論　文

拡散現象を指導原理とする自律分散フロー制御の実装に向けたアクティブフロー数計測技術の検討

高野　知佐[†a)]　　山内　正志[††]　　会田　雅樹[†††]

On Implementation Issues of Diffusion-Type Flow Control for Multiple Flows

Chisa TAKANO[†a)], Tadashi YAMAUCHI[††], and Masaki AIDA[†††]

あらまし　筆者らはこれまで，高速ネットワークの制御に求められる短い制御遅延の要求を満足するために，自律分散で動作する拡散型フロー制御技術を提案してきた．この技術は，局所的なネットワーク情報のみに基づくノードの自律動作によって，間接的にネットワーク全体を適切な状態に導く制御技術である．本論文では，自律分散システムとしてのモジュール構成を考慮しながらネットワークシミュレータ ns2 への実装を行い，動作特性の評価・分析を行った．この結果，複数フローが混在する環境での適切な制御動作を実現するためには，アクティブなフロー数に関する情報を局所的に推定する技術が必要であることが分かった．この課題を解決するため，拡散型フロー制御技術の改良方式を提案し，その効果を確認した．また，この改良により，複数フロー間のパケットスケジューリングを簡略化することが可能となり，より簡易な実装技術に結び付けることができた．

キーワード　トラヒック，フロー制御，拡散方程式，自律分散制御

1. まえがき

近年のインターネットの急速な普及・需要の拡大により，将来をにらんだ高速なネットワークの構築が進められている．超高速ネットワークでは，光速度で決まる固定伝搬遅延に比べてノードの動作速度が相対的に高くなり，ノードの高速動作を特徴づける時間スケールにおいて，各ノードが収集可能な情報は非常に限られたものになる．遠くのノードから収集した情報は過去の情報であり，ネットワーク全体に関する現時点の状態を知ることはできないからである．この意味で，ネットワークの高速化は各ノードが情報的に孤立する状況を招く．また，高速ネットワークでは制御遅延の影響が大きいため，非常に高速で動作する制御機構が必要となる．このように，超高速ネットワークで適切に動作する制御を実現するためには以下の課題を解決する必要がある．

- **局所的な状態情報のみで動作する制御機構**

超高速ネットワークでは，リンク伝搬遅延（光速度により決まる）によりネットワークの広範囲な状態情報をリアルタイムに把握することができない．このため，リアルタイムに知ることができる局所的な状態情報のみで動作可能な制御が必要になる．

- **超高速で動作する制御機構**

超高速ネットワークでは，ネットワーク状態が短時間で大きく変化するため，非常に短い制御遅延で動作するネットワーク制御技術が必要になる．

フロー制御の課題に対する多くの研究は，線形計画問題として定式化される [1]〜[3]．これらの定式化では，ネットワークの状態情報を収集が可能であることを前提にしている．しかし，高速ネットワークでは広範な状態情報をリアルタイムに収集することができない．また，線形計画問題の求解には十分な時間が必要なので，高速ネットワークに要求される非常に短い制御遅延を満足することが難しい．このように，ネットワーク全体の情報を 1 箇所に収集することを前提とした集中制御の枠組みは，超高速ネットワークでのフロー制御として適切でない．このため，自律分散制御

† 広島市立大学情報科学研究科，広島市
Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan
†† 東京都立科学技術大学工学部，日野市
Faculty of Engineering, Tokyo Metropolitan Institute of Technology, Hino-shi, 191–0065 Japan
††† 首都大学東京大学院システムデザイン研究科，日野市
Graduate School of System Design, Tokyo Metropolitan University, Hino-shi, 191–0065 Japan
a) E-mail: takano@hiroshima-cu.ac.jp

に基づくフロー制御機構が必要とされる．

現在，自律分散型のフロー制御として，例えばTCPのようなエンドホストによる制御が広く用いられ，その特性について活発に研究されている [4]〜[6]．エンドホストによるフロー制御では，タイムスケールはラウンドトリップタイム（RTT）程度となる．低速なネットワークでは，RTT 程度の制御遅延による影響を無視することが可能だが，ネットワークが高速化するとRTT 程度の制御遅延が大きな影響を与えるようになる可能性がある．ネットワークの高速化に対して RTT 自身の物理的な値は不変であっても，ノードの動作速度の時間スケールを時間の基準として見た RTT は大きくなるからである．これは，RTT 程度の時間内にノードが処理するパケット数が増大することを意味し，より制御遅延の影響を強く受けるようになる．RTT よりも短い時間スケールで俊敏に動作する制御を実現するためには，ノードが局所情報に基づいて自律動作するタイプの制御機構が必要になる．

我々は，高速ネットワーク環境でも適切に動作するフロー制御方式として，拡散型フロー制御技術（diffusion-type flow control：DFC）を提案してきた．このフロー制御方式は，各ノードが自分自身で利用可能な局所的情報のみに基づいて自律動作を行い，その結果としてネットワーク全体を安定させるものである．図 1 はネットワークの各種制御を，要求される制御遅延時間の大きさにより階層化したものである．routing 制御，session initiation protocol（SIP）などのシグナリング処理，TCP や explicit congestion notification（ECN）などの流量制御は，それぞれ数十分，秒，RTT 程度の制御遅延をもつ．DFC は，RTT よりも更に厳しい制御遅延の要求を満たすための制御技術として位置づけられる．DFC が対象とする制御遅延の時間スケールは，ネットワークが高速化するのに伴い重要性が増す領域になっている．

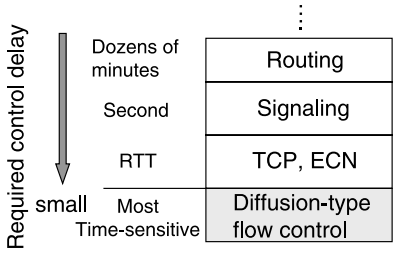

図 1　各種ネットワーク制御の要求される制御遅延による階層化
Fig. 1　Hierarchical structure of network controls with respect to their required control delay.

これまで，シミュレーションによって DFC によるネットワーク状態の安定性，状態変化に対する適応性，TCP との親和性を評価し，DFC がねらいどおりの性能を発揮することを確かめてきた [7]〜[10]．これらの結果に加え，DFC が実ネットワークでも正しく動作することを確かめるためには，DFC の実装に向けた技術課題として以下の課題を確認する必要がある．

- DFC が自律分散制御としての装置構成によって実装が可能であること．
- 自律分散制御として実装した DFC がねらいどおりの性能を発揮すること．

これまでの評価では，TCP と DFC との共存可能性や相互補完性を調べるために，ネットワークシミュレータである ns2 [13] を利用し，DFC 機能を ns2 に実装することで評価を行ってきた．ns2 は本来，装置のモジュール構成を反映したシミュレータとなっているため，自律分散制御としての装置構成を意識しながら DFC 機能を ns2 に実装することによって，DFC 機能の実装可能性を確認することができるはずである．しかし，これまでの評価では DFC の構成を完全に自律分散制御の装置構成で実装したわけではないため，DFC の実装可能性が確認できていなかった．

ns2 に組み込んだ DFC 機能のこれまでの実装では，シミュレーションモデルとして「各ノードは自分を通過するフロー数を知っている」という前提のもとで評価を行い，DFC が適切に動作することを確認してきた．自律分散的な装置構成では，すべてのフロー情報を知っている管理装置の存在を仮定していないので，各ノードは自分自身で局所的に通過フロー数の情報を取得しなければならない．もしネットワーク内のフローが常にトラヒックを発生させていれば，到着パケットを観測することでノードを通過するフロー数を知ることができる．しかし，一般にはフローが一時的にパケットを発生させない状態になることがあり得るし，発生させていてもパケット間隔が広く，適当な時間区間ではパケットの発生がない場合も考えられる．したがって，自律分散的な装置構成をとる限り，各ノードは自分を通過する正確なフロー数を知ることができない．

本論文では，モジュール構成を考慮した ns2 の特性を利用し，自律分散で動作するモジュールとして DFC の実装を行う．更に，実装した DFC の振舞いとこれ

までのシミュレーション評価結果との違いから，通過フロー数の情報取得に関連する問題点を明らかにし，その解決策を示す．更に改良した DFC がねらいどおりの特性を示すことを明らかにすることで，DFC の実装可能性を確認する．

本論文の構成は以下のとおりである．**2.** で，これまで評価してきた DFC の概要を示し，**3.** で，既存の DFC を ns2 に実装する方法を示す．また **4.** で既存の DFC を ns2 に実装した場合の問題点を明らかにした後，**5.** で DFC の実装に向けた改良方法を示す．更に，**6.** で，改良した DFC を評価し，その効果を実証する．最後に **7.** でまとめる．

2. 拡散型フロー制御の概要

2.1 制御の基本コンセプト

インターネット型のネットワークにおいても，エンドツーエンドでの QoS 保証を要求するフローについては，RSVP のように特定の経路を用いた通信を行う．本論文では，このような特定の経路をもつフローを対象とし，ノードではフローごとのパケットキューイングが行われることを仮定する．

DFC は，各ノードが自分自身の知り得る局所情報のみに基づいて自律的に局所フローを制御し，間接的にネットワーク全体の状態を望ましい方向に誘導するフロー制御技術の枠組みである．具体的には，ネットワークがふくそうに陥ることをを回避したり，ふくそう状態から回復するように動作する．各ノードが自律的に動作することで，状況の変化に即応した高速な制御動作が可能となる．

DFC は，ノードの局所的自律動作によってネットワーク全体の状態を望ましい方向に誘導するために，物理の近接作用の考え方に基づいた制御の仕組みを採用している．これを熱拡散現象を例にして説明する．図 2 のように鉄棒の一点を熱すると，温度分布は正規分布となり拡散する．このとき，鉄棒の一点は鉄棒全体の温度分布に関する知識はなく，左右の温度差に比例した熱量を高温側から低温側に流しているにすぎない．しかしながら，温度分布は拡散方程式の解として秩序ある振舞いを示す．このように，システムの各部分が局所情報のみに基づいて自律動作をしつつも，システム全体の状態に秩序を生むことが可能である．この仕組みをネットワークに適用すると，各ノードの動作ルールを適切に決めておけば，ノードの自律動作により間接的にネットワークの全体性能を望ましい方向に誘導することが可能であると考えられる．DFC では各ノードが拡散現象に似たルールで局所的なパケットフローを制御することにより，一部のノードへのパケットの集中を回避し，パケットロスの起きにくい状態を維持するように動作する．

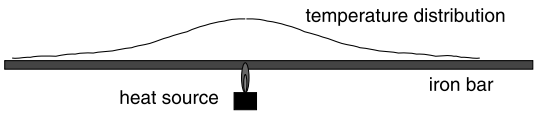

図 2　熱拡散現象の例
Fig. 2　Example of thermal diffusion phenomenon.

2.2 拡散型フロー制御方式

あるフローに沿って一次元的に連結したノード i（$i = 1, 2, \ldots, N$）を考え（図 3），ノード i とノード $i+1$ の間のリンクの帯域を B_i とする．また，ノード i とノード $i+1$ の間のリンクを共有するフロー数を M_i とする．ノード i は，時刻 t における下流ノード $i+1$ へのフロー j（$j = 1, 2, \ldots, M_i$）のパケット転送レート $J_i^j(t)$ を自分が知り得る情報のみに基づいて決定し，パケット転送を行う．また，上流ノード $i-1$ に向けてノード i の情報 $\mathbf{F}_i^j(t)$ をフィードバックする．パケット転送レート $J_i^j(t)$ の決定は，下流ノード $i+1$ から伝搬遅延 d_i を伴って通知される情報 $\mathbf{F}_{i+1}^j(t-d_i)$ の到着ごとに実行する．上流ノード $i-1$ に向けたフィードバック情報 $\mathbf{F}_i^j(t)$ の通知は，ノード $i-1$ と i 間の伝搬遅延 d_{i-1} に比例した時間間隔で行う．

以降では，パケット転送レート $J_i^j(t)$ とフィードバック情報 $\mathbf{F}_i^j(t)$ の算出方法について述べながら，DFC の全体像を説明するが，それに先立ち，DFC の説明に利用されるアクティブフローの概念について明らかにしておく．ノード i とノード $i+1$ の間のリンクを共有する，または共有する可能性のあるすべてのフロー数を M_i としたとき，ある時点で必ずしもすべてのフローにパケットが流れているとは限らない．本論文では，何らかの計測によってパケットが流れている

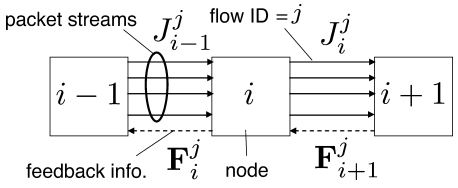

図 3　DFC の動作モデル
Fig. 3　Node interactions in our flow control model.

と判断されたフローをアクティブフローと呼ぶことにする(注1)．特定の時間帯においてあるノードがパケットの流れを観測したときに，実際にパケットが観測されるかどうかについては，音声の無音区間やパケットレートの高低などのトラヒックの特性に影響を受ける．このため，どのフローがアクティブと判断されるかは，時刻ごとやノードごとに一般に異なる結果となることがあり得る．

時刻 t，ノード i におけるフロー j の転送レート $J_i^j(t)$ は

$$J_i^j(t) = \max(0, \min(L_i^j(t), \tilde{J}_i^j(t))) \tag{1}$$

$$\tilde{J}_i^j(t) = r_i^j(t-d_i) - D_i\,(n_{i+1}^j(t-d_i) - n_i^j(t)) \tag{2}$$

によって決定する．ここで，$n_i^j(t)$ は時刻 t でノード i に存在するフロー j のパケット数，$r_i^j(t-d_i)$，$n_{i+1}^j(t-d_i)$ は下流ノード $i+1$ からのフィードバックにより，（リンク伝搬遅延 d_i を伴って）通知される情報で，それぞれ下流からの指示レート，下流ノード内のフロー j のパケット数を表す．L_i^j はノード i からノード $i+1$ に向かうリンクにおけるフロー j の利用可能帯域である(注2)．また D_i は D（>0）を定数として $D_i = D/d_i$ である．D は拡散係数と呼ばれ，$0 < D < 1/2$ の範囲の値をとる [11], [12]．フロー j に関するパケット転送レート $J_i^j(t)$ は，すべての $j = 1, 2, \ldots M_i$ について計算するのではなく，アクティブフローについてのみ行う．ここで得られたパケット転送レート $J_i^j(t)$ により，アクティブフローのパケット転送レートを更新する．アクティブフローでないフローのパケット転送レートは 0 とする．

一方，ノード i が生成するフロー j に関するフィードバック情報 $\mathbf{F}_i^j(t)$ は

$$\mathbf{F}_i^j(t) = (r_{i-1}^j(t), n_i^j(t)) \tag{3}$$

からなり，これを上流ノード $i-1$ に通知する．ここで，

$$r_{i-1}^j(t) = \max(0, \min(\tilde{J}_i^j(t), B_i/M_i^{\mathrm{act}}(t))) \tag{4}$$

$$M_i^{\mathrm{act}}(t) = \sum_{j=1}^{M_i} 1_{\{j=\mathrm{active}\}} \tag{5}$$

とする．ただし，$1_{\{j=\mathrm{active}\}}$ はフロー j がアクティブフローのとき 1，そうでなければ 0 となる指示関数であり，$M_i^{\mathrm{act}}(t)$ はその時刻 t でのアクティブフロー数を表す．つまり，あるフロー j の指定レート $r_{i-1}^j(t)$ の上限は，帯域 B_i をアクティブフロー数で割った値になる．

次に，DFC の直観的意味を説明する．議論を簡略化するため，ノード ID を連続化し，$i \to x$ で表す．また，連続極限をとってリンクの伝搬遅延を $d_i \to 0$ とする．このとき，$\tilde{J}_i^j(t) \to \tilde{J}(x,t)$，$r_i^j(t-d_i) \to r(x,t)$，$n_i^j(t) \to n(x,t)$ とすれば，式 (2) は

$$\tilde{J}(x,t) = r(x,t) - \kappa \frac{\partial n(x,t)}{\partial x} \tag{6}$$

の形で書ける．ここで $\kappa > 0$ は離散のときの D に対応する定数である．連続の式

$$\frac{\partial n(x,t)}{\partial t} = -\frac{\partial \tilde{J}(x,t)}{\partial x}$$

を用いると，パケット密度 $n(x,t)$ の時間発展は

$$\frac{\partial n(x,t)}{\partial t} = -\frac{\partial r(x,t)}{\partial x} + \kappa \frac{\partial^2 n(x,t)}{\partial x^2} \tag{7}$$

と書ける．これは拡散型の方程式である．したがって，DFC はノードが局所情報に基づいて自律動作をすることで，ネットワーク全体としてノード内パケット数を均一化する効果が期待できる．

2.3 ネットワーク入口でのパケットレート制限法

DFC がサポートされているネットワークを DFC がサポートされていないネットワークと接続するために，ネットワーク境界部分での境界条件を考える必要がある．

ネットワーク外のノードまたはエンドホストは拡散型フロー制御方式をサポートしていないことを前提とし，ネットワーク入口での入力パケットレートは，フィードバック情報 $r_0^j(t-d_0)$ で指定されたレートにシェーピング（バッファで遅延させることによるレートの調整）がされるものとする．ノード $i=1$ が通知する情報 $r_0^j(t)$ は，ネットワーク外のノードまたはエンドホストのバッファに存在するパケット数を 0 と仮定し，式 (2) によってレート $\tilde{J}_0(t)$ を計算する．つまり，$r_0^j(t)$ は式 (4) の代わりに，

$$r_0^j(t) = J_1^j(t) - D_0\, n_1^j(t) \tag{8}$$

を用いて決定する．$r_0^j(t)$ はノード $i=1$ で取得可能

(注1)：アクティブフローの判定方法は後述する．
(注2)：L_i^j の決定方法は **2.4** で述べる．

な量のみによって計算することができる．これにより，ネットワーク外からの入力パケットレートとネットワーク内部の DFC によるレートとの親和性を高めることができる．

しかし，DFC は局所情報に基づくノードの自律動作を用いた制御なので，ネットワーク内の状態を高速に平滑化させる効果には優れるが，ネットワーク内に外部から流入する余分なパケットを防ぐ効果は少ない．そこで，必要に応じてネットワーク入口でパケットレートを制限する機能を併用し，ネットワーク入口でのパケット流入量を適切に制限することが必要となってくる．これを実現するためには，ネットワーク入口部分でフローに沿った経路上のネットワーク状態を把握し，その状態に合わせてネットワーク入口でのパケットシェーピングを施すことが有効である．つまり，ノード i が生成するフィードバック情報 $\mathbf{F}_i^j(t)$ にリンクの最大利用可能帯域に関する情報 $\ell_i^j(t)$ を加え，

$$\mathbf{F}_i^j(t) = (r_{i-1}^j(t), n_i^j(t), \ell_i^j(t)) \tag{9}$$

を上流ノード $i-1$ に通知する．$\ell_i^j(t)$ は，

$$\ell_i^j(t) = \min(\ell_{i+1}^j, B_i/M_i^{\mathrm{act}}(t)) \tag{10}$$

のように生成する．$\ell_i^j(t)$ の計算は，下流のリンクのうちで一番小さい（とノード i が認識する）帯域の値を用いる．この値で，式 (8) の値を制限することにより，ネットワーク内に外部から流入する余分なパケットを防ぐことができる．

2.4 各フローの利用可能帯域の決定方法

リンク帯域は複数のフローで共有されているため，フロー間で適切に $L_i^j(t)$ を調節する必要がある．

ノード i からノード $i+1$ へのリンクの帯域を B_i としたとき，$L_i^j(t)$（$j = 1, 2, \ldots, M_i$）は，

$$\sum_{j=1}^{M_i} 1_{\{j=\mathrm{active}\}} \times L_i^j(t) = B_i \tag{11}$$

$$L_i^j(t) = 0 \quad (j \neq \mathrm{active}) \tag{12}$$

を満たすとする．これは，アクティブフローのみの利用可能帯域で全帯域を分け合うことを意味する．

DFC では，転送レートの理想値は $\tilde{J}_i^j(t)$ であり，$J_i^j(t)$ は理想値 $\tilde{J}_i^j(t)$ を物理的に利用可能な帯域を上限として制限したものである．多くのフローがある場合，理想の転送レート $\tilde{J}_i^j(t)$ は，$\sum_{j=1}^{M_i} \tilde{J}_i^j(t) > B_i$ となり，すべてのフローがそれぞれの理想の転送レート $\tilde{J}_i^j(t)$ を使用することができず，パケット密度のスムーズな平滑化を妨げる原因となり得る．$L_i^j(t)$ の最も単純な決定方法は，アクティブフローごとに重み $\tilde{J}_i^j(t)$ を付けて帯域 B_i を分け合うことである．つまり，それぞれのフローの重み $W_i^j(t)$ を

$$W_i^j(t) = \frac{\tilde{J}_i^j(t)}{\sum_{k=1}^{M_i} 1_{\{k=\mathrm{active}\}} \times \tilde{J}_i^k(t)} \tag{13}$$

とし，それぞれのフローの利用可能帯域を

$$L_i^j(t) = B_i \times W_i^j(t) \tag{14}$$

とすることで，より大きい重み $\tilde{J}_i^j(t)$ をもったフローがより大きい転送レートを確保でき，下流ノードにより多くのトラヒックを流すことができることになる．このとき，重みの小さなフローは相対的に他のフローの転送レートはより小さく制限されることになる．これによりフロー間で帯域が分配され，共通経路上にあるノード内パケット数のフロー間のばらつきが平滑化される．また，フローごとに利用可能帯域が適切に調整されるため，ボトルネックに対してフローの下流方向にノード内パケット数を平滑化する効果も得られる．

3. ns2 に実装された DFC の概要

本章では，**2.** で示した既存の DFC 機能を実装した ns2 の装置構成を示す．

3.1 各モジュールの概要

ns2 に実装された DFC 機能は主に四つのモジュールから構成されている（図 4）．以下にそれぞれの機能の概要を示す．

- `DFCFlowQ`

受信したパケットをフローごとにキューイングする．また，あるフローのパケットが `DFCQueue` に存在しなければ，そのフローに関する `DFCFlowQ` の先頭パケットを `DFCDBAgent` により定められた転送レート $\tilde{J}_i^j(t)$ で `DFCQueue` へ転送する（図 5 参照）．

- `DFCQueue`

`DFCFlowQ` から受信したパケットをリンクへ送出する．パケットを送出するフローは，各フローに定められた重みにより weighted round robin（WRR）で決定される（図 5 参照）．

- `DFCDBAgent`

下流ノードからのフィードバック情報が記された DFC パケットを受信し，受信した値と自ノードの情報をもとに自ノードのフィードバック情報を算出する．

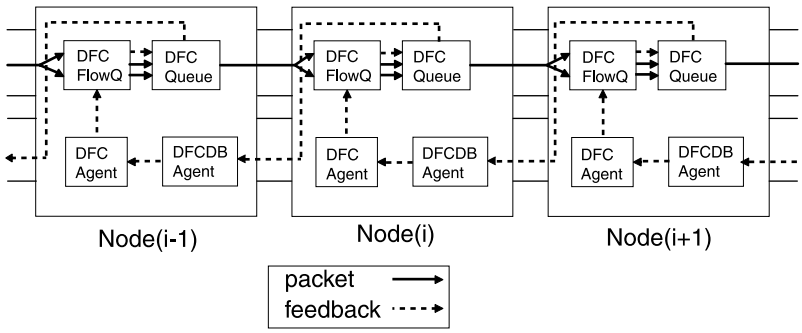

図 4　DFC のモジュール構成
Fig. 4　Modules of DFC function.

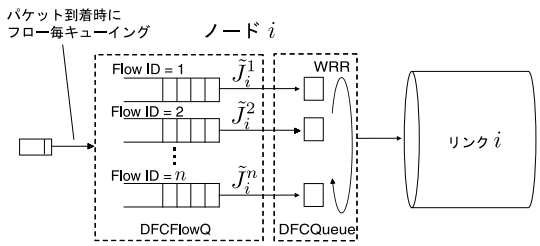

図 5　WRR 方式
Fig. 5　WRR scheduling-based rate control.

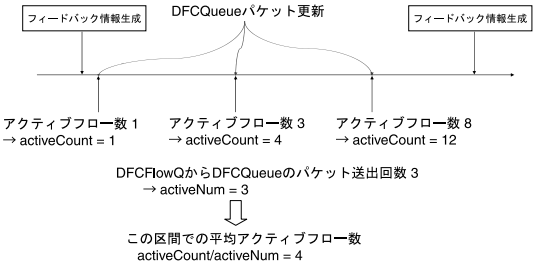

図 6　従来のアクティブフロー数の測定
Fig. 6　Measurements of the average number of active flows.

- `DFCAgent`

`DFCDBAgent` により算出されたフィードバック情報をパケット化し，`DFCFlowQ` を通して上流ノードへ送出する．

3.2　`DFCQueue` からリンクへのパケット送出方法

`DFCFlowQ` から `DFCQueue` へのパケット転送レートは $\tilde{J}_i^j(t)$ である．これは単一のフロー j に着目して算出された値であり，`DFCQueue` からリンクへのパケット送出時に，式 (1) のようにレート $\tilde{J}_i^j(t)$ はフロー j の利用可能帯域 $L_i^j(t)$ によって $J_i^j(t)$ に制限される．これを実現するために，これまでの実装では `DFCQueue` からリンクへのパケット送出方法を重み $J_i^j(t)$ の WRR 方式としている．

3.3　アクティブフロー数の測定方法

式 (4) と式 (10) で用いられている $M_i^{\mathrm{act}}(t)$ の測定方法について説明する．

`DFCFlowQ` から `DFCQueue` に送出が行われるごとに，パケットが到着したフロー数をアクティブフローとして測定し，フィードバック情報生成に必要なアクティブフロー数 $M_i^{\mathrm{act}}(t)$ を計算するための値を保持する．具体的には，

`activeNum`

　＝ (`DFCFlowQ` から `DFCQueue` への送出回数)

`activeCount`

　＝ (パケット到着のあったフロー数の累計)

とし，図 6 のようにして，`activeCount`/`activeNum` によりアクティブフロー数 $M_i^{\mathrm{act}}(t)$ を測定している．また，`activeNum`，`activeCount` はフィードバック情報が生成されるごとにリセットされる．この方法では，フィードバック情報が生成される間の平均アクティブフロー数を求めていることになる．

4.　DFC 実装上の問題点と原因の分析

本章では，自律分散制御としての装置構成で既存の DFC の動作特性を評価し，問題点抽出と原因の分析を行う．具体的には，既存の DFC が実装された ns2 を用いて，単純なモデルでシミュレーションを実行し，DFC の動作特性を分析する．また，本論文の評価では，DFC のパラメータを $D = 0.4$，フィードバック情報 $\mathbf{F}_i^j(t)$ の送信間隔を d_{i-1} とした．

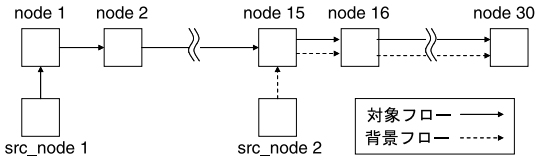

図 7　シミュレーションモデル
Fig. 7　Simulation model.

4.1　シミュレーションモデルとシナリオ

シミュレーションに用いるモデルは，あるフローに沿った一次元ネットワークを考える（図 7 参照）．ノード数を 30，各リンクの伝搬遅延を 0.1 ms でリンク長は固定，各ノードのバッファ容量 1800 パケット，1 パケットは固定長 1500 Byte で，1 リンク上の最大パケット数は 100 パケットとした．ネットワーク内のノードは DFC の機能をもつ．また，パケットの発生源としてソースノード 1 と 2 を配置した．これらは DFC 方式適用外（シェーピング機能はあり）のノードである．各ソースノードのパケット生成レートは，フロー制御がかかっていない状態で単位時間当り 100 パケットとした．

シミュレーションシナリオは次のとおりである．ネットワーク上にはフロー $j = 1$ とフロー $j = 2$ の二つの TCP フローを流す．ソースノード 1 で発生するフロー $j = 1$ は性能評価対象のフローで，ノード 1 からノード 30 へと流れる．ソースノード 2 で発生するフロー $j = 2$ はフロー $j = 1$ の妨げとなる背景フローで，対象フローの経路上のノード 15 に流入し，ノード 30 へと流れる．TCP の window size は両フローともに RTT の bandwidth-delay product より十分大きい値として 10,000 とした．フロー $j = 1$ は $t = 0.0$ s，フロー 2 は $t = 0.06$ s にそれぞれ送信を始め，$t = 0.12$ s でシミュレーション終了とする．フロー $j = 2$ が送信され始めると，ノード 15 からノード 16 へのリンクはボトルネックとなり，二つのフローは DFC のルールによって転送レートが規制されることになる．シミュレーション結果より背景フローが流入直後からのネットワーク状態の時間変化を調べることで，DFC の従来の実装について性能を評価する．

4.2　シミュレーション評価結果

フロー $j = 2$ が流入した直後からの，フロー $j = 1$ のネットワーク内のノード上パケット数の振舞いを調べた結果が図 8 であり，また各時間のノード上パケット数の総和を示したものが図 9 である．図 8 の横軸はフローの経路に沿ったノード ID を示しており，縦軸はそのノード内パケット数を表している．グラフごとに異なる時刻の状態を表示している．図 9 の横軸は時刻を示しており，縦軸はその時刻でのノード内パケット数の総和を表している．図 8 から，時刻 $t = 0.06$ s にフロー 2 が流入したことによるふくそうが，$t = 0.073$ s でいったん解消していることが分かる．しかし，その後 $t = 0.082$ s で再び小さなふくそうを起こしている．図 9 によれば，1 度目のふくそうが解消された後に，2 度目，3 度目の小さなふくそうが繰り返し起こっていることが分かる．つまり，ネットワーク内のノードに存在するパケット数が平滑化し，いったんふくそうが解消したにもかかわらず，何らかの原因で再度ふくそう状態が引き起こされていることになる．

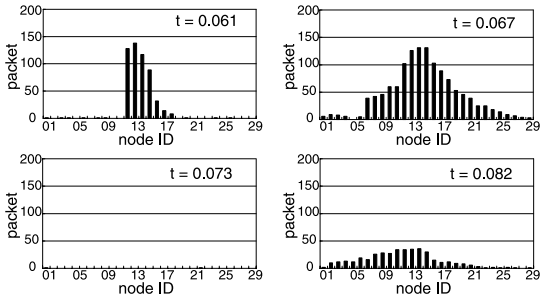

図 8　ノード内に存在するパケット数分布の経時変化
Fig. 8　Temporal evolution of the distribution of stored packets in each node.

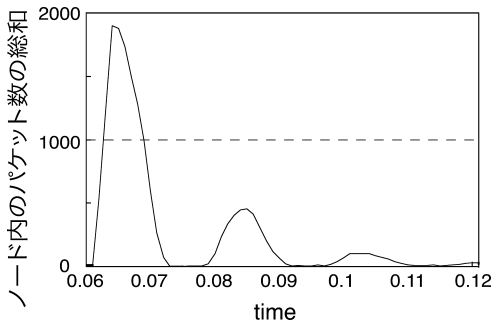

図 9　ノード内パケット数の総和の経時変化
Fig. 9　Temporal evolution of the total number of stored packets in nodes.

次に，フロー $j = 1$ のリンク上に存在するパケット数の総和の時間推移を図 10 に示す．この評価は，ネットワークがフロー 1 のパケットをどれだけ運んでいるかを示している．シミュレーション条件より 1 本のリ

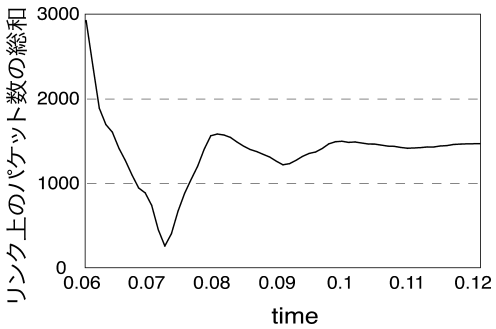

図 10　リンク上パケット数の総和の経時変化
Fig. 10　Temporal evolution of the total number of packets on links.

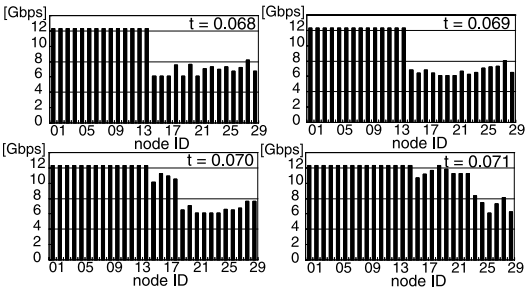

図 11　$B_i/M_i^{\mathrm{act}}(t)$ の値の時間変化
Fig. 11　Temporal evolution of B_i/M_i at each nodes.

ンク上に乗る最大パケット数が 100 なので，リンクの帯域がフロー $j=1$ とフロー $J=2$ で公平に分けられたとするならば，フロー $j=2$ が流入した後の，フロー $j=1$ のリンク上のパケット数の総和の適切な値は，約 1450 である．図 10 から，2 度目，3 度目のふくそうが起こっていた付近で，リンク上のパケット数の総和の値が適切な値を超えていることが分かる．

これまでの評価（ノードは通過フロー数を知っているという状況下）では，1 度ふくそうが解消されると再度ふくそうが現れるということはなく，ネットワーク性能は安定していた．今回の評価において複数回ふくそうが起こる原因として，ネットワーク内への過剰なパケット流入と，フロー $j=1$ のノード 15 への過剰なパケット送出の 2 点が考えられる．

ネットワーク内へのパケット流入量を決定するときに用いられるのは，$\ell_i^j(t)$ であり，各ノードの転送レートの決定するときに用いられるのは $r_i^j(t)$ である．双方ともふくそうが複数回起きる原因となっているので，それぞれの値の決定に共通して用いられている $M_i^{\mathrm{act}}(t)$ の決定方法に問題がある可能性が高い．これを確かめるため $B_i/M_i^{\mathrm{act}}(t)$ の値がどのように変化しているかを検証する．図 11 は各ノードでのフロー当りの利用可能帯域 $B_i/M_i^{\mathrm{act}}(t)$ をノードごとに表示し，グラフごとに異なる時間の状態を表示したものである．各グラフの縦軸は利用可能帯域 $B_i/M_i^{\mathrm{act}}(t)$，横軸はノード ID を表す．ノード 1 から 14 まではフロー $j=1$ の 1 本しか存在しないため，$B_i/M_i^{\mathrm{act}}(t) \simeq 12\,\mathrm{Gbit/s}$ 程度となり，ノード 15 以降は 2 本のフローで帯域を分け合うため，その半分程度の値となるのが適切である．この図より，1 度目のパケット平滑化が行われた後，適切な値から大きく外れていくことが確認できる．これらのことから，$\ell_i^j(t)$ と $r_i^j(t)$ の決定に用いた $M_i^{\mathrm{act}}(t)$ の振舞いが，DFC の適切な実装を妨げていることが分かる．

5. DFC の実装に向けた方式の改良

前章で，DFC を実装するためには $\ell_i^j(t)$ と $r_i^j(t)$ の決定に用いられていた $B_i/M_i^{\mathrm{act}}(t)$ の値が DFC のねらいどおりに変化していないことが分かった．これまで，シミュレーションモデルを用いた DFC の基本動作特性の評価では，フィードバック情報計算の対象となるフロー数，つまりノードを通過するアクティブフロー数 $M_i^{\mathrm{act}}(t)$ の情報はシミュレーションの初期設定のパラメータとして決定され，ノードにとって既知の情報としていた．このようなシミュレーション条件では，アクティブフロー数 $M_i^{\mathrm{act}}(t)$ の決定が不適切であっても DFC の動作に深刻な影響を与えることはなかった．しかし，モジュール構成を意識した DFC の実装では，ノードがアクティブフロー数 $M_i^{\mathrm{act}}(t)$ の情報を知ることができるとは限らない．そのため DFC を実装するモジュール内に，ノード自身がアクティブフローに関する情報を取得する仕組みを組み込まなければならない．

本章では $\ell_i^j(t)$ と $r_i^j(t)$ の決定方法の考察とノード自身がアクティブフローに関する情報を取得する仕組みの考察を行い，DFC の方式改良を行う．

5.1 $\ell_i^j(t)$ と $r_i^j(t)$ の制限方法

式 (2) の転送レート $\tilde{J}_i^j(t)$ はネットワーク内のノード内パケット数平滑化のために計算された値であるが，フロー j のみに着目して決定した値であり，他のフローの影響を考慮に入れていない．そのため，実際のパケット送出のレート $J_i^j(t)$ は，式 (1) に示すように $L_i^j(t)$ によって $\tilde{J}_i^j(t)$ を制限した値となっている．

式 (14) から分かるように，$L_i^j(t)$ はノード内の他のフローの転送レートの値を考慮して算出されたものである．

これまで，$\ell_i^j(t)$ と $r_i^j(t)$ の決定には $B_i/M_i^{\rm act}(t)$ の値で上限を決めていたが，転送レートの決定式 (1) と同様に $L_i^j(t)$ の値を上限値として用いれば，他のフローの影響を考慮に入れた DFC の転送レート決定法と整合がとれる．つまり，$\ell_i^j(t)$ と $r_i^j(t)$ の値をそれぞれ，

$$\ell_i^j(t) = \min(\ell_{i+1}^j(t), L_i^j(t)) \tag{15}$$

$$r_i^j(t) = \max(0, \min(\tilde{J}_i^j(t), L_i^j(t))) \tag{16}$$

とすれば，ネットワーク全体のパケット平滑化というねらいが考慮された値になると考えられる．$L_i^j(t)$ を適切に決定するにはアクティブフローの判定が必要であり，逆にアクティブフローの判定が適切に実行できれば，自然にアクティブフロー数 $M_i^{\rm act}(t)$ の値も明らかになる．そのため，以降ではアクティブフローの判定法と $L_i^j(t)$ の決定法を考察する．

5.2 アクティブフロー数の測定法と $L_i^j(t)$ の決定法

フィードバック情報が生成される間にどのフローがアクティブフローであるかを測定する方法は，従来のアクティブフロー数の測定方法を変更することで実現できる（図 12）．具体的には，まず DFCQueue がパケットを更新するごとに，それぞれのフローについて，DFCFlowQ と DFCQueue にパケットが存在しているか否かを調べる．調べたフローにパケットが存在するならば，そのフローのフロー情報にフラグを立てる．その後，DFCDBAgent がフィードバック情報を生成するごとにフラグをリセットする．この操作を行うことで，フィードバック情報が生成される間にどのフローがアクティブであったかを測定することができる．

DFCDBAgent はこの得られた情報を参考に，パケット転送レートやフィードバック情報の計算対象となるアクティブフローを特定することができる．また，式 (13) により $W_i^j(t)$ を計算し，式 (14) により $L_i^j(t)$ を決定することができる．

5.3 パケット送出方法に関する考察

DFC の実ネットワークへの実装を考えるとき，DFC はデータリンク層で実現され，DFCQueue は例えばネットワークカード内に実装されることが考えられる．このため，もしリンクへのパケット送出方法を簡略することができれば，既存のネットワークカードに DFC を実装することが容易になると考えられる．

従来の DFC では，DFCFlowQ から DFCQueue へのパケット転送レートを $\tilde{J}_i^j(t)$ とし，DFCQueue からリンクへの転送レートを WRR を用いて $J_i^j(t)$ に制限していた．しかし，今回の改良によりアクティブフロー数の情報を DFCDBAgent が利用できるようになるので，DFCFlowQ から DFCQueue へのパケット転送レートを $J_i^j(t)$ とすることが可能となった．この $J_i^j(t)$ は，下流のリンク帯域と，他のフローの転送レートを考慮に入れた値となっているので，DFCQueue からリンクへのパケット送出をよりシンプルな Round Robin（RR）や Single Server Queue に変更しても，従来の性能を保つことができると考えられる（図 13 参照）．DFC 機能を実装を考えると，DFCQueue では単純な FIFO queue を用いた実装が可能となり，ハードウェアの簡略化が実現することに結び付く．ここで，DFCFlowQ から DFCQueue へのパケット転送レート $\tilde{J}_i^j(t)$ が

$$\sum_{j=1}^{M_i} 1_{\{j=\text{active}\}} \times \tilde{J}_i^j(t) < B_i \tag{17}$$

を満たすならば，アクティブフローでないフローのパケットであっても DFCFlowQ から DFCQueue へ転送することができる．そのときのパケット転送レートは $B_i - \sum_{j=1}^{M_i} 1_{\{j=\text{active}\}} \times \tilde{J}_i^j(t)$ となる．このため，ノード i においてアクティブフローでないと判断され，

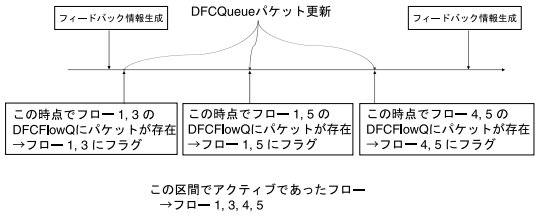

図 12　アクティブフロー数の測定
Fig. 12　Measurement of the number of active flows.

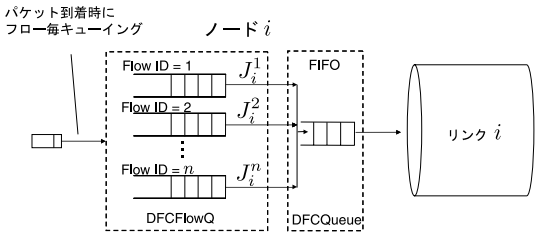

図 13　FIFO（Single Server Queue）方式
Fig. 13　Single server queue-based rate control.

パケット転送レート 0 が割り当てられたパケットがノード i に到着したとしても，混雑していなければ次のレート更新時刻を待たずに転送可能である．

6. 評　　価

上記の改良を行った ns2 を用いてシミュレーションを行い，自律分散制御の装置構成をもつ DFC 機能が適切な性能を発揮することを確認することで，改良した DFC の妥当性を評価する．なお，今回のシミュレーションでは，`DFCQueue` からリンクへのパケット送出方法を Single Server Queue に変更した．

6.1 ふくそうの繰返しを避ける効果についてのシミュレーションの評価結果

まず，**4.1** で用いたモデルとシナリオを用いて，ふくそうの繰返しを避ける効果についての評価を行う．図 14 と図 15 はそれぞれ改良前と改良後のフロー $j = 1$ の，ノード上パケット数の総和の時間推移，リンク上パケット数の総和の時間推移を示したものである．それぞれ，実線が改良後，破線が改良前の値を示している．これを見ると，複数回起こっていたふくそうが起こらなくなり，それに伴ってリンク上のパケット数も改良前に比べて早い段階で安定した値になっていることが分かる．

次に，ノード上の総パケット数及びリンク上の総パケット数に関して，DFC 機能による収束の速さを改良前後で比較する．図 16 は，横軸がふくそうが開始してからの時間，縦軸がその時間以降のノード上のパケット数の総和の最大値を表す．実線が改良後，破線が改良前の値を示している．この図から改良後は改良前に比べてノード上の総パケット数が速やかに減少していることが分かる．例えば，ノード上のパケット数の総和が 30 個以下に収束する時間は，改良前では 0.06 s，改良後では 0.015 s であり，改良前の約 25% まで削減できる．また図 17 は，横軸がふくそうが開始してからの時間，縦軸がリンク上のパケット数の総和と理想値（1450）との差の比率を表す．この結果から，

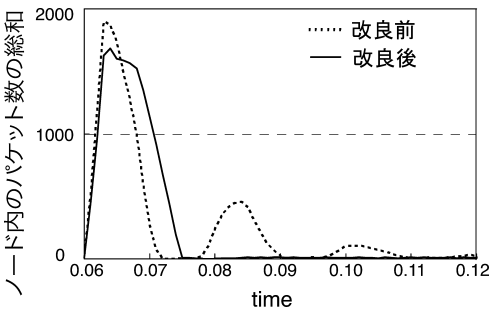

図 14　ノード上のパケット数の総和の時間変化の比較
Fig. 14　Comparison of temporal evolution of the total number of stored packets in nodes.

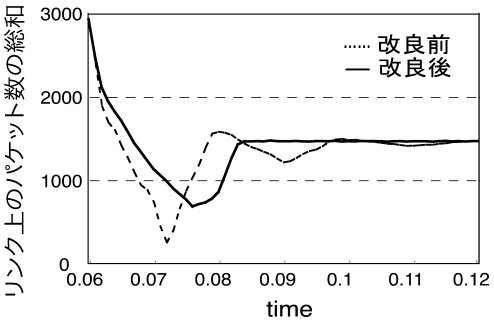

図 15　リンク上のパケット数の総和の時間変化の比較
Fig. 15　Comparison of temporal evolution of the total number of packets on links.

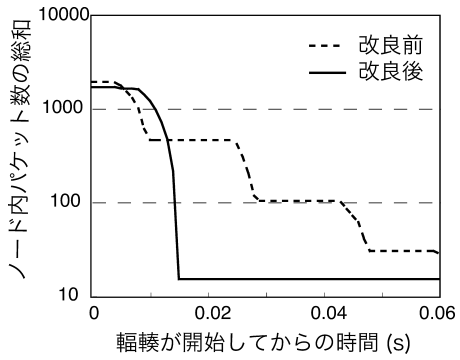

図 16　ノード上のパケット数の総和の収束の比較
Fig. 16　Comparison of the convergence of the total number of stored packets in nodes.

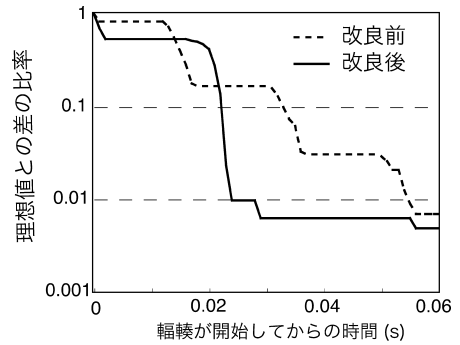

図 17　リンク上のパケット数の総和の収束の比較
Fig. 17　Comparison of the convergence of the total number of packets on links.

改良後のリンク上のパケット数の総和が改良前に比べて素早く理想値に近づいていることが分かる．例えば，理想値との誤差が 1%未満になる時間は，改良前では 0.055 s，改良後では 0.024 s であり，改良前の約 40%まで削減できる．

これらのことから，$\ell_i^j(t)$ と $r_i^j(t)$ の決定に用いられていた $B_i/M_i^{\rm act}(t)$ を $L_i^j(t)$ に変更することで，フロー $j=1$ の転送レートの値が適切な値となり，またネットワーク入口部分のシェーピング機能がうまく働き，ネットワークへの過剰なパケット流入が抑えられ，ふくそうが複数回起こるという現象を解決できたことが分かった．

6.2 アクティブフロー数の適切な管理についてのシミュレーションモデルと評価結果

ここでは，アクティブフロー数が適切に管理されているかについての評価を行う．この評価では，これまで用いたモデルを以下のように変更した．ノード 23 に新たにソースノードを接続し，$t=0.09$ s からノード 30 に向けてパケット送出を開始する（このフローをフロー $j=3$ とする）．更に同じ時刻に，フロー $j=2$ のパケット生成を停止させる．

まず，ネットワーク内からフロー $j=2$ のパケットが消滅するまでの時間を調べるために，フロー $j=2$ のリンク上パケット数の総和の時間推移についてのグラフを示す（図 18）．この図からフロー $j=2$ のすべてのパケットがノード 30 に到達した時刻が約 $t=0.15$ s であることが分かる．この時刻以降のフロー $j=1$ の転送レート $J_i^1(t)$ の変化を調べたものが図 19 である．横軸はノード ID，縦軸はそのノードの転送レート $J_i^1(t)$ を表し，それぞれのグラフは異なる時刻の状態を表示している．

フロー $j=2$ のパケットがネットワーク内から消滅すると，ノード 30 からノード 1 に向かって転送レート $J_i^1(t)$ の値がフローが 2 本の場合の値（このとき存在するフローは $j=1, 3$）に回復していき，$t=0.156$ s にすべてのノードの転送レートの値が回復したことが分かる．つまり，$t=0.156$ s 付近からフロー $j=1$ のリンク上パケット数の総和の値が，1450 パケット付近まで回復していれば，それぞれのノードでアクティブフロー数が適切に管理されていることが分かる．

図 20 はフロー $j=1$ のリンク上パケット数の総和の時間推移を示したものである．これを見ると，それぞれの時刻のフロー数に応じて，リンク上パケット数の総和が適切に変化していることが分かる．具体的には，アクティブフローが二つである $t=0.06$ s から $t=0.09$ s にかけては 1450 パケット付近，アクティブフローが三つである $t=0.09$ s から $t=0.150$ s にかけては 966 パケット付近，アクティブフローが一つ減

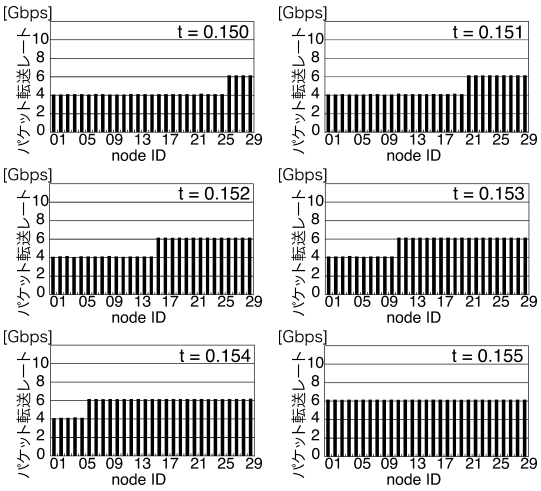

図 19 転送レート J_i^1 の時間変化
Fig. 19 Temporal evolution of the transmission rate J_i^1 at each node.

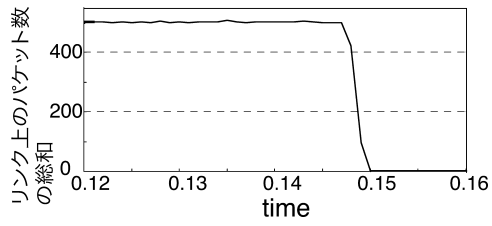

図 18 フロー $j=2$ のリンク上パケット数の総和の時間変化
Fig. 18 Temporal evolution of the total number of flow 2's packets on links.

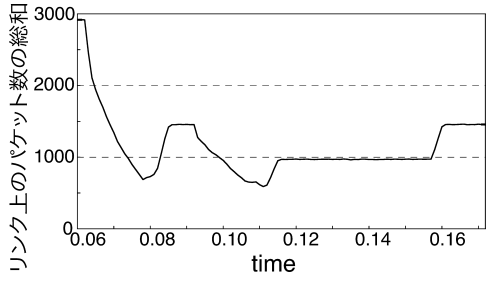

図 20 フロー $j=1$ のリンク上パケット数の総和の時間変化
Fig. 20 Temporal evolution of the total number of flow 1's packets on links.

少した $t = 0.150$ s 以降は 1450 パケット付近となっていることが分かる．

これらのことから，ノード上でのアクティブフロー数が適切に管理されていることが確認でき，本研究で示したアクティブフロー数に関する情報の取得方法が適切であることが確認された．

7. むすび

拡散型フロー制御（DFC）とは，拡散現象を指導原理としたフロー制御で，局所的なネットワーク情報に基づいたノードの自律動作が，間接的にネットワーク全体の状態を良好な特性に導く制御方式である．これまで，DFC がねらいどおりの特性をもつことがシミュレーションにより確かめられてきたが，その評価ではノードを通過するフロー数が既知であることを用いていた．実ネットワークに DFC を実装するためには，ノードを通過するアクティブフロー数の情報をノードが自律的に取得できる枠組みにする必要がある．

本論文では DFC の実装可能性を確かめるため ns2 に DFC 機能を組み込み，自律分散制御として実装可能であるかどうか検討を行った．その結果，従来の DFC の枠組みではノードの通過フロー数の情報を適切に取得することができないため，ネットワーク内に流入するパケット数を制御する機構が正しく動作せず，ネットワークがふくそう状態から回復するまでに時間がかかってしまうことが分かった．これを解決するために，フィードバック情報として通知される転送レートの制限方法にを改良し，更にノード自身が通過するアクティブフロー数を計測する方法を改良した．この改良により，DFC の実装に関して従来 weighted round robin が必要であったパケットスケジューリング部分を，単純な FIFO で置き換える単純化が可能となった．また，改良した DFC の特性を ns2 によりシミュレーションを行い，その効果を実証した．

謝辞　本研究の一部は，情報通信研究機構（NICT）の委託研究「新世代ネットワークの構成に関する設計・評価手法の研究開発」により実施した．

文　献

[1] Y. Bartal, J. Byers, and D. Raz, “Global optimization using local information with applications to flow control,” Proc. 38th Ann. IEEE Symp. on Foundations of Computer Science, pp.303–312, Oct. 1997.

[2] S.H. Low and D.E. Lapsley, “Optimization flow control-I: Basic algorithm and convergence,” IEEE/ACM Trans. Netw., vol.7, no.6, pp.861–874, 1999.

[3] K. Kar, S. Sarkar, and L. Tassiulas, “A simple rate control algorithm for maximizing total user utility,” Proc. IEEE INFOCOM 2001, pp.133–141, 2001.

[4] J. Mo and J. Walrand, “Fair end-to-end window based congestion control,” IEEE/ACM Trans. Netw., vol.8, no.5, pp.556–567, Oct. 1999.

[5] S. Kunniyur and R. Srikant, “A decentralized adaptive ECN marking algorithm,” Proc. IEEE GLOBECOM ’00, pp.1719–1723, 2000.

[6] R. Johari and D. Tan, “End-to-end congestion control for the Internet: Delays and stability,” IEEE/ACM Trans. Netw., vol.9, no.6, pp.818–832, Dec. 2001.

[7] C. Takano and M. Aida, “Stability and adaptability of autonomous decentralized flow control in high-speed networks,” IEICE Trans. Commun., vol.E86-B, no.10, pp.2882–2890, Oct. 2003.

[8] C. Takano, M. Aida, and S. Kuribayashi, “Autonomous decentralized flow control in high-speed networks with inhomogeneous configurations,” IEICE Trans. Commun., vol.E87-B, no.6, pp.1551–1560, June 2004.

[9] C. Takano and M. Aida, “Diffusion-type autonomous decentralized flow control for end-to-end flow in high-speed networks,” IEICE Trans. Commun., vol.E88-B, no.4, pp.1559–1567, April 2005.

[10] C. Takano, K. Muranaka, K. Sugiyama, and M. Aida, “Mutual complementarity of diffusion-type flow control and TCP,” IEICE Trans. Commun., vol.E89-B, no.10, pp.2850–2859, Oct. 2006.

[11] C. Takano, K. Muranaka, K. Sugiyama, and M. Aida, “Parameter design for diffusion-type autonomous decentralized flow control,” Lecture Notes in Computer Science, vol.4238, pp.461–470, Springer-Verlag Heidelberg, 2006.

[12] C. Takano and M. Aida, “Diffusion-type autonomous decentralized flow control for multiple flows,” IEICE Trans. Commun., vol. E90-B, no.1, pp.21–30, Jan. 2007.

[13] The Network Simulator—ns2. http://www.isi.edu/nsnam/ns/

（平成 20 年 1 月 15 日受付，5 月 13 日再受付）

高野　知佐　（正員）

平 12 阪大・工・電子通信卒．平 20 首都大東京大学院博士後期課程了．平 12 NTT アドバンステクノロジ（株）入社．平 20 広島市大大学院・情報科学研究科・准教授．通信トラヒック制御，自律分散制御技術の研究に従事．平 14 本会学術奨励賞，平 15 本会情報ネットワーク研究賞受賞．

山内　正志　（正員）

平 18 都立科技大・工・生産情報システム工卒．在学中は自律分散制御技術の研究に従事．現所属はトーワシップ債権回収（株）．

会田　雅樹　（正員）

昭 62 立教大・理・物理卒．平元同大大学院博士課程前期原子物理学専攻了．同年日本電信電話（株）入社．平 17 首都大東京・システムデザイン学部・准教授．平 19 同大大学院・システムデザイン研究科・教授．博士（工学）．通信トラヒックの設計技術，通信品質計測技術，自律分散制御技術，通信トラヒックのべき乗則の研究に従事．日本 OR 学会，IEEE 各会員．