TCP 輻輳制御アルゴリズムの動的切り替えによるインタラクティブ通信の性能改善に関する研究

小口 直樹

博士(情報学)

総合研究大学院大学
複合科学研究科　情報学専攻

平成 27 年度

(2015)

2016 年 3 月

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士 (情報学) 授与の要件として提出した博士論文である.

審査委員:

阿部 俊二 (主査)	国立情報学研究所/ 総合研究大学院大学
漆谷 重雄	国立情報学研究所/ 総合研究大学院大学
計 宇生	国立情報学研究所/ 総合研究大学院大学
戸出 英樹	大阪府立大学
福田 健介	国立情報学研究所/ 総合研究大学院大学
山田 茂樹	国立情報学研究所/ 総合研究大学院大学

(主査以外はあいうえお順)

A Study of Performance Improvement in Interactive Communication by Dynamic Changing of TCP Congestion Control Algorithms

Naoki Oguchi

DOCTOR OF
PHILOSOPHY

Department of Informatics,

School of Multidisciplinary Sciences,

The Graduate University for Advanced Studies (SOKENDAI)

March, 2016

A dissertation submitted to the Department of Informatics, School of Multidisciplinary Sciences,

The Graduate University for Advanced Studies (SOKENDAI) in partial fullfillment of the requirements for

the degree of Doctor of Philosophy

Advisory Committee:

Shunji Abe(Chair)	National Institute of Informatics/ The Graduate University for Advanced Studies
KensukeFukuda	National Institute of Informatics/ The Graduate University for Advanced Studies
Yusheng Ji	National Institute of Informatics/ The Graduate University for Advanced Studies
Hideki Tode	Osaka Prefecture University
Shigeo Urushidani	National Institute of Informatics/ The Graduate University for Advanced Studies
Shigeki Yamada	National Institute of Informatics/ The Graduate University for Advanced Studies

(Alphabet order of last name except chair)

目次

図目次

表目次

論文要旨

今日，クラウドサービスが広く普及しており，組織における PC の紛失や盗難による情報漏えい，設備投資削減，省電力などの観点から，多くの企業が，従業員の PC をシンクライアントに置き換え，クラウド上で提供される仮想デスクトップサービス(DaaS)への移行を検討している．

仮想デスクトップサービスのようなインタラクティブ(対話的) なアプリケーションにおいては人が快適に利用可能な応答時間は一秒以下とも言われる．そのため，数十 ms から数百 ms の性能劣化がアプリケーションの使用感に大きく影響する．

しかしながら，仮想デスクトップサービスの多くは，トランスポートプロトコルとして TCP (Transmission Control Protocol)を用いるため，外出先からモバイルデバイスにより国外のクラウドにアクセスするといった，ネットワークの遅延や，無線網におけるパケット廃棄が多い利用形態では，通信性能が劣化し易い．

また, 一般に, 大容量のファイル転送を行うアプリケーション(例えば FTP: File Transfer Protocol)は，十分な量の送信データを，TCP のパケット送信速度よりも速い速度で OS に対し送信依頼するため, TCP の輻輳ウィンドウサイズは輻輳制御アルゴリズムの設計上の最大速度で広がる．一方で，仮想デスクトップサービスのように，端末の操作に対しサーバが応答データとして画面の更新情報を送信するインタラクティブなアプリケーションは，比較的送信データ量が少ないため TCP の輻輳ウィンドウサイズが広がりにくく，瞬間的なデータ送信に時間がかかることから，アプリケーションの応答性能が低下するという問題がある．

こうした要因により，クラウド上の仮想デスクトップサービスは，応答性や操作性が悪化しやすく，なかなか利用が進まないのが現実である．

これまで非常に長い時間をかけて TCP の様々な輻輳制御アルゴリズムの研究が行われ，特定の通信環境において効率的に通信を行える輻輳制御アルゴリズムが複数提案されてきた．しかし，これらの輻輳制御アルゴリズムは，アプリケーションの送信データサイズや，往復遅延時間(RTT: Round Trip Time)，廃棄率

といったネットワークの特性により，その応答性能が優位となる通信環境が異なる[47]．例えば TCP Hybla[57]と呼ばれる輻輳制御アルゴリズムは，RTT が小さい時は他の輻輳制御アルゴリズムよりもゆっくりと輻輳ウィンドウサイズを広げるが，RTT が大きいと最も速く輻輳ウィンドウサイズを広げるようになるため，衛星通信など RTT が大きい環境で高い応答性能を示す．また，TCP Westwood と呼ばれる輻輳制御アルゴリズムは，無線のエラーが多い環境で高い応答性能を示す．このように，ある特定の通信環境において，理想的な輻輳制御アルゴリズムを設計することは可能である．しかし，無線アクセス網を介しクラウドを利用する形態のように，ネットワークの特性が様々な要因で変化し得る通信環境で常に理想的な応答性能を示す輻輳制御アルゴリズムを設計することは困難であると考えられる[39].

そこで，本論文ではインタラクティブな通信アプリケーションにおいて，通信路の品質に合わせ最適な TCP の輻輳制御アルゴリズムを動的に選択することで，通信アプリケーションが一時に連続送信するデータの最初から最後のパケットまでの転送時間(レイテンシ)を改善する方法(Reconfigurable TCP)を提案した．本提案方式は，3 個以上連続的に廃棄が発生した場合(バースト的廃棄)と高々2 個までの連続廃棄が発生した場合(非バースト的廃棄)という二つの廃棄パタンにより, Linux OS の TCP が輻輳制御において異なる挙動を示すことに着目し，これら廃棄パタン毎の廃棄率と RTT に応じて最適な輻輳制御アルゴリズムを選択するものである．新たな TCP の通信方式の提案では，TCP のフェアネスを考慮する必要がある．本提案方式は，標準 TCP である TCP Reno との公平性を示す TCP Friendliness については, Reconfigurable TCP が選択する対象の輻輳制御アルゴリズムを限定し，限られた性能改善範囲で利用するならば公平性の差を小さくできることがわかった．また，同じ輻輳制御アルゴリズム(ここでは Reconfigurable TCP)を用いた複数のセッションの RTT が異なる場合の公平性を示す RTT fairness については，Reconfigurable TCP は RTT によるレイテンシの差がでにくいため，公平性の差がでにくいことがわかった.

次に提案方式を NS-3 ネットワークシミュレータ[64]に実装し，GE (Gilbert -Eliott)チャネルモデル[49][50][51]を用いた固定無線環境において評価した. 本評価では実際の Wi-Fi 網の廃棄特性を計測し，この廃棄特性に合わせ GE モデルのパラメタを決定し提案方式を評価した．さらに，IEEE802.16e(WiMAX)をモデル

化した移動無線環境における評価も行った．シミュレーションによる評価では，最大 45%の応答時間の改善が得られた．

さらに，提案方式を Linux OS に実装し，LTE 網を用いた移動無線環境で実機評価を行った．実機評価では，最大 34%の応答時間の改善が得られることがわかった．また，こうした応答時間の改善が主観評価としてどれだけの改善につながるかを考察し，ITU-T 勧告で定める 5 段階のオピニオン評価(MOS)のスコアを 0.5 から 1.5 改善できることがわかった．

これらの評価から，通信路の品質に合わせ最適な輻輳制御アルゴリズムを動的に切替える本提案方式は，様々な環境で応答時間改善効果が得られ，クラウド環境で仮想デスクトップサービスなどのインタラクティブなアプリケーションを利用する際のレイテンシ改善の一手段になり得ることを示した．

Abstract

Today, many cloud services become widespread and it is expected that utilization of virtual desktop services over the cloud services are accelerated to avoid information leaking by loss or theft of PCs, to reduce capital investment for offices and to save electricity consumption. Many companies are propelling utilization of cloud services by replacing employees' PCs into thin clients.

In general, it is known that human feels comfortable when response time of interactive applications such as virtual desktop is less than one second. Therefore, response time degradation ranging from dozens of milliseconds to several hundreds milliseconds significantly affects the experience quality of these applications.

However, most interactive applications use the TCP protocol for the communication between servers and mobile devices. Packet losses in wireless networks and long delays on international lines have a great influence on end-to-end TCP sessions and degrade communication quality.

Applications that execute bulk transfer such as FTP (File Transfer Protocol) continuously write sufficient amount of data into the buffer between the applications and TCP stack with quicker speed than the packet send rate of the TCP stack. As there are sufficient transmitting data, the sender TCP stack sends data packets at the rate of current congestion window size and receives sufficient ACKs to fully expand the congestion window size. In this case, the sender TCP expands its congestion window size at the maximum rate on design of the congestion control algorithms.

On the other hand, interactive applications such as virtual desktop services expand TCP congestion window size slowly as those applications gradually send data with little chunks and it causes degradation of response time. For the reason of low usability with low responsiveness of virtual desktop services, companies make slow progress in utilization of cloud services in reality.

So far, various TCP congestion control algorithms have been studied for a long time, and plural algorithms that could communicate effectively in specific communication environment have been proposed.

It depends on the data transmission rate of applications and the characteristics of networks such as round trip time (RTT) and the packet loss rate[47] which congestion control algorithm is the most superior in performance. For example, when RTT is short, the congestion control algorithm called TCP Hybla[57] expands the congestion window size more slowly than other congestion control algorithms. However, it shows the highest performance in satellite communications because it opens the congestion window size more quickly when RTT is large. Moreover, the congestion control algorithm called TCP Westwood shows high performance in the wireless environment with many signal errors. In this way, it is thought to be possible to design an ideal congestion control algorithm in a particular communication environment. However, it is thought that many man-hours and verifications are necessary to design a congestion control algorithm that shows universally ideal performance for communication environment which characteristics can dynamically change as is the case accessing an overseas cloud service through a wireless access network.

Therefore, this dissertation proposes a method (Reconfigurable TCP) to improve response time in interactive communications, by dynamically selecting the most suitable congestion control algorithm according to network characteristics. The proposed method selects an algorithm according to packet loss rates for two packet loss patterns and RTT, utilizing the Linux's congestion detecting mechanism, which distinguishes congestions based on whether at most two continuous packet losses occurred or more than three continuous packet losses occurred. It is also important to take into account the fairness when proposing a new TCP variant. From the view point of TCP friendliness, Reconfigurable TCP can reduce the deviation from ideal TCP friendliness by restricting the selecting congestion control algorithms. From the view point of RTT fairness, Reconfigurable TCP has little deviation as it can reduce the deviation by changing congestion control algorithms.

Next, this study shows the results of evaluation using a GE channel model in fixed wireless environment by implementing the proposed method on the NS-3 network simulator. In this evaluation, the parameters of the GE channel model are decided reflecting the characteristic of a real Wi-Fi network. Furthermore, the proposed method was evaluated in mobile wireless environment with IEEE802.16e (WiMAX) by using

the simulator. It was revealed that up to 45% of improvement of the response time was achieved in the evaluation.

In addition, the proposed method was implemented on Linux OS and evaluated on a LTE (Long Term Evolution) network. It was revealed that up to 34% of improvements of the response time were achieved in the real machine evaluations. Then it was clarified how much these improvements contribute to the improvements of subjective evaluations. The proposed method can improve MOS score, which is an opinion evaluation defined in ITU-T recommendation, by 0.5 to 1.5.

From these evaluations it was revealed that for around 40% of improvement of response time was achieved, and the proposed method could be an effective mean to improve the response time when interactive communication such as virtual desktop services are used in cloud environment.

第1章 序論

1.1 はじめに

現在，クラウドサービスが広く普及してきているが，組織におけるセキュリティ，設備投資削減の観点，省電力の観点から，今後，クラウドを利用した仮想デスクトップサービスの利用が増加することが期待されている[6].

例えばセキュリティの観点では，企業における情報漏洩が社会的問題となっている．情報漏えいの原因のトップは PC の置き忘れや盗難である．仮想デスクトップを導入すれば，情報を企業側のサーバに集約できるため，PC を紛失したり，盗難にあっても情報漏洩を防ぐことができる．

また，IT(Information Technology)設備投資削減の観点では，IT 設備は技術革新のサイクルが速く耐用年数が短い．ソフトウェアはバージョンアップが多く，すぐに陳腐化するため固定資産として IT 設備を所有する利点は少なく，高額なハードウェアを購入する代わりに，クラウドを用いることで IT 資産を変動費用化できる．

さらに，日本における省電力の観点では，温室効果ガス排出削減目標として，2005 年度比で 3.8%減とする目標を立てている[1]. 地球規模の温暖化対策として多くの企業が温暖化ガス(CO_2)の排出量削減に様々な面で取り組んでいる[2]．その一つとして進めている，業務用 PC のクラウドへの移行は，温暖化対策の一手段となり得ると考えられている[3].

こうした状況から，図 1-1 に示すように多くの企業が従業員の端末をシンクライアントに置き換え，クラウドの利用を推進している．

ところで，仮想デスクトップなどのインタラクティブなアプリケーションでは人が快適に利用可能な応答時間の限度は一秒以下とも言われ[25][26]，数十 ms から数百 ms の応答性能の劣化がアプリケーションの使用感に大きく影響する．

ところが，近年，ネットワークへの接続形態が多様化してきているため，仮想デスクトップサービスの性能が出にくい．

クラウドサービスは様々な場所で提供されており，海外にあることも多い．そのため，サービスの提供場所により，RTT が異なってくる．

また，今日様々なワイヤレスブロードバンド技術が開発され，ユーザは情報漏洩対策が必要なモバイル端末を外に持ち出し，様々な場所から無線アクセス網を介してネットワークに接続できるようになったが，場所により通信特性が異なる．

さらに，近年の無線アクセス技術は，車や高速列車などの移動体からも通信できる仕様となっているが，ユーザの移動に伴い，無線環境の特性は時々刻々変化する．

このように，ユーザ端末とサービスを提供するサーバの間は，無線アクセス網，国際回線などの多種多様な特性を持ったネットワークを経由して通信する．このような通信環境は，無線アクセス網によるパケット廃棄が発生することに加え，国際回線における大きい遅延を生じやすい．

ところが，仮想デスクトップサービスに用いられる TCP(Transmission Control Protocol)プロトコルは，経由するネットワークに遅延や廃棄があると通信性能が劣化しやすい特性を持つことが知られている．そのため，クラウドで仮想デスクトップサービスを利用する場合，応答性や操作性が悪いことからなかなか利用が進まないという問題が生じている．

そこで，通信時のネットワーク特性の違いや，通信中のネットワーク特性の変化に応じて，適切な通信プロトコルを用いれば，ユーザは常に快適に仮想デスクトップを利用できるのではないか? そして，仮想デスクトップサービスの応答性能を改善できれば，クラウドの利用が積極的に進み，端末の紛失や盗難による情報漏洩についての懸念が軽減できると思われる．

本論文では，通信環境の変化に応じて，通信プロトコルを最適化し，通信アプリケーションの応答性能を改善することに取り組む．

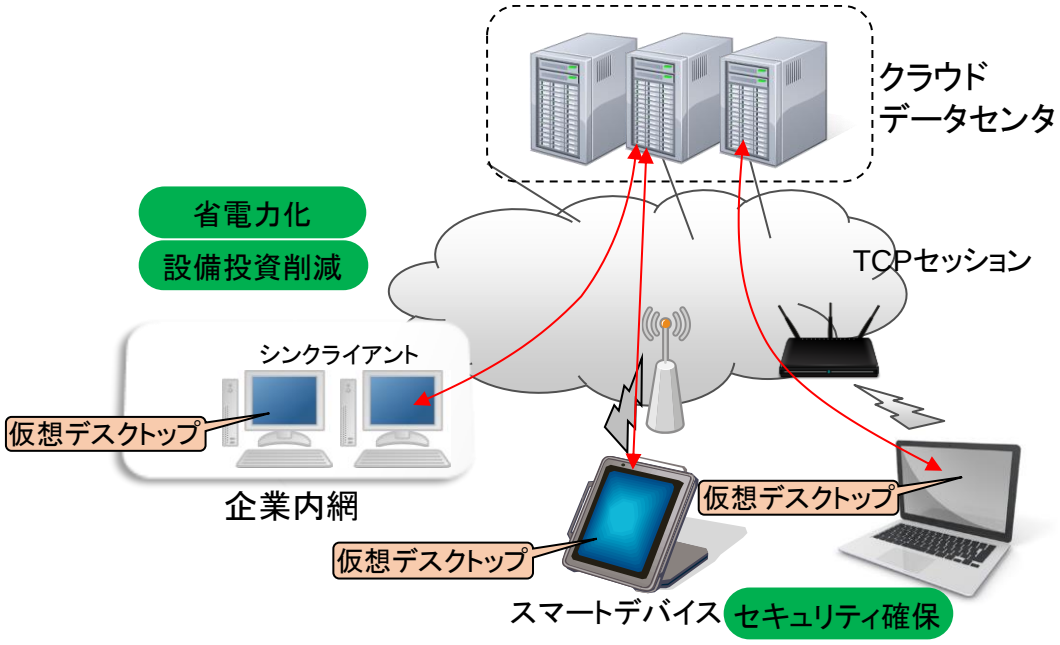


図 1-1 クラウド上の仮想デスクトップサービスの活用

1.2　クラウドサービス

本節では，クラウドサービスについて概観し，その中でも，今回着目している DaaS の仕組み，DaaS に用いられるプロトコルの動作について説明する．

1.2.1　クラウドサービスとは

クラウドサービスとは，主としてデータセンタのサーバ上に仮想 CPU，仮想ストレージ，仮想ネットワークなどの仮想環境を構築し，時間貸しなどの形態で契約した顧客に仮想環境を貸し出すサービスである．顧客にとっては，物理的な機器の管理をアウトソースできる，必要な時に必要な分の仮想環境を契約することで，費用を削減できるといったメリットがある．

これまで，クラウドサービスは大量データ処理，データバックアップなどの用途に多く利用されてきたが，近年，仮想デスクトップサービス(DaaS)[7]，クラウドゲーム[8]などの人間との対話を介するインタラクティブな用途に利用され始めている．クラウドサービスは，提供するサービスのレイヤにより，SaaS，PaaS，IaaS，DaaS，GaaS といった様々なタイプが存在する．

1.2.1.1　SaaS

Software as a Service(SaaS)は，アプリケーションプログラムを実行する環境を提供するもので，自前でサーバを立ち上げずとも，クラウド上のサーバでアプリケーションを実行し，インタネットへサービスを公開可能とする．また，WEB ブラウザの中にアプリケーションのウィンドウを表示して利用することができる．代表的な SaaS としては，[9]や[10]などがある．

1.2.1.2　PaaS

Platform as a Service (PaaS) は，アプリケーションの開発環境を提供するサービスで，Web アプリケーションを開発するための環境を提供する．代表的なものに，[11]などがある．

1.2.1.3　IaaS

Infrastructure as a Service (IaaS) は，サーバ，ストレージ，ネットワーク機器を利用者が自由に構築できるサービスで，利用者はサーバの OS や CPU スペックも料金に応じて選択できる．IaaS の代表的なものとして[12]がある．

さらに，膨大な量のデータを解析するために，計算リソースを強化したサー

ビスとして，Hadoop[13]クラウドを提供する[14]などがある．

また，クラウド上に大容量データを保存できるクラウドストレージサービスも, IaaS の一種と考えられている. クラウドストレージサービスとしては, [15], [16]，[17]，[18]，[19]などがある．

1.2.1.4 DaaS

Desktop as a Service (DaaS)は，リモートホストのデスクトップ画面をローカルホストに表示しリモートホストを操作可能とする仮想デスクトップアプリケーションを，クラウド上で提供するサービスである．現在，複数のクラウド事業者が商用の DaaS サービスを提供している[20] [21].

1.2.1.5 GaaS

Gaming as a Service (GaaS)は，ゲームの画面をローカルのデバイス(PC や移動デバイス)へストリーミング配信し，ローカルのデバイスの操作だけをクラウドへフィードバックし，クラウド上でゲームを実行可能とするサービスである．商用のサービスとしては，例えば，[22]や[23]がある．

1.2.2 DaaS の仕組み

図 1-2 に DaaS の仕組みを示す. DaaS のサーバとクライアントにおける仮想デスクトップアプリケーション間のプロトコルには，RDP (Remote Desktop Protocol)や PCoIP(PC over IP)といったプロトコルが用いられる．これらのプロトコルでは，仮想デスクトップのクライアントにおける，ユーザのキーボードやマウスの操作は，コマンドとしてクラウド上の仮想デスクトップサーバへ送信される．これに対し，仮想デスクトップサーバは操作に対するデスクトップの更新情報を送信する．この更新情報は最大 64KBytes 程度の大きさになる．キーボードやマウス操作から，デスクトップが更新されるまでの時間を**応答時間**と呼ぶこととする．

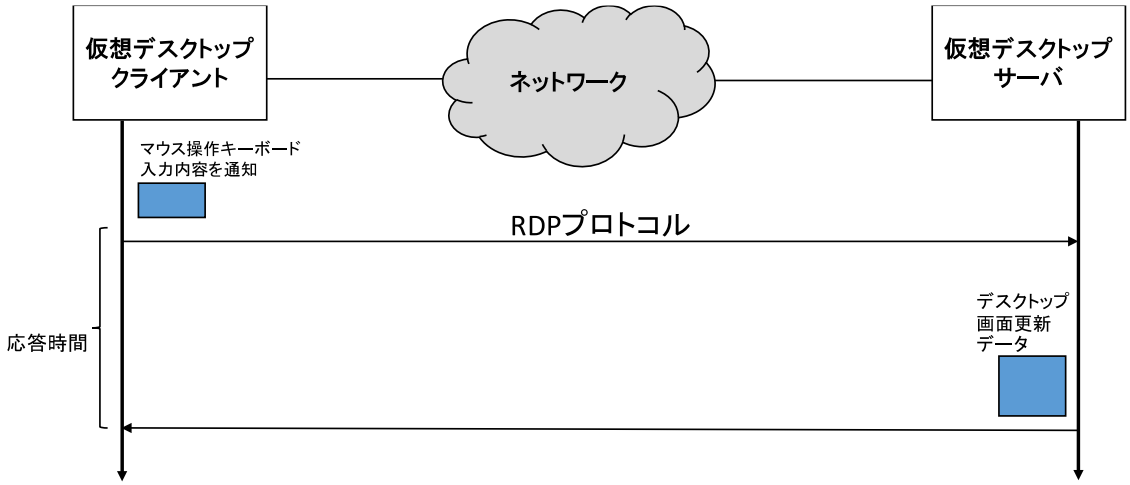


図 1-2 DaaS の仕組み

1.3 アプリケーションが求める通信性能

世の中には，様々なタイプのサービスを提供する通信アプリケーションがあり，ネットワークに求める性能は様々である．主に，以下のような性能要件を求めるものがある．

- 高い平均スループットで確実なデータ転送を求めるアプリケーション
- 確実かつリアルタイムなデータ転送を求めるアプリケーション
- リアルタイムなデータ転送を求めるアプリケーション

1.3.1 高い平均スループットで確実なデータ転送を求めるアプリケーション

大規模ファイル転送のように，データの受信側では，まとまった単位のデータを転送し終わってから受信データの処理を開始するアプリケーションがこのような通信性能を要求する．コンピュータがデータ処理を行うため，データの完全性は保証する必要があり，通信プロトコルとしてデータの完全性を保証している TCP が使われることが多い．人間が通信を介して行うオペレーションが含まれないためリアルタイム性(人間が遅さを感じない時間内に処理を完了する)は求められないが，なるべく早くデータの処理を開始できるよう高い平均スループットが期待される．

1.3.2 確実かつリアルタイムなデータ転送を求めるアプリケーション

データの確実性とリアルタイム性の両方を要求するアプリケーションは，代表的なものに仮想デスクトップのようなインタラクティブに通信が行われるアプリケーションがある．こうしたアプリケーションは，人間が遅さを感じない時間内に比較的少量のデータを転送しつつ人間のオペレーションを介しながら処理が進むアプリケーションである．転送するデータは確実に届く必要があるため TCP が使われる場合が多い．リアルタイム性が求められるため，応答の速さが求められる一方，転送するデータが比較的少ないためスループットの高さはそれほど求められない．インタラクティブなアプリケーションでは人が快適に利用可能な応答時間の限度は一秒以下と言われる[25][26].

1.3.3 リアルタイムなデータ転送を求めるアプリケーション

リアルタイム性のみ要求するアプリケーションとして，ストリーミングや VoIP(Voice over Internet Protocol)といった, 映像や音声を扱うアプリケーションがある．これらは，所定の時間内にデータが届かないと，人間が視聴した際に不自然さを感じるためリアルタイム性が求められる一方，多少のデータ廃棄があっても人間には感じられないことから，再送によるエラー回復を行わない UDP を使う場合が多い.

1.4 DaaS 応答性能劣化の通信による要因

1.2.2 節に示すように, 一定レベルの通信性能が求められているにも関わらず, TCP の通信性能は利用環境により劣化する．本節では，まず TCP 通信の仕組みについて述べ，今日の多様化した通信環境と，それにより引き起こされる TCP 性能劣化のメカニズムを説明する.

1.4.1 TCP プロトコルを用いた通信の仕組み

図 1-3 は通信を行うホストにおける TCP プロトコルスタックの構造を示している．ユーザプロセスである通信アプリケーションに対し，一般的に TCP プロトコルスタックは OS カーネル内の機能として提供される．そのため，通信アプリケーションは, ソケットと呼ばれる API(Application Programming Interface)を介して TCP の通信機能を利用する. TCP プロトコルスタックは, TCP 送信部と TCP

受信部があり，通信アプリケーションが送信するデータは，TCP 送信部で送信処理されるまで，送信ソケットバッファに蓄積される．TCP 送信部は，コネクション管理部，輻輳制御部から構成される．コネクション管理部は，通信相手との接続開始終了手続き，接続状態の管理を行い，輻輳制御部は，通信相手との間で輻輳が起きないよう，廃棄パケットの有無などに基づき送信レートの調整を行う．輻輳制御部における送信レートの調整を行うアルゴリズムは，輻輳制御アルゴリズムと呼ばれ，これまで様々なものが提案されてきた．TCP 送信部は，送信したパケットをそれに対する ACK(確認応答)を受信するまで再送バッファに格納する．TCP 受信部は，再送制御部と順序制御部から構成される．再送制御部は，通信相手から受信する ACK パケットのシーケンス番号抜けの検出などによりネットワークにおけるパケットの廃棄があったことを検出すると，確認応答されなかったパケットを再送バッファより取り出し再送する．順序制御部は，通信相手より受信したデータパケットのシーケンス番号を確認し，順序通りでない場合，一時的にリオーダリングバッファに格納しパケットを順番に受信するようにする．TCP 受信部は，正しい順序で受信したデータパケットを受信ソケットバッファに書き込み，通信アプリケーションに対し受信ソケットバッファのデータを読み出すように通知する．

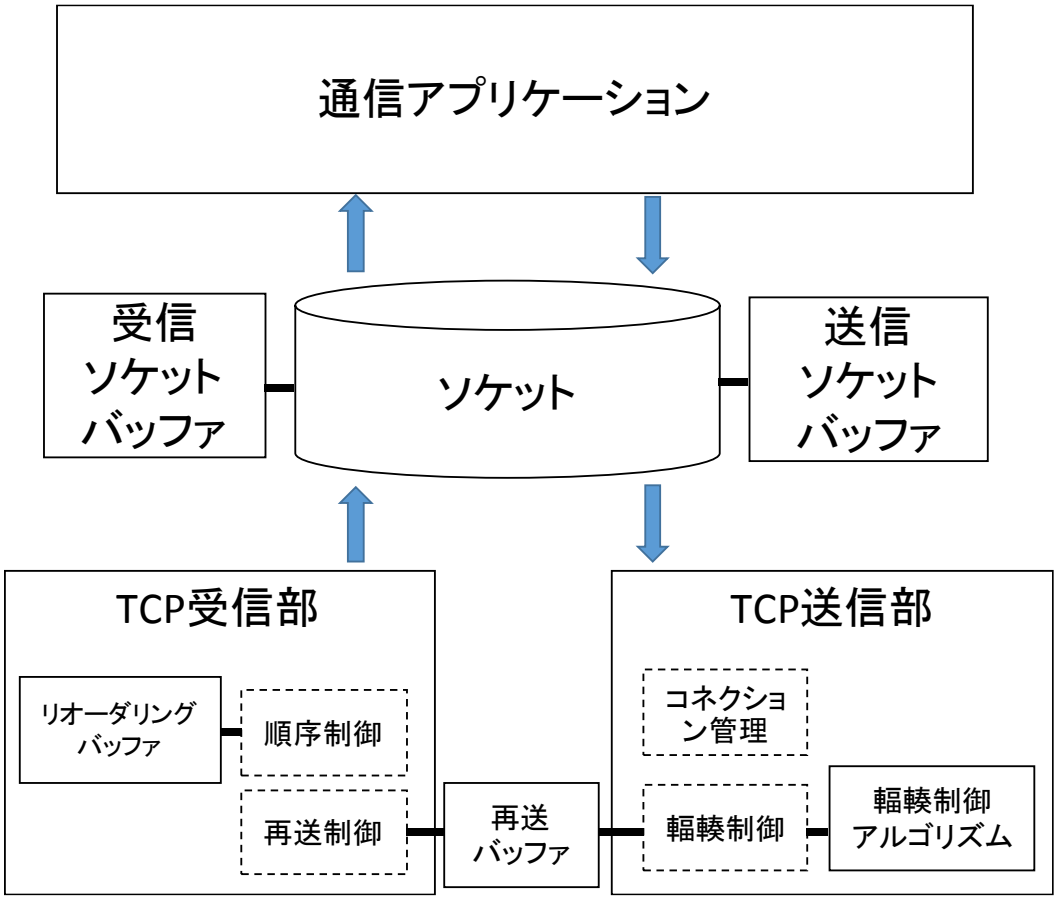


図 1-3 TCP のアーキテクチャ

1.4.1.1 ソケットインタフェースの役割

ソケットは，通信アプリケーションが，OS の提供する TCP の機能を利用するためのインタフェースであり，通信アプリケーションは，ソケットに対しシステムコールを実行することで，指定した通信相手に対し TCP コネクションを生成し，データを送信したり受信することができる．ソケットは，送信ソケットバッファと受信ソケットバッファを持つ．通信アプリケーションが送信ソケットバッファに一旦データを書き込み，TCP 送信部がネットワークの状態に応じた速度で，送信ソケットバッファからデータを取り出し，ネットワークへ送信する．ソケットバッファは，通信アプリケーションがソケットにデータを書き込む速度と TCP 送信部がデータを送信する速度の差を緩衝する役目を持つ．受信ソケットバッファは，TCP が受信したデータを一旦蓄積し，通信アプリケーションがこれらのデータを受け取るまで保持するためのバッファである．

1.4.1.2 コネクション管理

コネクション管理部は，通信相手との TCP コネクションの接続状態管理を行う機能ブロックである．

TCP は，図 1-4 に示す状態マシンによりコネクション状態を管理する．本状態マシンは，サーバ側としてソケットを開いて待ち受ける場合，クライアント側としてコネクションをはる場合の両方の状態遷移が合わせて示してある．以下では，サーバ側 TCP とクライアント側 TCP の代表的な状態遷移について説明する．

サーバとしての動作

通信アプリケーションがソケットを作成した時点では，状態マシンは CLOSED の状態から始まる．アプリケーションからのシステムコールにより受動的 OPEN が実行されると，LISTEN 状態に遷移する．クライアントから SYN パケットを受信すると SYN RCVD 状態へ遷移すると共にクライアントに対し SYN/ACK パケットを送信する．この SYN に対する ACK をクライアントから受信すると ESTAB 状態に遷移し，コネクションが確立する．

クライアントがコネクションを切断した場合は，クライアントから FIN パケットを受信する．TCP は CLOSE WAIT 状態へ遷移し，サーバから送信するデータが全てなくなった時点で CLOSE を行いクライアントに対し FIN を送信する．この FIN に対してクライアントより ACK を受信すると CLOSED 状態になりコネクションが解放される．

クライアントとしての動作

通信アプリケーションがソケットを作成した時点では，状態マシンは CLOSED の状態から始まる．アプリケーションからのシステムコールにより能動的 OPEN が実行されると，サーバに対し SYN パケットを送信し SYNSENT 状態に遷移する．サーバから SYN/ACK パケットを受信すると ESTAB 状態へ遷移すると共にサーバに対し ACK パケットを送信する．

クライアントがコネクションを切断した場合は，通信アプリケーションからの CLOSE システムコールを受けて FIN パケットを送信すると共に, FIN WAIT-1 状態へ遷移する．この FIN に対してサーバより ACK を受信すると FIN WAIT-2

へ遷移する．その後，サーバより FIN を受信すると TIME WAIT 状態へ遷移すると共に ACK を送信する．その後 2MSL 時間(最大セグメント生存時間; Maximum Segment Lifetime; 通常 2 分間に設定)が経過した後，CLOSED 状態になりコネクションが解放される．

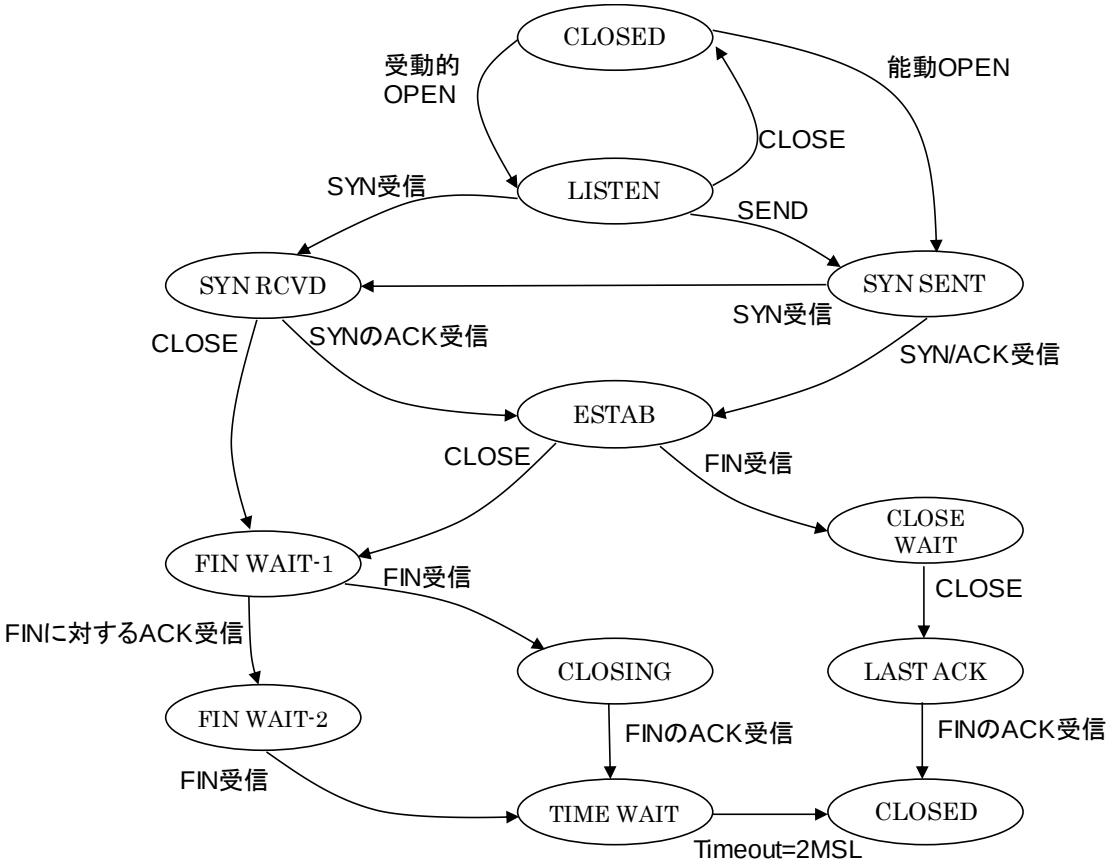


図 1-4 TCP のコネクション状態管理

1.4.1.3 TCP輻輳制御

TCP 輻輳制御部は，TCP の送信レートを制御する機能ブロックである．TCP 輻輳制御部は，始めスロースタートと呼ばれる動作モードで輻輳ウィンドウサイズを ssthresh (Slow Start Threshold)と呼ばれる閾値まで広げ, それ以降は輻輳回避モードと呼ばれる動作モードで輻輳ウィンドウサイズを広げる．この輻輳回避モードにおける輻輳ウィンドウサイズの広げ方の手順を輻輳制御アルゴリズムと呼んでいる．様々な輻輳制御アルゴリズムが提案されているが，一般に OS 内では設定により一つの輻輳制御アルゴリズムが用いられる．TCP の輻輳制御アルゴリズムは一般に以下に示す式により輻輳ウィンドウのサイズWを調整す

る．すなわち，TCP がデータパケットを送信し，それに対する ACK を受信する毎に式(1-1)に従い増加率 α 分，輻輳ウィンドウサイズを広げる．

$$W = W + \frac{\alpha}{W} \qquad (1\text{-}1)$$

一方で，TCP がネットワークの輻輳を検出した場合は，TCP は式(1-2)に従い減少率 β 分だけ輻輳ウィンドウサイズを縮小する．

$$W = \beta \cdot W \ (0 < \beta < 1) \qquad (1\text{-}2)$$

送信側 TCP は，ACK を受信するまではデータパケットの送信を最大でも輻輳ウィンドウサイズ内に収るように送信レートを制御し，送信したパケットについては，ACK を受信するまで再送バッファに保持する．

1.4.1.4 再送制御

再送制御部は，送信したデータパケットに対し受信側 TCP より ACK を受信すると，ACK されたシーケンス番号のデータパケットを再送バッファより削除する．一方で，TCP は以下に示す 3 つの場合に送信したパケットが廃棄されたと認識し再送を行う．

- 再送タイムアウト
- Fast Retransmit(三重重複 ACK を受信した場合)
- SACK によりパケット廃棄を通知された場合(SACK については後述)

再送タイムアウト

送信側 TCP は，送信したデータパケットに対する ACK を，RTT を基に計算する RTO(Retransmission Time Out)以内に受信できない場合，パケットが廃棄されたものと判断し，再送バッファに保持されている対象パケットを再送する．再送タイムアウトが発生した場合，TCP は輻輳ウィンドウサイズを初期ウィンドウサイズの値まで縮小する．

Fast Retransmit

三重重複 ACK（トリプル重複 ACK）は，図 1-5 に示すように，送信側 TCP が連続してデータパケットを送信し，そのうち一部のパケットが廃棄された場合(SEQ2500)に，受信側 TCP が廃棄されたパケットの直前のシーケンス番号

(ACK2499)を繰り返し ACK で通知する．送信側 TCP は ACK で同じシーケンス番号を 3 回通知されると輻輳と認識し，再送を行う．

この場合は, 送信側 TCP は輻輳ウィンドウサイズを輻輳制御アルゴリズム毎に決まっている割合で，輻輳検出前の輻輳ウィンドウサイズから引き下げる．

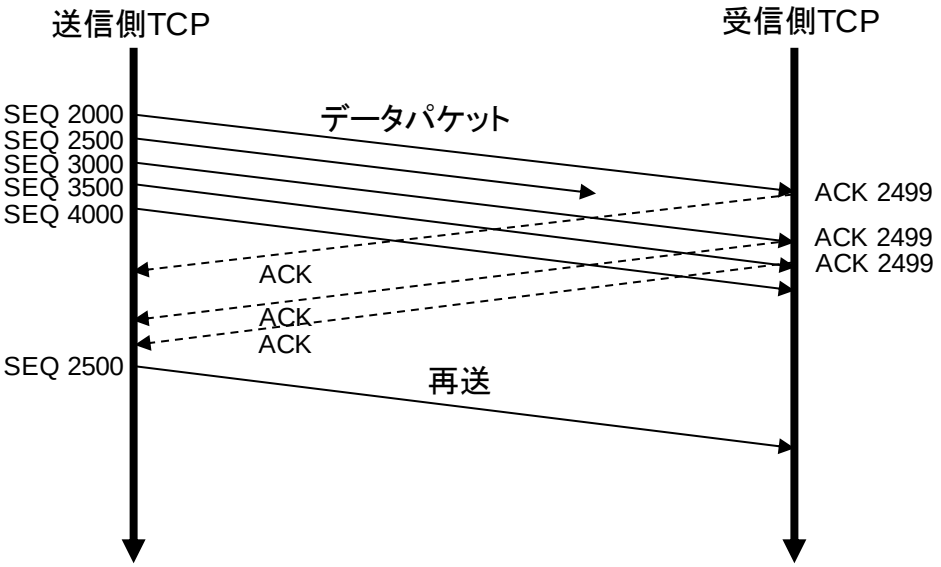


図 1-5 Fast Retransmission

SACK によるパケット廃棄の通知

既存の ACK は廃棄されたパケットの直前のシーケンス番号を ACK として伝える．具体的にどのシーケンス番号のパケットが廃棄されたかを通知できないため，廃棄パケット以降が全て再送となる．Selective ACK (SACK) 機能[24][46]は，廃棄されたシーケンス番号を直接通知できるため，廃棄されたパケットのみを効率的に再送することができる．TCP のコネクション開設時のネゴシエーション時に SACK 対応を通知するオプションを設定することで利用でき，今日の多くの OS では SACK が使えるようになっている．受信側 TCP は，連続してデータパケットの廃棄を検出した場合は，SACK により通常の ACK に加え，SACK option フィールドを使って, 廃棄パケット以外のどのセグメントが到着したかを送信側 TCP へ通知する．SACK option フィールドは，図 1-6 に示すように，到着シーケンスブロックの左端(LeftEdge)と右端(RightEdge)で表現され，最大 4 ブロックまで指定することが可能である．送信側 TCP は，SACK により通知された廃棄パケットを再度送信すると共に輻輳ウィンドウサイズを輻輳制御アルゴリズムで決められた割合まで縮小する．

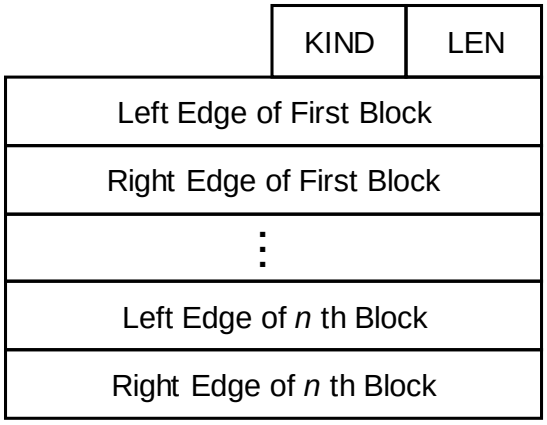


図 1-6 SACK オプション

1.4.1.5 順序制御

順序制御部は，受信したデータパケットに付与されているシーケンス番号に基づき，データの順番を並べ直す機能ブロックである．一般に TCP セグメントを転送する IP パケットは順序保証を行わない．そのため，TCP が順序制御部を用いて，データの到着順を保証している．具体的には，パケットに付与されたシーケンス番号を監視しており，シーケンス番号が連続しないパケットを受信した場合は，一旦リオーダリングバッファに蓄積する．正しい順序で受信できたデータは，受信ソケットバッファに書き込まれる．

1.4.2 DaaS 応答性能劣化の通信に起因する要因

端末とクラウドにおけるサーバ間で TCP 通信の性能が上がらない要因として以下が考えられる．

- 遅延や廃棄に対する TCP 輻輳制御の影響
- ネットワークの帯域が狭いことによる影響
- アプリケーションデータの少なさによる性能低下

1.4.2.1 遅延や廃棄に対するTCP輻輳制御の影響

今日様々なワイヤレスブロードバンド技術が開発され，ユーザはモバイル端末を外に持ち出し，様々な場所から LTE(Long Term Evolution) や WiMAX(Worldwide Interoperability for Microwave Access)といった無線アクセス網を介してネットワークに接続可能である．また，クラウドサービスは様々な国

や地域で提供されており，クラウドを提供するデータセンタが国外にあることも少なくない．この場合，クラウドサービスをアクセスするためには大きな遅延時間を持つ国際回線などを介して通信を行う．

無線アクセス網では，端末が移動しない場合であっても，フェージング，シャドーイング，干渉などの影響により廃棄が発生しやすい．近年の無線アクセス技術では，HARQ[41]といった無線レイヤにおける誤り訂正技術が備わっており，TCP が廃棄を検出しないようパケット廃棄を回復させることができるが，再送を繰り返す分だけ遅延時間が積み重なり TCP からは RTT が増加したように見える．そのため，電波強度に応じ RTT が変動しやすいという特性を持つ．また，非常に多くのパケットが廃棄された場合は，無線レイヤにおける誤り訂正で全てのパケット廃棄を再送しきれず，TCP にはパケット廃棄として見える．一方，端末が移動する場合は，基地局からの距離の変化，フェージング，干渉などにより廃棄や遅延の特性が大きく変化する．

国際回線は，大陸間に跨がって敷設され，パケット廃棄はそれほどないが，遅延が大きいという特徴を持つ．日本から米国間でおよそ 200ms，日本から欧州間で 300ms 程度の RTT を持つ．

モバイル端末から国外のクラウドにアクセスする場合，無線アクセス網，国際回線を介して通信するため，TCP は，廃棄が起きやすく，遅延が大きいネットワークを介して通信していることになる．

大容量のファイル転送を行うアプリケーション(例えば FTP)は，TCP が利用可能な帯域まで輻輳ウィンドウサイズを広げるのに十分な量の送信データを，TCP のパケット送信速度よりも速い速度でアプリケーションと TCP 間のバッファに連続的に書き込む．データ送信側の TCP は，ACK を受信することで，輻輳ウィンドウサイズを輻輳制御アルゴリズムの設計上の最大速度で広げる．TCP は，一般に送信したデータパケットに対する ACK パケットを受信する毎に輻輳制御アルゴリズムに基づき，輻輳ウィンドウサイズを拡大する．そのため，RTT が大きいほど輻輳ウィンドウサイズの広がりが緩やかになる．さらに，TCP は廃棄が発生すると輻輳ウィンドウサイズを一旦縮小し再び広げる動作を行う．図 1-7 に示すように，RTT が小さい場合は，廃棄があってもネットワークのボトルネックリンクの速度まですぐに輻輳ウィンドウサイズを広げることができるが，

RTT が大きい場合は，輻輳ウィンドウサイズが帯域に対して十分広がる前に輻輳が発生し，結果として，ネットワークの帯域があるにもかかわらずスループットが上がらない状態が発生する．

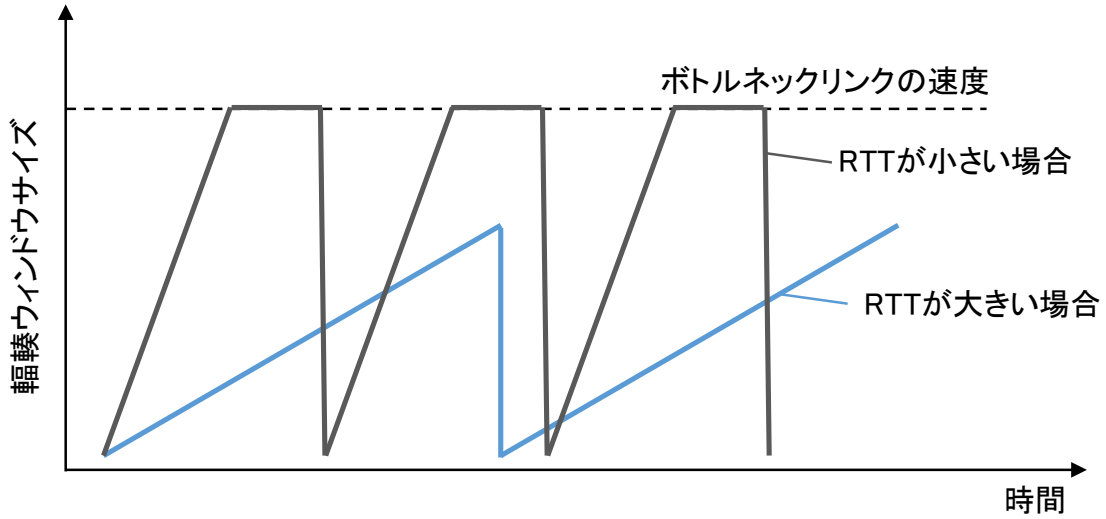


図 1-7 RTT が大きいネットワークでの輻輳ウィンドウの広がり

1.4.2.2 ネットワーク帯域の狭さによる性能ボトルネック

無線アクセス網などに使われる LTE や WiMAX といった無線通信技術では，適応変調と呼ばれる技術により，電波の強さに応じ，帯域の広さを動的に変える．送信側 TCP は，ボトルネックとなるリンクの速度に輻輳ウィンドウサイズを合わせる．そのため，こうしたリンクが通信パス上に含まれるとそれ以上スループットがあがらない．

1.4.2.3 アプリケーションデータの少なさによる性能低下

通信アプリケーションは，図 1-3 に示したようにソケットバッファを介して TCP にデータを渡すが，アプリケーションがソケットバッファにデータを書き込む速度と TCP プロトコルがソケットバッファからデータを取り出しネットワークに送信する速度は一致しない．インタラクティブ通信を行うアプリケーションは，比較的小容量のデータを断続的に送信しレスポンス時間を重視するアプリケーションであり，特にサーバからの応答を待って次のデータを送るクライアントアプリケーションの場合，データ送信速度が，TCP のプロトコルレイヤの送信速度を下回る．

TCP のパケット送信速度に比べゆっくりと非連続にデータを送信するアプリ

ケーションは，図 1-8 に示すように，輻輳ウィンドウサイズの広がりがアプリケーションのデータ送信速度に支配され緩やかになる．図中①は，送信データが十分ある場合に TCP が輻輳ウィンドウサイズを広げる設計上の広げ方を示している．しかし，インタラクティブ通信では，実際には，アプリケーションは相手からの応答を待って間欠的にしかデータを送信しないため，②に示すように輻輳ウィンドウサイズが広がる．その結果，インタラクティブ通信時の輻輳ウィンドウサイズの成長速度は③に示すようになる．

特に仮想デスクトップのように端末の操作に対しサーバが応答データを送信するインタラクティブアプリケーションは，送信データ量が少ないため TCP の輻輳ウィンドウサイズが広がるのに時間がかかり，小さい輻輳ウィンドウサイズで送信し ACK を待って更に送ることを繰り返すため，応答性能が低下するという問題がある．

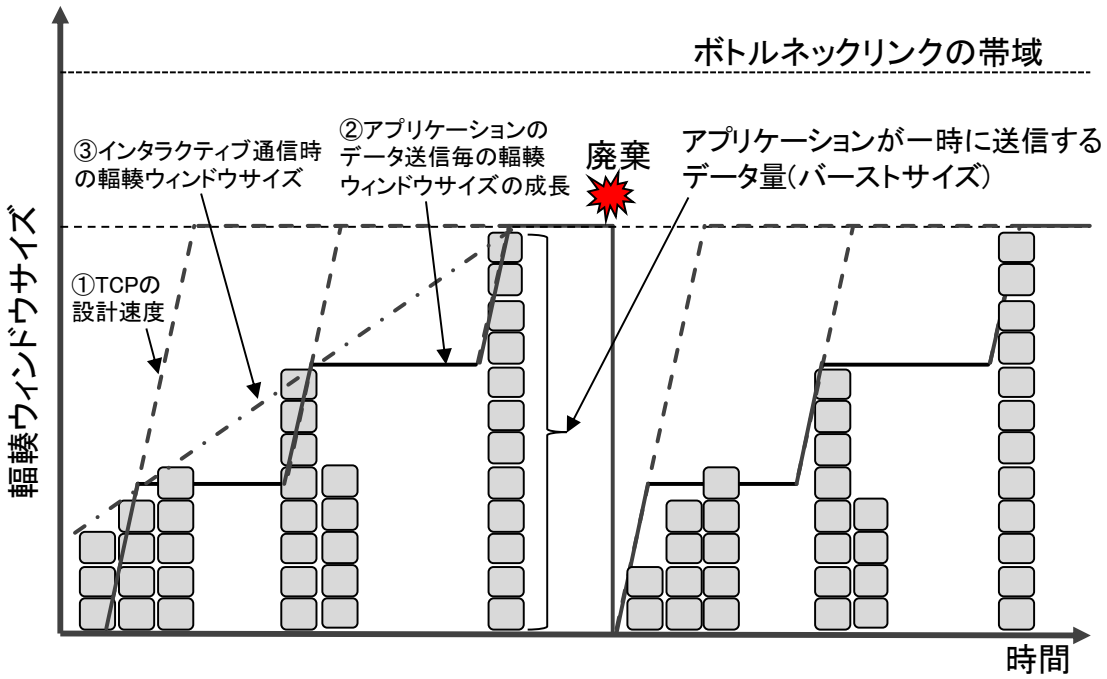


図 1-8 非連続データ転送アプリ利用時の輻輳ウィンドウサイズの変化

このような原理によりモバイル端末からクラウドとの間で TCP を用いてインタラクティブ通信を行う場合，応答時間が長くなりやすいという問題がある．

1.4.3　理想的な輻輳制御

インタラクティブ通信のレイテンシを改善するには，理想的には図 1-9 に示すような輻輳制御が良い．すなわち，輻輳ウィンドウサイズを広げる際には，ネットワーク内のボトルネックとなる通信速度に相当するサイズ，又はアプリケーションが一時に連続送信するデータ(バーストデータ)に相当するサイズのうち小さいサイズまでなるべく速く広げ，輻輳が発生した際には，輻輳ウィンドウサイズをあまり下げないようにし，バーストデータのレイテンシをなるべく短くするのが良い．

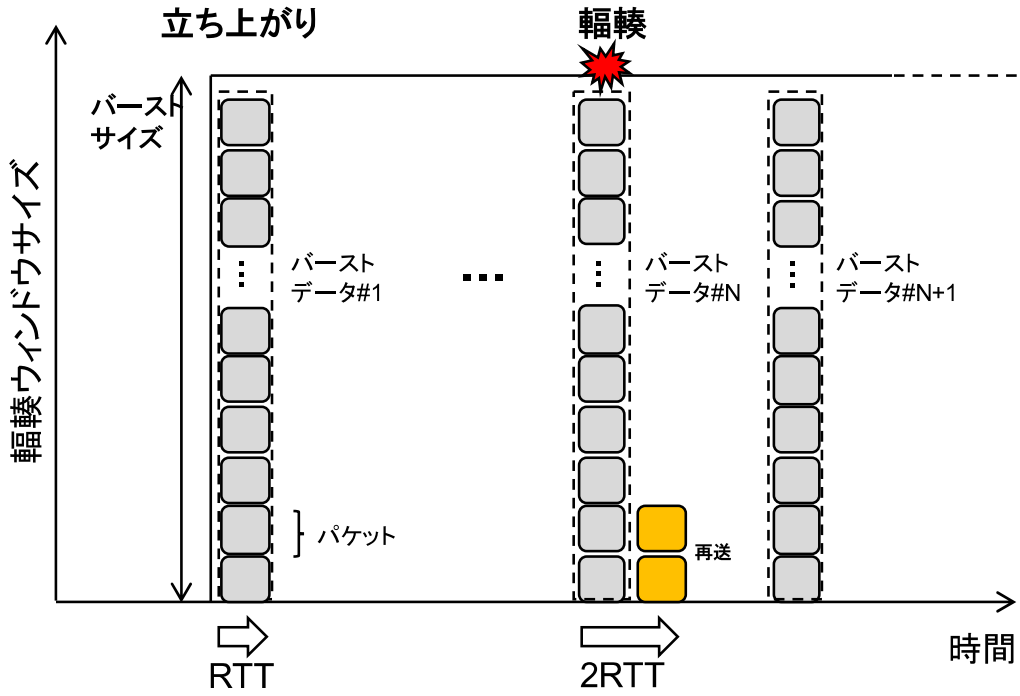


図 1-9 理想的な輻輳制御

1.5　本研究が取り組む課題と目的

本研究では，前節であげたインタラクティブ通信における TCP の応答時間劣化の課題に取り組むと共に，ユーザ毎に異なる多様な通信環境に対し，ユーザが意識することなく応答時間改善の解決策を適用できるメカニズムの実現を目的とする．

1.5.1 インタラクティブ通信時の TCP レスポンスの改善

仮想デスクトップを実現するには RDP (Remote Desktop Protocol)[4]や RFB (Remote Frame Buffer) Protocol[5]などのアプリケーションレイヤのプロトコルが用いられる．これらはトランスポートレイヤのプロトコルとして TCP プロトコルを用いて通信している場合がほとんどである．ところが TCP プロトコルは，1.4.2.1 節のように遅延が大きいネットワークでパケット廃棄が発生すると，通信性能が低下しやすいという特性を持っている．特に人間が遠隔の PC を操作する仮想デスクトップのようなインタラクティブなアプリケーションは，単位時間における転送データ量はそれほど多くないため，スループット性能は要求しないが，高い応答性能を必要とする．

仮想デスクトップクライアントとサーバのシーケンスを TCP プロトコルのレイヤでみると図 1-10 のようになる．クライアントからサーバへ送る RDP のメッセージにはマウス操作やキーボード入力などのコマンドが含まれるが一般的にこれらのサイズは小さいので，数個の TCP パケットとしてクライアントからサーバへ転送される．このメッセージをサーバへ送信するのに要する時間を**送信レイテンシ**と定義する．数個の TCP パケットの転送のため，平均的な送信レイテンシはネットワークの遅延時間とほぼ同じ程度である．

一方，サーバが RDP により送信するデスクトップ画面更新データは，最大 64Kbytes のデータサイズを持つため，ネットワークの MTU(Maximum Transfer Unit)長にもよるが，多くの場合複数の TCP パケットに分割されて送信される．デスクトップ画面更新データを転送するのに要する時間を**受信レイテンシ**と呼ぶこととする．受信レイテンシは TCP の輻輳制御の影響を大きく受け，ネットワークの遅延時間の何倍にもなる可能性がある．

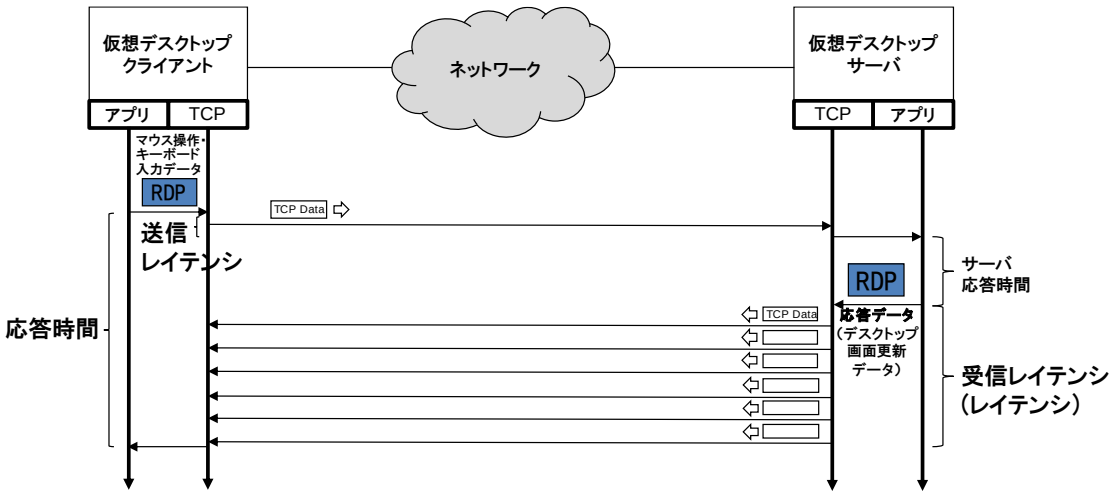


図 1-10 DaaS における TCP 性能の改善

仮想デスクトップクライアントが，キーボードなどを操作し，それが反映されてクライアント画面が更新されるまでの時間を**応答時間**とすると，応答時間は，送信レイテンシと，サーバが更新画面を計算しデスクトップ画面更新データを送信するのに要する時間であるサーバ応答時間，及び受信レイテンシを足した時間になる．サーバ応答時間は一般的には，ネットワークの遅延時間に比べ十分小さいと考えられるため，応答時間の支配項となる受信レイテンシを改善できれば応答時間を改善できる．そのため，本研究では，TCP の性能を改善し，受信レイテンシを改善することを目的とする．以降では受信レイテンシのことを単に**レイテンシ**と呼ぶ.

1.6 課題解決に向けた要件

1.6.1 ユーザにネットワークを意識させないインタラクティブ通信性能の改善

端末やサーバは，様々な位置にある通信相手と，遅延時間や廃棄率といった特性の異なるネットワークを経由し通信を行う.

クライアント端末は，図 1-11 に示すように同一拠点内にある社内サーバや，インタネットを介して海外のクラウドにあるサーバと同時に通信することが考えられる．例えば，海外のクラウドにあるサーバからストリーミング映像を表示しながら，社内サーバにアクセスし，ドキュメントを参照する場面が考えられる.

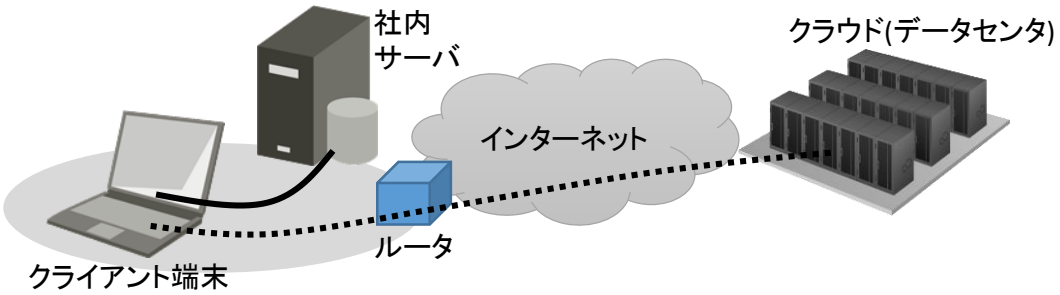


図 1-11 クライアント端末の通信先

一方，図 1-12 に示すように，サーバは，様々なデバイスからの接続を同時に受け付け，データを送信する．これらのデバイスの通信パスは千差万別で，その通信路の特性も異なる．インタネットを介し海外の拠点の端末にデータを送る場合，比較的遅延が大きく帯域が広い Long Fat Network(LFN)を経由して通信が行われることがある．さらに，航空機におけるインタネットサービスは，人工衛星を介して地上と通信し，サーバと通信を行う．こうした回線は，極端に大きな遅延を持ち，廃棄率も大きい．また，高速に移動する車両に対しては無線網を介した通信が行われる．この場合は，移動に伴う電波強度の変化や，無線基地局を切替えるハンドオーバなどが発生し，遅延や廃棄率が大きく変化する．移動が比較的緩やかなモバイルデバイスでも，使用環境によりビルなどのフェージングの影響や，他のモバイルデバイスとの干渉などにより遅延や廃棄率が大きく変化する．また，電波の状況に応じ変調方式やコーディングシステムを動的に切替え無線信号の復号化率を改善する適応変調と呼ばれる技術を用いると，帯域が動的に変動する．

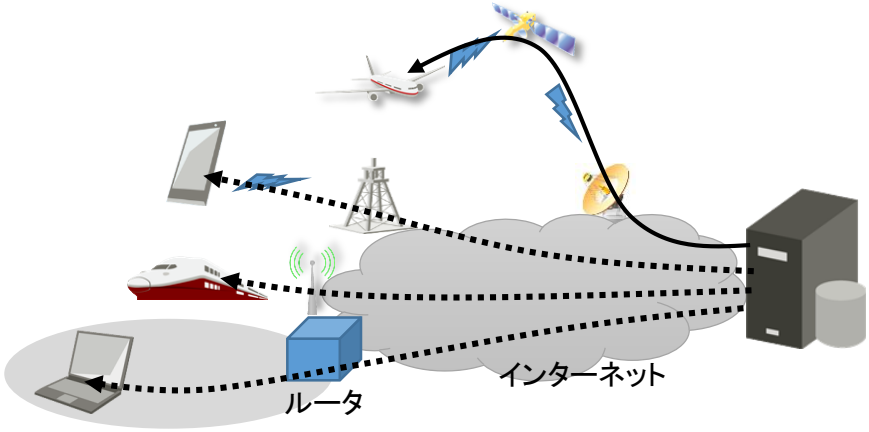


図 1-12 サーバ装置における通信先

このように，端末もサーバも様々な特性を持つネットワークの先にいる通信相手と同時に通信を行うことが求められている．また，それらの特性は時間と共に変わり得る．

TCP を用いた通信において，最適な通信環境を得るためには，ユーザがネットワークの特性を調べ，最適なチューニング方法を調べ，OS の設定を変更する必要がある．しかし，ネットワークの特性を調べるためには，ネットワークの専門的な知識が必要である．また，特性に合わせた TCP のチューニングを行うためには TCP に関する深い知識が必要である．さらに OS 毎に TCP のチューニングも異なり OS に関する専門的知識も必要となる．ネットワークの専門家であっても，利用する場面毎に通信性能を測定し適切に対処するには非常に手間がかかり，一般のユーザにとっては非現実的である．

仮にユーザによる TCP のチューニングがなされたとしても，今日の OS は，OS 全体で単一の輻輳制御アルゴリズムしか選ぶことができない．図 1-13 に示すように，端末やサーバの OS 上では，複数の通信アプリケーションが動作し，通信アプリケーション毎にソケットと呼ばれるインタフェース(API)が用意される．OS カーネルには，ユーザが一つに指定した輻輳制御アルゴリズムを含む TCP の処理部があり，ソケット毎に TCP を処理するインスタンスを構成する．そのため，全てのインスタンスは，同じ輻輳制御アルゴリズムに従う．特に，今日，ユーザが一つの OS で複数のアプリケーションを動かし，複数の通信相手と同時に通信を行うのが一般的になってきているため，各セッションが経由する通信パスの遅延や廃棄率に応じて，異なる輻輳制御アルゴリズムを用いた方が良い場合が考えられるが，実際には全てのソケットが，OS のカーネルパラメタに設定された輻輳制御アルゴリズムに固定されてしまう．そのため，各通信相手が，全く特性が異なるネットワークを経由する場合に，全ての通信相手に対し同時に最適となるような輻輳制御アルゴリズムを選ぶことは困難である．

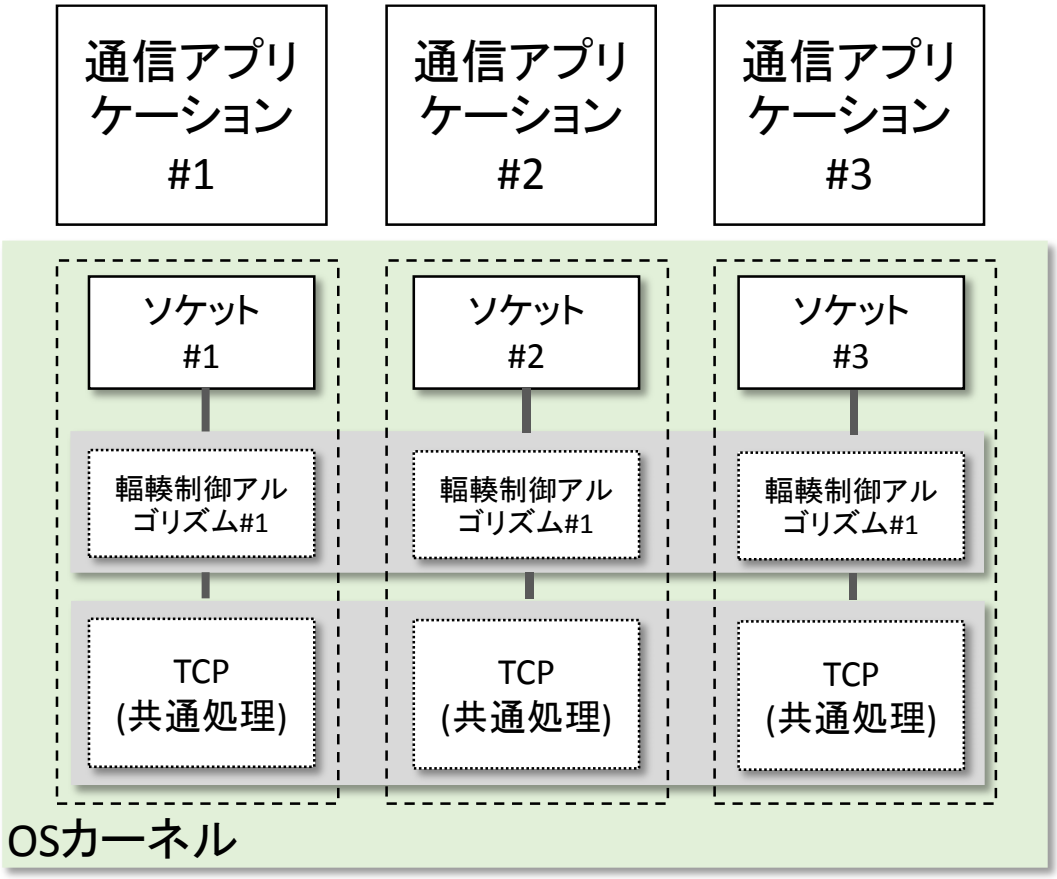


図 1-13 OS カーネルにおける TCP の構造

1.6.2 ネットワークの特性の変化に適応した通信性能の改善

ネットワークの特性は，時々刻々変化している．特に無線網は，移動デバイスと基地局との距離や，障害物，移動速度や他のデバイスによる電波の干渉などにより様々な影響を受ける. そして, 特性の変化は TCP パケットの廃棄や RTT の変化という形で現れ，TCP の性能に影響を与える．

また，有線のネットワークであっても昼と夜とでネットワークの利用状況は異なり，ネットワークサービスプロバイダによっては，ネットワークが混雑している場合には，通信セッション毎に帯域に制約を加えることもある．

そこで，こうしたネットワークの特性の変化に適応しながら，通信性能を改善する必要がある．

1.7　本論文の構成

以降の章は，次に示す内容となっている．

第 2 章では，関連研究及び動向について説明する．従来検討されてきた TCP の性能を改善するための様々な技術について説明する．特に標準 TCP とされる TCP Reno を改良した改良型輻輳制御アルゴリズム，無線におけるパケット廃棄とネットワークの輻輳を区別することで不用意なウィンドウサイズの縮小を回避する Loss Differentiation Algorithms (LDA)技術，無線網におけるパケット廃棄を効果的に再送する Snoop TCP 技術，インタラクティブ通信を高速化するための TCP オプション設定のテクニック，無線レイヤにおけるエラー訂正技術を説明する．さらに最適な輻輳制御アルゴリズムをコンピュータで設計する技術や，通信中に輻輳制御アルゴリズムを動的に切り替える技術について説明する．しかし，これら技術は，特定の環境，特定の用途に向けた対処であり，環境が変化するモバイル環境，クラウド環境に適合しないことを示す．

こうした課題に対し，通信環境の変化に応じ動的に輻輳制御アルゴリズムを切替える手法が有効と考えられ，第 3 章では，各輻輳制御アルゴリズムの廃棄や遅延に対する各輻輳制御アルゴリズムが優位となる適用領域を明らかにする．

第 4 章では，輻輳制御アルゴリズム動的切り替え手法について説明する．様々な遅延時間，パケット廃棄パタンが混在するネットワークにおいてレイテンシを改善する制御方法を提案し，その実装について説明する．また，提案方式の公平性についての検討も行う．

第 5 章では，シミュレーションによる性能評価について説明する．本章では，インタラクティブ通信を行うアプリケーションを想定し，提案方式を，GE モデル及び WiMAX を用いたシミュレーションモデルで評価する．

第 6 章では，LTE 網における実機評価について説明する．提案方式を Linux に実装し，実際の LTE 網を用いた提案方式の性能評価を行う．更に，提案方式によるインタラクティブ通信の応答性能の改善が，どの程度の主観評価の改善につながるかを考察する．

最後に，第 7 章では，本論文で明らかになったことをまとめ，提案方式の有効性を示すと共に，今後の課題を整理する．

1.8　本章のまとめ

本研究の背景として，PC の紛失や盗難による情報漏洩が大きな問題となっておりクラウド上の仮想デスクトップサービスが期待されているが，技術的問題により未だ普及していないことを示した.

次に, 本論文の提案に必要な知識として TCP の構造的な仕組みを示すと共に, TCP が様々な通信特性を持つネットワークで性能劣化するメカニズムを示した.

従来 LFN や無線ネットワークといったネットワークの特性毎に通信性能を改善する様々な輻輳制御アルゴリズムが提案されてきたが，ユーザはネットワークの特性を調べ最適な輻輳制御アルゴリズムを選択し切り替えることが困難であり，また，今日の OS では OS 全体として一つの輻輳制御アルゴリズムを指定することしかできないため，通信相手が異なったり，通信アプリケーションの種類が異なるものが同時に通信をすると最適な輻輳制御アルゴリズムで通信できないことを示した．こうした課題を鑑み，本論文では，ユーザがどのような環境において，どのような接続先と通信を行っていても，ユーザに意識させることなく，レイテンシを改善することを目的とすることを示した.

第2章 関連研究

本研究の目的である，ネットワークの様々な通信特性に応じた TCP 通信のレイテンシ改善に関する技術として，本章では，様々な通信環境に向けた TCP の通信性能を改善する技術や，TCP の性能を改善するよう通信環境に動的に対応する技術について説明する．

2.1 TCP の輻輳制御アルゴリズムを改善するアプローチ

無線アクセス網や，LFN といった様々な通信環境に合わせて通信性能が出やすくなるように最適設計された輻輳制御アルゴリズムが各種提案されている．本節では，これらの輻輳制御アルゴリズムについて述べる．

2.1.1 Highspeed TCP

Highspeed TCP[58]は，LFN 向けに開発された輻輳制御アルゴリズムである．標準的な輻輳制御アルゴリズムである Reno が，帯域が広い LFN では輻輳ウィンドウを広げるのに非常に時間がかかることを鑑み，帯域の幅に応じて，増加率αと，減少率βを変化させる．ACK を受信する毎に式(2-1)に基づき輻輳ウィンドウサイズを拡大する．また，パケット廃棄を検出すると式(2-2)に基づき輻輳ウィンドウ($cwnd$)を縮小する. 式中の⟻記号は, 代入を意味する. ここで, 表 2-1 は，ネットワークの帯域wに対する増加率αと，減少率βを示しており，式(2-1), 式(2-2)は，ネットワークの帯域wにより挙動が変化する．

$$cwnd \longleftarrow cwnd + \frac{\alpha}{cwnd} \qquad (2\text{-}1)$$

$$cwnd \longleftarrow cwnd - \beta \times cwnd \qquad (2\text{-}2)$$

式(2-1)から，輻輳ウィンドウサイズは，ACK の受信数に応じて広がることがわかる．

表 2-1 Highspeed TCP の増加率αと減少率β

ネットワークの帯域w	$\alpha(w)$	$\beta(w)$
38	1	0.50
118	2	0.44
221	3	0.41
347	4	0.38
495	5	0.37
663	6	0.35
851	7	0.34
1058	8	0.33
1284	9	0.32
1529	10	0.31
1793	11	0.30
2076	12	0.29
2378	13	0.28

2.1.2 Scalable TCP

Scalable TCP[61]は，LFN 向けに開発された輻輳制御アルゴリズムである．輻輳ウィンドウサイズ($cwnd$)は，式(2-3)に示すように，$cwnd$の値によらず ACK を 1 つ受信する毎に 0.01 増加する(増加率$\alpha = 0.01$)．そのため，$cwnd$が大きくなるほど RTT 内で送信するパケット数が多くなり，それに対する ACK の数も多くなるため，$cwnd$の成長速度が大きくなる．輻輳を検出した際は，式(2-4)に示すように，減少率 β=0.125 で減少する．

Scalable TCP は, 図 2-1 に示すように, 現在の輻輳ウィンドウサイズに応じて, 単位時間あたりの$cwnd$の増加率が大きくなり，Highspeed TCP のように表を使わずとも同様の効果が得られる．図中，c及びCはネットワークのボトルネック帯域に相当する$cwnd$の値であり$c < C$とする．また，どのような RTT であっても，パケット廃棄検出後に同じ時間($-\log(1-\beta)/\log(1+\alpha)$)で元の輻輳ウィンドウサイズまで回復するように設計されている．

$$cwnd \leftarrow cwnd + 0.01 \qquad (2\text{-}3)$$

$$cwnd \leftarrow cwnd - (0.125 \times cwnd) \qquad (2\text{-}4)$$

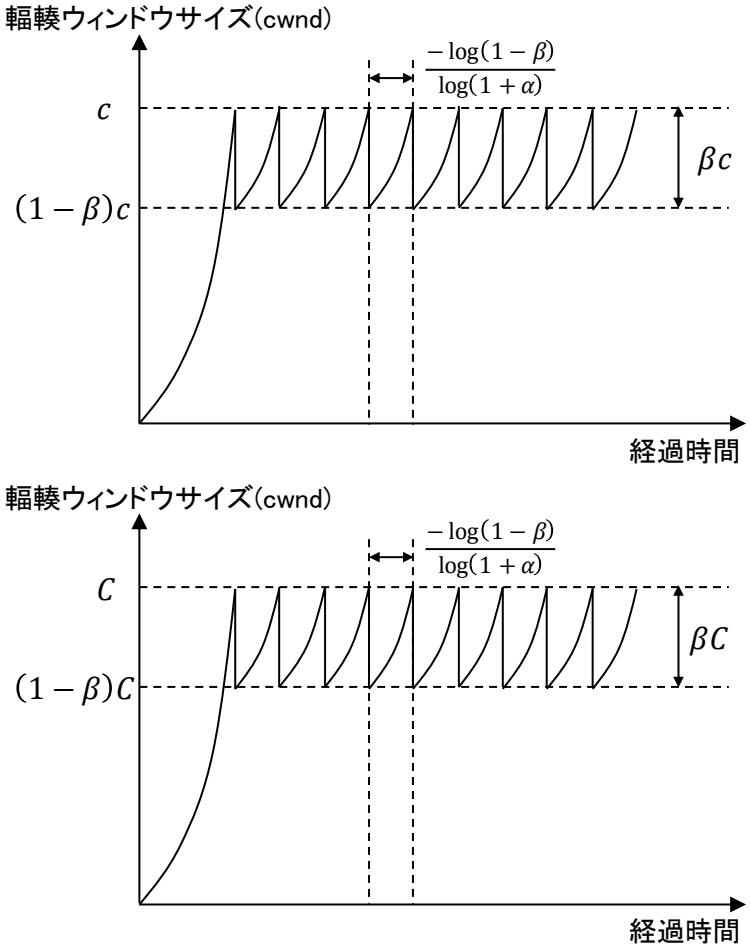


図 2-1 Scalable TCP の輻輳ウィンドウ制御

2.1.3 CUBIC TCP

CUBIC TCP[27]は，LFN 向けに開発された輻輳制御アルゴリズムであり，Linux2.6 以降，標準の輻輳制御アルゴリズムとして使われている．輻輳を検出する直前の輻輳ウィンドウサイズをW_{max}とすると，輻輳回避モードにおける輻輳ウィンドウサイズは式(2-5)に従い計算される．Linux OS においては，$C = 0.4, \beta = 0.3$を採用している．式(2-5)からわかるように，輻輳ウィンドウサイズは輻輳検出からの経過時間tの関数となっている点が特徴である．輻輳を検出すると，式(2-6)に従い輻輳ウィンドウサイズを縮小する．ここで用いるβは式(2-5)と同じ値で$\beta = 0.3$である．図 2-2 に示すように輻輳検出時刻を$t = 0$として，その後，輻輳を検出しない限り，輻輳ウィンドウサイズがW_{max}になるまで式(2-5)に従い輻輳ウィンドウサイズを広げていく(本動作をしている状態を Steady State Behavior という)．また，輻輳ウィンドウサイズがW_{max}に到達した後も，輻輳を検出しない場合は，式(2-5)に従い輻輳ウィンドウサイズを徐々に増加していく(本動作を

している状態を Max Probing という).

$$\left.\begin{aligned} cwnd &\longleftarrow C(t-K)^3 + w_{max} \\ K &= \sqrt[3]{w_{max}\,\beta/C} \end{aligned}\right\} \tag{2-5}$$

$$cwnd \longleftarrow \beta w_{max} \tag{2-6}$$

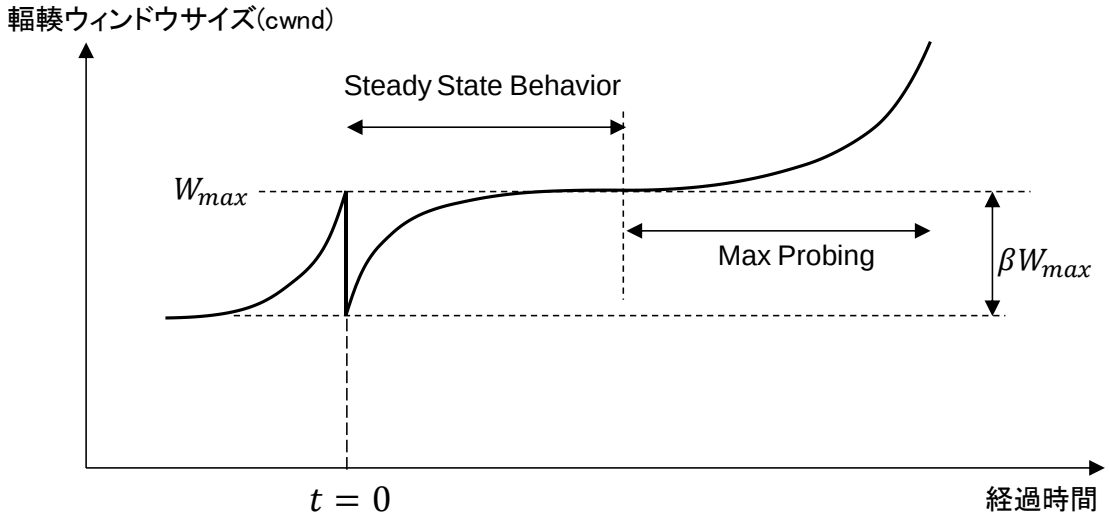


図 2-2 CUBIC TCP の輻輳ウィンドウサイズの変化

2.1.4 BIC TCP

BIC TCP[28]は，LFN 向けに開発された輻輳制御アルゴリズムである．輻輳ウィンドウサイズが，最後のパケット廃棄を検出した際の輻輳ウィンドウサイズ(W_{max})より小さい場合は，式(2-7)に示すバイナリサーチに基づきウィンドウサイズを広げる．輻輳ウィンドウサイズがW_{max}よりも大きい場合は式(2-8)に従い輻輳ウィンドウサイズを広げる．また，パケット廃棄を検出した際には，式(2-9)に従い輻輳ウィンドウサイズを縮小する．図 2-3 を用いてこれらの式に示した動作を説明する．以下では，w_iを現在の輻輳ウィンドウサイズ，w_{i+1}を RTT 経過後の輻輳ウィンドウサイズとする．はじめに輻輳ウィンドウサイズがW_{max}まで広がり輻輳を検出すると，式(2-9)に従い輻輳ウィンドウサイズを縮小する．

$w_i \leq W_{max}$の場合，輻輳ウィンドウサイズが縮小した直後に$(W_{max} - w_i)/2 > S_{max}$となっている場合は，RTT 毎に$S_{max}$づつ増加させていく．輻輳ウィンドウ

サイズが増加し，$S_{max} > (W_{max} - w_i)/2 > S_{min}$となっている区間では，$(W_{max} - W_i)/2$づつ増加させながら$W_{max}$に近づけていく．$S_{min} > (W_{max} - w_i)/2$となる場合は，RTT 毎に$S_{min}$づつ増加させていく．

また，$w_i > W_{max}$の場合は，新しいW_{max}を探すために，$W_{max} + S_{min}$，$W_{max} + 2S_{min}$，$W_{max} + 4S_{min}$…という具合に RTT 毎にS_{min}を倍増させていく．そして，$2^i S_{min}$がS_{max}に達すると以降は RTT 毎にS_{max}づつ増加させていく．

$w_i \leq W_{max}$の場合

$$\left.\begin{aligned} &w_{i+1} = w_i + \Delta \ (\text{for each RTT}) \\ &\Delta = \begin{cases} \dfrac{W_{max} - w_i}{2} \ , for \ \left(S_{max} > \dfrac{W_{max} - w_i}{2} > S_{min}\right) \\ S_{max} \ , for \ \left(\dfrac{W_{max} - w_i}{2} > S_{max}\right) \\ S_{min} \ , for \ \left(\dfrac{W_{max} - w_i}{2} < S_{min}\right) \end{cases} \end{aligned}\right\} \quad (2\text{-}7)$$

$w_i > W_{max}$の場合

$$\left.\begin{aligned} &w_{i+1} = W_{max} + 2^i S_{min} \ , for \ \left(2^i S_{min} < S_{max}\right) \\ &w_{i+1} = w_i + S_{max} \ , for \ \left(2^i S_{min} \geq S_{max}\right) \end{aligned}\right\} \quad (2\text{-}8)$$

$$\left.\begin{aligned} &cwnd \longleftarrow cwnd \times (1 - \beta) \\ &\beta = 0.5 \end{aligned}\right\} \quad (2\text{-}9)$$

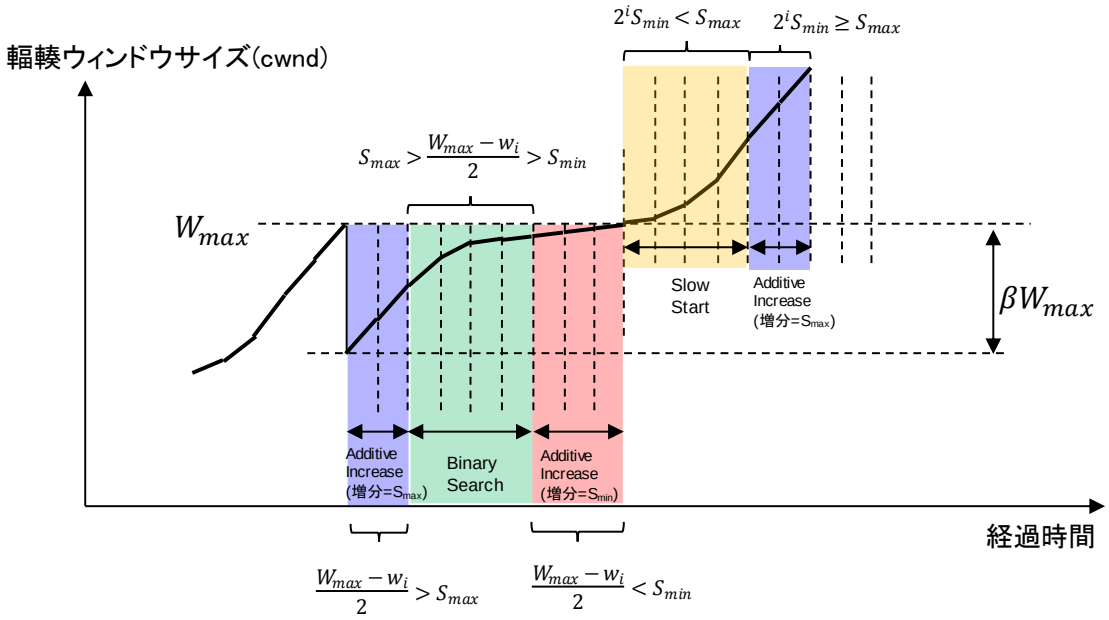


図 2-3 BIC TCP の輻輳ウィンドウサイズの変化

2.1.5 TCP Hybla

TCP Hybla は衛星回線などの RTT が大きいネットワーク向けに設計された輻輳制御アルゴリズムである．TCP Hybla は輻輳ウィンドウサイズを式(2-10)に従って広げる．式(2-10)は，輻輳ウィンドウサイズが RTT の大きさにより輻輳ウィンドウサイズの増加率が変化することを示している．一方で，パケット廃棄が発生した場合は，式(2-11)に従い輻輳ウィンドウサイズを縮小する．

図 2-4 を用いてこれらの式の動作を説明する．はじめに輻輳ウィンドウサイズがCまで広がったところで輻輳を検出すると，式(2-11)に従い輻輳ウィンドウサイズを半分の大きさに縮小する．式(2-10)は，観測した RTT の値RTTの，基準となる RTT の値であるRTT_0に対する比率により増加率ρ^2が変化することを示しており，RTTが大きい程，輻輳ウィンドウが速く広がるようになる．

$$\left.\begin{aligned} cwnd &\leftarrow cwnd + \frac{\rho^2}{cwnd} \\ \rho &= \frac{RTT}{RTT_0} \\ RTT_0 &= 25ms \end{aligned}\right\} \qquad (2\text{-}10)$$

$$\left.\begin{array}{l} cwnd \longleftarrow cwnd \times (1-\beta) \\ \beta = 0.5 \end{array}\right\} \qquad (2\text{-}11)$$

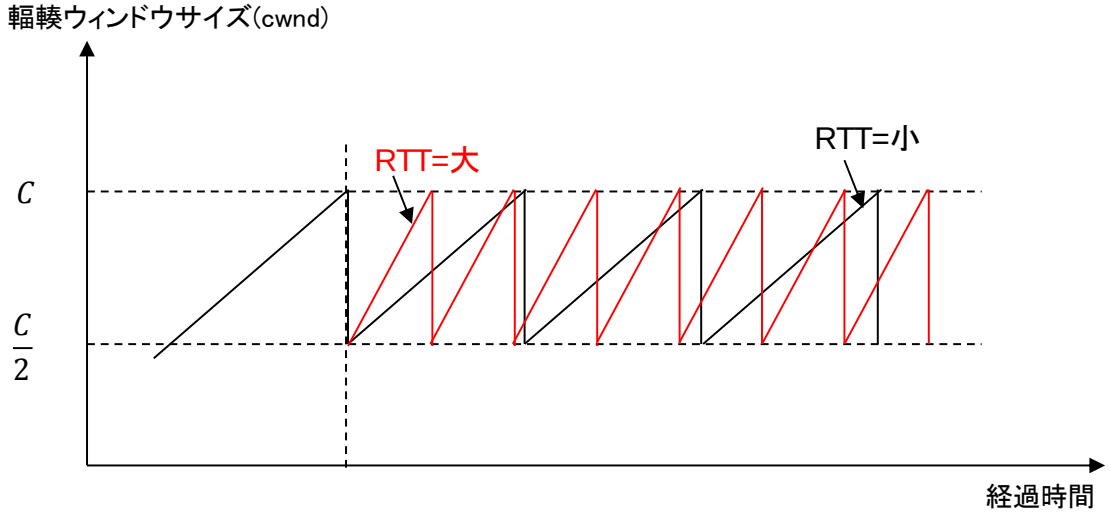


図 2-4 TCP Hybla の輻輳ウィンドウサイズの変化

2.1.6 Humilton TCP

H-TCP(Humilton TCP)[67]は, LFN 向けの輻輳制御アルゴリズムである. H-TCP は，式(2-12)に従い，輻輳ウィンドウサイズを広げる．この時，増加率αは，輻輳検出時刻からの経過時間Δの関数となっている．Δ^Lは広帯域回線と狭帯域回線を区別するための閾値であり，輻輳検出時刻からの経過時間に対する閾値となっている．狭帯域回線では，輻輳を検出して元の輻輳ウィンドウサイズに戻るまでの時間が短いのに対し，広帯域回線では元の輻輳ウィンドウサイズに戻るまでの時間が長いため，この値により広帯域回線を区別し，広帯域回線となる場合($\Delta > \Delta^L$)は，輻輳ウィンドウサイズの増加率をより大きくし早く輻輳ウィンドウサイズが広がるように設計されている[59][60]. また, 輻輳を検出した際は，式(2-13)に従い輻輳ウィンドウサイズを縮小する．

図 2-5 を用いてこれらの式の動作を説明する．はじめに輻輳ウィンドウサイズがCまで広がったところで輻輳を検出すると，式(2-13)に従い輻輳ウィンドウサイズを半分の大きさに縮小する．輻輳検出からの経過時間$\Delta \leq \Delta^L$の範囲では，式(2-12)において$\alpha = 1$として輻輳ウィンドウサイズを広げる．この時，2.1.7 節で述べる TCP Reno と同等の動作となる．輻輳検出からの経過時間$\Delta > \Delta^L$の範囲

では，式(2-12)において$\alpha = \alpha^H$として輻輳ウィンドウサイズを広げる．

$$\left.\begin{aligned} & cwnd \leftarrow cwnd + \frac{\alpha}{cwnd} \\ & \alpha = \begin{cases} 1 \ , for \ \ \Delta \leq \Delta^L \\ \alpha^H \ , for \ \ \Delta > \Delta^L \end{cases} \\ & \alpha^H = 1 + 10(\Delta - \Delta^L) + \left(\frac{\Delta - \Delta^L}{2}\right)^2 \end{aligned}\right\} \quad (2\text{-}12)$$

$$\left.\begin{aligned} & cwnd \leftarrow cwnd \times (1 - \beta) \\ & \beta = 0.5 \end{aligned}\right\} \quad (2\text{-}13)$$

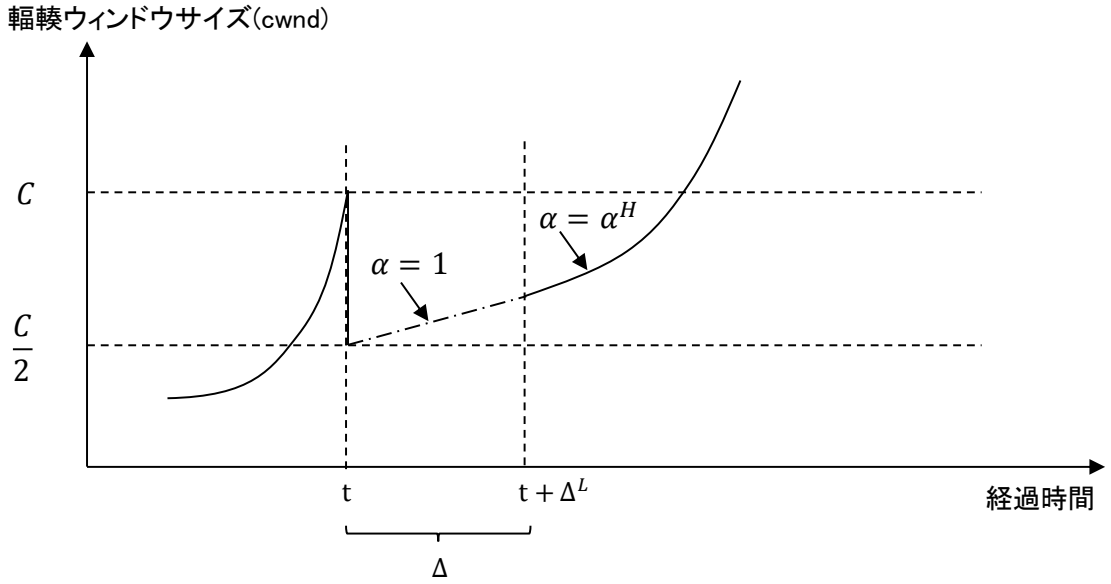


図 2-5 H-TCP の輻輳ウィンドウサイズの変化

2.1.7 TCP Reno

TCP Reno は現在，多くの TCP 実装により標準的に用いられる輻輳制御アルゴリズムであり，他の輻輳制御アルゴリズムとの性能比較の基準として用いられる．輻輳回避モードでは，TCP Reno は式(2-14)に基づき輻輳ウィンドウサイズを広げる．一方，輻輳を検出した際は，式(2-15)に従い輻輳ウィンドウサイズを縮小する．

図 2-6 を用いてこれらの式の動作を説明する．はじめに輻輳ウィンドウサイ

ズがCまで広がったところで輻輳を検出すると，式(2-15)に従い輻輳ウィンドウサイズを半分の大きさに縮小する．式(2-14)は，RTT 内に$cwnd$個のパケットを送信し，それに対する ACK が戻ってきたら$cwnd$が 1 増加することを意味しており，RTT 毎に輻輳ウィンドウサイズが 1 ずつ増加する．

$$cwnd \longleftarrow cwnd + \frac{1}{cwnd} \tag{2-14}$$

$$\left.\begin{aligned} cwnd &\longleftarrow cwnd \times (1-\beta) \\ \beta &= 0.5 \end{aligned}\right\} \tag{2-15}$$

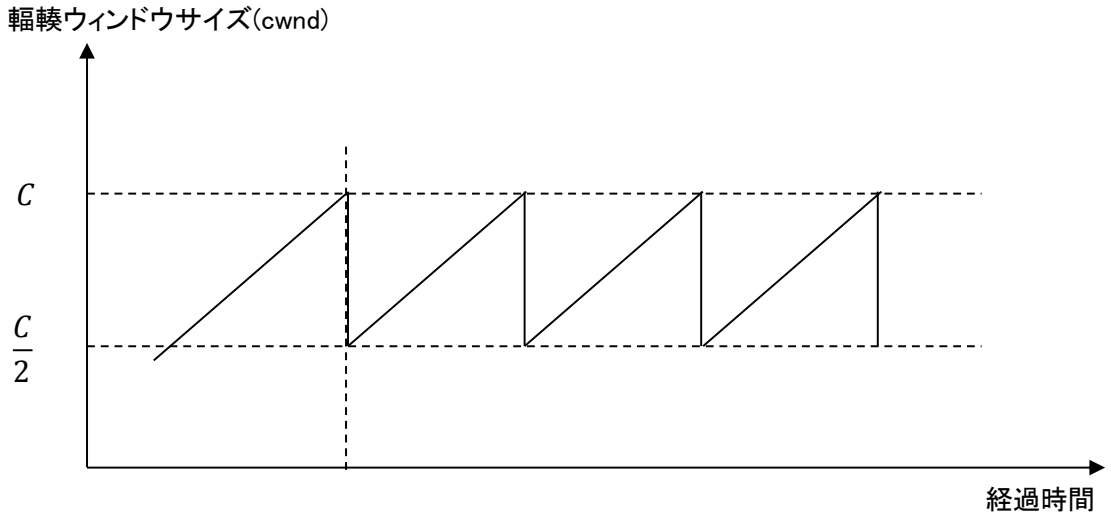


図 2-6 TCP Reno の輻輳ウィンドウサイズの変化

2.1.8 TCP Vegas

TCP Vegas[62][66]は，他の輻輳制御アルゴリズムと異なり，計測している往復遅延時間 RTT の変動により輻輳ウィンドウのサイズを変更する輻輳制御アルゴリズムである．

TCP Vegas は輻輳回避モードにおいて RTT 毎に式(2-16)に示す計算式に従い輻輳ウィンドウサイズ cwnd の値を更新する．$baseRTT$は，TCP が観測した最小の RTT である．ここで，$Diff$は，通信パス上での最大転送レートと現在のレートとの差分を表わしている．すなわち，レートの差分が小さい場合は，輻輳がな

いと判断し，輻輳ウィンドウサイズを広げていく．レートの差分が大きい場合は，輻輳していると判断し，輻輳ウィンドウサイズを縮小する．レートの差分が一定の範囲内に収まっている場合は，輻輳ウィンドウサイズを安定させる．

$$cwnd \leftarrow \begin{cases} cwnd + 1 \ , for \ \left(Diff < \frac{\alpha}{baseRTT}\right) \\ cwnd \ , for \ \left(\frac{\alpha}{baseRTT} < Diff < \frac{\beta}{baseRTT}\right) \\ cwnd - 1 \ , for \ \left(\frac{\beta}{baseRTT} < Diff\right) \end{cases} \quad (2\text{-}16)$$

$$Diff = \frac{cwnd}{baseRTT} - \frac{cwnd}{RTT}$$

文献[62]では，$\alpha = 1$，$\beta = 3$を採用している．

2.1.9 TCP Westwood

TCP Westwood は, TCP Reno を有線だけでなく無線網でも利用できるよう改善したものである．無線環境における TCP 通信では，伝送誤りやハンドオーバによるパケットロス(以降，無線網における信号劣化によるパケットの消失のみを指す場合にパケットロスと呼ぶこととする)と，ネットワークの輻輳によるパケット廃棄を区別できないため，不必要に輻輳ウィンドウサイズを下げるためスループットが出ないという問題がある．

TCP Westwood は，受信した ACK を観察することで，ネットワークの利用可能な帯域幅 Bandwidth Estimation for each flow (BWE)を推定する．測定される帯域幅は，$BWE = d_k/\Delta_k$と表わされる．t_{k-1}を，一つ前の ACK を受信した時刻とすると，$\Delta_k = t_k - t_{k-1}$である．また，d_kは，Δ_k期間中に ACK により通知される受信バイト数を表す．

TCP Westwood は，輻輳回避モードで輻輳ウィンドウサイズを広げる場合は，TCP Reno と同様に式(2-17)に従い広げる．一方で，輻輳を検出した場合は，式(2-18)に従い輻輳ウィンドウサイズを縮小する．この式の意味するところは，直近で流すことができた見積もりスループットを TCP のセグメントサイズで割っ

た分のウィンドウサイズとなるように$cwnd$を設定し，輻輳時に輻輳ウィンドウサイズを下げすぎないようにしている．

$$cwnd \leftarrow\!\!\shortmid\ cwnd + \frac{1}{cwnd} \tag{2-17}$$

$$cwnd \leftarrow\!\!\shortmid\ ssthresh = max\left(2, \frac{BWE \times RTT_{min}}{MSS}\right) \tag{2-18}$$

2.1.10 各種輻輳制御アルゴリズムの動作まとめ

前節までに説明したように，エンドツーエンドの TCP 通信性能を改善する手法として，様々な通信環境に合わせ改善された輻輳制御アルゴリズムが検討されてきた．表 2-2 は，これらの輻輳制御アルゴリズムをその仕組みにより整理している．同じグループの輻輳制御アルゴリズムは，似た挙動になると考えられる．

タイプ(a)は，現在の輻輳ウィンドウサイズ(cwnd)の値によらず一定の割合で cwnd の値を増加するタイプの輻輳制御アルゴリズムである．さらに，このタイプは輻輳の検出方法の観点から二つにカテゴライズできる．一つは(1)輻輳をパケット廃棄で検出する輻輳制御アルゴリズムである．他方は(2) 輻輳を RTT の変動で検出する輻輳制御アルゴリズムである．(1)のカテゴリの代表として TCP Reno があげられる．(2)のカテゴリの代表として TCP Westwood があげられる．

タイプ(b)は, 現在の輻輳ウィンドウサイズの値に応じて cwnd の増加率を変化させるタイプの輻輳制御アルゴリズムである．このタイプの代表として, Highspeed TCP や Scalable TCP があげられる．

タイプ(c)は，ネットワークの RTT の値に応じて cwnd の増加率を変化させるタイプの輻輳制御アルゴリズムである．このタイプの代表として，TCP Hybla があげられる．

タイプ(d)は, 輻輳検出時刻からの経過時間に応じて cwnd の増加率を変化させるタイプの輻輳制御アルゴリズムである. このタイプの代表として, CUBIC TCP や Humilton TCP があげられる．

また，表 2-3 に主な TCP 輻輳制御アルゴリズムの動作をまとめた．

表 2-2 各種輻輳制御アルゴリズムの機能的特徴

		(1) パケット廃棄による輻輳検出	(2) RTT の変動による輻輳検出
輻輳回避モードにおける輻輳ウィンドウサイズの広げ方	(a) cwnd 値によらず増加率が一定	Reno	Vegas, Westwood
	(b) 現在の cwnd 値により増加率を変更	Highspeed BIC Scalable	
	(c) RTT により増加率を変更	Hybla	
	(d) 輻輳検出からの経過時間により増加率を変更	CUBIC Humilton	

表 2-3 各種輻輳制御アルゴリズムの輻輳ウィンドウサイズ拡大・縮小方法

		Algorithms recognize congestion by detecting packet loss		
		Algorithms	Increase	Decrease
増加率固定		Reno	$cwnd = cwnd + 1/cwnd$	0.5
		Westwood	$cwnd = cwnd + 1/cwnd$ *Estimate the available bandwidth (BWE)	*ssthresh* = max(2, (BWE*RTTmin) / seg_size) *cwnd = ssthresh*
増加率が動的に変化	現在の cwnd 値により増加率を変更	HS-TCP	$cwnd = cwnd + a(w)/cwnd$ $a(w) = \frac{w^2 \times 2 \times b(w)}{(2 - b(w)) \times w^{1.2} \times 12.8}$	$b(w) = (0.1 - 0.5)\frac{\log w - \log 38}{\log(83000) - \log(38)}$
		BIC	$cwnd = cwnd + W_{INC}/cwnd$ $W_{INC} = \frac{(W_{max} - cwnd)}{2} \cdots case\ W_{max} > cwnd$ $W_{INC} = \frac{(cwnd - W_{max})}{2} \cdots case\ cwnd > W_{max}$	0.875
		Scalable	$cwnd = cwnd + 0.01\ \ for\ every\ ACK$	0.875
	RTT の大きさにより増加率を変更	Hybla	$cwnd = cwnd + \rho^2/cwnd$ $\rho = \frac{RTT}{RTT_0} \cdots Normalized\ RTT$ $RTT_0 : Refeerence\ RTT = 25ms$	0.5
	輻輳からの経過時間により増加率を変更	CUBIC	$cwnd = cwnd + W_{inc}/cwnd$ $W_{inc} = cwnd(t+RTT) - cwnd_t$ $cwnd = 0.4\left(t - \sqrt[3]{2W_{max}}\right) + W_{max}$ Wmax は輻輳検出時の cwnd の値	0.7
		H-TCP (Hamilton TCP)	$cwnd = cwnd + a/cwnd$ $\alpha = \begin{cases} 1 \cdots case\ \Delta \leq \Delta^L \\ \alpha^H \cdots case\ \Delta > \Delta^L \end{cases}$ $\alpha^H = 1 + 10(\Delta - \Delta^L) + \left(\frac{\Delta - \Delta^L}{2}\right)^2$	0.5

		Algorithms recognize congestion by detecting fluctuation of RTT		
		Algorithms	Increase	Decrease
増加率が動的に変化	ネットワーク内にバッファされているセグメントの見積もり量により増加率を変更	Vegas	$cwnd = cwnd + 1 \cdots (diff < \alpha)$ $cwnd = cwnd \cdots (otherwise)$ $cwnd = cwnd - 1 \cdots (diff > \beta)$ 一定時間(≒RTT)毎に cwnd の値を更新 ここで$diff = \frac{cwnd}{RTT_{min}} - \frac{cwnd}{RTT_{current}}$	0.75
		FAST-TCP (参考)	$cwnd = \min\left\{2cwnd,\ (1-\gamma)cwnd + \gamma\left(\frac{RTT_{min}}{RTT}cwnd + \alpha\right)\right\}$ 簡単にすると $cwnd = cwnd + \gamma(\alpha - x \cdot q)$ xは現在のスループット qは現在のキューイング遅延 αは目標バッファサイズ バッファ見積もり量が大きい程，増加幅，削減幅が大きくなる.	0.5

ここにあげた様々な輻輳制御アルゴリズムは，特定のネットワーク環境に対してはスループットの改善効果が大きいが，想定外のネットワーク環境で使われた場合には大幅に性能が劣化するという弱点がある.

2.2 TCP の輻輳ウィンドウサイズの初期値を変更しレイテンシを改善するアプローチ

現在の TCP が持つオプションやパラメタ設定の変更により，インタラクティブ通信のレイテンシを改善する方法がある．本節ではこれらについて説明する.

2.2.1 'slow-start restart' オプション

Linux の TCP 実装では，一般に通信データが送信されない状態が続くと，それ以降，データ送信が再開された時に，成長した輻輳ウィンドウの値を初期値に戻す機能 slow-start restart(SSR)オプションがある.

本機能が有効になっていると，インタラクティブ通信のように不連続にデー

タ通信が行われる際，その都度輻輳ウィンドウが初期値から始まる．

特に RTT が大きいネットワークでは，輻輳ウィンドウの成長に時間がかかるため, オプション指定により本機能を無効にすることができる[37]. これにより，インタラクティブ通信のように連続的にデータが流れない通信であっても，大きく広がった輻輳ウィンドウサイズを用いて次のデータ通信を始めることができる．

SSR を用いることで，輻輳ウィンドウサイズをより大きな値から始められるためパケット廃棄が全くない状況では本技術は有効であるが，パケット廃棄が発生している状況では，本オプションの有効，無効にかかわらず輻輳ウィンドウが閉じるため，本オプションの効果はほとんどない．

2.2.2 大きな初期ウィンドウサイズ

TCP は ACK を受信する毎に輻輳ウィンドウサイズを大きくするが，この輻輳ウィンドウサイズの初期値をいくつにするかが OS や，OS のバージョン毎に異なる．

十分広い帯域を持つが，遅延が大きい LFN と呼ばれる網では，大きな初期値を持つ方が，輻輳ウィンドウを大きい値から開始できスループットを速く高い値にできるため，10 程度の初期値を持つ．

一方で，無線回線のように比較的帯域が細いネットワークの場合，輻輳を検出し輻輳ウィンドウサイズを下げた際，大きい初期値から再開すると再び輻輳を引き起こし易いため小さな初期値を用いる必要がある．

インタラクティブ通信のように比較的小さいサイズのバーストデータを一時に送信するような通信では，初期ウィンドウサイズを大きく設定するとより速くバーストデータを送信できる[38].

Linux OS version 2.6 の初期ウィンドウサイズのデフォルト値は 2 であり, Linux OS version 3.0 以降では初期ウィンドウサイズのデフォルト値が 10 に引き上げられ，インタラクティブ通信の応答性能の改善が行われている．

本技術は，非常に小さなデータをやり取りしているインタラクティブ通信には効果的であるが，データサイズがある程度大きくなると輻輳回避モードの動作が多くなり，使用している輻輳制御アルゴリズムの影響が出てくる．近年，

4K 解像度の仮想デスクトップサービスも登場し，本技術だけでは問題を解決できない.

2.3 無線レイヤの廃棄を TCP に輻輳と認識させないアプローチ

2.3.1 Loss Differentiation Algorithms (LDA)

LDA はパケット廃棄が発生した時の遅延時間により，輻輳によるパケット廃棄か，無線のエラーによるパケットロスかを区別する手法である．LDA の方式は大きく３つある.

2.3.1.1 Biaz scheme

Biaz scheme[29][30][31]は，パケット廃棄の種類を区別するため，パケット到着間隔T_iを測定する．T_iは，パケット廃棄が発生する前に受信したパケット(P_i)と，パケット廃棄の後に最初に受信した不連続なパケット(P_{i+n+1})の間の到着間隔である. ここで, nはパケット廃棄したパケットの数である. Biaz scheme では，T_iが特定の時間範囲の場合は，無線によるパケットロスと判断し，その範囲より早く，あるいは遅く到着した場合は，輻輳によるパケット廃棄と判断する.

図 2-7 は，Biaz scheme における無線によるパケットロスと，輻輳によるパケット廃棄を判断する仕組みを説明している．nを連続したパケット廃棄数とする．T_{min}はそれまで観測された最小のパケット到着間隔である．T_i/T_{min}が$n+1$から$n+2$の範囲に入る時，無線によるパケットロスと判断する．それ以外は輻輳によるパケット廃棄に区別する.

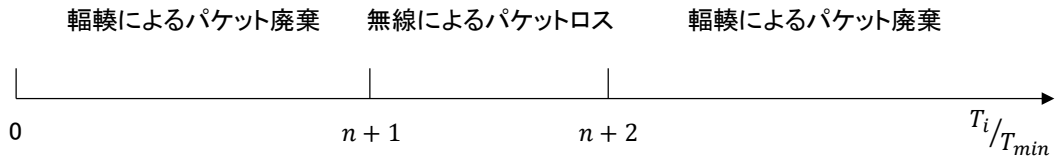


図 2-7 Biaz scheme によるパケット廃棄の識別閾値

2.3.1.2 Spike scheme

Spike scheme[32]は，輻輳の程度を区別する方法で，無線によるパケットロスと輻輳によるパケット廃棄を明確には分離しない．本手法では，相対片方向遅

延時間 ROTT (Relative One way Trip Time)を測定する．Spike 状態とは，輻輳に陥っていると判断される状態である．シーケンス番号iのパケットを受信した際，Spike 状態にない現在のコネクションにおいて，パケットiに対する ROTT が式(2-19)の閾値$B_{spikestate}$を超えた時，Spike 状態に入ると定義する．また，コネクションが Spike 状態にあり, ROTT が式(2-20)の閾値$B_{spikeend}$を下回った時, Spike 状態を抜けると定義する．

$$B_{spikestate} = rott_{min} + \alpha(rott_{max} - rott_{min}) \qquad (2\text{-}19)$$

$$B_{spikeend} = rott_{min} + \beta(rott_{max} - rott_{min}) \qquad (2\text{-}20)$$

ここで，$\alpha = 0.5, \beta = 0.33$である．

そして，コネクションが Spike 状態の時は，輻輳によるパケット廃棄に分類され，それ以外の状態では無線によるパケットロスに分類される．

2.3.1.3 Zig Zag scheme

Zig Zag scheme は，パケットの廃棄数nとi番目のパケットの RTT とi番目のパケットまでの平均 RTT との差により分類する. 式(2-21)の条件に合致する場合は無線によるパケットロスに分類される．これ以外の廃棄は，輻輳によるパケット廃棄に分類する．

$$\begin{aligned} &\mathrm{n} = 1 \ \ and \ \ rtt_i < rtt_{imean} - rtt_{dev} \\ &OR \ \ n = 2 \ \ and \ \ rtt_i < rtt_{imean} - rtt_{dev}/2 \\ &OR \ \ n = 3 \ \ and \ \ rtt_i < rtt_{imean} \\ &OR \ \ n > 3 \ \ and \ \ rtt_i < rtt_{imean} - rtt_{dev}/2 \end{aligned} \qquad (2\text{-}21)$$

LDA により網内の輻輳と無線のエラーによるパケット廃棄を区別し，無線のエラーによる TCP の輻輳制御を回避することが可能である．ところが，本方式は，無線のエラーには効果があるが，網内の輻輳による廃棄に対してはレイテンシを改善できない．

2.4 パケット廃棄が多いリンクにおけるエラーのリカバリをローカルで行うアプローチ

TCP 通信では，通信経路の途中でパケット廃棄が発生すると，本来 TCP の送

受信端で，これらパケットの廃棄を検出し再送を行う．ところが，通信経路が長い(すなわち RTT が大きい)場合は，パケット廃棄の検出に時間がかかる，さらにパケット廃棄の結果，ひとたび輻輳ウィンドウサイズを下げてしまうと輻輳ウィンドウサイズの回復に時間がかかる，といった問題がある．そこで，通信経路の中で，パケット廃棄が発生しやすいネットワーク(例えば無線網)の部分で，廃棄されたパケットに対する局所的な再送を行い，パケット廃棄を回復し，TCP 通信を行っている両端のホストに対しパケット廃棄を見せなくすることで，TCP の通信性能劣化を回避する手法が考えられてきた．本節では，これらの各種手法について説明する．

2.4.1 Snoop TCP

一般に TCP はパケットの廃棄を検出すると，廃棄されたパケットの再送(再送制御)と輻輳ウィンドウサイズを縮小し送信レートの抑制(輻輳制御)を行う．ところが，無線網では，ハンドオーバや，電波の状況に伴う無線リンクの切断や，高いビットエラーによりパケット廃棄が発生する場合がある．この場合，無線網でパケット廃棄が起きてもネットワークが輻輳しているわけではないので，再送のみ行えばよく輻輳制御は必要ない．ところが標準的な TCP は，輻輳によるパケット廃棄か，無線環境におけるパケット廃棄かを区別することができないため，無線網を介した TCP 通信の通信性能が低下するという問題があった．

Snoop TCP[35][36]の基本的な考えは，図 2-8 に示すように，Fixed Host から Mobile Host へデータを送る際には，Snoop Agent で中継した TCP セグメントを，逆方向に ACK が通過するまでキャッシュしておき，ACK が来ない場合は，Snoop Agent と Mobile Host 間で再送を行うというものである．

Mobile Host から Fixed Host へデータを送る場合は，シーケンス番号の不連続性から無線区間で紛失したパケットを検出し，TCP のタイムアウトによらず，Snoop Agent が Mobile Host へ NACK を送信し，データの再送を促す．

Snoop TCP は，ホストの TCP に何も変更を加えず，エンドエンドの TCP のセマンティクスを変更することなく通信性能を改善することができる．

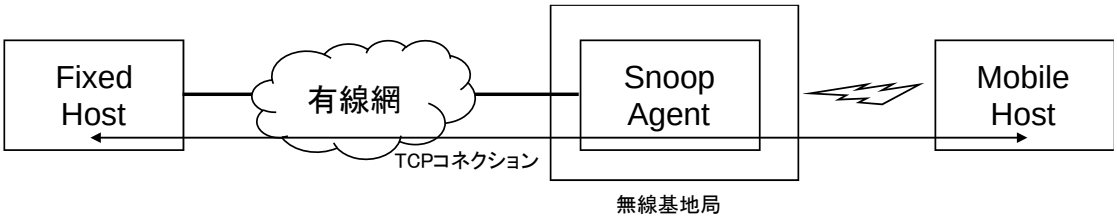


図 2-8 Snoop TCP の動作原理

2.4.2 Wireless profiled TCP (W-TCP)

W-TCP(Wireless profiled TCP)は，PDC(Personal Digital Cellular)，W-CDMA (Wideband Code Division Multiple Access), CDMA2000 などの 2〜2.5 世代，第 3 世代の携帯電話に最適化されたパケット伝送手順である．携帯電話の通信を最適化するために WAP(Wireless Application Protocol)フォーラムに提案され WAP2.0[33]の一部として公開されたプロトコルである. その後 RFC3481[34]として IETF から公開されている．

無線網は，電波の状況によりビットエラーやパケットの消失が発生しやすく，利用可能帯域や遅延が頻繁に変化するという特徴を持っている. そこで, W-TCP は，有線区間と無線区間で TCP のコネクションを分離するスプリット TCP(2.4.2.5 節で詳述)というアプローチと, 無線区間において既存の TCP のオプションを有効化したり, 拡張を行うことにより, 性能改善の工夫を行っている．以下にその工夫を示す.

2.4.2.1 ウィンドウスケーリング

W-TCP ではウィンドウスケーリングオプション[44]を有効化し，輻輳ウィンドウサイズを 64KB 以上のサイズにできるようにしている．

2.4.2.2 初期ウィンドウの増加

TCP の初期ウィンドウサイズを大きい値にする[45]ことで, 小容量のデータしか送らないコネクションでも少ない回数の RTT でデータを転送可能とする．

2.4.2.3 SACK対応

SACK オプション[46]を有効化することで，一部のデータが廃棄された場合でも TCP ウィンドウ内の全てのデータを再送することなく廃棄されたデータだけを再送できるようになる．

2.4.2.4　TCPタイムスタンプオプション

文献[44]に記載される TCP タイムスタンプオプションにより，TCP ヘッダにタイムスタンプを付けることで，RTT の精度を向上でき，再送タイムアウトをより正確に行うことが可能になる．

2.4.2.5　スプリットTCP

無線網と有線網の境界にゲートウェイを配置し，無線区間でパケット廃棄が生じた場合に，無線区間だけでデータの再送を行うことで，エンドエンドの装置間で再送を行うことを回避する．これにより TCP 性能の劣化を緩和している．本手法は RFC2757[43]として標準化されている．

2.4.3　無線レイヤにおけるエラー改善技術

無線網では，伝送路におけるメディアエラーにより信号誤りが発生することで，受信側装置で正しく復号することができずに無線フレームを廃棄することがある．無線レイヤでは，このような信号誤りを少なくするための様々な仕組みが用いられている．

2.4.3.1　適応変調

適応変調は，無線送信デバイスが無線受信デバイスへ信号を送る際に，電波の強度に応じて変調方式を適応的に変更して送信する技術である．電波強度が弱い時は，データ転送量が少ないが信号誤りに強い変調方式を用いる．電波強度が強い場合は，データ転送量が多いが，信号誤りに弱い変調方式を用いる[40]．

2.4.3.2　HARQ

Hybrid automatic repeat request (Hybrid ARQ or HARQ)は，エラーのあった無線フレームを無線レイヤで誤り訂正するメカニズムである[41]．HARQ は，送信データに冗長ビットが付加された上で，受信デバイスが復元できないフレームについては，再送を行うことで誤りを訂正する．電波状況が悪く信号を正しく復号できない場合は，無線レイヤで何度も再送を行うため，廃棄率を減らせる代わりに無線レイヤにおける遅延時間が大きくなる．

W-TCP や Snoop TCP などのパケット廃棄が多いリンクにおけるエラーのリカバリをローカルで行うアプローチにより廃棄が発生し易いローカルな区間に TCP セッションを区切り再送を行うことで，パケット廃棄のリカバリを高速に行えるが，途中にエージェントなどを配置する必要があり，エンドツーエンドのソリューションにならないといった課題がある．また，W-TCP では，TCP のセマンティクスが切れるため，アプリケーションによっては正しく動作しなくなるものがある．

無線リンク層でエラーをリカバリするアプローチは，HARQ などがある．HARQ は再送を行うためレイテンシは伸びる方向にあり，TCP からは極めて大きな遅延のあるネットワークのように見え，輻輳ウィンドウの成長に影響が大きくレイテンシの改善には寄与しにくい．

2.5 ダイナミックに輻輳制御アルゴリズムを切替える方式

近年の輻輳制御アルゴリズムには，より高いスループットを得るため，複数の輻輳制御アルゴリズムに基づき輻輳ウィンドウサイズを計算し，より大きい輻輳ウィンドウサイズの値を採用するものがある．

2.5.1 Compound TCP(CTCP)

Compound TCP は，パケット廃棄により輻輳を検出する輻輳制御アルゴリズムと，RTT の変動により輻輳を検出する輻輳制御アルゴリズムの両方で輻輳ウィンドウサイズを計算し，両アルゴリズムにより計算された値の和を輻輳ウィンドウサイズとして採用する[42].

本輻輳制御アルゴリズムを用いることで，無線網のように，輻輳が発生しているわけではないが，無線のメディアエラーによりパケット廃棄が発生しているネットワークにおいて，TCP の輻輳ウィンドウサイズを低下させることなく通信可能になり，性能を改善できる．

2.5.2 Scalable TCP，CUBIC TCP

Scalable TCP と呼ばれる輻輳制御アルゴリズムは，図 2-9 に示すように

Scalable TCP のアルゴリズムと同時に TCP Reno に基づいた輻輳ウィンドウサイズを計算し，その値が大きい方(実線で示した値)を採用することで性能を改善する．CUBIC TCP も同様の計算方法を行うことで，輻輳ウィンドウサイズを広げる際により大きな値となる輻輳制御アルゴリズムを選ぶことが可能となる．

このような，最も大きい輻輳ウィンドウサイズを持つ輻輳制御アルゴリズムを選択する手法は，輻輳通信環境が良好な場合は通信性能を改善できると考えられる．ところが，輻輳やパケット廃棄が頻発している環境で，再度の輻輳の引き起こしにくさ，通信アプリケーションの実効性能の高さといった観点の配慮がない．

再度の輻輳の引き起こしにくさという観点では，これらの手法は，輻輳状況下では，アグレッシブに輻輳ウィンドウを広げすぎるため再度輻輳を引き起こし易く，平均的な輻輳ウィンドウの値がかえって低下する可能性がある．

また，通信アプリケーションの実効性能の高さといった観点では，図 1-9 に示すようなインタラクティブ通信における理想的な輻輳制御に近づけるには，輻輳ウィンドウサイズの下げ幅が小さいアルゴリズムを選ぶのが良いが，輻輳制御アルゴリズムの増加率αと減少率βはアルゴリズム毎に決まっており，輻輳ウィンドウサイズの広がりが速い輻輳制御アルゴリズムが，必ずしも輻輳検出時にウィンドウサイズの減少率βの小さいアルゴリズムになるとは限らない．そのため，増加率αが大きい輻輳制御アルゴリズムであっても通信アプリケーションの実効性能は低くなる可能性がある．

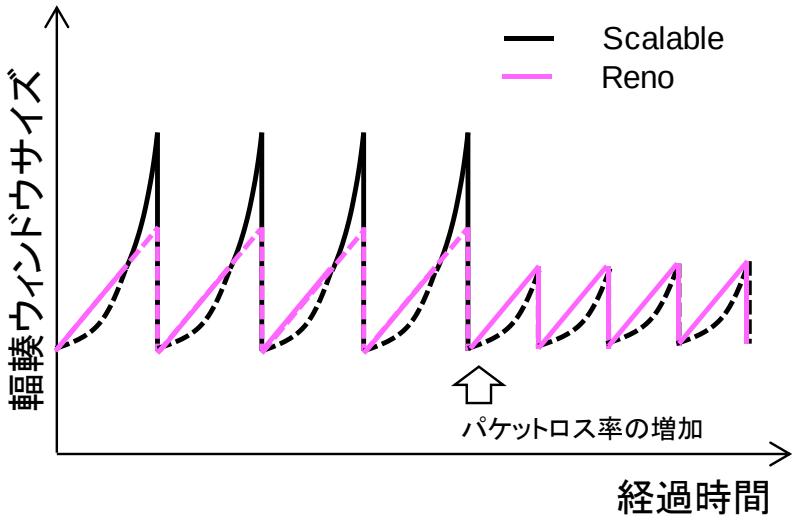


図 2-9 Scalable TCP の輻輳制御

2.6 輻輳制御アルゴリズムを自動で設計する技術

文献[39]は，コンピュータに TCP の輻輳制御アルゴリズムを設計させる手法を提案している．本手法では，ネットワークの前提条件，トラフィックモデル，目的関数を指定することで，最適な輻輳制御アルゴリズムを自動的に生成することができる．

具体的には，以下の 3 つを入力として与える．(1)リンク速度，RTT，網内のキュー長などのネットワークについての前提条件，(2)エンドホストにおいてトラフィックを発生している ON 時間，トラフィックを発生していない OFF 時間といったトラフィックモデル，(3)スループットを最適化するのかパケットの遅延を最適化するのかといった目的関数．

Remy と呼ばれる輻輳制御アルゴリズム生成ロジックは，(1),(2)の条件から生じるトラフィックをシミュレートし，その中で最も多く発生する(a)TCP パケット間隔，(b)ACK 間隔，(c)最小 RTT と平均 RTT の比，の 3 つのパラメタの組を特定する．そして，このパラメタの組に対し，輻輳制御アルゴリズムが目的関数を最適化するような輻輳制御アルゴリズムパラメタをカットアンドトライで抽出する．

本技術によると，ダンベル型トポロジーのネットワークにおいては，予め前

提条件として与えた帯域や RTT の範囲内での動作であれば，人間により設計された従来の輻輳制御アルゴリズムに比べ，より高性能な輻輳制御アルゴリズムを生成できることを示している．本論文では，輻輳制御アルゴリズムのチューニングパラメタにより，スループットの観点，パケット遅延の観点のどちらでも，既存の輻輳制御アルゴリズムに比べ高い性能を示すことを示している．

一方で，LTE を用いた無線網では，提案方式は 8 セッション以下の同時フローに対しては効果があるが，それ以上では既存の輻輳制御アルゴリズム(TCP Vegas)を用いた方が，パケット遅延が小さくなる結果を示している．

また，提案方式は，4.7.2 節で後述する RTT fairness については，課題を残している．

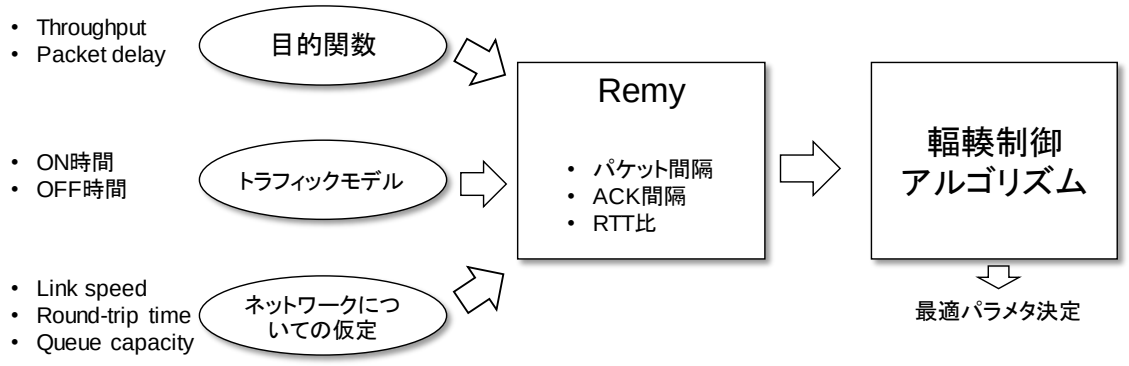


図 2-10 輻輳制御アルゴリズムを自動設計するアルゴリズム

輻輳制御アルゴリズムを自動で設計する技術は，様々なネットワークの特性毎に最適な輻輳制御アルゴリズムを個別に生成することはできるが，通信を行っているセッションが経由するネットワークの特性が動的に変化する場合に広く適応することは考えられていない．

文献[39]は，全ての環境に対し万能な輻輳制御アルゴリズムは存在しないという見地のもと，環境に合わせ最適な輻輳制御アルゴリズムを生成するというアプローチをとっている．そして，予め条件として与えたネットワークの特性に基づき目的関数を最適化する輻輳制御アルゴリズムをシミュレーションにより求めるため膨大な時間を要する．そのため，特性が変化しにくいネットワークでは効果的であるが，無線網のように時々刻々特性が大きく変化するネットワークにはこの方式は機能しにくいという弱点がある．

さらに，文献[39]は，ネットワークの前提条件が厳密であるほど，生成される輻輳制御アルゴリズムは，既存の輻輳制御アルゴリズムより高い性能を示すが，ネットワークの前提条件が広いレンジを取るほど，性能が低下してしまうことを示している．すねわち，特性が大きく変わるネットワークに向けた輻輳制御アルゴリズムを求めることも可能であるが，既存の輻輳制御アルゴリズムより高い性能を求めるのは困難であることを示している．

2.7 本章のまとめ

本章では，様々な特性に時々刻々変動するネットワークにおいてインタラクティブ通信のレイテンシを改善するという序論で示した目標に対する関連研究として，TCP の輻輳制御アルゴリズムを改良することでレイテンシを改善する技術，通信環境に応じ輻輳制御アルゴリズムを動的に変更することでレイテンシを改善する技術を示した．

TCP の輻輳制御アルゴリズムを改良することでレイテンシを改善する技術として，LFN や無線網で高い通信性能が出るように改良した様々な輻輳制御アルゴリズム，網内の輻輳によるパケット廃棄と無線のメディアエラーによるパケット廃棄を LDA と呼ばれる方法で区別し，無線によるパケット廃棄を TCP に輻輳と認識させないことで通信レートの低下を回避するアプローチ，TCP の輻輳ウィンドウサイズの初期値を変更しレイテンシを改善するアプローチ，W-TCP や Snoop TCP のように，パケット廃棄が多いリンクにおける再送をローカルで行い，末端の TCP に輻輳と認識させないことで通信レートの低下を回避するアプローチを紹介した．

しかし，これらのアプローチは，特定の通信環境や，限られた条件に向けた対処であったり，ネットワーク内に特別な機器やエージェントを配置するといった制約があり，モバイル環境を介しクラウドにアクセスする場合のように通信中に特性が時々刻々変動する通信には，個々のアプローチだけでは適応しきれないと考えられる．

そこで，さらに異なる通信環境に動的に適応する技術として，輻輳制御アルゴリズムを動的に切り替えるアプローチ，通信環境に合わせ輻輳制御アルゴリズムを自動的に設計するアプローチを調査した．

輻輳制御アルゴリズムを動的に切り替えるアプローチは，通信環境が良好な場合はスループットを改善できると考えられるが，常に最大の輻輳ウィンドウサイズを持つ輻輳制御アルゴリズムを基準に選択するため，輻輳状況下では，アグレッシブに輻輳ウィンドウを広げすぎることで再度輻輳を引き起こし，通信アプリケーションの実効性能がかえって低下する可能性があることを示した．

通信環境に合わせ輻輳制御アルゴリズムを自動的に設計するアプローチは，特定の通信条件に絞り込む程高い通信性能を示す輻輳制御アルゴリズムを設計することができるが，通信環境の変化に広く対応できる輻輳制御アルゴリズムを設計しようとすると既存の輻輳制御アルゴリズムに比して性能が劣化することを示しており，様々な環境に対応できる輻輳制御アルゴリズムを設計することが困難であることを示している．

こうした分析から，単一の輻輳制御技術で時々刻々変化する通信環境に即座に対処するのは困難であり，様々な通信環境に向けた異なる輻輳制御アルゴリズムを，廃棄のパタンや廃棄率に応じて，通信アプリケーションの実効性能を考慮しながら動的に選択し切替える技術が必要と考えられる．

第3章 インタラクティブ通信における各輻輳制御アルゴリズムの適用領域

第 2 章では，様々な関連研究について説明し，それらの課題を示した．そして，序章で示した，動的に変動し，様々な特性を持つネットワークにおいてレイテンシを改善するという目標に対しては，様々な通信環境に向けた異なる輻輳制御アルゴリズムを，廃棄のパタンや廃棄率に応じて，通信アプリケーションの実効性能を考慮しながら動的に選択し切替える方法が有効という見通しを得た．このように各種輻輳制御アルゴリズムを通信環境に応じて切り替えるためには，それぞれの輻輳制御アルゴリズムがどのような通信環境で優位となるのか，またどのような条件によりこれらの特性が変化するのかを把握しておく必要がある．そこで，本章では，各輻輳制御アルゴリズムの機能的特徴について述べると共に，各輻輳制御アルゴリズムが，インタラクティブ通信において優位な性能を示す通信条件について説明する．

3.1 輻輳制御アルゴリズムの性能に影響する要因

1.4.3 節における図 1-9 において，インタラクティブ通信における理想的な輻輳制御を示した．理想的な輻輳制御では，輻輳ウィンドウサイズを広げる際はなるべく速く立ち上がり，輻輳を検出した際にはなるべく輻輳ウィンドウサイズを小さくしないように制御するのが良い．輻輳ウィンドウサイズを広げる際には，各輻輳制御アルゴリズムの違いによる所が大きいが，表 2-2 の(c)に示したように立ち上がりの速さは RTT により変わりうる．

また，図 1-9 において輻輳を検出した際に輻輳ウィンドウサイズを縮める動作は，パケット廃棄により生じ，表 2-3 に示したように輻輳制御アルゴリズムにより下げ幅が異なると共に，パケットの廃棄パタンにより挙動が異なってくる．ネットワーク内では様々な原因でパケット廃棄が生じる．例えば，無線網におけるビットエラーのようなメディアエラーから生じるパケットロス，ネットワーク内に配置された中継装置にパケットが集中することによりバッファが溢れることで生じるパケット廃棄がある．特に昨今の無線アクセス技術では Hybrid Automatic Repeat Request (HARQ) [63]といった無線リンクレイヤの再送技

術が備わっており，メディアエラーによるパケットロスは無線リンクレイヤの再送によりリカバリされ，TCP には RTT が伸びたように見える．しかし，無線状態が悪化しパケット廃棄が増えすぎると HARQ の再送バッファが溢れ，TCP にはパケット廃棄として見えるようになる．

上記のような廃棄のパタンの違いは，TCP が廃棄を認識する際の挙動の違いにつながり，TCP の通信性能に差を生じさせるため，以下に述べるように大きく 2 つに分けて考える．一つは，パケット廃棄が連続して 3 つ以上発生する場合で本稿では**バースト的廃棄**と呼ぶこととする．他方，廃棄が高々2 個までしか続かない場合を**非バースト的廃棄**と呼ぶこととする．なお，ここでは TCP の SACK オプションが使用される設定になっていることを前提としている．

非バースト的廃棄が発生した場合，図 3-1 に示すように，送信側 TCP はパケットの廃棄を，重複 ACK を受信することで検出する．送信側 TCP は最初の重複 ACK を受信すると現在ネットワークに滞留しているパケットのおよそ半分まで輻輳ウィンドウサイズを引き下げるレートハービングと呼ばれる動作を行う．送信側 TCP は，その後，3 つ目の重複 ACK を受信した時点で輻輳と判断する．そして，輻輳を検出した際に，その時の輻輳ウィンドウサイズ(cwnd)に減少率βをかけて新たな輻輳ウィンドウサイズを決定する(多くの輻輳制御アルゴリズムは，この値を ssthresh の値に設定する)．このため，非バースト的廃棄の発生時は輻輳ウィンドウが初期ウィンドウサイズに近い値まで低下する．

一方，ネットワーク内でバースト的廃棄が発生した場合，受信側 TCP はパケットの連続廃棄を，Selective ACK (SACK)[46]を用いて送信側に通知する．この場合，送信側 TCP は即座に輻輳と判断し，レートハービングは行わず，SACK を受信した際の輻輳ウィンドウサイズに減少率βをかけた値まで輻輳ウィンドウサイズを下げる(図 3-2).

その結果，非バースト的廃棄の場合は，輻輳ウィンドウサイズが大きく落ち込むことにより一つのバーストを送るのに 2RTT 以上かかることが多くなるが，バースト的廃棄の場合では, バーストを送るのに 2RTT 以上かかることが最小限になる．そのため，両廃棄パタンでバーストを転送するためのレイテンシ性能に差が生じることになる．

このように，廃棄パタンによっても輻輳ウィンドウサイズの下げ幅が異なり，

理想の輻輳制御に近づけるためには，RTT，廃棄率に加え廃棄パタンも考慮してなるべく早く元の輻輳ウィンドウサイズに戻すことが可能な輻輳制御アルゴリズムを選択する必要がある．

以上に示したように，RTT の大きさ，非バースト的廃棄，バースト的廃棄といった廃棄パタンとその廃棄率の大きさが通信アプリケーションの実効性能に影響を与えると考えられる．

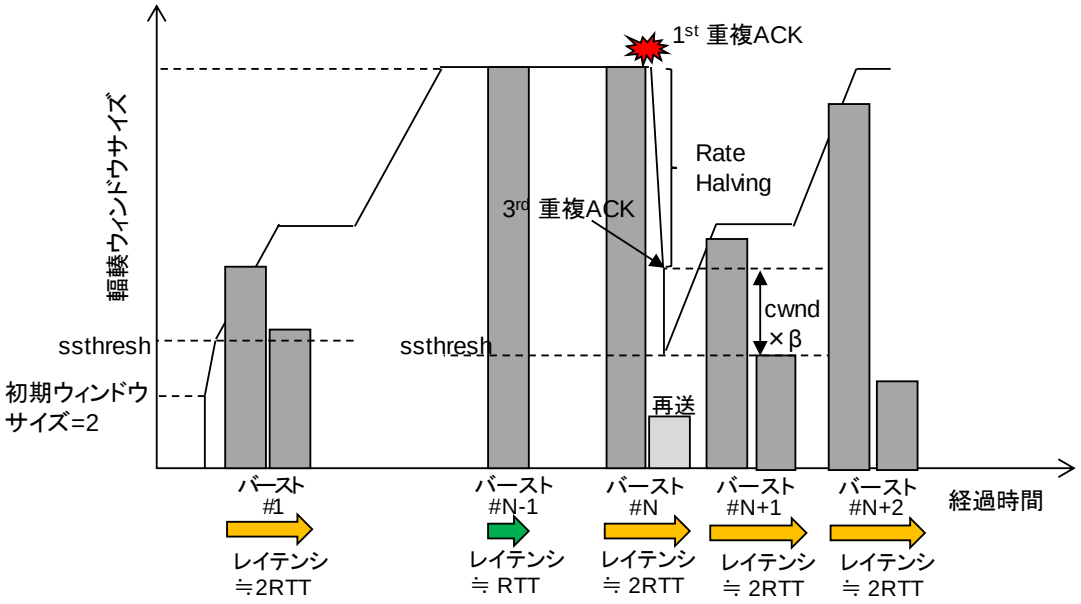


図 3-1 非バースト的廃棄時の輻輳制御

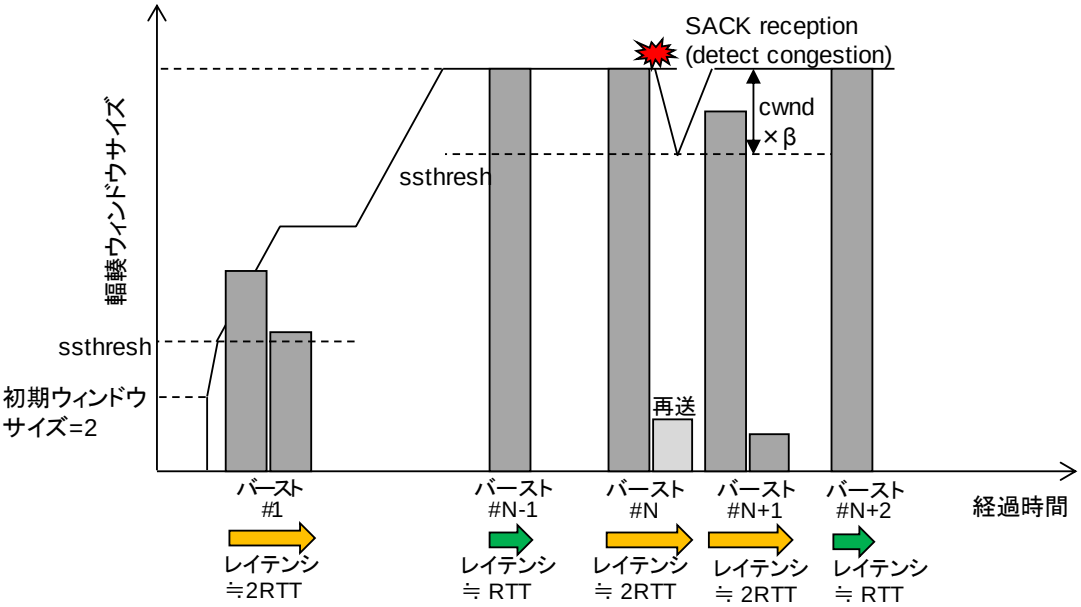


図 3-2 バースト的廃棄時の輻輳制御

3.2　各種輻輳制御アルゴリズムの挙動

本節では，どのような通信アプリケーションを用い，どのような通信条件の時にどの輻輳制御アルゴリズムが優位な性能を示すかという観点で各輻輳制御アルゴリズムの挙動を整理する．

想定する通信アプリケーション

本節では，各輻輳制御アルゴリズムの挙動を調べるため，大規模ファイル転送とインタラクティブ通信を想定している．表 3-1 に示すように，大規模ファイル転送では，アプリケーションが連続して送信するデータサイズが大きい．一方，インタラクティブ通信では，アプリケーションが連続して送信するデータサイズが小さい．1.4.2.3 節で説明したように，アプリケーションが送信するデータが少なくなる程 TCP の輻輳ウィンドウサイズの成長が遅くなりやすい．ここでは，アプリケーションが連続して送信するデータサイズを**バーストサイズ**と呼ぶこととし，バーストサイズの大小により各輻輳制御アルゴリズムの挙動を調べる必要がある．そして，大規模ファイル転送において，各輻輳制御アルゴリズムの優劣を評価する指標(性能指標)は平均スループットとしている．一方，インタラクティブ通信においては，バーストサイズが大規模ファイル転送に比べそれほど大きくない．インタラクティブ通信では，WAN 回線の速度，今日のインタラクティブな通信アプリケーションの特性を考慮し，64KBytes 以下をインタラクティブ通信で用いられるバーストサイズと想定している．また，インタラクティブ通信では性能指標として応答時間が重要である．図 1-10 に示した DaaS の場合では，レイテンシが小さいほど性能指標が高いと考えられる．これは，スループットで考えると，高い瞬間スループットを重視することを意味する．

表 3-1 大規模ファイル転送とインタラクティブ通信の違い

	大規模ファイル転送	インタラクティブ通信
一時に送信するデータのサイズ(バーストサイズ)	大	小
性能指標	平均スループット	応答時間(瞬間スループットに比例)

通信条件

3.1 節に示したように，輻輳制御アルゴリズムは RTT，非バースト的廃棄率，バースト的廃棄率により挙動が変化する．そこで，ネットワークの様々な特性を仮定し，RTT と廃棄率が以下に示す様々な条件を取る場合の組み合わせについて検討する．

- 廃棄率に関する想定

前節で述べたように各輻輳制御アルゴリズムは廃棄のパタンにより挙動が異なるので，各廃棄パタンに対する挙動を調べておく必要がある．ネットワーク内では，様々なパケット廃棄の要因があり，表 3-2 に示すように，これらの廃棄パタンが様々な組み合わせで発生する可能性がある．ここでは，それぞれの廃棄パタンに対する挙動を明らかにすることを目的とするため，表 3-2 の(1)～(3)の組み合わせについて検討する．バースト的廃棄率と非バースト的廃棄率の両方が高い場合も考えられるが，4.5 節の手順(5)で述べるように，(2)のケースと(3)のケースの統計的な多重として考えることができるので，ここでの検討からは除外する．

また，ここでは，各廃棄パタンにおける廃棄率の大小の境を 0.1%とした．これは，文献[72]に示される次世代ネットワークにおける品質規定値の高優先クラスにおけるパケット損失率の値を基準にしており，これより廃棄率が多い場合を廃棄率大，少ない場合を廃棄率小とした．

(1) バースト的廃棄率と非バースト的廃棄率の合計が小さい場合(ここでは廃棄率 0.1%未満を想定)

(2) バースト的廃棄率が高い場合(ここでは廃棄率 0.1%以上を想定)

(3) 非バースト的廃棄率が高い場合(ここでは廃棄率 0.1%以上を想定)

表 3-2 ネットワークにおける廃棄パタンの組み合わせ

	バースト的廃棄率 小	バースト的廃棄率 大
非バースト的廃棄率 小	(1)	(2)
非バースト的廃棄率 大	(3)	

- RTT に関する想定

ここでは，RTT の大小の境を 200ms とした．これは，文献[72]に示される次世代ネットワークにおける品質規定値の高優先クラスにおけるパケット転送遅延の値を基準にしており，これより RTT が大きい場合を RTT 大，小さい場合を RTT 小とした．

- RTT が小さい(ここでは RTT が 200ms 未満を想定)
- RTT が大きい(ここでは RTT が 200ms 以上を想定)

次節では，ここで示した，バーストサイズ，バースト的廃棄率，非バースト的廃棄率，そして RTT を変化させた時の各輻輳制御アルゴリズムの挙動を整理する．

3.2.1 パケット廃棄率が小さい場合の各輻輳制御アルゴリズムの適用領域

パケット廃棄が少ない環境では，より速くボトルネックとなる帯域に到達する輻輳制御アルゴリズムほど性能が優位と考えられる．後述の図 3-5 は，表 3-1，表 3-2 に従い，横軸にバーストサイズ，縦軸に RTT を取り，各輻輳制御アルゴリズムが性能優位となる領域を示した図である．以下にその理由を説明する．

3.2.1.1 RTTが小さい場合

図 3-3 に，廃棄が少なく RTT が小さい場合の各輻輳制御アルゴリズムの挙動を図示した．ここでは，通信アプリケーションから特定のバーストサイズのデータが繰り返し送信されている前提での挙動を示している．TCP Hybla のように RTT により増加率が変化する輻輳制御アルゴリズムは，RTT が小さい場合は成長が遅くなる．RTT が小さい場合は，Scalable TCP が最も速く輻輳ウィンドウが広がり，次いで CUBIC TCP の順に，ボトルネックの帯域に到達する．

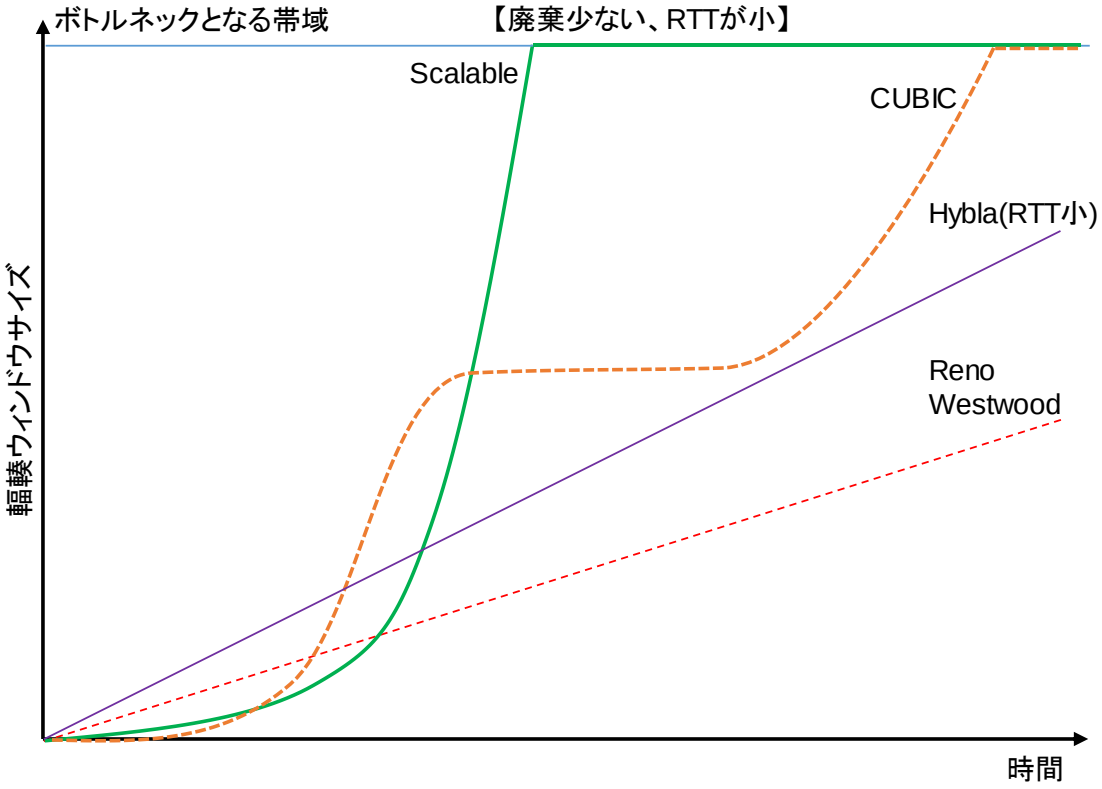


図 3-3 廃棄が少なく RTT が小さい場合の各輻輳制御アルゴリズムの挙動

● **バーストサイズが大きい通信**

RTT が小さく，バーストサイズが大きいデータ通信の場合 (図 3-5 の右下の領域) は，図 3-3 で示したように，Scalable TCP などの現在の cwnd 値により増加率を変更する輻輳制御アルゴリズムが高い平均スループットを示し性能優位となる．

● **バーストサイズが小さい通信**

RTT が小さく，バーストサイズが小さいデータ通信の場合 (図 3-5 の左下の領域) は，最大の輻輳ウィンドウサイズまで達する速さよりも，小さいバーストを送るのに十分な輻輳ウィンドウサイズまで速く広がる輻輳制御アルゴリズムが高い性能を示す．図 3-3 において，バーストサイズが大きい場合にボトルネックの帯域に速く到達する挙動を示す Scalable TCP は，バーストサイズが小さい場合は，通信アプリによるデータ送信速度の影響を受けるため，輻輳ウィンドウサイズの成長が緩やかになる．一方，CUBIC TCP のように輻輳からの経過時間に応じて増加率を変える輻輳制御アルゴリズムが，立ち上がりが速く性能優位となる．

3.2.1.2　RTTが大きい場合

RTT が大きい場合，図 3-4 に示すように，TCP Hybla のように RTT に応じて成長率が変化する輻輳制御アルゴリズムは，Scalable TCP のように現在の cwnd 値により増加率を変更する輻輳制御アルゴリズムよりも輻輳ウィンドウサイズの成長が速くなる．TCP Reno などの成長率がリニアな輻輳制御アルゴリズムが最もスループットが低くなる．

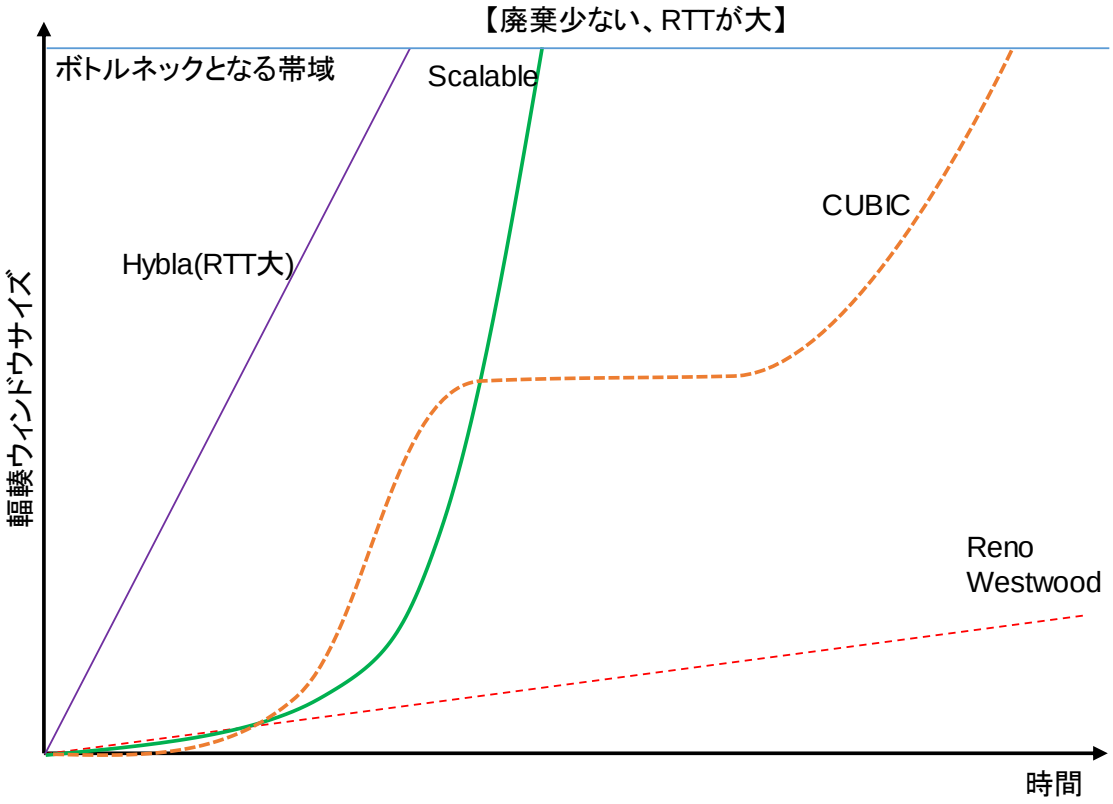


図 3-4 廃棄が少なく RTT が大きい場合の各輻輳制御アルゴリズムの挙動

● バーストサイズが大きい通信

RTT が大きく，バーストサイズが大きいデータ通信の場合（図 3-5 の右上の領域）は，図 3-4 に示すように，TCP Hybla も Scalable TCP も CUBIC TCP も立ち上がりの成長速度が大きく平均スループットが高くなる．

● バーストサイズが小さい通信

RTT が大きく，バーストサイズが小さいデータ通信の場合 (図 3-5 の左上の領域) は，TCP Hybla のように RTT に応じて cwnd 増加率を変化させるアルゴリズムが，図 3-4 に示すように立ち上がりの成長速度が極めて大きくなるため，他のアルゴリズムと比較して優位になる．

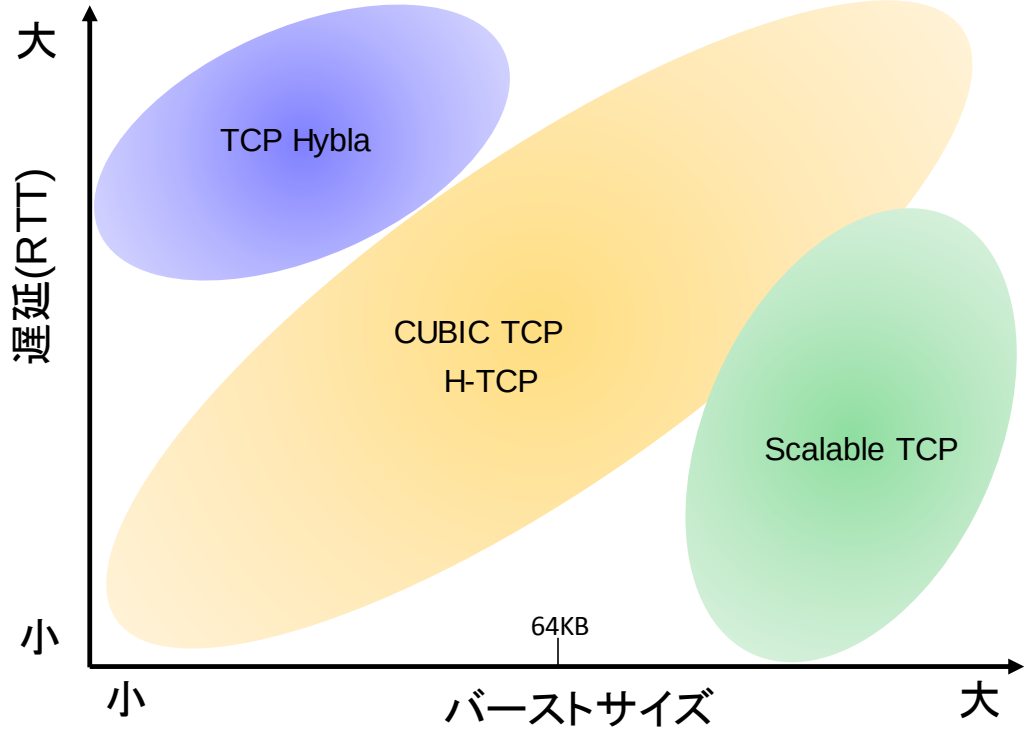


図 3-5 様々な輻輳制御アルゴリズムの適用領域(パケット廃棄が少ない場合)

3.2.2 バースト的廃棄が多い場合の各輻輳制御アルゴリズムの適用領域

バッファ溢れなどに伴うバースト的廃棄が多い環境では，パケット廃棄により輻輳を検出する輻輳制御アルゴリズム，RTT の揺らぎにより輻輳を検出する輻輳制御アルゴリズム共に, 廃棄の影響を受けやすい. 後述の図 3-8 は, 表 3-1, 表 3-2 に従い，横軸にバーストサイズ，縦軸に RTT を取り，各輻輳制御アルゴリズムが性能優位となる領域を示した図である．以下にその理由を説明する．

3.2.2.1 RTTが小さい場合

図 3-6 に，バースト的廃棄が多く RTT が小さい場合の各輻輳制御アルゴリズムの挙動を示した．TCP Hybla のように RTT により増加率が変化する輻輳制御アルゴリズムは，RTT が小さい場合は成長が遅くなる．バースト的廃棄がある状況では，CUBIC TCP や Scalable TCP は，図 3-2 に示したように輻輳を検出した際の削減率が小さいため，輻輳ウィンドウサイズが下がりにくい．また，バーストサイズが大きい場合では，バッファ溢れなどが発生しやすく，RTT の揺

らぎで輻輳を検出する TCP Westwood も輻輳を検出する．この場合は，削減率が小さい，CUBIC TCP や Scalable TCP が性能優位になる．特に CUBIC は，直前に輻輳を検出した輻輳ウィンドウサイズに一旦安定するため，再度輻輳を起こしにくく高いスループットが期待できる．

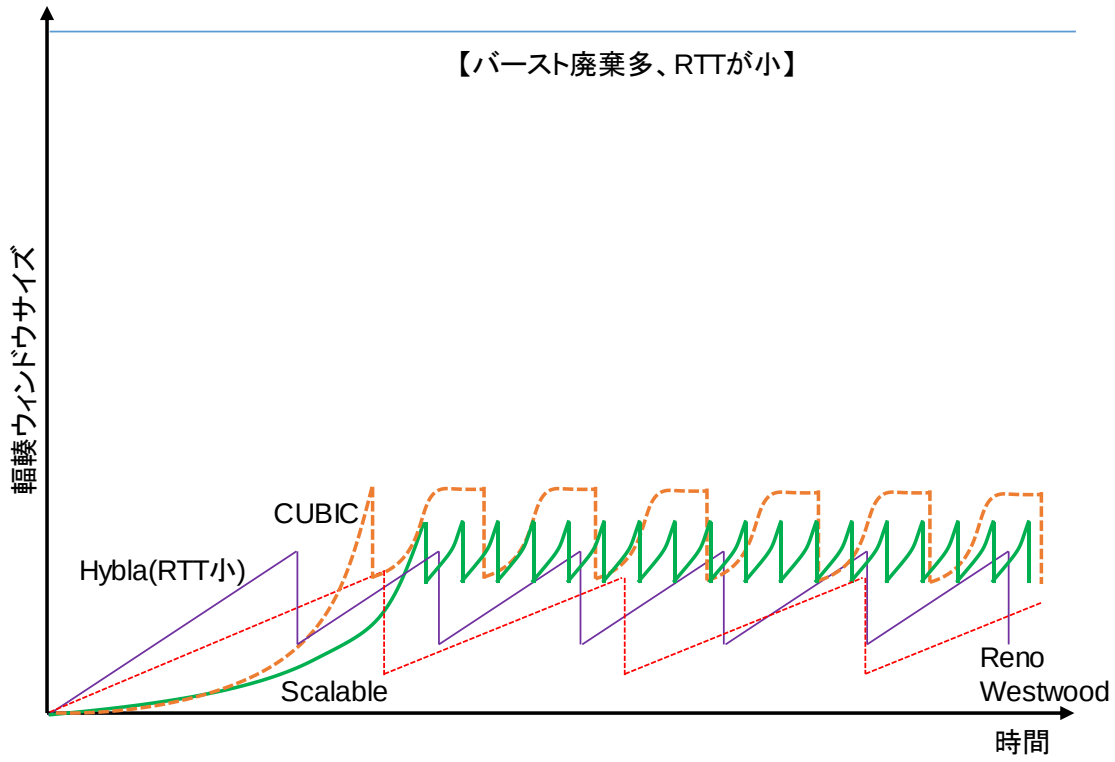


図 3-6 バースト的廃棄が多く，RTT が小さい場合の各輻輳制御アルゴリズムの挙動

● バーストサイズが大きい通信

RTT が小さく，バーストサイズが大きいデータ通信の場合 (図 3-8 の右下の領域) は, 受信する ACK が十分にあるため, 図 3-6 で示したように, Scalable TCP が，削減率が少ないことに加え，増加率も大きいため，高い平均スループットを示し性能優位となる．

● バーストサイズが小さい通信

バースト的廃棄が発生している場合は，SACK により廃棄を検出しやすい．RTT が小さく，バーストサイズが小さいデータ通信の場合 (図 3-8 の左下の領域) は，送信パケット数が少ないため，CUBIC TCP のように廃棄発生からの時間により輻輳ウィンドウサイズを広げる輻輳制御アルゴリズムが，高い輻輳ウ

ィンドウサイズを維持し，バーストデータを転送するためのレイテンシを小さくすることができる．

3.2.2.2　RTTが大きい場合

RTT が大きい場合は，CUBIC TCP や Scalable TCP のように輻輳を検出した際の削減率が小さい輻輳制御アルゴリズムが優位になる．また，無線のパケットロスなどでは輻輳を検出しにくい TCP Westwood も優位な性能を示す場合がある．

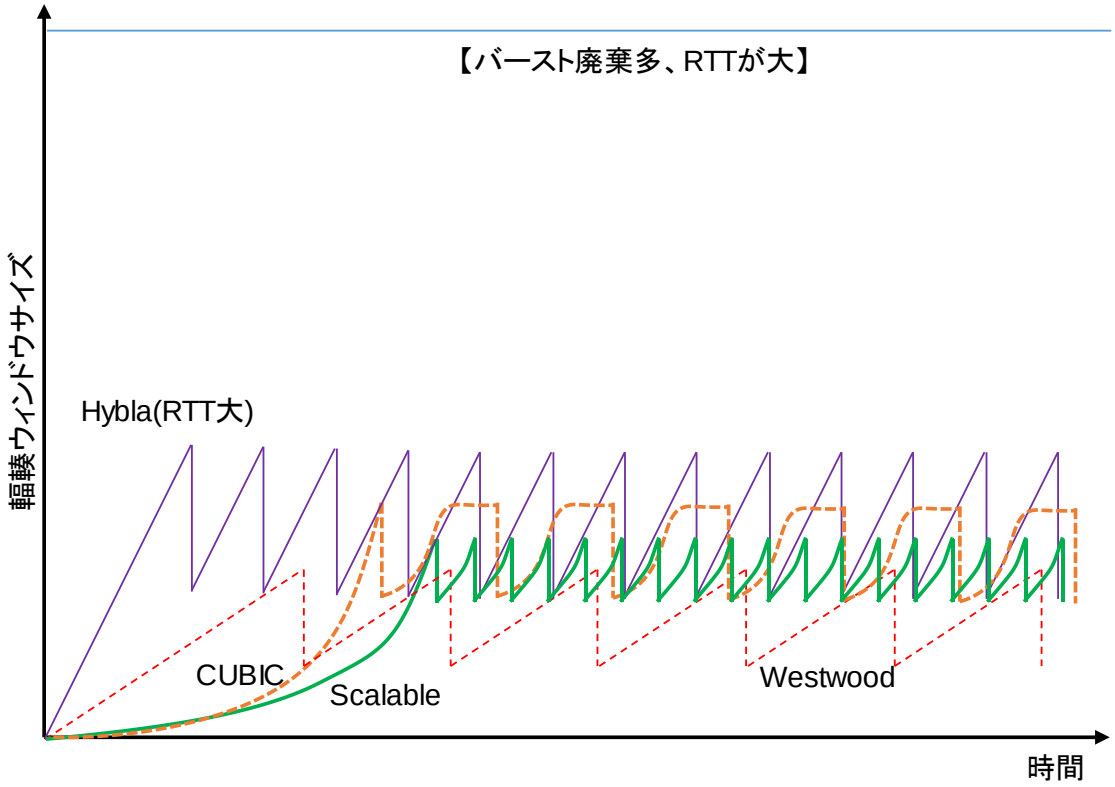


図 3-7 バースト的廃棄が多く，RTT が大きい場合の各輻輳制御アルゴリズムの挙動

● バーストサイズが大きい通信

RTT が大きく，バーストサイズが大きいデータ通信の場合 (図 3-8 の右上の領域) は，削減率が小さい Scalable TCP が性能優位となる．また，廃棄率が小さい場合は，パケット廃棄による輻輳を検出しにくい TCP Westwood が性能優位となる場合がある．TCP Hybla は，成長率がアグレッシブすぎるため，再度輻輳を起こしやすく，また輻輳検出時の削減率も大きいのでスループットとしてはそれほど高くならない．

● バーストサイズが小さい通信

RTT が大きく，バーストサイズが小さいデータ通信の場合 (図 3-8 の左上の領域) においても，バーストサイズが大きい通信時と同様に，Scalable TCP のように廃棄発生時の削減率が小さいアルゴリズムや，無線のパケットロスでは輻輳検出しにくい TCP Westwood が，廃棄発生時であっても高い輻輳ウィンドウサイズを維持し，バーストデータを転送するためのレイテンシを小さくすることができる．

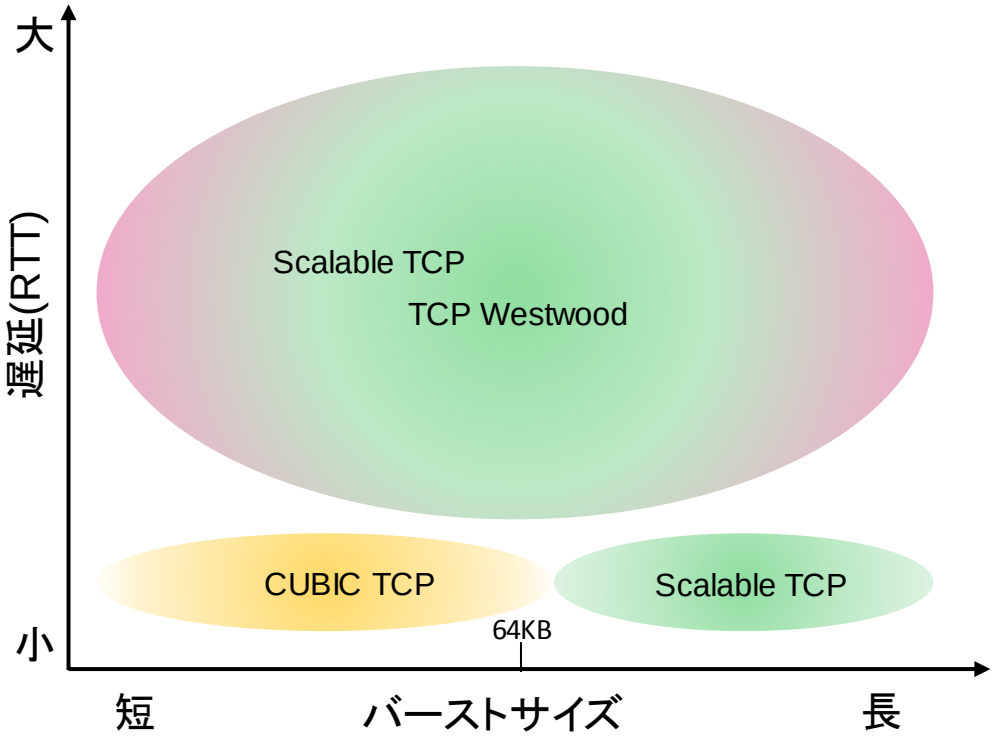


図 3-8 各種 TCP の適応領域 (バースト的廃棄が多い場合)

3.2.3 非バースト的廃棄が多い場合の各輻輳制御アルゴリズムの適用領域

非バースト的廃棄が生じると，図 3-1 に示したように，TCP はレートハービングを行うため，cwnd を大きく引き下げる．後述の図 3-11 は，表 3-1，表 3-2 に従い，横軸にバーストサイズ，縦軸に RTT を取り，代表的な輻輳制御アルゴリズムが性能優位となる領域を示した図である．以下にその理由を説明する．

3.2.3.1 RTTが小さい場合

TCP Hybla などの RTT に応じて成長率が変化する輻輳制御アルゴリズムは，

RTT が小さいため成長が極めて遅くなる. 非バースト的廃棄を検出した際の TCP はどの輻輳制御アルゴリズムも初期ウィンドウに近い値まで低下するため，成長の速さにより性能が決まる．CUBIC TCP や Scalable TCP は，廃棄率が極めて高い場合は，立ち上がりの速度が緩いため，Reno の方が性能優位になる場合がある．非バースト的廃棄の状況では，TCP Westwood は輻輳を検出しにくい．このため TCP Westwood が性能優位になりやすい.

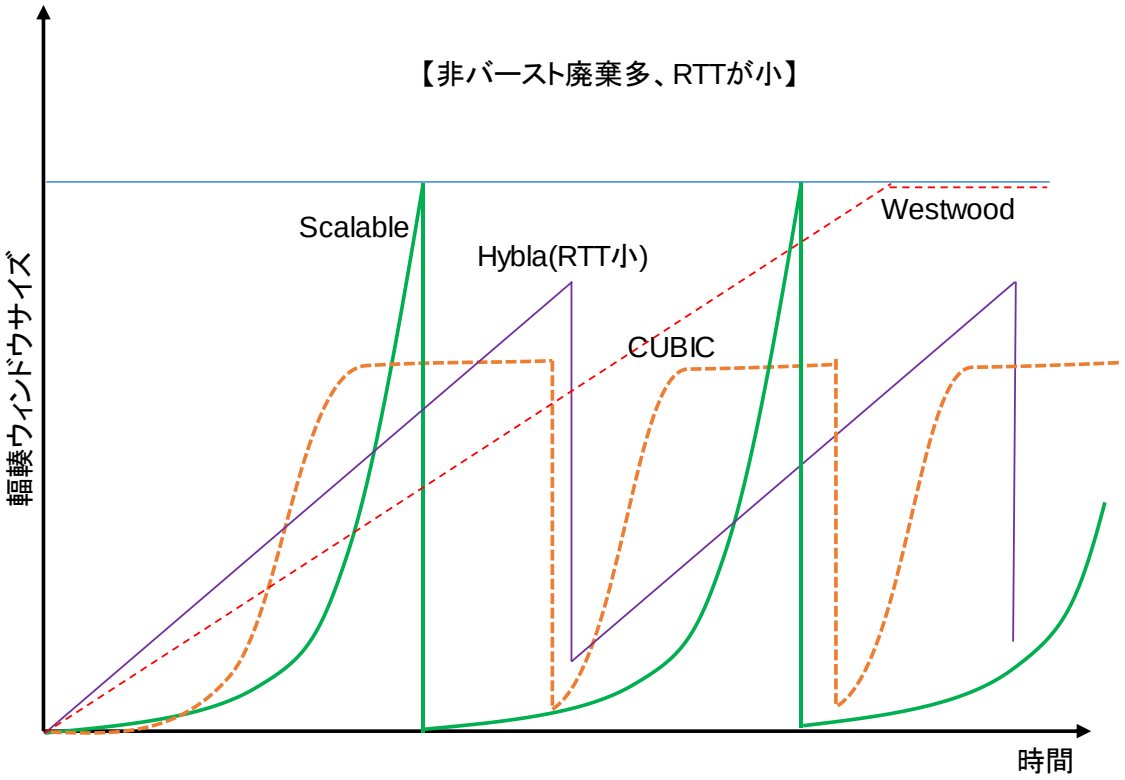


図 3-9 非バースト的廃棄が多く，RTT が小さい場合の各輻輳制御アルゴリズムの挙動

● バーストサイズが大きい通信

RTT が小さい環境における，バーストサイズが大きいデータ通信の場合 (図 3-11 の右下の領域) は，図 3-9 に示した輻輳ウィンドウサイズの平均値がより高くなる輻輳制御アルゴリズムが性能優位となる．TCP Westwood は，輻輳ウィンドウの立ち上がりは他の輻輳制御アルゴリズムに比べ遅いが，非バースト的廃棄では輻輳を検出しにくい特性を持つため，輻輳ウィンドウサイズの平均値が高く，性能優位になりやすい.

● バーストサイズが小さい通信

RTT が小さい環境における，バーストサイズが小さいデータ通信の場合 (図 3-11 の左下の領域) は，図 3-9 に示した輻輳ウィンドウサイズの立ち上がりがより速い輻輳制御アルゴリズムが性能優位となる．非バースト的廃棄により輻輳を検出した場合，輻輳ウィンドウサイズが初期ウィンドウサイズに近い値まで低下するため，少ない送信パケット数でも素早く成長する Scalable TCP がデータ転送のレイテンシの観点で性能優位となる．

3.2.3.2 RTTが大きい場合

TCP Hybla のような RTT に応じて成長率が変化する輻輳制御アルゴリズムは，RTT が大きいため成長が速くなる．非バースト的廃棄率が特に高い場合は，CUBIC TCP や Scalable TCP は，立ち上がりの初期で廃棄を検出するため，TCP Westwood, TCP Reno の方が性能優位になる場合がある．特に TCP Westwood は非バースト的廃棄では輻輳を検出しない場合が多いため，TCP Westwood が優位になりやすい．

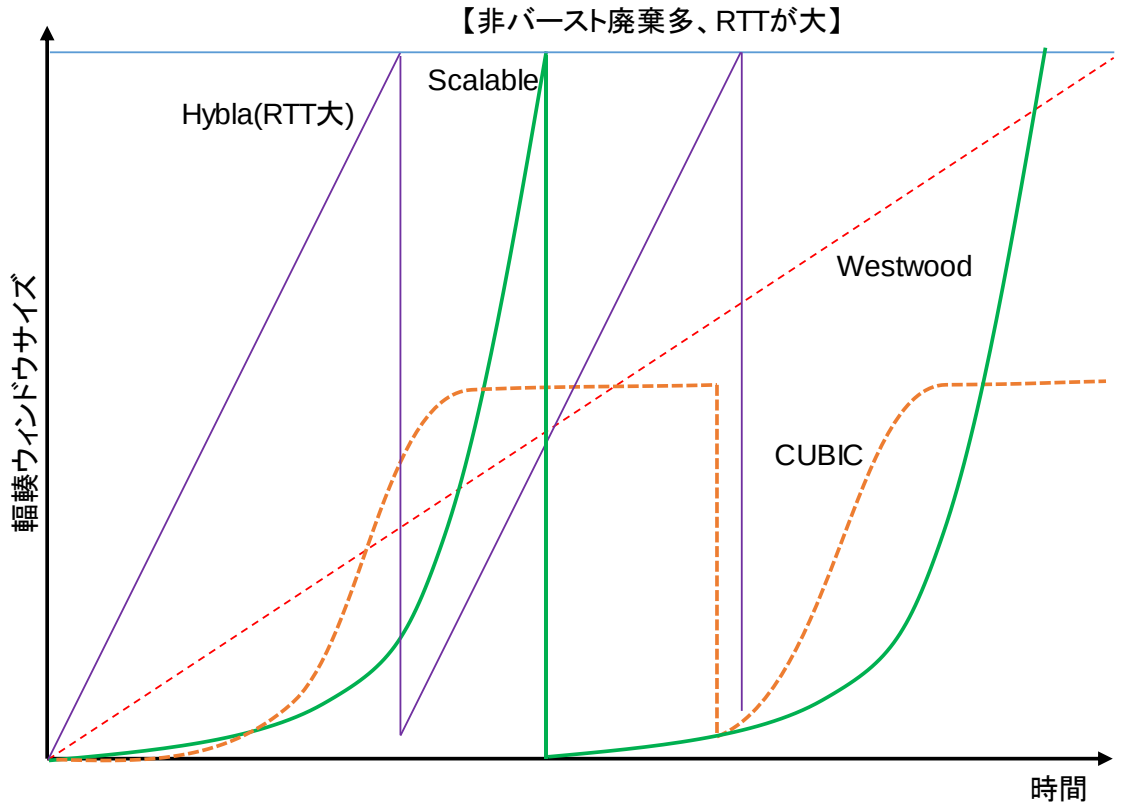


図 3-10 非バースト的廃棄が多く, RTT が大きい場合の各輻輳制御アルゴリズムの挙動

● **バーストサイズが大きい通信**

RTT が大きく，バーストサイズが大きいデータ通信の場合 (図 3-11 の右上の領域) は，図 3-10 で示したように，輻輳を検出しにくい TCP Westwood が性能優位になりやすい.

● **バーストサイズが小さい通信**

RTT が大きく，バーストサイズが小さいデータ通信の場合 (図 3-11 の左上の領域) は，TCP Hybla が輻輳ウィンドウサイズの成長率が大きいためデータ転送のレイテンシも小さい値となる.

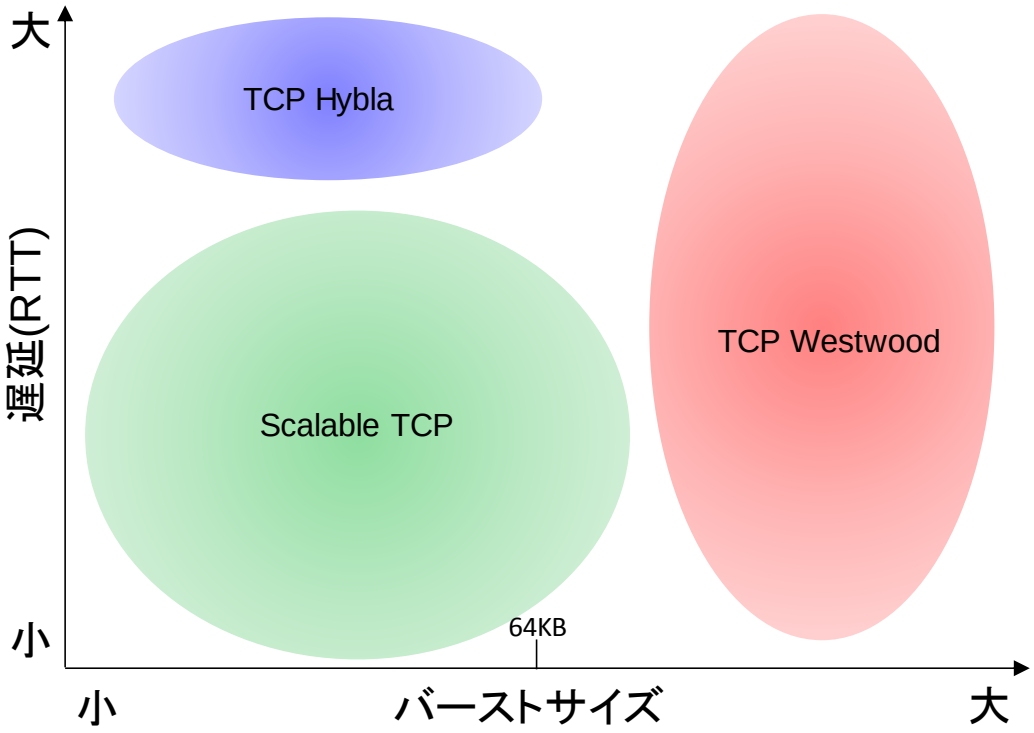


図 3-11 各種 TCP の適応領域 (非バースト的パケット廃棄が多い場合)

3.3 本章のまとめ

本章では，定性的な考察に基づき，通信の種類やネットワークの特性(RTT や廃棄率，廃棄パタン)に応じて最も性能優位となる輻輳制御アルゴリズムが変わることを示した.

大規模ファイル転送のようにバーストサイズが大きい場合は，性能の指標として平均スループットが重視され，小さなデータをやりとりするインタラクテ

ィブ通信では，バーストデータの転送に要するレイテンシが短いことが重視されることを示した．

廃棄が少ない場合は，輻輳による輻輳ウィンドウの減少がほとんどないため，立ち上がり時の成長が速い輻輳制御アルゴリズム(CUBIC TCP，Scalable TCP，TCP Hybla)が良い．

バースト的廃棄が多い場合は，輻輳検出時の輻輳ウィンドウサイズの削減率がより小さい輻輳制御アルゴリズム(Scalable TCP)が性能優位である．

非バースト的廃棄が多い場合は，輻輳ウィンドウサイズを成長させる速度がより大きい輻輳制御アルゴリズム(TCP Hybla，Scalable TCP)が性能優位である．TCP Westwood は，非バースト的廃棄を輻輳と認識しない場合が多いため，バーストサイズが大きい場合においては優位になる．

このように，条件により最適なアルゴリズムが異なる．しかし，今日の多くの OS では, TCP 輻輳制御アルゴリズムはアプリケーションセッションによらず OS 単位に固定であり，セッション毎に異なる輻輳制御アルゴリズムを用いることができない．加えて，アプリケーションの特性や通信品質が大きく変化する環境では，通信の途中で，ユーザがこれらの変化を調べ，適宜輻輳制御アルゴリズムを変更するのは困難である．そのため，ネットワークの通信品質の変化を監視し，通信中であっても動的に性能優位となる輻輳制御アルゴリズムを選択し，切替える仕組みが求められる．

第4章 輻輳制御アルゴリズム動的切り替え手法の検討

本章では，前章までの課題を鑑み，動的に輻輳制御アルゴリズムを切り替える Reconfigurable TCP (R-TCP[1])を提案するとともに, その動作について説明する.

4.1 Reconfigurable TCP の提案

前章では，二つの廃棄パタンにおける廃棄率，及び RTT に応じ，性能優位となる輻輳制御アルゴリズムが変化することを示した．そこで，二つの廃棄パタンを分離して計測し，各廃棄パタンにおける廃棄率と RTT に応じ適切なアルゴリズムを自動選択すれば常に最適な性能で通信を行えると思われる．以下では，廃棄率と RTT の計測，輻輳制御アルゴリズムの選択を自動で行うことにより，インタラクティブ通信におけるデータ転送のレイテンシを最小化する R-TCP を提案する.

4.2 Reconfigurable TCP の動作概要

図 4-1 に Reconfigurable TCP の動作概要を示す．R-TCP は，図 1-4 に示す TCP の ESTAB 状態において，ACK 受信やタイマーのエクスパイアといったイベントをトリガに動作する．既存の TCP は，ACK を受信すると RTT の平均を計算するが，R-TCP はこの RTT の値を参照する．そして，廃棄率計測タイマーがエクスパイアすると各廃棄パタンについて廃棄率を計算する．これらは，3.1 節に示す輻輳検出方法の違いにより分離して計算する．具体的には，Triple 重複 ACK により輻輳を検出した際は，廃棄パケット数を非バースト的廃棄としてカウントする．また，SACK を受信し輻輳を検出した場合は，廃棄パケット数をバースト的廃棄としてカウントする．なお，Triple 重複 ACK や SACK により輻輳を検出する仕組みは TCP の機能を流用する．一方，輻輳制御アルゴリズム更新タイ

[1] 音声や動画などのデータをリアルタイムに転送するための RTP(Real-time Transport Protocol)と呼ばれる伝送プロトコルがある．また，RTP とセットで使われ，実効帯域や遅延時間をデータ送信元に送り，通信状況に合わせて RTP の送信品質を調整するための RTCP(Real-time Transport Control Protocol)と呼ばれるプロトコルがある．ここでは，RTCP と区別するため，Reconfigurable TCP の略称は，R と TCP の間にハイフンを入れた R-TCP と記載する.

マーがエクスパイアすると，後述するレイテンシ特性データベースを参照し，最適な輻輳制御アルゴリズムを選択する．

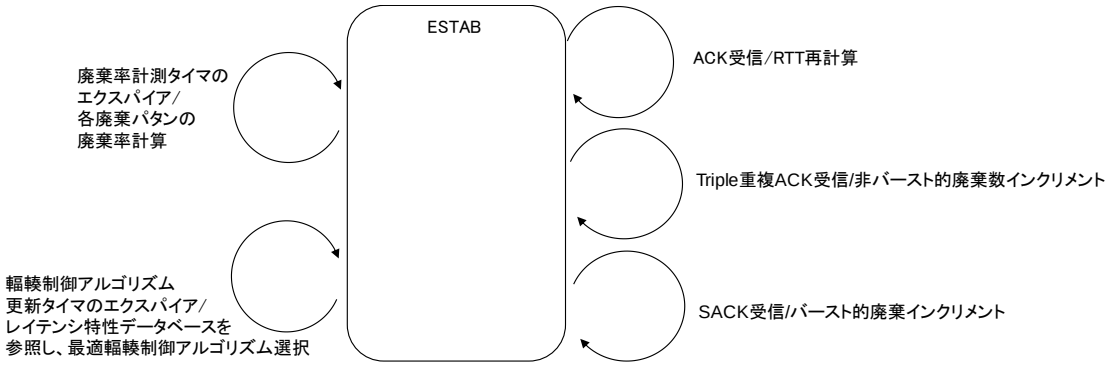


図 4-1 Reconfigurable TCP の動作概要

4.3 各輻輳制御アルゴリズムのレイテンシ特性

R-TCP が備えるレイテンシ特性データベースは，廃棄パタン毎に各輻輳制御アルゴリズムの廃棄率とレイテンシの関係を示したデータベースである．各輻輳制御アルゴリズムのレイテンシ特性は，NS-3 ネットワークシミュレータ[64]に，実コードをシミュレータ内で実行するフレームワークである Network Simulation Cradle (NSC) [65]を適用し, R-TCP を実装した Linux OS のプロトコルスタックをシミュレータ内で直接実行し特性を得ている．

図 4-2 に示すように，NS-3 シミュレータ上に NSC を用いて Linux OS のプロトコルスタック，及び R-TCP を実装し評価する．NSC は Linux OS のカーネルプロトコルスタックを NS-3 に結合するためのライブラリである．NS-3 上のアプリケーションからの送信パケットをソケット層で抜き出し Linux OS のプロトコルスタックのソケット層に挿入し，Linux OS の IP 層から出力されたパケットを NS-3 のプロトコル層に戻す．ネットワークから NS-3 が受信したパケットは NS-3 のプロトコル層で抜き出され Linux OS のプロトコルスタックのデバイス部に渡される．プロトコル処理を行いソケット層まで上がったデータは NS-3 のソケット層に戻される．なお，ここで用いる Linux OS のプロトコルスタックは，ソケット毎に異なる輻輳制御アルゴリズムを使えるように修正してある．

図 4-3 に示すように, R-TCP を実装した Linux OS のプロトコルスタックを持つサーバとクライアント間で，サーバ上のアプリケーションから所定のサイズ

のバーストデータを 2 秒間隔で送信した場合の平均レイテンシを，固定的に選択した 9 種類の輻輳制御アルゴリズムそれぞれについて計測する．この際，サーバとクライアント間のネットワークにおいて，図 4-3 上のようにランダムパケット廃棄を発生させ廃棄率を変化させた場合に，サーバが受信した ACK により検出する非バースト的廃棄率を計測しレイテンシと対応付けてプロットする．

さらに，図 4-3 下のように，中継装置の空き帯域(bandwidth of bottleneck link)を変化させバッファ溢れによるパケット廃棄を発生させた場合に，サーバが受信した SACK により検出するバースト的廃棄率を計測しレイテンシと対応付けてプロットする．また，パケット廃棄に加え，サーバとクライアント間の RTT も複数変化させそれぞれレイテンシ特性を示すグラフを作成する．そして，廃棄率，RTT の各パラメタを変化させ，R-TCP を適用するネットワークの特性にあわせて必要な複数のレイテンシ特性を用意する．

図 4-4 から図 4-8 にレイテンシ特性の例を示す．図 4-4 は RTT が 200ms の時の非バースト的廃棄率に対する各輻輳制御アルゴリズムのレイテンシ特性である．図 4-5，図 4-6 はセットで RTT が 200ms 時のバースト的廃棄率に対する各輻輳制御アルゴリズムのレイテンシを求めるグラフであり，図 4-5 は，現在使用中の輻輳制御アルゴリズムで検出したバースト的廃棄率からネットワークのボトルネックリンクの帯域を推定するグラフである．また，図 4-6 は，得られたボトルネックの帯域に対し，選択対象となる各輻輳制御アルゴリズムのレイテンシを推定するグラフである．また，図 4-7，及び図 4-8 は RTT が 100ms，300ms の場合の非バースト的廃棄率に対する各輻輳制御アルゴリズムのレイテンシ特性であり RTT の大きさによりレイテンシが小さい輻輳制御アルゴリズムが変化する．

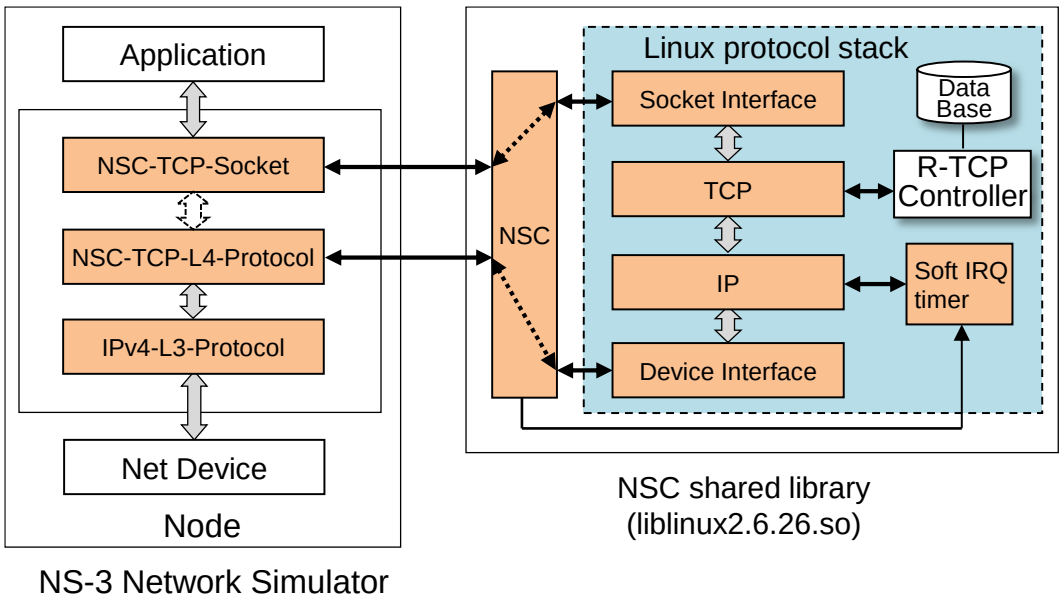


図 4-2 特性グラフ生成のためのシミュレーション方法

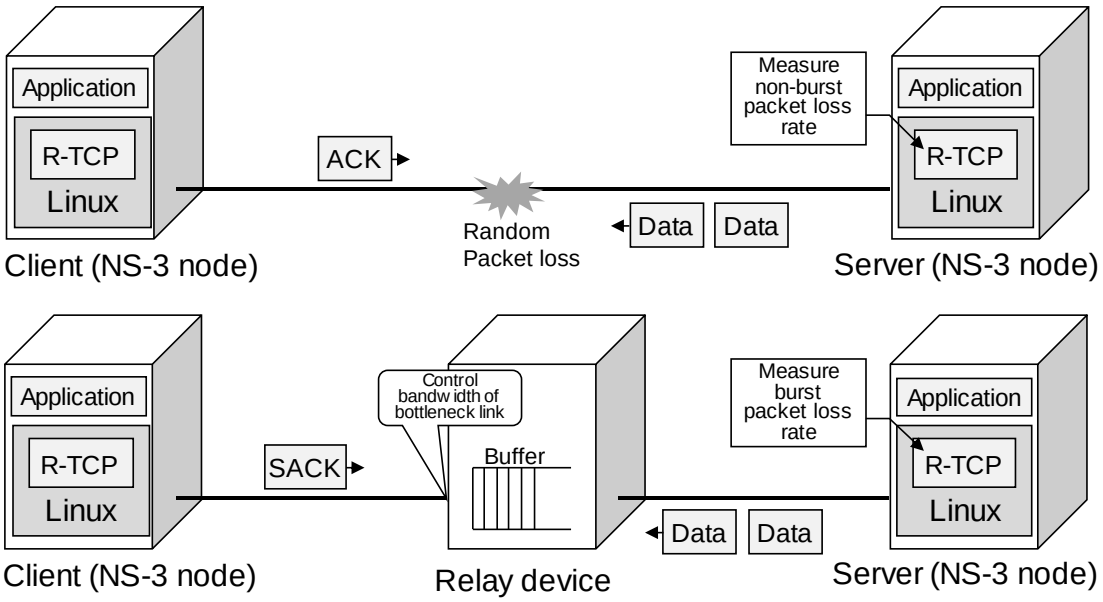


図 4-3 レイテンシ特性評価環境

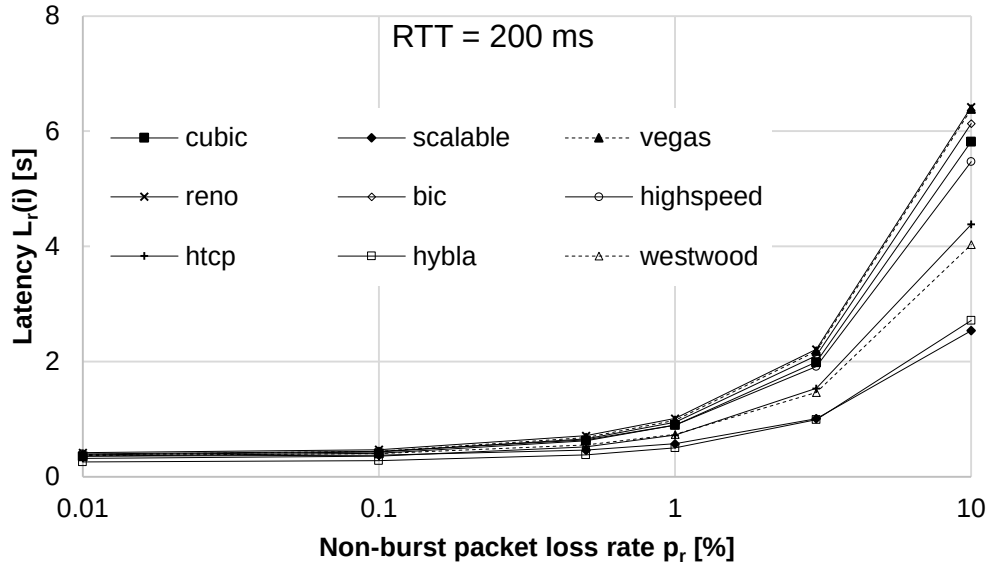


図 4-4 非バースト的廃棄時のレイテンシ特性 [RTT = 200 ms]

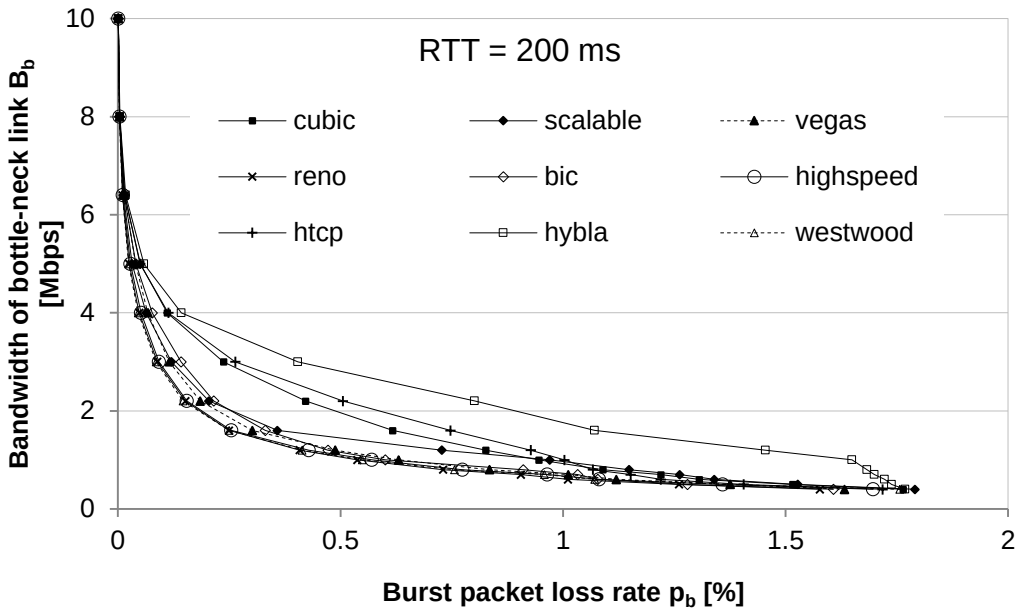


図 4-5 バースト的廃棄時のボトルネックリンク帯域特性 [RTT = 200 ms]

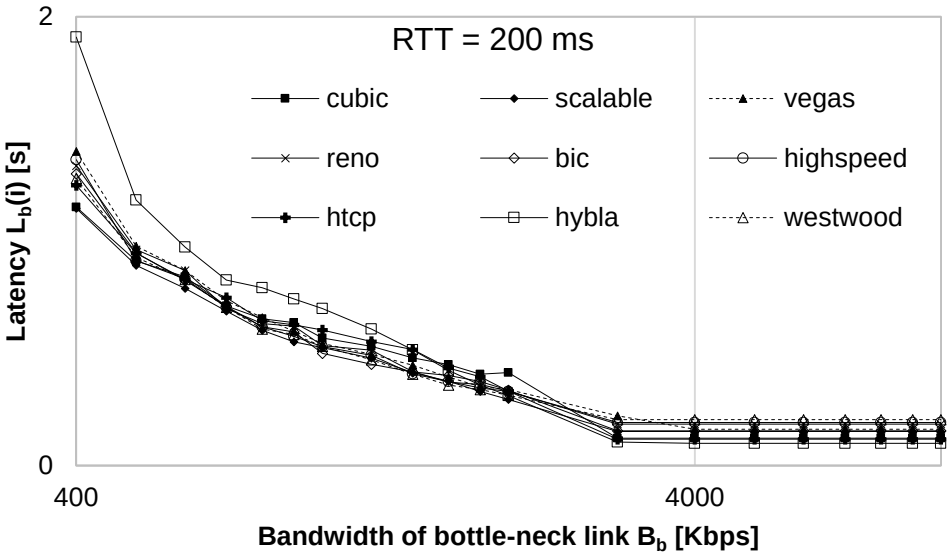


図 4-6 バースト的廃棄時のレイテンシ特性 [RTT = 200 ms]

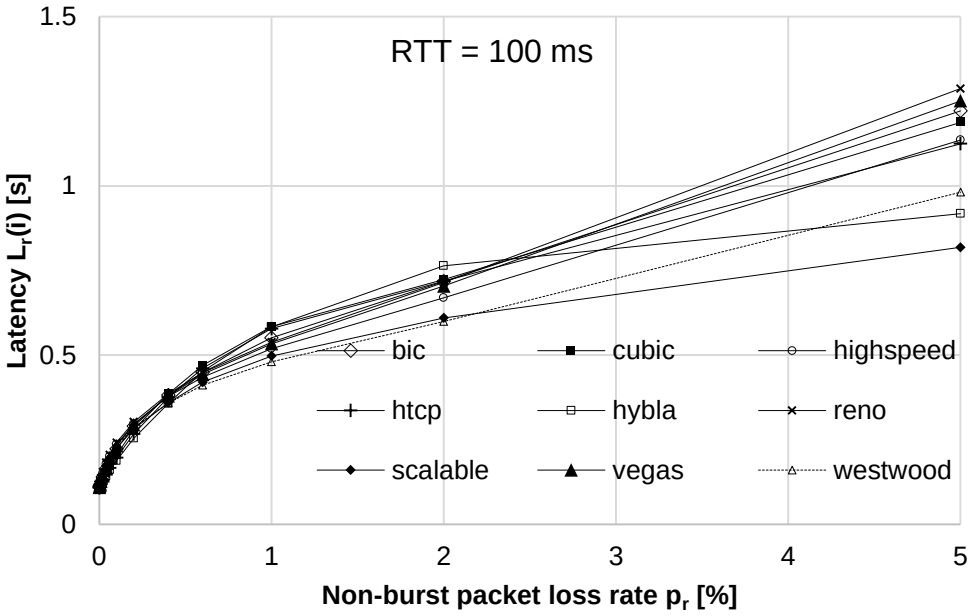


図 4-7 非バースト的廃棄時のレイテンシ特性 [RTT = 100 ms]

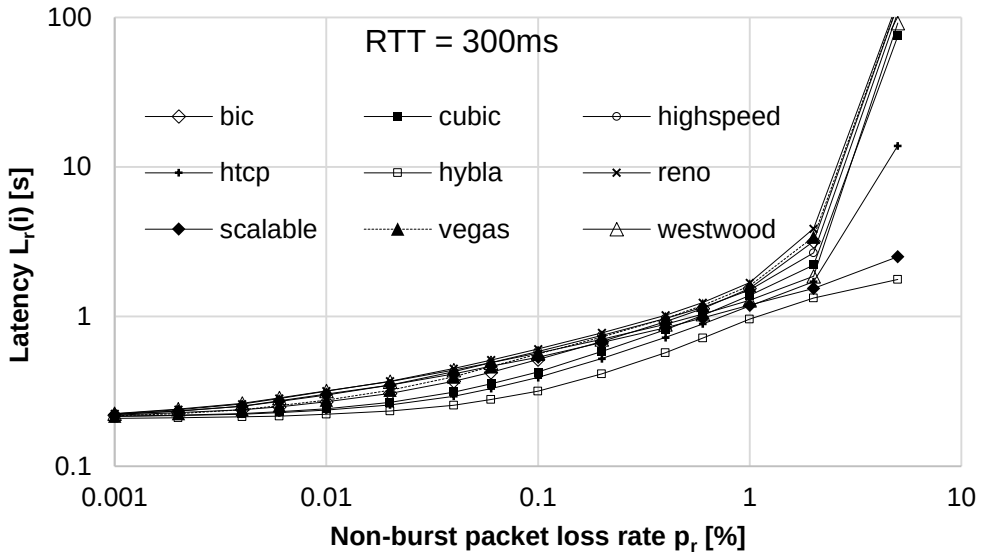


図 4-8 非バースト的廃棄時のレイテンシ特性 [RTT = 300ms]

実際には，これらのレイテンシ特性は，図 4-9 から図 4-17 に示すレイテンシ特性テーブルとして保持する．レイテンシ特性テーブルは，以下に示す 3 種類あり，更に各種類毎に RTT の値が異なるテーブルが存在する．ここでは，RTT が 100ms，200ms，300ms のものを示している．

i. 非バースト的廃棄時のレイテンシ特性テーブル(図 4-9 から図 4-11)

ii. バースト的廃棄時のレイテンシ特性テーブル(ボトルネック帯域推定)(図 4-12 から図 4-14)

iii. バースト的廃棄時のレイテンシ特性テーブル(レイテンシ推定) (図 4-15 から図 4-17)

非バースト的廃棄率	レイテンシ [s]								
	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
0.00001	0.10923	0.108567	0.109883	0.108362	0.107532	0.110369	0.109888	0.108935	0.110907
0.00002	0.112089	0.110267	0.113263	0.109941	0.108886	0.114495	0.113012	0.111479	0.115449
0.00004	0.115596	0.1125	0.118139	0.112044	0.11094	0.120065	0.117619	0.115497	0.121469
0.00006	0.119096	0.115275	0.123463	0.114431	0.113219	0.126238	0.123186	0.119839	0.127887
0.0001	0.126523	0.120392	0.132374	0.119187	0.117837	0.136874	0.132245	0.127643	0.137866
0.0002	0.139287	0.130863	0.14921	0.128905	0.126896	0.154645	0.147233	0.143131	0.155245
0.0004	0.164142	0.151368	0.178054	0.146888	0.142634	0.182888	0.172983	0.170489	0.181042
0.0006	0.186593	0.172567	0.201481	0.166475	0.160447	0.205727	0.19468	0.195551	0.2009
0.001	0.222645	0.204957	0.236591	0.197162	0.188429	0.241701	0.22503	0.232019	0.232988
0.002	0.290256	0.280085	0.296648	0.267386	0.254424	0.302695	0.282873	0.294755	0.286412
0.004	0.381534	0.385655	0.377746	0.375615	0.357385	0.386019	0.359303	0.379799	0.358099
0.006	0.44928	0.469568	0.435007	0.461093	0.451747	0.447585	0.420238	0.443029	0.411355
0.01	0.550732	0.584453	0.518545	0.578134	0.582941	0.538066	0.497337	0.533437	0.4802
0.02	0.717043	0.722897	0.669556	0.718259	0.763414	0.714326	0.609957	0.703765	0.59944
0.05	1.221516	1.187366	1.135122	1.124386	0.917763	1.287149	0.81824	1.251002	0.980929

図 4-9 レイテンシ特性テーブル(RTT=100ms，非バースト的廃棄時)

非バースト的廃棄率	レイテンシ [s]								
	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
0.00001	0.164619	0.162277	0.166251	0.161652	0.158265	0.167845	0.166321	0.163019	0.16882
0.00002	0.170432	0.164896	0.173847	0.164012	0.159878	0.177299	0.17364	0.168393	0.179082
0.00004	0.177484	0.16869	0.185279	0.167418	0.16232	0.190715	0.184906	0.177224	0.193277
0.00006	0.184948	0.173667	0.198404	0.171452	0.165177	0.206225	0.199208	0.187437	0.208673
0.0001	0.19839	0.181366	0.216456	0.178279	0.170154	0.228088	0.217999	0.202547	0.229693
0.0002	0.22026	0.19735	0.246594	0.192042	0.180086	0.259741	0.245336	0.230618	0.259752
0.0004	0.264836	0.230691	0.303424	0.21972	0.200248	0.314211	0.292362	0.282238	0.309198
0.0006	0.304366	0.262923	0.345446	0.246585	0.220945	0.356534	0.32988	0.327624	0.345057
0.001	0.36404	0.313441	0.408489	0.292059	0.255187	0.420999	0.381403	0.392739	0.400518
0.002	0.477896	0.427125	0.514717	0.390194	0.336129	0.529596	0.473441	0.503756	0.490352
0.004	0.635432	0.597391	0.65979	0.54232	0.475651	0.682167	0.596264	0.658936	0.61762
0.006	0.765146	0.741584	0.77329	0.672604	0.602508	0.809346	0.695135	0.783081	0.717805
0.01	0.977615	0.965444	0.95535	0.861405	0.80343	1.016195	0.83409	0.989264	0.866206
0.02	1.465216	1.382941	1.352605	1.175074	1.104619	1.514375	1.064287	1.452702	1.158774
0.05	15.92143	8.231233	5.855927	2.684692	1.446823	29.07076	1.572365	15.84005	3.115075

図 4-10 レイテンシ特性テーブル(RTT=200ms，非バースト的廃棄時)

非バースト的廃棄率	レイテンシ [s]								
	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
0.00001	0.218637	0.215656	0.221568	0.21465	0.208871	0.223908	0.22185	0.216431	0.224868
0.00002	0.227393	0.219129	0.233333	0.217726	0.21065	0.238743	0.233944	0.224648	0.24064
0.00004	0.238108	0.224677	0.251281	0.222359	0.213343	0.260115	0.251864	0.238411	0.262687
0.00006	0.249438	0.231608	0.270557	0.227857	0.216206	0.283088	0.272993	0.253419	0.285168
0.0001	0.270058	0.242219	0.299378	0.237113	0.222053	0.317179	0.302578	0.276846	0.318161
0.0002	0.305862	0.266484	0.348102	0.256817	0.233987	0.369601	0.347554	0.322419	0.367905
0.0004	0.368632	0.312573	0.428738	0.293488	0.255632	0.449242	0.413254	0.395558	0.438099
0.0006	0.424804	0.354481	0.492081	0.330496	0.278598	0.512264	0.466587	0.463106	0.49048
0.001	0.515973	0.425963	0.583972	0.391018	0.317553	0.606892	0.535149	0.560972	0.568971
0.002	0.681965	0.583148	0.74737	0.522849	0.412702	0.778903	0.668993	0.730655	0.703451
0.004	0.924523	0.818461	0.970514	0.725424	0.573358	1.019531	0.84241	0.972245	0.886223
0.006	1.131338	1.021006	1.15586	0.893488	0.717511	1.240175	0.979269	1.188371	1.041276
0.01	1.544273	1.382199	1.505186	1.177329	0.958887	1.685226	1.192086	1.604946	1.288003
0.02	3.190295	2.210167	2.65982	1.710833	1.327766	3.874027	1.544075	3.409993	1.8614
0.05	120.6843	75.10311	108.0646	13.87091	1.767224	128.3403	2.512226	121.8597	91.72299

図 4-11 レイテンシ特性テーブル(RTT=300ms，非バースト的廃棄時)

バースト的廃棄率									ボトルネックリンク帯域幅
bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood	
0.00045	0.00045	0.00045	0.00045	0.00055	0.00045	0.00045	0.00067	0.00055	400
0.00366	0.00378	0.00344	0.00378	0.00400	0.00334	0.00322	0.00388	0.00401	500
0.01375	0.01563	0.01144	0.01575	0.01630	0.01067	0.01220	0.01508	0.01145	600
0.03957	0.05044	0.02920	0.04958	0.05000	0.02430	0.03220	0.04054	0.02275	700
0.07738	0.11071	0.05118	0.11172	0.11288	0.04733	0.06185	0.06836	0.04511	800
0.14179	0.23581	0.09262	0.25065	0.26421	0.08988	0.12089	0.11543	0.08525	1000
0.21672	0.41435	0.15251	0.47223	0.58073	0.14612	0.20340	0.18141	0.14663	1200
0.33225	0.61508	0.25370	0.71887	0.93241	0.24476	0.35639	0.30084	0.24679	1600
0.47311	0.82099	0.42960	0.93414	1.07697	0.40505	0.71861	0.48756	0.41313	2200
0.60033	0.93977	0.56910	1.01330	1.15384	0.53682	0.97113	0.63153	0.54826	3000
0.91604	1.09264	0.78047	1.08990	1.36032	0.73491	1.15869	0.83605	0.76216	4000
1.03578	1.22789	0.98258	1.18265	1.59519	0.92393	1.30194	1.03903	0.97171	5000
1.08639	1.31105	1.10764	1.24901	1.84875	1.03965	1.38136	1.16389	1.10518	6400
1.34375	1.54814	1.44031	1.45864	2.13085	1.33436	1.58635	1.48724	1.44959	8000
1.80744	1.93227	1.92752	1.85354	2.18584	1.79428	1.88461	1.92069	1.97923	10000

図 4-12 レイテンシ特性テーブル(RTT=100ms，バースト的廃棄時ボトルネック帯域推定)

バースト的廃棄率									ボトルネックリンク帯域幅
bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood	
0.00045	0.00045	0.00045	0.00045	0.00055	0.00045	0.00045	0.00056	0.00045	400
0.00378	0.00400	0.00356	0.00400	0.00433	0.00334	0.00345	0.00389	0.00323	500
0.01343	0.01520	0.01067	0.01508	0.01706	0.01022	0.01209	0.01409	0.01033	600
0.04070	0.05043	0.02840	0.04987	0.05863	0.02499	0.03166	0.03942	0.02365	700
0.07676	0.11067	0.05203	0.11314	0.14227	0.04774	0.06324	0.06724	0.04544	800
0.14191	0.23683	0.09229	0.26335	0.40396	0.08837	0.12009	0.11517	0.08544	1000
0.21548	0.42147	0.15484	0.50532	0.80118	0.15125	0.20433	0.18464	0.14617	1200
0.33093	0.61712	0.25496	0.74670	1.07076	0.24940	0.35822	0.30152	0.24943	1600
0.47202	0.82643	0.42892	0.92742	1.45526	0.40831	0.72763	0.48797	0.41338	2200
0.60061	0.94542	0.57072	1.00302	1.64935	0.53772	0.96979	0.63027	0.54873	3000
0.91075	1.08923	0.77381	1.06683	1.68349	0.73075	1.14835	0.83492	0.75578	4000
1.03306	1.21951	0.96486	1.15192	1.69953	0.90520	1.26235	1.01190	0.95814	5000
1.07806	1.30584	1.08119	1.21971	1.72312	1.01108	1.34001	1.11990	1.07090	6400
1.27977	1.51622	1.35889	1.40572	1.73827	1.26039	1.52762	1.37567	1.37460	8000
1.60747	1.76421	1.69660	1.71811	1.76882	1.57684	1.79068	1.63279	1.75648	10000

図 4-13 レイテンシ特性テーブル(RTT=200ms，バースト的廃棄時ボトルネック帯域推定)

バースト的廃棄率									ボトルネックリンク帯域幅
bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood	
0.00045	0.00045	0.00045	0.00045	0.00055	0.00045	0.00045	0.00055	0.00045	400
0.00366	0.00389	0.00344	0.00389	0.00444	0.00344	0.00356	0.00377	0.00322	500
0.01409	0.01541	0.01200	0.01530	0.01902	0.01134	0.01266	0.01421	0.01089	600
0.03904	0.05190	0.02950	0.05178	0.06524	0.02587	0.03461	0.03877	0.02386	700
0.07618	0.11553	0.05326	0.11769	0.17083	0.04788	0.06854	0.06837	0.04641	800
0.12808	0.25493	0.09596	0.27781	0.47694	0.09309	0.13555	0.11889	0.08757	1000
0.21083	0.45522	0.15755	0.53598	0.84583	0.15217	0.24144	0.19265	0.14842	1200
0.30381	0.65822	0.26295	0.74169	1.11575	0.25310	0.50714	0.31661	0.24876	1600
0.45013	0.84242	0.44328	0.86835	1.23608	0.41965	0.86311	0.49992	0.41618	2200
0.58159	0.94190	0.57839	0.93092	1.26101	0.54358	0.96723	0.63477	0.55321	3000
0.90995	1.05212	0.76102	0.99737	1.28195	0.71517	1.07012	0.80504	0.74303	4000
1.00920	1.14200	0.90962	1.07244	1.31018	0.85577	1.15349	0.93636	0.91445	5000
1.03917	1.20892	0.99126	1.12340	1.33688	0.92819	1.22272	1.00768	1.00040	6400
1.14428	1.32855	1.11996	1.26492	1.36123	1.04022	1.35022	1.10264	1.21302	8000
1.26836	1.44400	1.17861	1.45957	1.39309	1.10185	1.55022	1.13515	1.36498	10000

図 4-14 レイテンシ特性テーブル(RTT=300ms，バースト的廃棄時ボトルネック帯域推定)

ボトルネックリンク帯域幅	レイテンシ［s］								
	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
400	0.698047	0.605153	0.681691	0.621326	0.653218	0.708512	0.608587	0.743528	0.652412
500	0.593146	0.52989	0.581509	0.551214	0.722485	0.597886	0.510395	0.632499	0.563662
600	0.525991	0.508866	0.516744	0.533566	0.860629	0.533945	0.47545	0.556388	0.506013
700	0.504582	0.512115	0.499068	0.545717	0.798903	0.509657	0.47587	0.532565	0.484789
800	0.478185	0.528777	0.476529	0.574971	0.79911	0.481226	0.492006	0.494675	0.463754
1000	0.453697	0.584728	0.445167	0.657295	0.798912	0.452839	0.517799	0.46556	0.432555
1200	0.440798	0.635858	0.427938	0.715023	0.794971	0.430789	0.520102	0.447339	0.414752
1600	0.393383	0.597188	0.353085	0.661217	0.732772	0.355178	0.394773	0.386486	0.342116
2200	0.313746	0.440193	0.283138	0.455683	0.459243	0.284084	0.294955	0.303036	0.279029
3000	0.250116	0.285697	0.232655	0.273673	0.252627	0.234069	0.232401	0.240209	0.231633
4000	0.185209	0.188469	0.183165	0.185905	0.170773	0.184437	0.179518	0.186187	0.188023
5000	0.14474	0.143603	0.150208	0.141443	0.135488	0.151612	0.145422	0.150486	0.157442
6400	0.120537	0.119106	0.125545	0.118865	0.116725	0.128917	0.122118	0.121416	0.133273
8000	0.111211	0.110196	0.112986	0.110154	0.109091	0.114387	0.111832	0.110662	0.116026
10000	0.10767	0.107573	0.108034	0.107497	0.106783	0.108305	0.107813	0.107355	0.108709

図 4-15 レイテンシ特性テーブル(RTT=100ms，バースト的廃棄時レイテンシ推定)

ボトルネック	レイテンシ [s]								
リンク帯域幅	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
400	1.333848	1.143401	1.324174	1.12638	1.387423	1.383551	1.106533	1.422877	1.256807
500	1.146427	0.997854	1.130134	1.009345	1.302475	1.173631	0.950505	1.230094	1.091062
600	1.006153	0.920958	0.995485	0.971024	1.307179	1.028678	0.874046	1.076313	0.969808
700	0.954141	0.900433	0.952528	0.978104	1.21982	0.97432	0.86151	1.020963	0.925095
800	0.875141	0.907931	0.884543	0.996948	1.299216	0.904371	0.854651	0.940205	0.861755
1000	0.788243	0.928965	0.809008	1.056798	1.312754	0.821666	0.85097	0.852812	0.781247
1200	0.742385	0.957885	0.748105	1.094008	1.313883	0.759484	0.831492	0.789353	0.720819
1600	0.653235	0.890196	0.618078	0.976669	1.156771	0.627473	0.642666	0.663438	0.599453
2200	0.533269	0.699916	0.499421	0.715102	0.8395	0.503541	0.493279	0.528081	0.489296
3000	0.435507	0.496426	0.410553	0.477074	0.502379	0.412242	0.400943	0.424898	0.407542
4000	0.325097	0.335124	0.328692	0.324933	0.327841	0.337011	0.320139	0.338283	0.338614
5000	0.246574	0.242223	0.25997	0.236598	0.227682	0.269726	0.250302	0.26204	0.282906
6400	0.192884	0.187155	0.20582	0.186559	0.182423	0.217476	0.19845	0.194651	0.230704
8000	0.170112	0.167262	0.176343	0.167335	0.16421	0.180522	0.173609	0.169493	0.186025
10000	0.160934	0.160485	0.161938	0.160324	0.157454	0.162742	0.161273	0.159355	0.164103

図 4-16 レイテンシ特性テーブル(RTT=200ms，バースト的廃棄時レイテンシ推定)

ボトルネック	レイテンシ [s]								
リンク帯域幅	bic	cubic	highspeed	htcp	hybla	reno	scalable	vegas	westwood
400	1.862158	1.588596	1.959736	1.594535	3.190519	2.058751	1.545125	2.06469	1.723447
500	1.645684	1.42013	1.718604	1.454239	2.926264	1.794756	1.367785	1.845772	1.579548
600	1.484703	1.307203	1.517382	1.379722	2.648845	1.57519	1.263629	1.633864	1.435671
700	1.415651	1.273509	1.424403	1.441966	2.539999	1.482268	1.23413	1.550249	1.360896
800	1.261923	1.263397	1.305188	1.425688	2.359619	1.350781	1.202391	1.405154	1.268361
1000	1.115638	1.245125	1.170982	1.499249	2.230695	1.2043	1.187166	1.255492	1.14073
1200	1.029596	1.259433	1.081236	1.522464	2.258746	1.102109	1.172718	1.143493	1.045033
1600	0.861367	1.169779	0.876052	1.316185	1.973338	0.890971	0.969154	0.930443	0.8595
2200	0.724565	0.951693	0.711691	1.023164	1.556855	0.719943	0.711606	0.744876	0.710529
3000	0.587042	0.707146	0.587886	0.710829	0.940443	0.598978	0.572931	0.617055	0.591376
4000	0.47791	0.5142	0.474314	0.507005	0.574452	0.481572	0.456147	0.495985	0.494605
5000	0.359896	0.354979	0.37368	0.34612	0.39271	0.389168	0.350341	0.369415	0.408839
6400	0.272852	0.264518	0.295474	0.263222	0.281337	0.310691	0.280416	0.27763	0.32814
8000	0.230945	0.226326	0.241349	0.22595	0.22034	0.248568	0.236097	0.232195	0.256773
10000	0.214208	0.213437	0.215843	0.213143	0.208466	0.217261	0.214693	0.211697	0.218942

図 4-17 レイテンシ特性テーブル(RTT=300ms，バースト的廃棄時レイテンシ推定)

4.4 廃棄パタンの検出と廃棄率の算出

次に，非バースト的廃棄率，バースト的廃棄率の計測について述べる．R-TCP は，非バースト的，及びバースト的廃棄を Linux の輻輳検出の仕組みを利用して区別する．前述したように，送信側 TCP は高々2 個までの廃棄(図 4-18 中の(a))を，重複 ACK により検出する．一方，連続 3 つ以上の廃棄(図 4-18 中の(b))については，SACK を受信することで検出が可能である．このことから，図 4-18 中(a)と(b)の単位時間当たりの各廃棄パタンの発生回数をカウントすることで，非バースト的廃棄とバースト的廃棄それぞれの廃棄率を算出する．

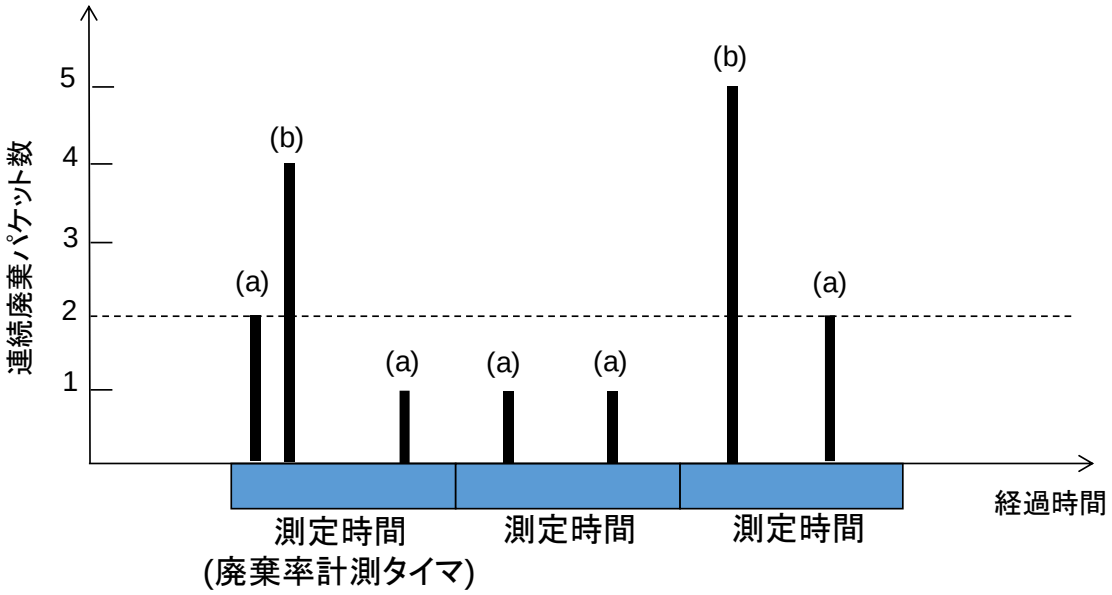


図 4-18 廃棄パタン毎の廃棄率計測

4.5 最適アルゴリズムの選択

既存の TCP のメカニズムにより得られる RTT，及び計測し算出した，非バースト的廃棄率，バースト的廃棄率を用いて以下の 7 つのステップからなる手順で最適な輻輳制御アルゴリズムを選択する．今回は，TCP Reno，Scalable TCP，CUBIC TCP，TCP Vegas，BIC TCP，Highspeed TCP，H-TCP (Hamilton TCP)，TCP Hybla 及び TCP Westwood の 9 種類のアルゴリズムから選択するものとする．

(1) まず，現在用いている輻輳制御アルゴリズムにおける非バースト的廃棄とバースト的廃棄を分けて測定し，それぞれの廃棄率をp_r，p_bとする．

(2) 非バースト的廃棄の廃棄率p_rを非バースト的廃棄時のレイテンシ特性(例えば図 4-4)に当てはめ，各輻輳制御アルゴリズムの期待レイテンシ$L_r(i)$を求める(iは輻輳制御アルゴリズムに固有の番号)．ここで，図 4-4 の特性で示される$L_r(i)$は，廃棄がない場合のレイテンシをL_0，非バースト的廃棄がある場合のレイテンシを$L_{r0}(i)$とした場合に，

$$L_r(i) = (1 - p_r) \cdot L_0 + p_r \cdot L_{r0}(i)$$

を計算したものである．

(3)　現在使用している輻輳制御アルゴリズムで観測したバースト的廃棄の廃棄率p_bを，ボトルネックリンクの空き帯域とバースト的廃棄率との関係グラフ(例えば図 4-5)に当てはめ，網内のボトルネックリンクの空き帯域を推定する(B_b).

(4)　ステップ(3)で得られたボトルネックリンクの空き帯域B_bをバースト的廃棄時のレイテンシ特性(例えば図 4-6)に当てはめ，各アルゴリズムの期待レイテンシ$L_b(i)$を求める．ここで，図 4-6 の特性で示される$L_b(i)$は，バースト的廃棄がある場合のレイテンシを$L_{b0}(i)$とした場合に，

$$L_b(i) = (1 - p_b) \cdot L_0 + p_b \cdot L_{b0}(i)$$

を計算したものである．

(5)　次に，各輻輳制御アルゴリズムを選択するために評価関数，

$$\begin{aligned} L(i) &= L_r(i) + L_b(i) \\ &= L_0 + (1 - p_r - p_b) \cdot L_0 + p_r \cdot L_{r0}(i) + p_b \cdot L_{b0}(i) = L_0 + \bar{L}(i) \end{aligned}$$

を求める．ここで，$\bar{L}(i)$は非バースト的，バースト的廃棄を加味したレイテンシの期待値を示す．本評価関数は，非バースト的廃棄，バースト的廃棄が同時に起こった場合のレイテンシ期待値に，廃棄がない場合のレイテンシL_0を加えた値になるが，L_0は輻輳制御アルゴリズムによらずほぼ同じ値と考えられるため，ステップ(6)に示すレイテンシ最小の輻輳制御アルゴリズムを決めるためにはL_0が加わっていても結果に影響を与えない．

(6)　各輻輳制御アルゴリズムの評価関数$L(i)$を比較し最も小さいレイテンシが期待できる輻輳制御アルゴリズムを選択する．

(7)　(1)から(6)のステップを一定周期毎に実行する．

なお，計測したパラメタの値が，用意したレイテンシ特性上の，測定点の中間や，範囲外になる場合は，最近傍となる値を参照するようにする．

4.6　実装構成

Linux OS をベースとして以下の機能を実装する.

4.6.1　全体構成

図 4-19 に提案方式の全体構成を示す. 本提案方式では, 既存の TCP プロトコルスタックに R-TCP 制御部(R-TCP Control)を設け, 予めシミュレータを用いて求めた, 利用可能な各輻輳制御アルゴリズムの非バースト的廃棄, 及びバースト的廃棄のレイテンシ特性を登録しておく. 次に R-TCP 制御部は, RTT, 非バースト的廃棄率, バースト的廃棄率といったネットワークの品質を計測する. そして, それらの通信品質を, 予め登録した非バースト的廃棄, 及びバースト的廃棄時のレイテンシ特性に当てはめ最も高い通信性能を示す輻輳制御アルゴリズムを選択し, 動的に切り替える.

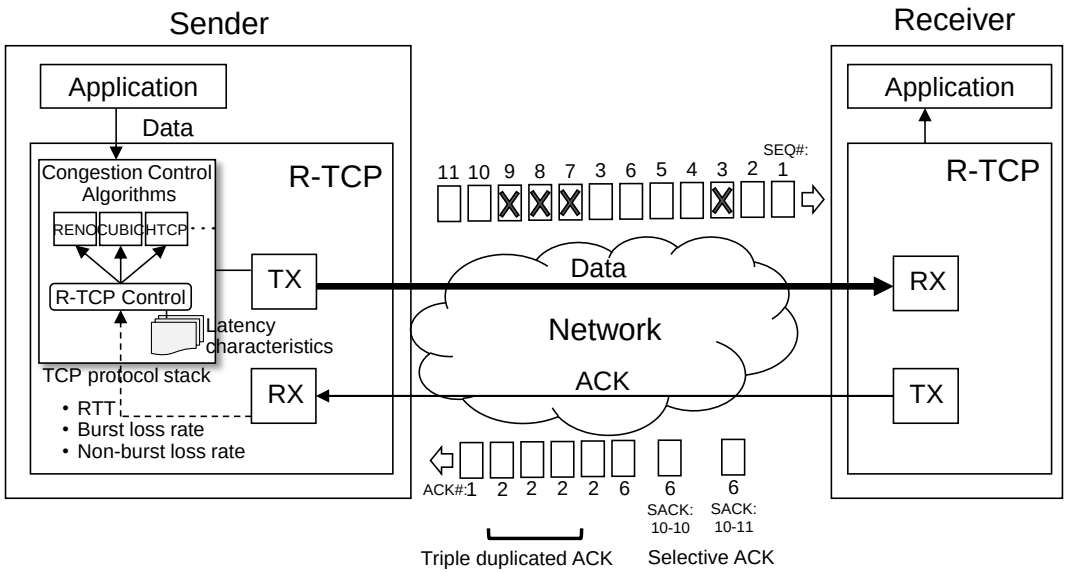


図 4-19 Reconfigurable TCP のコンセプト

また, 1.6.1 節で述べたように異なる通信セッション毎に異なる輻輳制御アルゴリズムを利用可能とするために, 図 4-20 に示すように, 上記のメカニズムをソケット毎に独立に実装する. Linux OS に対する変更としては, ソケット毎に現在使用している輻輳制御アルゴリズムを独立に指定するようにし, 通信セッション毎に廃棄率, RTT を独立して計測するようにしている.

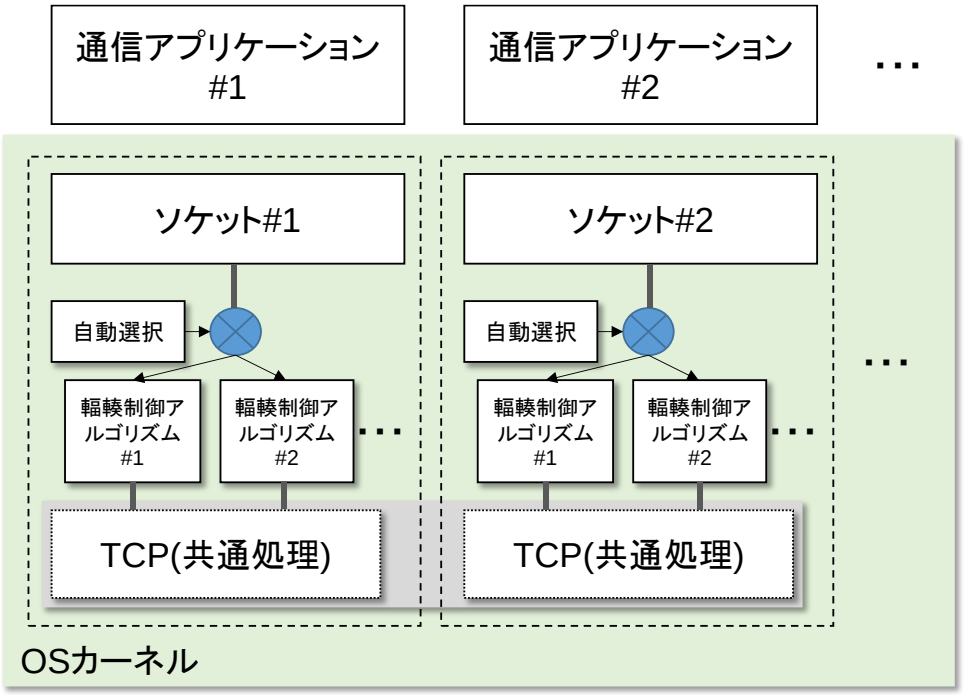


図 4-20 通信アプリケーション毎に最適な輻輳制御アルゴリズムを選択する TCP の構造

4.6.2 動作シーケンス

図 4-21 に示したシーケンス図を用いて，R-TCP の動作を説明する．図 4-21 は，アプリケーション，R-TCP コントロール部と送信側 TCP プロトコルスタック，受信側 TCP プロトコルスタックの間の処理シーケンスを示している．

まず，アプリケーションがソケットを作成するシステムコールを呼ぶことで，TCP プロトコルスタックは，プロトコルコントロールブロック(PCB)と呼ばれる管理情報を生成する．PCB は接続先のアドレスやポート番号などの情報が記録される．また，PCB の一部として，TCP コントロールブロックも生成する．TCP コントロールブロックは，シーケンス番号や，輻輳制御ウィンドウサイズ，再送タイマーなどが記録される．送信側の TCP プロトコルスタックは，PCB を作成すると R-TCP コントロールに通知する．R-TCP コントロールは，予め設定してあるデフォルトの輻輳制御アルゴリズムを TCP プロトコルスタックに設定する．

TCP プロトコルスタックは，3 ウェイハンドシェークにより受信側 TCP プロトコルスタックとの間にコネクションを開設する．

アプリケーションがデータを送信するためにソケットバッファにデータを書

き込むと，TCP プロトコルスタックは，複数の TCP データパケットに分割して送信する．送信側の TCP プロトコルスタックは，受信側の TCP プロトコルスタックから ACK を受信した場合は，RTT の計算値を更新する．送信したデータパケットに対し，非バースト的廃棄が発生した場合は受信側の TCP プロトコルスタックはトリプル重複 ACK を返す．この場合は，RTT の測定(図示省略)と共に非バースト的廃棄のカウンタをインクリメントする．また，送信データパケットに対し，バースト的廃棄が発生した場合は受信側の TCP プロトコルスタックは SACK を送信する．SACK を受信した送信側プロトコルスタックは，バースト的廃棄のカウンタをインクリメントする．そして，廃棄率計算タイマーがエクスパイアすると非バースト的廃棄率，バースト的廃棄率を計算する．

以下，同様に送信側 TCP プロトコルスタックは，アプリケーションからソケットバッファへ書き込みがある毎に，R-TCP コントロール部が指定した輻輳制御アルゴリズムを用いて計算した輻輳ウィンドウサイズに従い TCP パケットを送信する．

ここで，輻輳制御アルゴリズム更新タイマーがエクスパイアすると，送信側 TCP プロトコルスタックで計測した，RTT，バースト的廃棄率，非バースト的廃棄率と，4.3 節に示すレイテンシ特性テーブルを基に最適な輻輳制御アルゴリズムの選択を行う．選択の結果，現在使っている輻輳制御アルゴリズムと異なる場合は, 送信側 TCP プロトコルスタックの輻輳制御アルゴリズムを切り替える．

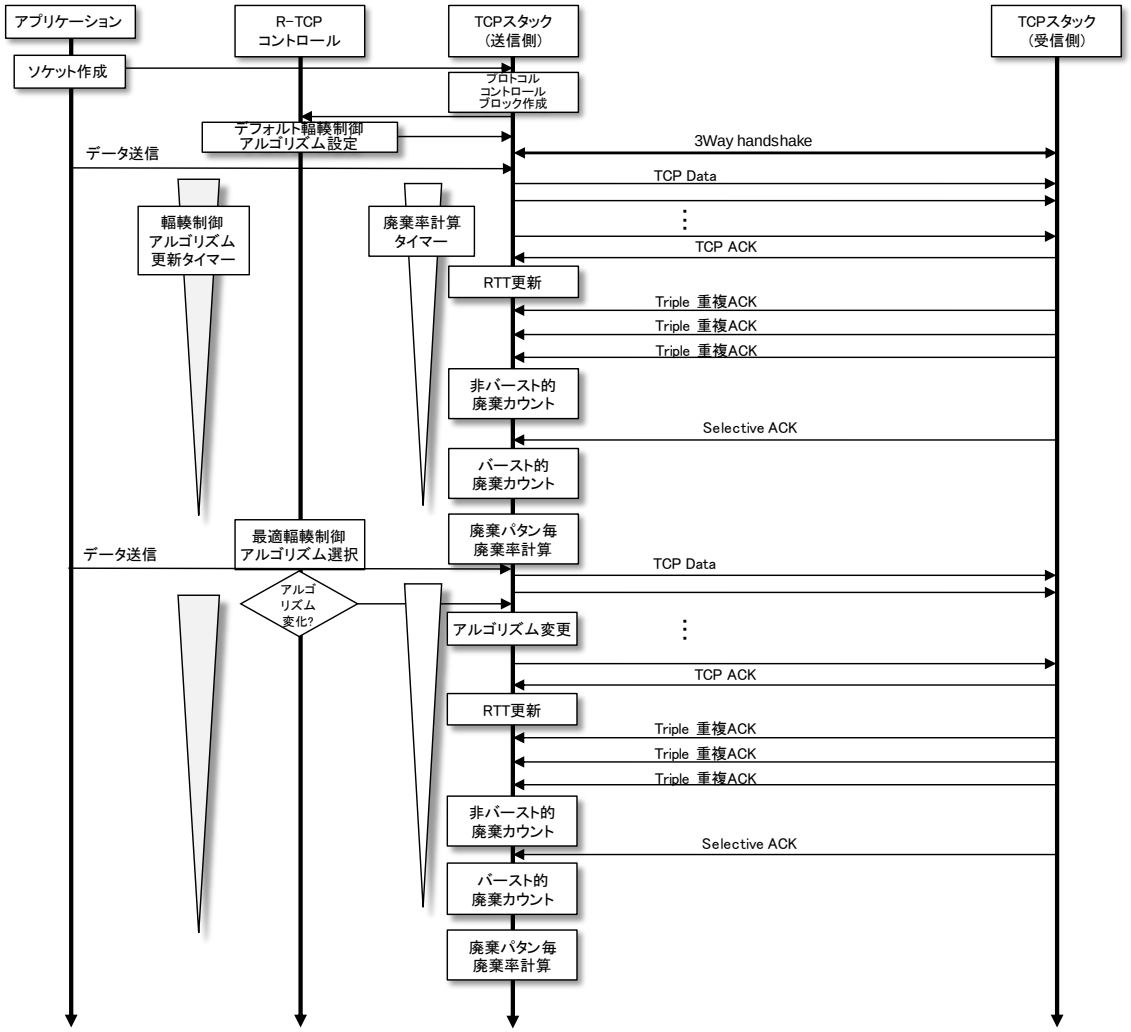


図 4-21 R-TCP の動作シーケンス

4.6.3 動作フローチャート

図 4-22 に，R-TCP の動作フローチャートを示す．R-TCP はタイマーイベントを起点に動作が始まる．タイマーイベントとしては，廃棄率計算タイマーと，輻輳制御アルゴリズム更新タイマーがある．

廃棄率計算タイマーは，4.4 節で述べた，単位時間当たりの非バースト的廃棄率と，バースト的廃棄率を計測するための単位時間を計測するタイマーである．本タイマーがエクスパイアすると，図 4-22 中のステップ S-1，S-2 において，4.4 節で述べる方法で非バースト的廃棄率p_rと，バースト的廃棄率p_bを計算する．この際，以前の計算値との移動平均をとる．その後，廃棄率計算タイマーを再度セット(S-3)しステップ S-0 に戻る．

輻輳制御アルゴリズム更新タイマーがエクスパイアすると，RTT，及び S-1 で予め計算した非バースト的廃棄率p_rを参照し(ステップ S-10，S-11)，4.4 節で詳述した非バースト的廃棄時のレイテンシ特性テーブルの中から最も近い RTT に対応するテーブルを参照し，S-11 で求めた非バースト的廃棄率における各輻輳制御アルゴリズムのレイテンシ$L_r(i)$(iは輻輳制御アルゴリズムを識別する番号を示す)を求める(S-12)．次に，S-2 で予め計算したバースト的廃棄率p_bを参照し(ステップ S-13)，まず，4.3 節で詳述したバースト的廃棄時のボトルネックリンク帯域特性の中から最も近い RTT に対応するテーブルを参照し，現在使用中の輻輳制御アルゴリズムにおけるバースト的廃棄率に対するボトルネックリンク帯域B_bを推定する(S-14)．次に，バースト的廃棄時のレイテンシ特性テーブル(ボトルネックリンク帯域対レイテンシ)を参照し，S-14 で推定したボトルネックリンク帯域B_bに対する各輻輳制御アルゴリズムのレイテンシ$L_b(i)$を求める(S-15)．ここで評価関数$L(i) = L_r(i) + L_b(i)$を計算し(S-16)，評価関数が最小となる輻輳制御アルゴリズムが，現在使用中の輻輳制御アルゴリズムと異なるかどうか判定する(S-17)．S-17 で YES の場合，現在の輻輳制御アルゴリズムが使用していた，シーケンス番号，ACK 番号，輻輳ウィンドウサイズ，スロースタート閾値(ssthresh)などのパラメタを保持し(S-18)，輻輳制御アルゴリズムを S-17 で決定した輻輳制御アルゴリズムに切り替える(S-19)．最後に輻輳制御アルゴリズム更新タイマーをリセットし(S-20)，イベント待ち(S-0)へ戻る．S-17 で NO の場合，現在の輻輳制御アルゴリズムを変更せずに輻輳制御アルゴリズム更新タイマーをリセットし(S-20)，イベント待ち(S-0)へ戻る．

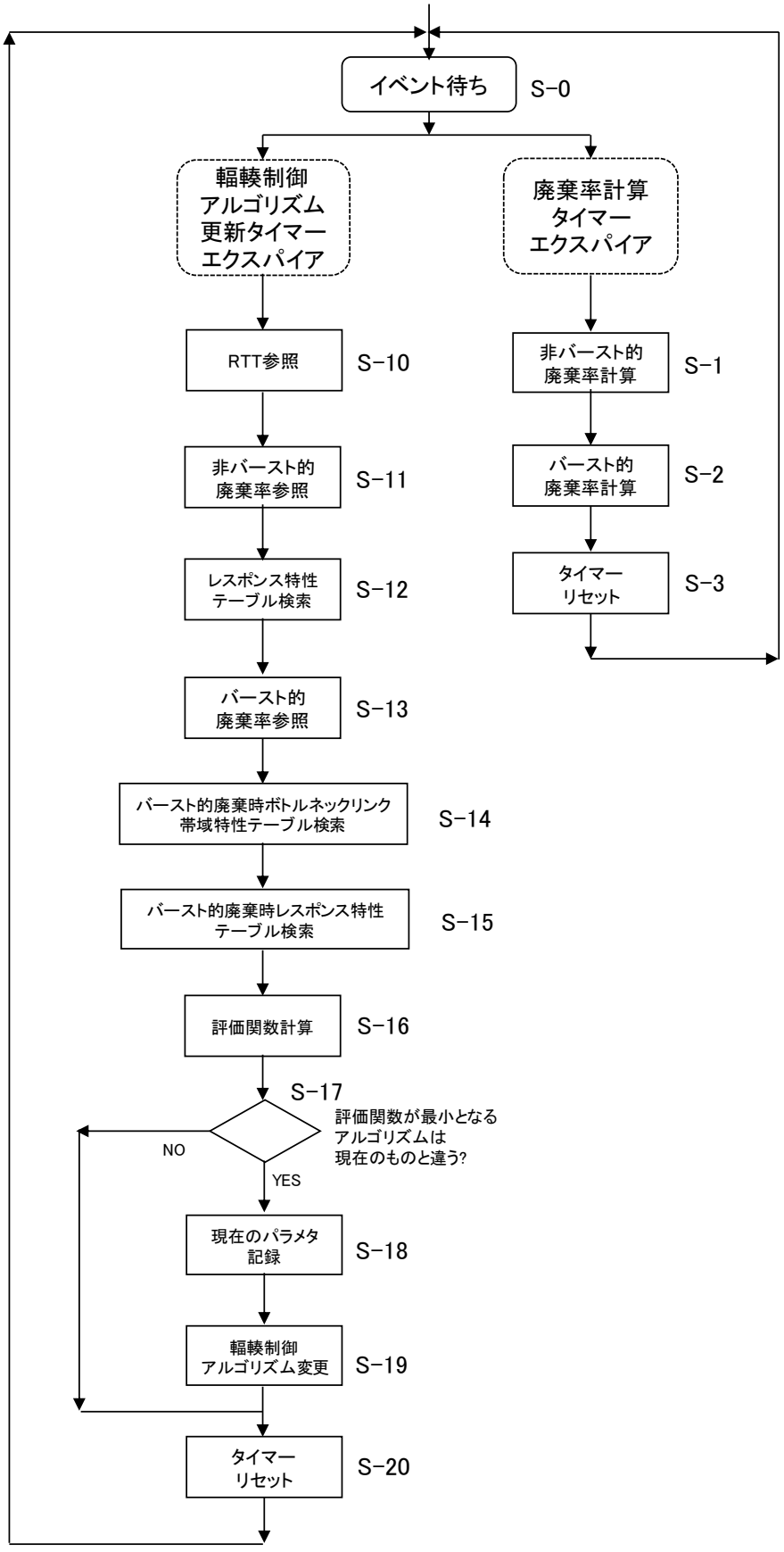


図 4-22 R-TCP のフローチャート

4.7　R-TCP の公平性

R-TCP を多くのユーザがインタネット上で利用することを踏まえ，公平性について考慮しておく必要がある．公平性には，主なものとして TCP friendliness と RTT fairness の二つの観点がある[48].

(1) TCP friendliness は，図 4-23 に示すように，異なる輻輳制御アルゴリズムを持つ複数の TCP セッションが一つの中継装置のバッファなどをシェアした際，使用しているアルゴリズムに応じて，性能が低く抑えられる通信セッションが生じる不公平性のうち，標準の輻輳制御アルゴリズムである TCP Reno と他の輻輳制御アルゴリズムとの間の不公平性である．

(2) RTT fairness は，図 4-24 に示すように，同一輻輳制御アルゴリズムで制御される複数の TCP セッションが一つの中継装置のバッファなどをシェアした際，通信セッションの RTT に応じて，性能が低く抑えられる通信セッションが生じる不公平性である．

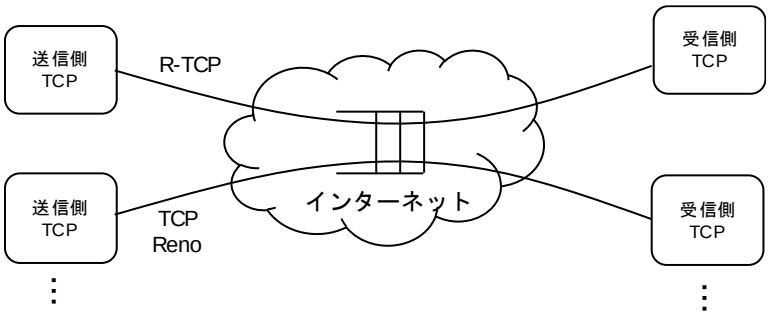


図 4-23 TCP friendliness

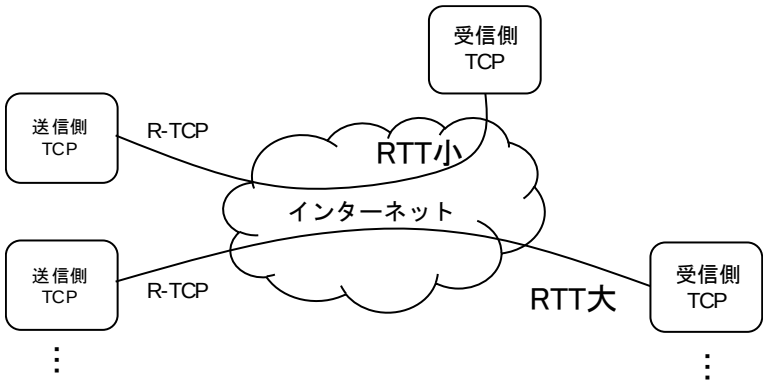


図 4-24 RTT fairness

4.7.1 R-TCP における TCP friendliness

各輻輳制御アルゴリズムは，廃棄率に対するスループットの特性をレスポンス関数と呼ばれる解析解として表すことができる．一方，TCP friendliness については, 多くの輻輳制御アルゴリズムが, 輻輳状態において標準 TCP である TCP Reno に近い挙動になるような工夫(2.5 章参照)をしている. こうした工夫により, 各 TCP 輻輳制御アルゴリズムのレスポンス関数では, 高廃棄率の際に TCP Reno とほぼ同等のスループットになる．図 4-25 は各輻輳制御アルゴリズムのレスポンス関数を示している．廃棄率が低い（すなわち輻輳していない）場合は，各輻輳制御アルゴリズムが改善された性能を示すが，廃棄率が高い（すなわち輻輳している）場合は，TCP Reno と同等の性能に近づく (中には TCP Reno に近づかない輻輳制御アルゴリズムもある)．従って，R-TCP をインタネットで使用する場合などフェアネスの配慮が必要な場合，R-TCP が自動選択する対象の輻輳制御アルゴリズムとして，輻輳時でも TCP Reno に近い挙動を示すものだけに限定し，その中から選択するようにすれば，輻輳発生時にも TCP Reno に近い性能となり, R-TCP 利用時も TCP friendliness を考慮した動作ができると考えられる．

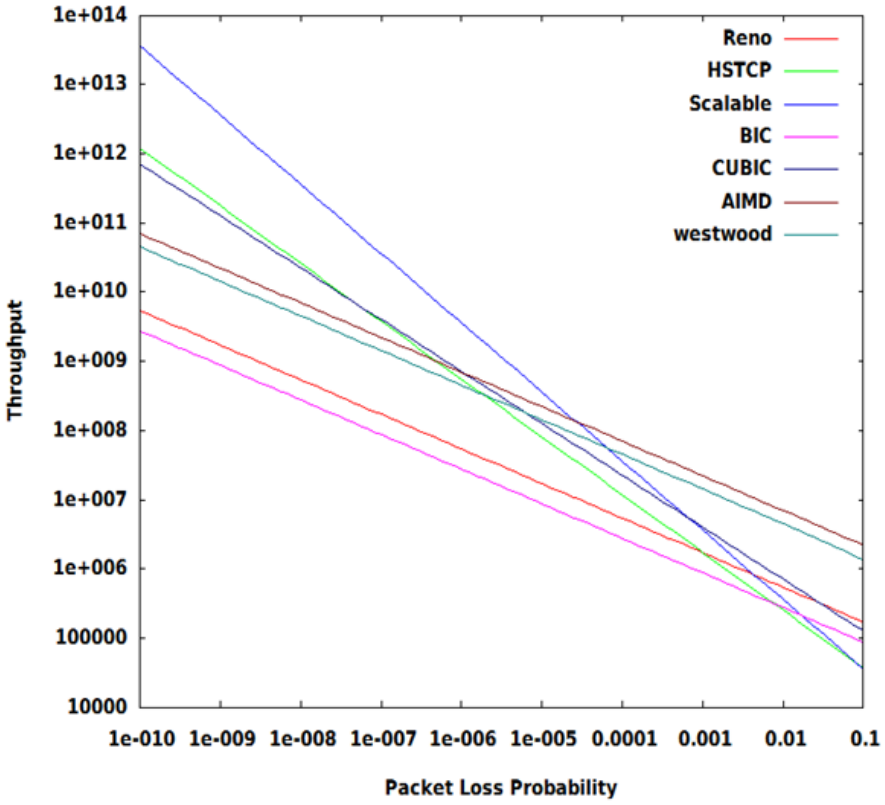


図 4-25 各種 TCP 輻輳制御アルゴリズムのレスポンス関数

4.7.2 R-TCP における RTT fairness

R-TCP を用いた通信セッションが複数ある場合，R-TCP は，RTT の違いによる性能の違いを判定し，より高い性能を示す輻輳制御アルゴリズムを選ぶことができる．そのため，R-TCP は，RTT の違いによるフェアネスを大きく改善できる．具体的に，R-TCP による RTT fairness の改善効果を図 4-26 を用いて説明する．図 4-26 は，輻輳により非バースト的廃棄率が 5%発生している時の RTT の増加に対するレイテンシの増加を，CUBIC と R-TCP について示している．CUBIC は, RTT が 100ms の場合は, レイテンシが小さい値であるが, RTT が 300ms に増加するとレイテンシが急激に悪化する．一方 R-TCP は RTT が増加しても輻輳制御アルゴリズムを切り替えることで，レイテンシを短く保つことができる．図 4-26 中 R-TCP のマーカー付近に, 選択している輻輳制御アルゴリズム名を示す．このため，RTT が異なる R-TCP の通信セッションが複数存在しても，ほぼ同等のレイテンシを保つことができると思われる．

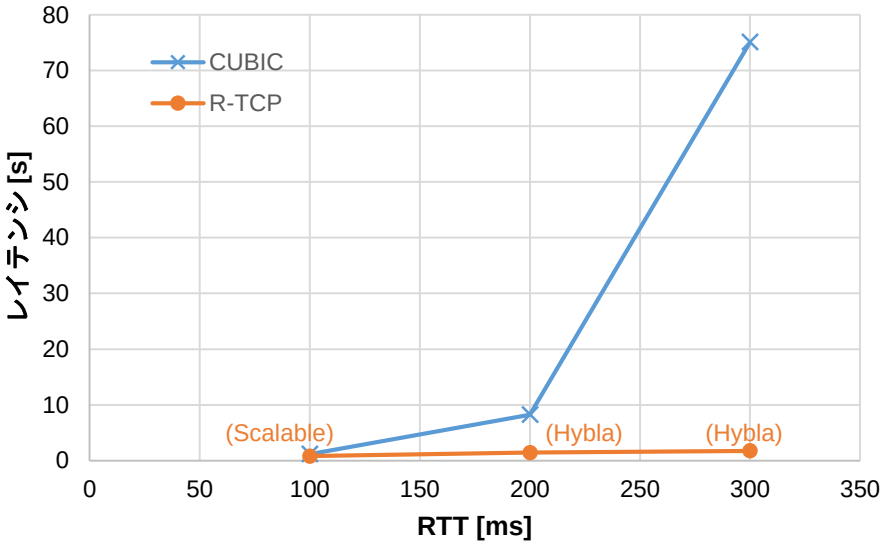


図 4-26 高廃棄率環境下での RTT によるレイテンシの変化

4.8 本章のまとめ

本章では，輻輳制御アルゴリズムをダイナミックに切替える Reconfigurable TCP を提案すると共に，Linux OS における実装構成を示した．

具体的には，各輻輳制御アルゴリズムのバースト的廃棄，非バースト的廃棄時におけるレイテンシ特性を，シミュレータを用いて調べ，どの廃棄率においてどの輻輳制御アルゴリズムが性能優位かを示した．

また，ネットワーク内で混在するバースト的廃棄，非バースト的廃棄を区別して計測する方法として，Linux OS の輻輳検出の仕組みを利用して区別する方法を示した．すなわち，送信側 TCP は高々2 個までの廃棄を重複 ACK により検出するが，連続 3 つ以上の廃棄については，SACK を受信することで検出する．この仕組みを利用してどちらの廃棄がどれくらいの割合で発生しているかを計測する．そして，バースト的廃棄，非バースト的廃棄を区別して計測し，その廃棄率に応じて予め調べたレイテンシ特性に基づき最適な輻輳制御アルゴリズムを選択する方法を示した．

さらに，提案方式がインタネット上で利用されることを踏まえ，公平性に関する考察を行った．その結果，TCP friendliness については，選択対象のアルゴ

リズムを限定するという条件ならば公平性の差を小さくできることがわかった．RTT fairness については，R-TCP は RTT によるレイテンシの差がでにくいため，公平性の差がでにくいことがわかった．

第5章 シミュレーションによる性能評価

本章では，固定無線環境における，バースト的な廃棄をモデル化したチャネルモデルである GE(Gilbert-Eliott)モデル[49][50][51]を用いたシミュレーション評価，及び移動無線環境におけるシミュレーション評価を行い，提案方式のレイテンシ改善効果があることを確認する．

5.1 固定無線環境のシミュレーション

図 5-1 に示すように，Wi-Fi 網(IEEE 802.11g)で接続された移動デバイス(Mobile Device)が有線網の先にあるサーバに対し無線基地局(Access Point)を介して接続し，仮想デスクトップを利用しサーバのデスクトップ画面を移動デバイスに表示する場面を想定し性能評価を行った．バースト的な廃棄を発生させやすいように，Wi-Fi 網におけるチャネルモデルとして，GE モデルを用いた．

本評価では，移動デバイスに表示したデスクトップ画面に対するマウス操作やキーボード操作の信号を受けてサーバが更新画面情報を無線端末に転送している場合を想定している．サーバは移動デバイスに対し平均 64KB のサイズのバーストデータを平均 2 秒間隔で送信する．有線網における RTT は 200ms としている．

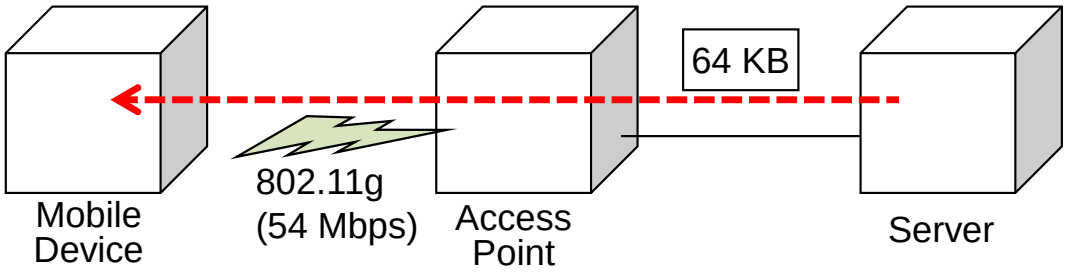


図 5-1 シミュレーションモデル

5.1.1 GE モデル

本シミュレーション評価では，図 4-2 に示したプロトコルスタックを，NS-3 ネットワークシミュレータを用いて実装した．また，IEEE802.11g の無線リンクのチャネルモデルとして GE モデルを実装し，次節に示す廃棄確率でパケット廃棄を発生させた．

GE モデルは, Edgar Gilbert と E.O.Elliot に提案された, G(Good)状態と B(Bad)状態の二つの状態を持つマルコフチェーンに基づくチャネルモデルである．GE モデルでは，図 5-2 に示すように Good 状態では，$1-k$の確率でパケット廃棄が生じる．また Bad 状態では$1-h$の確率でパケット廃棄が生じる．また，Good 状態から Bad 状態へは確率pで遷移し, Bad 状態から Good 状態へは確率rで遷移する．GE モデルは，これら 4 つのパラメタにより様々な廃棄特性を表現できるが，本評価では，GE モデルを特徴付けるパラメタを実際の Wi-Fi 網から実測した廃棄特性に基づき決定する．

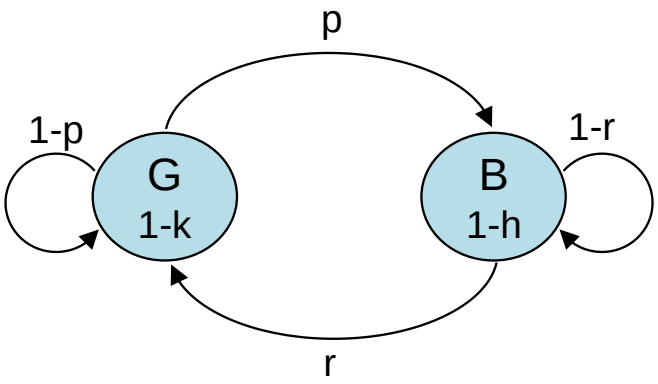


図 5-2 GE モデル

GE モデルのパラメタを，実際の Wi-Fi 網の回線の廃棄特性から[68]に示す手法を用いて決定した．

本手法によれば, GE モデルの変動係数(Coefficient of variation)$C_v(N) = \sigma(N)/\mu$は, p,r,k,hを含む式(5-1)のように表現できる．ここでNは送信したパケット数, $\sigma(N)$はNパケット送信時のスループットの標準偏差，μは平均スループットである．

$$c_v(N) = \left.\begin{aligned} &\frac{1}{\sqrt{N}}\sqrt{\frac{hp+kr}{\omega}+\frac{2pr(1-p-r)(h-k)^2}{\omega^2(p+r)}\left(1-\frac{1-(1-p-r)^N}{N(p+r)}\right)} \\ &where\ \ \omega=(1-h)p+(1-k)r \end{aligned}\right\} \qquad (5\text{-}1)$$

従って，Nの値をいくつか変化させた時の$C_v(N)$を実測値から計算し，式(5-1)をこの特性にフィットさせればGEモデルにおけるp,r,k,hの各値を決定できる．

そこで，図 5-1 においてサーバから移動デバイスに UDP パケットを 30Mbps の一定レートで送信し，移動デバイスにおいて受信するパケットをキャプチャし，キャプチャ結果から単位時間当たりのパケット数，その時間におけるスループットの標準偏差，平均スループットを計算する．

移動デバイスは，集合住宅内の Wi-Fi アクセスポイントから電波を受信する．集合住宅のため，他のアクセスポイントからの干渉もあり，同一チャネルへの干渉は 3 波あった．測定は，他チャネルのトラフィックが比較的多くなる夜間 22:00 から 23:00 の間に行った．

各パケットにはシーケンス番号が含まれており，約 1 時間転送したパケットをキャプチャし，横軸に経過時間，縦軸にシーケンス番号を取りプロットすると，図 5-3 のようになる．また，続けて廃棄されたパケット数も併せてプロットしている．この結果によるとバースト的にパケット廃棄が多く発生していることがわかる．

全観測時間を表 5-1 に示すより短いいくつかの観測時間毎に，平均スループット，標準偏差を求め，これから変動係数$C_v(N)$を計算する．

式(5-1)のフィッティングには数値解析言語である Gnu Octave[52]上に Levenberg-Marquardt (LM)法[53][54]をプログラムした．実測値から計算した値と，式(5-1)をフィッティングした結果は図 5-4 のようになる．概ね誤差範囲に含まれ，的確にフィッティングが行われていると考えられる．この結果から，GE モデルにおけるp, r, k, hの各値は表 5-2 に示す値を得た．

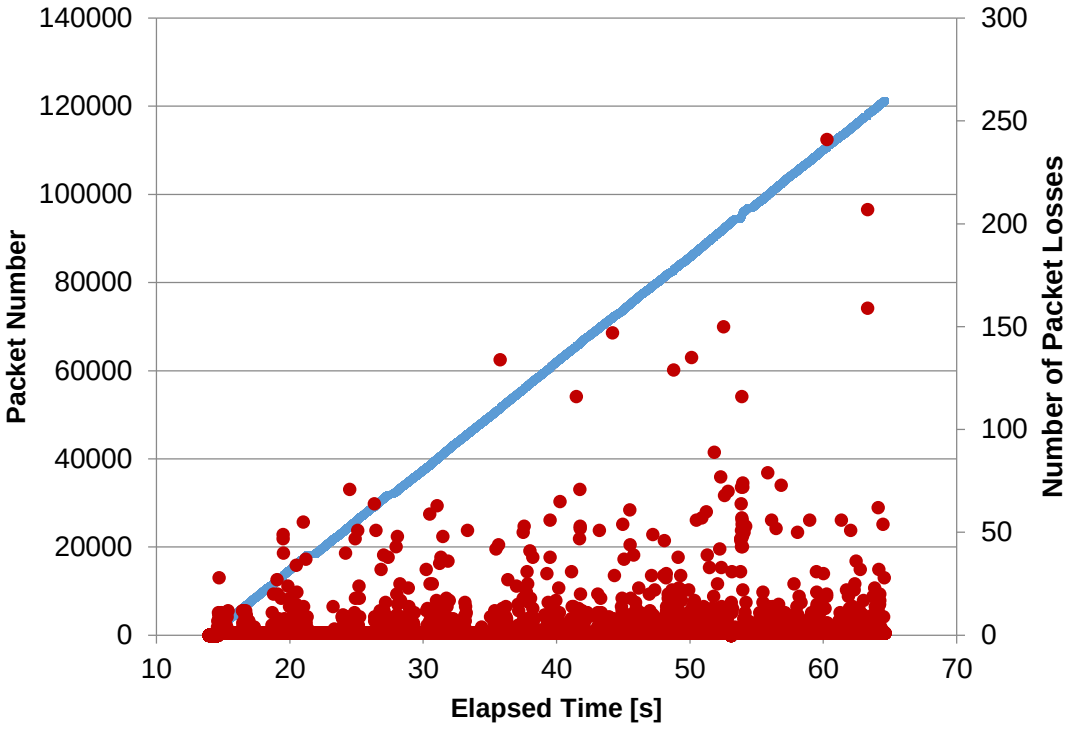


図 5-3 実 Wi-Fi 網におけるパケット廃棄

表 5-1 実 Wi-Fi 網における統計値

Number of Packets N	10	100	1000	10000
Observing Time (s)	0.00392	0.0392	0.392	3.92
Average Throughput (Mbps)	27.1	27.1	27.1	27.1
Standard deviation $\sigma(N)$	13753648	5013244	2971110	1243709
Coefficient of variation $C_v(N)$	0.735155	0.18469	0.1094	0.0457

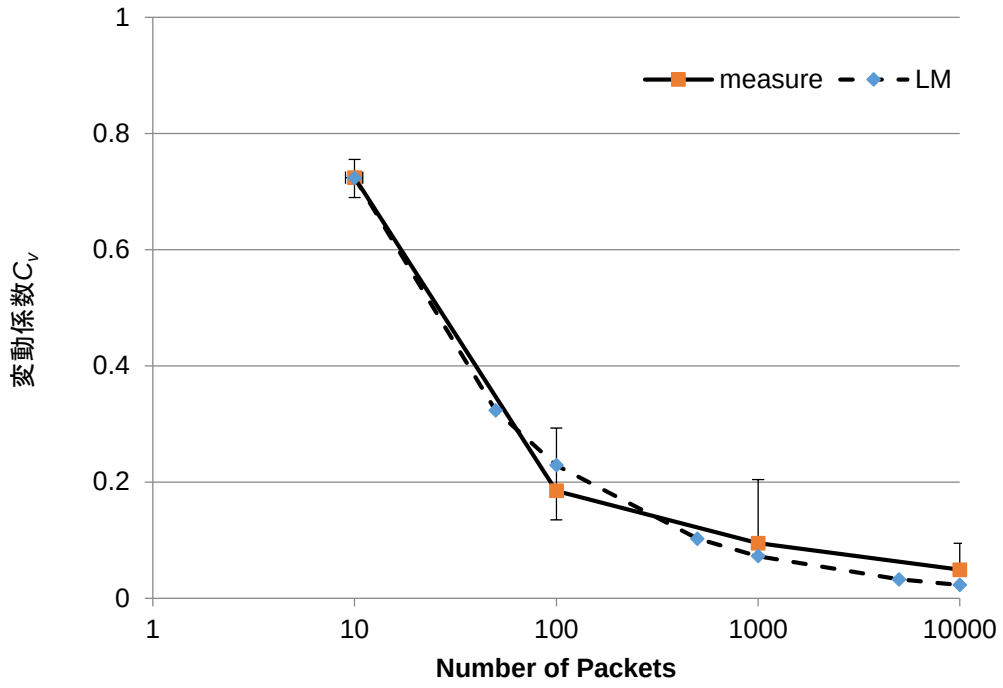


図 5-4 実 Wi-Fi 網を用いた GE モデルにおける変動係数のフィッティング

表 5-2 GE モデルパラメタ

parameters	p	r	k	h
results	0.00031249	0.00055368	0.79558	0.90315

5.1.2 評価結果

5.1.1 節で求めた GE モデルを用いて R-TCP の性能評価を行った．図 5-5 及び図 5-6 に評価結果を示す．図 5-5 は，Linux で標準的に使用される輻輳制御アルゴリズム CUBIC TCP を用いた場合の結果である．図 5-6 は，R-TCP を用いて輻輳制御アルゴリズムをネットワーク環境に合わせ動的に切替えた場合の結果である．R-TCP は図 5-7 に示すように，このネットワークの状況では大部分において TCP Hybla を選択しており，廃棄が少ない状況では図 4-4 に基づき TCP Hybla を選択していると考えられる．また，廃棄が特に多い状況では，廃棄パタンに応じ，非バースト的廃棄が多い状況では図 4-4 に基づき Scalable TCP を，バースト的廃棄が多い状況では図 4-5，図 4-6 により TCP Westwood を選択していると考えられる．

インタラクティブ通信では平均的な遅延の大きさも，遅延の揺らぎも利用者のアプリケーション操作性に大きく影響するためこれらを比較した．表 5-3 に示すように, R-TCP は CUBIC TCP に比べて平均レイテンシが約 45%減っていることに加え，揺らぎも 45%改善していることがわかる．

図 5-6 に示した R-TCP の評価結果において, CUBIC TCP に比べレイテンシは改善しているものの，人が遅さを感じる限界である 1 秒を越えるスパイクが何カ所か発生している．R-TCP は，TCP が輻輳を検出した頻度により輻輳制御アルゴリズムを選択し直す．より輻輳ウィンドウサイズを速く広げるアルゴリズムに切り替えるため，再送パケットをより速く再送できるようになり，CUBIC TCP に比べレイテンシを改善できる．ところが，R-TCP は，輻輳制御アルゴリズム選択のばたつきをなくすため移動平均をとり，ならしているため，輻輳制御アルゴリズムの切替えにタイムラグが生じる．この間にバースト的にパケット廃棄が発生した場合には，切替え前の輻輳制御アルゴリズムのままで再送が繰り返されてしまいバーストデータの転送時間が長くなると考えられる．

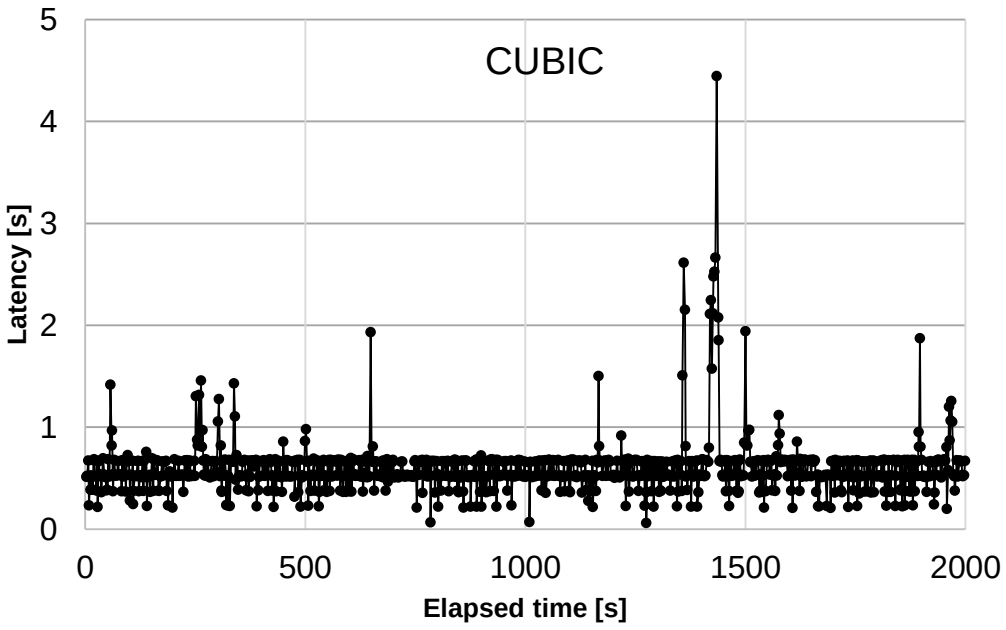


図 5-5 インタラクティブ通信における CUBIC TCP のレイテンシ特性

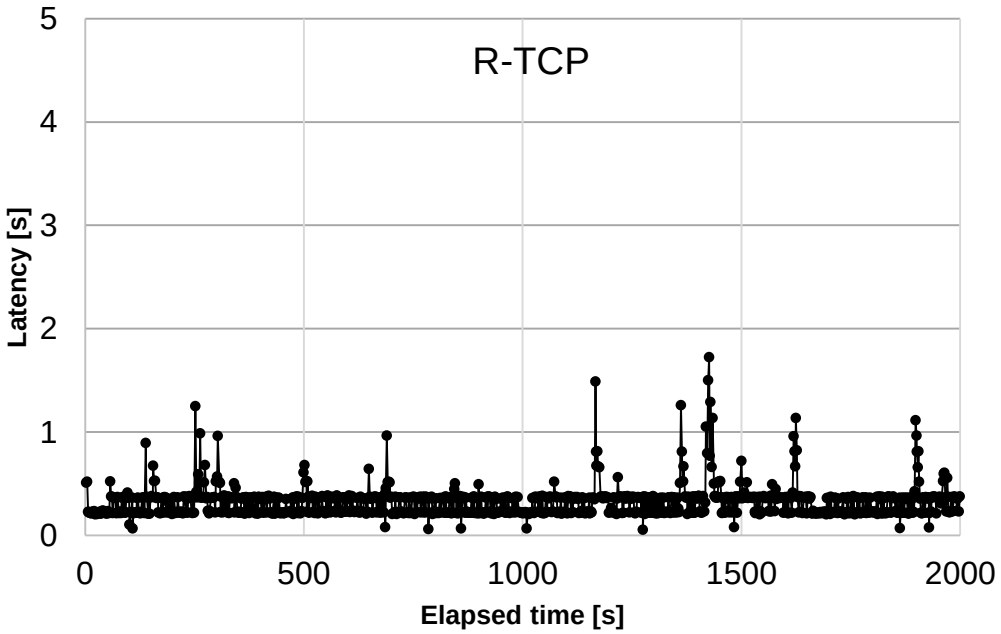


図 5-6 インタラクティブ通信における R-TCP のレイテンシ特性

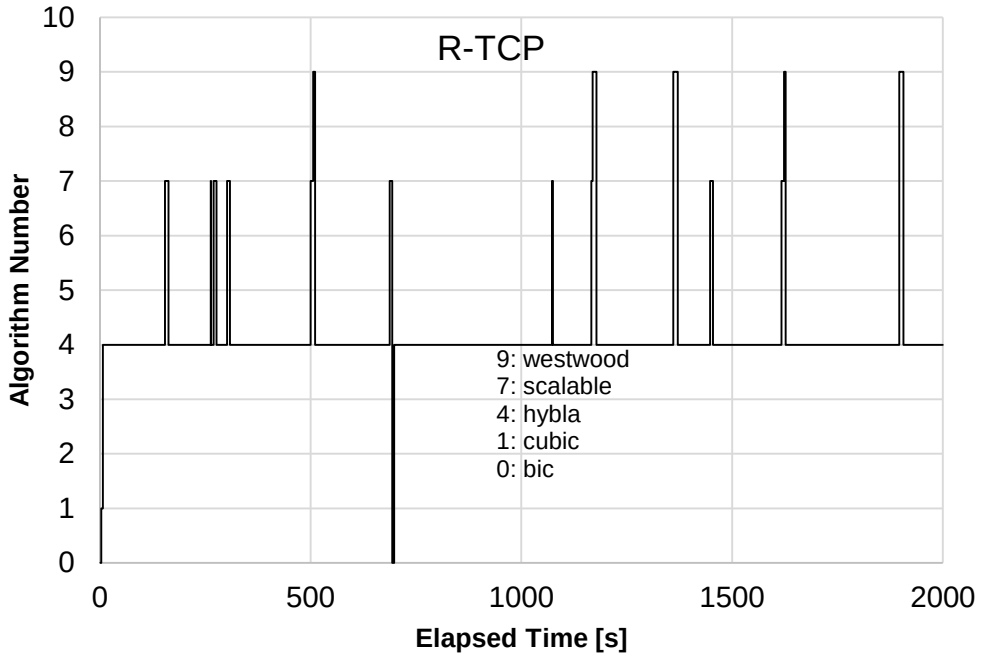


図 5-7 R-TCP の輻輳制御アルゴリズム変化(固定無線環境のシミュレーション)

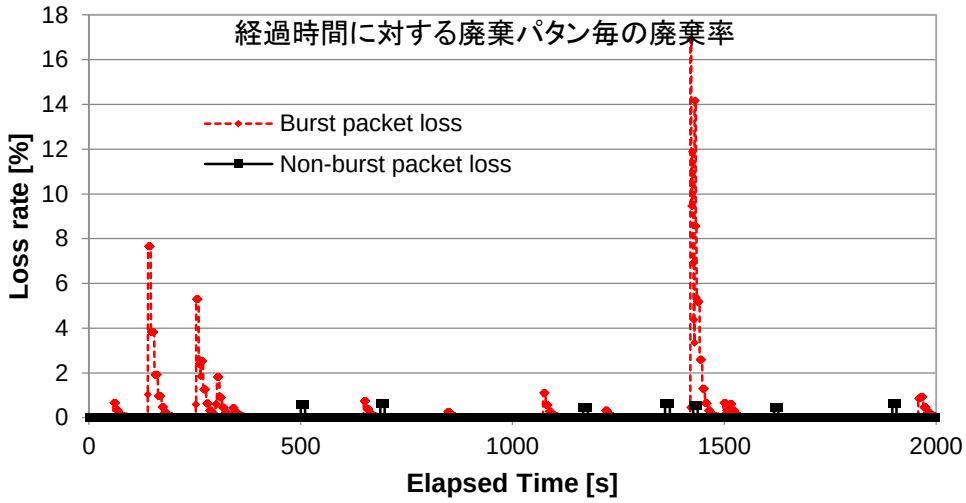


図 5-8 廃棄率の変化(固定無線環境のシミュレーション)

表 5-3 固定無線環境におけるレイテンシ評価結果

	Linux default (CUBIC TCP)	Proposed method (R-TCP)	改善率 [%]
Average Latency (s)	0.589	0.324	45.0
Latency Integration(s)	590.1	324.6	45.0
Standard Deviation (s)	0.28	0.155	44.6

5.2 移動無線環境のシミュレーション

本節では，図 5-9 に示す端末が移動する無線環境で R-TCP を利用した場合の性能改善効果をシミュレーションにより確認する．具体的には，単一の輻輳制御アルゴリズムを使い続ける場合と R-TCP により輻輳制御アルゴリズムを切替える場合のレイテンシを比較する．本シミュレーションでは，無線レイヤは以下に示す仕様とする．無線アクセス網では，ノイズやフェージング，干渉により無線メディアでエラーを生じやすい．こうしたエラーの多くは，無線の Moduration や Coding を適切に選択することで，受信装置においてある程度回復できる．近年の移動無線アクセス技術は，適応変調により，電波強度に応じてその強度で最も効率的な MCS(Moduration and Coding System)を選択するが, MCS の境界における電波強度では，無線エラーが多くなる．更に，無線エラーは ARQ

による再送である程度救済されるが，RTT が増加したり，救済しきれないものは TCP のパケット廃棄として見えてくる．そのため，移動無線網では，電波強度の変化が TCP のパケット廃棄に与える影響が大きいと考えられる．電波強度は，距離，フェージング，シャドーイング，干渉など様々な要因が影響するが，移動に伴うフェージングなどの影響は次章の実機評価で確認することとし，まずは，ランダム移動による基地局からの距離に応じた電波強度の変化に対する適応変調への R-TCP の適合性を確認できれば，実環境におけるフェージング，干渉などの影響への効果も概ね期待できると思われる．

- チャネルモデル：IEEE 802.16e (WiMAX)
  - ARQ による無線レイヤの誤り訂正を実装した．ARQ を適用することにより品質が悪い場合に RTT が大きくなりやすい．
  - 適応変調を実装した．電波の干渉やビル陰などによる SNR(Signal to Noise Ratio)の変化に応じ，MCS を変更する．これにより MCS が変化することで帯域が動的に変化する．
  - ハンドオーバはしない前提とする．
- 無線端末の移動モデル：基地局を中心に任意の方向へ移動する Random Way Point を用いた．実際には，無線端末は，人と共に移動したり，車両に搭載されることが多いため，その挙動に合わせた効率化が検討されるべきであるが，効率化を行わずランダムに移動する状況下の性能を調べることで，性能の限界を明らかにできると思われる．
- 無線伝播路モデル：電波強度の長区間変動として，東京の観測データに基づく，Cost231 Propagation Loss Model [55][56]を使用する．シャドーイングなどの短区間変動，及びフェージングなどの瞬時変動は考慮しない．

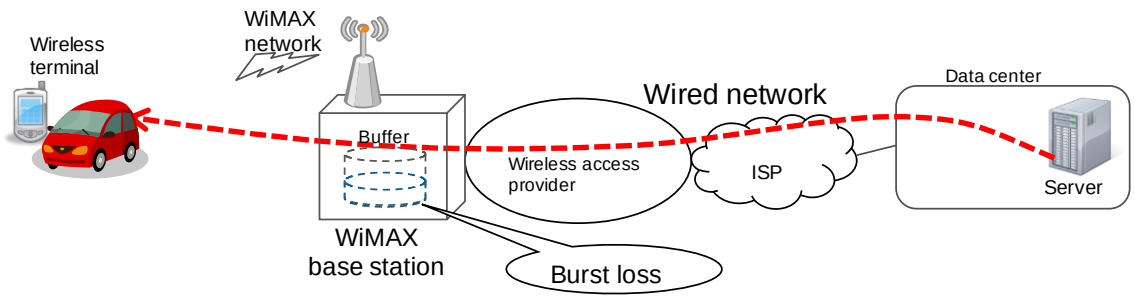


図 5-9 固定無線環境のシミュレーションモデル

5.2.1 チャネルモデル

以下に無線チャネルのモデルを示す.

5.2.1.1 適応変調

SNR が変化すると各 MCS は，図 5-10 に示すビットエラー特性を示す．本シミュレーションでは，SNR の変化によるビットエラー率が10^{-4}以下に収まるように適切な MCS に切替える適応変調を行う．適用する MCS の詳細を表 5-4 に示す．本シミュレーションでは 20ms 毎に MCS の切替え判定を行う.

表 5-4 SNR 範囲に対する適用する MCS

MCS	⓪ BPSK 1/2	①QPSK 1/2	②QPSK 3/4	③16QAM 1/2	④16QAM 3/4	⑤64QAM 2/3	⑥64QAM 3/4
SNR 範囲 (dB)	7.5～ 9.5	9.5 ～ 10.0	10.0 ～ 11.0	11.0 ～ 12.0	12.0 ～ 20.0	20.0 ～ 45.0	45.0～

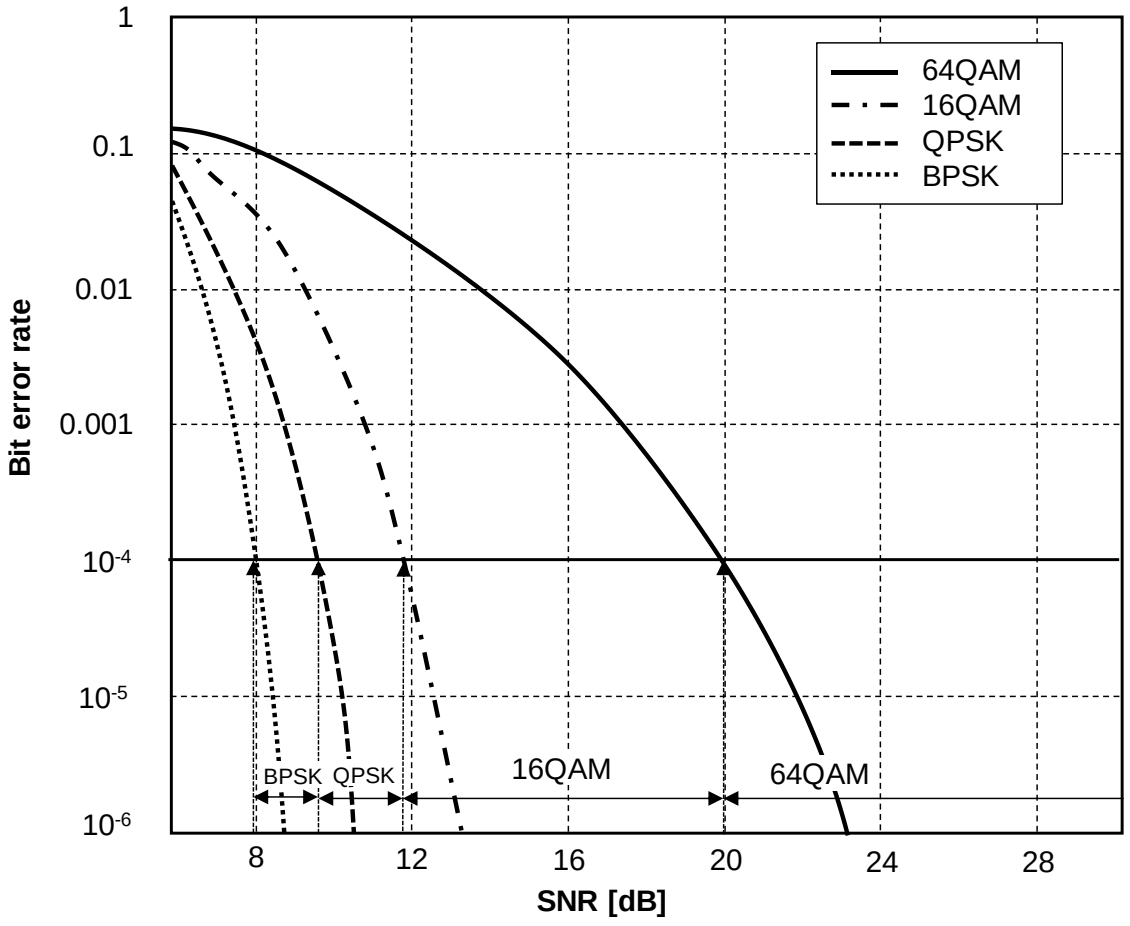


図 5-10 適応変調の動作

5.2.1.2　ARQ

無線レイヤにおける再送機能として ARQ を実装する．最大再送回数，再送遅延を設定し，パケット廃棄発生時，最大再送回数まで再送リトライを行う．本シミュレーションでは再送成功の場合は，パケット廃棄の回数分，再送遅延を付加し，上位レイヤへの転送を遅らせる(RTT 増加)．再送失敗(リトライアウト時)の場合は，パケットを廃棄する．

また，簡易順序制御機能も実装する．上位レイヤへの転送遅延(再送遅延，アクセス遅延)を考慮して，WiMAX の MAC レイヤにて順序制御を行う．転送遅延が満了されるまでは，上位レイヤへの転送を保留する．前パケットの転送タイミングより，次パケットの転送タイミングの方が早い場合，前パケットの転送が完了するまで保留する(図 5-11)．

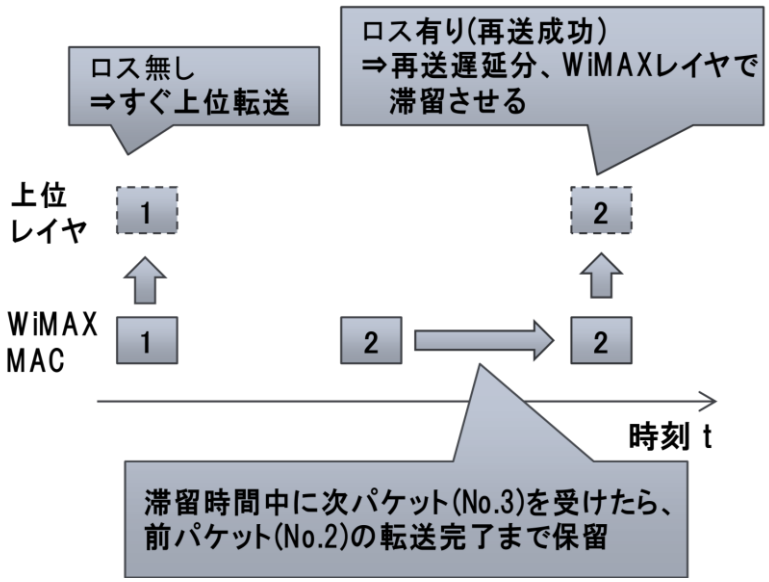


図 5-11 ARQ の動作

5.2.2　移動モデル

今回の評価では，移動端末の移動モデルとして，最も一般的に用いられるモデルである Random Way Point を用いた．本モデルは，位置，速度，加速度をランダムに選択する．図 5-12 に移動例を示す．

端末移動速度：0～20km/h

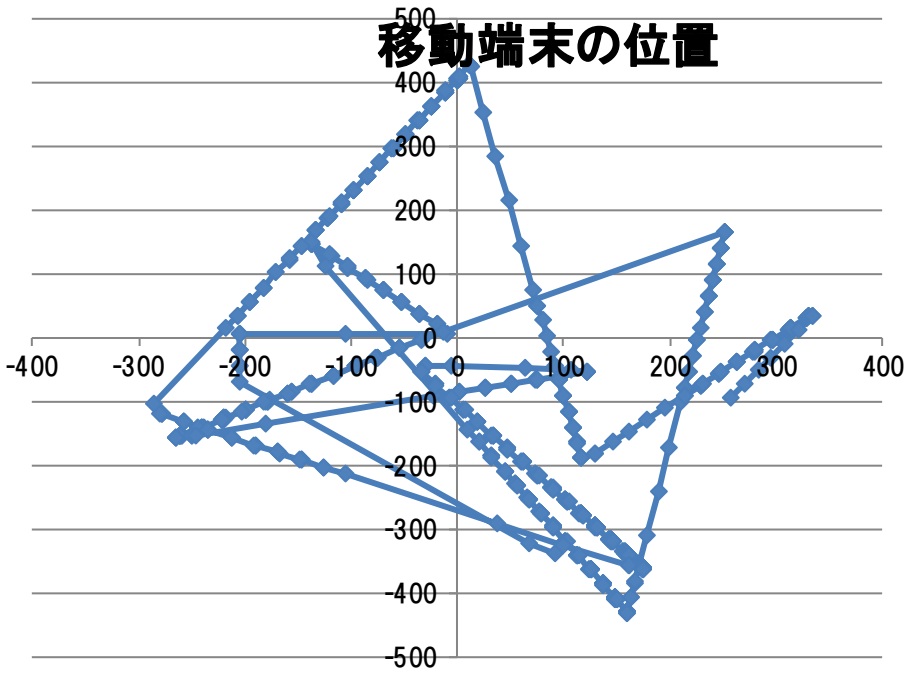


図 5-12 Random Way Point による移動端末の移動例

5.2.3 パスロスモデル

秦-奥村モデルを拡張したモデルである Cost231 Propagation Loss Model [55][56]を使用する．都市部に適用可能なモデルであり，郊外/田舎の開けたエリアの評価にも利用できる．図 5-13 に本パスロスモデルの基地局からの距離と SNR の関係を示す．

- 周波数：1500～2000[MHz]
- 移動端末(MS)アンテナ高：1～10[m]
- 無線基地局(BS)アンテナ高：30～200[m]
- BS-MS 間距離：～20[km]

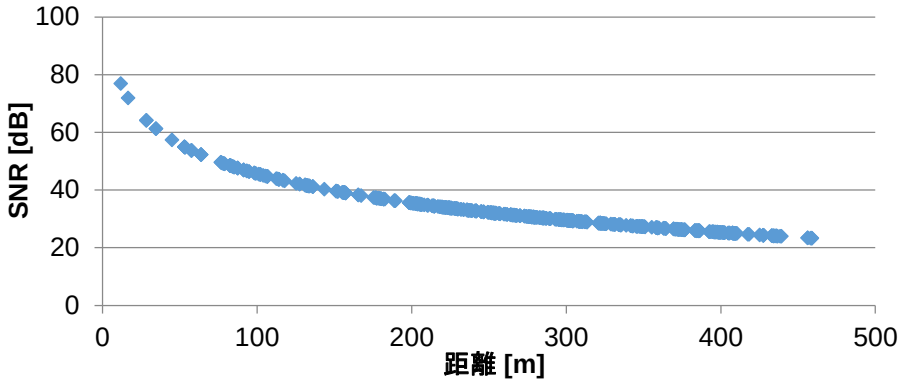


図 5-13 Cost231 Propagation Loss モデルにおける基地局からの距離と SNR の関係

5.2.4 評価結果

図 5-14 は，シミュレーション中の移動端末の移動の様子を示している．X, Y 軸の原点に基地局があり，基地局から離れるほど電波強度が弱くなる．図 5-15 は，移動端末と基地局間の距離に応じ SNR が変化し，それに応じて適応変調のメカニズムが，最適となる MCS を選択し，切替えている様子を示している．図 5-16 は，MCS の切り替えのタイミングで無線レイヤにおけるビットエラーが多く発生し，BER(Bit Error Rate)が上昇していることを示している．これは，適応変調のメカニズムがビットエラーを監視しながら MCS を切替えているためで，無線基地局がある特定の MCS を選択している際，電波強度が低下しビットエラーが増えてくると MCS を切替える．そのため，MCS の切り替えのタイミングでビットエラーが多く発生しているように見える．また，無線レイヤでのフレーム誤りに対しては無線レイヤにおいて ARQ による再送により誤り訂正を行うが，本 ARQ の実装では再送数を制限しているため，訂正しきれなかったパケットは無線レイヤで廃棄される．そのため，TCP におけるパケット廃棄として検出される．

図 5-17，図 5-18，及び表 5-5 に CUBIC TCP と R-TCP のレイテンシ評価結果を示す．図 5-17，図 5-18 を比べると，CUBIC TCP で発生しているレイテンシが 2 秒を超えるケースが R-TCP では大幅に減っていることがわかる．また，表 5-5 に示すように R-TCP は平均レイテンシ，及びトータルのレイテンシをおよそ 12%改善できることがわかる．また，レイテンシの標準偏差も 12%改善し揺らぎが減っていることがわかる．

図 5-21 は，CUBIC TCP と R-TCP における輻輳ウィンドウサイズの変化を比較している．ほとんどの場面において，R-TCP がより速く輻輳ウィンドウサイズを広げていることがわかる．

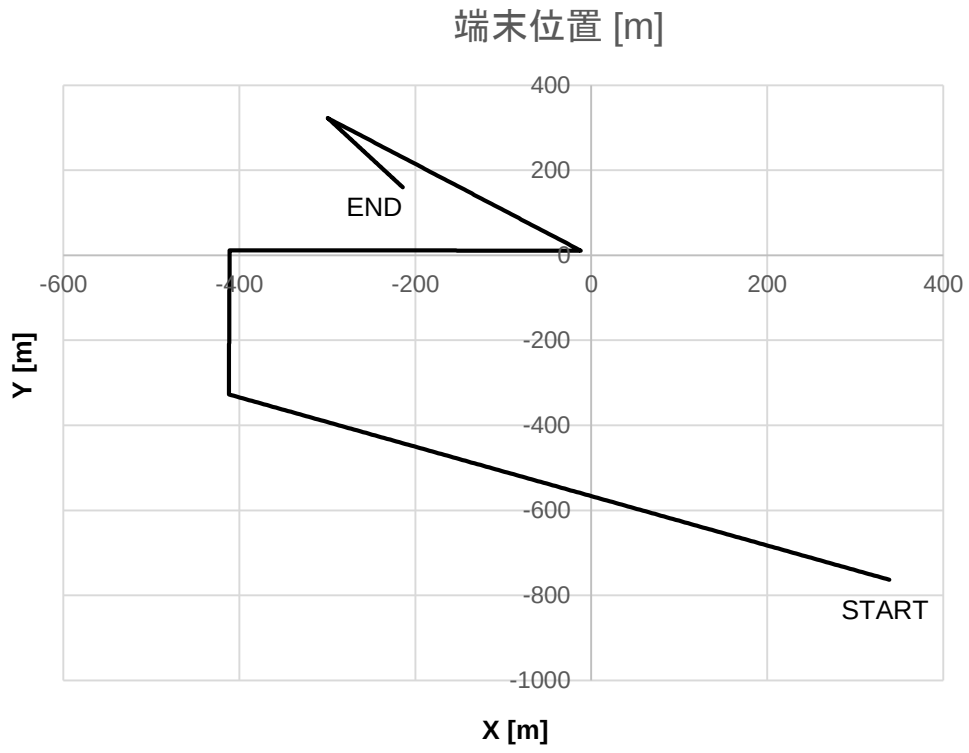


図 5-14 移動端末の動作

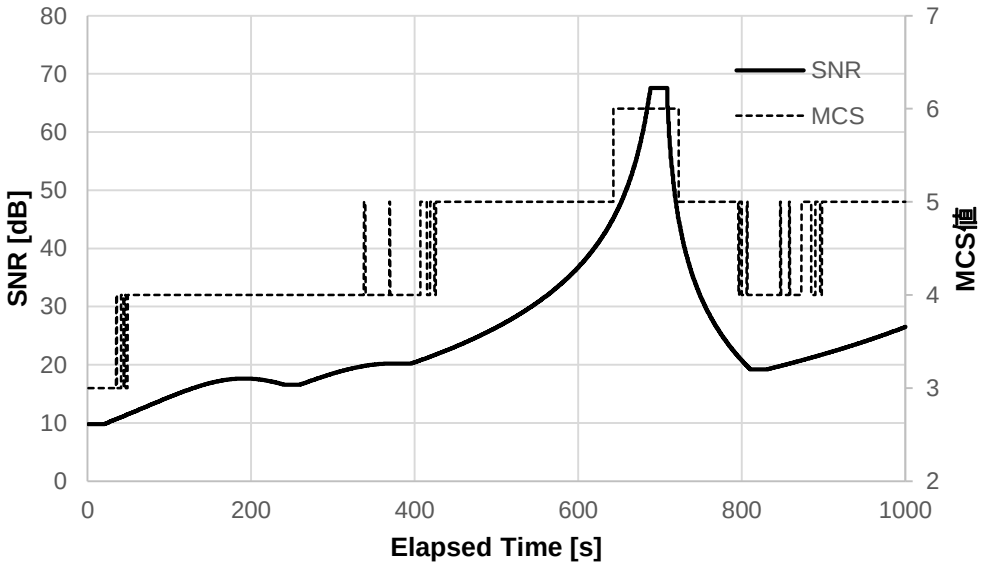


図 5-15 SNR の時間変化に対する MCS の変化

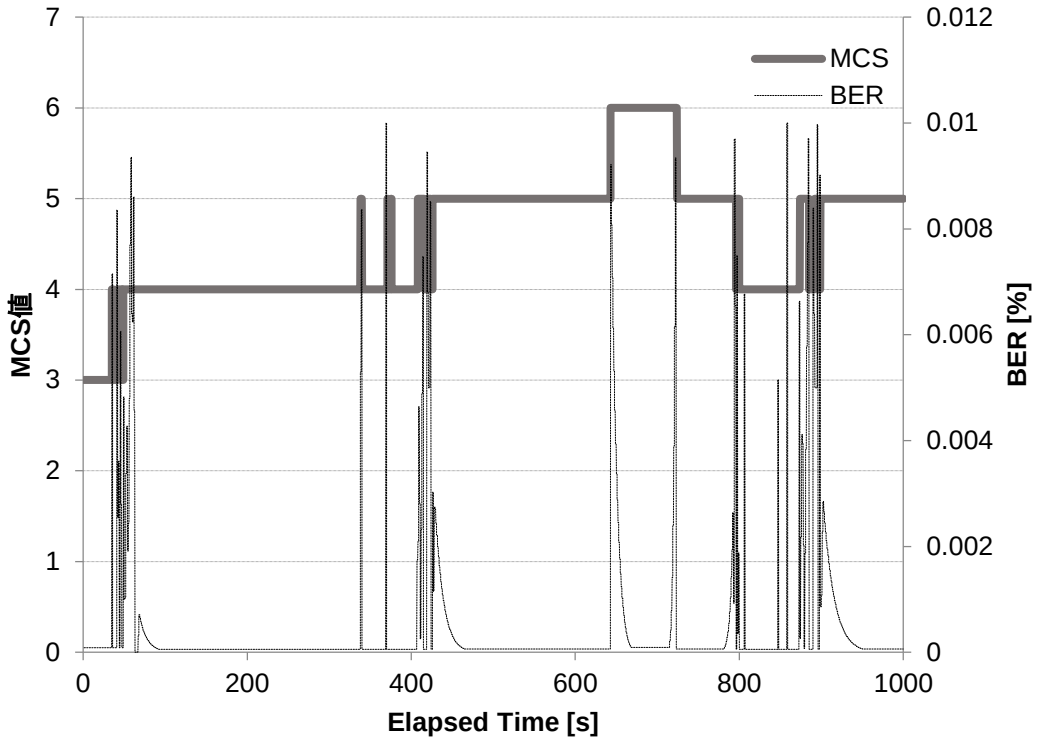


図 5-16 経過時間に対する MCS の変化と BER の変化

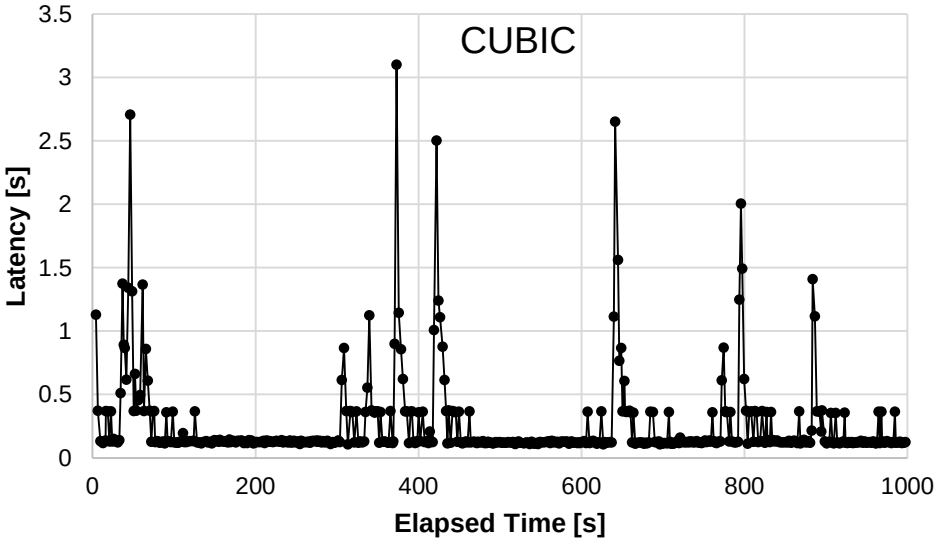


図 5-17 CUBIC のレイテンシの変化

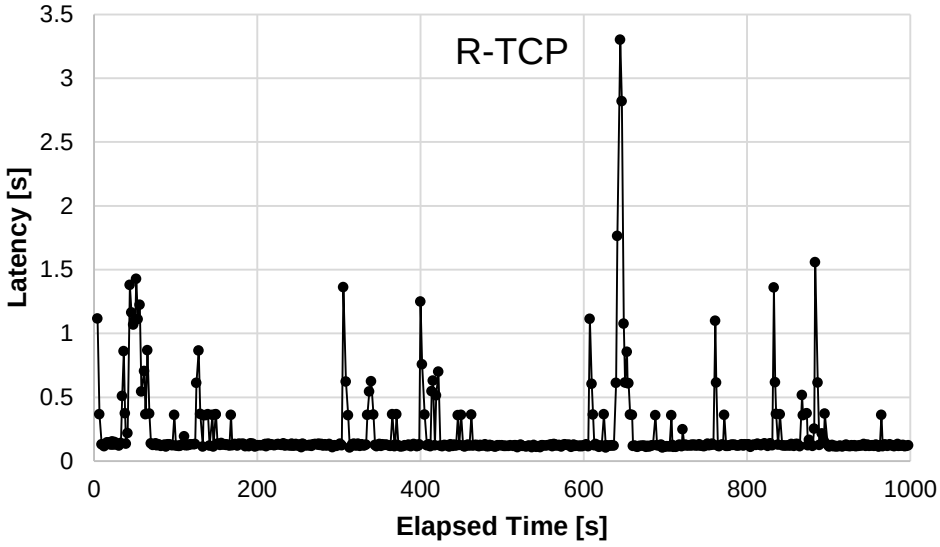


図 5-18 R-TCP のレイテンシの変化

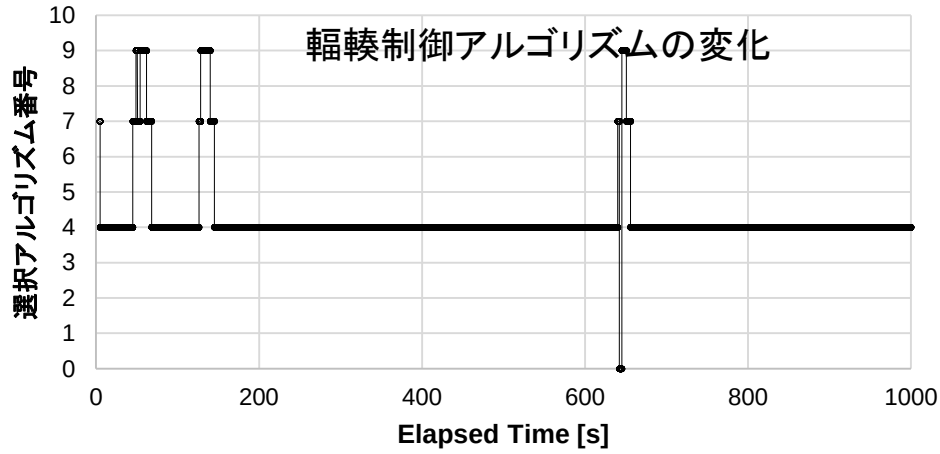


図 5-19 輻輳制御アルゴリズムの変化(移動無線環境のシミュレーション)

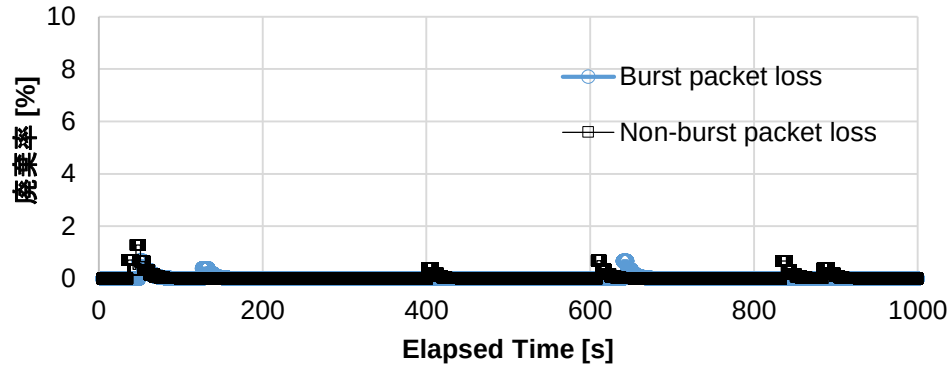


図 5-20 廃棄率の変化(移動無線環境のシミュレーション)

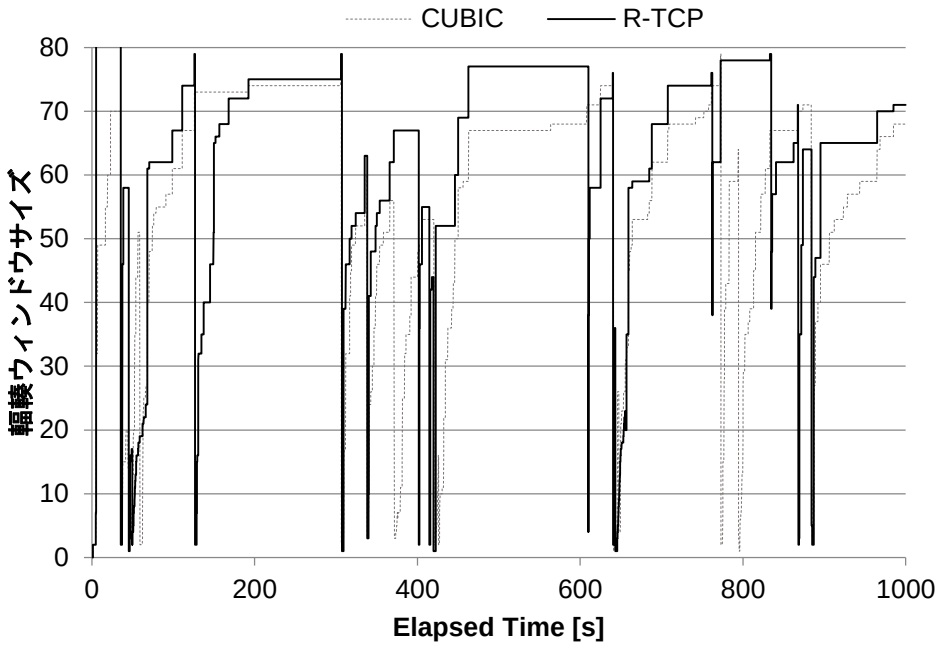


図 5-21 廃棄率と CUBIC の輻輳ウィンドウサイズの変化

表 5-5 移動無線環境におけるレイテンシ評価結果

	Linux default (CUBIC TCP)	Proposed method (R-TCP)	改善率 [%]
Average Latency [s]	0.248	0.217	12.5
Total Latency [s]	122.4	107.2	12.4
Standard Deviation [s]	0.339	0.298	12.0

5.3 本章のまとめ

提案方式の動作の正常性を確認すること，実機評価では試せない様々な環境で性能的にどこまで得られるかを検証するために，本章では，提案方式のレイテンシ改善効果を，シミュレーション環境を用いて評価した.

まず，IEEE802.11g(Wi-Fi)を用いた固定無線環境を，GE チャネルモデルを用いたシミュレーションモデルを用いて，レイテンシ改善効果を確認した．実際のバースト的廃棄が多く発生している Wi-Fi 網の廃棄特性を基に

Levenberg-Marquardt (LM) 法を用いてGEチャネルモデルのパラメタを特定した．その結果，R-TCP はバースト的廃棄が多い状況であっても，最適な輻輳制御アルゴリズムを選択し，表 5-6 に示すように，45%程度までレイテンシを改善できることがわかった．

また，IEEE802.16e(WiMAX)を用いた移動無線環境を模擬したシミュレーションモデルで提案方式のレイテンシ改善効果を評価した．端末の移動に伴う適応変調により MCS が変化する際パケット廃棄が起きやすく，R-TCP を用いることで，表 5-6 に示すように，レイテンシを約 12%改善できることがわかった．

R-TCP は，輻輳制御アルゴリズム選択のばたつきをなくすため移動平均をとり，ならしているため，輻輳制御アルゴリズムの切替えにタイムラグが生じる．この間にバースト的にパケット廃棄が発生した場合には，切替え前の輻輳制御アルゴリズムのままで再送が繰り返されてしまいバーストデータの転送時間が長くなり再送するためレイテンシが特に大きくなる．この要因に対応するためにバースト的な廃棄が起きた場合に輻輳制御アルゴリズムの切替えタイミングを改良するなどの工夫が必要である．

表 5-6 R-TCP のレイテンシ改善効果

評価環境	改善率 [%]
固定無線	45.0
移動無線	12.0

第6章 LTE 網における実機評価

第 5 章では，シミュレーションモデルを用い提案方式の動作の正当性，効果の程度を確認した．そこで，本章では，提案方式を実機に実装し LTE 網を用いた実環境で評価を行う．

提案方式を Linux OS (Fedora 13) に実装し，LTE 網を用いて性能を評価する．集合住宅内の定点，及び移動する電車内について，無線アクセス網が空いている状況，混雑している状況について評価を行う．

6.1　実機への実装

本評価では，R-TCP を実機評価するために，図 6-1 に示すように R-TCP を Linux OS のカーネル内に実装する．R-TCP コントローラが，TCP プロトコルスタックからパケット廃棄や RTT についての情報を受け取り，最適な輻輳制御アルゴリズムを動的に選択する．各輻輳制御アルゴリズムの廃棄パタン毎のレイテンシ特性のデータベースもカーネル内に実装している．コンフィグコマンドを実行し，廃棄率の計測周期や，輻輳制御アルゴリズムの切替え周期といったパラメタを変更することができる．R-TCP コントローラが計測した廃棄パタン毎の廃棄率や RTT，現在選択している輻輳制御アルゴリズムは，ログとして出力することができる．なお，実機に用いたプログラムコードは，第 5 章におけるシミュレーション評価で用いたものと同一のものを使用している．

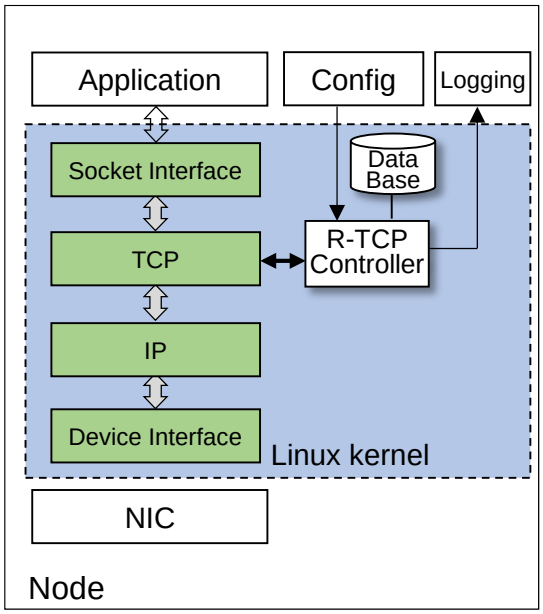


図 6-1 R-TCP の Linux カーネルへの実装

6.2 評価方法

LTE 網を介したサーバと端末間のインタラクティブ通信の性能を，提案方式と Linux OS の標準 TCP である CUBIC を用いる場合について以下 4 つの環境下で比較する．端末の移動に伴う電波環境の変化によるパケット廃棄率の違いと，LTE 回線の利用率に起因する RTT の違いに対する提案方式の効果を確認する．

(1) LTE 回線利用率が低い時刻の屋内固定ポイント

(2) LTE 回線利用率が高い時刻の屋内固定ポイント

(3) LTE 回線利用率が低い時刻の移動中の電車内

(4) LTE 回線利用率が高い時刻の移動中の電車内

評価環境としては，図 6-2 に示すように，インタネット上に配置したサーバに二つの仮想マシンを用意し，一方は提案方式を，他方は比較用に Linux OS 標準の CUBIC TCP を用いる．Windows8.1 の端末上に 2 台のデータ受信用の仮想マシン (クライアント) を用意する．また，Android OS のタブレットをノート PC と USB テザリングし LTE モデムとして利用する．両仮想マシンは共通の無線回線を用いてサーバからデータを受信する．

サーバの仮想マシンがクライアントの仮想マシンに対し仮想デスクトップサービスを提供する場面を想定し，このサービスが画面の更新情報として送り得

る最大のバーストサイズである 64Kbytes のデータを，ユーザの操作時間を踏まえ 2 秒間隔で送信する．

これらのバーストデータを端末でキャプチャし最初のパケットから最後のパケットまでの受信時刻の差をレイテンシとし，これを性能指標とする．本データの送信には，短いバーストデータを繰り返し送信できるように iperf [69]を改良したツールを用いた．

R-TCP のネットワーク品質計測については，サーバ，及びクライアント上で動作する R-TCP がそれぞれ，サーバからクライアント方向，クライアントからサーバ方向の廃棄率，及び RTT を 200ms 周期で計測し，この計測結果を基に 200ms 周期で最適な輻輳制御アルゴリズムの選択を行う．

ネットワーク品質計測については，今回の評価では RTT が 200ms より大きい環境が多いため，R-TCP は，非バースト的，及びバースト的廃棄率を 200ms 周期で計測すると共に, TCP が計測した RTT を 200ms 周期で取得するようにした．そして，この計測結果を基に 200ms 周期で最適な輻輳制御アルゴリズムの選択を行う．また，本評価では，非バースト的廃棄率はおよそ 0%から 10%，バースト的廃棄率はおよそ 0%から 1.8%，RTT は 0ms から 300ms のレイテンシ特性を用いておりこれらの範囲でレイテンシ推定が可能である．

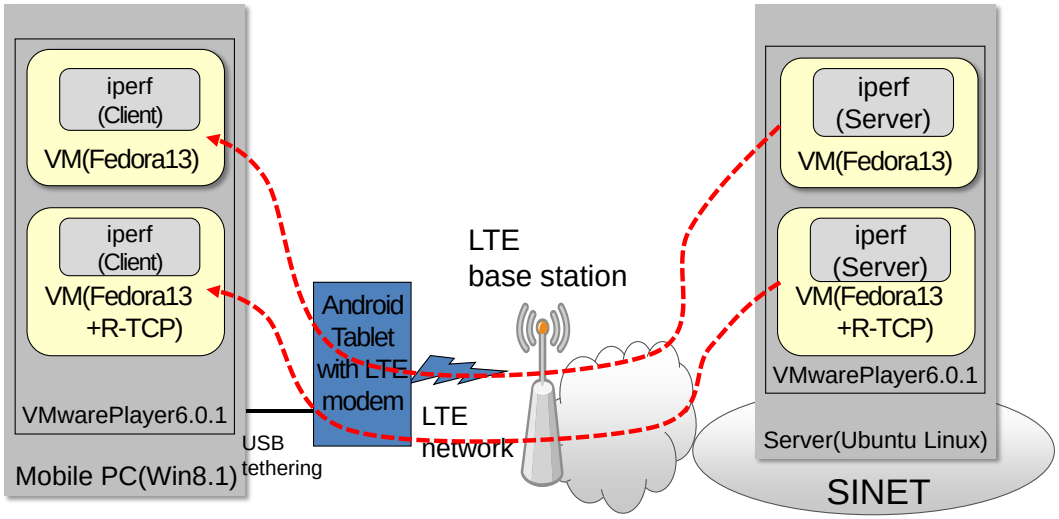


図 6-2 評価環境

6.3　評価結果

評価した 4 つの環境について結果を示す.

6.3.1　LTE 回線利用率が低い屋内固定ポイント

図 6-3 に CUBIC TCP と R-TCP のレイテンシ評価結果を示す. 図 6-4 は両 TCP で計測した RTT の時間変化を示す．図 6-6 は両廃棄パタンの廃棄率を，図 6-5 はその際に R-TCP が選択した輻輳制御アルゴリズムを示す番号を示している．図 6-7 は輻輳ウィンドウサイズの大きさを示すためにサーバが 1 秒毎に送信したパケット数を示した図である．表 6-1 は，これらの図から得られる統計データを CUBIC TCP と R-TCP で比較している．

本評価は, 屋内の固定ポイントで行い図 6-4, 表 6-1 に示すようにサーバと端末間の RTT は 200ms 前後であった．また，図 6-6 に示すようにバースト的廃棄はなく，非バースト的廃棄が低頻度で発生している．

本環境下では, 廃棄が無い時は, R-TCP は, RTT が 200ms の特性を示す図 4-4 を参照し，廃棄率 0.01%以下では最小のレイテンシを示す TCP Hybla を選ぶ．図 6-6 における経過時間 522 秒のように非バースト的廃棄率が高い場合は図 4-4 に基づき Scalable TCP を選択するが，経過時間 402 秒のように廃棄率が 3%以下の場合は TCP Hybla を選ぶ．経過時間 286 秒では，R-TCP，CUBIC TCP 共にタイムアウトによる再送を行ったため RTT は 300ms 以上の大きい値となっており，レイテンシも著しく増加している．この場合，R-TCP は RTT が 300ms の特性を示す図 4-8 を参照し(300ms の特性を参照する理由については 6.4 節で述べる), 廃棄率 6%における最小のレイテンシを示す TCP Hybla を選択する. 図 6-5 は実際に R-TCP が選択したアルゴリズムをプロットしており，設計通りに選択したことがわかる．

動的にアルゴリズムを選択した結果，R-TCP は表 6-1 のように，CUBIC TCP と比べ，平均レイテンシ，レイテンシの標準偏差を共に改善した．また，端末のリクエストに対しサーバがデータを応答するインタラクティブ型の通信の応答時間を考えた場合，リクエストに要する時間を RTT の半分($RTT/2$)とし，これにデータ転送のレイテンシ(L)を加えた時間が 1 秒より短くなる割合($P\{L + RTT/2 < 1\}$)は 93.5%となり，R-TCP を用いることでインタラクティブ通信の要件を満たせる確率が高くなる．

廃棄がほとんどない本環境では，RTT が 200ms と比較的大きいため，RTT により増加率αの大きさを変える TCP Hybla の輻輳ウィンドウサイズの広がりが速く，図 6-7 に示すように，立ち上がりで TCP Hybla は輻輳ウィンドウサイズを急速に広げるのに対し，CUBIC TCP は広がりが緩やかになる．そして，本環境では送信したバーストデータに対する全ての ACK が戻るのに時間がかかるため，CUBIC TCP において，最大輻輳ウィンドウサイズでデータを送信できない状態が一度生じると，次のバーストデータ送信時も最大輻輳ウィンドウサイズで送れない状況が続いてしまう．その結果，図 6-3 に示すように，TCP Hybla を選択している R-TCP と CUBIC TCP でレイテンシの差が生じる. このように，非連続的データ転送において RTT が比較的大きい場合は，廃棄が無い状況でも R-TCP により性能を改善できる.

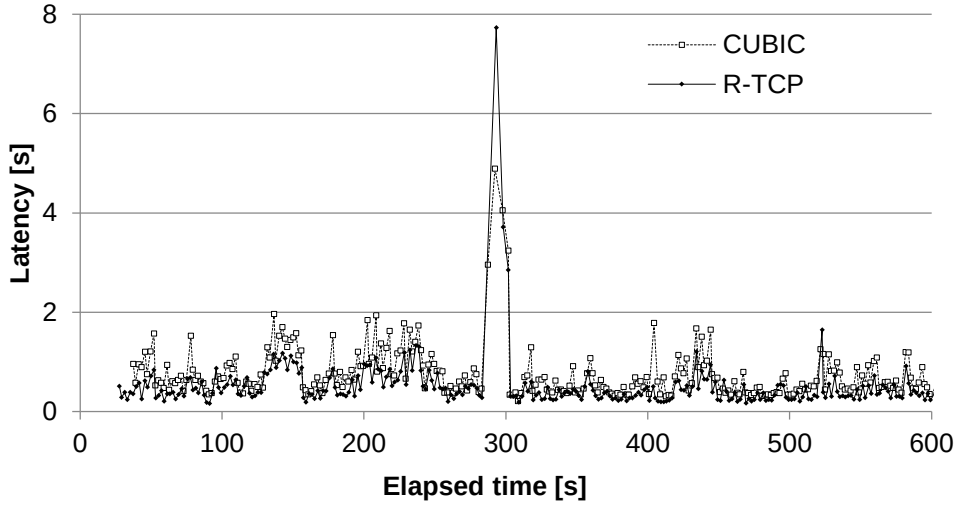


図 6-3 評価結果(LTE 回線利用率が低い屋内固定ポイント)

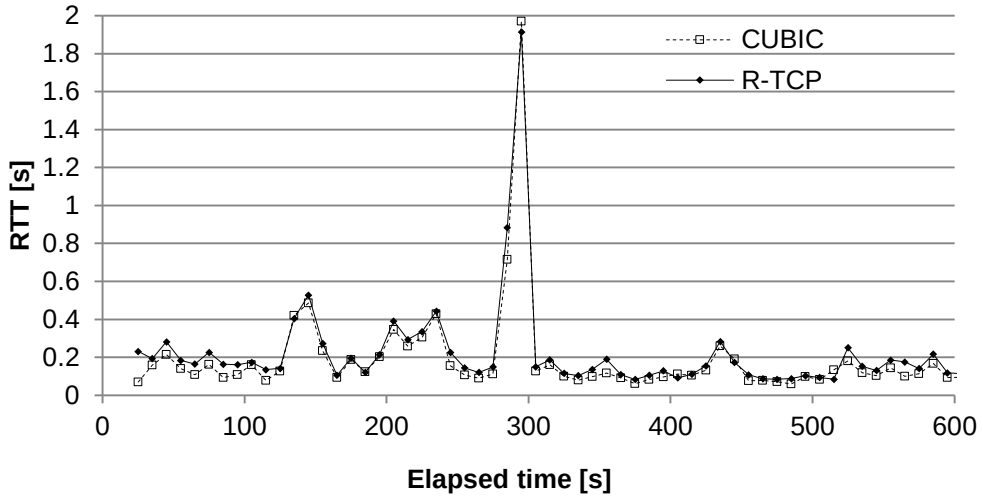


図 6-4 往復遅延時間(LTE 回線利用率が低い屋内固定ポイント)

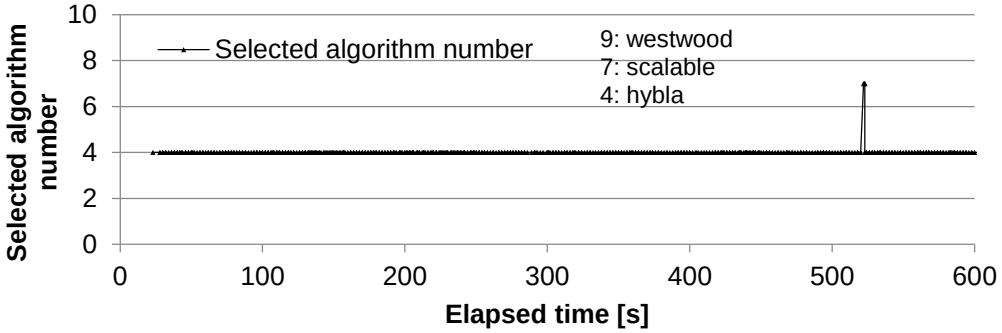


図 6-5 アルゴリズムの変化 (LTE 回線利用率が低い屋内固定ポイント)

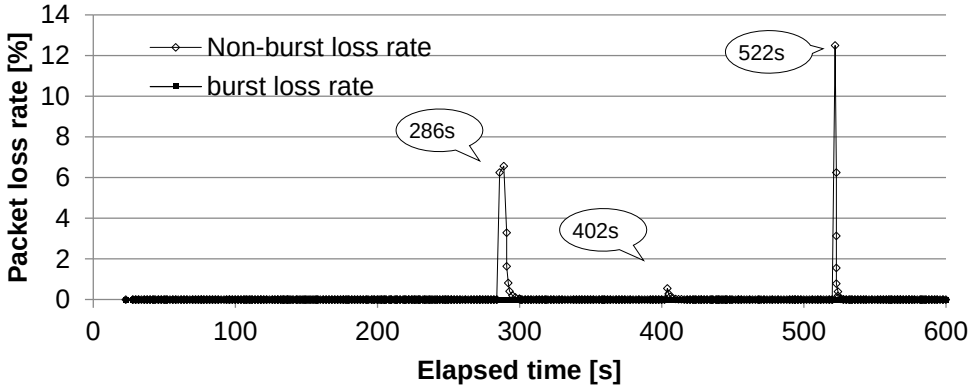


図 6-6 廃棄率の変化 (LTE 回線利用率が低い屋内固定ポイント)

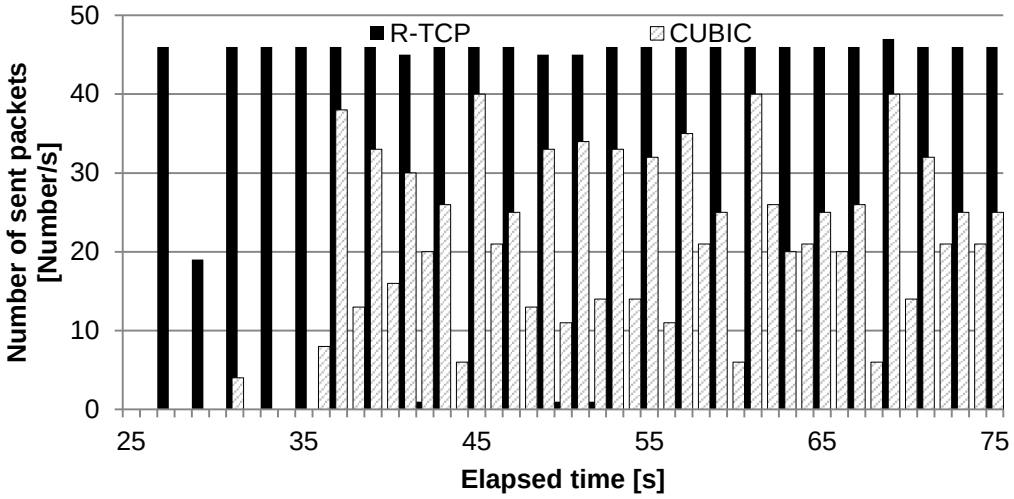


図 6-7 送信パケット数 (LTE 回線利用率が低い屋内固定ポイント)

表 6-1 統計データ(LTE 回線利用率が低い屋内固定ポイント)

	CUBIC TCP	R-TCP
Average RTT [ms]	184	217
Standard deviation of RTT [ms]	258	256
Average Latency [ms]	695	458
Standard deviation of latency [ms]	135	60
Total latency [s]	201.6	133.3
$P\left\{L + \frac{RTT}{2} < 1s\right\}$ [%]	77.3	93.5
Transmitted data [MB]	18.6	18.6

6.3.2 LTE 回線利用率が高い屋内固定ポイント

図 6-8 から図 6-11，及び表 6-2 に評価結果を示す．本評価は，6.3.1 節と同じ屋内の固定ポイントで，異なる時刻に行った．図 6-9 及び表 6-2 に示すようにサーバと端末間の RTT は 830ms 前後であった．他の LTE 回線利用者が増え，一回線当たりの帯域が減ることにより RTT が増加したものと考えられる．また，

図 6-11 に示すようにバースト的廃棄はなく，非バースト的廃棄が低頻度で発生している．

本環境下では，RTT が極めて大きいので廃棄が有る場合も無い場合も，R-TCP は，RTT が 300ms の特性を示す図 4-8 を参照し，どの廃棄率においても最小のレイテンシを示す TCP Hybla が選ばれる(図 6-10).

RTT が極めて大きいため，図 6-8，表 6-2 に示すように CUBIC TCP は平均レイテンシが 2 秒以上かかっており，アプリケーションのデータ送信間隔よりも長くなっている．この場合，前のバーストデータを送り終わってから次のバーストデータの送信となるため送信するバーストデータ数が減少する．表 6-2 に示すように，同じ評価時間内に転送したデータ量で比較すると R-TCP は CUBIC TCP よりも多くのデータを転送可能になる．

廃棄がほとんどない状況でも TCP Hybla が選択されている R-TCP と CUBIC TCP でレイテンシの差が生じる要因は，6.3.1 節と同じ理由である．

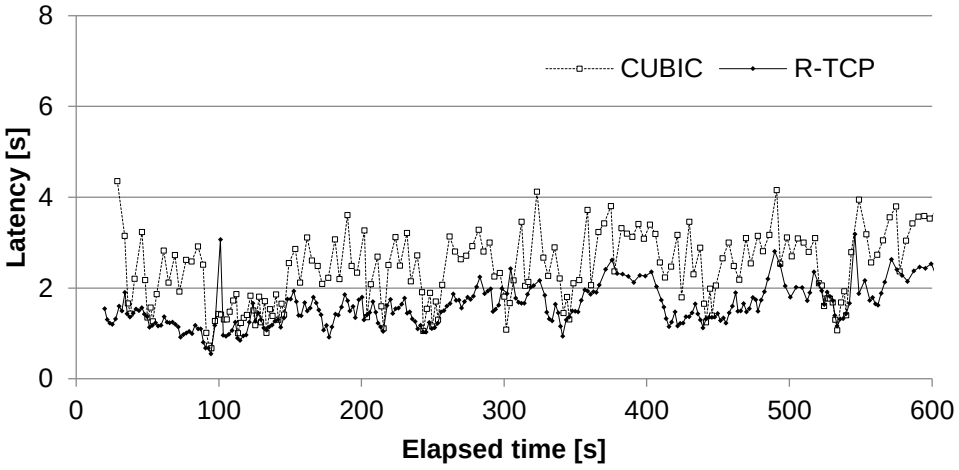


図 6-8 レイテンシの比較(LTE 回線利用率が高い屋内固定ポイント)

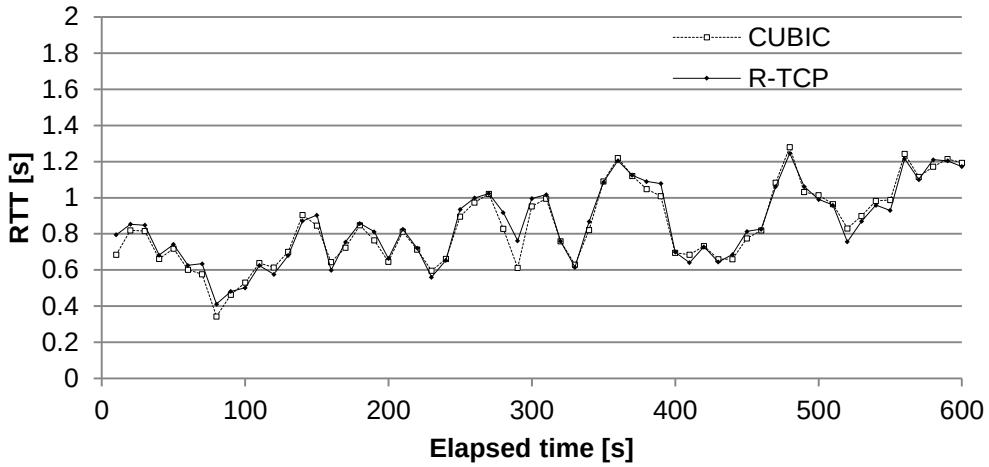


図 6-9 往復遅延時間(LTE 回線利用率が高い屋内固定ポイント)

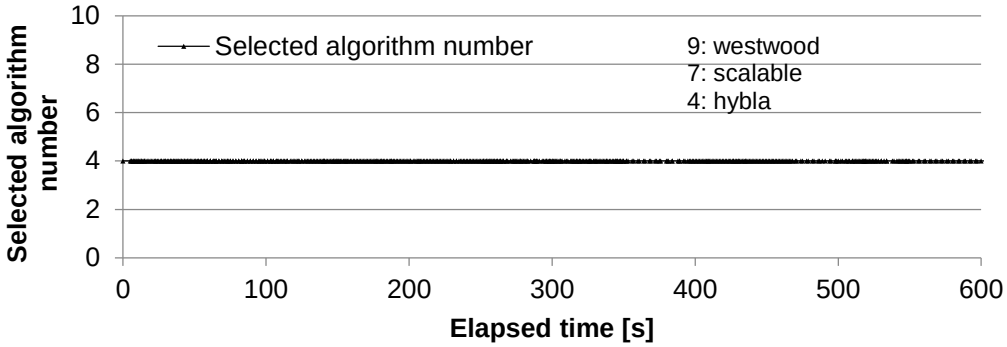


図 6-10 アルゴリズムの変化(LTE 回線利用率が高い屋内固定ポイント)

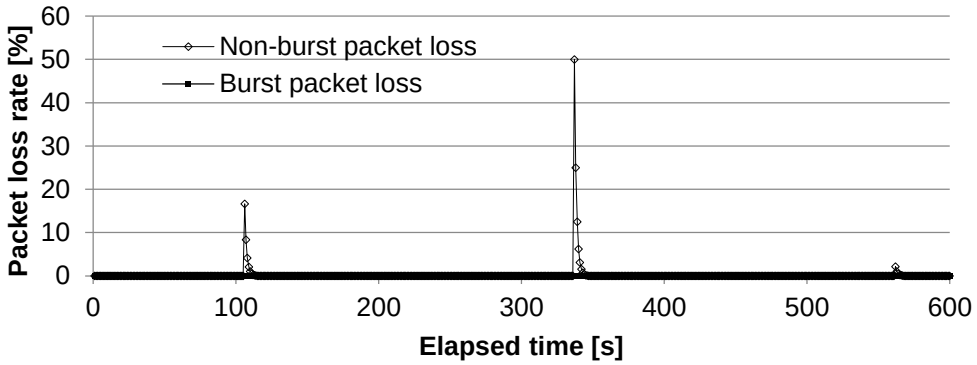


図 6-11 廃棄率の変化(LTE 回線利用率が高い屋内固定ポイント)

表 6-2 統計データ(LTE 回線利用率が高い屋内固定ポイント)

	CUBIC TCP	R-TCP
Average RTT [ms]	826	846
Standard deviation of RTT [ms]	235	208
Average Latency [s]	2.35	1.577
Standard deviation of latency [s]	0.658	0.19
Total latency [s]	428.4	408.6
$P\left\{L+\frac{RTT}{2}<1s\right\}$ [%]	17.1	39.5
Transmitted data [MB]	11.6	16.6

6.3.3 LTE 回線利用率が低い移動中の電車内

図 6-12 から図 6-16 と，表 6-3 に評価結果を示す．本評価は，比較的乗客が少ない時間帯の路線(JR 南武線[2])を利用して行った．他の移動デバイスからのトラフィックが少ないため, 図 6-15 に示すようにサーバと端末間の RTT は大きなピークをいくつか含むが平均は比較的短く 170ms 前後であった．ただし，移動しているため図 6-14 に示すようにパケット廃棄が多い環境である．

本環境下では，廃棄が無い時，R-TCP は，RTT が 200ms の特性を示す図 4-4 を参照し, 廃棄率 0%(図 10 は便宜上 0.01%まで表示)において最小のレイテンシを示す TCP Hybla を選ぶ．図 6-14 において非バースト的廃棄率が高い時刻で RTT が 125ms 以下の場合(図 6-15 参照)は, RTT が 100ms の時の特性を示す図 4-7 に基づき Scalable TCP や TCP Westwood を選択する(図 6-13)．一方で，経過時間 1232 秒のように，RTT が 300ms 以上の状態で非バースト的廃棄率が高い場合，R-TCP は RTT が 300ms の特性を示す図 4-8 を参照し廃棄率 6%における最小のレイテンシを示す TCP Hybla を選択する．

図 6-16 に示すように廃棄が発生している状況で R-TCP と CUBIC TCP でレイ

[2] 神奈川県川崎市と東京都立川市を結ぶ全長約 45 キロの路線

テンシの差が生じる要因は，一つには，廃棄がない時は，TCP Hybla は輻輳ウィンドウサイズを速く開くのに対し，CUBIC TCP は開き方が緩く，TCP Hybla を選択する R-TCP のレイテンシが小さくなるためである．また，廃棄が発生した際は，Scalable TCP といった，輻輳ウィンドウサイズの下げ幅の低いアルゴリズムを選ぶことで図 6-16 に示すように輻輳ウィンドウサイズの減少をなるべく抑えるように動作し，輻輳ウィンドウサイズの回復を早めるためである．このように，非連続的データ転送において，廃棄が多発している状況でも R-TCP を用いることで性能を改善できることがわかる．

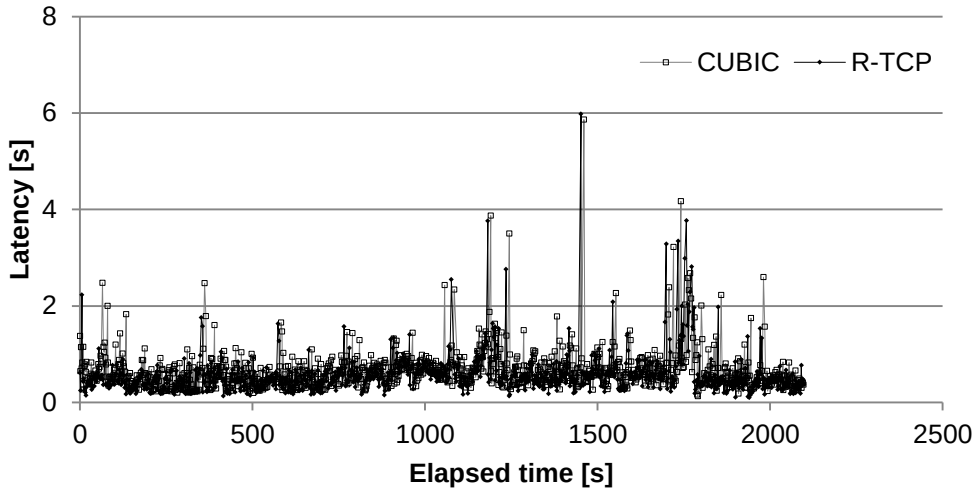


図 6-12 評価結果(LTE 回線利用率が低い移動中車内)

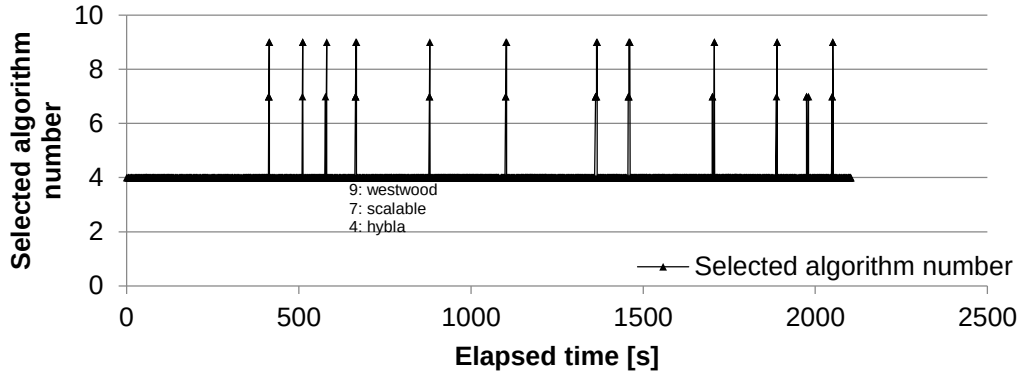


図 6-13 輻輳制御アルゴリズムの変化(LTE 回線利用率が低い移動中車内)

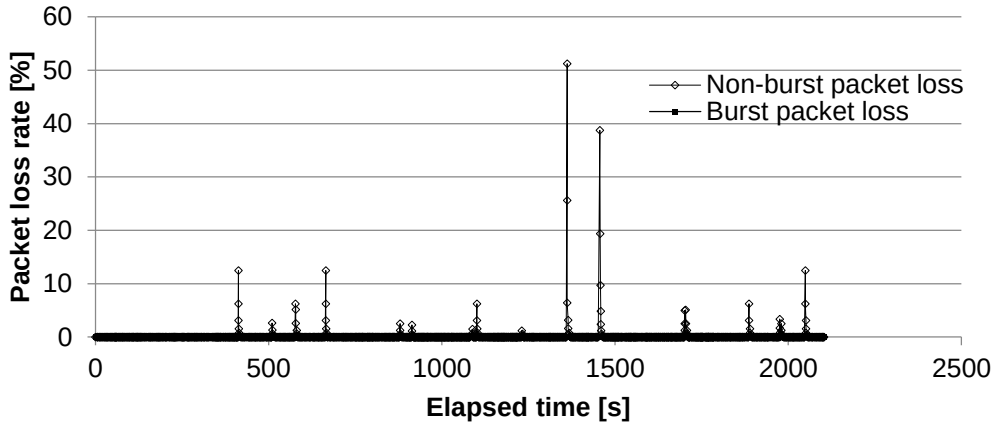


図 6-14 廃棄率の変化(LTE 回線利用率が低い移動中車内)

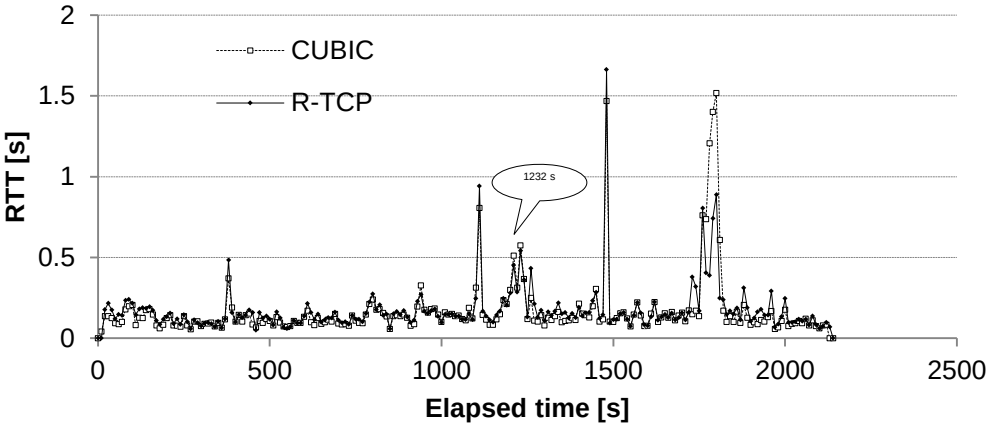


図 6-15 往復遅延時間(LTE 回線利用率が低い移動中車内)

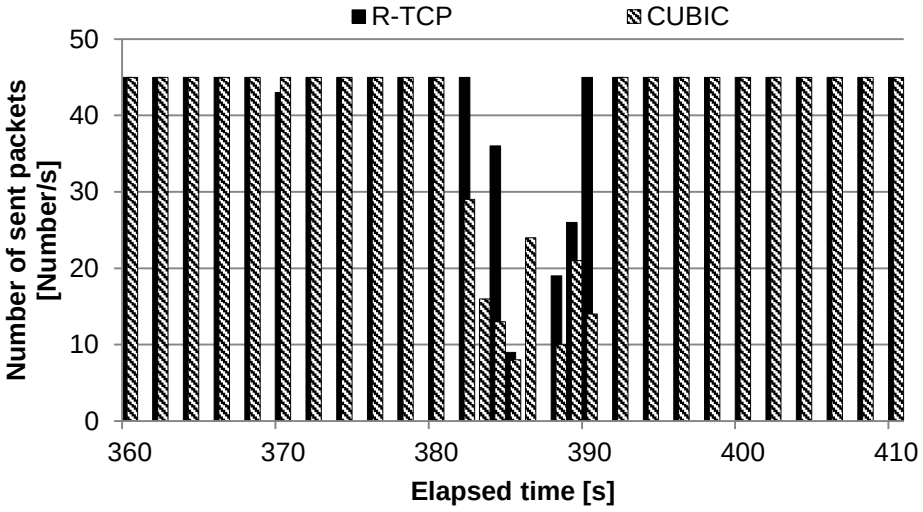


図 6-16 送信パケット数 (LTE 回線利用率が低い移動中車内)

表 6-3 統計データ(LTE 回線利用率が低い移動中車内)

	CUBIC TCP	R-TCP
Average RTT [ms]	169	175
Standard deviation of RTT [ms]	202	159
Average Latency [s]	0.666	0.562
Standard deviation of latency [s]	0.183	0.17
Total latency [s]	680.2	579.9
$P\left\{L+\frac{RTT}{2}<1s\right\}$ [%]	86.0	91.4
Transmitted data [MB]	65.3	66.0

6.3.4 LTE 回線利用率が高い移動中の電車内

図 6-17 から図 6-21 と，表 6-4 に評価結果を示す．本評価は，比較的乗客が多い時間帯の路線(JR 山手線[3])を利用して行った．本環境では，多くの乗客が移動デバイスを用いて通信している．図 6-20 に示すようにサーバと端末間の RTT は 800ms 前後であった．HARQ といった無線レイヤにおける再送の影響や本通信セッションに割当てられる帯域が少ないことから，データの発生速度に対し LTE 網がボトルネックになっていると考えられる．移動しているため図 6-19 に示すように廃棄が多い．このケースでは非バースト的廃棄が多く生じ，バースト的廃棄はなかった．

本環境下では，RTT が 300ms 以上の時に非バースト的廃棄があると図 4-8 に示す TCP Hybla のレイテンシが小さいためこれを選ぶ．非バースト的廃棄がない場合は，図 4-8 では，どのアルゴリズムでも同程度のレイテンシとなるが，バースト的廃棄率が 0%の場合は， 図 4-5，図 4-6 と同様に RTT が 300ms の場合でも TCP Hybla のレイテンシが最小となる(図 4-17 において最もボトルネック帯域幅が大きい場合に TCP Hybla のレイテンシが最小となる)ので，非バース

[3]東京都心部を走る一周 34.5 キロの環状の路線

ト的廃棄率から推定したレイテンシと，バースト的廃棄率から推定したレイテンシの和は TCP Hybla が最小となり TCP Hybla を選ぶ．一方で，173 秒，1554 秒, 1740 秒のように廃棄が発生した際に 100ms から 200ms 程度の RTT となっていた時刻は，図 4-4 や図 4-7 に基づき Scalable TCP を選択する．

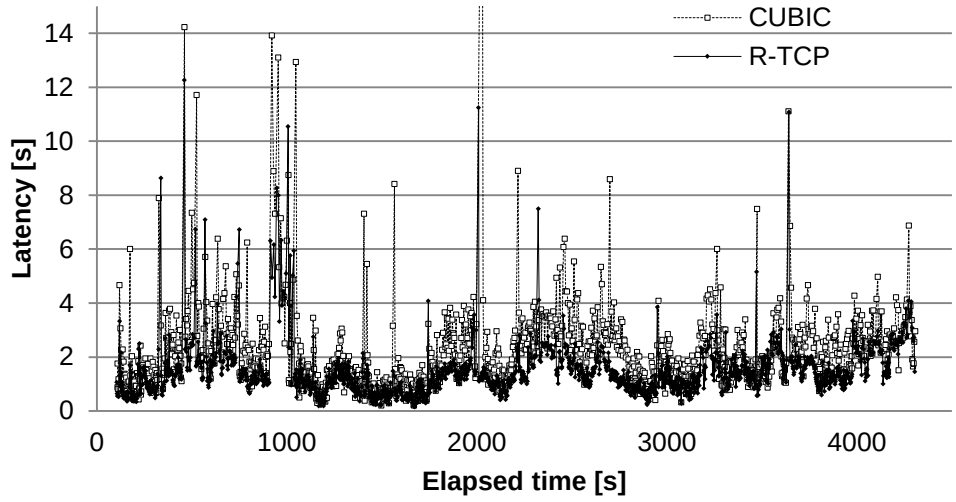


図 6-17 評価結果(LTE 回線利用率が高い移動中車内)

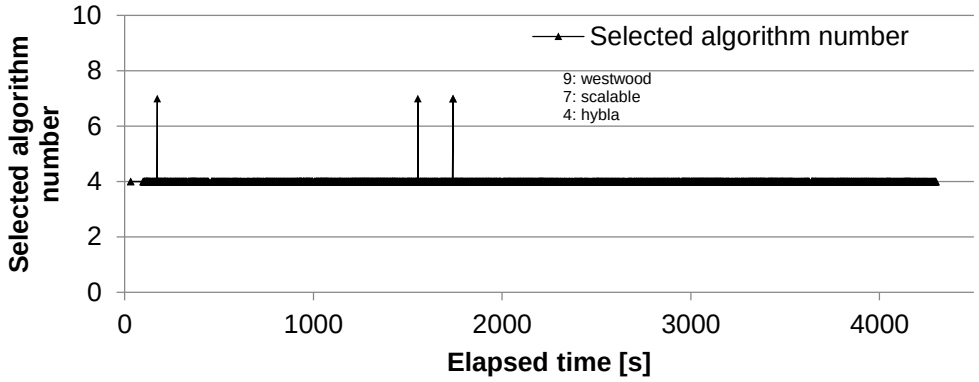


図 6-18 輻輳制御アルゴリズムの変化(LTE 回線利用率が高い移動中車内)

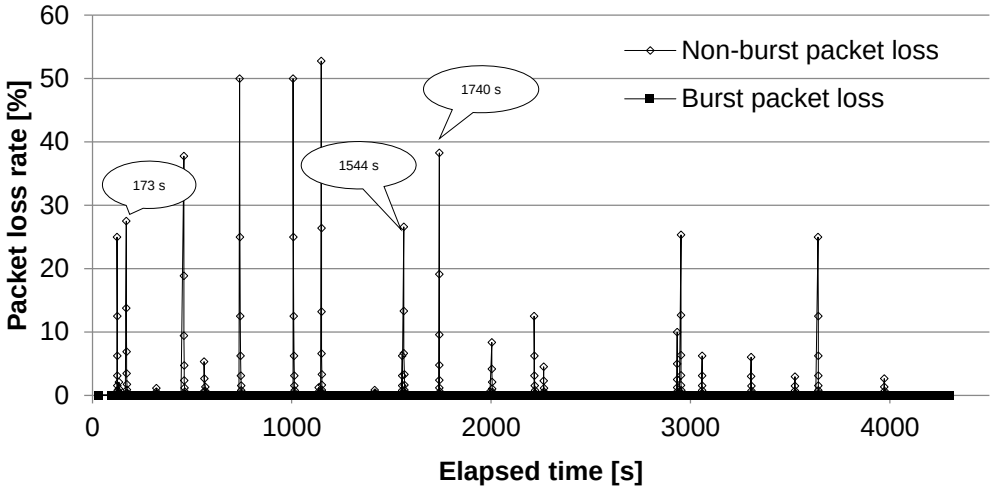


図 6-19 廃棄率の変化(LTE 回線利用率が高い移動中車内)

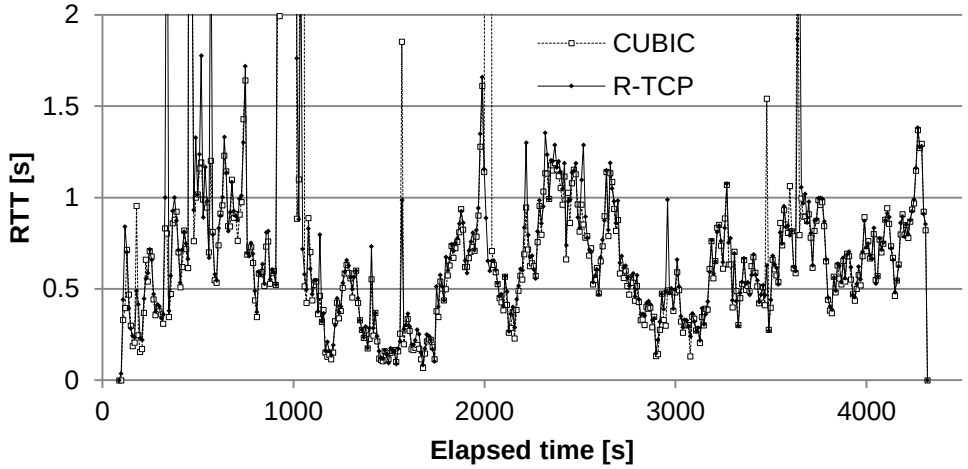


図 6-20 往復遅延時間(LTE 回線利用率が高い移動中車内)

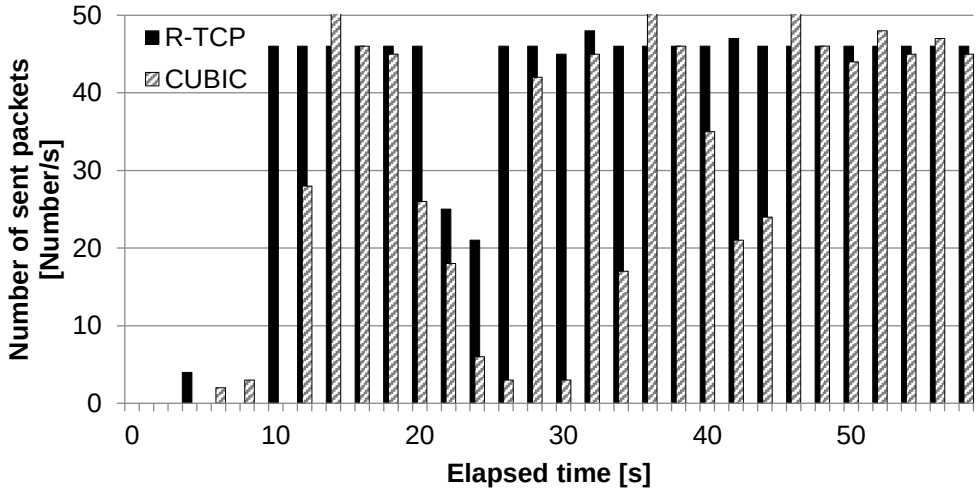


図 6-21 送信パケット数 (LTE 回線利用率が高い移動中車内)

表 6-4 統計データ(LTE 回線利用率が高い移動中車内)

	CUBIC TCP	R-TCP
Average RTT [ms]	854	763
Standard deviation of RTT [ms]	1625	745
Average Latency [s]	1.96	1.40
Standard deviation of latency [s]	2.29	0.876
Total latency [s]	2819.5	2515.1
$P\left\{L+\frac{RTT}{2}<1s\right\}$ [%]	4.46	10.6
Transmitted data [MB]	91.8	114.6

廃棄が発生した際，本ケースのように RTT が極めて大きい場合は，6.3.3 節と異なり輻輳ウィンドウサイズの減少率が小さいアルゴリズムを選ぶよりも，図 6-21 における 25 秒付近に示すように，輻輳ウィンドウサイズを速く広げるアルゴリズム TCP Hybla を選択する方が，速く輻輳ウィンドウサイズを元に戻すことができる．このように， RTT が極めて大きく，かつ廃棄が多く発生している状況でも R-TCP により性能を改善できる．

なお，図 6-21 で，CUBIC TCP においてバーストサイズに相当する 46 パケットより多くのパケットが送信されるのは，バーストデータのパケット送信に時間がかかり 1 秒の時間スロット内で送信しきれず次スロットでカウントされたためである．

また，本ケースでは RTT が大きいため，$P\{L+RTT/2<1s\}$となる確率は CUBIC TCP，R-TCP のどちらもさほど高くない．この場合，TCP の改善だけでは，仮想デスクトップのレスポンス改善は難しく，キャッシュやデータ圧縮など他の高速化技術を組み合わせる必要がある．

6.4　議論

6.3 節では, RTT と廃棄率が異なる各種環境で評価を行ったが, これらが R-TCP の輻輳制御アルゴリズムの選択にどう影響しているかを考察する．また，提案方式は，RTT や廃棄率について限られた範囲のレイテンシ特性テーブルを持つため，提案方式を適用できる範囲と，範囲を超えた場合のふるまいについて考察する. 一方, 評価結果は, 多くの場合に R-TCP が TCP Hybla を選択しており，TCP Hybla との性能差を明らかにするため，常に TCP Hybla を用いる場合と R-TCP により輻輳制御アルゴリズムを切替える場合を比較する．更に，本評価により R-TCP がレイテンシを改善できることを示したが，それにより人間が感じる主観評価をどれほど改善するかを考察しておく必要がある．以下では，これらについて詳細に述べる.

6.4.1　性能に影響を与える要因

図 6-22 にこれらの評価に対する R-TCP のふるまいをまとめた. これによると, インタラクティブ通信のレイテンシには RTT 及びパケット廃棄率が影響を与えることがわかる．今回の評価では，平均 RTT がおよそ 170ms 前後の中位の大きさの場合と，平均 RTT が 800ms と大きい場合の 2 種類があった．また，固定ポイントと電車内で評価することで廃棄の有無が異なった. 6.3.1 節や 6.3.2 節のように，廃棄がない環境の場合，増加率が最も大きい輻輳制御アルゴリズムが優位となる．ここでは，RTT が 170ms 以上では常に増加率が大きい Hybla が選ばれている．また，RTT が中位で廃棄が多い場合には，廃棄による輻輳を検出してもあまり輻輳ウィンドウを下げない Scalable や，非バースト的廃棄では，輻輳を検出しにくい Westwood が選択される．ただし，RTT が 800ms より大きいと，Hybla の増加率は極めて大きくなり，輻輳があっても，すぐに輻輳ウィンドウを元にもどせる Hybla が選択されるようになる.

RTT が増大すると CUBIC TCP は輻輳ウィンドウサイズをゆっくりと広げるが, TCP Hybla は RTT が大きい程，増加率αを大きくし，輻輳ウィンドウサイズを速く広げるため，R-TCP は TCP Hybla を選択することによってレイテンシを改善できる.

一方で，パケット廃棄は，TCP が輻輳ウィンドウサイズを下げる時の挙動に影響を与える. CUBIC TCP が輻輳検出時に輻輳ウィンドウサイズを，減少率βと

して 0.7 倍するのに対し，R-TCP が Scalable TCP を選ぶことで輻輳ウィンドウサイズの低下量を 0.85 倍に抑え，より大きい輻輳ウィンドウサイズから広げ始めるようにしている．

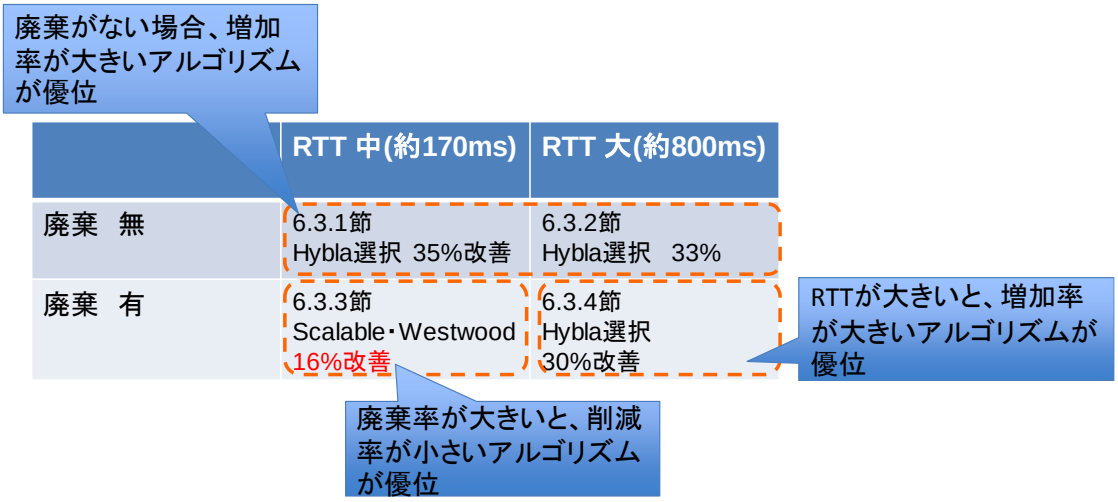


図 6-22 各評価における R-TCP の動作まとめ

6.4.2 提案方式の適用領域の考察

今回の評価結果では，非バースト的廃棄率が 50%，RTT が 2000ms に達し，用意したレイテンシ特性の範囲外となった．提案方式は，非バースト的廃棄が 10%以上の場合は 10%に値を丸め，RTT が 300ms 以上の場合は 300ms に丸めて推定を行う．図 6-23，図 6-24 は，非バースト的廃棄率 10%，及び 50%における RTT が 200ms から 2000ms までのレイテンシ特性を示しており，非バースト的廃棄率が 10%以上，RTT が 300ms 以上では TCP Hybla のレイテンシの優位性は保たれている．そのため，廃棄率，RTT の計測値を，それぞれ 10%，300ms に丸めた値でのレイテンシの推定は，正確なレイテンシを推定はできないが，今回のように非バースト的廃棄が単独で生じている状況であれば，全てのレイテンシ特性を用意せずとも適切な輻輳制御アルゴリズムを選択できることがわかった．

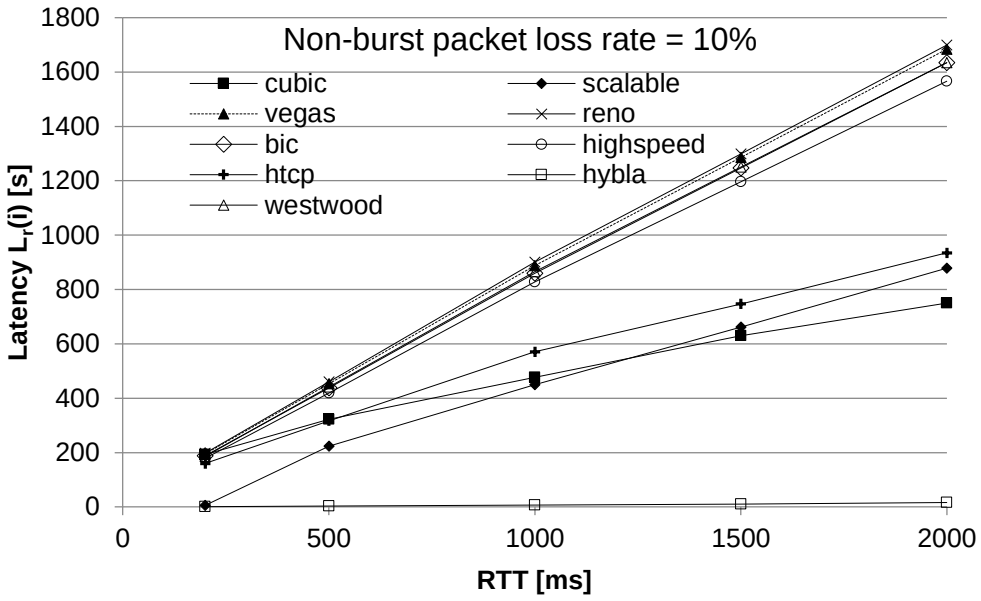


図 6-23 非バースト的廃棄率 10%時のレイテンシ特性 [RTT ≧ 200 ms]

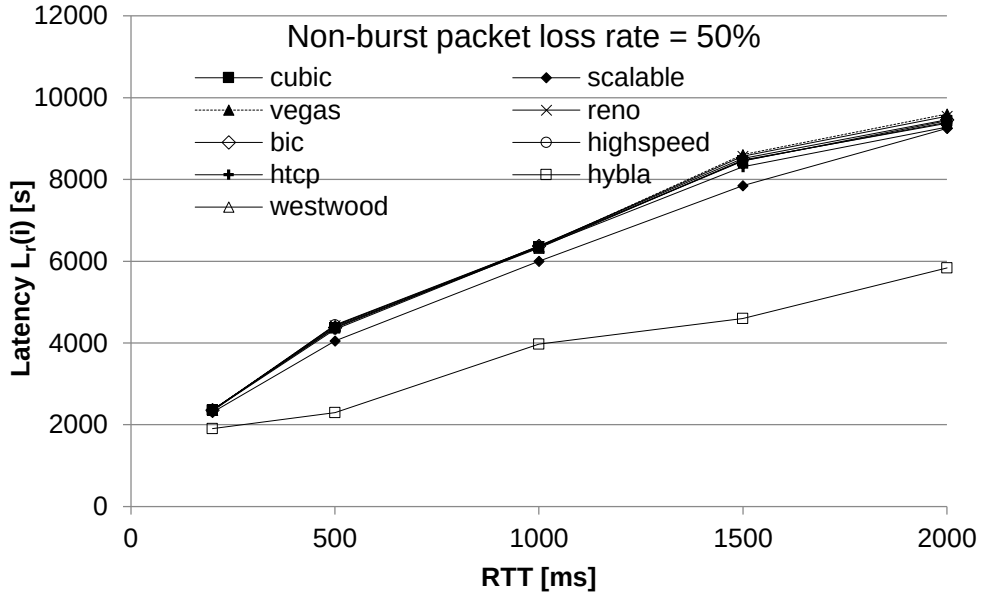


図 6-24 非バースト的廃棄率 50%時のレイテンシ特性 [RTT ≧ 200 ms]

6.4.3 TCP Hybla と R-TCP の比較

6.3.1 節から 6.3.4 節に示した評価では, R-TCP は多くの時点で TCP Hybla を選択し，CUBIC TCP に比べレイテンシを改善することがわかった．レイテンシの改善が TCP Hybla を用いたことによるものか動的な輻輳制御アルゴリズムの切替えによるものかを明らかにするため，輻輳制御アルゴリズムを TCP Hybla に

固定した場合と R-TCP で動的に切替える場合を比較した. 図 6-25 に示すように, 530 秒から 560 秒付近で R-TCP は輻輳制御アルゴリズムを Scalable TCP, もしくは TCP Westwood に切替えている. この際, 図 6-26 に示すように, R-TCP は TCP Hybla に比べレイテンシが小さくなっており, 動的に輻輳制御アルゴリズムを切替えたことによりレイテンシが改善していることがわかる.

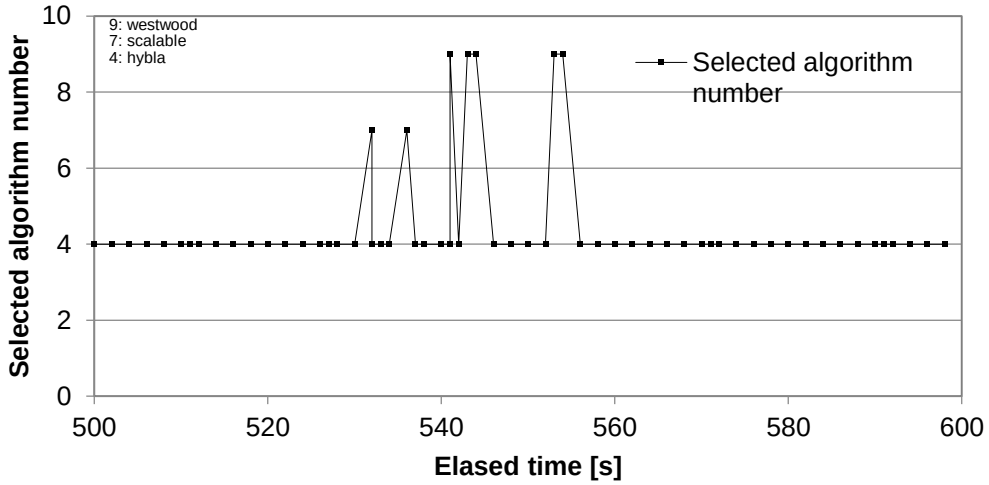


図 6-25 R-TCP の選択アルゴリズム

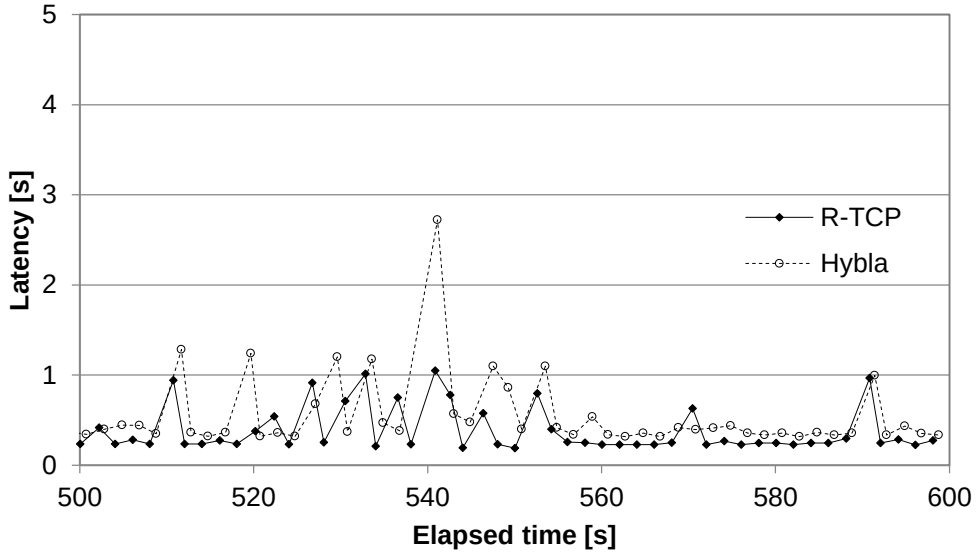


図 6-26 R-TCP と TCP Hybla のレイテンシ比較

6.4.4 主観評価における効果の見積もり

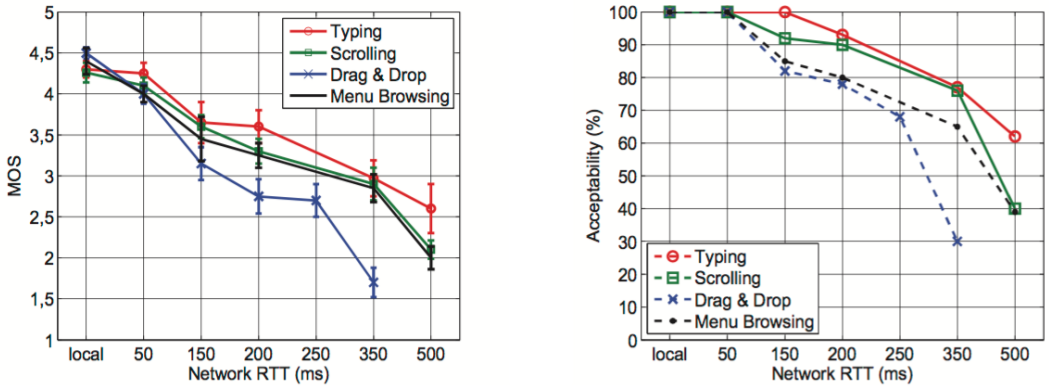
本章の評価により，本提案方式はレイテンシを 30～40%程度改善できることがわかった．ここで，レイテンシの改善が実際にはどの程度の体感品質(QoE: Quality of Experience)につながるかを考えておく必要がある．文献[70][71]は，レ

イテンシの改善が，QoE の改善につながることを示している．これによれば，本提案方式はレイテンシを改善できるため，QoE を改善できると考えられる．

具体的には，文献[71]は，図 6-27 に示すように，0ms から 500ms の RTT の増加に対し Mean Opinion Score (MOS)が，4.5 から 2 まで RTT に応じてほぼリニアに低下することを示している．MOS は，ITU-T 勧告 P.800 に規定されるオピニオン評価であり，表 6-5 に示すように 5 段階で評価し，数字が高いほど QoE が優れていることを示している．

表 6-5 ACR-9 Mean Opinion Score

MOS カテゴリ	評点
非常に良い(Excellenct)	5
良い(Good)	4
普通(Fair)	3
悪い(Poor)	2
非常に悪い(Bad)	1



(a) Overall Quality vs. RTT.　(b) Acceptability vs. RTT.

図 6-27 RTT に対する MOS の変化 (文献[71]より引用)

文献[71]では，図 6-28 に示すように，RTT が短くなると仮想デスクトップの操作に対する全体のレイテンシが短くなり MOS が改善することを示している．すなわち，図 6-28 では，RTT が短い(i)と，RTT が長い(ii)では，データ転送に同じ 3 往復かかっても RTT が短い(i)が，レイテンシが小さくなる．従って(i)の方が MOS を改善できる．一方，提案方式は，図 6-29 に示すように，RTT 自体を変えることはできないが，RTT 内に一度に連続して送ることができるデータ量

(輻輳ウィンドウサイズ)を増やすことで，ACK 待ちとなる回数を減らしレイテンシを改善している．すなわち，図 6-29(a)に示す RTT=X を 3 往復しレイテンシ= Lとなる通信を，1RTT 内で送る輻輳ウィンドウサイズを増やすことで，全データを 2 往復で送信し，レイテンシをL'に短縮している．レイテンシ= L'を(c)に示すように本来の 3 往復で送ったと仮定すると，レイテンシの短縮は RTT を Y (Y<X)に短縮したことと等価と考えることができる．今回の評価では，R-TCP を用いることで, CUBIC に対しレイテンシを表 6-6 に示すように RTT を短縮した(RTT の短縮時間)と考えることができる．すなわち，CUBIC の平均レイテンシを平均 RTT で割ると平均往復数が計算できる．平均レイテンシ改善時間を平均往復数で割った値が，1RTT 当たりの短縮時間と考えることができる．この結果から，回線利用率が低い電車内では一往復当たり 26.4ms，回線利用率が高い屋内では 258.1ms もの RTT を短縮したことと同等の効果を得たと考えることができる．そのため，R-TCP は，回線利用率が低い電車内では MOS を 0.5，回線利用率が高い屋内では MOS を 1.5 改善できると考えられ，体感できるほどの効果を得られると思われる．

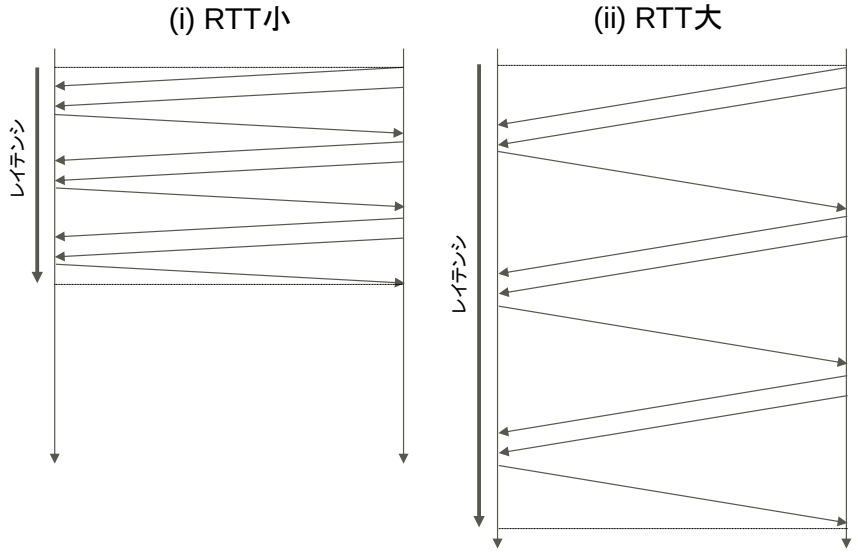


図 6-28 RTT の変化によるレイテンシの改善

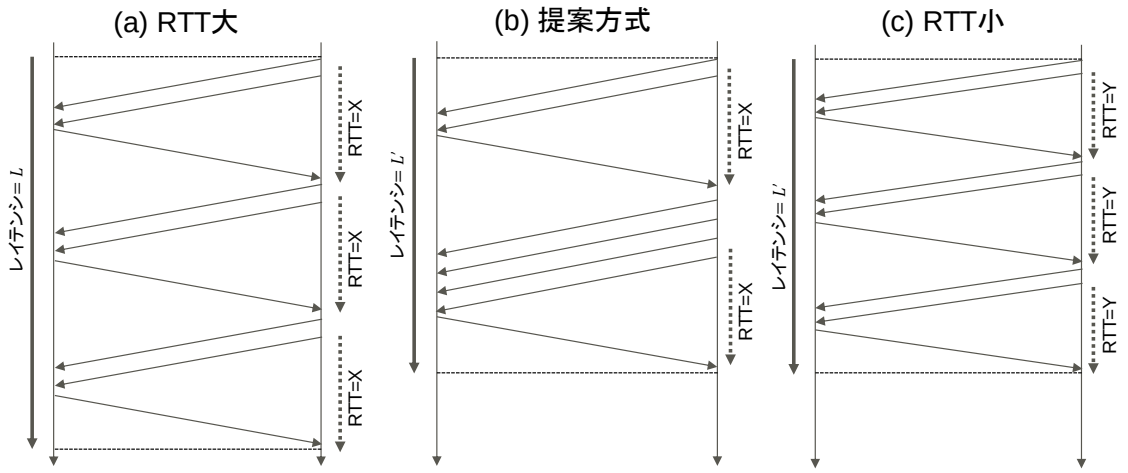


図 6-29 輻輳ウィンドウサイズの増加によるレイテンシの改善

表 6-6 R-TCP によるレイテンシ改善量

評価内容	CUBIC 平均レイテンシ [ms]	平均レイテンシ改善時間 [ms]	平均 RTT [ms]	平均往復数	RTT の短縮時間 [ms]
利用率低, 屋内	695	237ms	184	3.78	62.7
利用率高, 屋内	2350	733ms	826	2.84	258.1
利用率低, 電車	666	104ms	169	3.94	26.4
利用率高, 電車	1960	560ms	854	2.29	244.5

6.5 本章のまとめ

本章では，提案方式を，特性がダイナックに変化する LTE を用いた実環境に適用し，固定ポイント，及び移動する電車内で性能評価を行った．その結果，遅延が大きい環境，廃棄が多い環境であっても，インタラクティブ通信のバーストデータのレイテンシを改善できることを確認した(表 6-7)．また，レイテンシが，アプリケーションのデータ送信間隔より大きい場合には R-TCP により総データ転送量をより多くできることを確認した．

本結果から，RTT の大きさは輻輳ウィンドウサイズ拡大の立ち上がり速度に影響し，廃棄はウィンドウサイズの下げ幅に影響を与え，R-TCP はこれらの状況に合わせ最適な輻輳制御アルゴリズムを選択できることがわかった．

また，主観評価における効果を考察した結果，遅延が大きい環境，廃棄が多い環境であっても，R-TCP は，インタラクティブ通信におけるバーストデータのレイテンシを改善すると共に，体感品質の改善も期待できることを確認した．

バーストデータに対する最後のデータパケット又は ACK が廃棄された場合は，送信側 TCP は次のバーストデータが送信されるまでタイムアウトを待ち再送するためレイテンシが特に大きくなる．この要因によるレイテンシの増加は輻輳制御アルゴリズムの変更では解決できず今後 TCP の仕組みを改良する必要がある．

表 6-7 R-TCP のレイテンシ改善効果

評価環境	改善率 [%]
利用率低, 屋内	34.1
利用率高, 屋内	32.9
利用率低, 電車	15.6
利用率高, 電車	28.6

第7章 結論

本章では，本研究の成果について述べた後，本研究の課題と今後の展望を述べ，本論文の結びとする．

7.1 本研究のまとめ

今日，情報漏洩や省電力に対する社会的な問題意識の高まりとクラウドの普及が相まって, クラウド上の仮想デスクトップサービス(DaaS)の活用に対する期待が増加している．加えて，無線アクセス網の普及により，モバイル環境からの DaaS 利用への期待も大きい．

ところで, 人間が遠隔の PC を操作する仮想デスクトップのようなインタラクティブなアプリケーションは，単位時間における転送データ量はそれほど多くないため平均的なスループットの高さは要求しないが，高い応答性能が求められる．

しかし，これらの通信アプリケーションは TCP プロトコルを用いて通信している場合がほとんどであり，TCP プロトコルは，遅延が大きいネットワークでパケット廃棄が発生すると，通信性能が低下しやすいという特性を持っている．クラウドは，海外にあることも多く，また，営業の担当者などが外に持ち出すモバイル端末からアクセスする場合が多い．このような通信環境は，無線アクセス網によるパケット廃棄が発生することに加え，国際回線における大きい遅延を生じやすい．

遅延が大きいネットワークでパケット廃棄が発生すると，TCP は輻輳ウィンドウを小さくし，遅延が大きく ACK を受信するのに時間がかかるため，輻輳ウィンドウが広がるのに時間がかかり，結果として，仮想デスクトップで送るデータの転送時間が長くなる．また，仮想デスクトップのようなインタラクティブ通信アプリケーションの単位時間当たりのデータ量は比較的小さい．この場合，TCP の輻輳ウィンドウの広がりは，仮想デスクトップアプリケーションのデータ転送速度以上に制約され，輻輳ウィンドウの広がりが遅くなる．その結果，仮想デスクトップが送るデータの転送時間が長くなる．このような原理により TCP を用いてインタラクティブ通信を行う場合，遅延が大きいネットワー

クでパケット廃棄が発生するネットワークでは応答時間が長くなってしまう．

本論文では，バースト的廃棄，非バースト的廃棄といったパケットの廃棄パタンにより TCP の特性が異なることを示し，これら廃棄パタンや RTT に応じて適切な通信性能を示す輻輳制御アルゴリズムに切り替える R-TCP を提案し，Linux OS に実装した．また，本提案方式は，標準 TCP である TCP Reno との公平性を示す TCP Friendliness については，R-TCP が選択する対象の輻輳制御アルゴリズムを限定し，限られた性能改善範囲で利用するならば公平性の差を小さくできることを示した．また，同じ輻輳制御アルゴリズムを用いた複数のセッションの RTT が異なる場合の公平性を示す RTT fairness については，R-TCP は RTT によるレイテンシの差がでにくいため，公平性の差がでにくいことを示した．

さらに，本提案方式を，特性がダイナミックに変化する LTE 網を用いた実環境に適用し，固定ポイント，及び移動する電車内で性能評価を行った．その結果，遅延が大きい環境，廃棄が多い環境であっても，インタラクティブに送るバーストデータのレイテンシを改善できることを確認した．また，無線アクセス網の RTT が無線レイヤの再送により大きくなり，アプリケーションの送信するバーストデータがソケットバッファに蓄積する状況では，ファイル転送に近いデータ転送となるが，この状況でも，R-TCP を用いることで，総データ転送量をより多くできることを確認した．

本結果から，RTT の大きさは TCP 輻輳制御アルゴリズムの輻輳ウィンドウサイズ拡大の立ち上がり速度(増加率)に影響し，廃棄はウィンドウサイズの下げ幅(減少率)に影響を与え，R-TCP はこれらの状況に合わせ最適な輻輳制御アルゴリズムを選択できることがわかった．

また，連続的なパケット廃棄が 3 個以上のバースト的廃棄と，連続的な廃棄が高々2 個の非バースト的廃棄では，Linux OS の TCP は異なる方法で輻輳を検出する．その結果，非バースト的廃棄では，TCP は輻輳ウィンドウサイズをほぼ初期値に近い値まで下げるが，バースト的廃棄の場合は，各輻輳制御アルゴリズムで決まっている削減量まで輻輳ウィンドウサイズを下げる．そのため，バースト的廃棄の方が，輻輳ウィンドウサイズが小さくなり過ぎないことが判った．R-TCP ではこの特性も区別し，最適な輻輳制御アルゴリズムを選択する

ようにした.

無線網における廃棄率の変化は，極めて短時間で変動しており，R-TCP が輻輳制御アルゴリズムを切替える周期を RTT 程度にし，比較的短い周期で切替えた方が効果を得られやすいことがわかった.

また，バーストデータに対する最後のデータパケット又は ACK が廃棄された場合は，送信側 TCP は次のバーストデータが送信されるまでタイムアウトを待ち再送するためレイテンシが特に大きくなる．この要因によるレイテンシの増加は輻輳制御アルゴリズムの変更では解決できず今後 TCP の仕組みを改良する必要がある.

更に，こうした応答時間の改善が主観評価としてどれだけの改善につながるかを考察した結果，ITU-T 勧告で定める 5 段階のオピニオン評価(MOS)のスコアを 0.5 から 1.5 改善できる見通しを得た.

今回の評価は，ネットワークの特性が時々刻々変動する移動無線を用いた環境で行ったが，提案方式は，有線網や固定無線環境でも効果が期待できる．有線網や固定無線環境は，移動無線環境に比べ一般的にはネットワークの特性の変化が少ない．しかし，有線網であっても，利用する時刻によりその特性は変わるほか，アクセス先により通信特性は全く異なる．そのため，通信セッション開始時にネットワークの特性に合わせ最適な輻輳制御アルゴリズムが選択されるだけでも十分効果が見込める.

7.2 課題と今後の展望

今後，提案手法を実用化していくためには，以下のような課題が残っている.

- 混在する通信トラフィックへの対処
- レイテンシ以外の遅延時間
- 提案方式の産業界への還元方法
- 多くの R-TCP が混在する環境への対応

1. 混在する通信トラフィックへの対処

本論文では，インタラクティブ通信に着目し，輻輳制御アルゴリズムを切り替えることで，そのレイテンシが改善できることを示した．本手法は，大規模ファイル転送にも有効と考えられるため，今後は，ファイル転送の場合についても評価を行う必要がある．特に仮想デスクトップでは，その通信セッションの中で，インタラクティブな通信とファイル転送の両方の通信が行われる．どちらの場合であっても，効果が得られることを今後検証していく必要がある．

2. レイテンシ以外の遅延時間

今回の提案方式では，インタラクティブ通信における TCP のデータ転送のレイテンシを改善することでインタラクティブ通信の応答性能を改善することを試みている．しかし，物理的な距離に対する遅延時間や，通信回線上に極端に細いボトルネックがあるといった問題に対しては異なる対処が必要である．例えば，物理的な距離による遅延については，データのキャッシュや，重複データの転送を除去する仕組み，通信アプリケーションの処理シーケンスを簡略化し，無駄なシーケンスが流れることによる時間を節約するといった対処が必要である．通信回線上に極端に細いボトルネックがあるといった問題に対しては，転送するデータを圧縮して送信するなど，他の技術と組み合わせる必要がある．

3. 提案方式の導入方法

今回試作した提案方式の産業界への還元方法としては，Linux OS については，本評価で用いたプログラムコードをオープンソースとして公開することで比較的容易に配布できると考えられる．今回，提案方式の試作では，カーネルコードを修正して実装している．Linux OS カーネルのバージョンが上がった場合のポータビリティを考慮すると今後はユーザ空間でのプロセスとしての実装がより望ましい．また，Windows といった商用の OS については，TCP をプロキシするソフトとしてユーザの端末内に同居させるか，中継パスの途中にプロキシサーバとして配置するなどの仕組みを用いることで適用できると思われる．

4. 多くの R-TCP が混在する環境への対応

R-TCP は，各ホストが自律分散的に廃棄率や RTT を計測し，輻輳制御アルゴリズムを選択する．この方式では，輻輳発生をトリガに輻輳制御アルゴリズム

を切替えるため，ネットワークの利用効率が低下する．同一ネットワークで利用される通信アプリケーションの特性や，ネットワークの混雑具合，無線網の電波状況などの情報をトリガに輻輳制御アルゴリズムを切替えることができればより早いタイミングで切替えることができ，更に適切な制御が可能になると考えられる．

この課題に対しては，近年普及してきた SDN(Software Defined Network)と呼ばれる，コントローラからネットワーク機器を集中制御する手法が活用可能と思われる．今回の評価で，Linux OS カーネル内に実装した輻輳制御アルゴリズムを選択するロジックである R-TCP コントローラの機能を SDN のコントローラに持たせ，ネットワーク内の他 TCP セッションの利用状況や，ネットワークの混雑具合を踏まえながら，SDN のコントローラが各ホストの TCP 輻輳制御アルゴリズムを制御することにより，フェアネスへの配慮もより適切に行うことが可能となり，ネットワーク全体として効率的な制御が可能になると思われる．特にデータセンタ内に配置されたサーバにこうした制御を行うと効果的と思われる．

謝辞

阿部俊二准教授に於かれましては，総合研究大学院大学複合科学研究科情報学専攻博士後期課程の長きにわたりまして，ご指導を賜りました．先生には，研究の過程，内容のあり方，進め方等に関しましてもご指導頂き，また研究内容以外にも研究生活でのサポートをいただきました．心より感謝いたします．

漆谷重雄教授に於かれましては，研究への貴重なご意見，ご指摘をいただきました．心より感謝しております．

計宇生教授に於かれましても，長きにわたりご指導をいただきました．研究の要所に関しまして，その都度，貴重な助言をいただきまして深く感謝いたしております．

福田健介准教授に於かれましては，私の研究の状況に対して貴重なご意見をいただきまして，心より感謝いたします．

山田茂樹名誉教授に於かれましては，私が気づくことの無かった事項に関して大変貴重なご意見をいただきました．心より感謝しております．

大阪府立大学戸出英樹教授に於かれましては，国際会議，研究会において気にかけていただき，貴重なご意見をいただきました．心より感謝しております．

最後になりましたが，研究生活以外にも様々な点で支えて頂きました総合研究大学院大学，国立情報学研究所の職員の皆様，富士通研究所の諸先輩方，また友人家族に深く感謝しております．

参考文献

[1] 環境省, 地球環境・国際環境協力: http://www.env.go.jp/earth/ondanka/ghg/ert2020.html

[2] 一方井 誠治, 栗田 郁真, 堀 勝彦, “企業における温室効果ガス削減対策に関する実態調査,” KIER Discussion paper series, Kyoto institute of economic research, May 2011.

[3] 総務省, 平成 22 年版 情報通信白書, http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h22/html/md212000.html

[4] Microsoft, “Remote Desktop Protocol: Basic Connectivity and Graphics Remoting,”

[5] Tristan Richardson, “The RFB Protocol,” http://www.realvnc.com/docs/rfbproto.pdf

[6] 森 正弥, ”インターネットにおけるデバイス環境の多様化，サーバ環境の大規模化，プラットホームのエコシステム化への対応,” 電子情報通信学会論文誌. B, 通信 J93-B(10), 1348-1355, Oct. 2010.

[7] Madden B., Knuth G., “Desktops as a Service: Everything You Need to Know About DaaS & Hosted VDI,” Burning Troll Production, San Francisco, California, 2014.

[8] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen, “Gaming Anywhere: An Open Cloud Gaming System,” Proceeding of MMSys '13 Proceedings of the 4th ACM Multimedia Systems Conference, Pages 36-47, Feb. 2013.

[9] salesforce.com , http://www.salesforce.com/jp/

[10] Office 365, http://www.microsoft.com/ja-jp/office/365/

[11] Google App Engine, https://cloud.google.com/appengine/

[12] Amazon EC2, https://aws.amazon.com/jp/ec2/

[13] Hadoop, https://hadoop.apache.org/

[14] Amazon EMR, https://aws.amazon.com/jp/elasticmapreduce/

[15] Amazon Simple Storage Service(S3), https://aws.amazon.com/jp/s3/

[16] Dropbox, https://www.dropbox.com/ja/

[17] Google Drive, https://www.google.com/intl/ja_jp/drive/

[18] iCloud, http://www.apple.com/jp/icloud/

[19] One drive, https://onedrive.live.com/about/ja-jp/

[20] V-DaaS, http://fenics.fujitsu.com/outsourcingservice/lcm/workplacelcm/virtualdesktop.html

[21] Amazon Workspaces, http://aws.amazon.com/jp/workspaces/

[22] Cloud Gaming Platform, http://www.datahotel.ne.jp/info/cloudgaming/

[23] G-cluster, http://gcluster.jp/

[24] V. Jacobson, R. Braden "TCP Extensions for Long-Delay Paths," IETF RFC1072, Oct. 1988.

[25] Tolia, N., Andersen, G. D. and Satyanarayanan, M.: Quantifying Interactive User Experience on Thin Clients, IEEE Computer, vol.39, no.3, pp.46-52, 2006.

[26] Nielsen, J.: Usability Engineering, pp. 115-164, Academic Press, 1993.

[27] Sangtae Ha, Injong Rhee and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," ACM SIGOPS Operating Systems Review, vol.42, no.5, pp.64-74, July 2008.

[28] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control BIC for Fast Long-Distance Networks," INFOCOM 2004, vol.4, pp.2514-2524, Mar. 2004.

[29] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," in Proc. 1999 IEEE Symposium on Application-Specific Systems and Software Engr. and Techn., pp. 10–17, Richardson, TX, Mar 1999.

[30] S. Biaz and N. H. Vaidya, "Distinguishing Congestion Losses from wireless Transmission Losses: a Negative Result", Proc. of IEEE 7th Int. conf. on computer communications and Net works, New Orleans, LA, USA, Oct. 1998.

[31] Song Cen, Pamela C. Cosman, and Geoffrey M. Voelker, “End to end Differentiation of congestion and wireless losses,” ACM/IEEE transaction on networking, vol.11, no.5, pp.703-717, Oct. 2003.

[32] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, “Achieving moderate fairness for UDP flows by path-status classification,” in Proc. 25th Annual IEEE Conf. on Local Computer Networks (LCN 2000), pp. 252-261, Tampa, FL, Nov. 2000.

[33] WAP 2.0 Technical White Paper, Open mobile alliance, Jan. 2002.

[34] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, F. Khafizov, “TCP over Second (2.5G) and Third (3G) Generation Wireless Networks,” IETF RFC3481, Feb. 2003.

[35] Hari Balakrishnan, Srinivasan Seshan and Randy H.Katz, “Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks,” ACM Wireless Networks, May 1995.

[36] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan and Randy H.Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Transactions on Networking, vol5, no.6,pp756-769, Dec. 1997.

[37] Yin Zhang, Lili Qiu, and Srinivasan Keshav, “Optimizing TCP Start-up Performance,” computer science technical report, Feb. 1999.

[38] J.Chu, N. Dukkipati, Y. Cheng, M. Mathis, “Increasing TCP's Initial Window,” IETF RFC6928, Apr. 2013.

[39] Keith Winstein and Hari Balakrishnan, “TCP ex Machina: Computer-Generated Congestion Control,” Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp.123-134, New York, USA, 2013.

[40] Xiaoxin Qiu, and Kapil Chawla, “On the Performance of Adaptive Modulation in Cellular Systems, IEEE Transactions on Communications, vol.47, no.6, June 1999.

[41] Emina Soljanin, Ruoheng Liu, and Predrag Spasojevic, “Hybrid ARQ with Random Transmission Assignments,” DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 2004.

[42] Kun Tan Jingmin Song Qian Zhang, Murari Sridharan, “A Compound TCP Approach for High-speed and Long Distance Networks,” Proceedings of IEEE INFOCOM, Apr. 2006.

[43] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, N. Vaidya, “Long Thin Networks ,” IETF RFC2757, Jan. 2000.

[44] V. Jacobson, R. Braden, D. Borman, “TCP Extensions for High Performance,” IETF RFC1323, May 1992.

[45] M. Allman, S. Floyd, C. Partridge, “Increasing TCP's Initial Window,” IETF RFC2414, Sep. 1998.

[46] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgment Options,” IETF RFC2018, Oct. 1996.

[47] Oguchi, N. and Abe, S., “Performance Evaluation of Reconfigurable TCP Communication on Wi-Fi Network Using GE Channel Model,” Proc. IEEE 38th Annual Computer Software and Applications Conference (COMPSAC), pp. 541-550, July 2014.

[48] A. Tang, D. Wei, S. H. Low, “Heterogeneous Congestion Control: Efficiency, Fairness and Design,” Proceedings of the 14th IEEE International Conference on Network Protocols ICNP ’06, pp.127-136, Nov. 2006.

[49] Gilbert, E.N., “Capacity of a Burst-Noise Channel,” Bell System Technical Journal vol.39, no.5, pp.1253-1265, Sep. 1960.

[50] E. 0. Elliott, “Estimates of error rates for codes on burst-noise channels,” Bell System Technical Journal, vol.42, pp.1977-1997, Sep. 1963.

[51] Hong Shen Wang, Moayeri, N., “Finite-state Markov channel-a useful model for radio communication channels,” IEEE Transactions on Vehicular Technology, vol.44, no.1, pp.163-171, Feb. 1995.

[52] Gnu Octave, https://www.gnu.org/software/octave/

[53] K. Levenberg. “A Method for the Solution of Certain Non-Linear Problems in Least Squares,” The Quarterly of Applied Mathematics, vol.2, pp.164-168, 1944.

[54] D.W. Marquardt. “An algorithm for least-squares estimation of nonlinear parameters,” Journal of the Society for Industrial and Applied Mathematics, vol.11, no.2, pp.431-441, June 1963.

[55] Hata M, “Empirical formula for propagation loss in land mobile radio services,” IEEE Transactions on Vehicular Technology, vol.VT-29, pp. 317-325, Aug. 1980.

[56] COST Action 231 Digital mobile radio towards future generation systems Final report, European Communities, 1999.

[57] Carlo Caini and Rosario Firrincieli, “Tcp hybla: a tcp enhancement for heterogeneous networks,” International journal of satellite communications and networking, 2004.

[58] Floyd, S.: HighSpeedTCP for large congestion windows, Internet Draft draft-floyd-tcphighspeed-01.txt, Aug. 2003.

[59] D.Leith, R. Shorten, “H-TCP: TCP for high-speed and long-distance networks,” Proc. PFLDnet, Argnonne, 2004.

[60] D.Leith, R. Shorten, Y.Li, “H-TCP: A framework for congestion control in high-speed and long-distance networks,” HI Technical Report, Aug. 2005.

[61] Kelly, T.: Scalable TCP: Improving Performance in Highspeed Wide Area Networks, ACM SIGCOMM Computer Communication Review, vol.33, no. 2, pp. 83-91, Apr. 2003.

[62] L. S. Brakmo, S. W.O’Mally, and L. L. Peterson,“TCP Vegas: New techniques for congestion detection and avoidance,” Proc. ACM SIGCOMM’94,pp.24-35, Oct. 1994.

[63] Mohammad T. Kawser, Nafiz Imtiaz Bin Hamid, Md. Nayeemul Hasan, M. Shah Alam, and M. Musfiqur Rahman.: Limiting HARQ Retransmissions in Downlink for Poor Radio Link in LTE, International Journal of Information and Electronics Engineering, vol.2, no. 5, Sep. 2012.

[64] NS-3, https://www.nsnam.org/

[65] NSC, http://www.wand.net.nz/~stj2/nsc/

[66] Lawrence S. Brakmo, and Larry L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas in Communications, vol.13, no.8, pp.1465-1480, Oct. 1995.

[67] R.N.Shorten, D.J.Leith, J.Foy and R.Kilduff, “Analysis and design of congestion control in synchronised communication networks,” Proc. 12th Yale Workshop on Adaptive & Learning Systems, May 2003.

[68] G. Haßlinger, O Hohl, “The Gilbert-Elliott Model for Packet Loss in. Real Time Services on the Internet,” Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference Germany, pp.1-15, Apr. 2008.

[69] Iperf.fr, https://iperf.fr/

[70] C. Pedro, P. Dimitra, S. Giuseppe, S. Stefan, Z. Patrick and S. Raimund, “Quality of Experience in Telepresence and Remote Collaboration Cloud Services,” Proc. 4th International Workshop on Perceptual Quality of Systems, pp.167-172, Sep. 2013.

[71] P. Casas, M. Seufert, S. Egger, and R. Schatz, “Quality of Experience in Remote Virtual Desktop Services,” Proc. IFIP/IEEE IM2013 Workshop: 1st International Workshop on Quality of Experience Centric Management (QCMan), pp.1352-1357, May 2013.

[72] 日本電信電話株会社, 東日本電信電話株式会社, 西日本電信電話株式会社, ”フィールドトライアル版 次世代ネットワークインタフェース資料,” Jul. 2006

本論文に含まれる発表文献

査読つきジャーナル論文

1. 小口直樹, 阿部俊二, “非連続データ転送の通信性能を改善する Reconfigurable TCP 技術の提案,” 電子情報通信学会論文誌 B, vol.J98-B, no.10, pp.1070-1083, Oct. 2015.

査読つき国際学会

1. N. Oguchi, S. Abe, “Performance Evaluation of Reconfigurable TCP Communication on Wi-Fi Network Using GE Channel Model,” Proceeding of the 38th International Conference on Computer Software and Applications Conference (COMPSAC 2014), pp.541-550, July 2014. (フルペーパー)
2. N. Oguchi, S. Abe, “Dynamic Communication Protocol for Quick Response in Interactive Communication,” Proceeding of the 12th International Symposium on Applications and the Internet (SAINT 2012), pp.226-231, July 2012. (フルペーパー)
3. N. Oguchi, S. Abe, “Reconfigurable TCP: An Architecture for Enhanced Communication Performance in Mobile Cloud Services,” Proceeding of the 11th International Symposium on Applications and the Internet (SAINT 2011), pp.242-245, July 2011. (ポスター, 査読あり)

研究会

1. 小口直樹, 阿部俊二, “モバイル網を介したインタネット通信の性能改善手法の検討,” 電子情報通信学会技術研究報告(NS 研究会) vol.114, no. 477, pp.307-312, Mar. 2015.
2. 小口直樹, 阿部俊二, “ネットワーク特性に応じた動的 TCP 通信方式を Wi-Fi 網に適用した場合の GE モデルによる性能評価,” 電子情報通信学会技術研究報告(NS 研究会) vol.113, no.472, pp.573-578, Mar. 2014.

3. 小口直樹, 阿部俊二, “TCP 輻輳制御アルゴリズムの動的切替によるインタラクティブ通信性能改善方式の検討,” 電子情報通信学会技術研究報告(NS 研究会) vol.113, no.205, pp93-98, Sep. 2013.
4. 小口直樹, 阿部俊二, “モバイル環境における TCP を用いたインタラクティブ通信の性能改善技術の検討,” 電子情報通信学会技術研究報告(NS 研究会) vol.112, no.463, pp.355-360, Mar. 2013.
5. 小口直樹, 阿部俊二, “バースト通信がインタラクティブ通信の応答性能に与える影響の評価,” 電子情報通信学会技術研究報告(NS 研究会) vol.112, no.27 pp.1-6, May 2012.
6. 小口直樹, 阿部俊二, “高速レスポンスを実現する適応型通信性能改善技術の評価,” 電子情報通信学会技術研究報告(NS 研究会) vol.111, no.468, pp.47-52, Mar. 2012.
7. 小口直樹, 阿部俊二, “モバイルクラウドを実現する通信性能改善技術の検討,” 電子情報通信学会技術研究報告(NS 研究会) vol.110, no.448, pp.467-472, Feb. 2011.

電子情報通信学会総合大会

1. 小口直樹, 阿部俊二, “通信レイテンシを改善する適応型通信技術の検討,” 電子情報通信学会総合大会講演論文集 2012 年 p.141, Mar. 2012.