ボトルネックリンクの利用率に基づいたEnd-to-End輻輳制御

内海哲史[†1]　サリム　ザビル[†2]

従来の End-to-End 輻輳制御 (TCP NewReno など) では，ネットワークにおけるパケット損失によって，ネットワーク輻輳を検出していた．しかし，パケット損失は，ネットワーク輻輳だけではなく，無線ネットワークにおけるリンクエラーによるものもある．従来の End-to-End 輻輳制御では，パケット損失時，輻輳ウィンドウを縮小させるため，無線ネットワークにおけるリンクエラー時，性能が大きく低下していた．CHD アルゴリズムは，ネットワーク輻輳の検出を，パケット損失だけでなく，パケットのキューイング遅延の測定結果にも依存する新しい End-to-End 輻輳制御である．本稿では，ボトルネックリンクの利用率の測定結果にも基づいて動作する輻輳制御 Utilization-based Congestion Control(UCC) を提案する．

An End-to-End Congestion Control Based on Link Utilization

Satoshi Utsumi[†1] and S.M. Salim Zabir[†2]

Conventional end-to-end congestion controls like TCP NewReno detects the network congestion by packet losses. However, packet losses are generated by not only network congestion but also link error over wireless links. Since whenever a packet loss occurs, conventional end-to-end congestion control reduces its congesition window, performance is down over wireless networks. Caia-Hamilton Delay-based (CHD) Algorithm detects network congestion depending on queuing delay in addition to segment loss. We propose the new end-to-end congestion control, named as Utilization-based Congestion Control (UCC) in this paper, which algorithm attempts to detect network congestion by measuring utilization on bottleneck link with the queuing theory.

†1 鶴岡工業高等専門学校
Tsuruoka National College of Technology
†2 フランステレコム株式会社
France Telecom Japan

1. はじめに

近年，携帯電話や無線デバイスがインターネットにおいて急速に普及してきている．そのため，無線リンクにおいても，アプリケーションが満足に動作する必要性が増してきている．無線リンクは，有線ネットワークよりも高いリンクエラー率を持つ．TCP NewReno などの従来の End-to-End 輻輳制御は，有線ネットワークを想定して設計されているため，ネットワークにおけるパケット損失は，ネットワーク輻輳によってのみ発生すると仮定している．従来の End-to-End 輻輳制御では，パケット損失を検出したとき，ネットワーク輻輳が起きたと推測し，送り手は輻輳ウィンドウを縮小する．そのため，従来の End-to-End 輻輳制御が無線通信において使用されると，パケット損失がリンクエラーによって発生したとき，送り手はネットワーク輻輳が起きたと判断し，不必要に輻輳ウィンドウを縮小する．このようにして，無線通信における従来の End-to-End 輻輳制御は，スループットが大きく低下してしまう．

これまで，様々な End-to-End 輻輳制御が提案されており[1)]，無線ネットワークや衛星ネットワークでも高性能に機能する End-to-End 輻輳制御として TCP-Cherry[2)3)] などがあるが，本稿では，パケットのキューイング遅延によって，ネットワーク輻輳を検出する技術に着目し，無線ネットワークにおける新しい End-to-End 輻輳制御を提案する．特に，待ち行列理論を用いたボトルネックリンクの利用率の推定と，その無線ネットワークにおける End-to-End 輻輳制御アルゴリズムへの応用について述べる．

2 章では，キューイング遅延に着目した輻輳制御に関する関連研究についてまとめる．特に，HD 輻輳制御アルゴリズムと CHD 輻輳制御アルゴリズムについて，詳しく述べる．3 章では，待ち行列理論を用いたボトルネックリンクの利用率の推定方法について述べる．4 章では，その輻輳制御アルゴリズムへの応用について述べる．5 章で，シミュレーションによる評価について述べ，最後に，6 章で結論を述べる．

2. 関連研究

表 1 に関連研究を比較する[4)]．

2.1 HD 輻輳制御アルゴリズム[5)]

Leith ら[6)] は，遅延に基づく AIMD 輻輳制御について，詳しく述べている．遅延に基づく AIMD 輻輳制御は，リンク利用率が高く，遅延時間が小さく，輻輳に関わるパケット損失がない End-to-End の制御を提供することを保証する．このアイデアは Budzisz ら[5)] に

	遅延の測定方法	輻輳の推定方法	輻輳制御方法
Congestion Avoidance using Round-trip Delay (CARD)	RTT	正規化された遅延の傾斜, $(\frac{\tau_i-\tau_{i-1}}{\tau_i+\tau_{i-1}}) > 0$	AIMD($\beta = \frac{7}{8}$)
DUAL	every 2nd RTT	$\tau_i > \frac{\tau_{min}+\tau_{max}}{2}$	AIMD($\beta = \frac{7}{8}$)
TCP Vegas	RTT	$\tau_i > \tau_{min} + \theta$	AIAD
Fast TCP	平滑化された RTT	TCP Vegas と同様	MIMD
TCP-LP	平滑化された片方向遅延	$\bar{d_i} > d_{min} + \delta(d_{max} - d_{min})$	AIMD
TCP-Africa	平滑化された RTT	TCP Vegas と同様	ハイブリッド型
Compound TCP (CTCP)	平滑化された RTT	TCP Vegas と同様	ハイブリッド型
Probabilistic Early Response TCP (PERT)	RTT と平滑化された RTT	キューイング遅延 $(q_j = \tau_j - \tau_{min})$ に基づいた閾値	確率的バックオフ
Hamilton Delay (HD)	RTT	キューイング遅延に基づいた閾値	確率的バックオフ
Caia-Hamilton Delay (CHD)	RTT	キューイング遅延に基づいた閾値	確率的バックオフ
Caia Delay-Gradient (CDG)[7]	RTT	RTT の傾斜のタイプ	確率的バックオフ

表 1 遅延を用いた輻輳制御の比較.
（ここで β は Multiplicative Decrese 係数，θ は遅延の閾値，τ_i は RTT，τ_{min} は最小 RTT，τ_{max} は最大 RTT，d_i は片方向遅延.）

よって，パケット損失に基づく TCP アルゴリズムと公平に共存するために改良された.

Hamilton Delay-based (HD) 輻輳制御アルゴリズムでは，それぞれの確認応答を受け取るごとに，輻輳ウィンドウサイズ (w) を次のように評価する.

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(q_i) \\ w_i + \frac{1}{w_i} & otherwise \end{cases} \quad (1)$$

ここで，$g(q_i)$ は図 1 に示すバックオフ確率関数であり，$X \in [0,1]$ は乱数，p_{max} はバックオフの最大確率，$q_{max} = RTT_{max} - RTT_{min}$ は最大キューイング遅延の見積もり，q_{min} はターゲットとなる最小キューイング遅延，q_{th} は範囲 A と範囲 B を分ける閾値である.

パケット損失に基づくフローがリンク上に存在するとき，キューは範囲 B に移る．この範囲においては，遅延に基づくフローのバックオフ確率は低く，パケット損失に基づくフローとより公平に帯域を利用できる．パケット損失に基づくフローがボトルネックリンクに存在しないとき，範囲 B には移動せず，遅延に基づくフローは範囲 A の低遅延の状態に集中する.

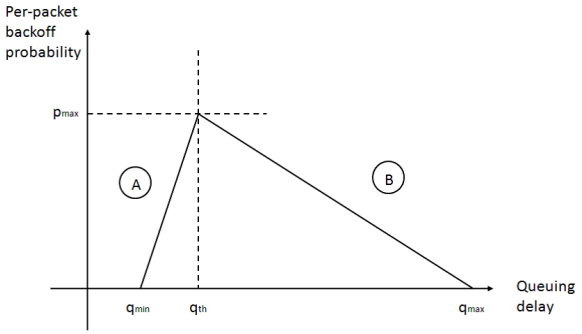

図 1 バックオフ確率関数[5].

2.2 CHD 輻輳制御アルゴリズム[4]

Caia-Hamilton Delay-based(CHD) 輻輳制御アルゴリズムは，図 1 と同じ確率関数を用いるが，HD 輻輳制御アルゴリズムに対して，以下の 3 つの修正をしている.

- 遅延に基づく輻輳ウィンドウ制御は，RTT ごとに実行される.
- CHD は，輻輳に関連しないパケット損失を推定し，そのパケット損失に対して耐性を持たせる.
- CHD は，パケット損失に基づく TCP アルゴリズムとの共存時の性能を改善する.

HD と同様，CHD は TCP の送り手のみの改変を必要とする．現在の TCP の受け手は変更を必要としない.

2.2.1 遅延に基づくウィンドウの更新

CHD では，RTT ごとに一回，輻輳ウィンドウ (w) を更新し，最新の RTT における最大キューイング遅延を使用する．最新の RTT r におけるキューイング遅延を $h_r = max_r(q_i)$ とすると，CHD は RTT ごとに w を次のように更新する.

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(h_r) \\ w_i + 1 & otherwise \end{cases} \quad (2)$$

$X < g(h_r)$ は，遅延に基づくウィンドウ縮小を表している.

2.2.2 パケット損失に基づくウィンドウの更新

CHD では，*shadow window*(s) を用いて，パケット損失に基づくフローとの公平性を実現している.

$$s_{i+1} = \begin{cases} max(w_i, s_i) & X < g(h_r) \wedge \mathcal{A} \\ max(w_i, s_i) & X < g(h_r) \wedge h_r > q_{th} \\ 0 & otherwise \end{cases} \quad (3)$$

ここで，$\mathcal{A} = h_r < q_{th} \wedge h_r > h_b$ であり，h_b は，遅延に基づく輻輳ウィンドウの縮小が起きたときの h_r の値である．

$s \neq 0$ のとき，shadow window は，TCP NewReno の輻輳ウィンドウを模倣する．また，shadow window は，最新の h が範囲 B にあるときにパケット損失が起きた場合にも用いられる．もし，パケット損失が範囲 A で起きたとき,（このパケット損失は輻輳に関連しない損失であると仮定できるので,）w の値は変わらない．パケット損失が起きたときの w の更新に関する CHD のルールは，以下の通りである．

$$w_{i+1} = \begin{cases} \frac{max(w_i, s_i)}{2} & packetloss \wedge h_r > q_{th} \\ w_i & otherwise \end{cases} \quad (4)$$

3. ボトルネックリンクの利用率推定

リンク利用率を ρ，そのリンクでのキューイング遅延が 0 の確率を p_0 とすると，待ち行列モデル M/M/1，M/D/1，M/G/1，G/M/1，G/G/1 のいずれのモデルにおいても，次式が成り立つ[8]．

$$\rho = 1 - p_0 \quad (5)$$

ここで $1 - p_0$ は，キューイング遅延が 0 でない確率と等しい．よって，N 回のパケット送信において，$RTT = RTT_{min}$（RTT の最小値）の回数を N_{min} 回とすると，ボトルネックリンクの利用率 ρ は，次式で与えられる[9]．

$$\rho = \frac{N - N_{min}}{N} \quad (6)$$

ただし，TCP SACK オプションを用いるとき，確認応答のパケットサイズは一定でなくなり，中継ノードでの確認応答の処理時間も一定でなくなるため，$RTT = RTT_{min}$ の回数を N_{min} としたとき，式 (6) は成り立たない．

そこで，式 (6) をより正確にするため，RTT ではなく，片方向遅延を用いて N_{min} を測定する．具体的には，TCP タイムスタンプオプションを用い，確認応答のタイムスタンプエコー応答値（データセグメント送信時の送り手の時刻）とタイムスタンプ値（確認応答送信時の受け手の時刻）の差分をとり，その最小値の回数を N_{min} とする．

4. UCC:リンク使用率に基づいた輻輳制御

本章では，無線ネットワークで高性能が期待されるリンク利用率に基づいた End-to-End 輻輳制御 Utilization-based Congestion Control(UCC) を提案する．

UCC では，式 (6) を用いて，ボトルネックリンクの利用率を推定する．式 (6) のリンク利用率 ρ は，キューイング遅延などと比べ,「中期的」あるいは「長期的」なリンクの状態を示す．

UCC は，バッファオーバフローによるパケット損失と言った「短期的」なリンクの状態だけではなく，ボトルネックリンクの利用率と言った「中期的」,「長期的」なリンクの状態にも基づいて制御をおこなう．すなわち，キューイング遅延によって検出されない潜在的な輻輳が，リンク利用率という観点から検出された場合に，輻輳ウィンドウを早期に縮小することで，共存する従来の輻輳制御の性能低下を防ぐ．

本稿では,「中期的」あるいは「長期的」な輻輳状態を回避できれば，輻輳崩壊[10] は起きないという仮定で議論を進める．UCC は，輻輳崩壊を起こさないで，無線ネットワークにおける Goodput などの性能を向上させることを目的とする．

4.1 パケット損失に基づくウィンドウの更新

あるパケット損失を検出した直後から，次のパケット損失が起こるまでの時間を $n \times RTT$ とする．その $n \times RTT$ の間に，送信されるデータパケットとその確認応答によって，片方向遅延を測定し，式 (6) から，ボトルネックリンクの利用率 ρ を推定する．

CHD では，キューイング遅延によって，パケットが輻輳によって損失したと判断したとき，輻輳ウィンドウを半分（または，シャドウウィンドウの半分）まで縮小した．一方，UCC では，式 (6) のボトルネックリンクの利用率に基づいて，輻輳ウィンドウを制御する．具体的には，重複確認応答によってパケット損失を検出したとき，ボトルネックリンクの利用率 ρ が α（たとえば，$\alpha = 0.99$）より大きければ，潜在的な輻輳が起きていると判断し，輻輳ウィンドウを半分まで縮小する．

すなわち，重複確認応答によってパケット損失を検出したとき，輻輳ウィンドウは以下のように更新される．

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & packetloss \wedge \rho > \alpha \\ w_i & otherwise \end{cases} \quad (7)$$

ここで，UCC における式 (6) の妥当性について検討する．以下は，式 (6) が成り立つための必要条件である．

- 正しい RTT の最小値（または，片方向遅延の最小値）が分かっている．
- RTT の最小値（または，片方向遅延の最小値）が変わらない．
- ボトルネックリンクでキューイング遅延が起きないとき，同一パス上の他のリンクでもキューイング遅延が起きない．

第 1 項について，待ち行列理論から，キューイング遅延が起きない確率は，$p_0 = 1 - \rho$[8] なので，$\rho = 0.95$ のときでも，100 個のデータパケットを送信すれば，5 回程度正しい RTT の最小値（または，片方向遅延の最小値）が測定される計算となる．(ただし，ここではボトルネックリンク以外のキューイング遅延は考えてない．) すなわち，多くのデータパケットを送信すれば，正しい RTT の最小値（または，片方向遅延の最小値）が分かるので,「中期的」あるいは「長期的」には，正しい RTT の最小値（または，片方向遅延の最小値）が分かるという前提が成り立つ．

第 2 項について,「中期的」あるいは「長期的」に，RTT の最小値（または，片方向遅延の最小値）が変わらなければよい．RTT の最小値（または，片方向遅延の最小値）が変化するとき，UCC はその値を更新する必要がある．たとえば，UCC は，RTT の最小値（または，片方向遅延の最小値）を定期的に更新するなどの処理が必要である．

第 3 項について，ボトルネックリンク以外でキューイング遅延が起こることは十分考えられる．ボトルネックリンク以外でキューイング遅延が起きるとき，式 (6) から，ρ は大きくなる方向に誤差が出る．ρ が実際より大きく測定されたとき，式 (7) により更新される輻輳ウィンドウは，理想より小さくなる．これは，スループットが小さくなるという意味では，最適ではないが，輻輳を起こさないという意味では安全である．

UCC におけるその他のアルゴリズムは，以下のとおりである．

- タイムアウトによってパケット損失を検出したとき，パケット再送を行い，輻輳ウィンドウを 1 パケットにして，Slow Start フェーズに移る．(重度の輻輳に対応するため.)
- Slow Start フェーズでは，確認応答を受け取るごとに，輻輳ウィンドウを 1 パケット分増加させる．
- Congestion Avoidance フェーズでは，RTT ごとに輻輳ウィンドウを 1 パケット分増加させる．

また，CHD は，バッファサイズなどのボトルネックリンクの特性に依存するキューイング遅延の閾値 q_{th} を何らかの方法で決定する必要があるが，UCC は，リンク利用率の閾値 α を決めるだけでよい．

5. 評　　価

ns-2[11] を用い，図 2 のシミュレーションシナリオで，UCC の Goodput と共存する従来の輻輳制御に与える影響を評価する．

また，比較対象は，TCP NewReno，CUBIC[12]，Compound TCP[13] と，CHD から式 (2) を除いた輻輳制御アルゴリズム（CHD w/o DCC: CHD without Delay-based Congestion Control) とする．ここで CHD ではなく，CHD w/o DCC を用いる理由は，リンクエラー発生時における，キューイング遅延による輻輳検出を用いた輻輳制御を評価するためである．

まず，図 2 において，$N1$ と $N2$ に同じ輻輳制御の送り手，$N3$ に TCP の受け手 (Sack Sink) を置き，2 フロー合計の Goodput を評価する．図 2 において，ルータ R と $N3$ の間の無線リンクの容量を上り下りそれぞれ 10Mbps とし，伝搬遅延時間を上り下りそれぞれ 20ms（往復 40ms）とする．ルータ R におけるバッファ容量を 50(segments) とし，1 セグメントの大きさを 1000(bytes) とする．また，CHD w/o DCC のパラメータ $q_{th} = 30ms$ とし，UCC のパラメータ $\alpha = 0.99$ とする．それぞれの輻輳制御は SACK オプションを実装しているものとする．シミュレーション時間は 100(s) とした．

図 3 が Goodput の評価結果である．

次に，図 2 において，$N1$ に対象の輻輳制御の送り手と $N2$ に TCP NewReno の送り手，$N3$ に TCP の受け手を置き，$N1$ の輻輳制御が $N2$ の TCP NewReno の性能に与える影響を評価する．

図 4 が $N2$ の TCP NewReno の Goodput である．$N1$ の送り手が UCC のときと $N1$ の送り手が NewReno のときにおける，$N2$ の NewReno の Goodput がほぼ一致することに注目．

次に，図 2 において，$N1$ に対象の輻輳制御の送り手と $N2$ に CUBIC の送り手，$N3$ に TCP の受け手を置き，$N1$ の輻輳制御が $N2$ の CUBIC の性能に与える影響を評価する．

図 5 が $N2$ の CUBIC の Goodput である．(リンクエラーが発生するとき，$N1$ の送り手が UCC のときと $N1$ の送り手が CUBIC のときにおける，$N2$ の CUBIC の Goodput がほぼ一致することに注目．)

次に，図 2 において，$N1$ に対象の輻輳制御の送り手と $N2$ に Compound TCP の送り手，$N3$ に TCP の受け手を置き，$N1$ の輻輳制御が $N2$ の Compound TCP の性能に与え

る影響を評価する．

図 6 が $N2$ の Compound TCP の Goodput である．($N1$ の送り手が UCC のときと $N1$ の送り手が Compound TCP のときにおける，$N2$ の Copmpound TCP の Goodput がほぼ一致することに注目．)

さらに，図 7 において，$N1$，$N2$ に対象の輻輳制御（UCC，CHD w/o DCC，TCP NewReno）の送り手を置き，$N4$ に TCP のその受け手を置いて，$N1$，$N2$ の 2 フロー合計の Goodput を評価する．このとき，$N3$ に TCP NewReno の送り手を置き，$N5$ にその受け手を置いて，マルチボトルネックのシナリオとする．全てのリンクの容量を上り下りそれぞれ 10Mbps とし，伝搬遅延時間を上り下りそれぞれ 20ms（往復 40ms）とする．ルータ $R1$，$R2$ におけるバッファ容量を 50(segments) とし，1 セグメントの大きさを 1000(bytes) とする．また，CHD w/o DCC のパラメータ $q_{th} = 30ms$ とし，UCC のパラメータ $\alpha = 0.99$ とする．それぞれの輻輳制御は SACK オプションを実装しているものとする．シミュレーション時間は 100(s) とした．

図 8 が 2 フロー合計の Goodput の評価結果である．UCC の Goodput は，CHD w/o DCC と TCP NewReno の Goodput の中間に位置している．

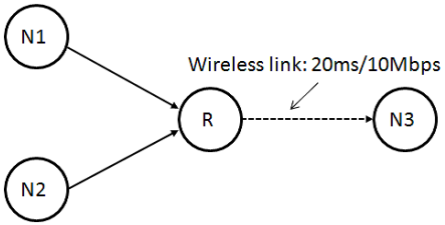

図 2 シミュレーションシナリオ．

6. ま と め

従来の End-to-End 輻輳制御 (TCP NewReno など) では，ネットワークにおけるパケット損失によって，ネットワーク輻輳を検出していた．しかし，パケット損失は，ネットワーク輻輳だけではなく，無線ネットワークにおけるリンクエラーによるものもある．従来の End-to-End 輻輳制御では，パケット損失時，輻輳ウィンドウを縮小するため，無線ネットワークにおけるリンクエラー発生時，性能が大きく低下していた．ネットワーク輻輳の検出

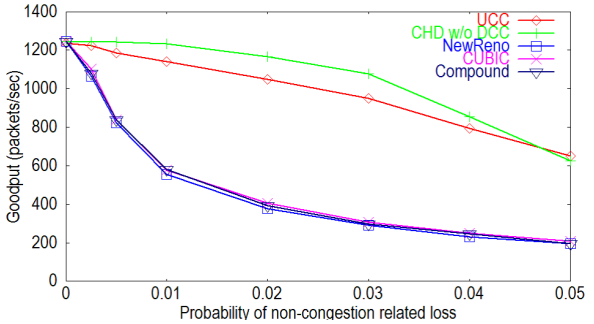

図 3 Goodput の比較．

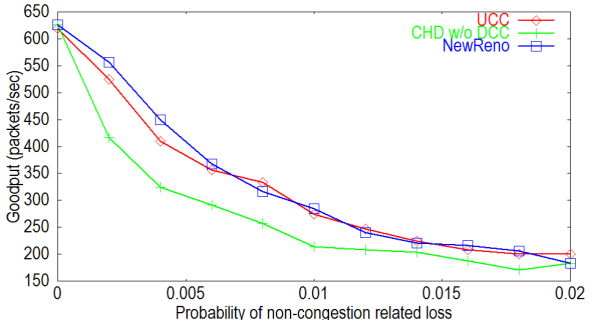

図 4 TCP NewReno の Goodput に与える影響．

を，パケット損失だけでなく，パケットのキューイング遅延の測定結果にも依存する輻輳制御として CHD が提案されている．本稿では，ボトルネックリンクの利用率の測定結果にも基づいた新しい End-to-End 輻輳制御 Utilization-based Congestion Control(UCC) を提案した．シミュレーションによる結果，UCC は，従来の End-to-End 輻輳制御（TCP NewReno，CUBIC，Compound TCP）との親和性の観点から，関連技術より最適な性能を実現できることが確認できた．また，UCC では，バッファサイズなどのボトルネックリンクの属性に依存するキューイング遅延の閾値を決める必要がなく，リンク利用率の閾値を決めるだけでよい．

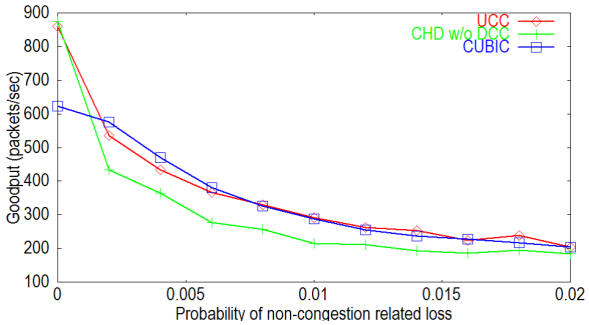

図 5 CUBIC の Goodput に与える影響.

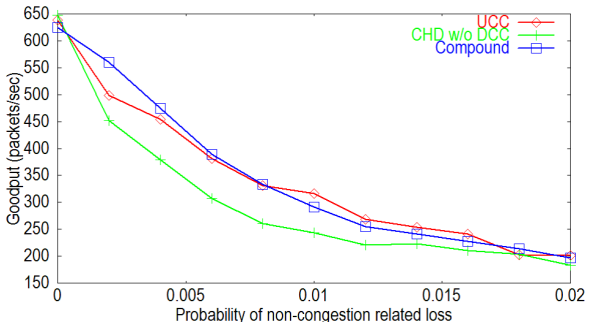

図 6 Compound TCP の Goodput に与える影響.

謝　　辞

本研究は，日本 OR 学会待ち行列研究部会第 200 回記念部会（2007 年 7 月）での講演「情報ネットワークの未来とそれを支える科学のあり方」（講演者：村田正幸教授（大阪大学））に強く影響を受けている．また，本稿の執筆に当たり，村田正幸教授から多大なるご意見を頂いた．村田正幸教授に深く感謝する．最後に，本稿の修正のため，たくさんのコメントをくださった査読者の方々に深謝する．

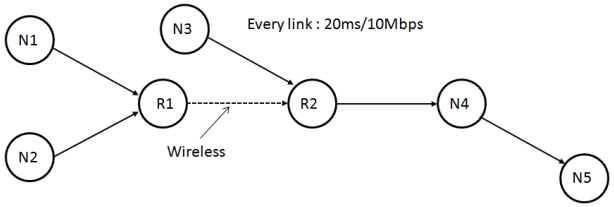

図 7 マルチボトルネックのシミュレーションシナリオ.

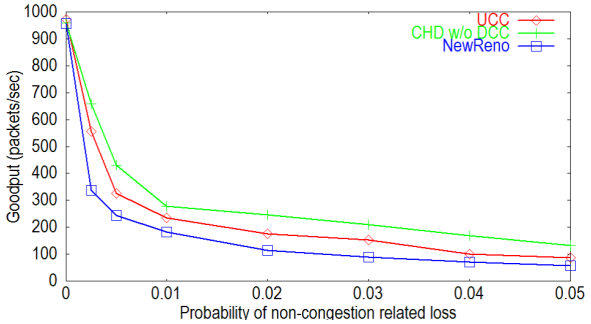

図 8 マルチボトルネックにおける Goodput の比較.

参　考　文　献

1) 長谷川剛, 村田正幸,“TCP の輻輳制御機構に関する研究動向,”電子情報通信学会論文誌, vol.J94-B, no.5, pp.663-672, May 2011.
2) Satoshi Utsumi, S.M. Salim Zabir, and Norio Shiratori, ”TCP-Cherry: A new approach for TCP congestion control over satellite IP networks,” Computer Communications (Elsevier), vol. 31, issue 10, June 2008, pp. 2541-2561.
3) Satoshi Utsumi, S.M. Salim Zabir, and Norio Shiratori, ”TCP-Cherry for satellite IP networks: Analytical model and performance evaluation,” Computer Communications (Elsevier), vol. 32, issue 12, July 2009, pp. 1377-1383.
4) David A. Hayes, and Grenville Armitage, ”Improved coexistence and loss tolerance for delay based TCP congestion control,” 35th Annual *IEEE* Conference on Local Computer Networks(LCN 2010), Denver, Colorado.
5) L. Budzisz, R. Stanojevic, R. Shorten, and F. Baker, ”A strategy for fair coexis-

tence of loss and delay-based congestion control algorithms," *IEEE Commun. Lett.*, vol. 13, no. 7, pp.555-557, Jul. 2009.
6) D.Leith, R. Shorten, G. McCullagh, J.Heffner, L. Dunn, and F. Baker, "Delay-based AIMD congestion control," in *Proc. Protocols for Fast Long Distance Networks*, California, 2007.
7) David A. Hayes, and Granville Armitage, "Revisiting TCP Congestion Control using Delay Gradients," IFIP/TC6 Networking, Valencia, Spain May 2011.
8) R. Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons Inc., USA, 1991.
9) 滝田英勝，杉崎義雄，山口実靖，淺谷耕一，" RTT を用いたボトルネックリンクの帯域使用率推定法，" 信学技報， IA2008-15，pp.13-17，Jul.2008.
10) S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control," IEEE/ACM Transaction on Networking, August 1999.
11) K. Fall, K. Varadhan, ns Notes and Documentation, Technical Report, the VINT UC Berkeley, LBL, USC/ISI, Xerox PARC, 2003.
12) I. Rhee, L. Xu, "CUBIC: A new TCP-Friendly high-speed TCP variant," in Proceedings of PFLDNet 2005, 2005.
13) K. Tan, J. Song, Q. Zhang, M. Sridharan, "A compound TCP approach for high-speedand long distance networks," in Proceedings of INFOCOM 2006, 2006.