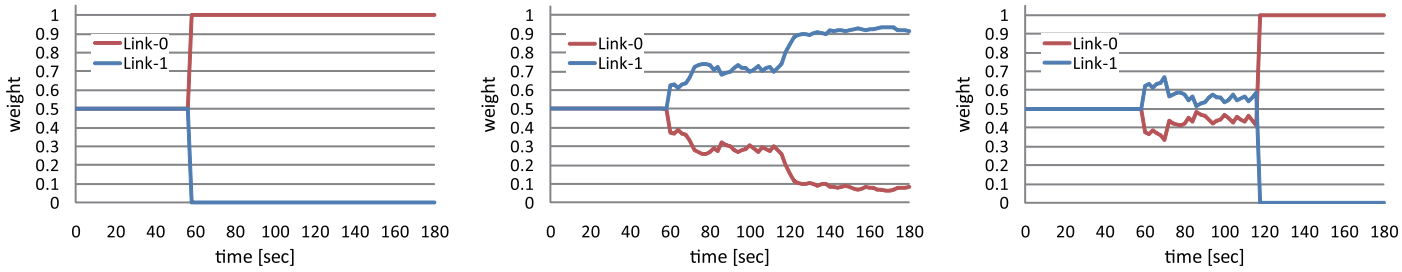
全体のスループットのうち，各ノードがどれぐらいのトラフィックを流しているか確認するために，(h) において Node-B，Node-C，Node-D のトラフィックを Node-A で観測した．**図 10** は，各システムで計測した 1 分間のスループットの平均であり，Node-B，Node-C，Node-D からのトラフィックをそれぞれ分類している．

RI2N に比べて RI2N+，RI2N++の Node-C におけるトラフィックが非常に増加していることが分かる．一方，Node-B と Node-D はシングルリンクしか搭載していないので，それらのノードによるトラフィックの増加は少ない．Node-C の Link-0，Link-1 に対する weight（パケットの送信割当て比率）は，RI2N+では Node-A の endpoint カウンタ情報によって 1 : 3，RI2N++では表 6 で示したように 1 : 10.6 となっており，全体のスループットが向上した主な要因は Switch-1 を経由して Link-1 へ転送された Node-C からのトラフィックの増加が大きかったからであることが分かる．

4.4　トラフィックが偏った通信パターン

3.2 節で取り上げた，トラフィック量に大きな差があるときの通信パターンについて確認する．(d) において，Node-B から Node-A へ ping による 1 秒間隔の通信を行い，Node-C から Node-A へは片方向のバースト転送を行う．**表 8** はこのときに Node-C と Node-A 間のスループットを 100 秒間観測し，その平均を計算した値を示す．

RI2N+は RI2N に比べて約 10%の性能低下が見られるが，ネットワークのトラフィック量を意識して設計した RI2N++では約 0.5%しか低下が見られない．これは測定誤差の範囲にあると考えられ，このことから RI2N++はトラフィックが偏った通信パターンへも適用



(a) Node-B　(b) Node-C　(c) Node-D

図 9　各ノードにおける weight の変化
Fig. 9　Time transition of weight in each node.

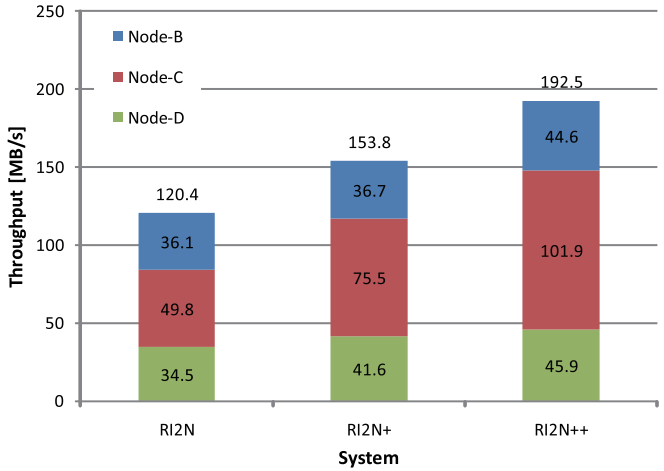


図 10　構成 (h) におけるスループット向上とその内訳
Fig. 10　Throughput improvement and breakdown in case (h).

可能であることが確認できる．

5. 今後の課題

RI2N++を導入することにより，RI2N+の弱点であったトラフィックの偏りが大きい場合の問題については解決できたが，4.2 節で示した (f) のようなネットワークにおけるトラフィックの完全な平均化は達成していない．さらに RI2N++の現在の結果では，(d) における Node-C の weight 比が “1 : 5” となっているが，理論上の最適解は “0 : 1” である．もし，Node-C から送られるすべてのパケットが Network-1 を経由するならば，Node-B は Network-0 を占有することができ，Node-A における Link-0 と Link-1 の受け取るパケット数はバースト転送時に同量となる．しかし，ネットワークの構成がより複雑になってくると，その判断はいっそう難しくなるため，より効率的かつ効果的なアルゴリズムを考えなければならない．

表 8　トラフィックに偏りがある通信パターンの評価
Table 8　Evaluation under the unbalanced traffic.

System	Throughput [MB/s]
RI2N	235.5
RI2N+	213.8
RI2N++	234.2

RI2N++においては，トラフィック量を観測して weight を決定するが，実際にはパケット数を観測して送信ノードへのフィードバックを行っている．しかし，通信に使われるパケットサイズは必ずしも一定ではなく，パケット数とトラフィック量が直結しないこともある．今回の評価だけではこの問題がボトルネックとなることはなかったが，今後実アプリケーションによる評価が必要である．

また，現在は「スイッチがどのくらいのクライアントで共有されているか」という観点で

考えているが，スイッチの段数が増えた場合には期待した動作をしない可能性がある．そこで，これを「共有しているスイッチ」ではなく「共有しているネットワークパス」という形に拡張し，多段構成のより複雑なネットワークで検証しなければならない．

実アプリケーションへの適用に関しては，アプリケーションが *HBSPAN* を超える間隔で間欠通信を行った場合，現アルゴリズムでは通信時の weight を無通信時の結果から算出した値で決定してしまい，RI2N と同様に全体のスループットが伸び悩む可能性がある．この問題を解決するには，weight を直前の情報からのみ決定するのではなく，一定期間の移動平均をとる等の処置が必要だろう．しかし，移動平均をとると weight の変化は緩やかになるため，トポロジ変化への対応の即時性とはトレードオフの関係にある．ゆえに，システムのネットワーク性能と即時性の関係についてさらに考慮し，場合によっては *HBSPAN* の動的調整等も検討する必要がある．

6. おわりに

本論文では，非対称なネットワークに対してトラフィック制御を行い，ネットワーク全体のトラフィックを平均化するマルチリンク Ethernet 制御システム，RI2N+および RI2N++を提案した．クラスタの大規模化が進む今日において，非対称なネットワークをクラスタ上で構築できることは非常に重要なことであるが，マルチリンク Ethernet 接続に関する Linux 上のデファクトスタンダードである LCB では，非対称なネットワークをサポートしていない．従来の RI2N でも非対称ネットワークには結果的に対応できていたが，元来このような状況を想定した実装とはなっていなかった．

非対称なネットワークでより高い性能を出すために，通信リンク数を基にトラフィックの平均化を行う RI2N+が開発されたが，RI2N+は設計上の理由から最高性能が得られないトラフィックパターンが多かった．そこで，実際のトラフィックを観測し，パケット送信割合の決定を行う RI2N++が開発された．

本提案により，非対称なネットワーク上で RI2N に比べて RI2N+ではスループットが最大約 30%，RI2N++では最大約 60%向上したことが確認された．また，いずれのシステムにおいてもハートビートパケットにより動的にリンクの情報を検出できるため，もし NIC やケーブル，スイッチの故障等によりネットワークの構成が変化しても RI2N+，RI2N++は対応することができる．この特徴はオリジナルの RI2N における耐故障機能を継承したものであり，性能面と機能面において RI2N と上位互換性を持つことを意味する．

今後の課題として，より効率的にトラフィックを平均化するアルゴリズムの検討はもちろんのこと，トラフィックパターンが時々刻々と変化する実アプリケーション上において，最大限の性能を得られるような制御方法を検討する必要がある．

謝辞　本研究の一部は，JST-CREST 研究領域「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」，研究課題「省電力でディペンダブルな組込み並列システム向け計算プラットフォーム」による．

参考文献

1) InfiniBand Trade Association: InfiniBand. http://www.infinibandta.org/
2) Myricom: Myri-10G Solution. http://www.myri.com/
3) TOP 500 Supercomputing Sites: TOP500. http://www.top500.org/
4) Davis, T.: Linux Ethernet Bonding Driver. http://sourceforge.net/projects/bonding
5) Red Hat, Inc.: Red Hat Linux. http://www.redhat.com/
6) Novell, Inc.: SUSE Linux. http://www.novell.com
7) Miura, S., Hanawa, T., Yonemoto, T., Boku, T. and Sato, M.: RI2N/DRV: Multi-link Ethernet for High-Bandwidth and Fault-Tolerant Network on PC Clusters, *The Workshop on Communication Architecture for Clusters with IPDPS2009*, pp.1–7 (2009).
8) Stewart, R.: Stream Control Transmission Protocol, RFC 4960 (Internet standard) (2000).
9) Okamoto, T., Miura, S., Boku, T., Sato, M. and Takahashi, D.: RI2N/UDP: High bandwidth and fault-tolerant network for a PC-cluster based on multi-link Ethernet, *The Workshop on Communication Architecture for Clusters with IPDPS2007*, pp.1–8 (2007).
10) 岡本高幸，三浦信一，朴　泰祐，塙　敏博，佐藤三久：ユーザ透過に利用可能な高性能・耐故障マルチリンク Ethernet 結合システム，情報処理学会論文誌　コンピューティングシステム，Vol.1, No.1, pp.12–27 (2008).
11) 三浦信一，米元大我，塙　敏博，朴　泰祐，佐藤三久：高性能・耐故障マルチリンク Ethernet 結合システムの性能評価，情報処理学会研究報告（ハイパフォーマンスコンピューティング），Vol.2009-HPC-120, No.9, 情報処理学会 (2009).
12) IEEE: IEEE 802.3ad — Link Aggregation (2000). http://www.ieee802.org/3/ad/index.html
13) Salim, J.H., Olsson, R. and Kuznetsov, A.: Beyond softnet, *ALS '01: Proc. 5th annual Linux Showcase & Conference*, Berkeley, CA, USA, p.18, USENIX Association (2001).
14) Intel Corporation: Intel PRO Network Connections User Guides.

(平成 21 年 7 月 24 日受付)
(平成 21 年 12 月 9 日採録)

米元　大我（学生会員）

昭和 61 年生．平成 21 年筑波大学第三学群情報学類卒業．現在，筑波大学大学院システム情報工学研究科博士前期課程．クラスタおよび分散コンピューティング，コンピュータネットワークに興味を持つ．

塙　　敏博（正会員）

平成 10 年慶應義塾大学大学院理工学研究科計算機科学専攻博士課程修了．博士（工学）．東京工科大学コンピュータサイエンス学部講師，筑波大学計算科学研究センター研究員を経て，平成 20 年筑波大学システム情報工学研究科准教授．ディペンダブルシステム，クラスタコンピューティング，計算機アーキテクチャ等に関する研究に従事．

三浦　信一（正会員）

昭和 54 年生．平成 14 年千歳科学技術大学光科学部光応用システム学科卒業．平成 20 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士課程修了．博士（工学）．現在，筑波大学計算科学研究センター研究員．クラスタコンピューティング，ネットワークに関する研究に従事．

朴　　泰祐（正会員）

昭和 35 年生．昭和 59 年慶應義塾大学工学部電気工学科卒業．平成 2 年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和 63 年慶應義塾大学理工学部物理学科助手．平成 4 年筑波大学電子・情報工学系講師，平成 7 年同助教授，平成 16 年同大学大学院システム情報工学系助教授，平成 17 年同教授，現在に至る．超並列計算機アーキテクチャ，ハイパフォーマンスコンピューティング，クラスタコンピューティング，グリッドに関する研究に従事．平成 14 年度および平成 15 年度情報処理学会論文賞受賞．IEEE CS, ACM 会員．

佐藤　三久（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成 3 年通産省電子技術総合研究所入所．平成 8 年新情報処理開発機構並列分散システムパフォーマンス研究室室長．平成 13 年より，筑波大学大学院システム情報工学研究科教授．平成 19 年より，同大学計算科学研究センターセンター長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術，グリッドコンピューティング等の研究に従事．IEEE, 日本応用数理学会各会員．