# ネットワーク・端末総合系における
# TCP スループットの上限値に関する研究

2008 年 9 月

伊藤　正也

# ネットワーク・端末総合系における
# TCP スループットの上限値に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　　2008 年 9 月

伊藤　正也

# 内容梗概

本論文は，筆者が平成 14 年から平成 17 年までに大阪大学情報科学研究科大学院博士後期課程在学中に行った，ネットワーク・端末総合系における TCP(Transmission Control Protocol)スループットの上限値に関する研究とその応用であるエンドツーエンド TCP 制御方式に関する研究の成果をまとめたものである．

近年，ネットワークの高速・広帯域化が急速に進んでいる．一方で，端末機器としての PC(Personal Computer)については，機能面の充実は目覚しいが，高速化の点ではネットワークに対して比較的遅く，その処理速度は，ネットワークの実効転送速度と比較して同程度あるいは遅いという逆転現象が見られる．実際のネットワークは，広帯域ではあるが通信距離に比例する遅延特性を有し，種々の要因から輻輳とパケットロスが発生するため，大容量な非リアルタイム系通信では，状況に応じた輻輳制御と再送制御を提供する TCP が広く用いられている．このとき，エンドツーエンド(E2E)システム間のスループットは TCP の挙動に大きく依存し，これまでに，高いスループットを得る TCP 制御方式のアルゴリズムが数多く研究されて来た．また，効率のよいアルゴリズムを用いた場合に，スループットの上限値を求める研究も行われてきた．TCP の主機能は，ネットワークの輻輳制御と輻輳起因のパケットロス補償である再送制御である．送信端末側からネットワークへ送出するパケット量の制御が重要な役目を果たすため，ネットワークの遅延特性とネットワーク帯域（通信路容量）特性を知る必要がある．遅延特性は，従来から周回遅延時間として送信端末側で測定され TCP 制御に利用されている．しかし、ネットワーク帯域特性は，現在まで，実時間内に測定する有効な方法がなく，TCP 制御には効果的に利用されていない．

従来のスループット評価法では，データ転送時間が周回遅延時間項のみで表されているため，端末間周回遅延時間が極端に小さくなると，スループット値は無限に大きくなり，ネットワーク帯域を超えるという矛盾があった．さらに現在まで，広範囲の遅延特性に適用できる TCP スループットの限界についての評価はなされていないのが現状である．

本論文では，従来のモデルである TCP ウィンドウサイズと遅延量を基にスループットを算出する方式とは異なり，TCP 通信系をネットワーク・端末総合系として捉え，ネットワーク帯域特性を受信端末内パケット通過時間から求める．従来は，端末 PC の処理速度は，ネットワー

ク帯域に比べて非常に速くほとんど無視できたが，最近では光ネットワークの普及により端末速度が無視できなくなってきた．端末 PC の処理速度がネットワーク帯域と同程度あるいはそれ以上になると，TCP スループットは，端末 PC ハードウェア構成，転送メカニズム，MTU（Maximum Transfer Unit）などで規定される上限値に制約されることになる．PC 内部構成に基づく処理速度を考慮し, 従来から示されている TCP スループットの評価値より正確な上限値を理論的に求め，測定により得られる端末内部の遅延時間から算出されるスループット上限値と比較検証した．

TCP スループットは，端末メモリ間のデータ転送量対遅延時間比と定義される．エンドツーエンド端末とその間のネットワークは，一体の総合系を形成し，TCP データ通信は，ネットワーク帯域を最大限有効に利用しスループットを最大にするフィードバック制御と見なすことができる．この制御系において，送信端末をコントローラ，受信端末をセンサーとし，エンドツーエンドの TCP スループット上限値の理論的考察を行い, TCP スループットの上限値に限りなく近い特性を示す TCP 制御方式を求めた．

最初に，端末内部構造を考慮に入れたスループットの上限値求め，TCP スループット低下の改善指標を示し，TCP 輻輳制御方式への指標を与える．

次に，TCP スループットを決定するネットワーク・端末総合系の受信端末でのパケット通過時間の分布をリアルタイムに求めることにより，遅延量と利用可能帯域が分かり，ネットワークを共有する端末間の公平性を最大にするウィンドウ更新方法を検討した．

本論文は全 5 章から構成している．以下，第 1 章に序論を述べ，第 2 章では TCP 制御方式の概要と従来からの TCP 方式課題について明らかにしている．第 3 章では，ネットワーク端末総合系スループットモデルについて考察した後, 受信端末内パケット通過時間を用いた TCP スループット上限値の理論的な導出を行い，それらの実験結果との比較検討を行った結果を示している．第 4 章は，受信端末パケット通過時間から推定できるネットワーク帯域遅延積を基にした TCP ウィンドウ制御アルゴリズムを提案し, 本方式と従来方式との比較をシミュレーションにより行った結果ついて述べている．第 5 章に本論文の結果を述べている．

# 関連発表論文

## I. 学術論文誌論文

伊藤正也，木下和彦，戸出英樹，村上孝三，“ネットワーク・端末総合系における TCP スループット上限値の導出”電子情報通信学会，条件付採録.

## II. 国際会議発表論文（査読付）

[1] M.Ito, K.Asai, T.Murakami, R.Suzuki, K.Kinoshita, H.Tode, K.Murakami, “An Estimation of the Upper Bound of TCP Throughput in High-speed Networks”, *Proceeding of EuroIMSA*, pp.142-147, February 2006.

[2] M.Ito, Y.Nakaki, K.Tsutsumi, O.Ito, “An Evaluation Method of Recording Characteristics of the Light Intensity Modulation Direct Overwrite(DOW)”, *Proceeding of MORIS ’92*, December 7-9, 1992.

## III. 国内研究会等発表論文

[1]伊藤正也，木下和彦，戸出英樹，村上孝三，“エンドツーエンドフィードバック型 TCP 制御方式”，電子情報通信学会技術報告，CS2005-82, pp.61-66, 2006 年 1 月.

[2]伊藤正也，木下和彦，戸出英樹，村上孝三，“TCP スループット上限値の一考察”電子情報通信学会ソサイエティ大会，B-6-7, 2005 年 9 月.

[3]伊藤正也，木下和彦，戸出英樹，村上孝三，“マルチストリームレートベース TCP”，電子情報通信学会ソサエティ大会，B-6-11, 2003 年 9 月.

# 目次

# 第 1 章

# 序論

近年，ネットワークの高速・広帯域化が急速に進んでいる．一方で，端末機器としての PC については，機能面の充実は目覚しいが，高速化の点ではネットワークに比べて比較的遅く，その処理速度はネットワークの実効転送速度と比較して同程度あるいは逆に遅いという逆転現象が見られる. 実際のネットワークは, 広帯域ではあるが通信距離に比例する遅延特性を有し，種々の要因から輻輳とパケットロスが発生するため，大容量な非リアルタイム系通信では，状況に応じた輻輳制御と再送制御を提供する TCP が広く用いられている．このとき，エンドツーエンド(E2E)システム間のスループットは TCP の挙動に大きく依存しネットワーク帯域が広帯域化するに従って端末内部の遅延が無視できなくなってきた．これまでにネットワークを介した TCP の上限値を求める研究が多数行われてきた[1]，[2]，[3]．しかし，ネットワーク・端末総合系としての端末内部の遅延を含んだ正確な TCP スループットの上限値は求められていない．また，ネットワーク・端末総合系としての TCP スループット上限値は TCP スループット低下の改善指標を示すことになり，TCP 輻輳制御方式への指標を与えることになる．遅延特性に反比例する形の従来の評価法[3]では，遅延が極端に小さくなると，スループット値は，無限に大きくなり，ネットワーク帯域を超えるという矛盾があった．現在まで，広範囲の遅延特性に適用できる TCP スループットの限界についての評価はなされていないのが現状である. 本論文では，特にプロトコルレイヤーモデルで採用されている端末バッファ方式について考察を加え，各層に設けられるバッファ間の最適サイズを求め，従来の TCP 制御方式のスループットモデルを求める.

次に，TCP スループットの上限界についての評価を行い，広範囲の遅延に対して成立する正確かつ有限な上限値を求める．特に周回遅延量 RTT（Round Trip Time）が小さい場合には，従来評価法では得られなかった優れた上限値を示す．具体的には，TCP スループット上限値を，従来のネットワーク帯域，TCP 輻輳制御ウィンドウ(cwnd：congestion window) サイズに加え，今まで考慮されていなかった要因である端末内部の CPU とメインメモリ間のバス(Front Side Bus)帯域および最大パケット転送サイズ MTU(Maximum Transfer Unit)を含めた形式で，理論的

に求める[4]．更に，この上限値は，受信端末での 1 パケット通過時間を測定すればこの上限値を求められることから，ネットワークの帯域推定にも応用可能であることを示す．次に，送受信端末を用い，受信端末にパケット通過時間を測定するプログラムを実装し，実測した通過時間から算出されたスループット上限値と理論的上限値との比較検証を行う．

多くの TCP バージョンで採用されているソケットバッファ方式におけるスループットの理論的上限値を考察し，この理論値に限りなく近い最適な TCP ウィンドウ制御方式と更新アルゴリズムを提案する．従来の TCP モデルでは，TCP ウィンドウサイズとネットワーク遅延量と IP 層のキューイング遅延量を基にスループットを算出していた[1]，[2]，[3]． TCP プロトコルスタックによる処理では，TCP 層内に設けられた輻輳制御ウィンドウが，TCP 層以下のネットワークの帯域遅延積に等しくなるときスループットが最大となるが，従来方式では遅延量という時間軸データのみ利用しているため，ネットワークの帯域遅延積を検知できず，最適（スループット最大）な制御方式とは言えない結果となっていた．

従来の TCP 方式では，端末 PC は，帯域情報を使わずにバースト状にパケット塊を送出するため，広帯域ネットワークにおいては，最適な輻輳制御ウィンドウサイズが，パケットロス率で規定される塊サイズを超えるような場合があった．しかし一方では，端末 PC の高機能化および高速化が進展し，CPU によるパケットの転送時間を直接端末 PC で計測することが可能となってきたため，今までには利用されていなかった制御パラメータのネットワーク帯域が計測利用可能となってきた．受信端末パケット通過時間（受信端末パケット到着時刻と受信端末メモリ格納時刻差）は，ネットワーク帯域と強い相関があることから，この時間を実時間で計測することによりネットワーク帯域を知ることができるようになった．その結果，TCP スループットを決定するネットワーク・端末総合系のパラメータであるネットワーク帯域と遅延量を送信端末が知ることにより，帯域遅延積に等しくなるようなフローレベルの TCP ウィンドウ制御アルゴリズムが求められる．

本論文は全 5 章から構成される．以下，2 章では，特にプロトコルレイヤーモデルで採用されている端末バッファ方式について考察を加え，各層に設けられるバッファ間の最適サイズを求め，従来の TCP 制御方式のスループットモデルを求める．また TCP 方式の動向について考察し，従来方式の性能限界と課題について述べる．

3 章においては，端末内部構成，遅延時間を考慮した，一般的な広範囲の *RTT* 値に成立する TCP スループットの上限値式を求める．さらに，本章では，受信端末パケット通過時間と理論的スループット上限値の関係を，一対の送信受信端末を設置し，理論上限値と実機測定上限値との比較検証を行う．

4 章では，ネットワーク端末総合系モデルにおいて，受信端末パケット通過時間から推定できるネットワーク帯域遅延積を基にした TCP ウィンドウ制御アルゴリズムを提案し，本方式と従来方式との比較シミュレーションについて述べる．

5 章において，本研究で得られた成果を要約するとともに，今後に残された課題について述べ，結論とする．

# 第 2 章

# TCP制御方式とその課題

## 2.1 緒言

ネットワークの高速・広帯域化が急速に進む中，ネットワーク遅延は，通信距離に比例する特性を有し，通信距離が伸びるに従い増大する傾向にある．また種々の要因から輻輳とパケットロスが発生するため，スループットの低下を招く結果となる．大容量な非リアルタイム系通信では，状況に応じた輻輳制御と再送制御を提供する TCP が広く用いられている．このとき，エンドツーエンド(E2E)システム間のスループットはTCPの挙動に大きく依存することになる．TCP スループットは，TCP の制御機能である輻輳制御方式と関係があり，スループットを評価するには，端末のパケット処理を行うのに必要な各層のサービスを担うソフトウェアと必要なメモリ構成を検討する必要がある．端末内には，プロトコルに従った各層にバッファが設けられ，パケットのヘッダ処理を効率よく行えるようになっている．本章では，端末内部のバッファ構成を明確にし，TCP 輻輳制御に必要な最適バッファサイズを求め，各層バッファに基づく従来の TCP 制御方式を概観し，制御方式の課題を明確化する．

## 2.2 スループットと端末内部バッファ構成

端末を OSI アーキテクチャに従って見ると，プロトコルを複数の階層に分けて<N>層が<N+1>層にサービスを提供する事により，<N+1>層のデータを運ぶプロトコルデータユニットを基本にし，データが処理される．各層では，パケットエンベロープからデータを取り出す処理が行われるため，層内にバッファが設けられる．各層のバッファは，PC メインメモリのカーネルエリアに設けられ，ネットワークとメインメモリは PCI（Peripheral Component Interconnect）バスと FSB(Front side Bus)により接続され，プロトコルデータユニットに対する処理はすべてソフトウェア処理される．端末内部の基本的構成を図 2.1 に示す．

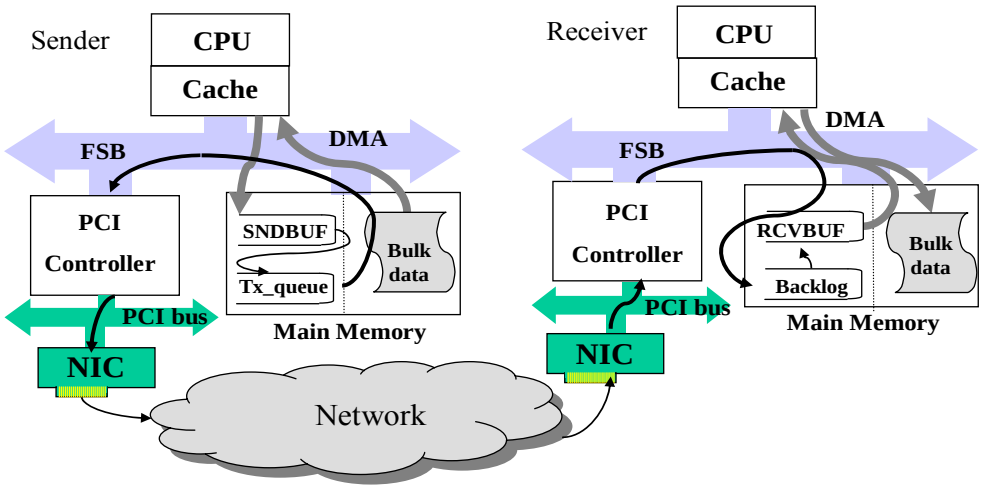



図 2.1　端末内部バッファとパケットフロー

各層での処理は、TCP より上位層としては 3 層あり，アプリケーション層ではユーザーアプリケーションに対してネットワークサービスを提供，アプリケーションの同期，エラー回復の折衝，データ整合性の保証が行われる．プレゼンテーション層では，データ表現，データ暗号化，データ圧縮，セッション層では，アプリケーション間のセッション確立・管理･終了が実行される．中間層としての TCP 層は，上位層に対してエンドツーエンドのデータ転送サービスを提供するためパケットのソケットと呼ばれるポートにシーケンス番号を割り振り，番号の順序，連続性をチェックし，データ転送を行う．TCP 層と上位層の間の PDU 処理を実行するため，通常ソケットバッファが設けられる．各層に従った端末構成を図 2.2 に示す．

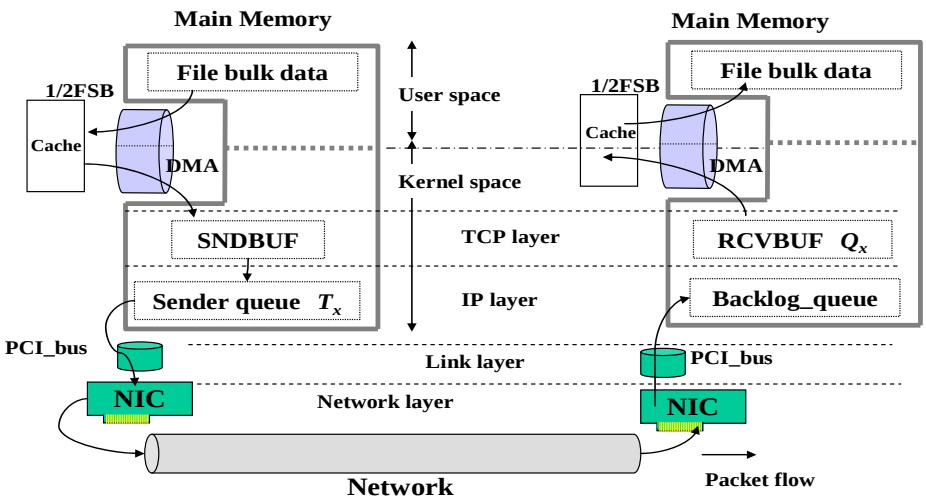



図 2.2　端末内部バッファ構成

各層でのサービスは，トランスポート層においては，上位層プロトコルデータユニット（PDU）のデータストリーム分割(セグメント化)，当該セグメントからの上位層 PDU のデータストリームへの再組立て，エンドツーエンドの接続確立，ホスト間のセグメント転送，輻輳回避としてのフロー制御と輻輳制御等である．図 2.3 に PDU とサービスの関係を示す．

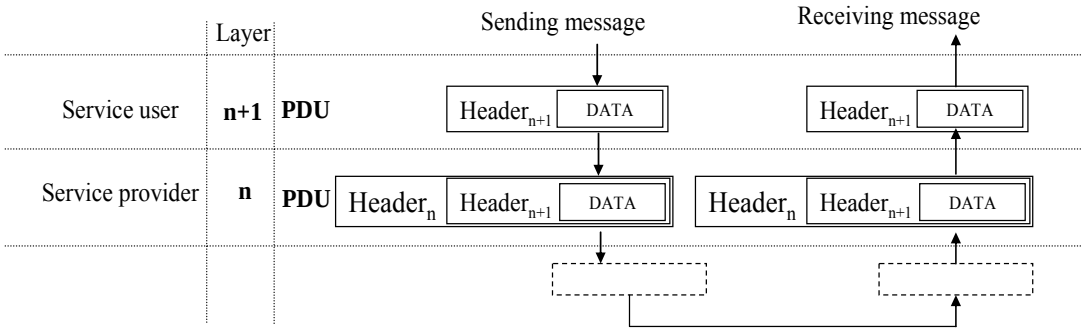



図 2.3　PDU とサービス

フロー制御の詳細は,

- バッファリング
- パケットレート制御
- パラレル化（マルチリンクによる広帯域確保）

輻輳制御の詳細は

- ネットワーク帯域遅延積算出
- 輻輳ウィンドウサイズ決定
- パケットロスに対する再送制御

コネクション型転送を実行する TCP においては，信頼性の保証が特に重要であり，信頼性を確保するためにパケットロス検知と不達パケットの再送を行っている．

TCP層でのサービスを実行する場合その性能を示す指標として，TCPスループットがRFC 1242[5]の 3.17 において，端末機器のアプリケーション層からの送信フレーム（PDU）が輻輳などによる損失のない場合の最大データ転送レートと規定されている．TCP層データストリーム中の１つのフレーム(PDU)が欠けた場合は, 多重確認応答の受信あるいはTCP層のタイムアウト信号を受信し，パケット損失を検知する．検知したときにはパケット塊の送出サイズを決める輻輳ウィンドウサイズを縮小しバーストサイズを小さくしてから損失パケットの再送を行うため，またタイムアウト信号受信の場合には，タイムアウト周期でのデータ転送となり，スルー

プットは極端に悪くなる．性能指標の劣化を防ぐには，輻輳回避を行いつつネットワーク帯域を最大限利用する最適な輻輳制御ウィンドウサイズの決定が必要である．

## 2.3 TCP制御に係わる各層内最適バッファサイズ

各層内に設けられるバッファサイズは、スループットに大きく影響する．送信端末の最上位層からパケットを送出し，最下層のネットワーク層を経由し，再び受信端末の最下層から最上層へ到る経路を考え，TCP に関与するバッファが設置されている IP 層と TCP 層での最適サイズを導出する．図 2.4 に送受信端末とネットワークの関係と各層内のバッファを示す．

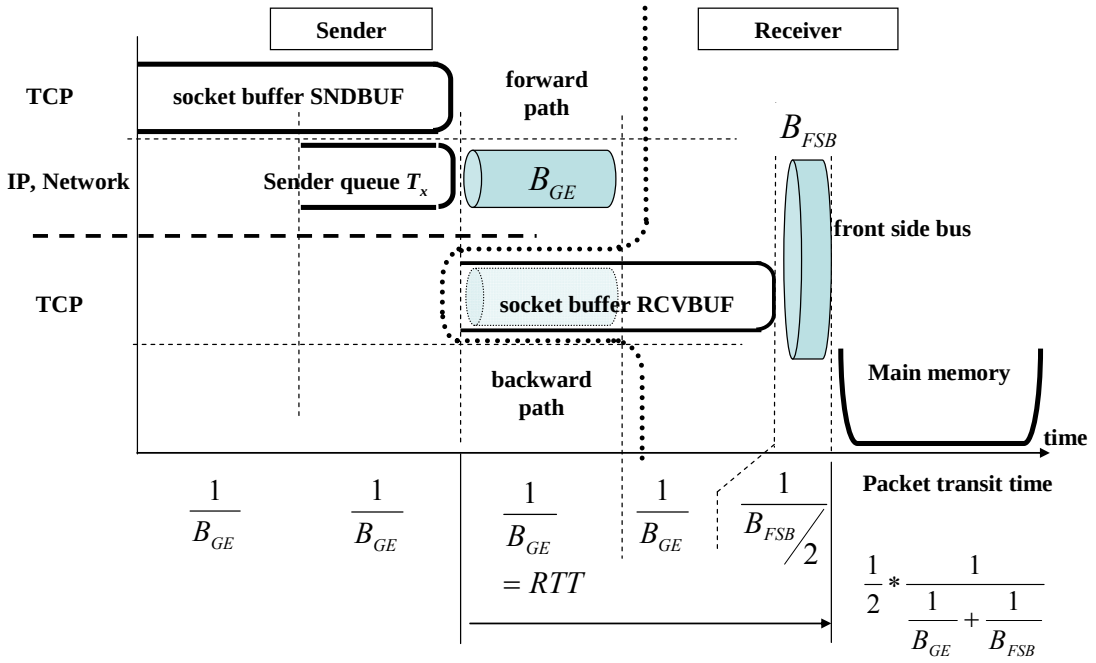



図 2.4　TCP/IP 層バッファ構成

図 2.4 において，$B_{GE}$はネットワークの帯域，$B_{FSB}$は端末内部のメモリとCPU間のバス帯域を表し，送信端末TCP層内ソケットバッファをSNDBUF，IP層内送信バッファ$T_x$（サイズ$Buf$[Byte]），受信端末内ソケットバッファをRCVBUF（サイズ$Q_x$[Byte]），ネットワーク遅延を$RTT$[sec]とする．

必要なバッファサイズは，送信端末 IP 層に設けられた送信バッファとネットワーク層間のスループットが最大になるような条件式(2.1)から，求められる．図 2.5 に IP 層とネットワーク層のバッファサイズ最適値の関係を示す．

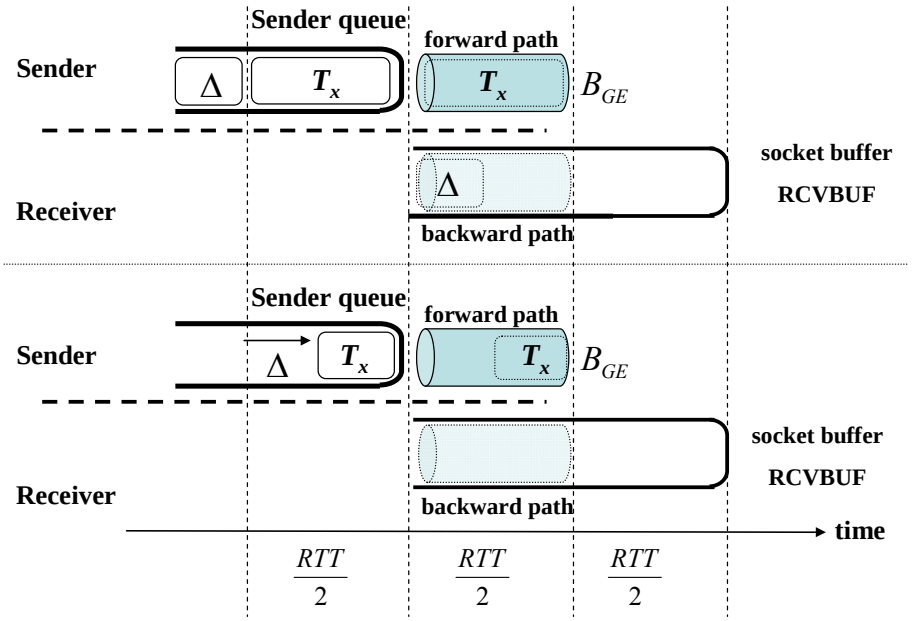



図 2.5 IP 層最適バッファサイズ

$$
\begin{aligned}
Throughput\_IP-Network &= B_{GE} * \frac{T_x + \Delta}{\left(\frac{RTT}{2} + \frac{2*\Delta}{B_{GE}}\right) * B_{GE}}, \quad \Delta > 0 \\
&= B_{GE} * \frac{T_x}{\left(\frac{RTT}{2} * B_{GE}\right)}, \qquad \Delta = 0 \\
&= B_{GE} * \frac{T_x - \Delta}{\left(\frac{RTT}{2} * B_{GE}\right)}, \qquad \Delta < 0
\end{aligned}
\tag{2.1}
$$

送信バッファの最適サイズは，式(2.2)を満足するとき，最大となる．

$$
T_x = B_{GE} * \left(\frac{RTT}{2}\right) \tag{2.2}
$$

同様の条件から，受信端末 RCVBUF の最適サイズ，送信端末 SNDBUF の最適サイズは式(2.3)となる．

$$
RCVBUF = SNDBUF = Q_x = 2 * T_x = RTT * B_{GE} \tag{2.3}
$$

IP 層に設けられるバッファサイズは式(2.3)が示すように, ネットワークの帯域遅延積に等しく, TCP 層のバッファはその 2 倍に等しくする必要がある.

## 2.4 輻輳制御ウィンドウ（cwnd :congestion window）

エンドツーエンドの TCP 通信において，送信端末はスループット向上のため，パケットをいくつかの塊としてバースト状にパケットレート制御なしでネットワークへ送出し，また送受信端末間のネットワーク中にはクロストラフィックが存在するため，パケット輻輳が発生する可能性がある．このためネットワークの輻輳を検知し回避する必要がある．回避方法は，パケットロス検知による方法と，周回遅延（*RTT*）による方法があるが，一般的にパケットロスを輻輳のシグナルと見なす．しかし，制御を行う際にパケットロスに関する十分な情報が得られない問題がある．すなわち，以下の仮定を設けている．

- ビット誤りによるパケットロスと輻輳によるパケットロスを区別できない.
- 受信端末の受信ウィンドウサイズは広告されるが, ネットワークパス中に存在する中間ノード，特にボトルネックになるルータのバッファサイズは不明.
- パケットロスは少ない.

言い換えると，輻輳回避のために，輻輳から発生するパケットロスを用いて回避するという矛盾した方法をとっている.

パケットロス以外には Vegas[6] のようにスループットから輻輳を検知する方法もあるが, 通常パケットロスを検知するのに，次の三つの方法がよく使われる.

- 再送タイムアウト(RTO)
- 3 回の重複確認応答（ACK：Acknowledge）
- SACK （選択確認応答，Selective ACK）

大別すると，ネットワークの混雑具合(輻輳)に応じて，データ転送レートを調整する方式と送出パケット塊サイズを調整する方式とに別れる．ネットワークによる遅延時間を下に，*RTT* が上がると転送レートを下げるように輻輳ウィンドウサイズを制御するレートベース制御（Rate-based control），代表的なものにTCP Vegas がある．一方時間指標ではなく，連続送出パケット数を制御するTCP Reno[7] で採用されているウィンドウベース制御（Window-based control）方式は，受信側からの ACK を待たずに送信できるデータ量を制限する．ウィンドウサイズがネットワークに注入できるデータ量に等しくなるとき，最大スループットが得られることに基づいている．これに従ってウィンドウサイズを動的に制御する．ACKの到着間隔を利

用して，使える帯域を予測する方式として，TCP Westwood[8]がある．

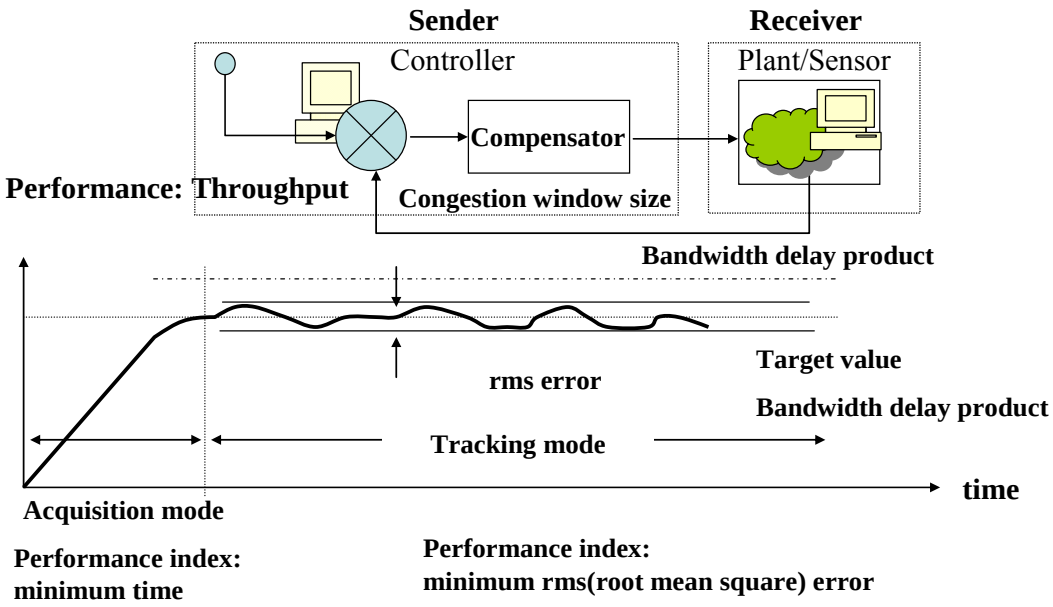



図 2.6　送受信端末ネットワーク総合系とフィードバック制御系との対応

送信端末の輻輳ウィンドウ制御は，図 2.6 に示すように， 送信・受信端末とネットワーク総合系を，輻輳ウィンドウサイズを制御量，ネットワーク帯域利用度を制御目標，スループットを制御指標とするフィードバック制御系と見ることができる．受診端末をネットワークの状態を検知するセンサーとみなしネットワークの帯域と遅延を検知し，送信端末の輻輳ウィンドウサイズを決定する制御系と等価である．このことにより，輻輳制御の方法は，ネットワーク帯域を超えない条件下でスループットを最大にする，輻輳制御ウィンドウサイズの最適値を決めることへと帰着される．

フィードバック制御系における端末内部の各層内のバッファと制御量の関係のブロック図を図 2.7 に示す．この図において，制御の流れは，送信バッファ$T_x$と片側ネットワークの帯域遅延積とのマッチングが図られ，$T_x$内パケット数とネットワーク内インフライトパケット数の総計をバッファするように受信端末ソケットバッファ（awnd : advertise window，広告ウィンドウ）$Q_x$が働き，$Q_x$サイズを送信端末へ告知するように行われる．$Q_x$の具体的値は，データ転送開始時不明であり，デフォルト値が用いられる．帯域遅延積(BDP:Bandwidth Delay Product)と$T_x$，$Q_x$，の関係は，式(2.3)から式(2.4)となる．

$$W = 2 * T_x = 2 * BDP = Q_x \tag{2.4}$$

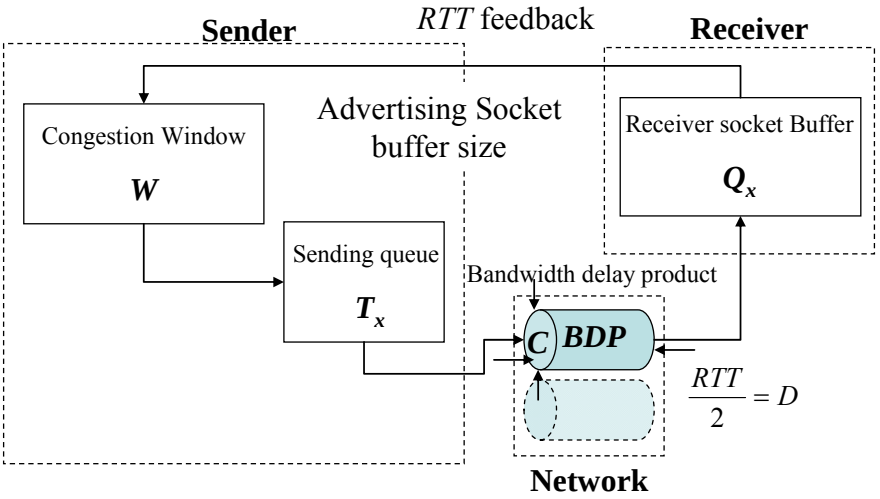



図 2.7　フィードバック制御系におけるバッファと制御の流れ

受信端末ソケットバッファ（広告ウィンドウ）$Q_x$に対して，送信側が輻輳を起こさずに送信可能と推測したウィンドウが輻輳ウィンドウである．輻輳ウィンドウ（cwnd: congestion window）$W$ [Byte] は，ACKを受信する毎に更新される．

輻輳制御は基本的に捕捉モード(制御系での目標値へ最速で到達)に相当するスロースタートフェーズと追尾モードに相当する輻輳回避の 2 フェーズが存在し，それぞれで輻輳ウィンドウの挙動に違いがある．

フィードバック制御においては，スロースタートは，利用可能帯域をセンサーである受信端末が計測しそれに基づく輻輳制御ウィンドウサイズを算出し輻輳ウィンドウサイズを設定する．追尾モードでは，目標値と制御系内部状態値との差を検出し，差を最小にするように輻輳ウィンドウサイズが制御される．フィードバック制御系においては制御系の状態変数として，遅延時間と帯域を制御器である送信端末は検知する必要がある．状態変数の内の遅延時間は，周回遅延時間として確認応答が返ってくるまでの時間を測定し検知できるが，帯域は検知が困難である．

## 2.5 既存の輻輳ウィンドウ制御方法

現在実用化されている輻輳ウィンドウ制御方法は，帯域遅延積の遅延時間を計測するものであり，帯域情報を利用していない．以下，最近までに実用化されている制御アルゴリズムの概略と課題を概観する．スロースタートは，指数関数的に輻輳ウィンドウサイズを広げ，輻輳回避フェーズは線形的にウィンドウサイズを広げる．二つのフェーズの境界は，スロースタート

閾値（*ssthresh*: slow start threshold) と呼ばれる．スロースタートの利点は，バースト的なトラフィック流入を防ぎ，また帯域遅延積をすばやく推定することができることである．スロースタートの目的は利点となる反面，転送速度が収束するのに時間がかかる．一方，パケットロスを検知すると輻輳状態と推測し，ウィンドウサイズを小さくする．再送タイムアウト（RTO: Retransmission Time Out）が発生すると，*ssthresh* を半分にして，*cwnd* を最小（通常 1 または 2）にし，スロースタートする．このようにウィンドウサイズは，スロースタートに始まり，輻輳回避アルゴリズムにより，鋸波形状の挙動を示す．各バージョンの違いは，利用可能帯域の見積り，ネットワークの輻輳の検出方法と輻輳ウィンドウ操作の種類による．

現在広く使われている TCP バージョンの Reno あるいは多重パケットロス対応の改良版 NewReno[9]は高速再送，高速リカバリのフェーズが追加され，送信レート制御にスライディングウィンドウを採用し，*RTT* 中に送出するパケット数を輻輳ウィンドウサイズで決定する．そして，通信経路の利用可能帯域の見積りを基に，帯域と遅延の積である帯域遅延積と輻輳ウィンドウが一致するように輻輳ウィンドウを拡大縮小する．

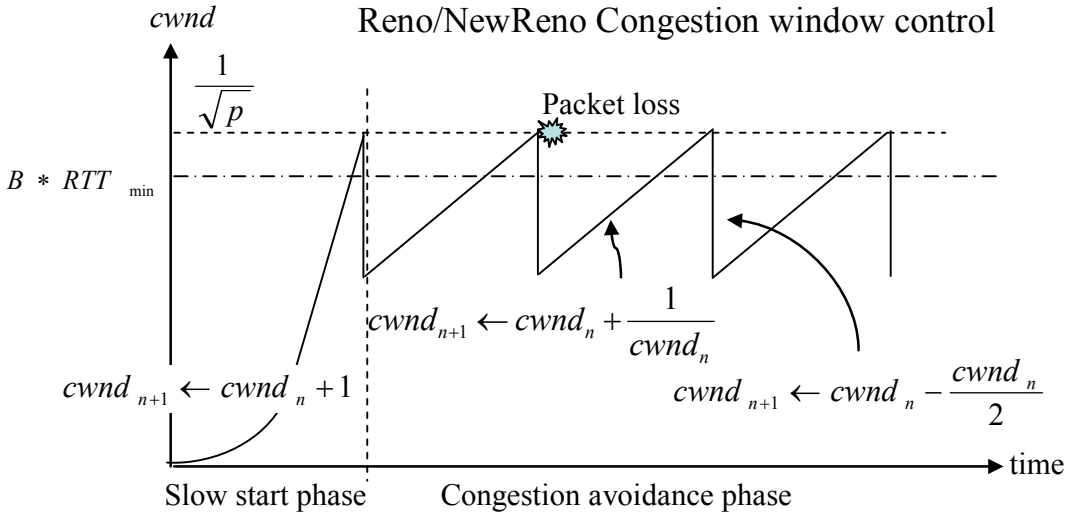



図 2.8　TCP ．輻輳制御ウィンドウ(*cwnd*)の動き

図 2.8 は，の輻輳ウィンドウの挙動であるが，非常に単純なルールに基づいて輻輳ウィンドウを決定している．RenoまたはNewRenoでは，輻輳をパケットロスによって検出し，パケットロス発生時の送信レートが利用可能帯域であるとみなす．具体的には，重複ACKを 3 回連続して受け取ると，パケットロスが発生したと見なして，輻輳ウィンドウサイズを半分にする．そして，再度*RTT*ごとに 1 パケット毎に輻輳ウィンドウを大きくする．このように，ウィンドウ

サイズを徐々に大きくして，輻輳を検出したら一気に落とす挙動を，Additive Increase Multiplicative Decrease (AIMD)と呼ぶ．RenoまたはNewRenoの場合，AIMD(1, 1/2)である．Reno の改良版として，HSTCP（High-speed TCP）[10]，STCP（Scalable TCP）[11]がある．cwnd更新のアルゴリズムは，HSTCPは式(2.5)，STCPは式(2.6)で示される．

$$\begin{aligned} ACK &: cwnd_{i+1} \leftarrow cwnd_i + \frac{a(cwnd_i)}{cwnd_i} \\ LOSS &: cwnd_{i+1} \leftarrow cwnd_i - b(cwnd_i) * cwnd_i \end{aligned} \tag{2.5}$$

式(2.5)の $a(W), b(W)$は現在の輻輳ウィンドウサイズ $W$ とパケットロス率 $p$ から算出される数値である．W が 38 より小さい時は，$a(W)$= 1 ，$b(W)$=0.5 であり，Reno または NewReno に等しくなる．$W$ が 84000 付近では，$a(W)$=72，$b(W)$=0.1 である．HSTCP は $W$ が大きい場合は，各 ACK に対してより大きなウィンドウ増加を行い，各ロスに対してはより小さく減少させるという方式である．HSTCP の $a(W)$，$b(W)$は実装される場合には，数値テーブル形式であるが，テーブル参照をなくした方式が STCP である．STCP は，式(2.2)において，各 ACK に対して常に現在のウィンドウサイズの 1 %を増加させ，各ロスに対しては 8 分の 1 を減少させる方式である．

$$\begin{aligned} ACK &: cwnd_{i+1} \leftarrow cwnd_i + 0.01 * cwnd_i \\ LOSS &: cwnd_{i+1} \leftarrow cwnd_i - \frac{1}{8} * cwnd_i \end{aligned} \tag{2.6}$$

他方のレートベース型の代表例である TCP Vegas の輻輳ウィンドウ制御概略を下図 2.8 に示す．この方式は，再送機構と *RTT* の計測粒度を上げて，タイムアウトの解像度を上げる．*RTT* をより細かな粒度で測定し，*RTT* が上がれば転送レートを下げる(輻輳ウィンドウを小さくする). タイムスタンプオプションは利用しない. 輻輳回避アルゴリズムは, Reno または NewReno のようにパケットロスではなく，キューイング遅延(エンドツーエンドでは *RTT* に等しい)によって輻輳を検知する．Reno または NewReno の場合はパケットロスが起きた直前を最大帯域と見なしているが，Vegas では最小 *RTT* から最大帯域を見積もる．その他，スロースタートフェーズにおいては，ACK 受信ごとに輻輳ウィンドウを倍増させるのではなく，*RTT* が変化するごとに倍増し，パケット対にてスループットを見積り，*ssthresh* を制限する．

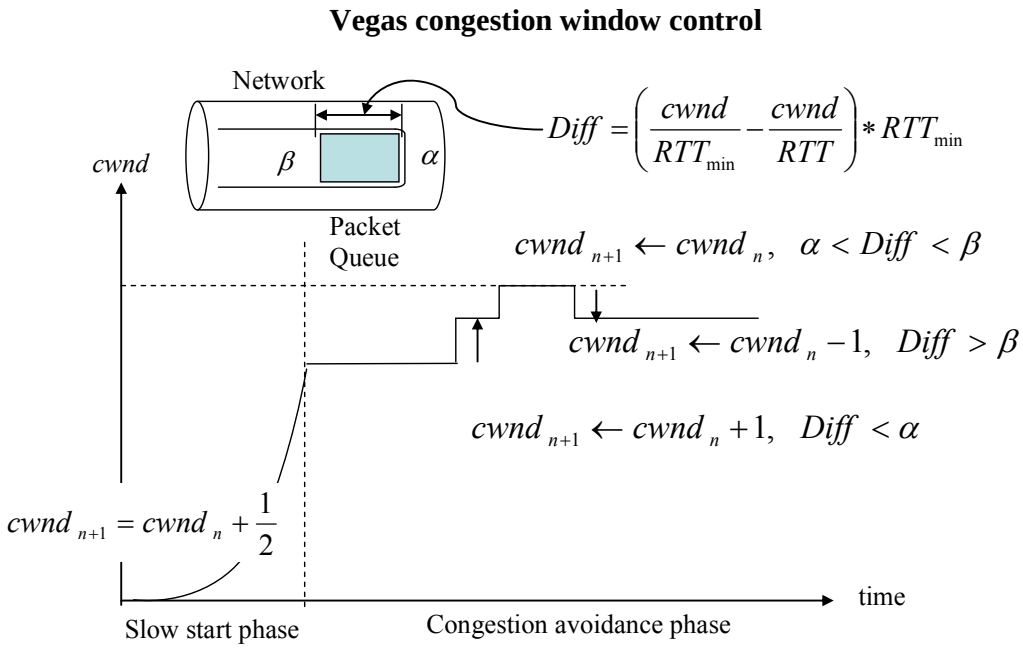



図 2.9　TCPVegas 輻輳制御ウィンドウ(*cwnd*)の動き

また，ACK を受信してすぐに *cwnd* を増加させるのではなく，*RTT / cwnd* 間隔で *cwnd* を増加させる．Vegas のアルゴリズムは，Reno または NewReno に比較して幾分複雑である．概略は以下のように示される．

転送レートの期待値と実際の値を一致(収束)させる方向で輻輳ウィンドウを制御する．

(1)測定中の最小の *RTT* を輻輳がない場合の *RTT* ，現在の $cwnd / RTT_{min}$ をスループットの期待値．

(2)送信バイト数/（現在の *RTT*）を実際のスループット．

(3)期待値と実際のスループットの差分を $Diff/ RTT_{min}$ ．

これらの量から，ウィンドウを以下のように更新する．

(4)*Diff* < α の場合，*cwnd* を+1 増加．

(5)α < *Diff* < β の場合，*cwnd* は不変．

(6)β < *Diff* の場合，*cwnd* を-1 減少．

α，β は，最低 α パケット，最大 β パケットがネットワークの中間ノードにキューイングされている状態を表す整数である．*Diff* が定数 γ（β の初期値）を超えれば，スロースタートから輻輳回避フェーズに遷移する．

TCP Vegasの改良版に，FAST（Fast Active Queue Management Scalable TCP）[12]がある．輻輳制御アルゴリズムは，式(2.3)のような更新手順に従う．パケットレベルとフローレベルの両

面から検討されている．パケットレベル(AIMD)の実装として設計され，同時に公平性，性能，安定性の観点からフローレベルでの解析が行なわれた．単純にウィンドウサイズを大きくしRenoまたはNewRenoをはじめとした HSTCP，STCPなど過去に提案されている広帯域高遅延積向けの輻輳制御方式との違いは，フローレベルでの収束性，パケットロス率 $p$ とウィンドウサイズ$W$の関係を示したレスポンス関数$W = 1/\sqrt{p}$ との親和性などのゴールを明示的に意識して設計された点である．さらに，FASTが特徴的なのはTCP Vegas同様，パケットロスではなく，キューイング遅延を利用して輻輳状況を見積り，ウィンドウ制御に反映させている点である．高いリンク利用率の達成は，ボトルネックルータのキューがオーバフローする(パケットロス)危険性を同時に持ち合わせているが，キューイング遅延を使うと定期的にパケットロスを起こしてしまう他の方式と比べ，パケットロスを利用せずにボトルネックリンクルータのキューを安定に維持できる．

$$cwnd_{i+1} \leftarrow \min\left\{2 * cwnd_i, (1-\gamma) * cwnd_i + \gamma * \left(\frac{RTT_{\min}}{RTT} * cwnd_i + \alpha\right)\right\} \tag{2.7}$$

ただし，0<γ<1，α は 1 より大きい増加パケット数，$RTT_{min}$は測定値の最小値．

パケットロスを起こさなくとも，ルータが明示的に輻輳を通知するECN（Explicit congestion notification）もあるが，ルータの対応が必要であり，普及には至っていない．FASTはECNが利用可能であれば利用し，必須ではない．また，パケットロス検知後の挙動はRenoまたはNewReno[4]に従う．さらに，高速リカバリを抜けるまでは，$RTT$ が信用できないので，ルーティング遅延に反応しないようになっている．また，バースト的に送出する際の送信レートを$RTT$毎に送信レートの見積り，輻輳ウィンドウサイズを基にバースト時間を制御するするバースト制御も行う．定数α，γの最適値の決定方法に課題がある．

## 2.6 課題

端末およびネットワークのレイヤーモデルにおいては，各層において PDU の処理が行われるため，各層にバッファメモリが設けられる，このバッファメモリを使い TCP プロトコルスタックによる制御が行われる．TCP エンドツーエンド通信系を制御系と見た場合の各層のバッファ処理によるスループットを劣化させないためには，ネットワーク層の帯域遅延積に上位層のバッファ量を等価させる必要がある．ネットワーク帯域遅延積は時間とともに動的に変化する

ため，端末により測定しなければならない．帯域遅延積を大きく上回ると，ネットワークには輻輳が生じスループットは極端に落ち込むため，送信端末では，輻輳制御ウィンドウを使い輻輳を回避させる．Reno または NewReno に代表されるウィンドウベース型は，帯域遅延積値をパケットロスにより検知する方法であり，本質的に輻輳回避の方法ではない．また，Vegas に代表されるレートベース型では，遅延量は測定されて検知しているが，帯域量は測定されず，利用されていないため正確な帯域遅延積量が検知できない. エンドツーエンド TCP 通信においては，信頼性確保のためにパケット受信の確認応答が受信端末から返送される．この ACK を使い遅延量は正確に測定できるため，もう一方の帯域の測定法が確立すれば，理論的にも最適な TCP 輻輳制御方式が見出せることになる．帯域推定を行う方法を採った TCP バージョンとして TCP Westwood[5]が報告されている. ACK ストリームを基に利用可能なバンド幅を見積り，スロースタートフェーズと輻輳回避フェーズとの切り換え閾値(*ssthresh*)と輻輳ウィンドウサイズ(*cwnd*)を設定する．初期の輻輳から離れている場合は，帯域はかなり正確に検知できるが，ネットワーク利用を上げるため輻輳に近い状態では遅延量が大幅に変化するため，いずれの TCP バージョンでも，正確な帯域は得られない.

ネットワークのブロードバンド化に伴いネットワーク利用度を最大限利用した高スループット TCP 制御方式を確立する必要性もあり，帯域遅延積に関して正確な値を検知あるいは利用ということが従来からの方式における重要な課題であった.

端末 PC の処理能力の向上に従って，実時間で種々の量を端末で実時間に測定できるようなり，受信端末によりパケット通過時間を測定し，送信端末に返送することにより，帯域遅延積に必要な帯域を求めることが可能となってきた. 正確な TCP エンドツーエンドスループット上限値が評価可能となると同時に，可能なパケット通過時間を検知することにより，正確な帯域遅延積を輻輳ウィンドウ制御に利用する課題解決の一方式を提案できることになる．この制御方式は，従来のウィンドウベース型，レートベース型 TCP 制御方式よりネットワーク利用率の高い制御方式が提供できる.

## 2.7 結言

端末では, PDU の処理のため各層に設けられたバッファメモリを使い TCP プロトコルスタックによる制御が行われる．TCP エンドツーエンド通信系を制御系と見た場合の各層のバッファ処理によるスループットを劣化させないためには，ネットワーク層の帯域遅延積に上位層のバッファ量を等価させる必要がある．ネットワーク帯域遅延積は時間とともに動的に変化する

ため，端末により測定しなければならない．従来の方法であるウィンドウベース型は，帯域遅延積値をパケットロスにより検知する方法であり，本質的に輻輳回避の方法ではない．また，レートベース型では，遅延量は測定されて検知しているが，帯域量は測定されず，利用されていないため正確な帯域遅延積量が検知できない．

本研究では，端末の各層内バッファ量をネットワークの帯域遅延積に等化させ各層間でのスループット劣化をなくし，ネットワーク・端末総合系としての TCP スループットの劣化を最小限にするような TCP 方式を求めた．スループット低減を最小にするには，正確なネットワークの帯域遅延積を端末において検知する必要がある．TCP 通信に用いられるパケットとその受信確認応答を用いて，ネットワーク遅延量と非輻輳時に，ネットワーク帯域を複数のパケットを用いて測定し，帯域遅延積を検知する従来ではなされていなかった方法について検討した．

# 第 3 章

# ネットワーク・端末総合系における TCP スループット上限値

## 3.1 緒言

ネットワークシステムを構成する端末機器としての PC は，機能面の充実は目覚しいが，高速化の点ではネットワークに比べて比較的遅く，ネットワークの高速・広帯域化が急速に進んでいる中で，その処理速度は，ネットワークの実効転送速度と比較して同程度あるいは逆に遅くなっている. 実際のネットワークは, 広帯域ではあるが通信距離に比例する遅延特性を有し，種々の要因から輻輳とパケットロスが発生するため，大容量な非リアルタイム系通信では，状況に応じた輻輳制御と再送制御を提供する TCP が広く用いられている．このとき，エンドツーエンド(E2E)システム間のスループットは TCP の挙動に大きく依存し，これまでに，その上限値を求める研究が多数行われてきた．TCP スループットの上限値を求めることは，TCP スループット低下の改善指標を示すことになり，TCP 輻輳制御方式への指標を与えることになる[13]. しかし，遅延特性に反比例する形の従来の評価法[1]では，遅延が極端に小さくなると，スループット値は，無限に大きくなり，ネットワーク帯域を超えるという矛盾があった．さらに現在まで, 広範囲の遅延特性に適用できる TCP スループットの限界についての評価はなされていないのが現状である.

本章では，TCP スループットの正確な限界についての評価を行い，広範囲の遅延に対して成立する正確かつ有限な上限値を求める．従来のモデルでは，TCP ウィンドウサイズ，遅延量を基にスループットを算出していた[2]，[3]．しかし，ネットワーク帯域が，端末 PC の処理速度と同程度あるいはそれ以上になると，TCP スループットは，端末 PC ハードウェア構成，転送メカニズム，最大パケット転送サイズ(MTU:Maximum Transfer Unit)などで規定される上限値に制約されることになる．特に周回遅延量（RTT:Round Trip Time）が小さい場合には，従来評価法では得られなかった優れた上限値を示す．具体的には，TCP スループット上限値を，従来のネットワーク帯域，TCP 輻輳制御ウィンドウサイズ(cwnd:congestion window size)に加え，今まで考慮されていなかった要因である端末内部の CPU とメインメモリ間のバス（Front Side Bus）

帯域および MTU を含めた形式で，理論的に求める[4]．更に，この上限値は，受信端末での 1 パケット通過時間を測定すればこの上限値を求められることから，ネットワークの帯域推定にも応用可能であることを示す．

エンドツーエンドの理論値を検証するために2台のPC(Linux Fedora Core2 P4, TCP NewReno) 送受信端末を用い，受信端末にパケット通過時間を測定するプログラムを実装し，実測した通過時間から算出されたスループット上限値と理論的上限値との比較検証を行う．

受信端末から得られる情報は，ACKパケットに載せられて送信端末にフィードバックされる．TCP スループットを決定するネットワーク端末総合系のパラメータであるネットワーク帯域と遅延量を送信端末が知ることにより，帯域遅延積に等しくなるようなフローレベルの TCP ウィンドウ制御アルゴリズムが求められる．また，受信端末でのパケット通過時間の分布をリアルタイムに求めることにより，最大帯域と利用可能帯域が分かり，ネットワークを共有する端末間の公平性を最大にするウィンドウ更新方法が分かる．

## 3.2 従来の TCP スループット評価モデル

従来のTCPスループット評価モデルでは，端末PCの処理速度は考慮されず，端末間の距離に依存する周回遅延量，通信路の種類により決まる帯域，ランダムエラーおよび輻輳に基づくパケット廃棄から決まるパケットロス率を基にしていた[1]，[2]，[3]．スループットは，送信端末のアプリケーション層から投入されたパケット数対所要時間比として定義され，ネットワーク特性パラメータを含んだ形になり，時間内にできるだけ大きなパケット塊が送出された場合に大きな値をとる．しかし，パケット塊の最大サイズはパケットロス率$p$とネットワーク帯域$B$[bps]の制限を受け，無限に大きくはならず有限の値に制限される．この従来モデルから算出されるTCPスループット$T_{conv}$[bps]は，パケットロス率$p$により制限される最大輻輳制御ウィンドウサイズ$W_{max}$[パケット数], 周回遅延時間$RTT$ [sec], 最大パケット転送サイズ$MTU$[Byte]を用いて，式(3.1)のように表される[1].

$$T_{conv} = \min\left(\frac{W_{\max} * MTU}{RTT}, \frac{MTU}{RTT * \sqrt{\frac{4p}{3}} + T_0 * \min\left(1, 3\sqrt{\frac{3p}{4}}\right) * p\left(1 + 32p^2\right)}\right) \tag{3.1}$$

ここで$T_0$ [sec] は確認応答タイムアウト時間である．

TCPは，パケットロスが発生すると再送を行い，その際に輻輳制御ウィンドウサイズを減少させるため，最大輻輳制御ウィンドウサイズ$W_{max}$と，$RTT$内にネットワークに存在するパケット数$N$は，以下の式(3.2)を満足することになる．図 3.1 にパケットロス率$p$と最大輻輳ウィンドウサイズ$W_{max}$との関係を示す．

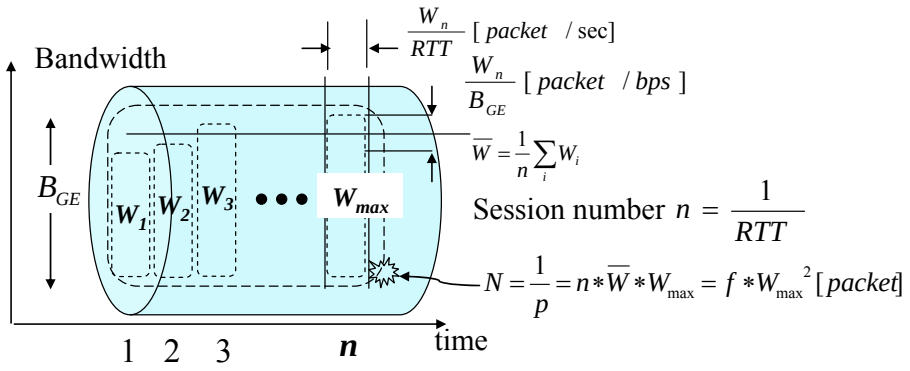



図 3.1　パケットロス率 $p$ とウィンドウサイズの関係

$$N = \int_0^{RTT} \int_0^{B_{GE}} \frac{W_i}{RTT} * \frac{W_{max}}{B_{GE}} dtdB = f * W_{max}^2 \tag{3.2}$$

ここで$f$は，輻輳制御ウィンドウの増減方法により決まる 1 より小さい定数である．パケットロスは，ランダムなビットエラーにより発生する場合と，輻輳状態から，パケット廃棄により発生する場合とがある．パケット廃棄を検知する方法は，パケット確認応答（ACK : Acknowledge）が返ってくる毎に輻輳制御ウィンドウを増加させ，パケット廃棄を発生させて行く．RenoおよびNewRenoでは，$W_{max}$/2 から開始し，$W_{max}$/2 回目に$W_{max}$に到達するように線形増加を行い，$f$=3/8 となる．理想的な検出方法は，$W_{max}$-1 から開始し，$W_{max}$回目に$W_{max}$に到達する離散的な増加方法であり，この場合は，$f$=1 となる．上限値を検討するために，以後この理想的なパケット検知方法が実装されている系においての考察を行う．

なお，このネットワーク中にある $N$ 個目の 1 個が損失する場合を考慮すると，一定時間 $RTT$ 内のパケットロス率 $p$ とすると，$p$ と $N$ の間には，次式(3.3)で示す関係がある．

$$p = \frac{1}{N} \tag{3.3}$$

これらの式から，以下の不等式(3.4)が成立する[2]，[3].

$$\frac{W_{\max} * MTU}{RTT} \le B,\ W_{\max} \le \frac{1}{\sqrt{p}} \tag{3.4}$$

なお，本モデルでは，簡単のため，ネットワークでのパケットロスはその発生箇所によらず全て確率*p*にまとめて反映させ，遅延は全て*RTT*に含まれているものとしている．通常，光ファイバを用いた通信でのビットエラーレートは，$10^{-10}$以下であるので，$\sqrt{p}$ は $10^{-3}$のオーダーとなり，また$W_{max}$は 2 パケットからスタートし，*p*値で規定された個数まで時間と共に増加するので，端末間遅延が 1msec以下から 1*sec*以上の広い範囲に亘って，式(3.1)は式(3.5)のようになる．

$$T_{conv} = \frac{W_{\max} * MTU}{RTT} \le B,\ 2 \le W_{\max} \le \frac{1}{\sqrt{p}} \tag{3.5}$$

この上限値モデル(3.5)式は，すべての*RTT*値に対し，スループットはネットワーク帯域*B*以下であることを意味している．一方，図 3.2 に示す仮想的なE2Eシステムにおいて，伝送路内伝播速度を光速度$c$ = 3x$10^8$ [m/sec]と(媒体屈折率)$^{-1}$の積とし，最大パケット転送サイズ *MTU* [bit] ，端末間距離 *L* [m]，と定義すると遅延*RTT*は式(3.6)で表される[14]．通常のLAN

$$RTT = \frac{L}{0.66c} + \frac{MTU}{B} \tag{3.6}$$

(*L*<50m) では，伝播遅延*L* / 0.66cは，0.25μsec より小さく，伝送遅延*MTU* / *B*は 12μsec (Gigabit Ether)あるいは 120μsec (Fast Ether)であるため，伝送遅延が支配的である．このように*L*が十分小さい場合，*RTT*の最小値$RTT_{min}$は式(3.7)で与えられる．

$$RTT_{\min} \cong \frac{MTU}{B} \tag{3.7}$$

$W_{max}$は 2 より大きいので

$$T_{conv} = W_{\max} * B \ge 2 * B \tag{3.8}$$

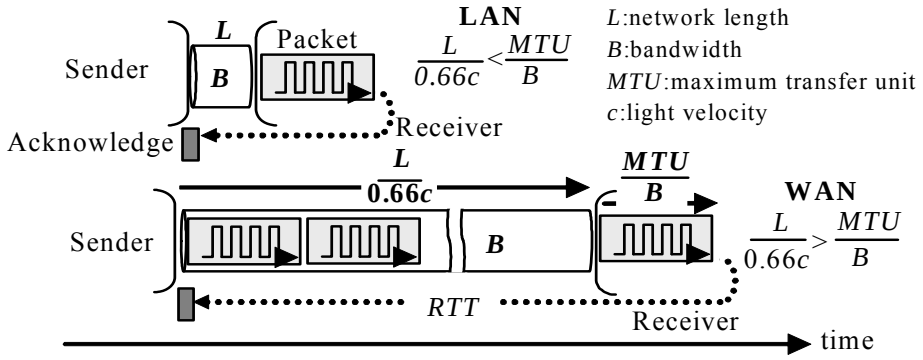



図 3.2　ネットワーク距離対 RTT

となり，式(3.5)と矛盾する結果となる．これは，式(3.5)の分母の遅延時間成分が *RTT* 値のみから構成されていることによるものである．従来モデルは，以下のことから欠点を補うことができる[4]．スループットは，端末メモリから端末メモリまでの総データ量と総転送時間の比で定義されるから，総転送時間に端末内部処理時間を加える必要がある．このことにより，*RTT* がパケットの伝送遅延のみの非常に小さい値をとる場合から，地球規模の長距離通信のような *RTT* の非常に大きな場合にいたる広範囲な値に対して成立するスループット上限値式を導出する．

## 3.3　TCP スループット上限値の導出

本節では，まず *RTT* が十分に小さい場合のスループットを求め，その後広範囲な *RTT* の場合へと一般化する．

### 3.3.1　RTT 最小域でのスループット

RTT が最小になるのは，LAN 環境の場合である．このときは，*MTU* 単位の各パケットの伝送を終了すると直ちに確認応答(ACK)が返ってくるので，パケットを 1 単位毎に連続して高速に送信できる．確認応答 ACK と *RTT* のタイミングを図 3.3 に示す．したがって，1 パケット毎の受信端末通過時間からスループットが求められる．図 3.4 にパケットフローを示す．端末内部を含めた E2E システムを見た場合, これは端末 PC のハードウェア構成, 転送メカニズム, 最大パケット転送サイズで規定されることになる[4].

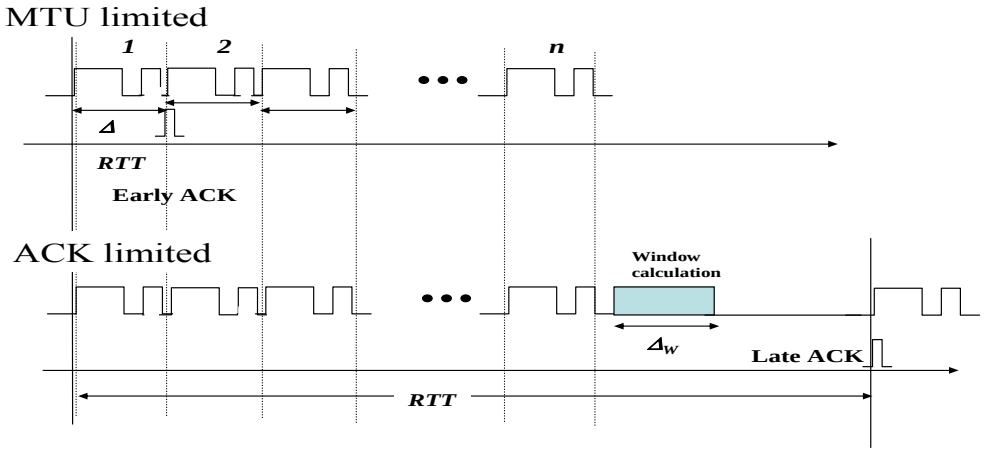



図 3.3 ACK と *RTT* のタイミング関係

端末内部では，パケット化処理，TCP/IPヘッダ処理，チェックサム処理，確認応答処理，再送処理，輻輳防止制御など多くの処理を経てデータがアプリケーション層へ転送される．CPUがこれらを効率よく行えるように，ソケットバッファが設けられ，データが一旦バッファされるため，スループット値を決める遅延時間をさらに増大させることになる[15]，[37]．PCは一般にメインメモリ，CPU，I/O，メインメモリとI/Oを結ぶ PCI(Peripheral Component Interface)バス(帯域$B_{PCI}$ [bps])， ソケットバッファとCPUを結ぶFSB(Front Side Bus，帯域$B_{FSB}$ [bps] )から構成され，送受信端末間のネットワーク帯域を$B_{GE}$ [bps]とすると，各帯域の関係は，式(3.9)のようになる．ここで，$B_{FSB}$ については読み書きのため有効帯域が常に半分になることを考慮に入れている．送受信端末の送信バッファと受信バッファはこれらのバスで結ばれ，パケットは独立な 2 つのCPUにより非同期で転送される．

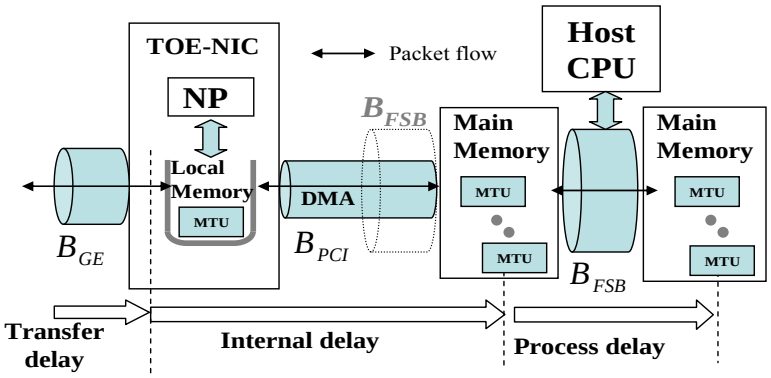



図 3.4 パケットフロー

$$B_{GE} < B_{PCI} < \frac{1}{2} * B_{FSB} \tag{3.9}$$

図 3.5 に端末バッファとそれらを結合するネットワークと端末内部バスの構成を示す. また, パケット転送最大サイズ MTU に対してのカスケード接続帯域数値を示す. 表 3.1 にネットワーク対端末内部帯域比を MTU の関数値として示す.

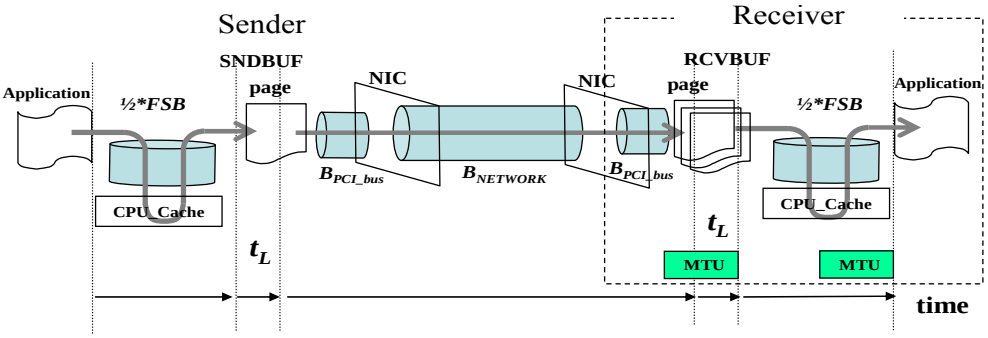



図 3.5　端末バッファとネットワークと端末内部バスの構成

E2Eシステムとして見た場合, 転送中の受信バッファ内のパケットとメインメモリへ転送完了したパケットの 2 つのパケットが, *RTT*内に存在する. したがって, 1 パケット当たりのスループット$T_{1P}$は, 式(3.10)で表される.

表 3.1　$B_{GE}$対端末内部帯域比

| *MTU* [Byte] | 1500 | 6000 | 9000 |
|---|---|---|---|
| $\frac{1}{1+\frac{2*B_{GE}}{B_{FSB}}+\frac{t_L*B_{GE}}{MTU}}$ | 0.433 | 0.594 | 0.610 |

$$T_{1P} = (\text{送信端末パケットMTU} + \text{受信端末パケットMTU}) / (\text{送信端末遅延} + \text{受信端末遅延}) \tag{3.10}$$

図 3.6 に 1 パケットの受信端末での, 遅延時間を示す. 送信端末の転送時間は受信端末から見ると受信端末のパケット待ち時間に等しいため, 受信端末での平均通過時間 *d* は式(3.11)のように求められる.

$$d = \text{パケット待ち時間} + \text{受信端末内部遅延} \tag{3.11}$$

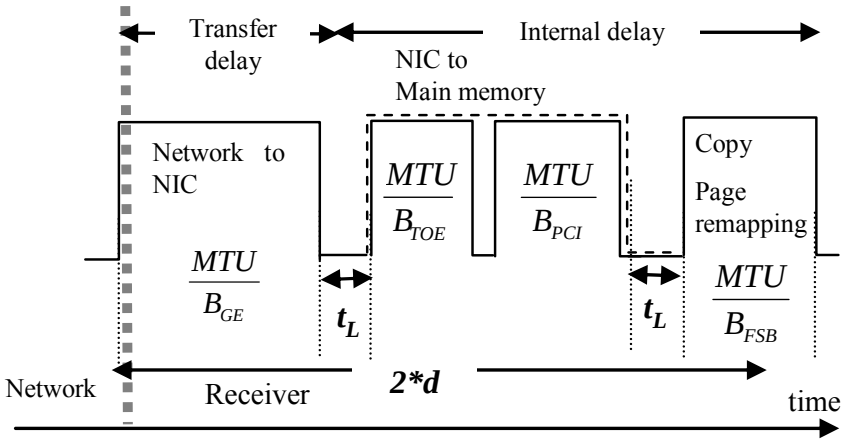



図 3.6 1 パケット通過遅延時間

図 3.6 において, $t_L$は$MTU$に依存しないTCPスタック処理時間とプロセス切り替え時間の総和を表し，$t_{CSUM}$はチェックサム演算時間である．特に，TOEなしのNIC(Non_TOE NIC)では，$t_{CSUM}$は 0 になる．式(3.11)の$d$を用いて受信端末でのスループット$T_{1P}$を求めると，式(3.12)のように表すことができる．

$$T_{1P} = \frac{2 * MTU}{\left(\frac{MTU}{B_{GE}} + t_L\right) + \left(\frac{MTU}{B_{GE}} + \frac{MTU}{1/2 * B_{FSB}} + t_{CSUM} + t_L\right)} < \frac{1}{\frac{1}{B_{GE}} + \frac{1}{B_{FSB}}} \tag{3.12}$$

### 3.3.2 一般化されたスループット

前節 3.3.1 の結果を$RTT$が大きい場合へ一般化する．$RTT$が$MTU$ / $B_{GE}$を超える場合には，送信端末のTCP輻輳ウィンドウ制御が有効に働き，確認応答が返ってくるまでに送信端末輻輳ウィンドウサイズ$cwnd$ [Byte] 相当のパケットを塊として送出する．図 3.8 に各層内のパケットフローとバッファ内のパケット数を示す. 確認応答を待たずに送出できるパケット数を$n$とすると，送信端末側の送信バッファ内パケット数とネットワーク中のパケット数との総和が$n$となるように輻輳制御される．同時に，受信側ではフロー制御が働き，TCP層に設けられたソケットバッファ内パケット数が$n$となるようにバッファリングされる[16],

[17]．ここで，前節 3.3.1 で定義した 1 パケット当たりの受信端末遅延$d$の他に次の時間量を定義し，タイミング関係を明確にする．まず，IP層内送信バッファ内，下り片側伝送路中，TCP層内受信ソケットバッファ中にあるパケット数をそれぞれ求める．次に，送信バッファから受信ソケットバッファまでの遅延時間を求め，スループットを算出する．送信端末輻輳ウィンドウ分のパケット送信終了時間を$t_{end}$，パケット転送空隙時間を$d_{void}$ ，確認応答が送信端末へ帰ってからパケットが受信端末に届くまでのインタラプト応答時間を$d_W$ ，ネットワーク上の遅延を$d_{ntwk}$とすると，$d_{void}$と$t_{end}$との関係は式(3.13)で表される．タイミング関係を図 3.7 に示す．

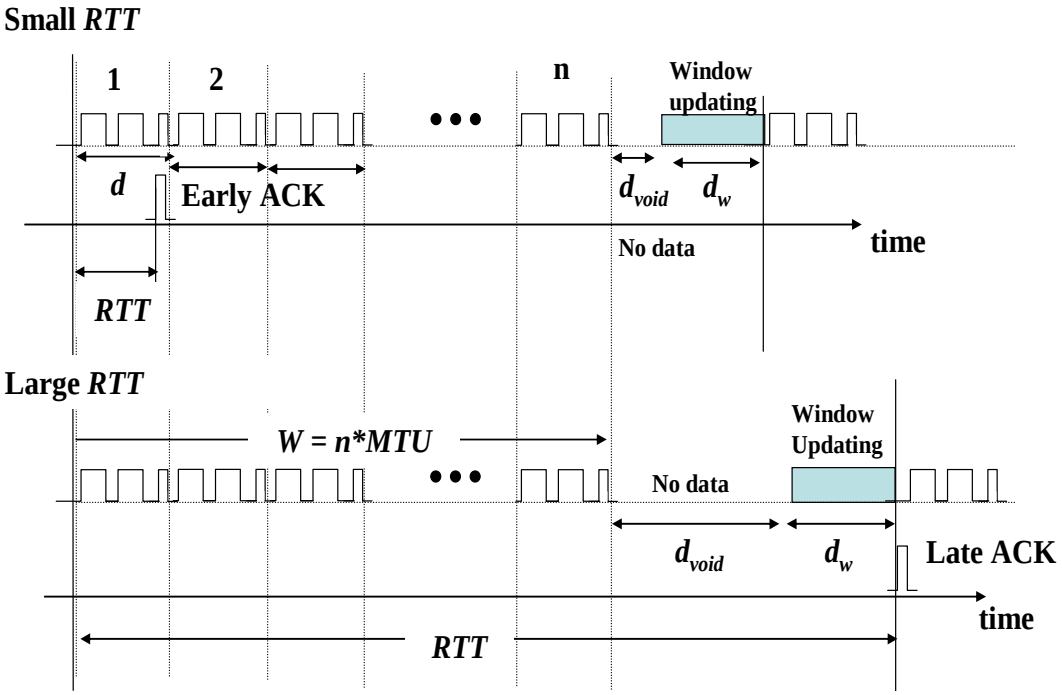



図 3.7 $d_{void}$, $d_W$ , $d_{ntwk}$ タイミング関係

また，図 3.8 に E2E 端末内部とネットワーク中のパケット数とそのタイミング関係を示す．

$$d_{void} = \frac{RTT}{2} - t_{end} \tag{3.13}$$

一般にTCPスループットはパケット到着間隔の変動による影響を受けて低下するが, 解析の容易性のため, クロストラフィックがなくパケット到着間隔が一定の理想的な通信を仮定すると，パケット転送空隙時間$d_{void}$は式(3.14)として求められる．周回遅延$RTT$は，送信端末で確認応答パケット到着時間を実測して知ることができる．

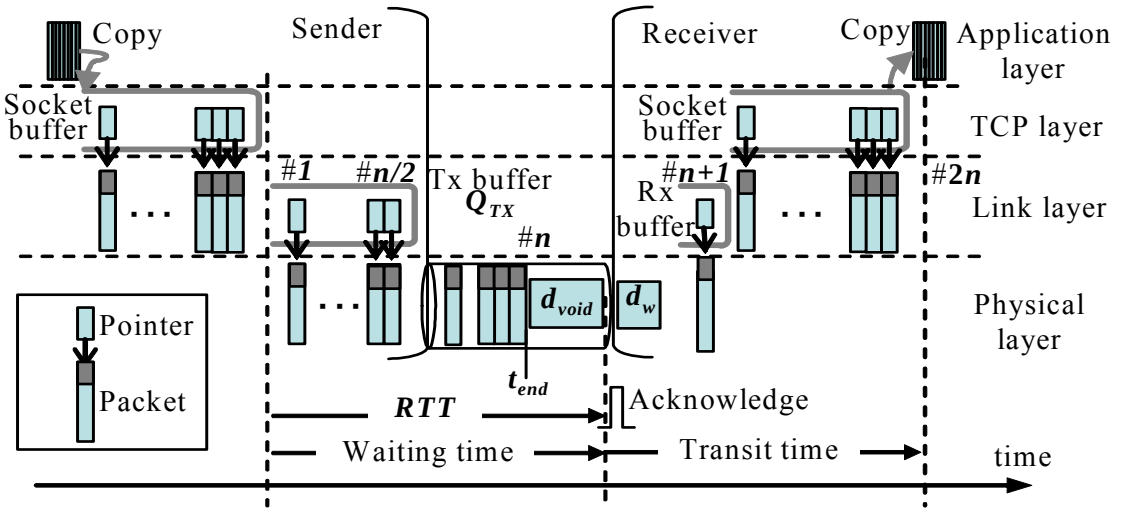



図 3.8 E2E システム内パケット数

$$t_{end} = n * \frac{MTU}{B_{GE}},\ RTT = \frac{2L}{0.66c} + \frac{MTU}{B_{GE}}$$
$$d_{void} = \frac{L}{0.66c} + \left(\frac{1}{2} - n\right) * \frac{MTU}{B_{GE}} \tag{3.14}$$

送信端末側の輻輳制御により，$d_{void}$は極小になるように制御され，このときの輻輳制御ウィンドウサイズ換算パケット数$n$は，式(3.15)より求められる．

$$n * \frac{MTU}{B_{GE}} = RTT \tag{3.15}$$

IP層には送信バッファが設けられ，ネットワーク層にパケットを送出する場合の速度調整が行われる．この送信バッファサイズを$Q_{TX}$として，輻輳制御状態にあり確認応答までに送出したパケット数$n$を用いて，IP層以下のスループット$T_{IP2N}$を求めると式(3.16)となる．

$$T_{IP2N} = \begin{cases} \dfrac{Q_{TX} + B_{GE} * RTT/2}{2 * \dfrac{Q_{TX}}{B_{GE}}} & \dfrac{RTT}{2} \le \dfrac{Q_{TX}}{B_{GE}} \\ \dfrac{2 * Q_{TX}}{\dfrac{Q_{TX}}{B_{GE}} + \dfrac{RTT}{2}} & \dfrac{RTT}{2} > \dfrac{Q_{TX}}{B_{GE}} \end{cases} \tag{3.16}$$

$T_{IP2N}$が最大となるのは，$Q_{TX}$が式(3.17)を満足する場合であることが分かる．

$$Q_{TX} = \frac{B_{GE} * RTT}{2} = \frac{n * MTU}{2} \tag{3.17}$$

すなわち，送信バッファ中のパケット数とネットワーク中の下り片側伝送路中に存在する受信端末へ未到達であるインフライト状態のパケット数が等しくなるようにバッファリングされるとき，送信端末から見たネットワークへのスループットが最大になり，この結果，上位層 TCP スループットも上限値をとる．

一方，受信端末ソケットバッファサイズ*awnd*［Byte］は，送信バッファ 内パケットとインフライトパケットの合計を連続してバッファリングする必要があり，$Q_{TX} + n*MTU/2 = 2*Q_{TX}$必要となる．受信端末内に設置された受信ソケットバッファが限界に達すると，停止情報が送信端末に確認応答パケットに付加される形態で送信端末に伝えられ，送信ソケットバッファからの送信が停止する（センドストール）．受信端末は，あらかじめ$2*Q_{TX}$のサイズに設定されたソケットバッファサイズを送信端末に通知し，送信端末との間でフロー制御を行う．送信端末は，センドストールが発生しないように，輻輳ウィンドウサイズ*cwnd*を(3.18)で示す値に設定する．

$$cwnd \geq awnd = 2 * Q_{TX} \tag{3.18}$$

*cwnd*，*awnd*の条件式(3.18)下において，ネットワーク・端末総合系内にあるパケット総数を求め，そのパケット数分の遅延時間からスループットを算出する．送信端末送信バッファ中に$Q_{TX}$，ネットワーク下り片側伝送路中に$n*MTU/2$，受信端末ソケットバッファ中に$awnd=2*Q_{TX}$ ，3 箇所に含まれるパケット数は$2n$あり，総伝送量は式(3.19)で与えられる．

$$Q_{TX} + \frac{n * MTU}{2} = 2 * Q_{TX} = 2n * MTU \tag{3.19}$$

次に，遅延時間の総計を求める．受信端末からみたパケット到着待ち時間$t_{WAIT}$は，式(3.20)となる．

$$t_{WAIT} = n * d_{ntwk} + d_{void} / 2 + d_W / 2 = \frac{RTT}{2} \tag{3.20}$$

受信端末ソケットバッファの通過時間$t_{BUFF}$について，このバッファが送信端末の送信バッファ内パケットとネットワーク下り片側伝送路中のパケットとの総数をバッファリングすることから，次式(3.21)が成り立つ．

$$t_{BUFF} = t_{WAIT} \tag{3.21}$$

送信端末内部における$Q_{TX}$へのデータ転送と受信端末内ネットワーク層からTCP層へのパケット転送は，実データ転送ではなく，ソケットバッファポインタ構造体のコピーを取ることにより論理的に行われる． BSD，Linux等の一般的実装では，ソケットバッファポインタ構造体は，通常 4[Byte]境界 248[Byte]となっており，ポインタ構造体を生成するのに必要な時間は，4[Byte]読み書き間接アドレシング命令と 2[Byte]オペランドをプログラムカウンターへロードする回数対CPUインストラクション実行速度比から，次式(3.22)のように求められる．

$$\frac{(1+1/2)*248*n}{1/4*B_{FSB}} = \frac{372*n}{1/4*B_{FSB}} \tag{3.22}$$

次に，MAC 層，IP 層，TCP 層のヘッダ処理時間を求めると，パケットギャップ 12[Byte]，MAC アドレス 12[Byte]，IP アドレス 20[Byte]，TCP アドレス 20[Byte]，計 64[Byte]，読み書き間接アドレシング回数 52/2 回から式(3.23)となる．

$$\frac{64*n+26*n}{1/4*B_{FSB}} = \frac{90*n}{1/4*B_{FSB}} \tag{3.23}$$

さらに，チェックサム確認のため，パケットはNIC内NPまたは，ホストCPUによる計算が実行される． 1 パケット*MTU*[Byte]に対し，IPヘッダから作られる 12[Byte]擬似ヘッダ，TCPヘッダ 20[Byte]，ペイロード(*MTU*-40)[Byte]の総計(*MTU*-8)[Byte] のデータの総和を計算するのに必要なインストラクション数は，2[Byte]単位のチェックサムに対し，桁上がりを高速処理するため 4[Byte]単位の加算 16 回を 1 塊で実行し，最後に 2[Byte]の和に変換する演算数となる．ただし，まとめ加算の回数を求めるために 6 インストラクション，更に 4[Byte]加算を 2[Byte]加算に折り返すのに 6 インストラクション必要である．1 インストラクションは 4[Byte]，またOSのオー

バヘッドが数バイト加わるので, この演算量$P_{CSUM}$[Byte]は, 式(3.24)で与えられる[19]. 記号⌈ ⌉は，小数点以下切り上げによる整数化関数である．

$$P_{CSUM} = 12 * 4 + \left\lceil \frac{(MTU - 8)}{4 * 16} \right\rceil * (2 * 16 + 1) \cong \frac{MTU}{2} + 50 \tag{3.24}$$

$n$パケットのチェックサムに要する時間は，TOE-NICのネットワークプロセッサの処理速度を$F_{TOE}$とすると，式(3.25)となる．

$$t_{CSUM} = \frac{n * P_{CSUM}}{F_{TOE}} \tag{3.25}$$

一方，Non_TOE-NIC では，ホスト CPU がこれを実行するため式(3.26)となる．

$$t_{CSUM}^{Non-TOE} = \frac{n * P_{CSUM}}{B_{FSB}} \tag{3.26}$$

最後に, アプリケーション層のデータに対応したページ単位のデータを渡すために TCP 層に設けられたソケットバッファからアプリケーション層へデータがコピーされる．コピーに要するデータ転送時間は全てホスト CPU による時間遅延である．コピー時間は式(3.27)で与えられる．

$$copy\_time = \frac{n * MTU}{1/2 * B_{FSB}} \tag{3.27}$$

また，パケット単位の処理に対して，プロセスの切り替え時間$t_{psw}$の$n$倍がオーバヘッドとして加わる．

$$Overhead\_time = n * t_{psw} \tag{3.28}$$

Non_TOE-NICを搭載した端末を用いた場合のスループット $T_{TCP_nonTOE}$は，TCP処理をすべてホストCPUが実行するため，NIC内でDMA転送のために 4 ページ分，16K [Byte]（$MTU$=9Kに対応）以上がバッファされ，式(9)の仮定の条件下，PCIバス経由メインメモリへパケットが転送

される．遅延時間の式(3.22)，式(3.23)，式(3.26)，式(3.27)，式(3.28)を合計し，以下の式(3.29)で表される[18].

$$T_{TCP\_NonTOE} = \frac{2n * MTU}{t_{WAIT} + t_{BUFF} + \frac{P}{B_{GE}} + \left( \frac{n * P_{CSUM} + 2n * MTU + 1848n}{B_{FSB}} \right) + \frac{n * MTU - P}{B_{PCI}} + n * t_{psw}} \quad (3.29)$$

輻輳制御が働くと，連続して送出することのできるパケット数 $n$ は，送信端末輻輳制御ウィンドウサイズ $cwnd$ から決まり，式(3.30)を満足する．記号$\lfloor\ \rfloor$は，小数点以下切捨てによる整数化関数である．

$$1 \le n = \left\lfloor \frac{cwnd}{2 * MTU} \right\rfloor, \ t_{WAIT} + t_{BUFF} + \frac{P}{B_{GE}} = RTT \quad (3.30)$$

Non_TOE NICに対して$n$を$cwnd$で置き換え，式(3.20)において$d_W/2 = P/B_{GE}$ ，式(3.21)から，式(3.31)が得られる．

$$T_{TCP\_NonTOE} = \frac{1}{\frac{RTT}{cwnd} + \left( \frac{949}{B_{FSB}} + \frac{t_{psw}}{2} \right) * \frac{1}{MTU} + \left( \frac{1.25}{B_{FSB}} \right) + \frac{1 - \Delta}{2 * B_{PCI}}} \quad (3.31)$$

ここで，$\Delta$ は式(3.32)を満たす定数である．$n$ は送信端末送信バッファからネットワークを経由して受信端末メインメモリまでにあるパケット数であり，NIC 内のバッファ $P$ は，4 K[Byte]以上，最大$n/2 * MTU$ [Byte]必要である．この条件から，次式(3.32)が成り立つ．

$$\frac{4096}{n * MTU} < \Delta = \frac{P}{n * MTU} \le \frac{1}{2} \quad (3.32)$$

TOE-NIC搭載の端末では, 同様にデータを NIC上ローカルメモリから端末メインメモリへNPによるDMA転送を行うが, 各層のプロトコルヘッダーが除かれたMSSのデータ転送となり遅延時間は最大約 0.96 倍短縮される．さらに，ソケットバッファポインタ構造体は設けないので，これによる処理遅延時間式(3.22)の増加はない．式(3.20)，式(3.23)，式(3.25)，式(3.27)，式(3.28)

を合計し，TCPスループット$T_{TCP_TOE}$は式（3.33）[20]，[21]，[22]，[23]となる．

$$T_{TCP\_TOE} = \frac{2n*MTU}{t_{WAIT} + t_{BUFF} + \frac{P}{B_{GE}} + \frac{n*(P_{CSUM}+360)}{2*F_{TOE}} + \frac{2n*MTU}{B_{FSB}} + 0.96*\frac{n*MTU-P}{2*B_{PCI}} + n*t_{psw}} \quad (3.33)$$

NIC でのパケット処理遅延が，パケット伝送遅延より小さい条件式(3.34) の場合には，式(3.33)は，式(3.35) となる．

$$\frac{n*(MTU/2+50)}{F_{TOE}} < \frac{n*MTU}{B_{GE}} \quad (3.34)$$

$$T_{TCP\_TOE} = \frac{1}{\frac{RTT}{cwnd} + \left(\frac{205}{F_{TOE}} + \frac{t_{psw}}{2}\right)*\frac{1}{MTU} + \left(\frac{1}{B_{FSB}} + \frac{1}{8*F_{TOE}}\right) + \frac{0.12}{B_{PCI}}} \quad (3.35)$$

TOE-NICの処理速度が十分ではなくパケット処理遅延が, パケット伝送遅延より大きくなると，ローカルメモリの受信バッファが満杯となり，フロー制御が働き，$cwnd/F_{TOE}$ [sec]後にACKが送信端末に返ることになる． $cwnd/F_{TOE}$の値は, ネットワーク遅延$RTT$より大きくなり, 式(3.36)が成り立つ.

$$\frac{RTT}{cwnd} < \frac{1}{F_{TOE}} \quad (3.36)$$

この場合は，式(3.35)の分母第一項が，式(3.36)で示す$F_{TOE}$で置き換えられため，スループット値は低くなる．$F_{TOE}$は，NPの性能から決まる．

図 3 で定義した$t_L$を$B_{FSB}$を用いて表すと，Non_TOE-NICの場合は式(3.37)となる．

$$t_{L\_NonTOE} = \frac{949}{B_{FSB}} + \frac{t_{psw}}{2} \quad (3.37)$$

TOE-NIC の場合は式(3.38)となる．

$$t_{L\_TOE} = \frac{205}{F_{TOE}} + \frac{t_{psw}}{2} \tag{3.38}$$

$t_{L_nonTOE}$と$t_{L_TOE}$を用いて，式(3.31)と式(3.35)は，式(3.39)と式(3.40)のように表現できる．

$$T_{TCP\_NonTOE} = \frac{1}{\frac{RTT}{cwnd} + \left( \frac{1.25}{B_{FSB}} + \frac{0.25}{B_{PCI}} \right) + \frac{t_{L\_NonTOE}}{MTU}} \tag{3.39}$$

$$T_{TCP\_TOE} = \frac{1}{\frac{RTT}{cwnd} + \left( \frac{1}{B_{FSB}} + \frac{1}{8 * F_{TOE}} + \frac{0.12}{B_{PCI}} \right) + \frac{t_{L\_TOE}}{MTU}} \tag{3.40}$$

上式(3.39)と(3.40)分母第 1 項は送信端末の輻輳制御によるパケット塊の受信端末までの遅延時間，第 2 項括弧内は受信端末データ転送時間，第 3 項は 1 パケットのレイテンシーとTCPスタック受信端末処理時間の和に相当する．両式(3.39)と(3.40)で表される上限値について，スループット上限値と*RTT*との関係について以下のことが分かる．両式分母において，*cwnd*はネットワーク固有の*RTT*値の下に制御される変数であり，輻輳制御によりパケットを連続した塊として送出するように働く．ネットワークの空間的な広がりを示す帯域遅延積$B_{GE}$×*RTT*は*awnd*[Byte]に等しく，*cwnd*は*awnd*以下であり，パケット再送が発生する時点での*cwnd*は．パケット塊の帯域遅延積$MTU/\sqrt{p}$に等しい．従って，*cwnd*の到達最大サイズは式(3.41)で表され，*RTT/cwnd*の最小値は式(3.42)となる．

$$cwnd = \max\left( B_{GE} * RTT , \frac{MTU}{\sqrt{p}} \right) \tag{3.41}$$

$$\frac{RTT}{cwnd} \geq RTT \Big/ \max\left( B_{GE} * RTT, \frac{MTU}{\sqrt{p}} \right) = 1 \Big/ \max\left( B_{GE}, \frac{MTU}{\sqrt{p} * RTT} \right) \tag{3.42}$$

最大セグメントサイズ*MSS*（Maximum Segment Size:1460）[Byte]より小さな*MTU*に対しては，

IP フラグメンテーションが起こるため，1 *MTU* サイズのパケット数は$\lfloor MSS/MTU \rfloor$となり，1 *MSS*タイムスロットの*MTU*単位のパケット数は，式(3.43)となる．記号$\lfloor\ \rfloor$は小数点切捨てによる整数化関数である．式(3.39)と(3.40)のTCP上限値$T_{TCP_UPPER}$は式(3.44)と(3.45)のように書き換えられる．両式(3.44)，(3.45)において，分母第 1 項の最小値の境界となる*RTT*は，

$$\alpha = \left\lfloor \frac{MSS}{MTU} \right\rfloor \Bigg/ \left\lfloor \frac{MSS + 40 * \left\lfloor \frac{MSS}{MTU} \right\rfloor}{MSS} \right\rfloor = \begin{cases} 2 & MSS = 1460 \ , \ MTU = 500 \ [Byte] \\ 1 & MSS = 1460 \ , \ MTU \geq 1000 \ [Byte] \end{cases} \tag{3.43}$$

$$T_{TCP\_NonTOE\_UPPER} = \frac{1}{\min\left(\frac{1}{\alpha * B_{GE}}, \frac{\sqrt{p} * RTT}{\alpha * MTU}\right) + \frac{1.25}{B_{FSB}} + \frac{0.25}{B_{PCI}} + \frac{t_{L\_NonTOE}}{MTU}} \tag{3.44}$$

$$T_{TCP\_TOE\_UPPER} = \frac{1}{\min\left(\frac{1}{\alpha * B_{GE}}, \frac{\sqrt{p} * RTT}{\alpha * MTU}\right) + \frac{1}{B_{FSB}} + \frac{1}{8 * F_{TOE}} + \frac{0.12}{B_{PCI}} + \frac{t_{L\_TOE}}{MTU}} \tag{3.45}$$

ネットワーク帯域遅延積$B_{GE}$×*RTT*とパケット塊の帯域遅延積*MTU*/$\sqrt{p}$が等しいときであり，次ぎに示す条件の場合である．$\sqrt{p}$ =$10^{-3}$，$B_{GE}$=$10^9$ [Gbps], *MTU*=1500[Byte]，*RTT*=12[msec]．また式(3.44)と(3.45)は，*MTU*サイズのパケットが，ネットワーク中と端末内部のバッファ中に帯域制限を受け隙間なく充填される場合の上限を与えるものである．一方，$p$ が有限で，*RTT*が 12 [msec]より大きい領域では，相対的に$B_{FSB}$ の影響が小さくなることから，従来モデルとの差が縮まり，式(3.5)に示す従来モデルによるスループットに近づく．従来モデルにおけるスループットは，式(3.44)と(3.45)において，式(3.46)に示すように受信端末内部バス帯域$B_{FSB}$がネットワーク帯域$B_{GE}$に比べ十分大きい場合に相当する．式(3.44)と(3.45)は従来モデル(3.5)に対してもより一般的な式となり，式(3.46)が成立し，従来TCPモデルより厳密な上限値を与える．

$$T_{TCP\_NonTOE}, T_{TCP\_TOE} < \frac{1}{\left(\frac{RTT}{cwnd}\right)} = \frac{cwnd}{RTT} = T_{conv} \tag{3.46}$$

図 3.9 に，$\sqrt{p}$ =$10^{-3}$，Non_TOE-NIC（RealTek8139）搭載端末（CPU 2.4GHz-Pentium4 ：$B_{FSB}$=6.5[Gbps], $B_{PCI}$= 1 [Gbps]， $t_{L\_NonTOE}$=1.2[μsec]）の場合，式(3.44)で表される上限値を，MTU サイズと*RTT*の関数として示す.

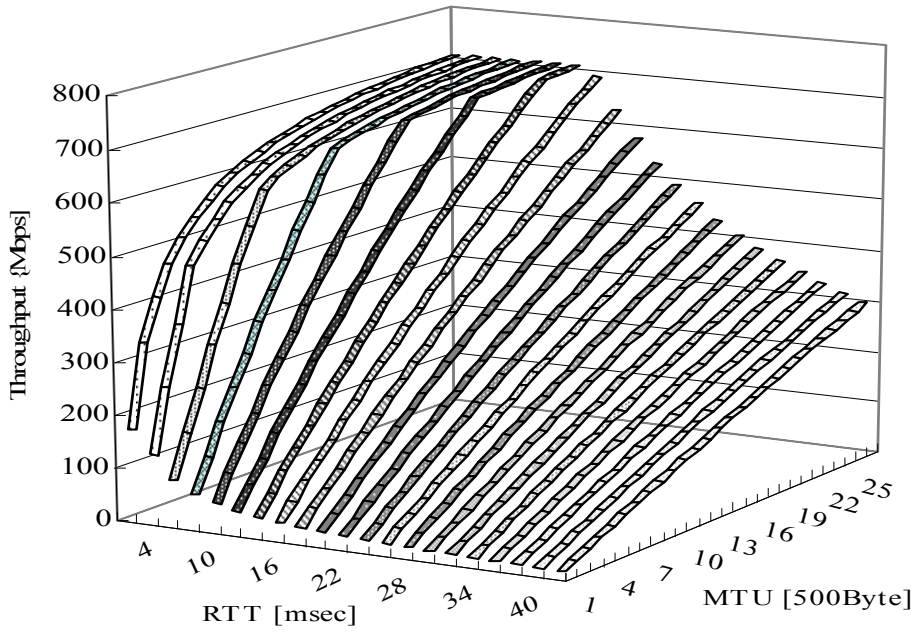



図 3.9　Non_TOE-TCP スループット上限値

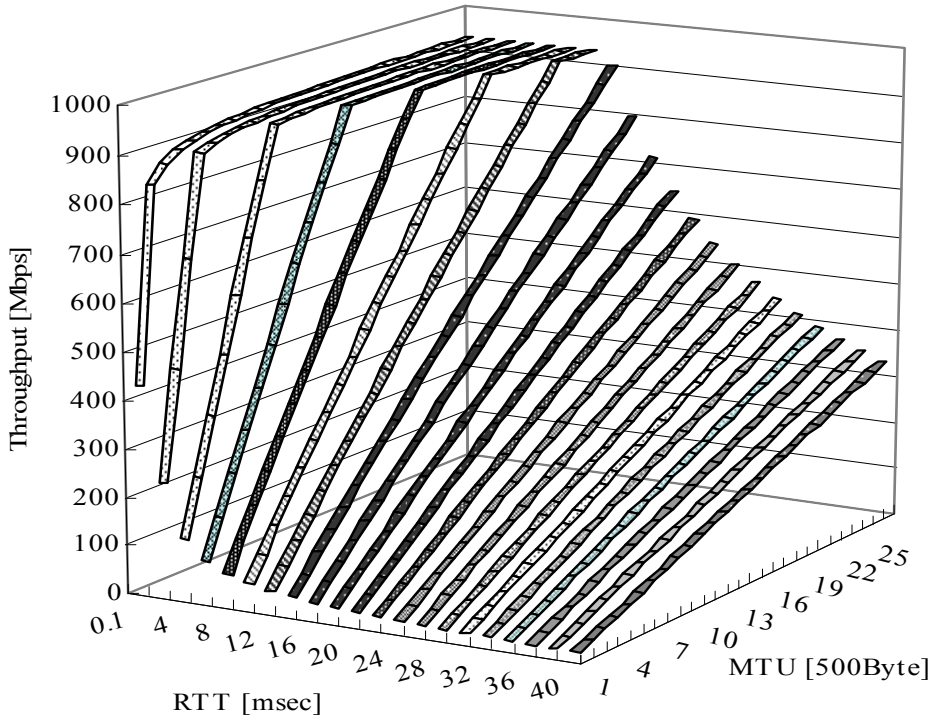



図 3.10　TOE-TCP スループット上限値

次に，TOE-NIC（Broadcom5701）搭載端末（CPU 3.2GHz-PentiumD-EE，$B_{FSB}$=12[Gbps], $F_{TOE}$=1[GIPS]=4[GBps]=32[Gbps],$B_{PCI}$=2.5[Gbps]，$t_{L\_TOE}$=0.1[μsec]），$\sqrt{p}=10^{-3}$の場合の式(3.45)を，図 3.10 に示す.

両式(3.44)と(3.45)で表される端末の上限値は，$B_{PCI}$にも依存することが分かる．通常，$B_{PCI}$は，シングルチャンネルPCI-Expressでは 2.5[Gbps]，PCI-X2.0-QDRは，16.8[Gbps]である．また，*cwnd*の最小値は，端末間距離の非常に短い場合（1m以内）の*RTT*実測値（pingコマンド使用）から，10×*MTU*となり，次式(3.47)が成立する.

$$\left(\frac{1.25}{B_{FSB}}+\frac{0.25}{B_{PCI}}\right)+\frac{t_{L\_NonTOE}}{MTU}>\left(\frac{1}{B_{FSB}}+\frac{1}{8*F_{TOE}}+\frac{0.12}{B_{PCI}}\right)+\frac{t_{L\_TOE}}{MTU}$$

$$\therefore T_{TTCP\_nonTOE}<T_{TCP\_TOE} \tag{3.47}$$

TOE-NIC 搭載の場合の方が，スループット値は高くなる.

## 3.4 受信端末内部遅延の測定

理論値を検証するために，2台のNon-TOE搭載端末I(OS:Linux Fedora Core 2.4.20[38], PC:Pentium4 2.4GHz, NIC:RealTek8139 Non_TOE, TCP NewReno)を*RTT*が十分小さくなるネットワークに接続し，Non_TOE受信端末パケット通過時間実測値から得られるスループット値と導出した理論上限値式(44)から得られる値との比較を行った．次に，別のTOE搭載端末II2台(OS:Linux Fedora Core 2.6.20,PC:PentiumD-EE 3.2GHz,PCI-Express BUS，NICBroadcom5701-TOE, TCP NewReno)を用いて，理論式(3.45)に対する理論上限値式評価を行った.

### 3.4.1 受信端末でのパケット通過時間測定

受信端末内部のパケット通過時間は，式(3.44)，式(3.45)の分母において，受信端末へパケットが到達するまでの待ち時間$MTU/B_{GE}$，アプリケーション層への転送時間$MTU/B_{FSB}$とソケットバッファの処理時間$t_{L_NonTOE}$ の 3 種類の遅延時間の合計になっている．受信端末内部の 1 パケット通過時間*d*は, スループットを与える式(3.45)における分母第 3 項までの*MTU*倍で表され, Non-TOE端末の場合は式(3.48)，TOE端末の場合には式(3.49)となる．受信端末における

遅延時間 $d$ を図 3.11 に示す.

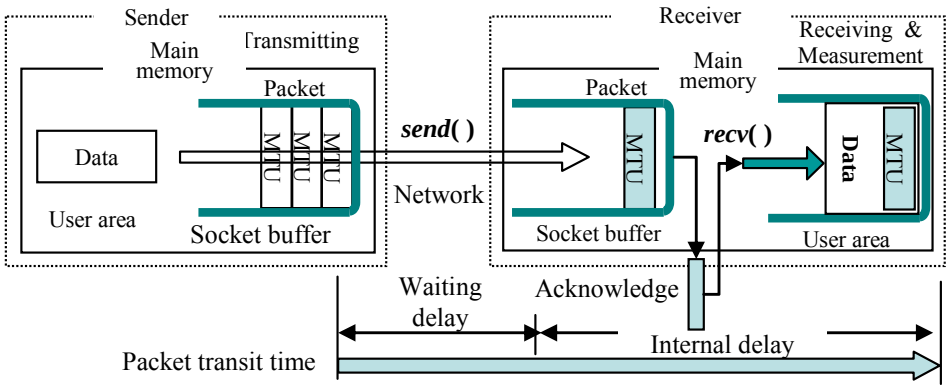



図 3.11　受信端末遅延 $d$

$$d = \frac{MTU}{B_{GE}} + \left( \frac{1.25}{B_{FSB}} + \frac{0.25}{B_{PCI}} \right) * MTU \ + t_{L\_NonTOE} \tag{3.48}$$

$$d = \frac{MTU}{B_{GE}} + \left( \frac{1}{B_{FSB}} + \frac{1}{8 * F_{TOE}} + \frac{0.12}{B_{PCI}} \right) * MTU + t_{L\_TOE} \tag{3.49}$$

TCPスループット上限値と$d$の関係は，式(3.44)から，式(3.50)で表される．

$$T_{TCP\_UPPER} = \frac{1}{\left( \frac{d}{MTU} \right)} \tag{3.50}$$

式(3.48)，式(3.49)で求められたパケット通過時間$d$は，パケット転送を行う端末内のプログラムの流れから見ると，受信端末に実装されているシステム関数*recv*( )の実行時間に等しくなっているため，プログラムによる測定が可能である．このことは，受信端末でのパケット通過時間から，TCPスループットが求められ，TCP通信中実時間で系の帯域推定が可能であることを示している．次に，$d$を求めるため，パケット受信を実行するシステム関数*recv*( )の開始終了点に時間測定システム関数*gettimeofday*( )を付加したパケット送出用プログラム（BM_client）とパケット受信･通過時間測定プログラム（BM_server）を用い，実時間で通過時間を測定し推測するために，以下4項目の標準カーネルTCPプログラムに変更を加えた．

(1) 標準実装TCPスロースタート処理の禁止

(2) 初期ウィンドウサイズ 10

(3) 送信輻輳制御ウィンドウ初期値拡張 10 パケット

(4) ソケットオプション SO_RCVLOWAT 値をパケットサイズに設定

これらのプログラムを用い，TCP通信を実行し実時間（TCP転送時間内）統計処理[24]を行って$d$を求めた. 受信端末システムコールrecvの実行時間を測ることにより通過時間$d_i$を計算する. 図 3.12 に通過時間測定プログラムの擬似コードを示す.

測定値は，以下の 3 つの場合に分類できる．SO_RCVLOWAT 値で指定されるパケット 1 個分の時間, 2 個分の時間, 極端に短いパケットコピーのみの時間である. 10 等分の階級値を設け，最小ビンと最大ビンに入った標本は異常値として，統計計算から排除した後，これらの場合に相当するパケット個数をプログラムから求め，以下の処理を施す．標本値は正常（大部分）値の集合と 2 倍値（パケット 2 個分）の集合に大別できる.

```
BM_Server
        i=0
        while (rc_i > 0) {
              gettimeofday( start ) = t^s_i ;
              rc_i = recv( ) ;
              gettimeofday( end ) = t^e_i ;
              i++ ;
        }
```

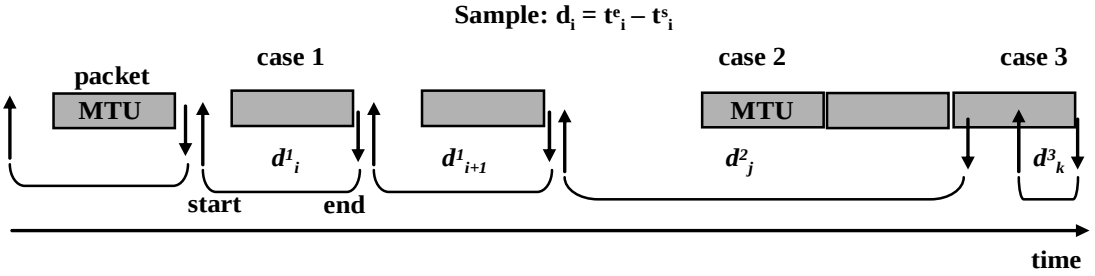



図 3.12 通過時間測定擬似コード

最頻度値を計算し，そのビンに入る標本を選択し，平均値をとり，その平均値をパケット通過時間$d$とする. 受信端末通過時間によるTCPスループットは式(3.50)で表される. なお, $B_{GE}$に対し, 統計を取るのに最低限必要なデータ数を 10 パケット程度とすると，それらの通過時間は 120μ秒程度である. CPU としてクロック周波数 3.2GHzのPentiumDクラスのものを考えると, 時間計測のためのアセンブリコード実行時間は，タイマー値読み取りとメモリ格納 1 回につき数十n秒を要することから，数百n秒となる．また，10 個のパケットの統計処理ステップ数はメモリアクセス 30 回(2 値読み

取り 1 値書き込み各 10 回)，統計演算 6 種，1 演算各 4 ステップと仮定すれば 1800 程度になることから，処理に要する時間は高々数μ秒であり，パケットの通過時間に比べて十分小さいと言える．

### 3.4.2 フロントサイドバス(Front Side Bus)帯域実測

FSB帯域$B_{FSB}$ は，ネットワークとは独立したPC固有量であるため，BM_clientとBM_serverを 1 台の受信端末PC上で使用するループバック試験から実測により求めることができる．ループバック試験のパケットフローを図 3.13 に示す．送信受信端末間のネットワークは，ループバックデバイスという仮想論理デバイスに置き換えられ，これに対するパケット処理は，ポインタ処理のみでネットワークを介在せず，高速に行われる．式(3.40)の分母第二括弧項を$B_{int}$とすると，ポインタの転送はループバックでは，PCIバスを経由しないが，ポインタの読み出しと書き込みが加わるので，測定遅延時間$d$と$B_{int}$の関係は式(3.51)となる．

$$\frac{1}{B_{\text{int}}} = \frac{1.25}{B_{FSB}} + \frac{0.25 \times 2}{\left(B_{FSB}/2\right)} = \frac{2.25}{B_{FSB}} \tag{3.51}$$

ループバック測定では，CPU とメインメモリ間のみの実コピー処理が実行されるため，フロントサイドバス帯域を転送速度から求めることができる．表 3.2 に測定値を示す．

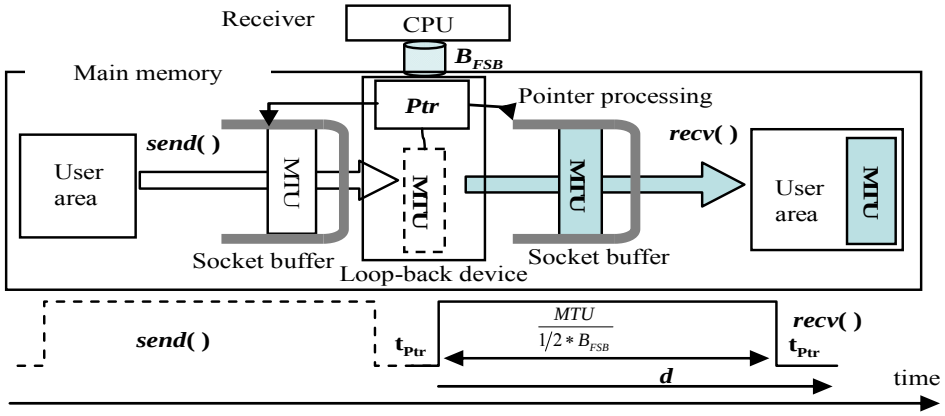



図 3.13　ループバック測定

Non_TOE端末I（通常PCIバス）に対しては，*MTU*サイズを変えて対応する受信端末通過時間$d$

を測定（測定値を表 2 に示す）し，式(3.52)から，式(3.53)に示す線形補間を用いて，FSB帯域$B_{FSB}^{NonTOE}$を算出した．

$$\frac{MTU}{d}=\frac{2*MTU}{\frac{MTU}{1/2*B_{\text{int}}}+2*t_{L\_NonTOE}}$$

$$\therefore \frac{1}{d}=\frac{B_{\text{int}}}{1+t_{L\_NonTOE}*B_{\text{int}}} \tag{3.52}$$

$$\frac{1}{d}=\frac{B_{\text{int}}}{MTU+t_{L\_NonTOE}*B_{\text{int}}}=\frac{552.648\times10^{6}}{MTU+5217.15}$$

$$B_{\text{int}}=552.648\times8/1.1=4.2\ [Gbps] \tag{3.53}$$

$$B_{FSB}^{Non\_TOE}=2.25\times4.2=9.5[Gbps]$$

TOE端末II（PCI-Express）に対しては，同様の計算から，FSB帯域$B_{FSB}^{TOE}$は式(3.54)となる．

$$B_{FSB}^{TOE}=9.5\times2.25=21[Gbps] \tag{3.54}$$

表 3.2 ループバック試験結果

| Resultt of FSB Bandwidth measurement | | | | |
|---|---|---|---|---|
| FSB BW(MB/s) | Equivalent MTU (Byte) | Recv( ) LOWAT(Byte) | Send( ) SNDBUF | Measured Time difference ( $\mu$ sec) |
| 276 | 7000 | 6940 | 6940 | 29 |
| 289 | 8000 | 7940 | 7940 | 29 |
| 314 | 9000 | 8940 | 8940 | 29 |
| 322 | 10000 | 9940 | 9940 | 29 |
| 356 | 11000 | 10940 | 10940 | 29 |
| 392 | 12000 | 11940 | 11940 | 30 |
| 404 | 13000 | 12940 | 12940 | 33 |
| 427 | 14000 | 13940 | 13940 | 33 |
| 433 | 15000 | 14940 | 14940 | 33 |
| 472 | 16000 | 15940 | 15940 | 34 |

## 3.5 TCP スループット上限値の実測評価

前章と同様の実験システムを用いて *MTU* を変化させながら *d* を測定し，式(3.49)で表される Non_TOE 受信端末パケット通過時間測定による帯域実データと，式(3.44)で表される *MTU* 依存性の TCP 上限理論値との比較検証を行った．さらに比較には，標準的に用いられている帯域測定プログラム Iperf-ver2.0.2[25]を参考として用いた．Iperf 測定に必要なパラメータである輻輳制御ウィンドウサイズ *cwnd* は，以下の条件を満足するように設定する．

遅延時間$RTT$ をPC内蔵pingコマンドにより実測した結果は，$RTT \approx 0.12$[msec]であった．式(3.39)の分母第一項$cwnd / RTT$は$B_{GE}$ を超えることがないため，パケット数$n$には以下の式(3.55)を満足するような最大の値を選ぶ．

$$\frac{cwnd}{RTT} = \frac{n * MTU}{RTT} < B_{GE}$$
$$\therefore \quad n < \frac{RTT * B_{GE}}{MTU} = 10 \tag{3.55}$$

Iperfは，実時間計測ではなく，10 秒間テストパターンを送出 しデータ転送量と時間の比率をスループットにしているため，輻輳制御開始時のスロースタートの影響を含み，式(3.44)による理論上限値より低い値を示す．なお，式(3.37)の$t_{L_Non\text{-}TOE}$は，LMbench（benchmarkプログラム）[26]を用いて実測した．

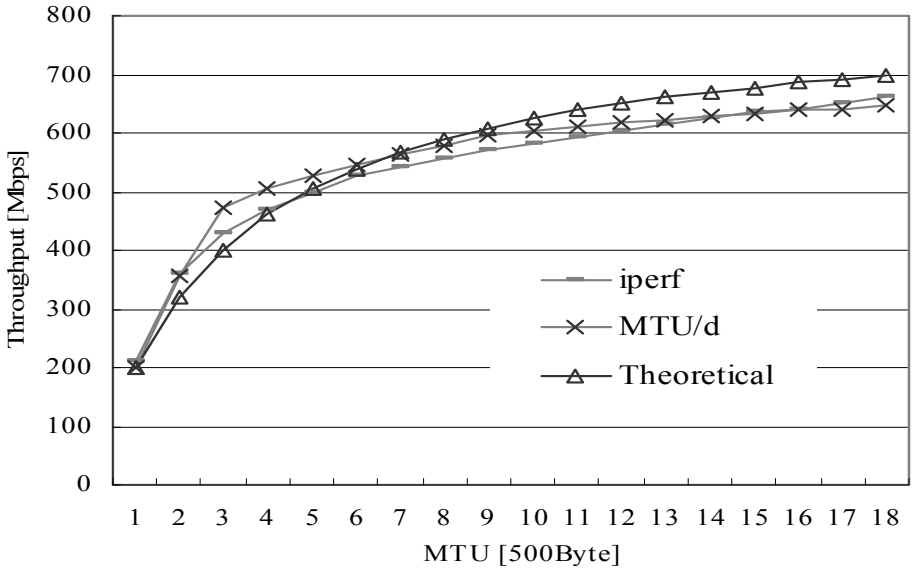



図 3.14　Non_TOE-TCP スループット上限値

図 3.14 に，Non_TOE-NIC受信端末Iに対する式(3.44)による理論上限値，パケット通過時間実測式(3.50)による上限値，Iperf（測定評価プログラム）による上限値を，*MTU*をパラメータにして示す．理論上限値は，*MTU*の関数式(3.57)を用いている．式(3.57)は，理論上限値式(3.44)において，*MTU*の最小サイズ 500 バイトの*m*倍をパラメータにし，以下の既知量を代入し得られる．前節 3.4.2 のFSB実測値を用い，$B_{FSB}{}^{Non\_TOE}$ = 9.8 [Gbps]，$B_{PCI}$ =1.06 [Gbps]，仕様値から$B_{GE}$ =1 [Gbps]， *MTU*=500×*m* [Byte]と置き， $949/B_{FSB}/MTU = 0.2/m$，
$1/B_{GE} + 1.25/B_{FSB} + 0.24/B_{PCI} = 1.36$, $t_{L\_NonTOE}/MTU \cong 1.2/m$．*MSS*より小さい*MTU*に対しては，式(3.43)で決まる $\alpha$ を用いた．送信端末輻輳制御ウィンドウサイズ*n*はTCP層でのデータ塊サイズ*MSS*から算出されるため，式(3.39)と式(3.40)の分母第二項以降($D_2$+$D_3$）は*MSS/MTU*に比べて小さいため，式(3.56)が成り立つ．

$$\min\left(\frac{RTT}{cwnd}\right) = \frac{MSS}{\alpha * MTU} >> D_2 + D_3 \tag{3.56}$$

式(3.39)に求められた数値を代入し，Non_TOE 端末の場合のスループットは，式(3.57)となる．

$$\begin{aligned} T_{non\_TOE} &= 1024 * \frac{MTU}{MSS} * \alpha && MTU < MSS \\ &= \frac{1024}{1.36 + 1.2 \Big/ \left(\frac{MTU}{MSS}\right)} && MTU \geq MSS \end{aligned} \tag{3.57}$$

次に， TOE-NIC 受信端末 II に対する理論上限値（式(3.45)による），パケット通過時間実測式(3.50)による上限値，Iperf による上限値を，*MTU* をパラメータにして，図 3.15 に示す．理論上限値は，理論式(3.45)において，*MTU* 値以外の定数に以下の値を代入した *MTU* の関数式(3.58)で表される．

$$\begin{aligned} T_{TOE} &= 1024 * \frac{MTU}{MSS} * \alpha && MTU < MSS \\ &= \frac{1024}{1.09 + 0.1 \Big/ \left(\frac{MTU}{MSS}\right)} && MTU \geq MSS \end{aligned} \tag{3.58}$$

$B_{FSB}{}^{TOE}$ = 21 [Gbps]，$B_{GE}$ =1[Gbps]，$B_{PCI}$ =2.5[Gbps]，$F_{TOE}$ =32[Gbps]，$MTU$=500×$m$[Byte]，LMbenchによる実測から，$t_{L\_TOE}$=0.4[μsec]，$205/B_{FSB}/MTU = 0.03/m,\ RTT \approx 0.12[m\,\mathrm{sec}]$，$1/B_{GE} + 1/B_{FSB} + 1/(8 * F_{TOE}) + 0.12/B_{PCI} = 1.09,\ t_{L\_TOE}/MTU \approx 0.1/m$．両式(3.57)と(3.58)で表される理論上限値は，上限式(3.48)，式(3.49)による$d$実測値から求められるスループット上限値と，$MTU$の比較的大きい領域では，かなり良く一致し（図 3.14，図 3.15 参照），TCPスループット上限値は，受信端末パケット通過時間$d$と強い相関があることが分かる．

さらに，TOE機能をオフにした場合とNon_TOE-NICの場合のスループット上限値を比較評価するため，TOE-NIC搭載端末IIにおいて，TOE機能をOSによりオフとした場合の理論上限値を求めた．TOEをオフにした場合の理論上限値式(3.59)，パケット通過時間実測式(3.50)による上限値，Iperfによる上限値を，$MTU$をパラメータにして図 3.16 に破線で示す．Non_TOE-NICの場合のスループット上限値は，$MTU$をパラメータにして図 3.16 に実線で示す．この場合は， PCI バス上の転送がパケット単位となり，MTUがページサイズより小さい値のところでは，ページ単位からMSS単位へ，さらにMTU単位へ２重の細分化の影響を受け， 1 ページ 4K[Byte]内にMSSが 2 個しか存在しないため，理論式(3.45)から得られる $MTU$関数式は，式(3.59)となる．$MTU$が 2K[Byte]以上の領域では，上限値は$B_{FSB}$と$B_{PCI}$の値に従い，2K[Byte]以下では，$MTU$サイズに比例することが分かる．

$$T_{TOE\_off} = \begin{cases} 1024 * \dfrac{MTU}{2 * MSS} * \alpha & MTU < 2048\ [Byte] \\ \dfrac{1024}{1.09 + 0.1 \Big/ \left( \dfrac{MTU}{MSS} \right)} & MTU \geq 2048\ [Byte] \end{cases} \tag{3.59}$$

また，スループット上限式(3.58)と式(3.59)の PCI バス帯域の依存性について，受信端末 I(PCI)に Broadcom5701-NIC を搭載した場合と，受信端末 II(PCI-Express)に Broadcom5701-NIC を搭載した場合の両スループットを比較検討した．スループット値は， PCI 帯域をパラメータにし，Iperf を使って測定した．図 3.17 にスループット上限値の PCI バス帯域依存性を示す．スループット上限値は，端末の FSB 帯域と PCI バス帯域とで決まり，TOE-NIC 内部の NP の処理速度が，１GIPS（=32[Gbps]）以上であれば，ほとんどスループットには影響を与えない．端末内部の FSB 帯域は十分大きいため，遅延時間への影響は少なく，スループット上限値は PCI バス帯域に強く依存することが分かる．

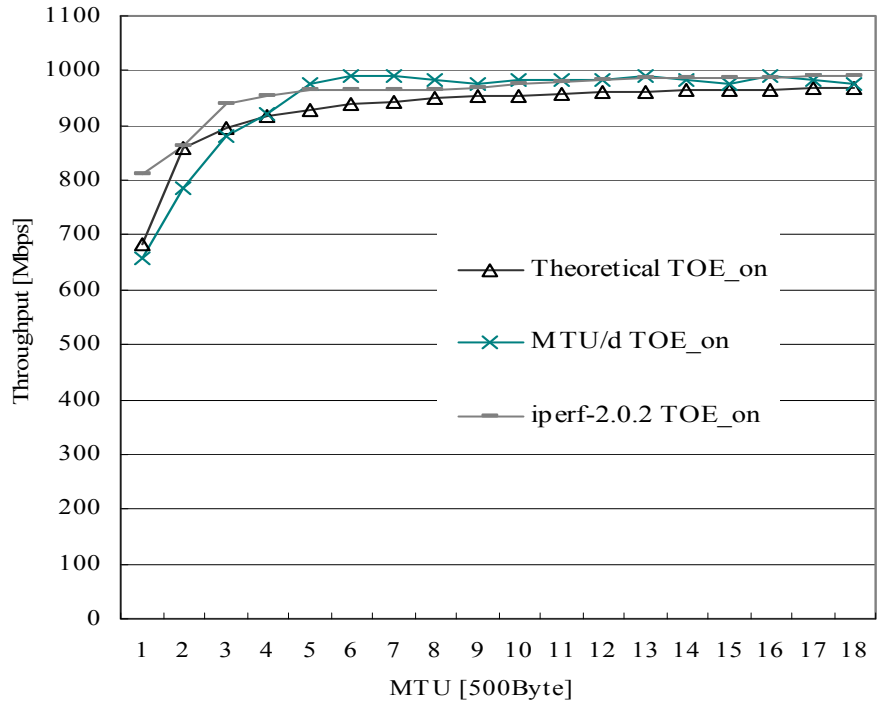



図 3.15 TOE-TCP スループット上限値

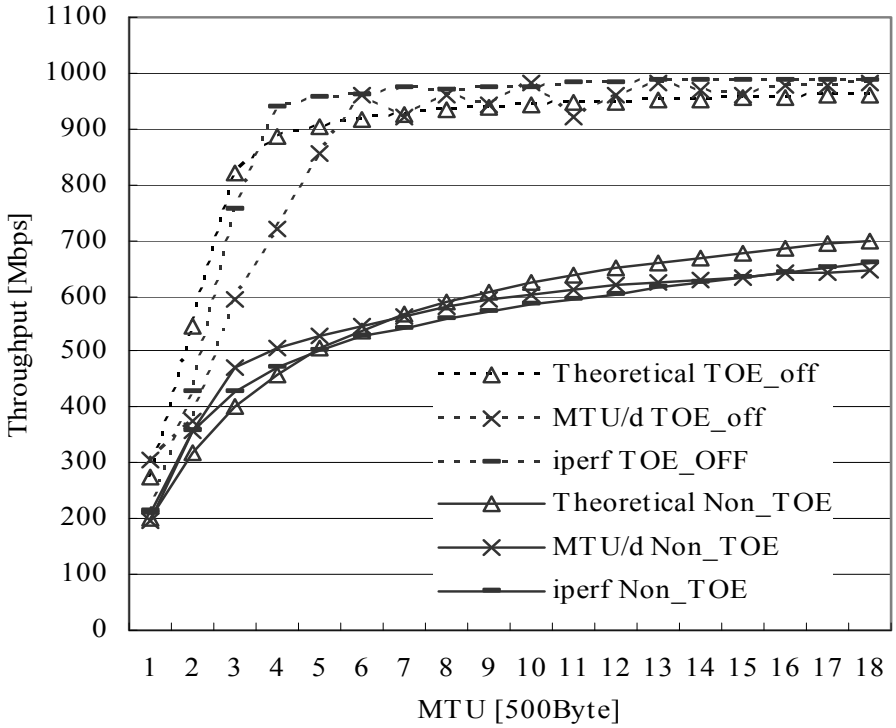



図 3.16 TOE オフ対 Non_TOE スループット上限値

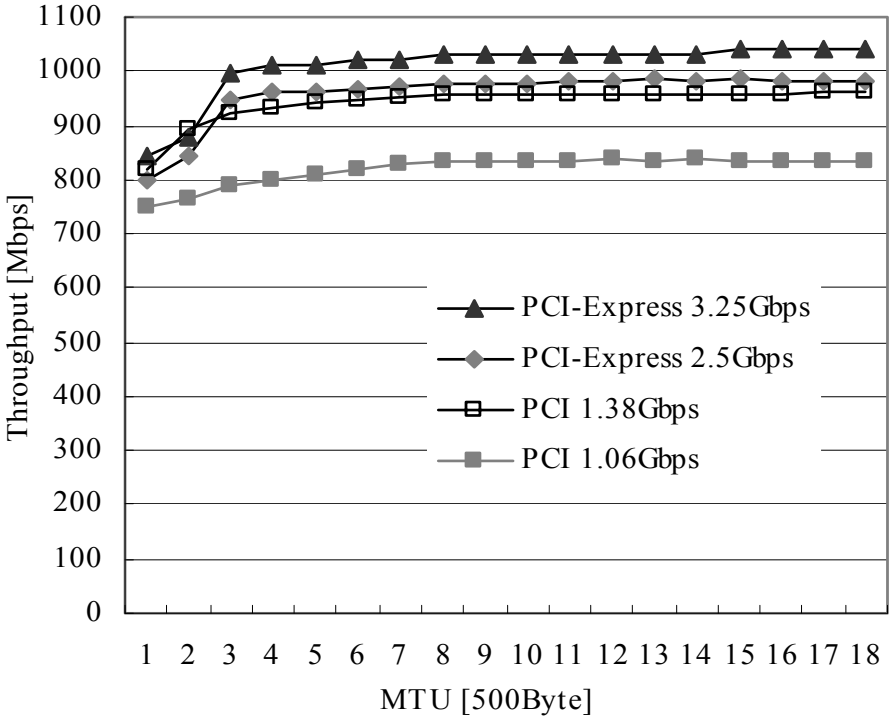



図 3.17　スループット上限値の PCI 帯域依存性

## 3.6　ネットワーク利用可能帯域推定への応用

利用可能帯域推定は，現在まで多くの研究がなされてきているが[14]，本節では，受信端末パケット通過時間$d$を用いた利用可能帯域の推定法を示す．式(3.49)において，$B_{FSB}$は節 3.2 で示したようにループバック試験により，また右辺の$d$は実測で求められるから，$MTU$をパラメータにして，ネットワーク帯域$B_{GE}$ を推定することができる．実測値$d$の時刻値を得るシステムコール関数による遅延に起因する測定誤差を$e>0$ とすると，真値は$d - e$であるから，式(3.60)が得られる．

$$\frac{MTU}{d-e}=\frac{1}{\frac{1}{B_{GE}}+\frac{1.25}{B_{FSB}}+\frac{1}{4*B_{PCI}}+\frac{t_{Ptr}}{MTU}+2.5\times10^{-11}}$$
$$\therefore B_{GE}=\frac{1}{\frac{d}{MTU}-\frac{1.25}{B_{FSB}}-\frac{1}{4*B_{PCI}}-\frac{t_{Ptr}+e}{MTU}-2.5\times10^{-11}} \quad (3.60)$$

式(3.60)からネットワーク帯域推定に必要な*MTU*サイズ（1500×*x*[Byte]）を求めると，以下式(3.61)のようになる．$B_{GE}$推定値を*MTU*の関数としてプロットしたグラフを図 3.18 に示す．

$$\frac{\Delta B_{GE}}{B_{GE}} = \frac{B_{GE}}{MTU} * \Delta e = \frac{B_{GE}[Gbps]}{12x[Kbps]} * \Delta e[\mu \sec]$$

$$\Delta e \le 2[\mu \sec], \quad \frac{\Delta B_{GE}}{B_{GE}} \le 3\%, \ \therefore \ x \ge 6 \tag{3.61}$$

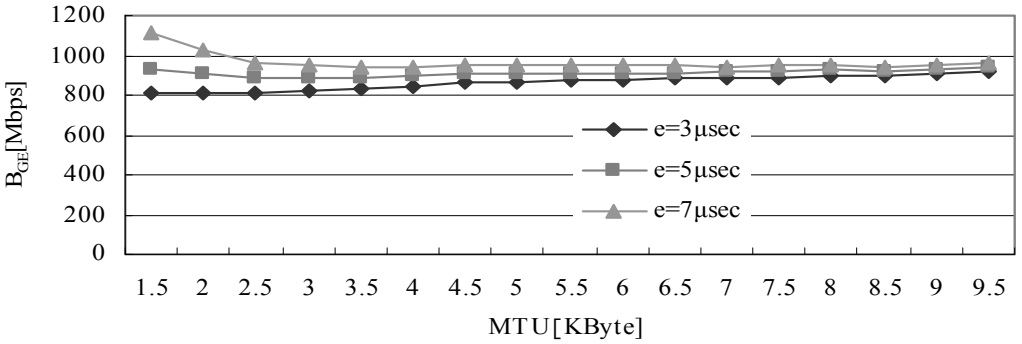



図 3.18 ネットワーク帯域推定

式(3.61)および図 3.18 から，*MTU*サイズが約 9K[Byte]（*x*=6）のジャンボパケットを用いれば，クロストラフィックのない，*RTT*が小さいLAN環境でのネットワーク帯域$B_{GE}$は，約 950Mbps であることが推定できる．

なお，*d*は実測値であるため，イーサネットにおけるパケットヘッダとパケット間隙が有効データ長の減少とパケット通過時間の減少を招くオーバヘッドとして影響する．これは*MTU*が小さいほど顕著になるため，*MTU*<2.5*K*の領域では，ずれが大きくなる傾向にある．

## 3.7 結言

広範囲の遅延に対し成立する TCP スループットの正確な上限値を，ネットワーク・端末総合系として捉え端末処理速度を考慮し解析的に求めた．受信端末パケット通過時間を測定することにより求めた実測値を用いて検証を行い，理論値式(3.44)と式(3.45)は，実測*d*値による式(3.50)の値とよく一致する結果を得た．従って，理論値式(3.44)と式(3.45)は，TCP スループット低下改善の指針を与えるものであると言える．以下の項目が，結論できる．

(1) スループット上限値の *MTU* サイズ依存性を明確にした．なお，仮想的に *MTU* を無限大と考えると，上限値は端末ハードウェアバス帯域およびネットワーク帯域の調和合計の逆数に収束する．

(2) 受信端末でのパケット通過時間 $d$ 測定においては，10 パケット程度の個数で通過時間が求められ，本測定により，実時間 1 *RTT* 内に帯域推定が可能であることが分かる．

(3) *MTU* サイズ 9K バイトのパケットを用いて，受信端末でパケット遅延 $d$ を測定すれば，ネットワーク帯域を推定することができる．

(4) 従来の TCP スループットモデルに対しても上限値として成立する．

今後の課題として，クロストラフィックが存在する環境や，パケット到着間隔の変動がある場合を考慮した提案手法の拡張が必要である．

# 第 4 章

# スループット上限値を実現する TCP 制御方式

## 4.1 緒言

LAN 環境からグローバルな有線通信，衛星通信，宇宙通信，移動体通信に至る，種々の環境下において信頼性の高いデータ通信を行う際には TCP が用いられることが一般的である．このとき，エンド端末システム間のスループットは *RTT* に大きく依存し，高遅延系では実帯域と比較して著しく低い実効帯域しか利用できないという問題がある．

ネットワークの高速・広帯域化が急速に進む一方で，端末機器としての PC については，機能面の充実は目覚しいが，高速化の点ではネットワークに比べて比較的遅く，その処理速度はネットワークの実効転送速度と比較して同程度あるいは逆に遅い傾向にある．実際のネットワークは，広帯域ではあるが通信距離に比例する遅延特性を有し，種々の要因から輻輳とパケットロスが発生するため，大容量な非リアルタイム系通信では，状況に応じた輻輳制御と再送制御を提供する TCP が広く用いられている．このとき，エンドツーエンド(E2E)システム間のスループットは TCP の挙動に大きく依存する結果になる．これまでに，高いスループットを得る TCP 制御方式のアルゴリズムが数多く研究されて来た．また，効率のよいアルゴリズムを用いた場合に，スループットの上限値を求める研究も行われてきた．

エンドツーエンドの TCP スループットの理論的考察を行い，前第 3 章で得られたスループット上限値に限りなく近い特性を示す TCP 制御方式を求めた．端末内部構造を考慮に入れたスループットの上限値求め，TCP スループット低下の改善指標を示し，TCP 輻輳制御方式への指標を与える．エンドツーエンド端末とその間のネットワークを一連のフィードバック制御系と見なし，受信端末をセンサーとし，ネットワーク特性を測定し，TCP スループットが上限値に極めて近い制御方式を求めた．

TCP の制御は，ネットワークの輻輳制御と輻輳起因のパケットロス補償である再送制御が主目的となる．送信端末側からのネットワークへ送出するパケット量の制御が重要な役目を果たすため，ネットワークの遅延特性とネットワーク帯域（通信路容量）特性を知る必要がある．

遅延特性は,従来から周回遅延時間として送信端末側で測定され TCP 制御に利用されている．しかし，ネットワーク帯域特性は，現在まで，実時間内に測定する有効な方法がなく，TCP 制御には反映されていない．

本章では，TCP 通信系を端末ネットワーク総合系として捉え，ネットワーク帯域特性を受信端末内パケット通過時間から求め，この値を用いて，従来から示されている評価値より正確な TCP スループット上限値を求め，多くの TCP バージョンで採用されているソケットバッファ方式における最適な TCP ウィンドウ更新アルゴリズムを提案する．

従来のモデルでは，TCP ウィンドウサイズとネットワーク帯域，遅延量を基にスループットを算出していた．しかし，端末 PC の処理速度がネットワーク帯域と同程度あるいはそれ以上になると，TCP スループットは，端末 PC ハードウェア構成，転送メカニズム，MTU（Maximum Transfer Unit）などで規定される上限値に制約されることになる． 受信端末から得られる情報は，確認応答(ACK:Acknowledge)パケットに載せられて送信端末にフィードバックされる．TCP スループットを決定するネットワーク端末総合系のパラメータであるネットワーク帯域と遅延量を送信端末が検知することにより，帯域遅延積に等しくなるようなフローレベルの TCP ウィンドウ制御アルゴリズムが求められる．また，受信端末でのパケット通過時間の分布をリアルタイムに求めることにより，最大帯域と利用可能帯域が分かり，ネットワークを共有する端末間の公平性を最大にするウィンドウ更新方法が分かる．

本章では，ネットワーク端末総合系スループットモデルについて述べ，受信端末パケット通過時間から推定できるパラメータを基にした TCP ウィンドウ制御アルゴリズムを提案する．さらに，本方式と従来方式との比較シミュレーションについて述べる．

## 4.2 端末レイヤーモデルにおける各層のバッファサイズ

### 4.2.1 IP 層内送信バッファ最適サイズ

プロトコルモデルを実現するためのハードウェアでは，各層内の処理を独立させるため，各層間にバッファが設けられる．TCP スループットはアプリケーションからソケットバッファを通して見たときの，下位層へのパケット投入量対所要時間（遅延時間と処理時間の和）比で表され，式(4.1)のように定義される．ただし，$t$ はソケットバッファ内における時間である．図 4.1 に端末のバッファ構成を示す．

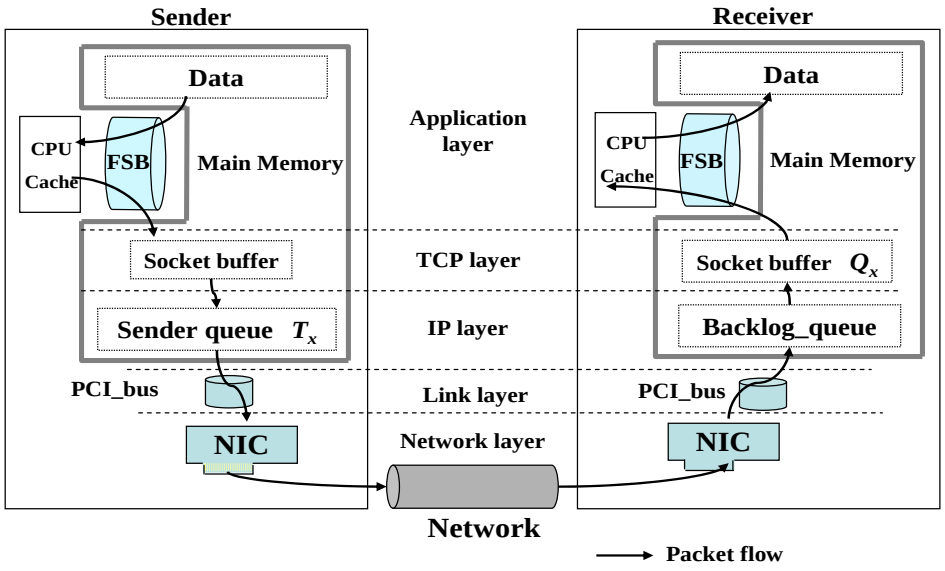



図 4.1 端末バッファ構成

$$Throughput = \frac{\frac{1}{L}\int_{buffer}(Packets)dt}{\frac{1}{L}\int_{buffer}(Latency)dt} \tag{4.1}$$

IP 層バッファは，ネットワーク層とより高速の上位層のパケット送出レートマッチングの目的でそのサイズが決められ，バッファ制御が行われる[27]，[28]，[29]，[36]．端末バッファ内とネットワーク内のパケットのタイミングを図 4.2 に示す．送信端末とネットワークの間には，

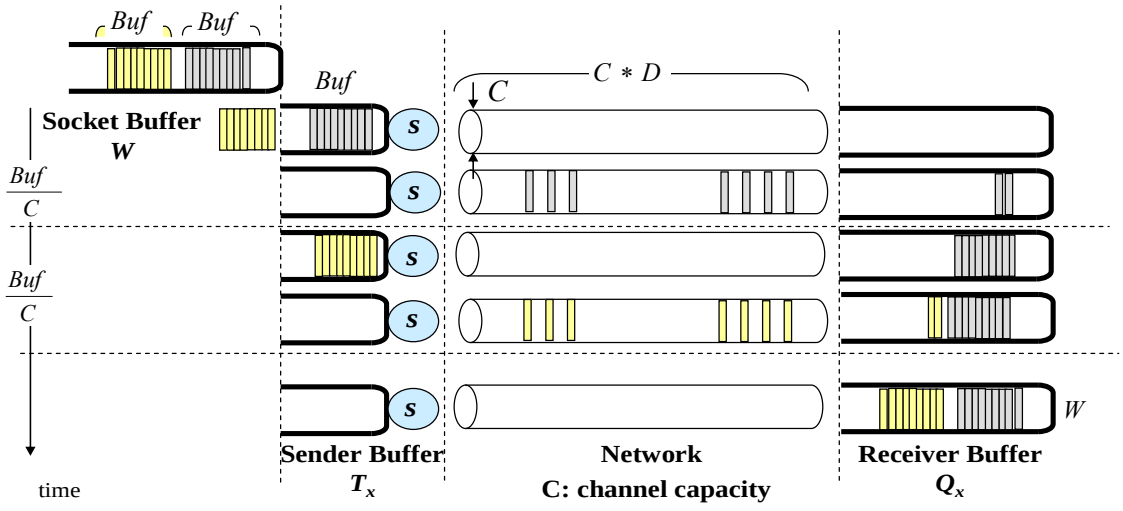



図 4.2　端末各層内バッファとパケットタイミング

レートマッチングのために，送信バッファが IP 層にもうけられる．図 4.3 に示される IP 層バッファサイズ *Buf*，チャンネル容量 *C*，片側遅延 *D=RTT/2*，TCP 層ウィンドウサイズを *W* で表し，送信バッファの送信レート *s* とすると式(4.2)が成り立つ．

$$C = s \tag{4.2}$$

図 4.2 において，1/2**RTT* = *D*，$B_{GE}$ = *C*，受信ソケットバッファ$Q_x$=*W*,送信バッファ送出レートはネットワークインターフェースカードの送出レート*s*で規定され，*s*=*C*である．

送出バッファ$T_x$（サイズ*Buf*[Byte]）とネットワーク 2 層に含まれるパケット対遅延時間からスループットを求め，スループットが最大になる*Buf*を求める．*Buf*が十分でない場合と大きすぎる場合を考え，送信バッファとネットワーク間のインターフェースでのスループットに注目し，送出バッファサイズ$T_x$とネットワーク両系内のパケット数[Byte]とパラメータ *t*（ネットワーク経路）を使って，スループットを求める[43]．パケットはネットワークを通過するのに一定の時間必要であるから，スループットを与える式(4.1)分母の遅延は，式(4.3)で表される．

$$1/L\int(Latency\ )dt = \frac{Buf}{C} + D \tag{4.3}$$

最初に，送信バッファサイズが小さい場合のスループットが最大になる最適バッファサイズ$T_x$を算出する．$T_x$単位でバッファリングすれば，ネットワーク中を間断なくパケットの送出が出来る．ネットワーク層の帯域遅延積*C*×*D*と*Buf*を用いて，スループットは，式(4.4)のように表される．図 4.3 にバッファが小さい場合の構成図を示す．図 4.4 にパケット数と遅延の関係を示す．

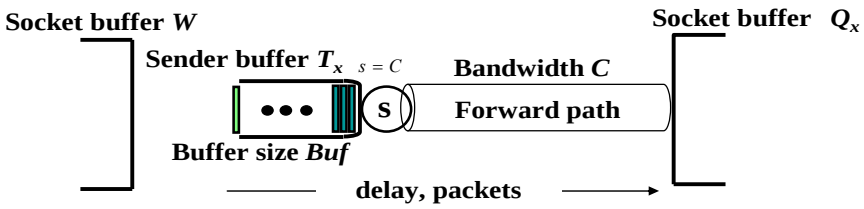



図 4.3　ネットワーク帯域遅延積より送信バッファ小の場合

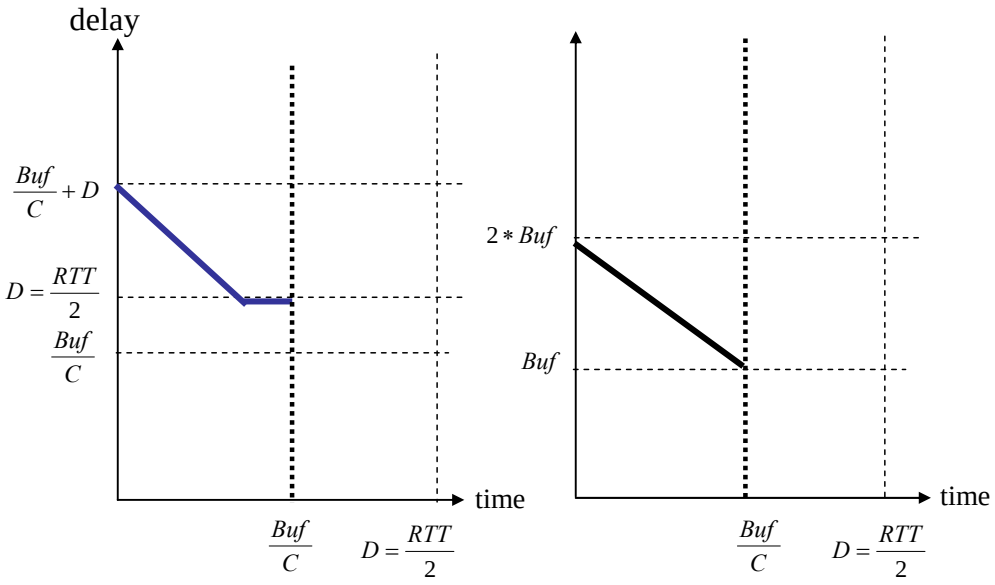



図 4.4　アンダーバッファのパケット遅延とパケット数

$$Throughput=\frac{\frac{C}{Buf}\int_0^{\frac{Buf}{C}}(Packets)dt}{\frac{C}{Buf}\int_0^{\frac{Buf}{C}}(Latency)dt}=C*\frac{3}{3+\left(1-\frac{C*D}{Buf}\right)^2} \tag{4.4}$$

次に，送出バッファサイズが大きすぎる場合のバッファ構成を図 4.5 に示す．また，送出バッファサイズ *Buf* とフォワードパスの帯域遅延積 *C*×*D* の関係を図 4.6 に示す．スループット劣化度を考察する．この場合は，パケットの空隙はないが，余分なパケットを処理するのに遅延が発生するため，小さすぎる場合と同様にスループットは減少する．

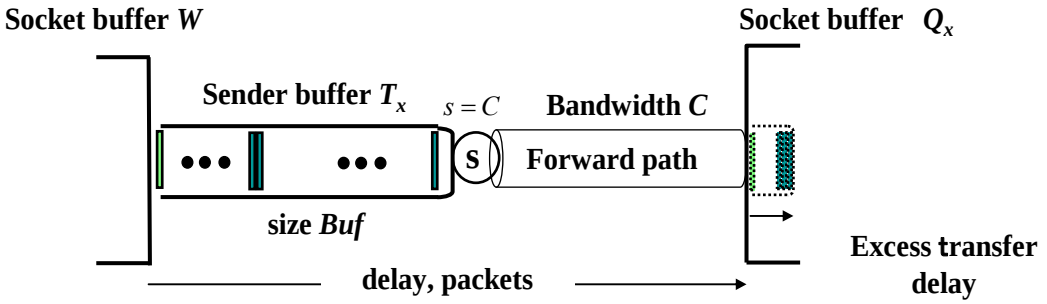



図 4.5　送出バッファサイズがネットワーク帯域遅延積より大きい場合

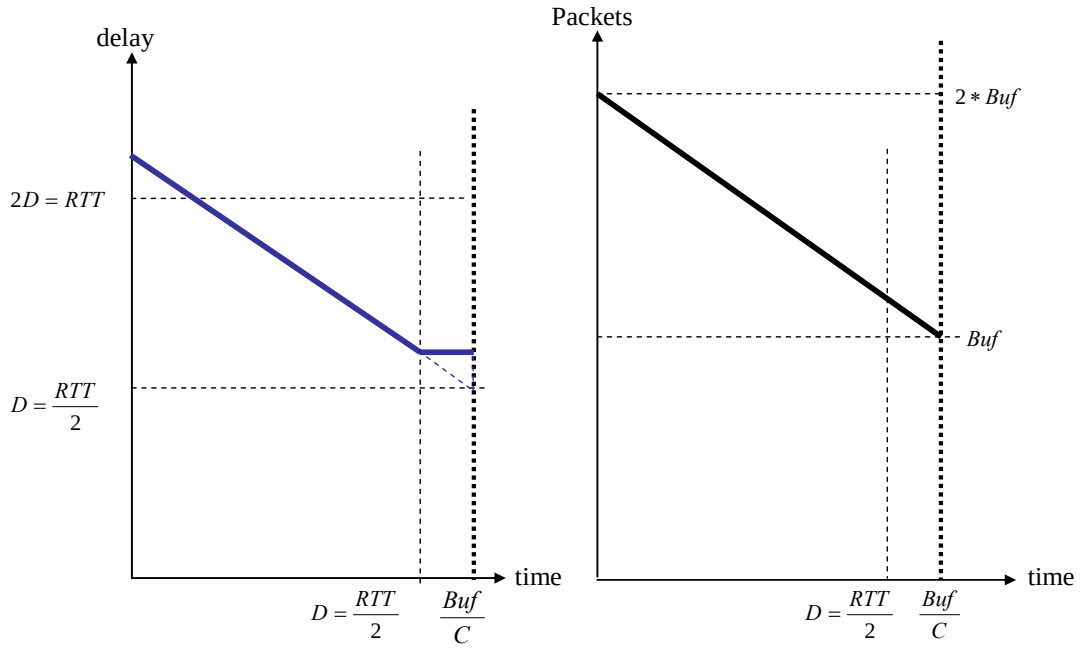



図 4.6　オーバーバッファのパケット遅延とパケット数

バッファサイズが大きくなり，帯域遅延積より大きくなると，スループットは式(4.5)となる．

$$Throughput = \frac{1+\frac{\left(1-\frac{C*D}{Buf}\right)^2}{4\left(\frac{C*D}{Buf}\right)-\left(\frac{C*D}{Buf}\right)^2}}{1+\frac{2*\left(1-\frac{C*D}{Buf}\right)^2}{4\left(\frac{C*D}{Buf}\right)-\left(\frac{C*D}{Buf}\right)^2}} \tag{4.5}$$

式(4.5)から，小さい場合と同様，バッファサイズ *Buf* が帯域遅延積 *CD* に等しいとき，すなわち式(4.6)で示す条件のとき，最大となる．

$$Buf = C*D = T_x \tag{4.6}$$

### 4.2.2 受信端末ソケットバッファ最適サイズ

ネットワークと受信端末インターフェースにおいて，高速の受信端末とネットワークのスループットをマッチングさせるため，TCP層にソケットバッファが設けられる．受信端末は，一般的にネットワークより高速であるため，IP層に設けられるバッファは上位層のTCP層内ソケットバッファで代用される．受信端末ソケットバッファ$Q_x$に対するスループットを求めるには，流入対象パケットがネットワークフォワードパス内インフライトパケットと送出バッファ内パケットの総和と受信端末ソケットバッファにいたるまでの遅延を評価する必要がある．

図 4.7 に受信端末ソケットバッファと送信端末送信バッファの構成を図示する．受信端末ソケットバッファサイズ$Q_x$の最適値（ネットワークから受信端末へのスループット最大）は，送出バッファサイズがネットワーク片側帯域遅延積より小さいアンダーバッファの場合とネットワーク片側帯域遅延積より大きいオーバーバッファの場合に分け，図 4.7 に示す送出バッファ

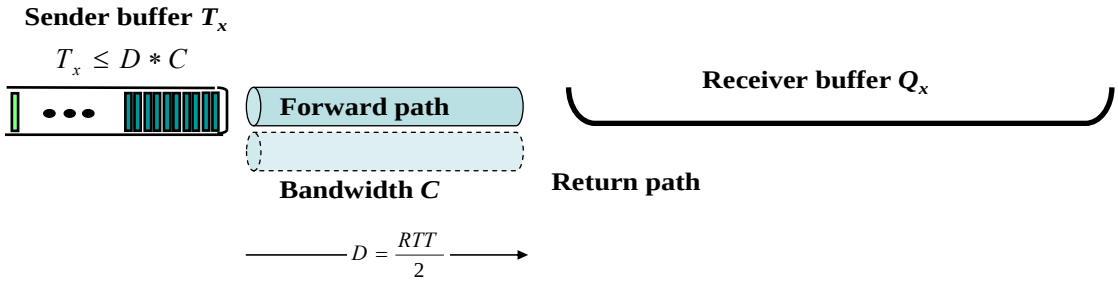



図 4.7　受信バッファサイズとパケット数

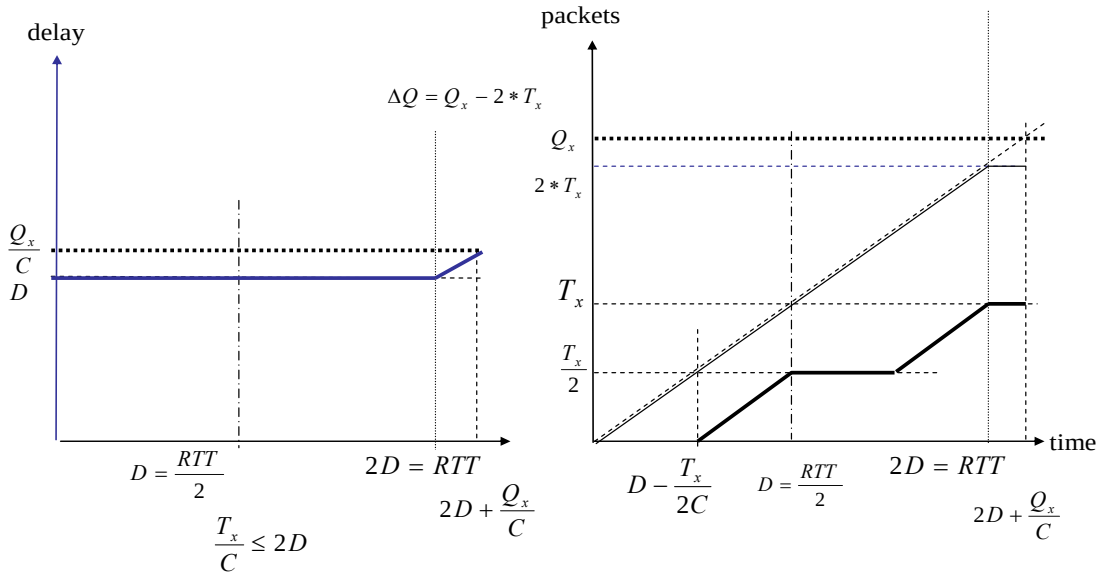



図 4.8　受信バッファ遅延とパケット数の関係

から，受信端末ソケットバッファにいたるまでの経路上の遅延とパケット数を求めることにより，算出する．受信ソケットバッファ内のパケット数と遅延時間をグラフにしたものを図 4.8 に示す．グラフからスループットを求めると式(4.7)となる．

$$Throughput = \frac{\frac{1}{2*D}\int_0^{2*D}(Packets)dt}{\frac{1}{2*D}\int_0^{2*D}(Latency)dt} = \frac{C*\left(\left(\frac{T_x}{C*D}\right)+\left(\frac{T_x}{C*D}\right)^2\right)+\frac{\Delta Q}{C*D}}{2+\frac{\Delta Q}{C*D}+\left(\frac{\Delta Q}{C*D}\right)^2 \Big/ 2} \tag{4.7}$$

式(4.7)において，ネットワークから受信バッファ$Q_x$へのスループットは，
$T_x \to C*D, \Delta Q \to 0, Q_x = 2*T_x$のとき最大値をとるので，式(4.8)のとき最大となる[28].

$$\Delta Q = 0,\ Q_x = 2*T_x \tag{4.8}$$

ただし，$Buf$=$T_x$.

したがって，ネットワークと受信端末内受信バッファの必要サイズは，送信バッファ内パケットとネットワーク中インフライトパケットの総和に等しい量である．最初にバッファされるパケットは，フォワードパス内のインフライトパケットに相当し，次のネットワーク遅延時間後にバッファされる量は，送信バッファ内のパケットに相当する．インフライトパケットはリターンパス内のパケットに相当する．このように受信端末受信バッファはネットワークのダブルパス遅延時間を等化する．ネットワークから受信端末へのバッファリングは，パケット欠落が発生しないように，IP 層にバックログキューを設置する．

## 4.3 TCP ウィンドウ制御アルゴリズム

受信端末内でのパケット通過時間を用いて送信端末側で帯域$C$を求める[28]．TCPコネクション確立フェーズ後の輻輳制御ウィンドウ初期値を$P_{init}$=10 とし，受信システムコールrecv( )の実行時間の分布を求める[23]．パケットはバースト的に送受を繰り返すため，最初のセッションで, 10 個のパケットを送出し図 4.9 に示す平均値$\delta_j$を算出し, 利用可能帯域$B$を求める[30], [31], [32]，[33]．初期セッションの時間関係を図 4.10 に示す．レイヤーモデルでは，送信端末

の送信バッファに充填するパケット数を，ネットワーク層の帯域遅延積に等しくする必要があり，ネットワーク帯域と通信路長相当の最小遅延時間$RTT_{min}$を知る必要がある．ネットワーク通信路がTCP通信を開始した初期は，ネットワークには十分空き時間が存在し，十分小さい一定サイズのパケット塊を送出し，塊の単位時間当たりの広がりを示す帯域は，パケット単位でもフローレベル（ビット単位）でも同じと考え，1 個のパケットより大きいがそれほど大きくない単位として 10 個のパケットを選定する[32]．次のセッションで，$RTT$の最小値$RTT_{min}$を求め，$P_{init}$の受信端末通過$\Delta_{init}$を求める．

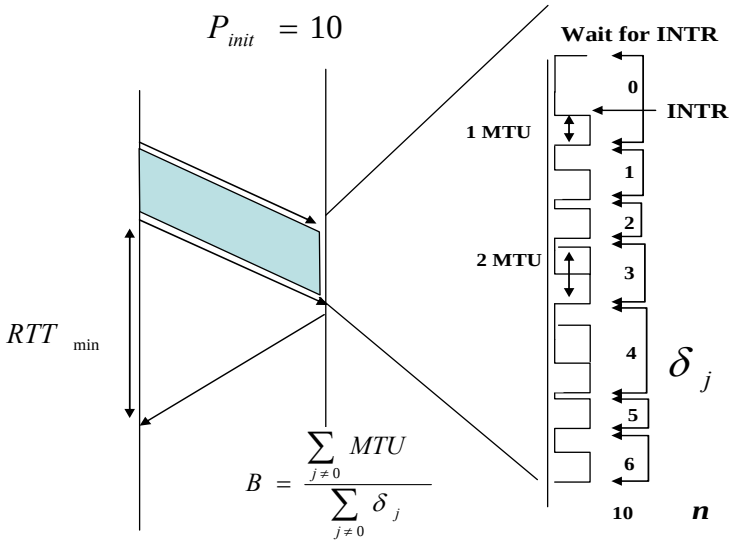



図 4.9 受信端末による帯域測定

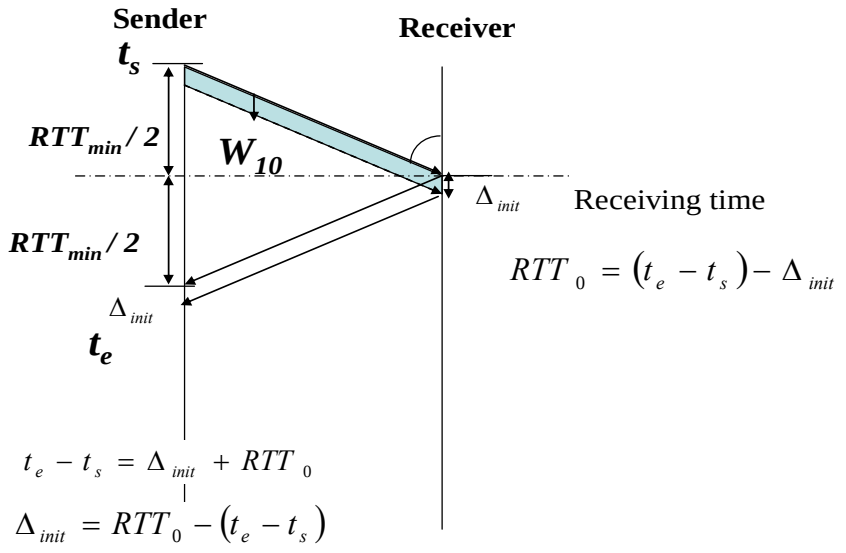



図 4.10　初期セッションのタイミングチャート

ここで$t_s$は，送信端末でのパケット送信開始時刻，$t_e$は送信端末でACKを受信した時刻である．第２番目のセッションでは，同じパケット数 10 のパケット塊を送出し，$RTT_{min}$を算出する[24]．初期セッションは，ARPが作用し，遅延時間が大きくなることが多いため，第２セッション以降の中の最小値を，ネットワーク通信路長相当を表す量として$RTT_{min}$とする・第２セッション以降の時間関係を図 4.11 に示す．

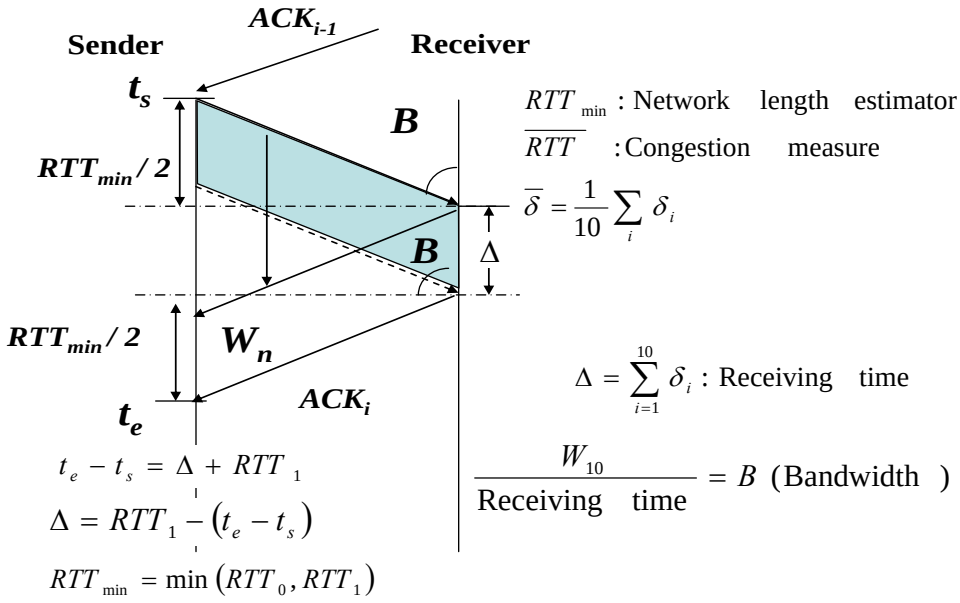



図 4.11　第 2 セッション以降のタイミングチャート

さらに，続くセッションでは，図 4.12 に示すような$\overline{RTT}$を計算し，推定された帯域 $B$ を用いて式(4.9)の条件の確認をする．

$$\begin{aligned} \frac{W}{B} &\le RTT_{\min} : Acquisition \\ &> RTT_{\min} : Tracking \end{aligned} \tag{4.9}$$

アクイジッションモードであれば，更に $W$ を線形に増加させる．トラッキングモードであれば，図 4.13 に示すように，更に$\overline{RTT_{cong}}$を計算し，表に示す除算で減少させる．到達可能な最大スループットを与える $W$ は式(4.10)のようになる．

$$W = B * RTT_{\min} = \frac{P_{init}}{d} * RTT_{\min} \tag{4.10}$$

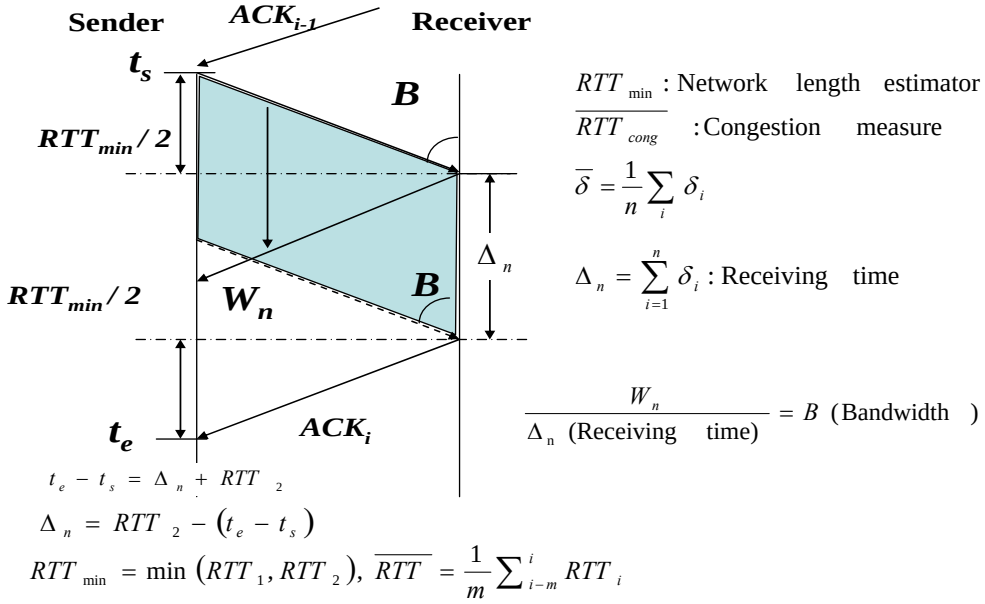



図 4.12　一般的なアクイジッション時の *RTT* の計算

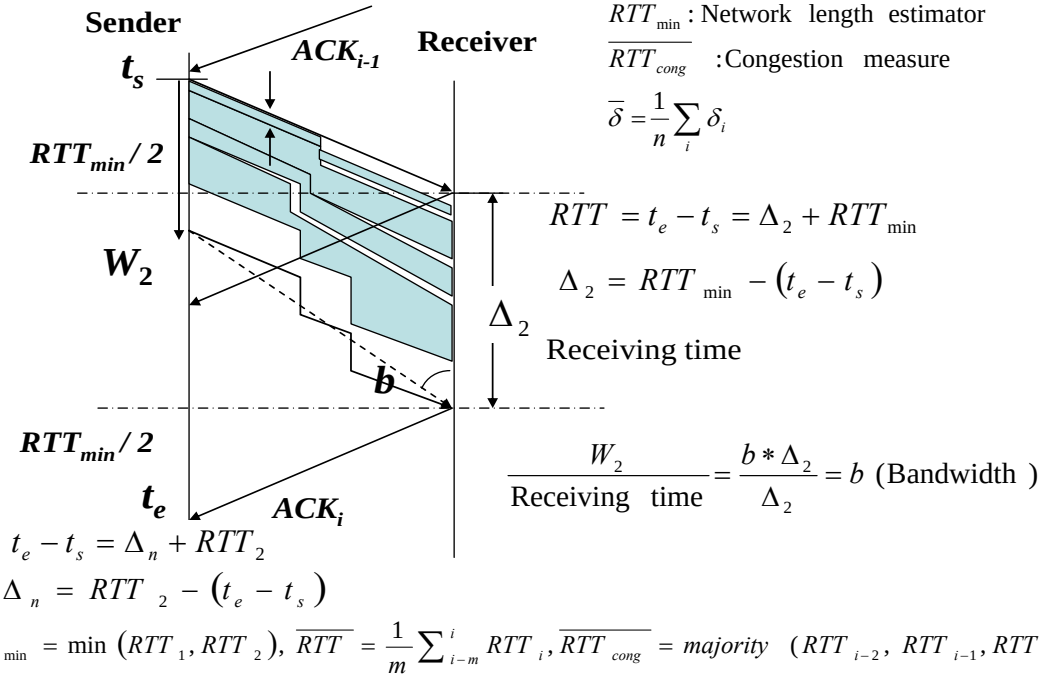



図 4.13　輻輳時の *RTT* 計算

*d* を受信端末でリアルタイム計測し各ラウンド終了時に送信端末へフィードバックし，初期時は，式(4.11)に従って，送信端末の TCP ウィンドウ（輻輳制御ウィンドウ）推定値を送信端末で求める．

$$\frac{\Delta W}{unit\,time} = W - \frac{P_{init}}{d} * RTT_{min} < 0 \tag{4.11}$$

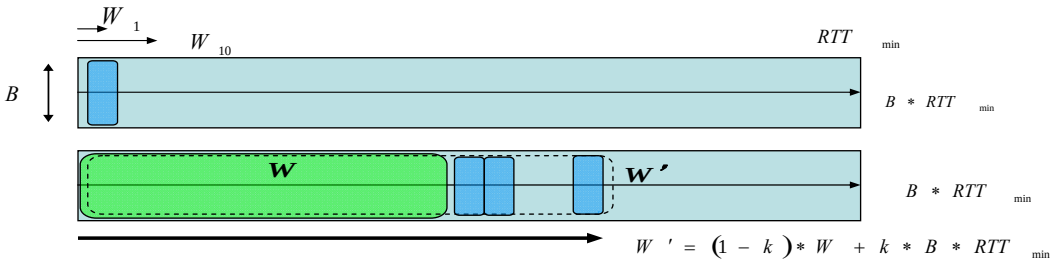



図 4.14　非輻輳時ウィンドウ更新

受信端末からのフィードバック値$d$，計測に必要なラウンド最初のパケット数$P_{init}$，現在のウィンドウサイズ$W$から，次のステップまでの差分値を⊿$W$，時間を$t$とすると，図 4.14 に示す空き帯域遅延積を埋めるように式(4.12)を用いてウィンドウを増加させる. 非輻輳時はネットワーク中に空隙が存在するため，この空隙をパケット加算で充満させる様にウィンドウを増加させる．ネットワーク帯域をパケット（$P_{init}=W_{10}$）通過時間から推定し，空隙に充填できる最適パケット数を求める.

$$\begin{aligned} W' &= W + \Delta W = (1-k) * W + k * B * RTT_{\min} \\ \frac{\Delta W}{W} &= k * \left( \frac{B * RTT_{\min}}{W} - 1 \right) \end{aligned} \tag{4.12}$$

$$B * RTT_{\min} - W = e^{-k * \frac{t}{RTT}},\ 0 < k < 1 \tag{4.13}$$

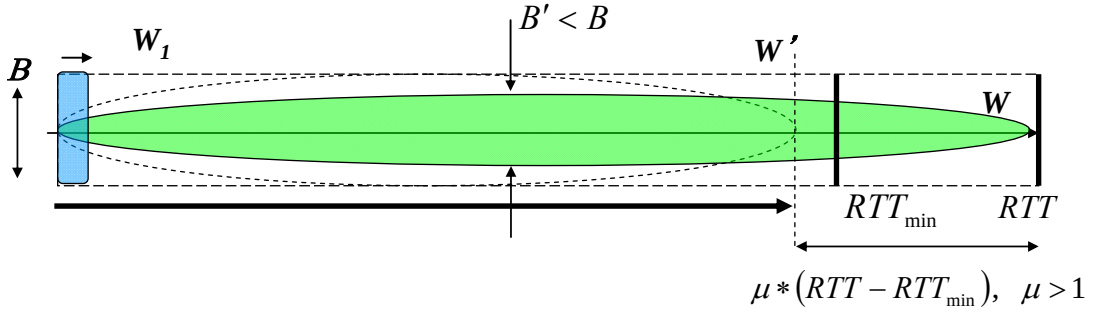



図 4.15　輻輳時ウィンドウ更新

輻輳時は，輻輳状態を*RTT*の超過時間から識別する．図 4.15 に示すように，ネットワーク中にはパケットが充満し空隙はない．輻輳ウィンドウサイズが過大になっていることを意味し，減少させる必要がある．この場合は受信端末の受信バッファ$Q_x$とネットワーク層とソケット層の間に設けられたバックログキューを使ってネットワークから溢れたパケットを吸収する．

パケットは有効にバッファされていると考え，ネットワーク通信路長（時間換算）が伸びたと見なし，通信路長を*RTT*から$RTT_{min}$まで縮めることにより輻輳ウィンドウを減少させる．減少手順は，除算による．輻輳時は，非輻輳時と違い空隙があるときのよう明確な帯域は定義されないため，*RTT*から決まる式(4.14)で表される帯域*B'*を等価帯域とすることにより非輻輳時と同じ量が定義できる．

$$B' = \frac{W * MTU}{RTT} \tag{4.14}$$

また，輻輳を回避する目的から，$RTT_{min}$相当の帯域遅延積より小さいウィンドウサイズまで縮小させるため，式(4.15)で表される定数を導入する．

$$1 < \mu \tag{4.15}$$

を使って，以下の量を計算する．この乗数$\mu$はトラフィック数に関係した量と考えられ，*n*本のトラフィックがある場合は，*n* とすることにより速く公平性を最大にする点に収束させることができる．輻輳ウィンドウサイズ*W*は，式（4.16）の関係から式(4.17)が求められ，式(4.18)となる．

$$\begin{aligned} &W \propto B' * RTT,\ W' \propto B' * \left(RTT - RTT_{\min}\right) = B * \mu * \left(RTT - RTT_{\min}\right) \\ &\mu = \frac{B}{B'},\ B = B_{GE},\ B' = \frac{W * MTU}{RTT} \end{aligned} \tag{4.16}$$

$$\begin{aligned} &\frac{W'}{W} = 1 + \frac{\Delta W}{W} = \frac{RTT_{\min}}{RTT_{\min} + \mu * \frac{\left(W - B' * RTT_{\min}\right)}{B'}} \\ &\frac{\Delta W}{W} = \frac{B' * RTT_{\min}}{\mu * W + \left(1 - \mu\right) * B' * RTT_{\min}} - 1 = -\frac{RTT - RTT_{\min}}{RTT - \left(1 - \mu'\right) * RTT_{\min}} \\ &\left(\frac{1 - \mu'}{W} + \frac{\mu'}{W - B' * RTT_{\min}}\right) * \Delta W = -1 \end{aligned} \tag{4.17}$$

$$\begin{aligned}&(1-\mu')\ln W+\mu'\ln(W-B'*RTT_{\min})=-\frac{t}{RTT}\\&\therefore W^{1-\mu'}*(W-B'*RTT_{\min})^{\mu'}=e^{-\frac{t}{RTT}},\ \mu'<1,\ \mu'=\frac{1}{\mu}\end{aligned}\tag{4.18}$$

輻輳時は，$W$ が十分大きくなって帯域遅延積を超えると受信端末のバックログキュー，あるいはネットワーク内ルータキューの蓄積が急激に増加し始めパケット廃棄が発生する可能性が高くなる．ウィンドウ推定値はこの超過分を減少させるため，式(4.19)のように制御する必要がある．

式(4.13)と式(4.18)において，収束点を評価するために，単一トラフィックの場合を考えると，μを 1 とすることができ，簡単のために，端末内部の帯域は無限大と仮定すると，$B'=B$

$$\begin{aligned}&\frac{dW}{W}=-\left(1-\frac{RTT_{\min}}{RTT}\right)\\&W=B*RTT_{\min}*e^{-\left(1-\frac{RTT_{\min}}{RTT}\right)t}\end{aligned}\tag{4.19}$$

輻輳状態の指標値として，$RTT$対$RTT_{min}$比を使用し，空き帯域遅延積を目標値として，ウィンドウは増減を繰り返し収束するように制御される．さらに輻輳が始まった結果，パケットロスが発生すると，TCPウィンドウ制御は，FR/FR（高速回復／高速再送）モードに入るため，ウィンドウを急激に減少させる必要がある．パケットロス率$p$はFR/FRモード時の 1 ラウンド期間のみでの急激なウィンドウ縮小制御の指標値として使用する．このモードに入ると現在ウィンドウを半分にし，輻輳回避および回復を行う．また，ウィンドウの増加減少を式(4.13)，式(4.18)に示すようなAIMD(Additive Increase Multiplicative Decrease)型で行うとネットワーク上に他のTCPストリームが存在する場合の公平性を補償することができる．

TCP ウィンドウサイズ（輻輳制御ウィンドウサイズ）$W$ は，次ステップまでの差分$\Delta W$ の期待値として以下の式(4.20)で表される．増加するウィンドウサイズを$\Delta W_+$，減少させるウィンドウサイズを$\Delta W_-$と表し，輻輳状態にある確率を$\varepsilon$ とすると，式(4.21)が得られる．収束点でのウィンドウサイズ$\overline{W}$ は，式(4.22)を経て，式(4.23)となる．

$$r=\frac{W}{RTT},\ \frac{\Delta r}{\Delta t}=\frac{1}{RTT}*\frac{E[\Delta W]}{\left(\frac{RTT}{W}\right)},\ \ \frac{\Delta W}{\Delta t}=W*\frac{E[\Delta W]}{RTT},\ \ \frac{\Delta W}{W}=\frac{\Delta t}{RTT}*E[\Delta W] \tag{4.20}$$

$$\begin{gathered}E[\Delta W]=\frac{\Delta W_{+}}{W}(1-\varepsilon)-\Delta W_{-}*\varepsilon=\frac{(a-W)}{W}*(1-\varepsilon)-\gamma*W*\varepsilon,\\ a=B*RTT_{\min},\ \ \gamma=1-\frac{RTT_{\min}}{RTT},\ \ \ \varepsilon=erf\left(\frac{\overline{RTT}}{\sigma_{RTT}}\right)\end{gathered} \tag{4.21}$$

$$\frac{\Delta W}{\Delta t}=W*\frac{E[\Delta W]}{RTT}=\frac{1}{RTT^{2}}\left\{(1-\varepsilon)*k(a-W)-\gamma*\varepsilon*W^{2}\right\} \tag{4.22}$$

$$\frac{\Delta W}{\Delta t}=0,\ \ \overline{W}=B*RTT_{\min}\left\{1-O\left[\left(\frac{\varepsilon}{1-\varepsilon}*\frac{4*a*\gamma}{k}\right)^{2}\right]\right\} \tag{4.23}$$

となる．この値は，ネットワーク帯域遅延積より少し小さく，輻輳が始まる少し前の値に収束することを意味する．*RTT* 諸量の関係を図 4.16 に，パケットロス率との関係を図 4.17 に示す．

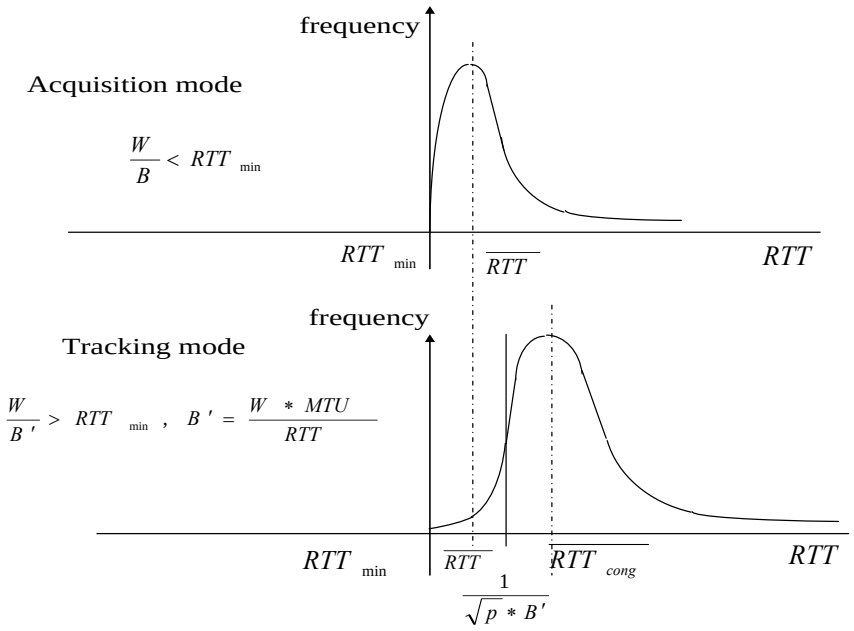



図 4.16　RTT 諸量の関係

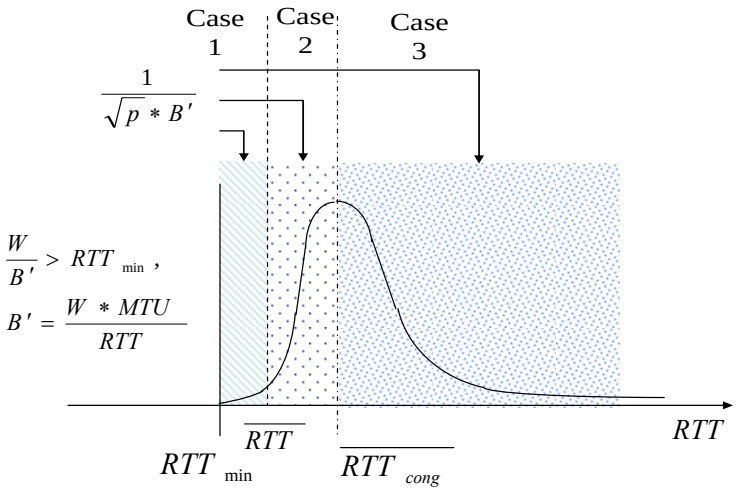



図 4.17　パケットロス率と RTT 量の関係

次に，各TCPセッションでの*RTT*のサンプル値に対し，帯域*B*，*B'*を求め輻輳ウィンドウ*W*を更新するモードとその手順を図 4.18 に示す．さらに，図 4.19 に初期状態Aから途中状態$T_2$，$T_3$を経由し収束状態$T_1$に至る輻輳制御ウィンドウ状態遷移を示す.

| Mode \ Update | | Additive increase ACK | | Multiplicative decrease 3DUP-ACK, RTO |
|---|---|---|---|---|
| Acquisition $\frac{W}{B} < RTT_{\min}$ | A | $\Delta W = k(B * RTT_{\min} - W)$ | | $\frac{1}{2}$ |
| | | $1 - p$ | | Packet loss rate $p$ |
| Tracking | T1 | Case 1 | $+1$ | -1 |
| | | | $1 - p$ | Packet loss rate $p$ |
| $\frac{W}{B'} > RTT_{\min}$ , $B' = \frac{W * MTU}{RTT}$ | T2 | Case 2 | **mode** A | $\Delta W = -W * \frac{\overline{RTT_{cong}} - RTT_{\min}}{\overline{RTT_{cong}} - (1-\mu') * RTT_{\min}}$ |
| | | | $1 - p$ | Packet loss rate $p$ |
| | T3 | Case 3 | **mode** A | $\Delta W = -W * \frac{\overline{RTT_{cong}} - RTT_{\min}}{\overline{RTT_{cong}} - (1-\mu') * RTT_{\min}}$ |
| | | | $\varepsilon = erf\left(\frac{\overline{RTT}}{\sigma_{RTT}}\right)$ | $1 - \varepsilon$ |

図 4.18　輻輳制御モード

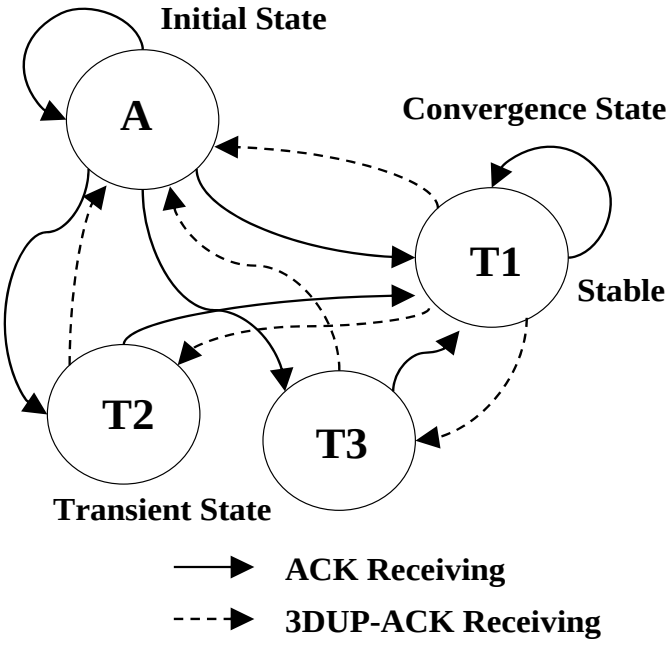



図 4.19　輻輳ウィンドウ状態遷移

## 4.4 公平性

クロストラフィックが存在する場合，ボトルネックとなるリンクの帯域を平等に配分する必要がある．輻輳制御に使われる，AIMD は，同じバージョンの TCP を使うことを前提にして，ネットワーク利用を最大にし，かつ公平性をも最大にすることが分かっている．図 4.20（クロストラフィックは 1 本と仮定）において，ACK が返ってくる場合のウィンドウ増加中に，自己以外のクロス分あると，それぞれ等量の線形増加をすることにより，自己の増加量の 2 倍（クロストラック数が複数の場合 ）のパケット増加が観測される．自己およびクロス分の和が，ネットワーク帯域遅延積を超えた時点で，超過分を分母にして比率による除算減少は，全体量を 1/2 乗(複数トラフィック)することに相当し，(*a,b*)空間で原点を結ぶ直線上を比率分原点へ移動させた点となる．次のセッションでは，検出差分の半分（複数トラフィック）の等量線形増加を行い，同様の手順で輻輳ウィンドウを更新する．公平性を最大にする輻輳ウィンドウ制御の更新アルゴリズムを図 4.21 にまとめる．図 4.21 において， 3 DUP-ACK は 3 回重複 ACK を表し，RTO は再送タイムアウト値を示し，$\overline{RTT}$ は $RTT$ 標本全体平均値，$\overline{RTT_{cong}}$ は $RTT_{\min}$ を超える値の平均値である．

公平性の収束点は，任意の(*a,b*)　*a*=自己トラフィック，*b*=クロストラフィック，からスタートし，AIMD でウインドウ更新を実行すると，式(4.24)が成り立つ．

$$a = b = \frac{B * RTT_{\min}}{2} \tag{4.24}$$

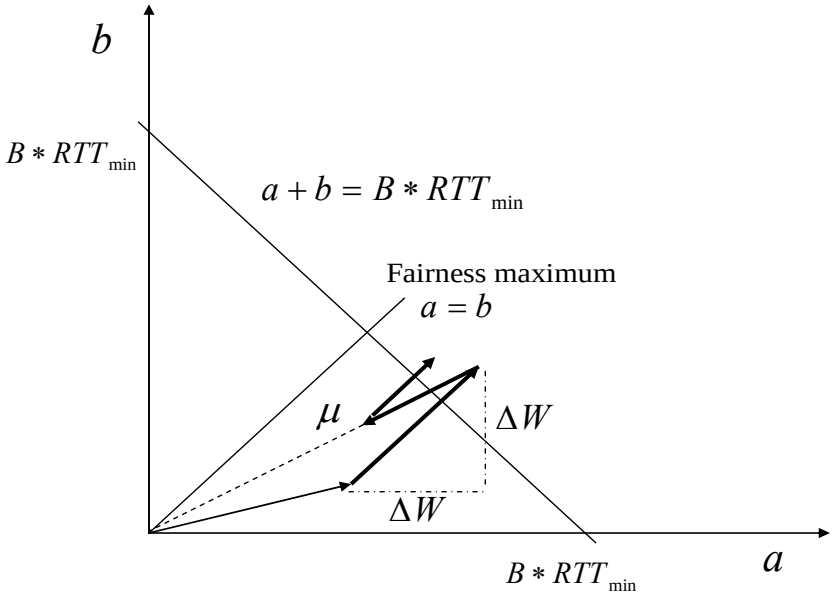
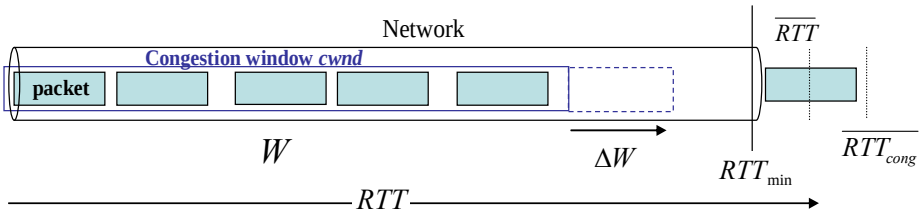



図 4.20　公平性（フェアネス）収束点




| | | $ACK_{i-1}$ | $ACK_i$ | | |
|---|---|---|---|---|---|
| | **AIMD** | | $\overline{RTT} < \overline{RTT_{cong}}$ | $\overline{RTT_{cong}} < \overline{RTT}$ | **3DUP-ACK, RTO** |
| **Acquisition** | **Inc** | $\Delta W$ | $n = \frac{\left(B * \overline{RTT} + B' * \overline{RTT_{cong}}\right) - W}{\Delta W}$ | | $\mu = \frac{1}{\mu'} = \frac{1}{n}$ |
| | **Dec** | | | | $\Delta W = -W * \frac{\overline{RTT_{cong}} - RTT_{\min}}{\overline{RTT_{cong}} - (1-\mu') * RTT_{\min}}$ |
| **Tracking** | **Inc** | $+1$ | | | |
| | **Dec** | | $\Delta W = -W * \frac{\overline{RTT_{cong}} - RTT_{\min}}{\overline{RTT_{cong}} - (1-\mu') * RTT_{\min}}$ | $-1$ | |

図 4.21　公平性を最大にするウィンドウ更新

本TCPウィンドウ更新アルゴリズムは，輻輳回避および回復制御のために，受信端末からの指標値$RTT/RTT_{min}$比とパケット通過時間$d$を使用し，$W$を帯域遅延積に収束させるようにし，輻輳によるパケットロスが発生した場合，FR/FRモードに対応しウィンドウを縮小し，他のトラフィックとの公平性を保つようパケットロス率$p$も併用した．これによりVegas，FAST系の収束性とNewReno系の公平性両者の利点を合わせ備えることになる．

## 4.5 ns-2 によるシミュレーション

ネットワーク端末系を図 9 に示す亜鈴型基本形とし，シミュレータ ns-2（Version2.29）[34]を用い，NewReno，Vegas，FAST[35]，提案方式を比較した．この図 4.22 のノード 2，3 はルータであり，そのキュー長は 40 パケットである．端末 1 から端末 5 へ UDP クロストラフィックをパケットサイズ 1000 B, レート 100 Mbps で 125 秒間流し, その間に端末 0 から端末 4 へ FTP（over TCP）でパケットサイズ 552 B の転送を行った．図 4.23 に結果を示す．縦軸は端末 0 の輻輳制御ウィンドウサイズ，横軸は時間である．

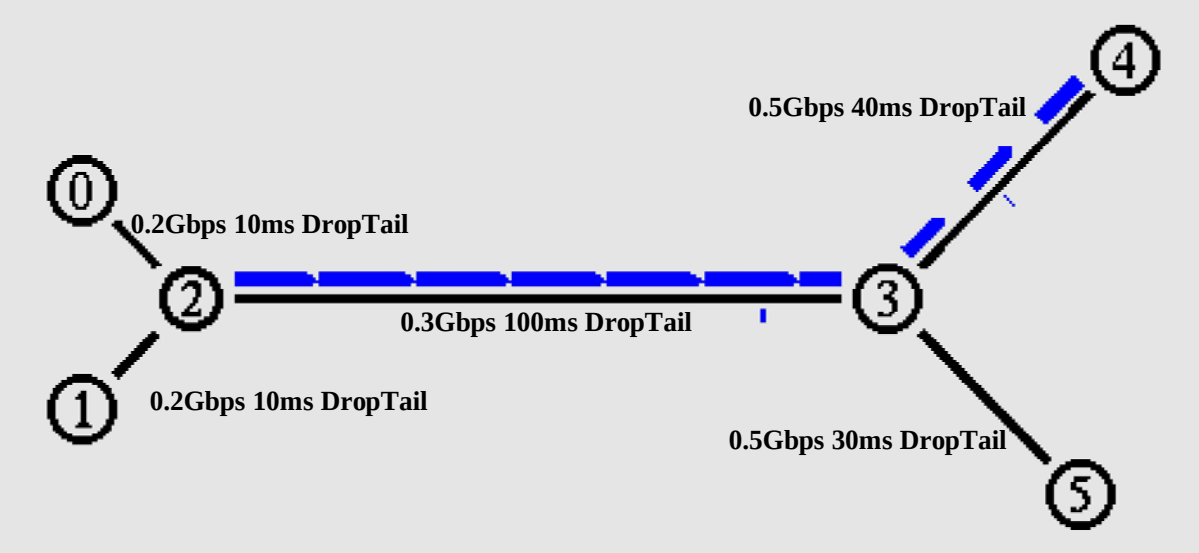



図 4.22 シミュレーションモデル

FAST は, 輻輳ウィンドウサイズ増加時, パラメータ $\alpha$ で示される加算増加定数を使用するが, ネットワーク内に存在するルータなどのパケットキュー長を知っておく必要がある．

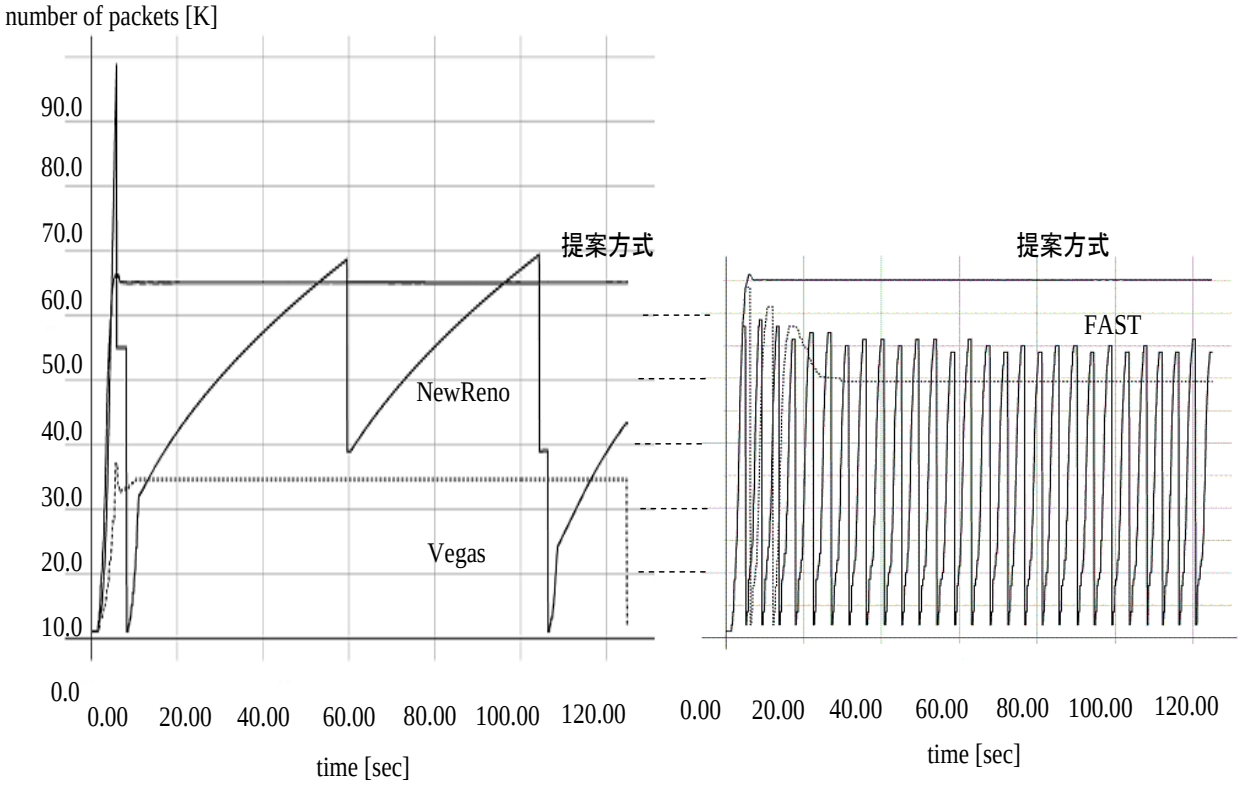



図 4.23　NewReno, Vegas, FAST, 提案方式ウィンドウサイズ

$\alpha$ が大きく見積もられると, FAST の輻輳制御ウィンドウは振動する解となる. 本提案方式は, ネットワーク帯域遅延積から，加算増加量を検知するため，このような振動解とはならない. 提案方式は，優れた収束特性を示しているが，NewReno と比較すれば，その収束性と高いスループットを実現できていることが分かる.

## 4.6　結言

端末 PC ハードウェア構成，転送メカニズム，MTU を考慮した TCP スループットモデルを確立した．そこから，受信端末内でのパケット通過時間をリアルタイム測定し，送信端末にフィードバックすることにより帯域遅延積を送信側で知り，ウィンドウと帯域遅延積を等しくする方式を提案した.

本方式は, 非輻輳時は, 受信端末パケット通過時間から推定されるネットワーク帯域を基に, 周回遅延最小値$RTT_{min}$から求められる帯域遅延積に等しくなるように，輻輳制御ウィンドウサイズを増加させ，輻輳状態に入ると，従来のウィンドウベース型TCP制御方式とは異なり輻輳状態を$RTT$の増加から検知し輻輳状態を回避させるように制御される．この意味からは,

レートベース型の特徴とレートベース型の特長を兼ね備える TCP スループット上限値に近い性能を与える方式である．高いスループットを確保できると共に，高い公平性を保証できることが期待できる．

実際のネットワークに適用するには，端末 PC へ本方式 TCP を実装し，クロストラフィックやパケット遅延変動などの影響を評価するエミュレーションが必要である．

# 第 5 章

## 結論

ISO レイヤーモデルにおいては，各層において PDU の処理が行われるため，各層にバッファメモリが設けられる，このバッファメモリを使い TCP プロトコルスタックによる制御が行われる．

TCP エンドツーエンド通信系を制御系と見た場合の各層のバッファ処理によるスループットを劣化させないためには，ネットワーク層の帯域遅延積に上位層のバッファ量を等化させる必要がある．ネットワーク帯域遅延積は時間とともに動的に変化するため，端末により測定しなければならない．従来の方法であるウィンドウベース型は，帯域遅延積値をパケットロスにより検知する方法であり，本質的に輻輳回避の方法ではない．また，レートベース型では，遅延量は測定されて検知しているが，帯域量は測定されず，利用されていないため正確な帯域遅延積量が検知できない．

本研究では，端末の各層内バッファ量をネットワークの帯域遅延積に等化させ各層間でのスループット劣化をなくし，ネットワーク・端末総合系としての TCP スループットの劣化を最小限にするような TCP 方式を求めた．スループット低減を最小にするには，正確なネットワークの帯域遅延積を端末において検知する必要がある．TCP 通信に用いられるパケットとその受信確認応答（ACK）を用いて，ネットワーク遅延量と非輻輳時に，ネットワーク帯域を複数のパケットを用いて測定し，帯域遅延積を検知する方法について検討した．

最初に，広範囲の遅延に対し成立する TCP スループットの真の上限値を，ネットワーク・端末総合系として捉え端末処理速度を考慮し解析的に求め，受信端末パケット通過時間を測定することにより求めた実測値を用いて検証を行い，理論上限式と受信端末パケット通過時間による値はよく一致する結果を得た．従って，理論上限式は，TCP スループット低下改善の指針を与えるものであると言える．

また，受信端末でのパケット通過時間測定においては，10 パケット程度の個数で通過時間が求められ，本測定により，実時間 1 *RTT* 内に帯域推定が可能であることが分かる．さらに，以下の項目が，結論できる．

・従来の TCP スループットモデルに対しても上限値として成立する．
・スループット上限値の MTU サイズ依存性を明確にした．なお，仮想的に *MTU* を無限大と考えると，上限値は端末ハードウェアバス帯域および通信路帯域の調和合計の逆数に収束する．
・MTU サイズ 9K バイトのパケットを用いて，受信端末でパケット遅延 *D* を測定すれば，ネットワーク帯域を推定することができる．

今後の課題として，クロストラヒックが存在する環境や，パケット到着間隔の変動を考慮した提案手法の拡張を検討している．

端末 PC ハードウェア構成，転送メカニズム，MTU を考慮した TCP スループットモデルを確立した．そこから，受信端末内でのパケット通過時間をリアルタイム測定し，送信端末にフィードバックすることにより帯域遅延積を送信側で知り，ウィンドウと帯域遅延積を等しくする方式を提案した．

本方式は，従来のウィンドウベース型 TCP 制御方式とは異なり輻輳状態を *RTT* の増加から検知し輻輳状態を回避させるように制御される．この意味からは，レートベース型に近い TCP スループット上限値を与えるものであり，高いスループットを確保できると共に，同じプロトコルを使うという条件下では，高い公平性を保証できることが期待できる．

今後に残された課題としては，シミュレーションの充実と，端末 PC へ本方式 TCP を実装し，エミュレーションを行うことが挙げられる．さらに従来方法として現在も広く使用されている TCP バージョンたとえば NewReno や最近の FAST などインタープロトコル環境での公平性などを検討しシミュレーションする必要性がある．

# 謝辞

本研究遂行並びに本論文作成の全過程を通じて，多大なる御指導と御鞭撻を賜わりました大阪大学大学院情報科学研究科情報ネットワーク学専攻の村上孝三教授に謹んで感謝の意を表します．本論文をまとめるにあたり，情報ネットワーク学全般に関して，懇切なる御教示と御助言を賜わりました大阪大学大学院情報科学研究科情報ネットワーク学専攻の村田正幸教授，今瀬真教授，東野輝夫教授，中野博隆教授に感謝の意を表します．種々懇切なる御教示と御助言を賜わりました大阪府立大学工学部戸出英樹教授と大阪大学大学院情報科学研究科情報ネットワーク学専攻村上研究室の木下和彦准教授に厚く感謝申し上げます．本研究を進める際にさまざまな点でお世話になりました大阪大学大学院情報科学研究科情報ネットワーク学専攻村上研究室事務補佐員の佃真理子氏，研究室において御協力を頂きました，村上研究室大学院生各位に感謝します．

また，大学院博士後期課程に進学するにあたり，常日頃から種々の提言援助をいただいた，三菱電機株式会社情報総合技術研究所の役員技監村上篤道博士，マルチメディア符号伝送部浅井光太郎部長，三菱電機株式会社コミュニケーションネットワーク製作所光トランスポート部本島邦明部長，三菱電機株式会社情報総合技術研究所の方々に感謝します．ご理解とご援助いただきました，けいはんな新産業創出・交流センター長長岡良富博士に感謝します．

# 参考文献


[1] J. Padyhe, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” *SIGCOMM 1998*, vol.28, no.4, pp.303-314, September 1998.

[2] M. Mathis, J. Semke, J. Mahdavi and T. Ott, “The Macroscopic Behavior of the TCP connection Avoidance Algorithm,” *ACM SIGCOMM Computer Communication Review*, vol.27, pp.67-82, July 1997.

[3] M. Ripeanu, “Transport Level Protocols: Performance Evaluation for Bulk Data Transfers,” poster, International Conference for High Performance Computing, Networking, Storage, and Analysis *SC 2001*, November 2001.

[4] M. Ito, K. Asai, T. Murakami, R. Suzuki, K. Kinoshita, H. Tode, and K. Murakami, “An Estimation of the Upper Bound of TCP Throughput in High-Speed Networks,” *Proceeding of EuroIMSA*, pp.142-147, February 2006.

[5] IETF RFC1242 “Benchmarking Terminology for Network Interconnection Devices,” Network Working Group S. Bradner, Editor Harvard University July 1991.

[6] L. Brakmo and L. Peterson. “TCP Vegas: End to End Congestion Avoidance on a Global Internet.” IEEE Journal on Selected Areas in Communication, Vol 13, No. 8 (October 1995) pages 1465-1480.

[7] IETF RFC2581 “TCP Congestion Control,” Network Working Group M. Allman NASA Glenn/Sterling Software V. Paxson ACIRI / ICSI W. Stevens Consultant.

[8] A. Zanella, G. Procissi, M. Gerla, and M. Y. Sanadidi, "TCP Westwood: Analytic Model and Performance Evaluation", In *Proceedings of IEEE Globecom 2001*, Volume: 3, pp 1703-1707, San Antonio, Texas, USA, November 25-29, 2001.

[9] IETF RFC2582 “The NewReno Modification to TCP's Fast Recovery Algorithm,” April 1999, EXPERIMENTAL: NOW OBSOLETED, Obsoleted by 3782.

[10] IETF RFC3649 “High Speed TCP for Large Congestion Windows,” Network Working Group S. Floyd ICSI, December 2003.

[11] Tom Kelly, “Scalable TCP: Improving Performance in High speed Wide Area Networks”. *Computer Communication Review* 32(2), April 2003.

[12] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh, (2005). "FAST TCP : from theory to experiments," *IEEE Network* 19 (1): 4–11. doi:10.1109/MNET.2005.1383434.

[13] M. Ito, Y. Nakaki, K. Tsutsumi, O. Ito, “An Evaluation Method of Recording Characteristics of the Light Intensity Modulation Direct Overwrite(DOW)”, *Proceeding of MORIS ’92*, December 1992.

[14] B. Sikdar, S. Kalyanaraman and S. Vastola, “Analytic Models for the Latency and Steady-State Throughput of the TCP Tahoe, Reno and SACK,” *IEEE/ACM Transaction on Network*, vol.11, pp.959-971, December 2003.

[15] N. Cardwell, S. Savage and T. Anderson, “Modeling TCP Latency,” *INFOCOM 2000*, vol. 3, pp.1742-1751.

[16] V. Guffens, “Compartmental Fluid-Flow Modeling Packet Switched Networks with Hop by Hop Control,“ PhD. Thesis, April 2003.
*http://edoc.bib.ucl.ac.be:81/ETD-db/collection/available/BelnUcetd-12062005-160308/unrestricted/guffens_PhD_thesis.pdf*

[17] L. Deri, “Improving Passive Packet Capture: Beyond Device Polling,” *SANE 2004*, October 2004.

[18] V. Anand and B. Hartner, “TCP/IP Network Stack Performance in Linux Kernel 2.4 and 2.5,” *Linux Symposium*, pp.8-30, July 2003.

[19] RFC 1071 R. Braden, D. Borman, C. Partridge, ”Request for Comments: 1071,” BBN Laboratories, September 1988.L. Kleinrock, “The Latency/Bandwidth Tradeoff in Gigabit Networks,” *IEEE Communications Magazine*, vol.4, pp36-40, April 1992.

[20] Intel Product Brief, “Intel IXP1200 Network Processor Family,”
*http://download.intel.com/design/network/ProdBrf/27904001.pdf*.

[21] H. Kamezawa, M. Nakamura, J. Tamatsukuri, N. Aoshima, M. Inaba and K.Hiraki, “Inter-layer coordination for parallel TCP streams on Long Fat pipe Networks,” *SC2004*, Nov. 2004.

[22] 玉造潤史，吉野剛史，稲上克史，菅原豊，稲葉真理，平木敬,"10 ギガビットネットワーク上での高効率TCP/IP通信の実現," SWoPP 2006 (並列/分散/協調処理に関するサマーワークショップ),情報処理学会研究報告 2006-HPC-1077, pp. 299-304, August 2006.

[23] 長谷川 洋平，下西英之，村瀬 勉,”ハードウェア TCP-NIC(TCP オフロードエンジン)の TCP 動作検証と性能評価・考察,“ 第 13 回インターネット技術第 163 委員会研究会, ITRC meet13, 2003 年 5 月.

[24] L. Kevin and M. Baker, “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay,” *SIGCOMM 2000*, pp.283-294, September 2000.

[25] A. Tirumala, F. Qin, J. Dugan, J. Ferguson and G. Kevin, “Iperf 1.7.0 / 2.0.2,” May 2005. http://dast.nlanr.net/Projects/Iperf/.

[26] L. McVoy and C. Staelin, “LMBench 1.0,” *http://www.bitmover.com/*.

[27] 寺井達彦，松尾孝広，長谷川剛，村田正幸，“TCP による高速データ転送のための通信処理軽減手法の提案と実装，”信学技報 IN2003-23(2000-05).

[28] M. Jain, R. S. Prasad and C. Dovrolis, “The TCP Bandwidth-Delay Product revisited: network buffering, cross traffic, and socket buffer auto-sizing”.
*http://www.cercs.gatech.edu/tech-reports/tr2003/git-cercs-03-02.pdf*.

[29] Q. He, C. Dovrolis, and M. Ammar, “On the Predictability of Large Transfer TCP Throughput,” *SIGCOMM 2005*, pp.145-156, August 2005.

[30] J. Navratil and R. L. Cottrell, “ABwE: A Practical Approach to Available Bandwidth Estimation,” *PAM 2003*, April 2003.
*http://www.slac.stanford.edu/grp/scs/net/papers/noms/noms14224-122705-d.doc*.

[31] 津川知郎，長谷川剛，村田正幸，“インライン計測に基づく TCP によるバックグラウンド転送方式，”信学技報 IN2004-174(2005-02).

[32] A. Baiocchi, A. DeVendictis, A. Monticelli, “Simple Models and their Limits for TCP/IP Network Analysis and Dimensioning,” *ICC 2002*, vol.4, pp.2299-2303, April 2002.

[33] R. Bin and Z. Xiaolan, “Implementation of Transaction TCP in Linux Kernel 2.4.2,” *http://www.cl.cam.ac.uk/~br260/doc/ttcp_impl.pdf*.

[34] ns-2, *http://www.isi.edu/nsnam/ns/*.

[35] S.Hedge, D. Lapsley, B. Wydrowski, J. Lindheim, D. Wei, C. Jin, S. Low and H. Newman, “FAST TCP in High-speed Networks: An Experimental Study,” *Proceedings of GridNets*, San Jose, CA, October 29, 2004.

[36] V. Anand, B. Hartner, “TCP/IP Network Stack Performance in Linux Kernel 2.4 and 2.5.,” *Proceeding of the Linux Symposium*, Ottawa, June 2002.

[37] T. Henriksson, U. Nordqvis and D. Liu, “Embedded Protocol Processor for Fast and Efficient Packet Reception,” *ICCD 2002*.

[38] R. Bin and Z. Xiaolan, “Implementation of Transaction TCP in Linux Kernel 2.4.2,” *http://www.cl.cam.ac.uk/~br260/doc/ttcp_impl.pdf*.

# ネットワーク・端末総合系におけるＴＣＰスループットの上限値に関する研究

２００８年９月

伊藤正也