平成23年度　修士論文

無線LAN環境におけるTCP差別化と無線リソース利用効率に関する研究

早稲田大学　基幹理工学研究科　情報理工学専攻

5110B071-6

園田　和秀

指導　甲藤　二郎　教授

2012 年 1月 31日

指導教授印 受付印

指導教授印	受付印

目次

第１編　TCP バージョン識別と EDCA 制御を併用した TCP 差別化手法

第２編　確率的に変動するリソースを考慮した最適寄り道経路特性

第１編　TCP バージョン識別と EDCA 制御を併用した TCP 差別化手法

第1章 序論

1.1 はじめに

近年、無線 LAN の普及に伴い、ノート PC やスマートフォン、タブレット端末といった様々な無線デバイスが登場してきている。また、無線デバイスの多様化に伴い、モバイル無線ルータの普及や地下鉄での Wifi 利用が可能になるなど、無線 LAN の利用機会が急激に増大してきている。今後も無線通信速度の向上や周波数の有効利用を目的としたコグニティブ無線通信技術により、無線 LAN の利用機会の更なる増大が予想される。

一方、Youtube や Ustream などの映像配信や Skype などによるビデオ通話の需要が増大してきている。現在、映像配信には通信プロトコルとして、主に TCP が使用されている。TCP には TCP-Reno に代表される様々な輻輳制御方式が提案されている。これらは大きく、パケットロスを輻輳検出の手段に用いる loss-based 手法、RTT を輻輳検出手段に用いる delay-based 手法、loss-based 手法と delay-based 手法の利点を組み合わせた hybrid 手法に分けられる。現在提案されている TCP 輻輳制御の多くは、無線 LAN 環境において、CSMA/CA の送信待ち時間の影響により、バッファリング時間が増大し、RTT が大きくなってしまう。一方、TCP 輻輳制御の中で、delay-based 手法の TCP Vegas は、RTT を基に輻輳制御を行うことで、バッファにパケットを溜めないため、RTT を抑えることができる。また、バッファ溢れによる再送を最小限に抑えることができる。このため、単独では無駄のない通信が可能である。このような特性から、近年普及してきているライブストリーミングのようなリアルタイム通信に適していると考えられる。しかし、TCP Vegas は TCP Reno などの他の TCP と競合した際に、他 TCP に帯域を奪われてしまうといった親和性の問題がある。このような問題を改善するために、TCP バージョンの推定や識別が行われている。

1.2 研究目的

本研究では、リアルタイム通信など低遅延が求められる通信を有効に行うことを考え、低遅延な通信が可能である delay-based 手法の TCP Vegas を活かすため、TCP バージョン間の親和性の問題の改善を目的とする。そこで、ルータや NIC で TCP バージョンの識別を行い、TCP Vegas を優先させることによって、TCP バージョンを差別化することを検討する。そして、シミュレーションと実機実験によって評価を行い、TCP 差別化の効果を明らかにする。

1.3 本論文の構成

第 2 章では TCP の輻輳制御方式、特に本論文で使用した従来の TCP の輻輳制御方式について述べる。

第 3 章では無線 LAN のアクセス制御方式である IEEE802.11DCF と本稿での利用した優先制御の基となっている IEEE802.11eEDCA について述べる。

第 4 章では TCP バージョン推定・識別に関する従来研究について述べる。

第 5 章では提案手法である TCP 差別化手法について述べる。

第 6 章ではシミュレーション・実機実験の概要、結果について述べる。

第 7 章ではまとめと今後の課題について述べる。

第2章 TCP の輻輳制御方式

2.1 TCP の輻輳制御[1,2]

TCP の輻輳制御では、1 ラウンドトリップタイム(RTT)に転送できるデータ数をウィンドウサイズという概念を用いて制限し、データ転送を行う。ネットワークの状態に応じてウィンドウサイズを増減させることにより、送信量を調節している。

TCP の輻輳制御では、スロースタートフェーズと輻輳回避フェーズの 2 つから構成されている。以下では TCP の輻輳制御方式の基本となっている TCP-Tahoe のスロースタート、輻輳回避、Delayed ACK について述べる。

2.1.1 スロースタートフェーズ

TCP は新しい通信を開始する場合や輻輳期間の後に再びトラフィックを増加させようとする場合に、スロースタートフェーズに移行する。輻輳ウィンドウの初期値は 1 に設定されている。そして、確認応答セグメント(ACK)を受信する度に 1MSS(Maximum Segment Size)分の大きさだけ輻輳ウィンドウの値を増加させていく。つまり、スロースタートフェーズでは、輻輳ウィンドウは 1 ラウンドトリップごとに 2 倍ずつ更新されていく指数関数的増加となる。増加前の輻輳ウィンドウの大きさを $cwnd$、増加後の輻輳ウィンドウの大きさを $cwnd_{new}$、最大セグメント長を MSS とする。スロースタートフェーズの輻輳ウィンドウの上げ方は以下の式(2.1)のように示すことが可能である。

$$cwnd_{new} = cwnd + MSS \tag{2.1}$$

図 2.1 にスロースタートフェーズの通信の様子を示す。

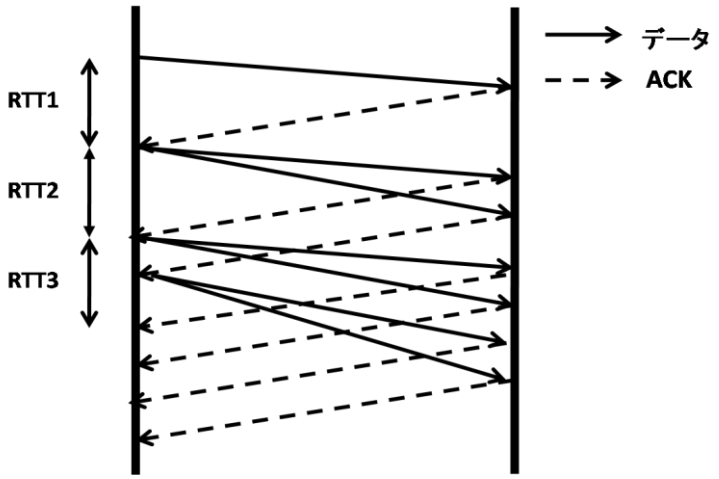


図 2.1　スロースタートフェーズ(出典：参考文献[2])

2.1.2 輻輳回避フェーズ

輻輳回避フェーズでは、TCP は ACK を受け取る毎に、輻輳ウィンドウサイズ分の 1 だけ輻輳ウィンドウを増加させる。通常、TCP では 1RTT 間に輻輳ウィンドウサイズ分の ACK を受信するため、輻輳回避フェーズでは 1RTT 毎に 1MSS 分輻輳ウィンドウが増加することになる。増加前の輻輳ウィンドウの大きさを *cwnd*、増加後の輻輳ウィンドウの大きさを $cwnd_{new}$、最大セグメント長を MSS とすると、この操作は以下の式(2.2)のように表すことができる。

$$cwnd_{new} = cwnd + 1MSS \times MSS / cwnd \qquad (2.2)$$

図 2.2 に輻輳回避フェーズにおける TCP の通信を示す。

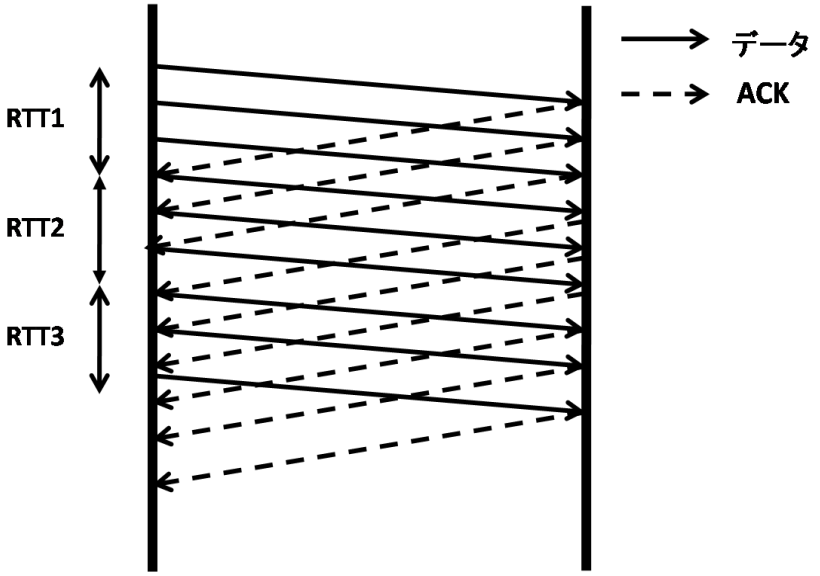


図 2.2 輻輳回避フェーズ(出典：参考文献[2])

2.1.3 通信フェーズの移行

TCP では、輻輳ウィンドウの値がパケット廃棄またはタイムアウトを検出した際に設定されるスロースタート閾値(ssthresh)を超えると、スロースタートフェーズから輻輳回避フェーズへと移行する。TCP-Tahoe では、パケット廃棄を検出すると輻輳ウィンドウを 1、スロースタート閾値を輻輳検出時の輻輳ウィンドウの半分に設定し、再びスロースタートフェーズを開始する。図 2.3 に TCP-Tahoe の輻輳ウィンドウの変化を示す。

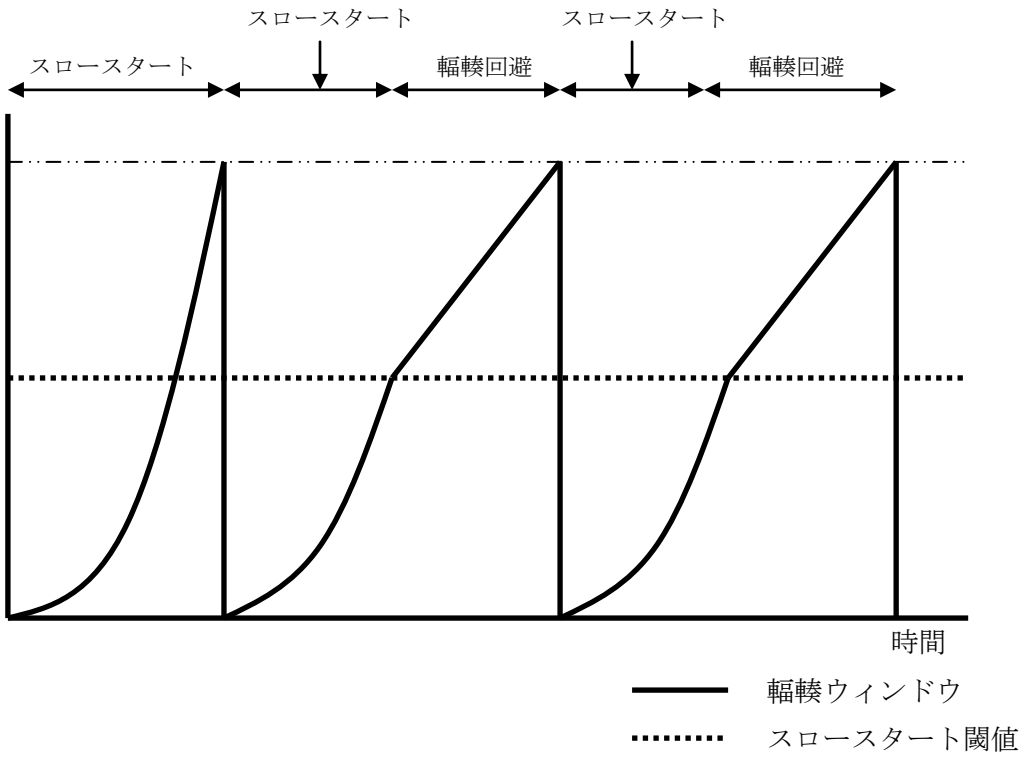


図 2.3　TCP-Tahoe の輻輳ウィンドウの変化(出典：参考文献[2])

2.1.4　Delayed ACK[3]

通常、TCP はデータパケット 1 個に対して 1 個の ACK を返す。Delayed ACK は、受信したデータパケットに対して即座に ACK を返すのではなく、データパケット複数個に対して 1 個の ACK を返す。もしくは、一定時間にパケットが届かなった場合に ACK を返す。Delayed ACK によって、ACK の個数が減少するため、ネットワーク利用効率が良くなる。Linux では、デフォルトで Delayed ACK が有効になっている。RFC1122 では、データパケット 2 個に対して 1 個の ACK を返し、遅延は最大 500[ms]となっている。図 2.4 に Delayed ACK の通信の様子を示す。

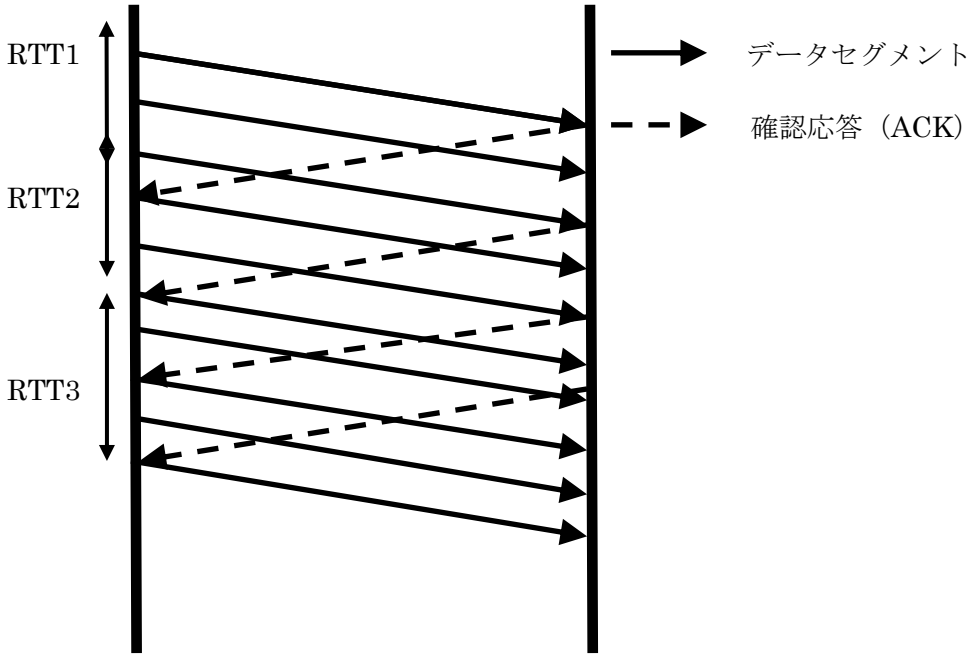


図 2.4　Delayed ACK 使用時のパケット送信の様子

2.2 TCP-Reno[4]

TCP-Reno は多くの loss-based 手法の基本となっている。TCP-Tahoe を用いた場合、1%のパケットロスに対して、50%～70%の転送性能の劣化が生じることが指摘されている。TCP-Reno はこの点を改善するために、TCP-Tahoe の輻輳制御アルゴリズムに高速リカバリアルゴリズムと呼ばれる機構を加えている。これにより、パケットロスによる転送性能の劣化を抑えることができる。TCP-Reno のパケットロスの検出方法も TCP-Tahoe と同様に、再送タイムアウトと重複 ACK によって行うが、TCP-Reno では、輻輳を検出すると、輻輳ウィンドウをスロースタート閾値と同じ半分に設定される。したがって、輻輳ウィンドウを減少させた後、輻輳回避フェーズへと移行する。このアルゴリズムを高速リカバリアルゴリズムと呼ぶ。タイムアウトが起きた場合は TCP-Reno も TCP-Tahoe と同じく、輻輳ウィンドウを 1 にする。

図 2.5 に TCP-Reno の輻輳ウィンドウの変化を示す。

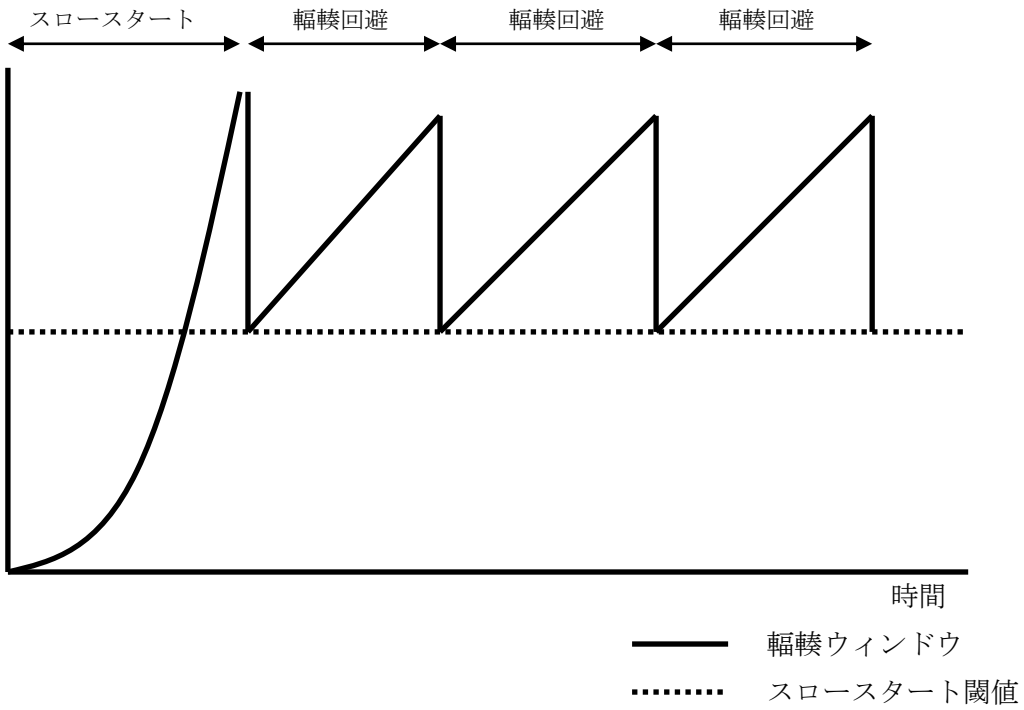


図 2.5　TCP-Reno によるウィンドウサイズの変化(出典：参考文献[2])

現在では、TCP-Reno の高速リカバリアルゴリズムを改良したアルゴリズムを使用している TCP-NewReno[5]が広く使用されている。TCP-NewReno は重複 ACK を 3 回受信することによって再送するというアルゴリズムはTCP-Renoと同じだが、再送後の制御として、パケットが連続して損失していた場合の改良を加えている。TCP-Reno では、パケットが連続して損失していても、パケットが損失するたびに重複 ACK を 3 回受信してから再送を行う。再送の度に輻輳ウィンドウを半分にするためにスループットが向上しない問題がある。この問題を解決するため、TCP-NewReno では重複 ACK を 3 回受信するまでに送信したパケットの ACK が送られてくるまでは、再送モードを維持し続けるようにしている。

図 2.6 に TCP-NewReno の再送制御の様子を示す。

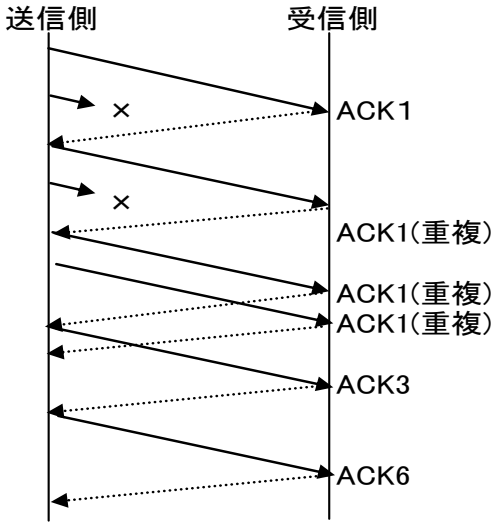


図 2.6　TCP-NewReno の再送制御(出典：参考文献[2])

2.3 TCP-Vegas[6,7]

TCP-Vegas は多くの delay-based 手法の基本となる輻輳制御アルゴリズムを使用している。TCP-Vegas は以下の式(2.3)により、ネットワーク上に蓄積されているパケット数を推測する。

$$diff = \frac{cwnd}{baseRTT} - \frac{cwnd}{RTT} \tag{2.3}$$

ただし、cwnd は輻輳ウィンドウサイズ、baseRTT は今までに観測された RTT の最小値、RTT は最新の RTT の値、diff はボトルネックとなるルータのキュー内パケット数の推定値である。TCP-Vegas では diff の値に基づき、以下の式(2.4),(2.5)に従って輻輳ウィンドウサイズを RTT に 1 度増減させる。

[Slow Start Phase]

$$cwnd = \begin{cases} cwnd + 1, & if \ \ diff < \gamma \\ cwnd \times (1 - p), & if \ \ diff \geq \gamma \end{cases} \tag{2.4}$$

[Congestion Avoidance Phase]

$$cwnd = \begin{cases} cwnd + 1, & if \ \ diff < \alpha \\ cwnd, & if \ \ \alpha < diff < \beta \\ cwnd - 1, & if \ \ \beta < diff \end{cases} \tag{2.5}$$

ここで、γ,p, α, β は制御パラメータである。

TCP-Vegas では、他のトラフィックが存在しない場合に、TCP-Reno よりも 30%～70%、

スループットを改善することが出来ると言われている。しかし、TCP-Reno との親和性問題があり、TCP-Reno と競合すると、TCP-Vegas 自身の輻輳ウィンドウサイズを下げてしまい、TCP-Reno に帯域を奪われてしまう。

2.4 CUBIC [8]

CUBIC は BIC[9]を改良したものであり、BIC とよく似た輻輳制御を行うが、名前の通りパケットロスからの経過時間の 3 乗の関数を用いることで、BIC で必要だったモードの切り替えが不要となった TCP であり、現在の Linux で実装されている。CUBIC では輻輳ウィンドウサイズをパケットロスからの経過時間を利用して決定することから RTT が異なるフローが複数存在している時も、RTT に依存せずに輻輳ウィンドウを決定できるため RTT 公平性が高いとされている。

CUBIC は ACK 受信時に以下の式(2.6)に従いウィンドウサイズを設定する。

$$W(t) = C(t-K)^3 + W_{last\_\max} \qquad K = \sqrt[3]{\frac{W_{last\_\max}\beta}{C}} \tag{2.6}$$

C は CUBIC の定数(デフォルトで 0.4)、β はパケットロス時の減少幅(デフォルトで 0.2)、W_{last_max} は前回のパケットロス時の輻輳ウィンドウサイズ、t は前回のパケットロスからの経過時間である。

パケットロス時には減少幅 β を用いて以下の式(2.7)のように設定する。

$$cwnd = cwnd * (1-\beta) \tag{2.7}$$

またパケットロスを検出したときの輻輳ウィンドウサイズが前回ロスを検出した時の値より小さいときには輻輳が起こっていると判断し、以下の式(2.8)ように W_{last_max} を変更する Fast Convergence モードを持つ。これにより 3 次関数の水平な部分の値が 10%ほど減少し、その結果余剰帯域が生まれることになりフローが競合するときに輻輳ウィンドウサイズが収束していく。

$$W_{last\_\max} = cwnd * (2-\beta)/2 \tag{2.8}$$

図 2.6 に CUBIC の輻輳ウィンドウの振る舞いを示す。輻輳ウィンドウサイズが 2000Kbyte に達するとバッファあふれによりパケット廃棄が発生して 20％減少し 1600Kbyte になる。この減少前の値が W_{last_max} になり、次に増加するときの 3 次関数の水平部分となる。またロスが起こったときの 2 回に 1 回は 3 次関数の水平部分が 10%減少して 1800Kbyte になっている。これは Fast Convergence モードのためであり、元の 2000Kbyte まで回復するのに約 2 倍の時間がかかっていることが分かる。

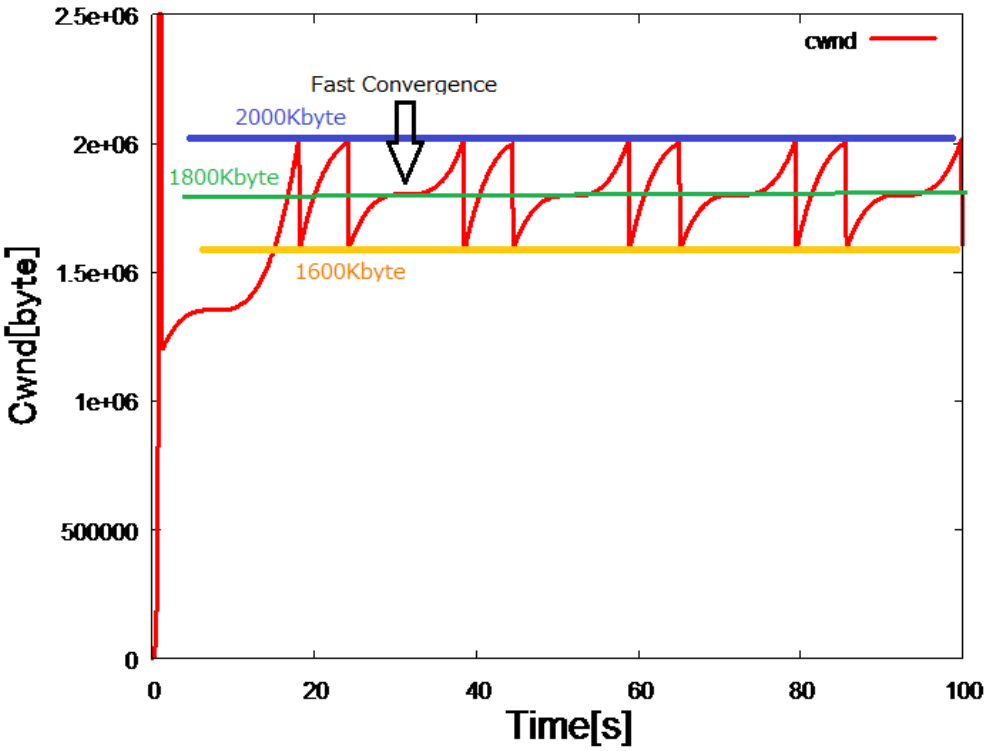


図 2.6　CUBIC の輻輳ウィンドウの変化(出展[10])

CUBIC は輻輳ウィンドウサイズの制御が RTT に依存しない方式となっているため、RTT が小さいネットワークなどでは TCP Reno に比べ、輻輳ウィンドウの上げ幅が小さくなり TCP Reno と公平に帯域を分け合うことができなくなってしまう。そのため CUBIC では TCP Reno の輻輳ウィンドウサイズをモデル化した式で計算し $W_{tcp(t)}$として保持する。そして先ほどの $W_{(t)}$と比較をして $W_{tcp(t)}$と比べ小さい場合は、TCP モードとなり $W_{tcp(t)}$を cwnd とする。$W_{tcp(t)}$の計算式を以下の式(2.9)に示す。

$$W_{tcp(t)} = W_{\max}(1-\beta) + 3\frac{\beta}{2-\beta}\frac{t}{RTT} \qquad (2.9)$$

2.5　Compound TCP(CTCP)[11]

CTCP は Microsoft Research によって開発された hybrid 手法[12,13]の TCP であり、Windows Vista 以降の Windows に実装されている。

まず、CTCP では TCP Reno と同様の制御を行う Loss-based の輻輳ウィンドウ(*cwnd*)と TCP Vegas と同様にバッファ内パケット数の推定値(*diff*)によって制御される Delay-based の輻輳ウィンドウ(*dwnd*)の２つの変数を用いている。CTCP の輻輳ウィンドウサイズ(*win*)は以下の式(2.10)で決まる。

$$win = \min(cwnd + dwnd, awnd) \tag{2.10}$$

awnd は受信側の広告ウィンドウサイズである。

そして RTT 毎の輻輳ウィンドウサイズの更新は以下の式(2.11)ように行われる。

$$win(t+1) = win(t) + \alpha \cdot wind(t)^k \tag{2.11}$$

パケットロス時には輻輳ウィンドウサイズは以下の式(2.12)ように更新される。

$$win(t+1) = win(t) \cdot \beta \tag{2.12}$$

α、β、k は定数であり、デフォルトでは α=1/8、β=1/2、k=0.75 となっている。高速性と親和性を実現するために重要となる *dwnd* は以下の式(2.13)ように更新される。

$$dwnd(t+1) = \begin{cases} dwnd(t) + (\alpha \cdot win(t)^k - 1), \text{if } diff < \gamma \\ (dwnd(t) - \xi \cdot diff), \text{if } diff \geq \gamma \\ (win(t) \cdot (1-\beta) - cwnd/2), \text{if loss is detected} \end{cases} \tag{2.13}$$

$$diff = \left(\frac{win}{baseRTT} - \frac{win}{RTT}\right) \cdot baseRTT$$

デフォルトでξ=1、γ=30 となっている。

図 2.7 に CTCP の輻輳ウィンドウの振る舞いを示す。

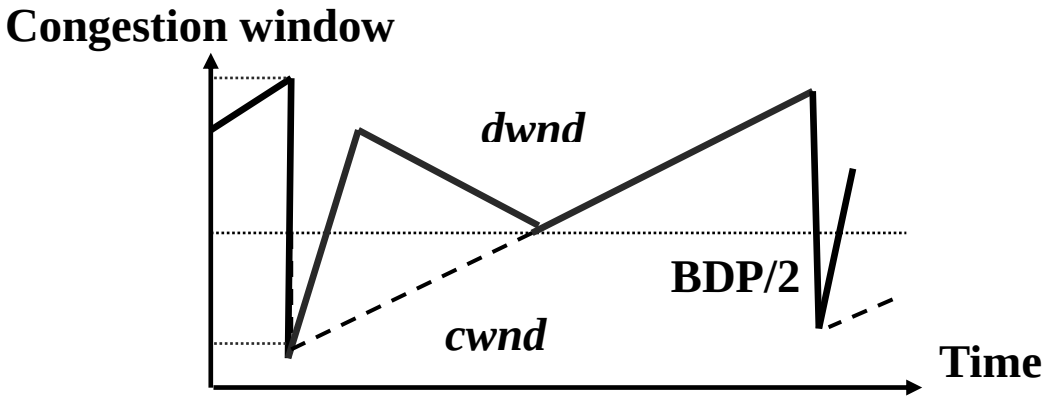


図 2.7　CTCP の輻輳ウィンドウの振る舞い(出展[2])

第3章 無線 LAN のアクセス制御方式

3.1 IEEE802.11 無線 LAN のアクセス制御方式[14]

IEEE802.11 無線 LAN では同一の無線チャネルを複数の端末で共有するためのアクセス制御機能が実装されている。アクセス制御方式には、各局が自律的に送信タイミングを決定する DCF(Distributed Coordination Function)と、オプションとして基地局がすべての送信を集中的に制御する PCF(Point Coodination Function)がある。本論文では、主に DCF について述べる。

3.1.1 IEEE802.11DCF(Distributed Coordination Function)[14,15]

IEEE802.11DCF は自律分散制御であり、各端末ではフレームの衝突をできるだけ回避するために、無線チャネルの使用状況を確認してからフレームを送信するかどうかを決定するアクセス方式として CSMA/CA 方式を使用している。CSMA/CA では、フレームを送信する前に無線チャネルが使用中であるかどうかを確認するため、キャリアセンスを行う。無線端末はビジー(使用中)からアイドル(未使用)への移行を契機にフレーム間隔(IFS:Inter Frame Space)の時間だけ待ち、引き続きバックオフと呼ばれるランダム時間分キャリアセンスを行い、継続してアイドルであることを確認すると送信権を得て、宛先へフレームを送信する。このとき、無線端末同士が同時に送信を行った場合はフレーム衝突が起きる。DCF では IFS として、チャネルの連続未使用期間である DIFS(DCF IFS)と ACK フレームなどを送信する際に用いる SIFS(Short IFS)を使用している。また、フレームの衝突を回避するためのバックオフ制御パラメータとして CW(Contention Window)が規定されている。バックオフ時間とは、0～CW の範囲でランダムに選ばれた値にスロット・タイムを乗算した時間である。CW は最小値が CWmin、最大値が CWmax の値の範囲であり、フレーム衝突などによる再送ごとに、CW=(CWmin+1)× 2^n －1(n は再送回数)の指数関数で CW の範囲は増加する。そして、CWmax に達した場合、フレームは破棄される。DCF のアクセス制御手順を以下の図 3.1 に示す。

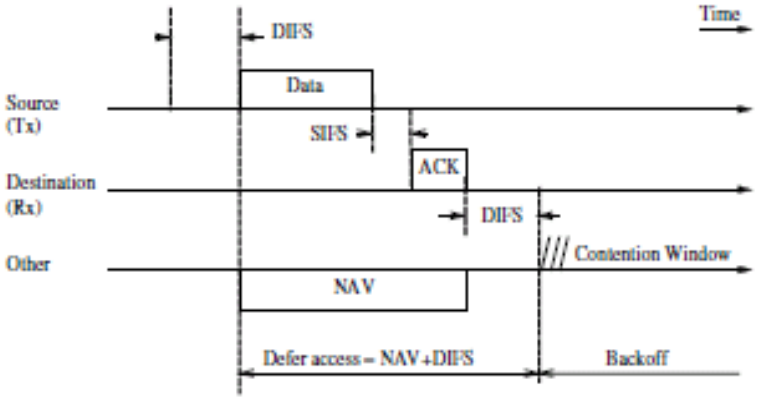


図 3.1　DCF のアクセス制御(出典：参考文献[15])

3.1.2　RTS/CTS[14,15]

無線通信では、無線端末間の距離や電波を通さない障害物などの影響により、お互いの無線通信が到達しない状態、すなわちキャリアセンスが機能しない環境になってしまう隠れ端末問題がある。

そこで、802.11 規格ではキャリアセンスが有効に機能しない環境に対応するために、送信要求を知らせる RTS(Request to Send)、受信準備が完了したことを知らせる CTS(Clear to Send)と呼ばれる機能がある。

隠れ端末が存在する場合、最初に送信権を得た端末(Source)は DIFS に加えてバックオフ時間をキャリアセンスし、データフレーム送信前に RTS を宛先(Destination)に送信する。このとき、キャリアセンスできる環境下にある Destination 以外の端末が RTS を受信した場合は、RTS フレームに記載されている期間(NAV：Network Allocation Vector)だけ送信を禁止することによって衝突を回避する。一方、Destination は RTS を受信した後、SIFS 時間待ってから Source に CTS を返信する。このときも RTS の場合と同様に、キャリアセンスできる環境下にある Source 以外の端末が CTS を受信した場合は、CTS フレームに記載されている期間(NAV)だけ送信を禁止し、衝突を回避する。そして、CTS を受信した Source は SIFS 時間待ってデータフレームを送信する。ここで、SIFS<DIFS であるため、送信前に DIFS ではなく SIFS を用いて優先権を持たせることによって、一度 RTS が正常に受信されると、その手順中のフレームは妨害されることなく送ることができる。

RTS/CTS を用いることによって、送信端末の信号をキャリアセンスできない環境下の端末が存在しても、宛先が送信する CTS を受信することによって送信端末の存在を知ることができる。以下の図 3.2 に RTS/CTS を用いた DCF のアクセス制御手順を示す。

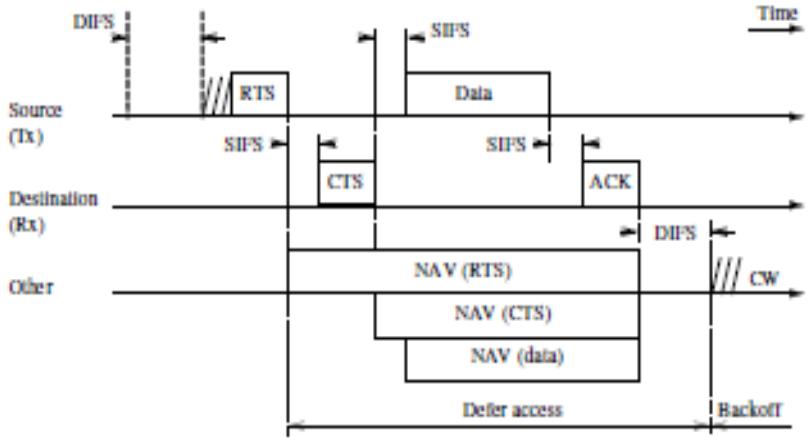


図 3.2　RTS/CTS を用いた DCF のアクセス制御(出典：参考文献[15])

3.2　IEEE802.11e のアクセス制御方式[14,15,16,17]

IEEE802.11e は音声やビデオなどの低遅延時間を要求するトラフィックのために、MAC レイヤに QoS 機能を追加した規格である。802.11e では、QoS をサポートするためのアクセス制御方式として、DCF と PCF の機能を統合的に提供する HCF(Hybrid Coordination Function)が規定されている。HCF には、EDCA(Enhanced Distributed Channel Access)または HCF contention based channel access と呼ばれる DCF を拡張し、データ送信時に優先制御を行う優先制御型と、HCCA(HCF controlled channel access)と呼ばれる PCF を拡張し、帯域幅や遅延時間などのパラメータ保証型のアクセス制御方式がある。本論文では主に EDCA について述べる。

3.2.1　IEEE802.11eEDCA(Enhanced Distributed Channel Access)

IEEE802.11eEDCA は、送信するフレームを 4 種類のアクセスカテゴリー(AC)に分類し、AC ごとに提供するサービスの品質に差をつけることによって、優先制御を提供している。AC には、音声用 AC_VI、ビデオ用 AC_VO、ベストエフォート用 AC_BE、背景トラフィック用 AC_BK が規定されている。

EDCA は、IEEE802.11DCF で規定されている CSMA/CA 方式を採用しているため、基本的な制御は DCF と変わらないが、EDCA では DIFS の代わりに AIFS(Arbitration IFS)を使用している。802.11e では、AIFS[i]=SIFS+AIFSN[i]×Slot Time と規定されている。また、チャネルへのアクセス権を取得した後、排他的にチャネルの使用が認められている時間を表す TXOP(Transmission Opportunity)というパラメータも使用されている。EDCA

では AIFS、CW、TXOP のパラメータを AC ごとに設定することによって優先制御を行っている。すなわち、802.11eEDCA ではこれらのパラメータを任意に設定できる。EDCA アクセス制御、EDCA フレーム送信手順、EDCA デフォルトパラメータ値をそれぞれ以下の図 3.2、3.3、表 3.1 に示す。

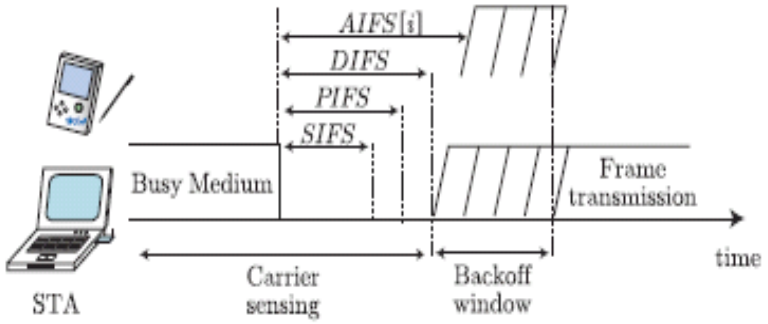


図 3.2　EDCA のアクセス制御(出典：参考文献[17])

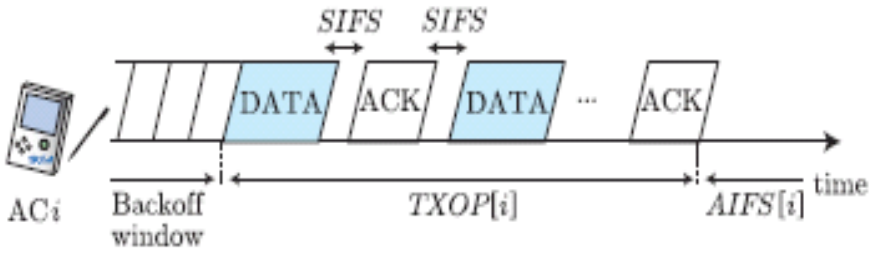


図 3.3　EDCA のフレーム送信手順(出典：参考文献[17])

パラメタ	アクセスカテゴリ (AC_i) (トラヒックの種類)			
	$i=0$ (VoIP)	$i=1$ (Video)	$i=2$ (BE)	$i=3$ (BK)
$AIFSN[i]$	2	2	3	7
$CW_{\min}[i]$	7	15	31	31
$CW_{\max}[i]$	15	31	1023	1023
$TXOP[i]$	3.264	6.016	0	0

表 3.1　EDCA デフォルトパラメータ値(出典：参考文献[17])

第4章 TCP バージョン推定・識別の従来研究

4.1 TBIT(TCP Behavior Inference Tool)を用いた推定・識別[17,18,19]

本章では、TCP バージョン推定・識別の従来研究について説明する。最初に、TBIT(TCP Behavior Inference Tool)というツールを用いた推定・識別方法について説明する。

TBIT はウェブサーバが使用する TCP の振る舞いを明らかにするためのツールである。TBIT はクライアント側で利用され、パケットを発生させることや意図的にパケットロスを引き起こすことができる。[18]では、まず、TBIT によってパケットロスを引き起こす。第 2 章で説明した初期に提案された TCP である Tahoe、Reno、NewReno はパケットロス後の Fast Retransmission に違いがあるため、Fast Retransmission の振る舞いを TBIT で観測することによって、TCP バージョンを識別する手法である。図 4.1 より、パケットロス後の振る舞いに違いがあることが見て取れる。

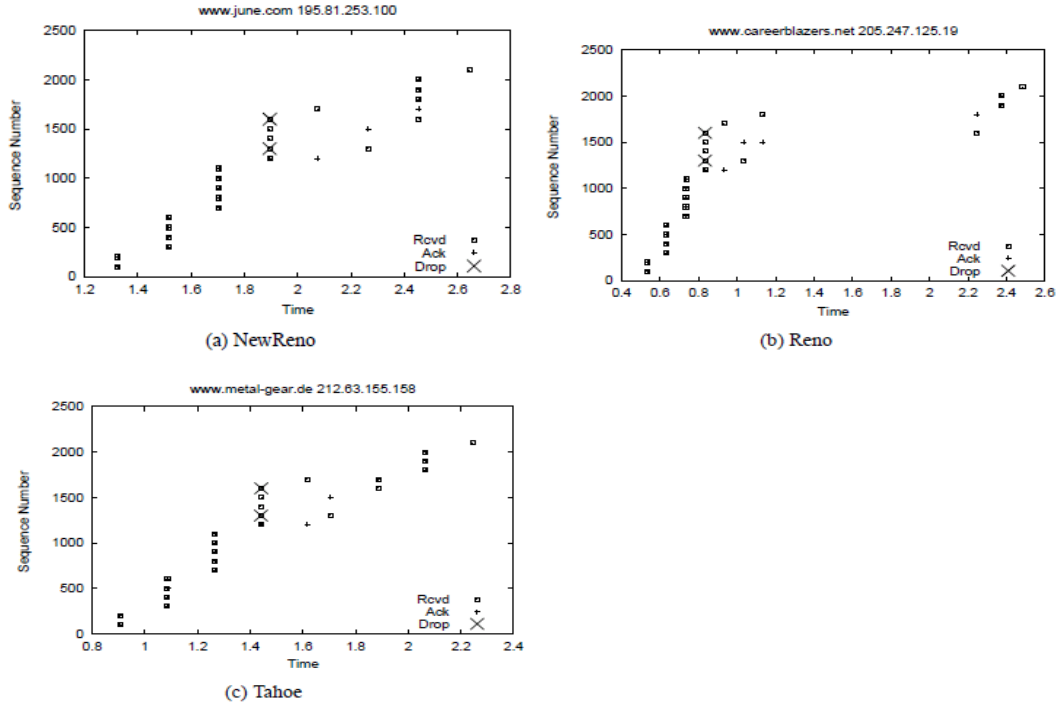


図 4.1 Example of congestion control behavior(出展：参考文献[18])

[19]では、TBIT を用いて、NewReno 以降の TCP である BIC や CUBIC などの識別を行っている。上記に述べたように[18]では Fast Retransmission を基に TCP バージョンの推定を行うため、Fast Retransmission のアルゴリズムが NewReno と同じである BIC や CUBIC などの TCP の識別ができない。そこで[19]では、TBIT を用いた cwnd サイズの推定から TCP バージョンを識別している。識別方法はまず、ハンドシェイク後、クライアン

トは受信したすべてのパケットを格納し、TBIT を用いて受信したパケットの ACK を返さない。次にクライアントは決められた時間待機した後、すべてのパケットの ACK を返す。この決められた時間とは cwnd サイズ分のパケットを受信するまでの時間が必要である([19]では 1 秒)。そして、ACK を返すまでに受信したパケット数をカウントすることで cwnd サイズを推定する(図 4.2)。このようにして推定した cwnd の振る舞いから TCP バージョンの識別を行う。なお、スロースタートフェーズから輻輳回避フェーズに強制的に移行させるため、63 番目のパケットを意図的にロスさせている。

[19]では、Ubuntu8.4 を入れた 2 台の PC を用いて、500KBs の Web ページを送信する実験を行い、各 TCP バージョンの cwnd を推定している。Reno、BIC、CUBIC と識別できたときの cwnd 推定結果を図 4.3~4.5 に示す。これらより、CUBIC は Reno の 3 倍、BIC の 2 倍速いという結果が得られている。

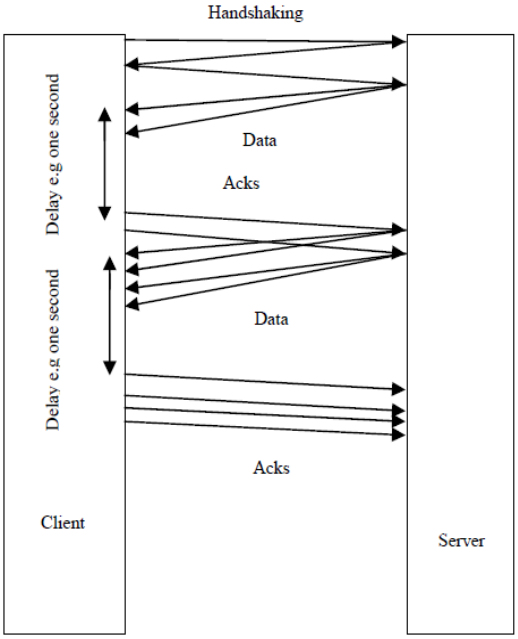


図 4.2　Detecting CWND size(出展：参考文献[19])

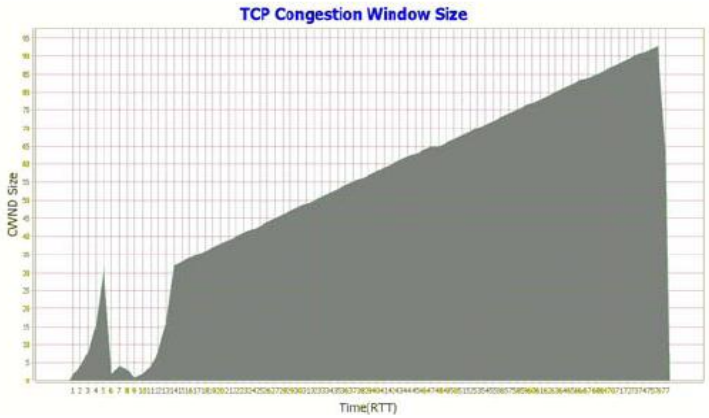


図 4.3　Reno CWND size(出展：参考文献[19])

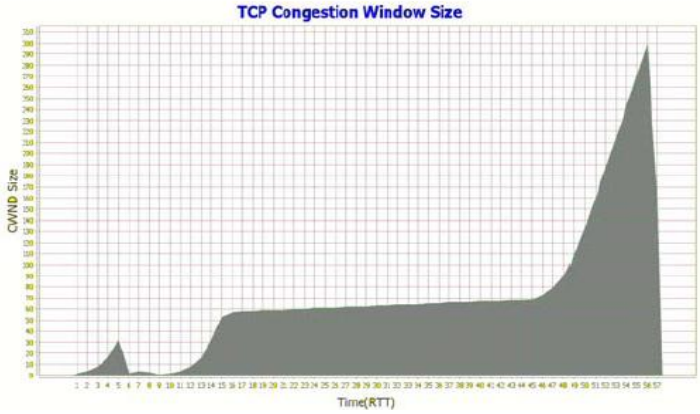


図 4.4　BIC CWND size(出展：参考文献[19])

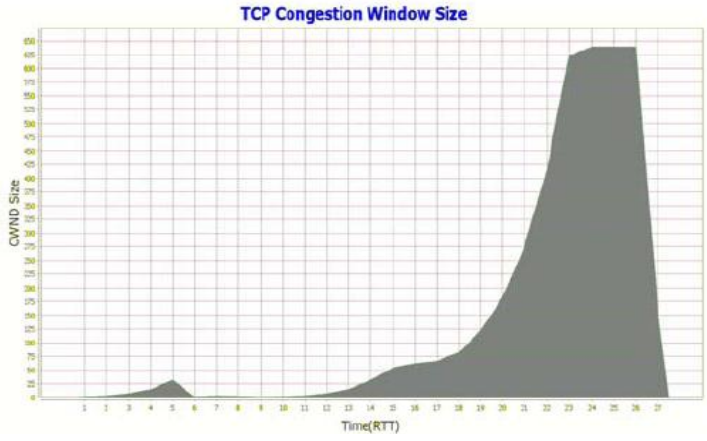


図 4.5　Cubic CWND size(出展：参考文献[19])

[20]では、TBIT の発展版として、TCP バージョンの識別を行うツールである CAAI(TCP Congestion Avoidance Algorithm Identification)を提案している。識別方法は、受信側から送信端末の輻輳ウィンドウの変化をエミュレーションした同一条件下で推定し、その変化の特徴から輻輳制御方式の判別を行う。TCP バージョンには cwnd の制御に RTT 毎に特定の値を加える AIMD や RTT からの経過時間を利用する CUBIC、RTT の増減を用いる Delay-based 手法などがある。

まず各サーバで推定する条件を統一して比較するために、図 4.6 のように ACK を返すまでの時間を調整することで RTT を一定の値（[20]では 1 秒）にエミュレーションして統一する(Network Environment A とする)。また設定したタイムアウトパケット数（[20]では 512pkt）を超えると TBIT によって ACK を返さずにパケットロスを擬似的に発生させる。

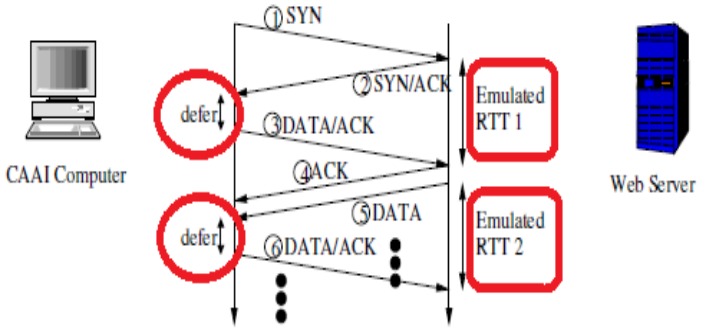


図 4.6　パケットの振る舞い(出展：参考文献[20])

また RTT が変化する環境(Network Environment B とする)（図 4.7 の破線）をエミュレーションして同様に測定を行う。この場合の RTT の変化は以下のようになる。RTT が途中で増加するため Delay-based 手法を検出できる。

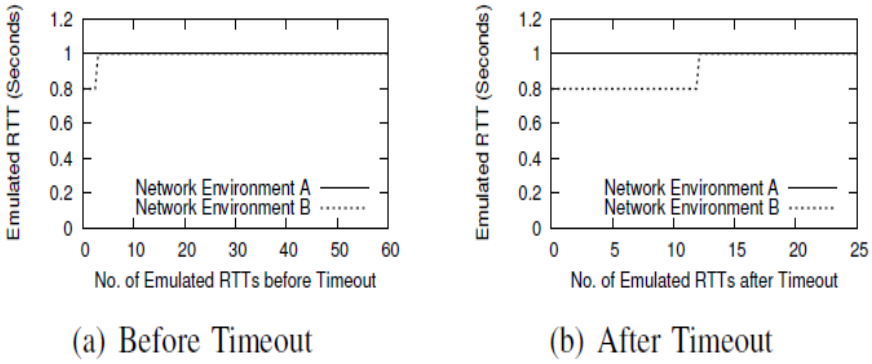
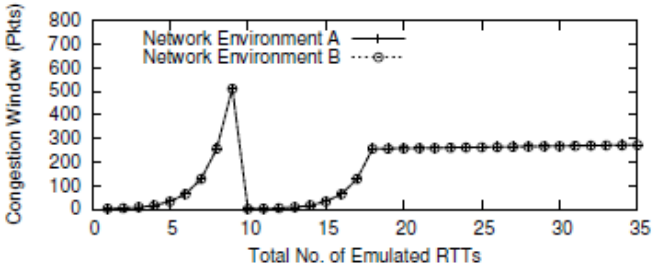
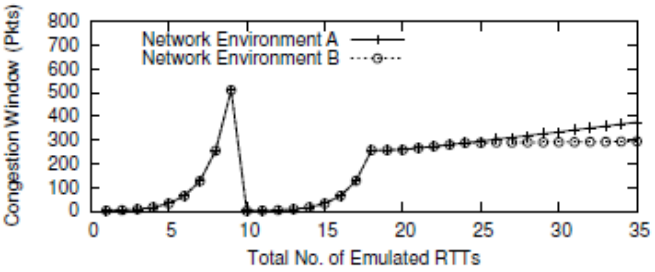


図 4.7　エミュレーションする RTT の変化(出展：参考文献[20])

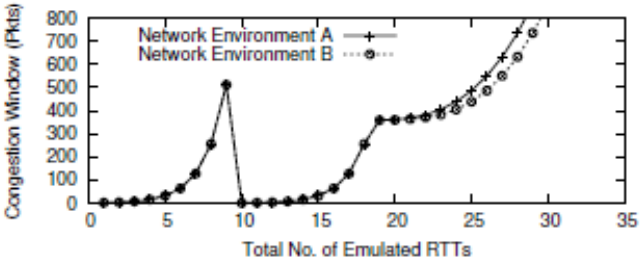
上記の環境下での RTT 毎の輻輳ウィンドウの変化を計測したものを図 4.8 に示す。いずれの方式もスロースタートに違いはなく、1RTT ごとに輻輳ウィンドウが 2 倍になり 9RTT で 2^9＝512pkt となる。ここで擬似的にタイムアウトを発生させその後の輻輳ウィンドウの挙動により判別を行う。TCP Reno の識別では 10RTT からのスロースタートフェーズでは 8RTT で 2^8＝256pkt に達し、輻輳回避フェーズに入りその後は 1RTT で 1pkt ずつ増加する。CTCP の識別では、RTT が一定の Network Environment A では TCP Reno と区別しにくいが、RTT が増加する Network Environment B では挙動が異なることを利用する。CUBIC の識別では、スロースタート閾値の設定がロス時の輻輳ウィンドウサイズの 0.7 であるため他の方式比べ輻輳回避フェーズに高い値で入る。その後三次関数で増加する。TCP Vegas の識別では、RTT が一定の Network Environment A では TCP Reno との違いが識別しにくいが、RTT が増加する Network Environment B では輻輳ウィンドウが増加しないため識別が可能となる。



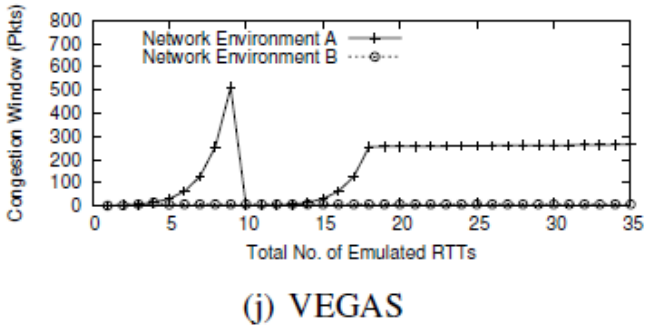
(a) AIMD



(c) CTCP (Windows Server 2008)



(e) CUBIC (Linux kernel 2.6.27)



(j) VEGAS

図 4.8　輻輳ウィンドウの RTT ごとの変化(出展：参考文献[20])

4.2　Linux で扱える 14 種類の TCP バージョンの識別[21]

[21]では Linux で扱える 14 種類の TCP の 2 種類のバージョンが競合している場合の識別を行っている。識別方法はまず、中継ルータで RTT を推定する。推定方法は flight method[21,22]と呼ばれる手法を使用している。flight method はパケットの到着間隔を利用している。TCP の輻輳制御では cwnd の大きさ分のパケットを送信後、送信側は送信データの ACK が到着するまで、次のパケットを送信しない。そのため、送信データの ACK が到着し、次のパケットを送信する時にパケット間隔は通常より大きなものとなる。この到着間隔が変化したときを cwnd の切れ目として判断し、RTT を測定する(図 4.9)。

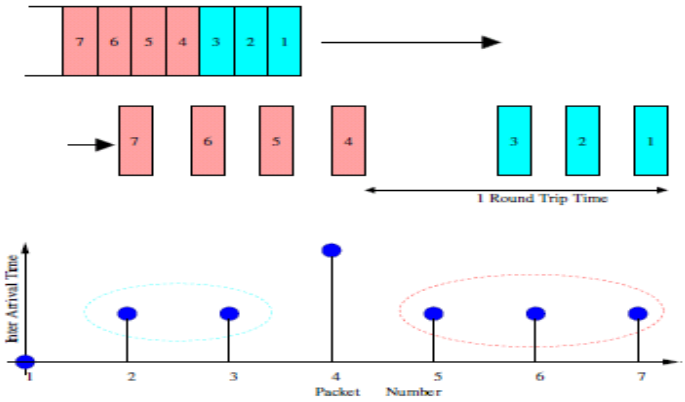


図 4.9　Flight based (packet) view of TCP (出展：参考文献[22])

次に cwnd を推定する。cwnd の推定は cwnd が 1 RTT 中に送信されるパケット数を示すという性質を利用する。中継ルータにおいて、flight method によって推定した RTT 間に到着したパケット数を cwnd として推定する。そして、推定した cwnd から SBS(Step by Step)、R(Rate)、N(No Change)、RC(Rate Change)の 4 つの特性値を求める。各特性値について以下に述べる。

・ SBS(Step by Step)

$$S = \frac{I_S}{I_S + I_M + I_N}\,[\%] \qquad (1)$$

I_S ：cwnd の増加幅が 1 である回数

I_M ：cwnd が 1 より大きい回数

I_N ：cwnd の増加幅が 0 である回数

SBS は主に cwnd の増加幅が大きい TCP と小さい TCP の識別に用いる。図 4.10 に SBS を用いた識別例を示す。

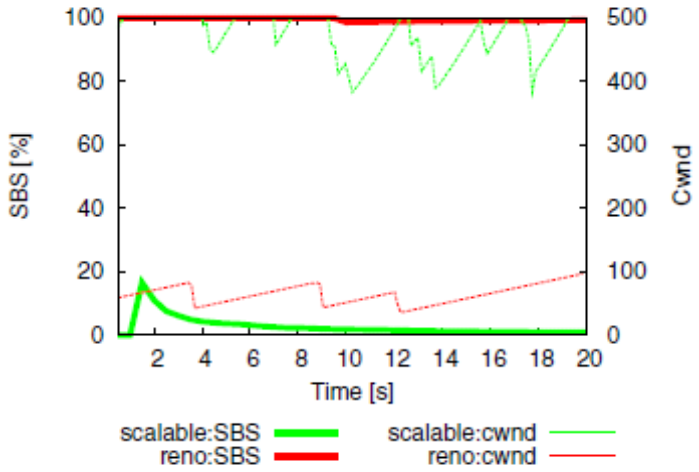


図 4.10　SBS による Reno と Scalable TCP の識別(出展：参考文献[21])

・ R(Rete)

$$R = \frac{r_T}{R_M}\,[\%] \qquad (2)$$

r_T ：cwnd の増加率 r_i が 1 以上であるときの r_i の累積値

R_M ：cwnd の増加率 r_i が 1 以上である回数

R は主に cwnd の増加率が異なる TCP 同士の識別に用いる。図 4.11 に R を用いた識別例を示す。

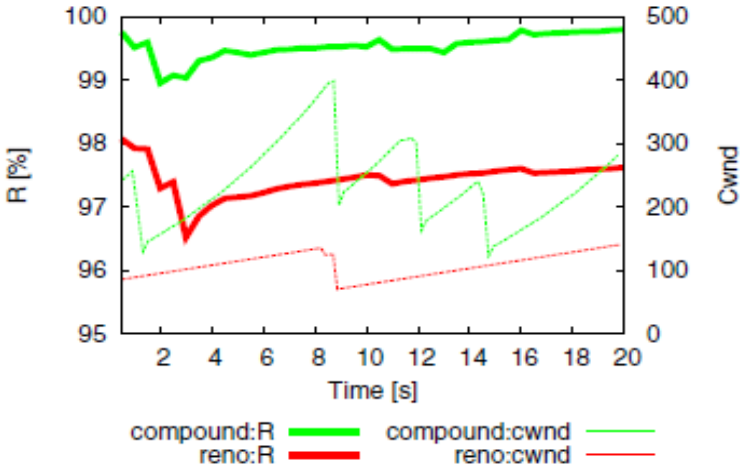


図 4.11　R による Reno と Compound TCP の識別(出展：参考文献[21])

・ N(No change)

$$N = \frac{I_N}{I_S + I_M + I_N}[\%] \qquad (3)$$

N は主にある期間 cwnd を一定に保つ振る舞いをする TCP との識別に用いる。図 4.12 に N を用いた識別例を示す。

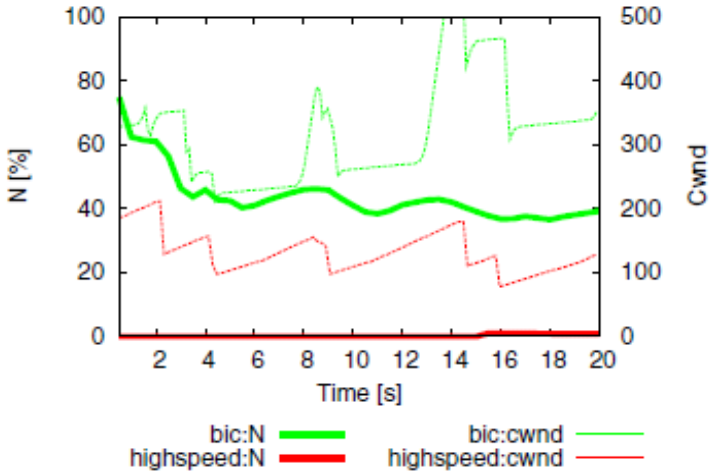


図 4.12　N による Highspeed TCP と BIC の識別(出展：参考文献[21])

・ RC(Rate Change)

$$RC = \frac{D}{C}[\%] \qquad (4)$$

D：i-1 番目の cwnd と i 番目の cwnd の値が異なる回数

C：cwnd を推定した回数

RC は主にコンスタントに cwnd を増加させる TCP と増加幅が頻繁に変わる TCP との識別に用いる。図 4.13 に RC を用いた識別例を示す。

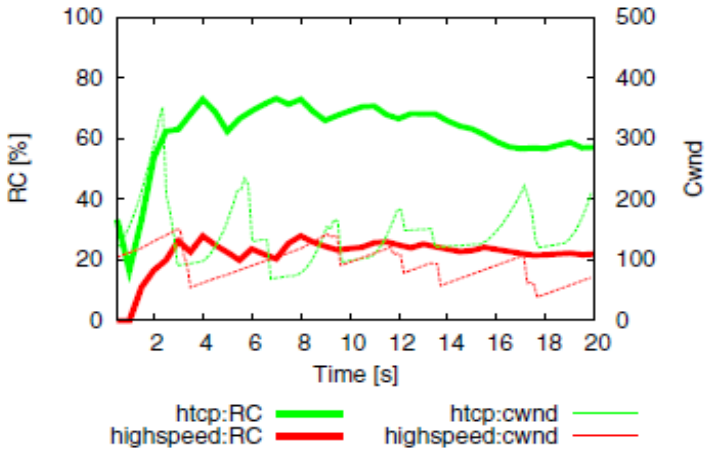


図 4.13　RC による Highspeed TCP と HTCP の識別(出展：参考文献[21])

これらの特性値は各 TCP バージョンで異なる値を得ることができる。求めた特性値を利用し、異なる TCP バージョン 2 種類が競合している環境において、求めた各特性値の比較や複数の特性値を組み合わせによって TCP バージョンを識別する。

第5章 提案手法

5.1 提案手法の概要

本章では､提案手法である TCP 差別化手法の概要について説明する｡提案手法ではまず､ルータ(AP)で IP アドレスとポート番号からフローを判別し、フロー毎に TCP バージョンの識別を行う。TCP バージョン識別には RTT 推定、cwnd 推定を利用する。TCP バージョン識別後はロスベース(TCP Reno,CUBIC,Compound)と遅延ベース(TCP Vegas)の TCP パケットを別々のバッファに格納する。その後、別々に CSMA/CA による無線アクセス制御を行うことになるため、EDCA のパラメータを変更することで、遅延ベースの TCP パケットを優先して送り、TCP の差別化を行う。

5.2 RTT 推定 1

TCP バージョン識別に用いる RTT 推定について説明する。RTT 推定には TCP のシーケンスナンバーとタイムスタンプオプションを利用する。図 5.1 のように、RTT = Wireless Delay + Wired Delay として計算する。Wireless Delay は、図 5.1 の a 地点でのデータパケットのシーケンスナンバーと b 地点での ACK パケットのシーケンスナンバーが等しい場合の時間の差分を取る。Wired Delay は、c 地点でタイムスタンプオプションによって送信時間を取得し、d 地点をデータパケットが通過した時間の差分を取る。ここでは、無線 LAN が CSMA/CA によって遅延変動が大きいのに対し、有線はほとんど遅延変動がないと考えられるため、sender-AP 間の遅延の 2 倍を Wired Delay とする。

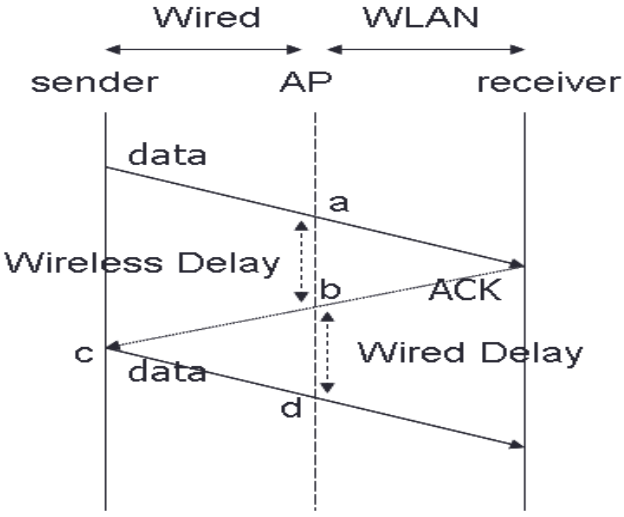


図 5.1 RTT 推定方法

5.3　RTT 推定 2

もう 1 つの RTT 推定について説明する。今度は TCP の輻輳制御を利用する。有線環境のみの場合、ネットワークが輻輳している状態であると、ボトルネックとなる帯域によってパケットの送信間隔が決まる。図 5.2 のネットワークトポロジの場合では、図 5.3 のようなパケット到着間隔となる。図に示すように、TCP は cwnd を 1 増加させる際、1 ラウンド中の最後の ACK に対してデータパケットを短い間隔で 2 つ連続送る(図 5.3 上)。cwnd を 1 減少させる際は、1 ラウンド中の最後の ACK に対してデータを送信しない。そのため、パケット 2 つ分の到着間隔となる(図 5.3 下)。このパケット送信間隔が変動した部分を 1RTT の切れ目として RTT を測定する。しかし、本論文での実験環境は有線無線混在環境を想定しているため、cwnd が 1 増加する場合の特性(図 5.3 上)は見られるが、無線の CSMA/CA によるランダム性によって、cwnd が 1 減少する場合の特性(図 5.3 下)が見られない。

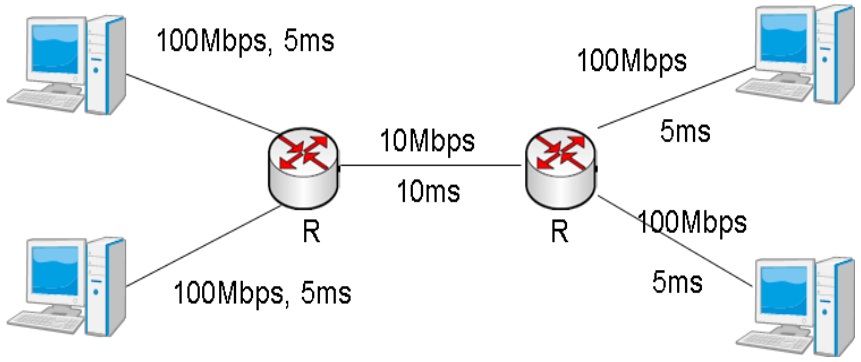


図 5.2　有線ネットワークトポロジ

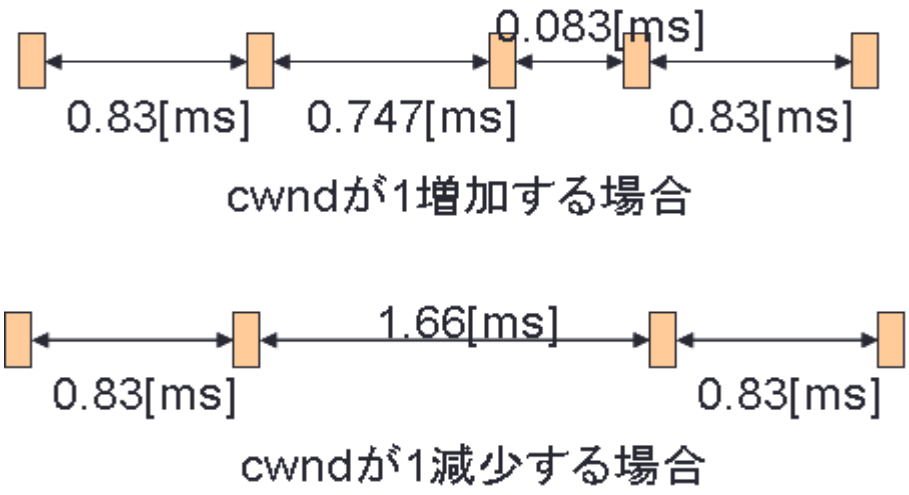


図 5.3　TCP のパケット到着間隔

5.4　Cwnd 推定

Cwnd 推定には前章の RTT 推定から行う。TCP の cwnd は 1RTT 中に送信されるパケット数を表すため、ルータ(AP)において、推定した RTT 間に到着したパケット数をカウントし、cwnd とする。例として、TCP Reno と TCP Vegas を競合した場合の RTT、cwnd の実測値と推定結果を図 5.4~5.6 に示す。図 5.4~5.6 より、RTT、cwnd 共に実測値とほぼ等しい値を推定できていることがわかる。なお、推定結果は CSMA/CA による遅延変動やパケットロスの影響で誤差が生じる。

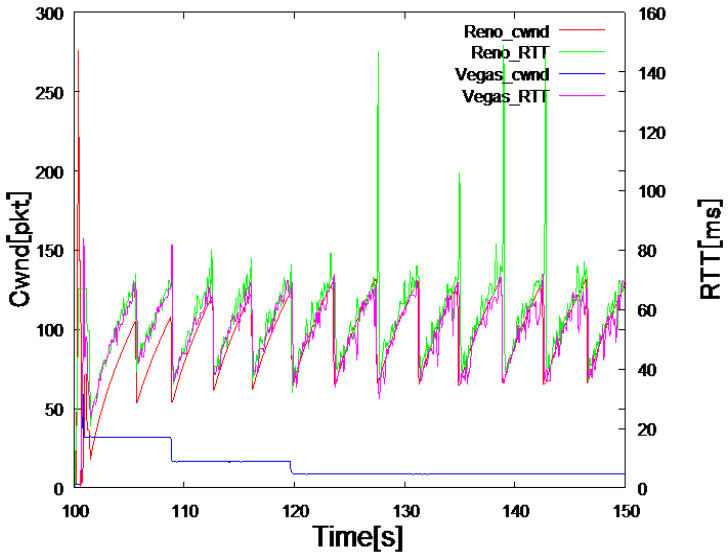


図 5.4　RTT、cwnd の実測値

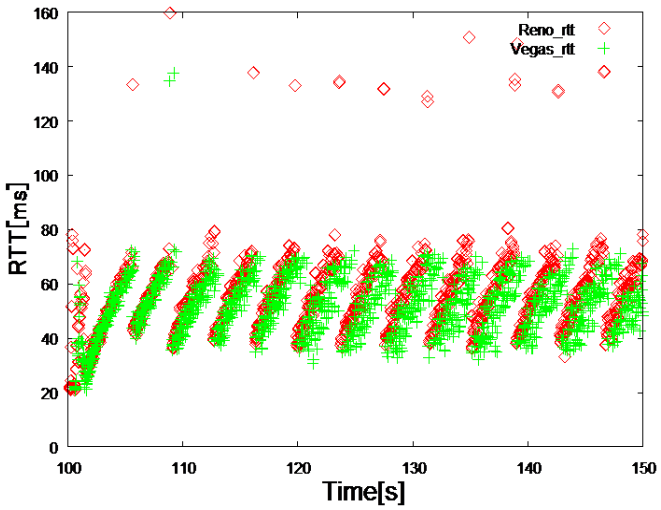


図 5.5　RTT 推定結果

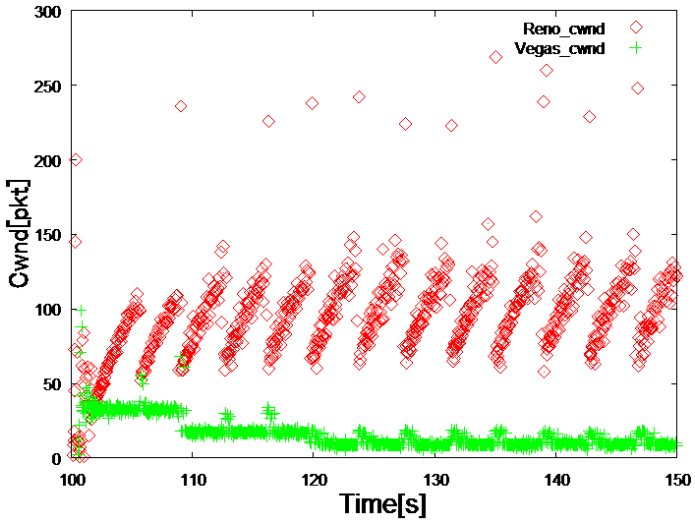


図 5.6　cwnd 推定結果

5.5　TCP バージョン識別

TCP バージョンの識別は、推定した cwnd による cwnd の振る舞いから判断する。第 2 章や 5.4 章の cwnd の結果で述べたように、TCP のロスベースと遅延ベースでは輻輳制御に大きな違いがある。理論的には輻輳回避フェーズにおいて、ロスベースの TCP は 1RTT に 1 度、cwnd を増加させていく。それに対し、遅延ベースである TCP Vegas は cwnd を一定に保つように振る舞う、あるいは、他 TCP と競合したときは cwnd を減少させていく。このように、cwnd の振る舞いに大きな違いがある。この違いを利用し、まず、各フローにおいて cwnd 推定を n(=20)回行う。そして、無線 LAN の遅延変動などによる推定値のばらつきを考慮し、cwnd の増分が m(=10)よりも小さい場合は TCP Vegas、m(=10)以上の場合は TCP Reno であると判断する。この識別方法はシンプルではあるが、推定値のばらつきにより、ロスベースと遅延ベースとを誤識別する可能性がある。あくまで優先制御であるため、100%の識別率が求められるわけではないが、より精度の高い方法を検討する必要があると思われる。

5.6　EDCA の新規利用による TCP 差別化

TCP バージョンの識別を行った後は、ロスベースと遅延ベースで別々のバッファを用意し、それぞれのバッファにパケットを格納していく。そして、第 3 章で説明した EDCA のパラメータを変更することで TCP 差別化を行う。ロスベースと遅延ベースの TCP をそれぞれ別のバッファに格納する理由は、遅延ベースがロスベースの TCP の影響を受け、RTT が増加するのを防ぐためである。また、別々のバッファに分けることで、各バッファで EDCA のパラメータを変更し、優先制御することが可能となる。

無線 LAN 環境での優先制御として利用している EDCA において、通常、EDCA はビデオや音声などといったアプリケーションの種類別に AC を用意し、優先制御を行っていく。しかし、本研究では、TCP Reno や TCP Vegas といった TCP のバージョン別に AC を用意し、優先制御を行うという EDCA の新規利用をしている。一方、有線環境で優先制御を行うことを考える場合は、一般的に広く知られているラウンドロビンや WFQ(Weighted Fair Queueing)などの方法を利用することが考えられるであろう。

本研究で用いた Ubuntu10.04 では、AC の分類に IP ヘッダの ToS を見て行っている。そのため、実機では TCP バージョンを識別後、TCP Vegas と識別したフローの ToS を変更することによって、別の AC にパケットを分類することができる。

第6章 シミュレーション・実機実験

6.1 シミュレーション・実機実験環境 1

本章では、TCP 差別化の効果を明らかにするために、シミュレーションと実機実験にて評価を行う。シミュレーションには ns-2[24]を用いる。実機実験では、AP として Ubuntu10.04 に MadWifi[25]を導入し、利用している。MadWifi は、Linux 上で動作する無線 LAN のドライバであり、子機として無線 LAN 通信を行うことや AP として利用することもできる。シミュレーション環境は極力実機実験に合わせた環境を設定した。

実験環境 1 は受信端末が 2 台以上存在する場合を想定する。ネットワークトポロジーを図 6.1 に、実験パラメータを表 6.1 に示す。無線リンクは、無線 LAN 規格として IEEE802.11g(54[Mbps])を使用し、RTS/CTS はオフとする。有線リンクの帯域は 100[Mbps]とする。図 6.1 のネットワークトポロジーにおいて、Wired Node からロスベース(Reno、Compound、CUBIC)と遅延ベース(Vegas)の TCP フローを端末 1 台につき 1 フローずつ Wireless Node に送信する。実機実験では、iperf を利用し、パケットを送信した。本環境では AP がボトルネックとなるため、AP での TCP バージョン識別を行う。なお、Compound TCP はハイブリッド手法の TCP であるが、本環境においてはロスベースと同様の振る舞いをすることを確認しているため、ロスベースとして扱う。また、TCP の Delayed ACK について、Linux ではデフォルトで Delayed ACK が使用されているため、シミュレーションにおいても Delayed ACK を適用する。

優先制御で使用する EDCA のパラメータを表 2 に示す。パラメータ 1 は提案手法を適用していない場合(差別化なし)、パラメータ 2 は提案手法を適用した場合(差別化あり)のロスベースと遅延ベースのパラメータである。

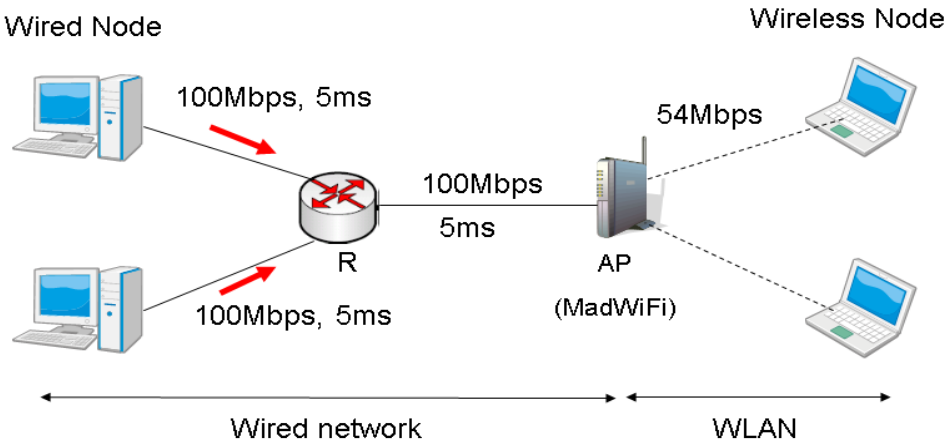


図 6.1 ネットワークトポロジー

表 6.1　実験パラメータ

無線リンク伝送レート		54[Mbps]
キューサイズ (シミュレーション)		100[pkt]
MAC		802.11g
ルーティング		DSDV
TCP	ロスベース	Reno Compound CUBIC
	遅延ベース	Vegas
パケットエラーレート (シミュレーション)		1[%]
TCPパケットサイズ		1500[Byte]

表 6.2　EDCA パラメータ

		CWmin	CWmax
差別化なし		15	1023
差別化あり	ロスベース	15	1023
	遅延ベース	7	15

6.2　シミュレーション・実機実験結果 1

6.2.1　TCP Reno と TCP Vegas を競合させた場合の TCP 差別化の効果

基本的な TCP 差別化の効果を明らかにするため、TCP Reno と TCP Vegas を各 1 フロー競合させた実験を行った。シミュレーション・実機実験のスループット、cwnd・RTT の結果をそれぞれ図 6.2~6.4、図 6.5~6.7 に示す。

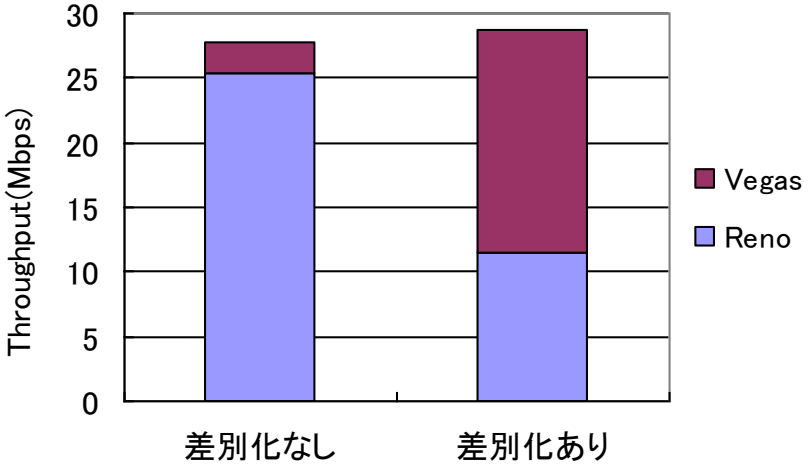


図 6.2　スループット結果(シミュレーション)

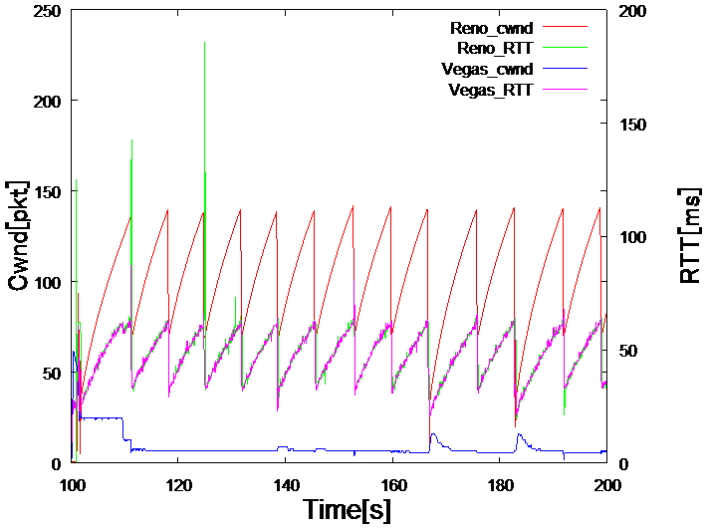


図 6.3　差別化なしの場合の cwnd、RTT 結果(シミュレーション)

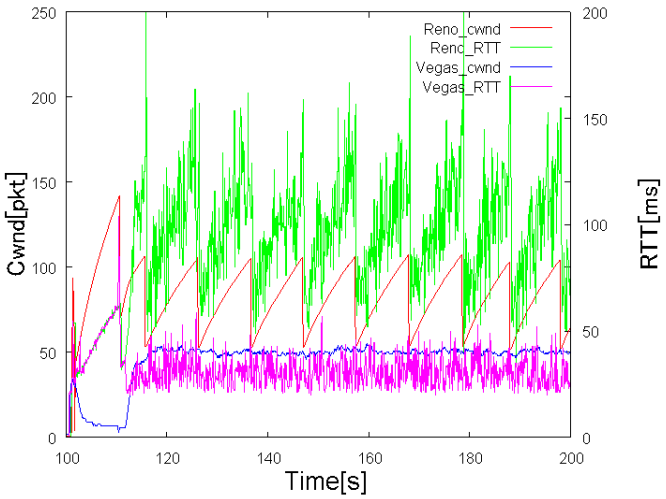


図 6.4　差別化ありの場合の cwnd、RTT 結果(シミュレーション)

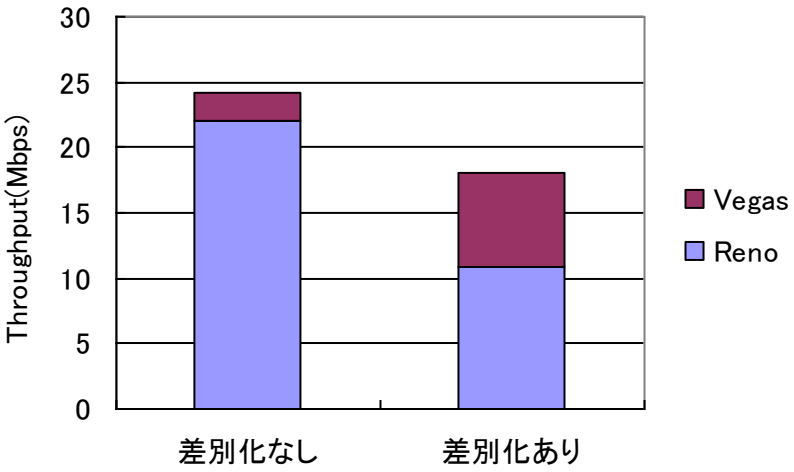


図 6.5　スループット結果(実機)

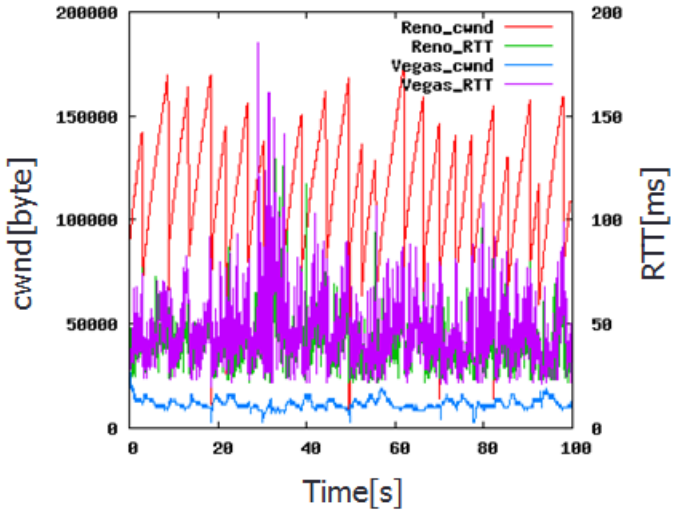


図 6.6　差別化なしの場合の cwnd、RTT 結果(実機)

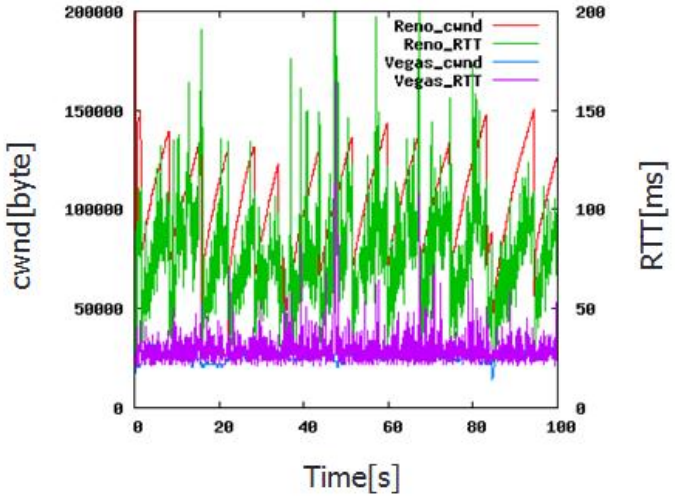


図 6.7　差別化ありの場合の cwnd、RTT 結果(実機)

図 6.2 より、シミュレーションでは、TCP 差別化をすることで、TCP Vegas は差別化なしの場合よりも約 8 倍のスループットを得ることができている。また、図 6.3 より、実機では、TCP 差別化をすることで、差別化なしの場合よりも約 4 倍のスループットを得ることができている。実機実験では、差別化なしと差別化ありでスループットに差が生じている。実機では、無線 LAN の電波強度やビットエラーレート、Linux 端末の性能など様々な要因によってスループットに大きな影響を与えることが原因と考えられる。

図 6.3,6.5 より、シミュレーションと実機ともに差別化なしの場合は、TCP Vegas の cwnd が非常に低いことがわかる。これは、AP で TCP Reno と TCP Vegas が同じバッファを共有していることにより、TCP Vegas が TCP Reno の影響を受け、バッファリング遅延が増大していることが原因である。図においても、TCP Reno と TCP Vegas の RTT の値がほぼ等しくなっている。一方、図 6.4,6.7 より、差別化ありの場合では、TCP Vegas が高い cwnd を得られている。これは、TCP Reno と TCP Vegas のバッファを別々に分けたことにより、TCP Vegas のバッファにパケットが溜まらず、有効的な振る舞いができているからである。TCP Vegas は同様の理由から RTT も小さく抑えられている。しかし、TCP 差別化によって、TCP Vegas の RTT が小さく抑えられている反面、低優先度の TCP Reno は RTT が増大している。本研究では、TCP Vegas の低遅延を映像配信に活かすことを目的としているため、この点については大きく問題視しない。

これらの結果より、TCP 差別化をすることで、TCP Vegas は非常に大きいスループットの改善効果が得られることが言える。

6.2.2 他 TCP バージョンを競合させた場合の TCP 差別化の効果

6.2.1 章では TCP Reno と TCP Vegas の競合について、評価を行ったが、本章では、TCP Vegas が TCP Reno 以外の TCP バージョン(Compound TCP、CUBIC を使用)と競合した場合の TCP 差別化の効果を明らかにする。6.2.1 章の結果からシミュレーションと実機で結果の傾向が変わらないことが予想されるため、シミュレーションでのみ評価を行う。図 6.8 に各 TCP バージョンと競合させたときのスループット結果を、図 6.9,6.10 に Compound TCP と CUBIC の cwnd、RTT 結果を示す。

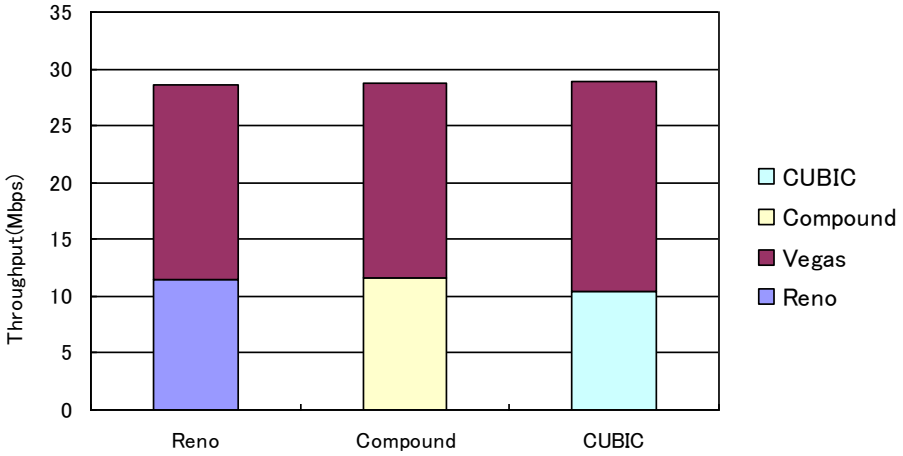


図 6.8　各 TCP バージョンと競合させた場合のスループット結果

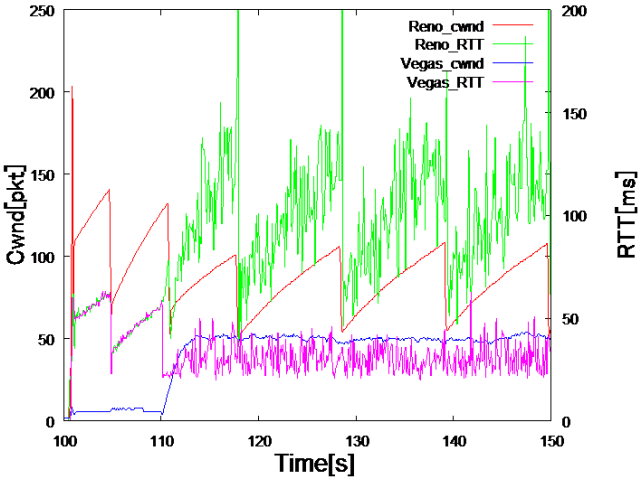


図 6.9　Compound TCP と競合させた場合の cwnd、RTT 結果

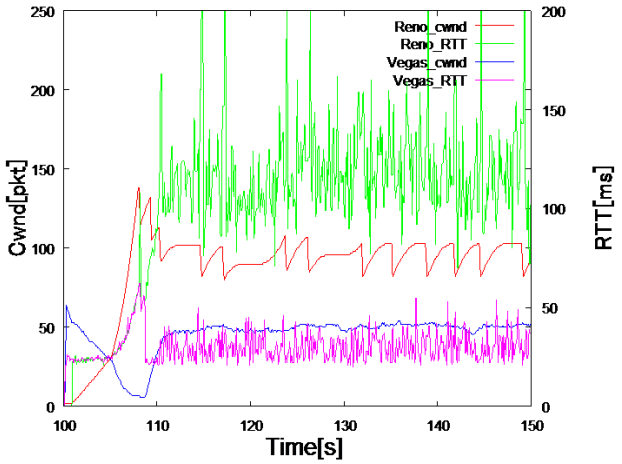


図 6.10　CUBIC と競合させた場合の cwnd、RTT 結果

図 6.8 より、Compound TCP や CUBIC と競合させた場合でも、TCP Vegas は TCP Reno と競合させた場合とほぼ同程度のスループットが得られている。これは、優先制御を MAC 層で行っており、MAC 層でパケットの送信機会が変動するため、スループットはトランスポート層である TCP のバージョンには依存しないことが言える。

図 6.9,6.10 より、cwnd、RTT 結果においても TCP Reno と競合させたときと同様の傾向が得られている。TCP 差別化をすることで、TCP Vegas は高い cwnd が得られており、RTT も小さく抑えられている。その反面、Compound TCP と CUBIC の遅延が大きくなっている。

6.2.3 EDCA パラメータの CW を変動させた場合の TCP 差別化の効果

表 6.2 で示した EDCA パラメータの CWmin を変動させ、どの程度 TCP 差別化の効果に違いが出てくるのかをシミュレーションにより評価する。TCP Vegas のパケットを格納するバッファの CWmin を 1~1023 まで変動させ、CWmax を 1023 とする。CWmin を変動させた場合のスループット結果を図 6.11 に示す。

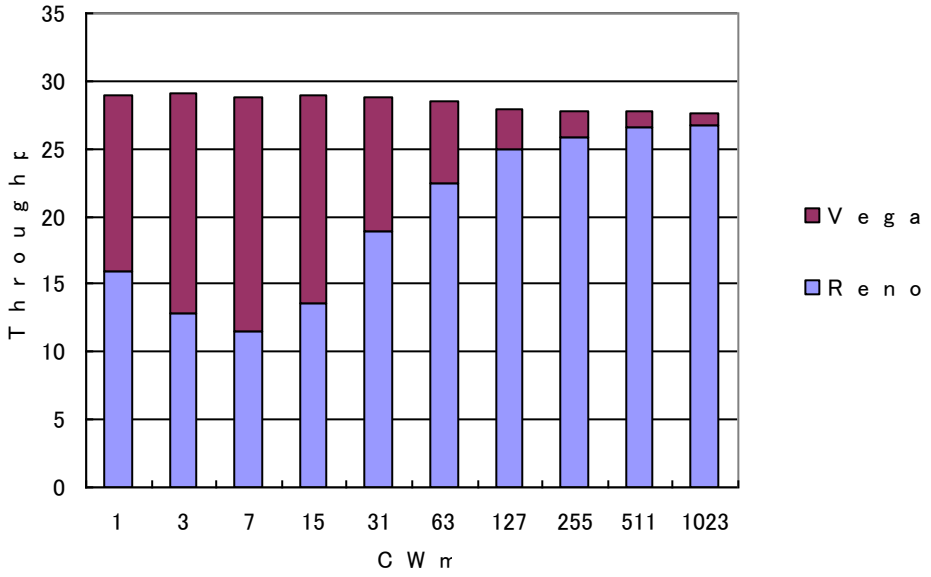


図 6.11　CWmin を変動させた場合のスループット結果

図 6.11 より、TCP Vegas のスループットは CWmin=7 の場合が一番高くなっている。CWmin が 7 以下の場合に TCP Vegas のスループットが下がっている理由として、CWmin が非常に小さい場合、TCP Vegas のパケットは AP に到着後すぐに送信することができる。Delayed ACK の効果で、データパケット側の送信機会を通常より有効に利用できる環境下

では、TCP Vegas がバッファにパケットを溜めない制御になることから、TCP Vegas が素早くパケット送信を行える分、ネットワークにわずかな空き帯域が生じることが考えられる。その空き帯域を TCP Reno が利用していることが理由と考えられる。また、CWmin が 15 から大きくなるにつれて、スループットが低下していっている。これは、単純に TCP Vegas の優先度が TCP Reno よりも小さくなっていくからである。

6.2.4 マルチフローにおける TCP 差別化の効果

マルチフローにおける TCP 差別化の効果について評価する。送受信端末をそれぞれ 10 台にし、TCP Reno と TCP Vegas のフローを各 5 フロー流す。差別化なしと差別化ありの場合の各フローのスループット結果を図 6.12,6.14 に、TCP Reno と TCP Vegas を各 1 フロー分の cwnd、RTT の結果例を図 6.13,6.15 に示す。

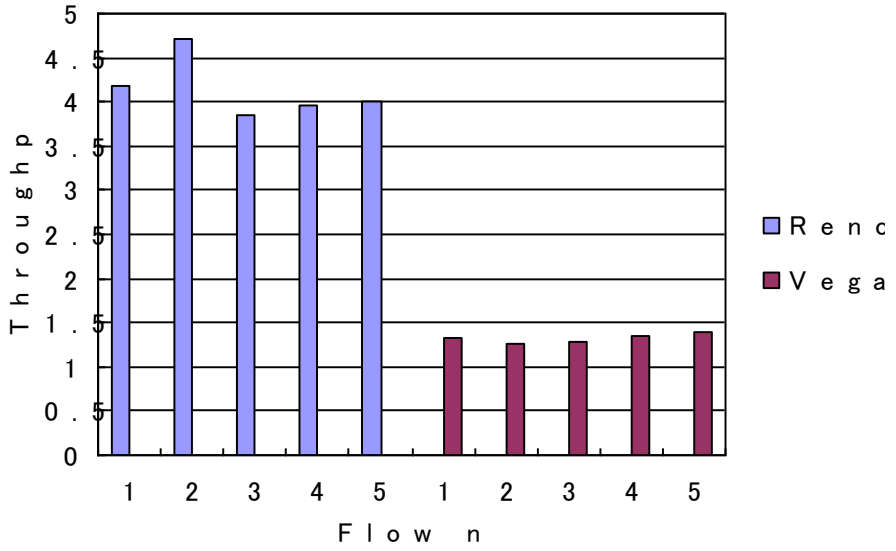


図 6.12　差別化なしの場合の各フローのスループット結果

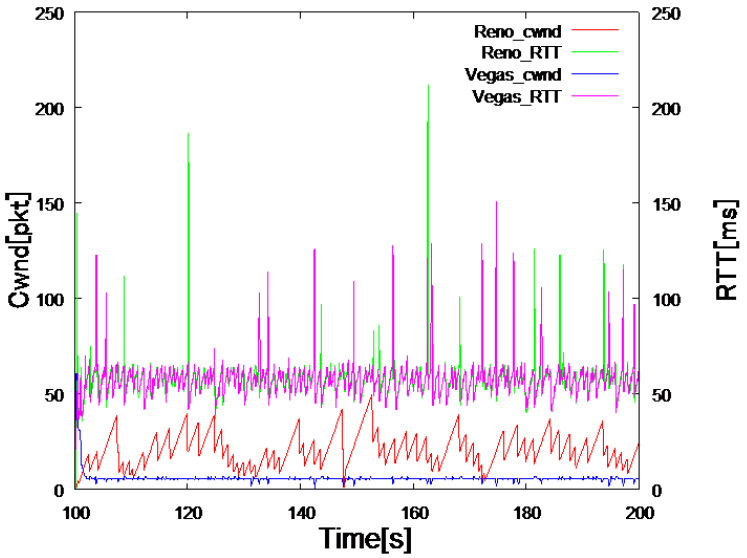


図 6.13　差別化なしの場合の cwnd、RTT 結果例

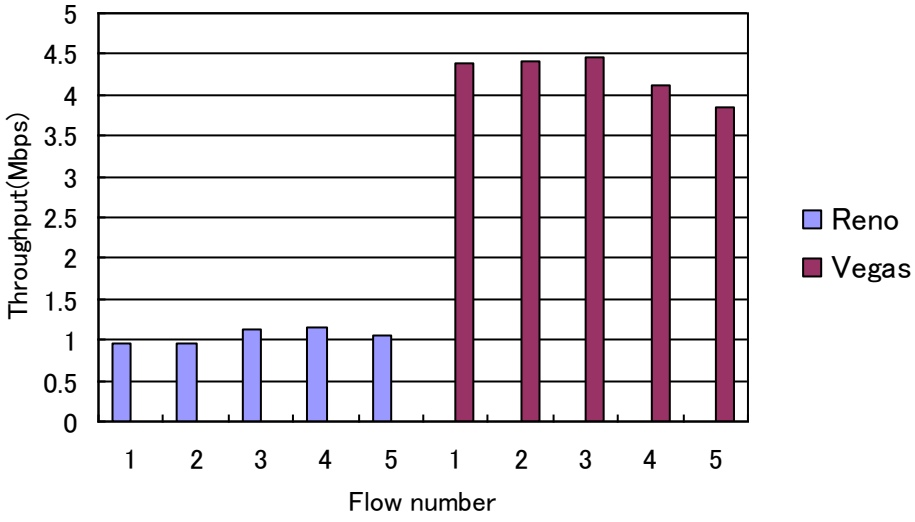


図 6.14　差別化ありの場合の各フローのスループット結果

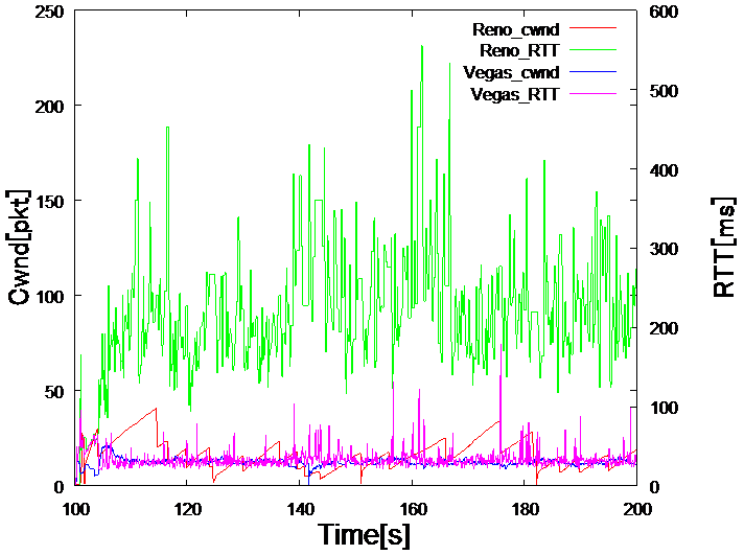


図 6.15　差別化ありの場合の cwnd、RTT 結果例

図 6.12,6.14 より、マルチフローの場合においても、TCP 差別化をすることで、TCP Vegas は高いスループットを得られている。また、TCP Reno と TCP Vegas のパケットを別々のバッファに格納するため、TCP Reno は Reno 同士で、TCP Vegas は Vegas 同士で帯域を分け合っていることがわかる。さらに、フロー数が増加すると、低優先度のフローは CSMA/CA の待ち時間が増大するため、高優先度のフローのスループットの有利性が増すことが言える。図 6.13,6.15 より、差別化ありの場合、低優先度である TCP Reno の RTT が差別化なしのときに比べ、4 倍近く大きくなっている。

6.3　シミュレーション・実機実験環境 2

実験環境 2 は受信端末が 1 台のみ存在する場合を想定する。ネットワークトポロジーを図 6.16 に示す。実験パラメータや EDCA パラメータは実験環境 1 の場合と同様に表 6.1,6.2 の値を用いる。ネットワーク帯域や無線 LAN 規格など、その他の条件も実験環境 1 と同様の設定で実験を行う。

受信端末を 2 台以上の場合との違いは、受信端末が 1 台であると、受信側でロスベースと遅延ベースの TCP の ACK が同じバッファを共有することになる。この点が大きく違ってくる。

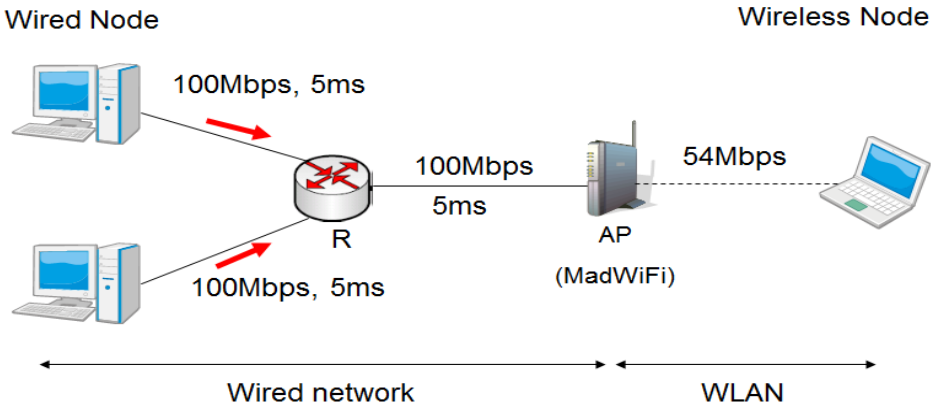


図 6.16　ネットワークトポロジー

6.4　シミュレーション実機実験結果 2

6.4.1　TCP Reno と TCP Vegas を競合させた場合の差別化の効果

基本的な TCP 差別化の効果を明らかにするため、TCP Reno と TCP Vegas を各 1 フロー競合させた実験を行った。なお、実験環境 2 では Delayed ACK が結果に大きく影響してくるため、シミュレーションにおいては Delayed ACK がありの場合となしの場合、それぞれについて評価を行った。シミュレーション・実機実験のスループット、cwnd・RTT の結果をそれぞれ図 6.17~6.21、図 6.22~6.24 に示す。

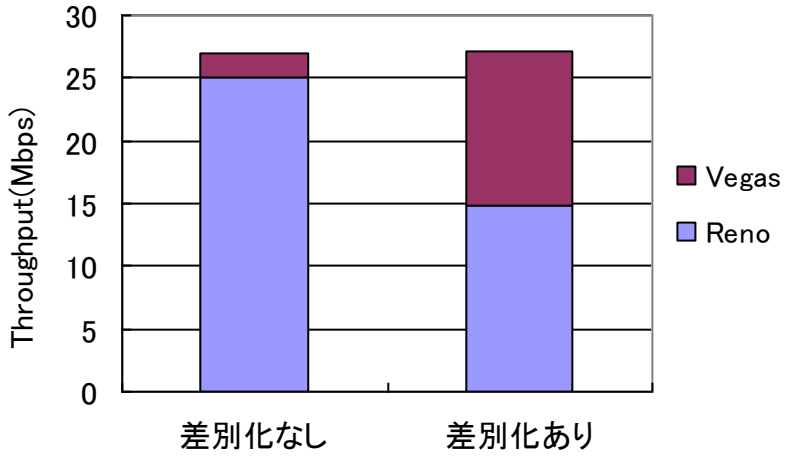


図 6.17　スループット結果(シミュレーション、Delayed ACK あり)

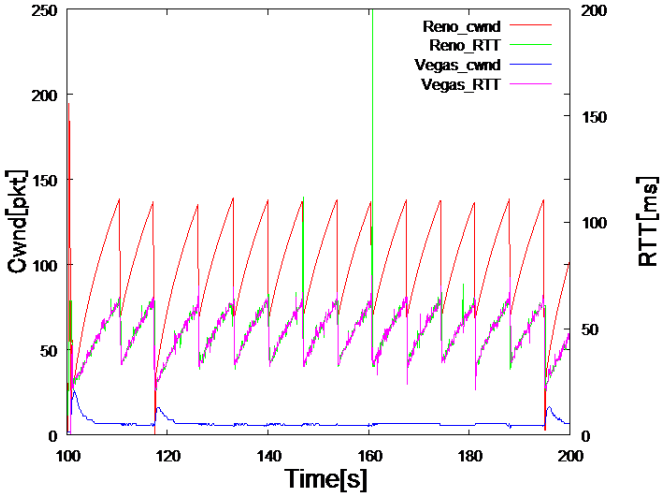


図 6.18　差別化なしの場合の cwnd、RTT 結果(シミュレーション、Delayed ACK あり)

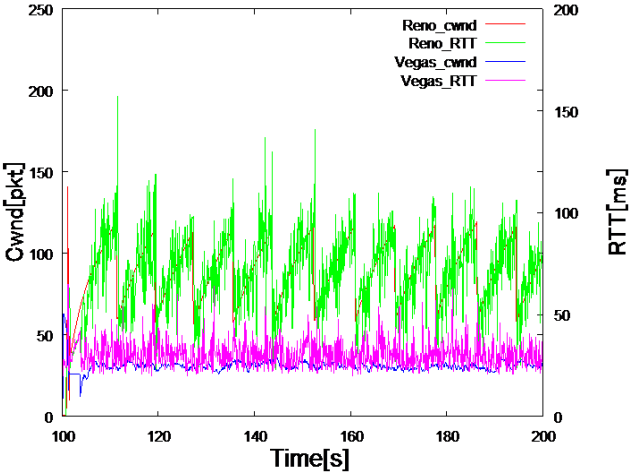


図 6.19　差別化ありの場合の cwnd、RTT 結果(シミュレーション、Delayed ACK あり)

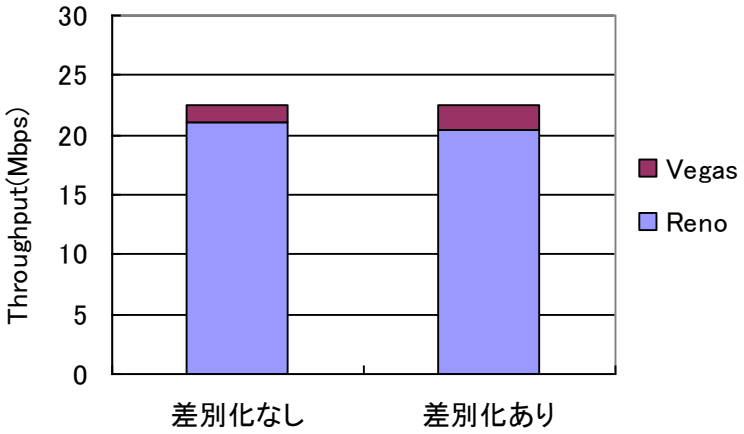


図 6.20　スループット結果(シミュレーション、Delayed ACK なし)

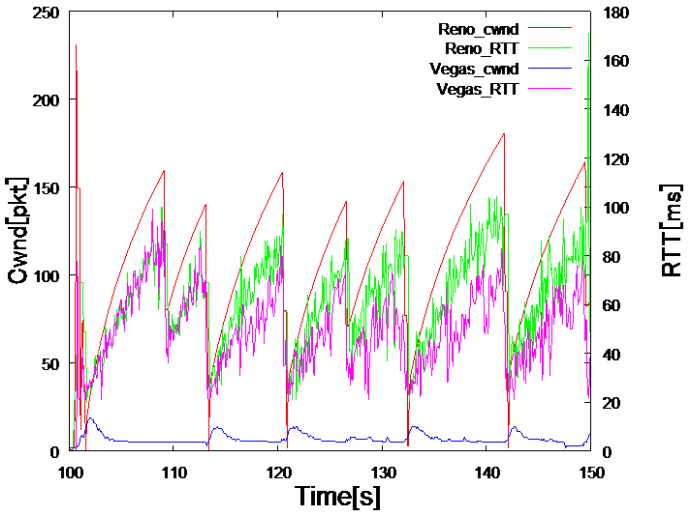


図 6.21　差別化ありの場合の cwnd、RTT 結果(シミュレーション、Delayed ACK なし)

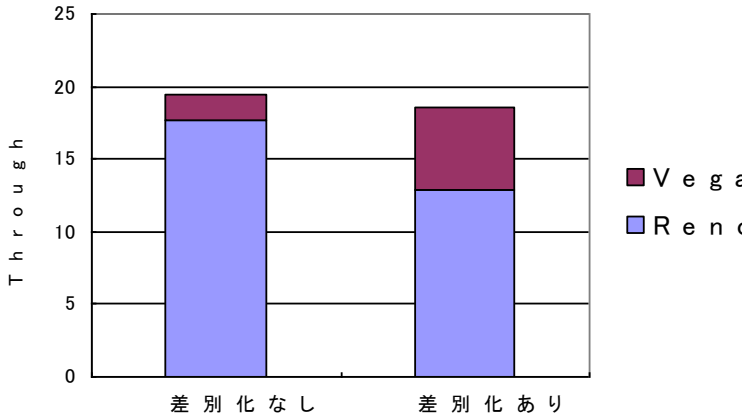


図 6.22　スループット結果(実機)

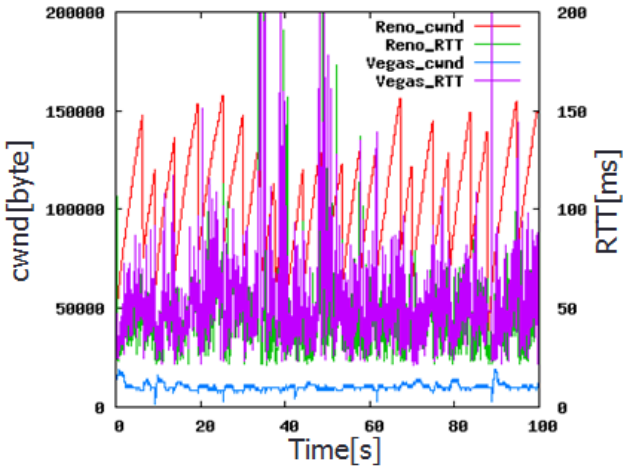


図 6.23　差別化なしの場合の cwnd、RTT 結果(実機)

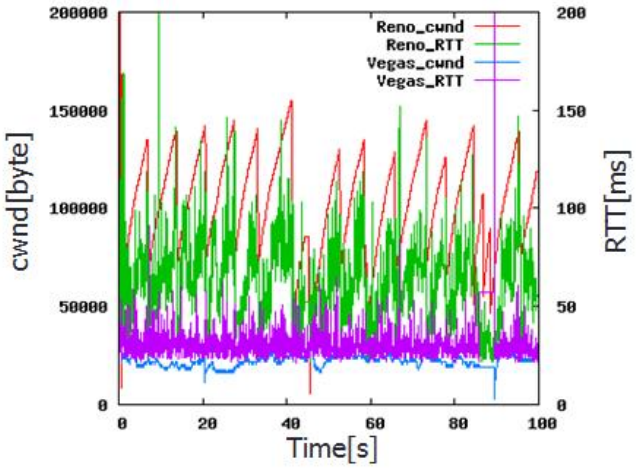


図 6.24　差別化ありの場合の cwnd、RTT 結果(実機)

図 6.17 より、シミュレーションでは、TCP 差別化をすることで、TCP Vegas は差別化なしの場合よりも約 6 倍のスループットを得ることができている。また、図 6.21 より、実機では、TCP 差別化をすることで、差別化なしの場合よりも約 3 倍のスループットを得ることができている。これは Delayed ACK が有効になっている場合の結果である。Delayed ACK がない場合、受信側で TCP Reno と TCP Vegas の ACK が同じバッファを共有しているため、受信側でバッファリング遅延が増大し、その影響で TCP Vegas はスループットを下げてしまうことが考えられる。図 6.20,6.21 より、差別化しても差別化なしの場合に比べて TCP Vegas のスループットの改善は小さく、遅延も大きな改善が見られない。一方で、Delayed ACK が存在する場合、データパケット 2 個に対して 1 個の ACK を返すため、受信バッファに溜まる ACK の個数が大幅に減少することになる。そのため、バッファリング遅延の影響が低減され、TCP 差別化をすることで TCP Vegas は比較的高いスループットを得ることができている。

図 6.19,6.24 より、シミュレーションと実機ともに差別化ありの場合は Delayed ACK の効果で、TCP Vegas は比較的遅延を小さく抑えることができている。cwnd も高い値を維持できている。差別化なしの場合は、図 6.18,6.23 より、実験環境 1 と同様に TCP Vegas の cwnd が非常に低くなっている。このように、受信端末が 1 台の場合は、Delayed ACK が TCP 差別化に大きな影響を与えることがわかる。つまり、Delayed ACK が有効である場合は、TCP 差別化をすることで、TCP Vegas は非常に大きいスループットの改善効果が得られることが言える。

6.4.2 他 TCP バージョンを競合させた場合の TCP 差別化の効果

TCP Vegas が TCP Reno 以外の TCP バージョン(Compound TCP、CUBIC を使用)と競合した場合の TCP 差別化の効果を明らかにする。6.4.1 章の結果からシミュレーションと実機で結果の傾向が変わらないことが予想されるため、シミュレーションでのみ評価を行う。図 6.25 に各 TCP バージョンと競合させたときのスループット結果を、図 6.26,6.27 に Compound TCP と CUBIC の cwnd、RTT 結果を示す。

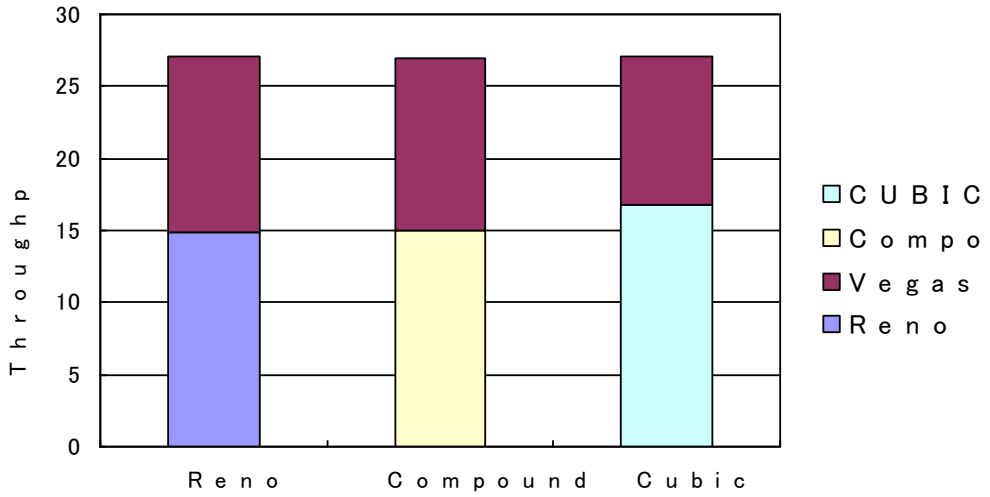


図 6.25　各 TCP バージョンと競合させた場合のスループット結果

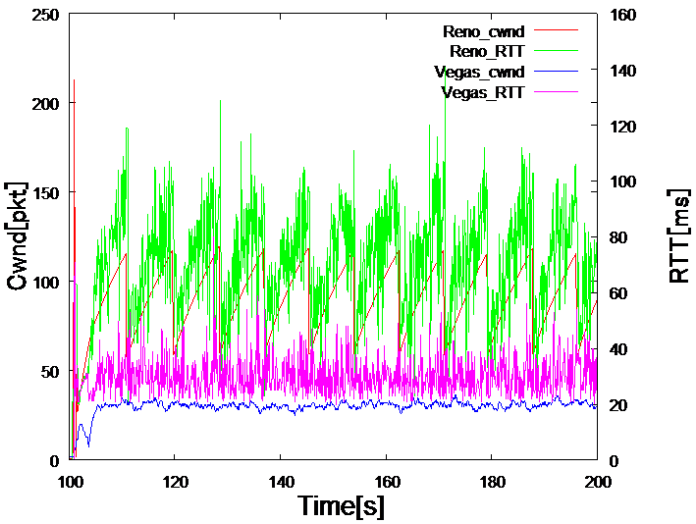


図 6.26　Compound TCP と競合させた場合の cwnd、RTT 結果

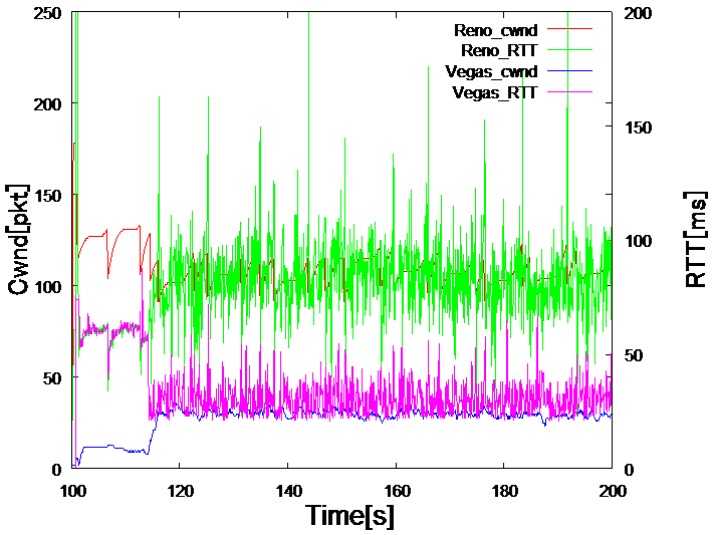


図 6.27　CUBIC と競合させた場合の cwnd、RTT 結果

図 6.25 より、Compound TCP や CUBIC と競合させた場合でも、TCP Vegas は TCP Reno と競合させた場合とほぼ同程度のスループットが得られている。これは、実験環境 1 と同様の理由が考えられ、優先制御を MAC 層で行っていることから、スループットはトランスポート層である TCP のバージョンには依存しないことが言える。

図 6.26,6.27 より、cwnd、RTT 結果においても TCP Reno と競合させたときと同様の傾向が得られている。TCP 差別化をすることで、TCP Vegas は高い cwnd が得られており、RTT も小さく抑えられている。その反面、Compound TCP と CUBIC の遅延が大きくなっている。

6.4.3　EDCA パラメータの CW を変動させた場合の TCP 差別化の効果

表 6.2 で示した EDCA パラメータの CWmin を変動させ、どの程度 TCP 差別化の効果に違いが出てくるのかをシミュレーションにより評価する。TCP Vegas のパケットを格納するバッファの CWmin を 1~1023 まで変動させ、CWmax を 1023 とする。CWmin を変動させた場合のスループット結果を図 6.28 に示す。

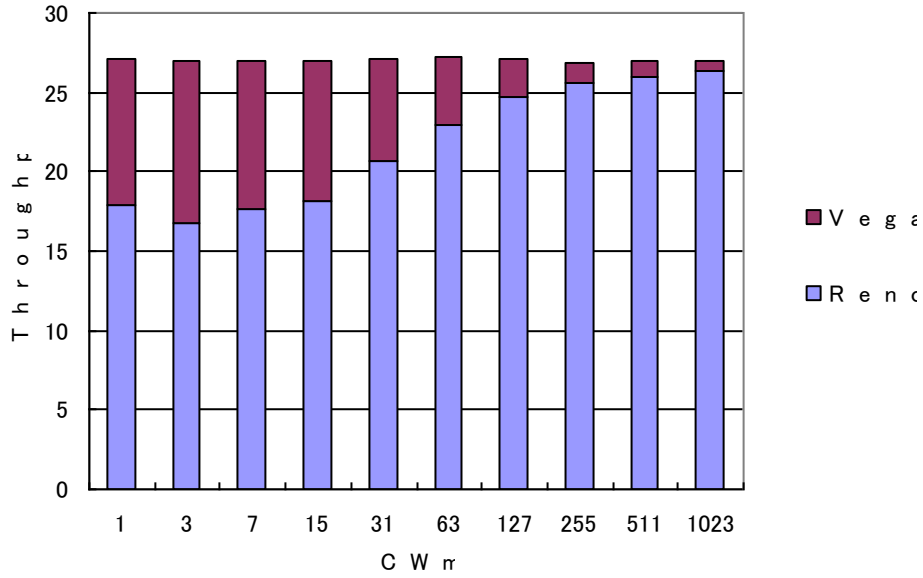


図 6.28　CWmin を変動させた場合のスループット結果

図 6.28 より、TCP Vegas のスループットは CWmin=3 の場合が一番高くなっている。実験環境 1 のときとは違い、CWmin=3 が一番高くなっているのは、実験環境 2 では CWmin=3 のとき、TCP Vegas が帯域を最も有効利用できているからだと言える。CWmin が 3 以下の場合に TCP Vegas のスループットが下がっている理由としては、実験環境 1 と同様のことが考えられ、CWmin が非常に小さい場合、TCP Vegas のパケットは AP に到着後すぐに送信することができるため、Delayed ACK の効果で、データパケット側の送信機会を通常より有効に利用できる環境下では、ネットワークにわずかな空き帯域が生じ、その空き帯域を TCP Reno が利用していることが考えられる。また、CWmin が 15 から大きくなるにつれて、スループットが低下していっている。この理由も実験環境 1 と同様に、単純に TCP Vegas の優先度が TCP Reno よりも小さくなっていくからである。

第7章 まとめと今後の課題

本編では、低遅延である TCP Vegas を有効利用を目的とし、親和性の問題を改善するため、TCP バージョン識別手法と EDCA 制御を併用した TCP 差別化手法を提案し、シミュレーションと実機により評価を行った。実験結果より、ルータ(AP)で TCP バージョンを識別し、Vegas を優先的に送信することによって、TCP バージョンを差別化できることを確認した。受信端末が 2 台の環境において、シミュレーションでは、TCP 差別化をすることで、TCP Vegas は差別化なしの場合よりも約 8 倍、実機では約 4 倍のスループットを得られることを確認した。また、受信端末が 2 台よりも多いマルチフローの場合でも提案手法が有効に働くことを確認した。一方、受信端末が 1 台の環境では、Delayed ACK が大きな影響を及ぼすことを明らかにした。Delayed ACK ありの場合、シミュレーションでは、TCP 差別化をすることで、TCP Vegas は差別化なしの場合よりも約 6 倍、実機では約 3 倍のスループットを得られることを確認した。

今後の課題として、無線 LAN 環境に適した RTT 推定、TCP バージョン識別方法を提案することが挙げられる。また、低遅延の TCP Vegas を用いて、実機実験によるライブストリーミングなどの映像配信評価に検討対象を広げる。

第２編　確率的に変動するリソースを考慮した最適寄り道経路特性

第1章　　序論

1.1　はじめに

近年、携帯電話やスマートフォン、タブレット端末などの普及に伴い、外出先や移動中でも、インターネットや email の送受信、ファイルダウンロードなどネットワーク接続の需要が増大している。また、今後ロボット間あるいは遠隔操作などを含む M2M 通信も増加していき、移動中も常にネットワークで通信するという需要が高まると予想される。これに対し、通信インフラとしては、セルラー通信、WiMAX、無線 LAN などが普及しており、ある程度の移動通信は可能である。しかしながら、携帯電話で使用されている 3G 移動通信は、電波カバレッジは十分なものの、通信速度は遅く、コストも高い。一方、無線 LAN は、電波カバレッジが限られているものの、通信速度は速く、コストも低い。このように、現状ではアプリケーションニーズの多様化に対して、必ずしもアプリケーション QoS を満足できるほど、無線アクセス方式が多様化している状況とはいえない。ただし、今後は無線 LAN 基地局や Femto セル[26]が増加することが望め、これらの通信環境を上手く活用すれば、QoS の良好な移動通信を享受することができると予測される。また、コグニティブ無線通信技術[27]を用いれば、電波カバレッジの無いところでアドホックに良好な通信を行うようなことも可能になるとされている。

そこで、無線リソースの利用効率を上げることを考え、移動時に最短距離ではなく、多少余計に時間がかかっても無線 LAN の基地局近辺など通信が良好な経路を利用することを「寄り道」経路の利用として提案している[27,28]。許容時間範囲内で最大限基地局に近づくといった寄り道経路を用いることで効果(増加情報量 対 増加距離)が向上することが、様々な環境で既に確認されている。

1.2　研究目的

これまでの研究では、ネットワークの帯域などの通信リソースを理想的に(仕様通り)使用できる状況における評価のみが行われていた。しかし、実際の通信環境においては、ネットワークの輻輳や障害など、理想的ではない状況を考慮する必要がある。本研究では、ネットワークの輻輳により、有線ネットワークにボトルネックが発生する場合の、寄り道効果を評価し、理想状況である従来評価との違いを明らかにする。

1.3　本編の構成

第 2 章では提案する寄り道経路の関連研究及び寄り道自体の課題について述べる。
第 3 章では本研究で定義している最適寄り道経路と寄り道の従来評価モデルについて述べる。
第 4 章では提案モデルである確率的に変動するリソースを考慮した最適寄り道経路の特性について述べる。
第 5 章ではまとめと今後の検討について述べる。

第2章　寄り道と関連研究

2.1 寄り道[27,28]

地点間の移動においては、通常その最短経路を辿るのが最も望ましいと考えられるが、無線モバイルユーザにとってはそうではなくなるかもしれない。携帯電話やスマートフォン、タブレット端末などの普及に伴い、外出先や移動中でも、インターネットや email の送受信、ファイルダウンロードなどネットワーク接続の需要が増大してきている中で、移動中でも QoS の良好な移動通信をしたいユーザが増加すると考えられる。

本研究で定義している「寄り道」とは、3G 移動通信に比べて通信速度の速い、無線 LAN のリソースを有効に利用することを目的とし、移動時に最短距離ではなく、多少余計に時間がかかっても無線 LAN の基地局近辺など QoS の良好な移動通信ができる経路を利用することである。図 2.1 に寄り道の概観について示す。

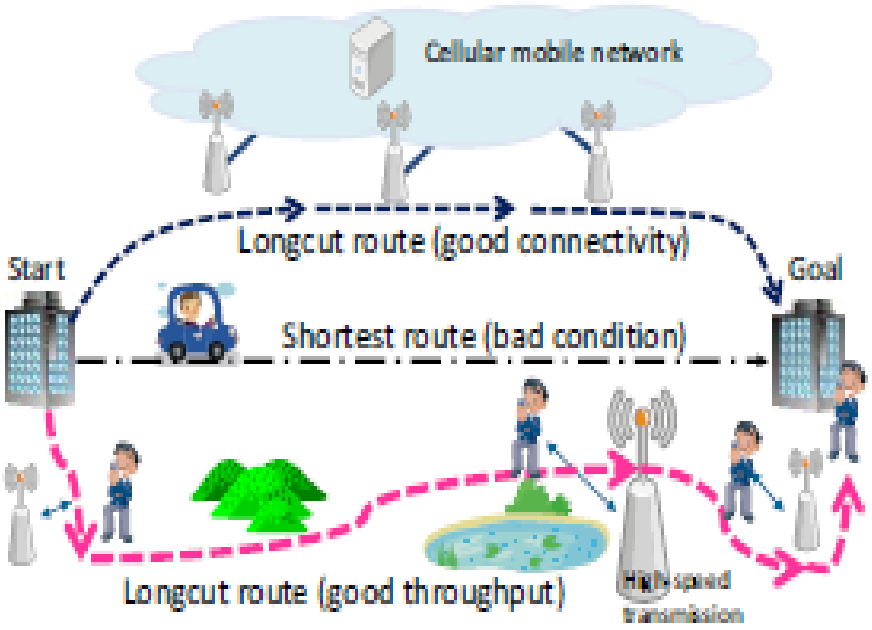


図 2.1　寄り道の概観(出展：参考文献[28])

2.2 関連研究

提案する寄り道経路[27,28]の関連研究及び寄り道自体の課題について述べる。

寄り道に関する関連研究として、モバイルセンサでの情報収集がある。モバイルセンサの場合には、指定されたセンサの情報を必ず収集する、つまり、ある程度決められたところまでセンサに接近する必要がある[29,30]が、「寄り道」では、そのような条件なしに、最も良い経路を求める必要がある点が異なる。

また、パケットルーティングにおいて、類似の技術として、制約範囲内のベストパス（例えば、帯域が最大となる条件での最短ホップ数のパス）を求めるような技術[31][32]が知られている。しかし、IP ネットワークのような実際のネットワークでは、かなり限定されたネットワークトポロジーが仮定され、かつ、パケットがネットワーク内部で積極的に留まることを許容する仮定を含む手法は、知りうる限り存在しない。一方、「寄り道」では原理的には自由な方角距離の移動を許容し、IP ネットワークのトポロジとは比較にならない大きさのグラフから、停止を許容する条件下で最善パスを導くことが大きな違いである。ユーザが最短経路ではなく「寄り道」を選択するかどうかについては、ある程度の利得があった場合には、ユーザが場所を移動することを許容する、といった研究がある[33][34]。なお、移動中に効率よく基地局に接続する技術として、[35]などが提案されている。

寄り道経路の研究では、すでにその効果(増加情報量 対 増加距離)についての評価が行われている。しかしながら、評価の過程においては、寄り道によって得られる通信リソースは常に保証されているとしている。現実的には、通信帯域などは、ネットワークの輻輳などにより変化するため、そのような状況を前提として評価が必要である。

第3章　最適寄り道経路と従来評価

3.1　最適寄り道経路

市街地上の 2 地点間の移動を考え、スタート地点に存在するモバイルユーザがゴール地点まで移動したいものとする。また、市街地上のいくつかの地点には無線通信用アクセスポイント（以下、基地局）が存在するものとする。このとき、以下の 4 つの制約条件を設ける。

1. ユーザはゴール地点に到着するまでの間、基地局からの距離に応じた通信速度で通信することができる。
2. ユーザは移動開始から少なくとも最短経路の移動時間の何倍かの時間までにゴール地点へ到着する必要があるという制約時間を設ける。
3. 制約時間内で、通信品質の良い経路を選択し、移動する。
4. 制約時間内であれば、最も通信速度が大きい地点で停止することができる。

最適寄り道経路とは上記の制約条件を満たす寄り道経路の中で、総転送量が最大になる経路のことをいう。最適寄り道経路の一例を図 2.2 に示す。

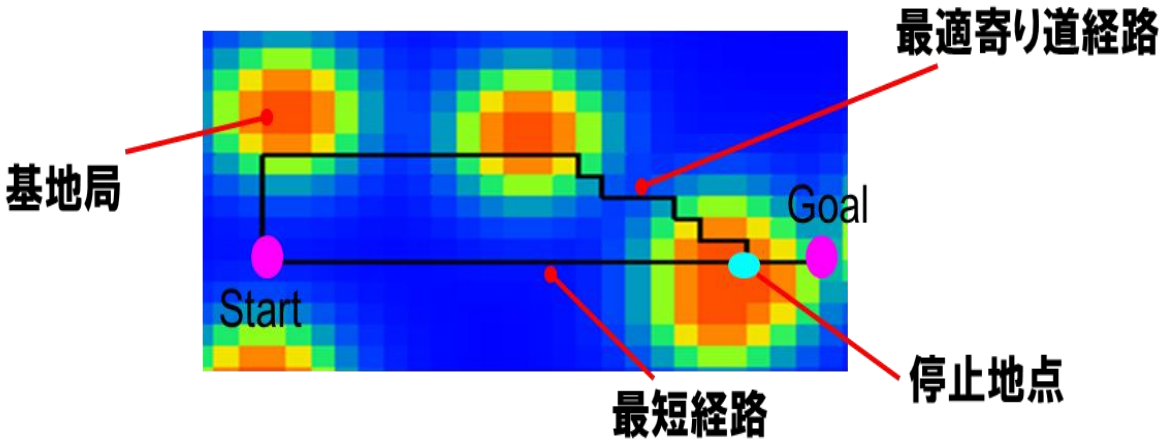


図 2.2　最適寄り道経路の一例

3.2 従来評価モデル[26,27]

本節では、従来評価について説明する。従来評価では、無線 LAN のリソースを理想的に利用できる状況において、最短経路に対する寄り道の通信品質の改善度を評価している。

市街地を想定し、図 3.1 に示すような 40×40=1600 個の頂点（交差点）から成る正方格子の実験フィールドを考える。図 3.1 において、下から i 番目、左から j 番目に位置する頂点を (i, j) で表す。隣接するすべての 2 頂点間の距離は 20[m]であるとするため、頂点 (i, j) と (i', j') との距離は $20\sqrt{(i-i')^2+(j-j')^2}$ となる。ユーザは常に 1[m/s]の速度で移動するものとする。また、スタート地点とゴール地点の位置は（Start=(20,8),Goal=(20,32)）とする。図 3.1 に複数の基地局を配置し、ユーザは頂点 v からの距離が最も近い基地局と距離に応じた通信速度で通信を行えるとする。頂点 v と基地局の存在する頂点との距離を d[m]、基地局の高度を 3[m]、無線規格に IEEE802.11g を用いるとしたとき、ユーザが頂点 v 上で利用可能な通信速度 $P(v)$ を以下で求める。

$$P(v) = \min(54, 20\log_2(1+\frac{1.2\times10^7}{(d^2+3^2)^2})) \text{ [Mbps]}$$

通信速度 $P(v)$ はシャノンの式とフリスの式より、導き出した式であり、基地局からの距離に応じて、通信速度が減少していく式となっている。なお、通信速度 $P(v)$ は干渉などの影響を考慮していないため、今後はこの点も考慮した仮定を立てる必要があると考えられる。通信速度 $P(v)$ は IEEE802.11g の無線 LAN 規格を想定しているため、無線 LAN の通信速度の最大値を 54[Mbps]としている。ただし、実環境においてはチャネル使用効率の問題から 54[Mbps]のスループットは得られない。例えば、ネットワークシミュレータ ns-2 では、TCP を用いて送受信端末 1 対 1 の無線通信を行った場合、IEEE802.11g を用いると最大約 22[Mbps]のスループットであった。また、端末数に応じて CSMA/CA 時のコリジョンや RTS/CTS の使用などの要因で性能劣化が起こる。これらにより、スループットは低下するが、寄り道の特性としては、大きな差はないと考えらえるため、数値計算で得られるデータ転送量や Improvement ratio(4.2 章で説明)の値、グラフの傾きに多少の変化はでてくるものの、得られる結果の特性にも大差はないと考える。

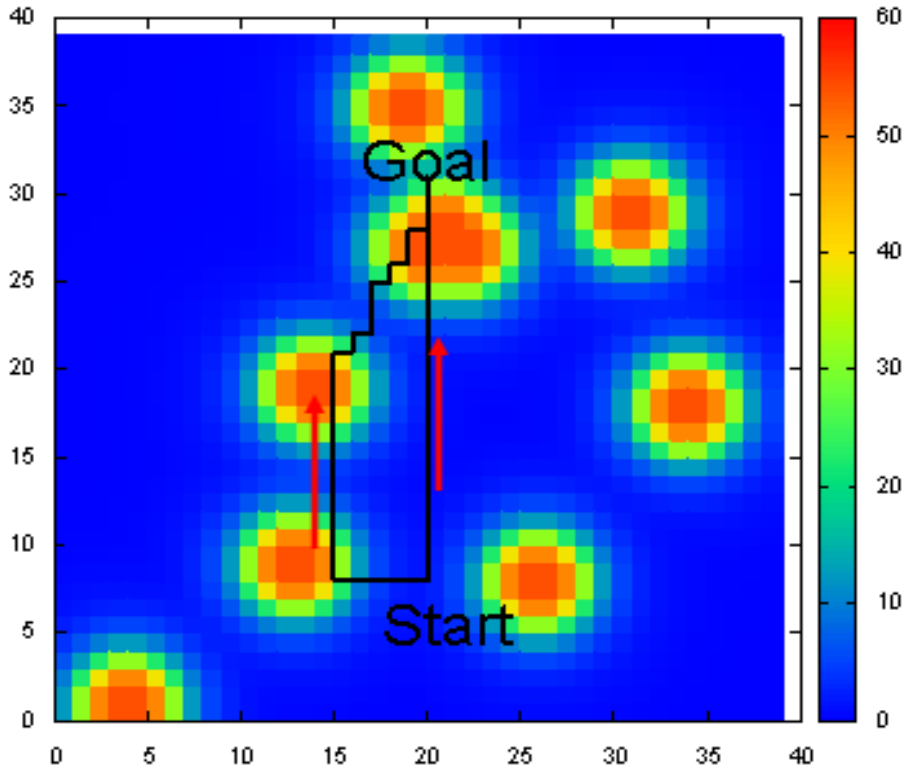


図 3.1　実験フィールド

最適寄り道経路の探索アルゴリズムは、基本的には制限時間を超えない範囲で深さ優先探索(DFS)全経路探索を行う。しかし、探索対象を、停止を含む移動経路とすると、探索空間が非常に大きくなってしまうため、停止を含まない経路に探索対象を限定する。この停止を含まない経路の中で総転送量が最も多い経路を分岐限定法により見つけ出し、最も総転送量が多い経路を最適寄り道経路とする。

3.3 従来評価モデルの実験結果

従来研究では、基地局をランダムに配置した 1000 個のマップに対して、以下の評価を行っている。また、マップは図 3.2 に示すように 2 つフィールドを用いて評価を行っている。

評価 1：　制限時間と最適寄り道経路の通信品質の関係

評価 2：　基地局数と総転送量改善率の関係

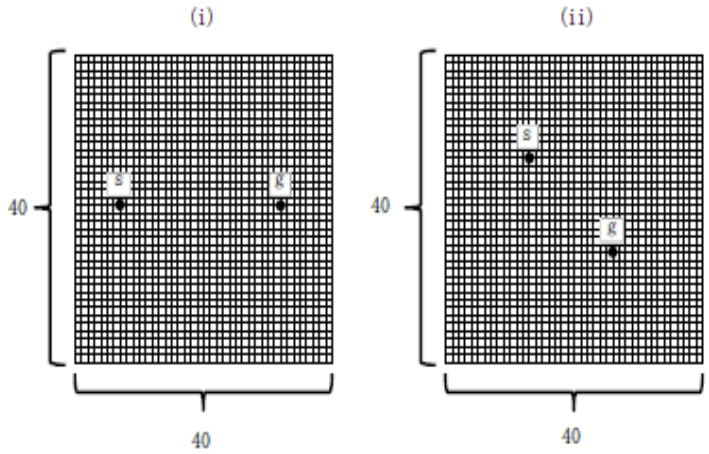


図 3.2　2 つ実験フィールド （ｉ）フィールド 1 （ii）フィールド 2(出展：参考文献[27])

3.3.1　制限時間と最適寄り道経路の通信品質の関係

評価 1 では、寄り道の効果を定量的に把握するため、基地局の配置が異なる 1000 個のマップに対してその最適寄り道経路と最適最短経路の総転送量を評価する。マップの基地局数は 10 個とする。10 個の基地局はマップ中の各頂点に重複を避けて一様ランダムに配置される。制限時間を 480[s]（図 3.2 の s から g までの移動に要する最短時間）から 960[s]（最短距離の 2 倍）まで変化させたときの最適寄り道経路の総転送量を求めている。制限時間と最適寄り道経路の通信品質の関係を図 3.3~3.5 に示す。

結果からわかるとおり、フィールド 1、フィールド 2 ともに、制約時間が増加するにつれて、総転送量が線形的に増加することが確認されている。フィールド 1 とフィールド 2 の最適最短経路の総転送量に r = 480）に差が生じるのは、フィールド 1 においては最短経路がひとつしか存在しないのに対し、フィールド 2 においては最短経路が多数存在するために、最短経路で総転送量の高いものを選択できるためである。

制限時間の増加によって総転送量が増加する要因は 2 つある。ひとつは，制限時間の増加によって通信品質のより良好な経路を選択できるようになること。もうひとつは、通信速度最大の点で停止できる時間が増加することである。これらの要因の貢献度は、最適寄り道経路の総転送量を移動時の転送量と停止時の転送量に分割して評価することで把握できる。フィールド 1、フィールド 2 のいずれにおいても、制約時間が大きい場合、移動時転送量の増加よりも停止時転送量の増加の方がより総転送量の増加に貢献している。特に、最適最短経路の総転送量が高くなり易いフィールド 2 においては移動時転送量の増加が乏しい。

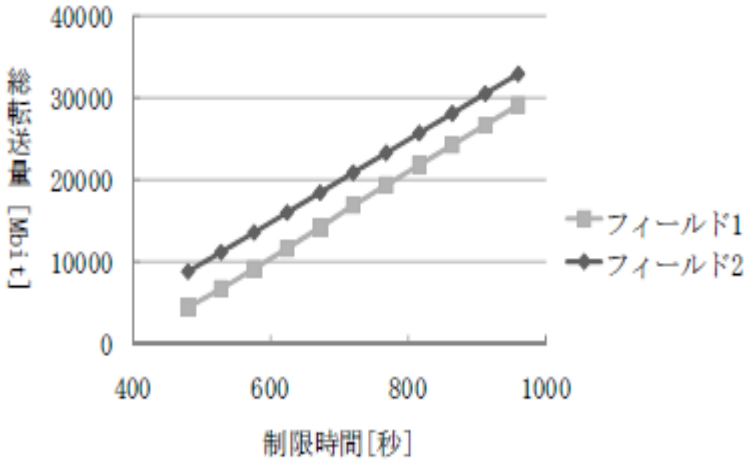


図 3.3　制限時間と最適寄り道経路の通信品質の関係(出展：参考文献[27])

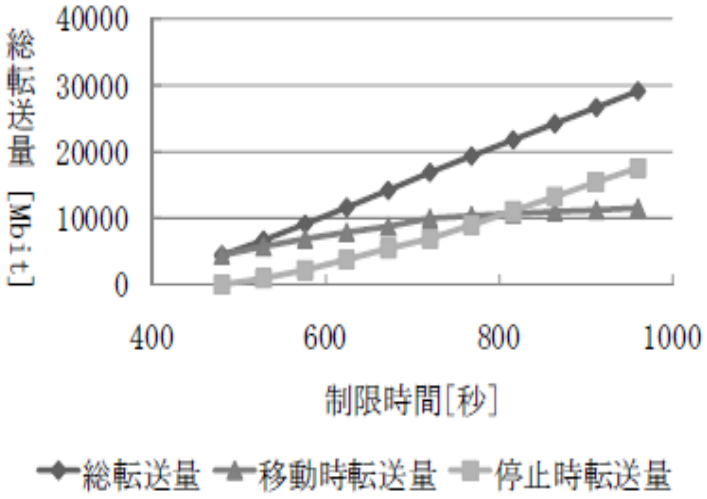


図 3.4　移動時転送量と停止時転送量(フィールド 1) (出展：参考文献[27])

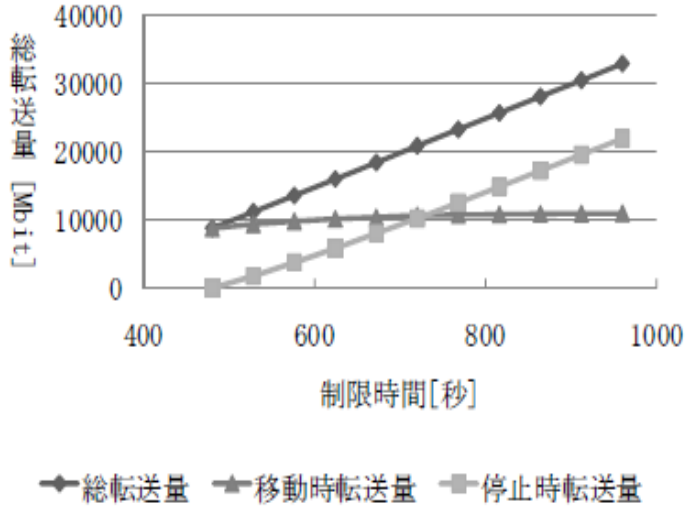


図 3.5　移動時転送量と停止時転送量(フィールド 2) (出展：参考文献[27])

3.3.2　基地局数と総転送量改善率の関係

評価 2 では、フィールド中の基地局数と最適寄り道経路の総転送量の関係を評価する。制約時間が最短経路の 1.3 倍、1.6 倍、2 倍のそれぞれの場合において、基地局数を変動させた場合の総転送量改善率を求めている。評価 1 と同様、基地局はマップ中の各頂点に重複を避けて一様ランダムに配置される。基地局数と総転送量改善率の結果を図 3.6,3.7 に示す。

結果からわかるとおり、寄り道による総転送量改善率は基地局の少ない場合に顕著であり、基地局を増加させていくと、総転送量改善率は小さくなることが確認されている。

最短経路では、総転送量はその経路上の付近に存在する基地局数に大きく影響を受けると考えられる。特に、始点と終点の間に基地局が存在しない場合、総転送量は非常に小さなものになる。一方、寄り道の場合は、基地局が多少遠方に存在する場合でも、寄り道を行うことでその基地局周辺を通過することができる。また、基地局周辺を通過することさえできれば、通信速度の最も高い地点で停止することにより総転送量が稼げる。したがって、寄り道の場合には、少量の基地局数で一定の総転送量を確保できると予想される。しかし、基地局数の増加に対する最適寄り道経路の総転送量の増加は、基地局数がある一定以上の値を超えると緩慢になると考えられる。なぜなら、多数の基地局の周辺を縫うようにして通過するよりも、寄り道の距離をなるべく抑え、通信速度の高い地点で長時間停止する方が、総転送量が大きくなるからである。また、最短経路上の付近に基地局が存在する確率が高くなるため、最短経路の総転送量も大きくなることが理由として考えられる。

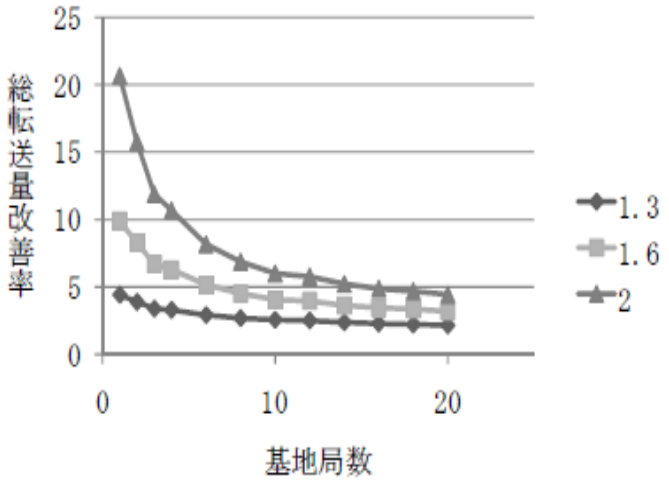


図 3.6　基地局数と総転送量改善率(フィールド 1) (出展：参考文献[27])

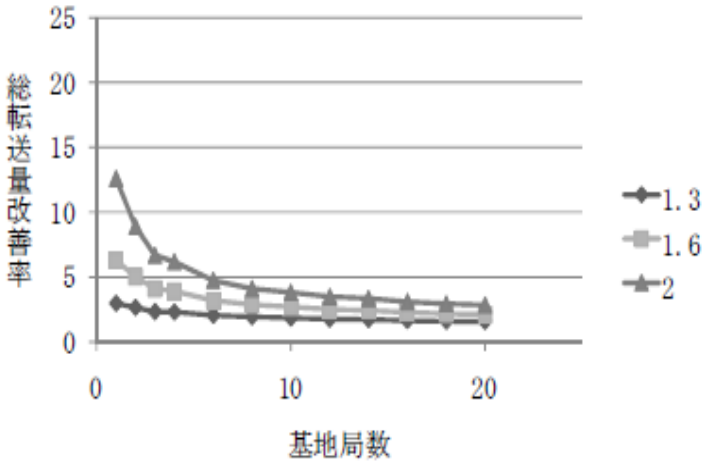


図 3.7　基地局数と総転送量改善率(フィールド 2) (出展：参考文献[27])

第4章　確率的に変動するリソースを考慮した最適寄り道経路の特性評価

4.1 提案評価モデル

本章では、従来評価モデルに、さらにネットワークの輻輳によってボトルネックが発生する場合を想定し、最適寄り道経路の通信特性評価を行う。

第 3 章で説明した従来評価では、無線 LAN のリソースを理想的(仕様通り)に利用できる状況を想定しているが、実際には、ネットワークの輻輳により、ボトルネックが発生する。そこで、図 4.1 のように有線ネットワーク側にボトルネックが存在する場合を考える。各基地局は別々の有線ネットワークを介してインターネットに繋がっており、各基地局に繋がれている有線ネットワークに、独立にボトルネックが発生するものとする。ここで、ボトルネックとは、有線リンクの仕様上の容量に対して、トラフィック負荷や障害のために、実際に使用できる帯域が一定の割合になることを意味する。ユーザの移動速度や通信速度の式、寄り道の探索アルゴリズムなどのその他の設定は、従来評価モデルと同一のものを使用する。フィールドは図 3.1 の実験フィールドを用いる。

提案評価モデルにおいて、基地局をランダムに配置した 1000 個のマップに対して、評価を行い、理想状況との違いとボトルネックが存在することによる最適寄り道経路の特性を明らかにする。

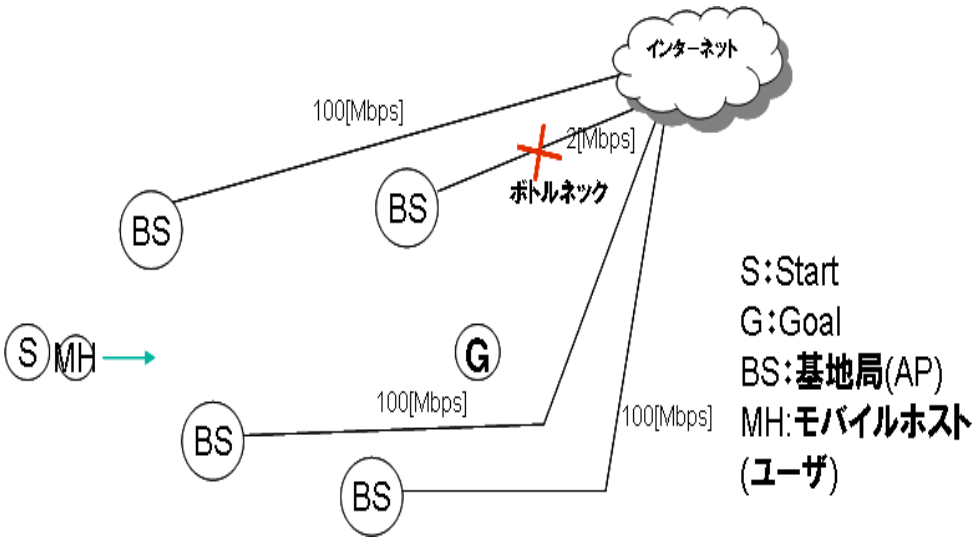


図 4.1　ネットワークトポロジー

4.2 ボトルネック予測と寄り道経路

有線ネットワークにボトルネックが存在することにより、仕様上の通信速度を前提として計算した場合とは異なる性能が得られることが予想される。今回は、ボトルネックが発生した場合として、ユーザがボトルネックの発生箇所を予測できるか否かの 2 つの場合を考える。ユーザがボトルネックの発生箇所を予め知ることが出来る場合を「予測可」と呼び、予測無しに最適寄り道経路を選んだ結果、ボトルネックの影響を受ける場合を「予測不可」と呼ぶ。

予測可の場合、寄り道は、制約時間が許す範囲でなら、有線ネットワークにボトルネックが存在し、ボトルネックの影響を受ける基地局を避けた経路を選択することができる。予測不可の場合、寄り道しても有線ネットワークにボトルネックが存在し、ボトルネックの影響を受ける基地局の近くを通過する可能性がある。これが予測可と予測不可の基本的な特性である。最短経路は一直線の経路しか存在しないため、最短経路上にボトルネックの影響を受ける基地局が存在する場合、必ずボトルネックの影響を受けるといった特徴がある。

評価においては、様々なボトルネック形態を考慮して、特性評価を行った。ボトルネックを表現するパラメータとして、有線ネットワークにボトルネックがあり、ボトルネックの影響を受ける基地局の数(存在する基地局全体に対する割合)、ボトルネック回線の利用可能帯域の２つを用いている。

4.3 提案評価モデルの特性評価

特性評価のための指標である Improvement ratio を後述のように定義し、4.2 章で述べたボトルネックを表現するパラメータを変化させて、最適寄り道経路の通信特性を数値計算により評価した。具体的には以下の 5 つの関係について、予測可・予測不可の場合、それぞれにおける評価を行った。

評価 1：1000 個のマップと Improvement ratio の関係
評価 2：Longcut ratio と Improvement ratio の関係
評価 3：Base station number と Improvement ratio の関係
評価 4：Bottleneck node fraction と Improvement ratio の関係
評価 5：Bottleneck bandwidth と Improvement ratio の関係

Improvement ratio は、最適寄り道経路の総転送量が最短経路の総転送量の何倍かを表す。Longcut ratio は最適寄り道経路が最短経路に対して何倍の経路までを許容するかを表す。Base station number はマップに配置する基地局数である。Bottleneck node fraction は配置した基地局数が何%の確率でボトルネックとなるかを表す。Bottleneck bandwidth は有

線ネットワークがボトルネックとなった場合の有線ネットワークのネットワーク帯域(通信速度)を表す。ボトルネックが存在することにより、仕様上の容量を前提として計算した場合とは異なる性能が得られる。

4.3.1　1000 個のマップと Improvement ratio の関係

寄り道の特性は、基地局の配置に大きく依存することが従来評価により知られているが、ボトルネックが存在する場合には、基地局配置にさらに大きく依存することになるため、はじめに、基地局の配置と Improvement ratio の関係について述べる。各パラメータは固定値を用い、Longcut raio =2、Base station number =10、Bottleneck node fraction =70%、Bottleneck bandwidth =2Mbps とする。基地局は各頂点に重複を避けて一様ランダムに配置する。1000 個のマップと Improvement ratio の関係を図 4.2 に示す。

基地局をランダムに配置した 1000 個のマップを用いた場合、各マップの Improvement ratio は基地局の配置に大きく依存する。Improvement ratio が大きいパターンとしては、最短経路に基地局が存在せず、最適寄り道経路は基地局の近くを通過することができるパターンがある。反対に、Improvement ratio が小さいパターンとしては、最短経路と最適寄り道経路共に、基地局が存在する、あるいは、基地局が存在しないパターンなどがある。また、上述した通り、予測可と予測不可では、ボトルネックの発生箇所を予測できるほうが Improvement ratio が高くなる。

図 4.2 は、1000 個のマップそれぞれの Improvement ratio を Improvement ratio の小さい順にソートして、プロットしたものである。図 4.2 から、Improvement ratio が最大では 200 以上、小さい場合は 0 に近い値と、基地局の配置パターンにより、大きなばらつきが存在していることを確認した。また、予測不可よりも予測可のほうが、平均的には高い Improvement ratio の値を取るであろうことが予想でき、図 4.2 の結果からも、マップ全体において予測不可よりも予測可のほうが、高い Improvement ratio の値を取ることがわかった。

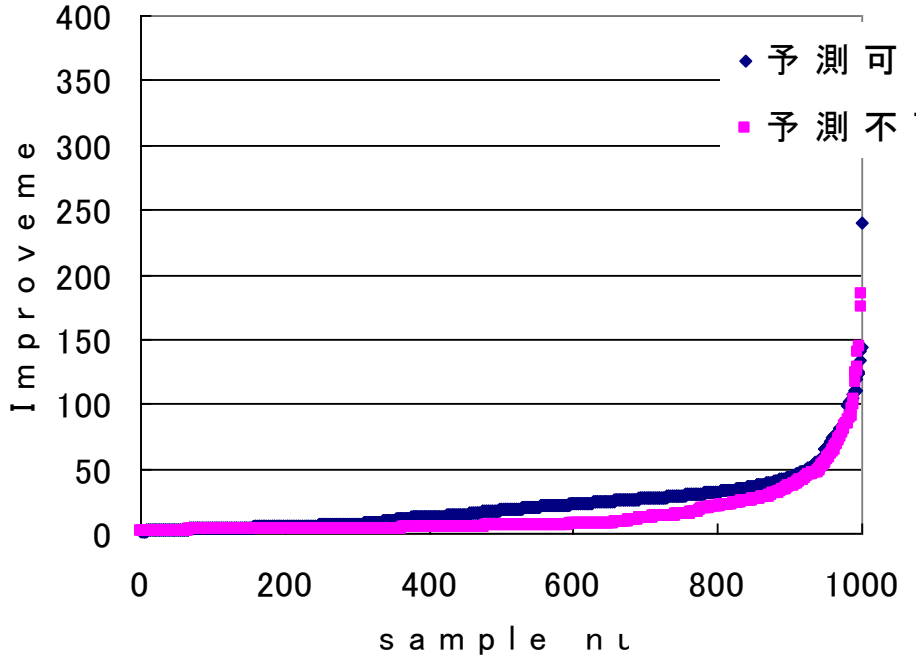


図 4.2　1000 個のマップと Improvement ratio の関係

4.3.2　Longcut ratio と Improvement ratio の関係

ボトルネックが存在する場合の制約時間に対する最適寄り道経路の基本的な特性を明らかにする。各パラメータは、Base station number =10、Bottleneck node fraction =70%、Bottleneck bandwidth =2Mbps とする。基地局は各頂点に重複を避けて一様ランダムに配置する。Longcut ratio と Improvement ratio の関係を図 4.3 に、停止時と移動時の総転送量の関係を図 4.4 に示す。

Longcut ratio が増加すると、最適寄り道経路は通信品質のより良好な経路を選択できるようになることや最大の通信速度の地点で停止できる時間が増加することにより、Improvement ratio は増加する。図 4.3 より、Longcut ratio を増加させると、ボトルネックの有無に関わらず、Improvement ratio の値がほぼ線型に増加していることが確認できる。

また、ボトルネックが存在する場合において、最短経路上にボトルネックの影響を受ける基地局が存在する場合、最短経路はボトルネックが存在しないときよりも転送量を稼ぐことができなくなる。そのため、ボトルネックが存在することで寄り道をする効果は大きくなる。特に、予測可の場合は、ボトルネックの発生箇所が予測できるため、移動時・停止時に関わらず、可能な限りボトルネックの影響を受けない良好な通信が行え、より大きな改善効果が得られる。図 4.3 より、Longcut ratio=6.0 のとき、Improvement ratio は予測可で約 60、予測不可で約 35、ボトルネック無しで約 30 となっている。ボトルネックが

存在する場合(予測可)はボトルネックが存在しない場合に比べ、寄り道効果が約 2 倍大きくなることが確認できる。また、予測可は予測不可に比べ、寄り道効果が約 1.7 倍大きいことが確認できる。

図 4.4 の停止時と移動時の総転送量の関係より、予測可も予測不可も停止時に転送量を大きく稼いでいることが確認できる。停止時の総転送量は Longcut ratio が増加すると増加しており、移動時の総転送量は Longcut ratio を増加させていくと飽和することがわかる。これは、Longcut ratio が増加しても、最適寄り道経路には変更がなく、最大通信速度の地点での停止時間が長くなっているからである。また、予測可が予測不可より転送量を稼ぐことができる理由は、予測可がボトルネックの影響を受けない箇所で停止できる可能性が高いからである。

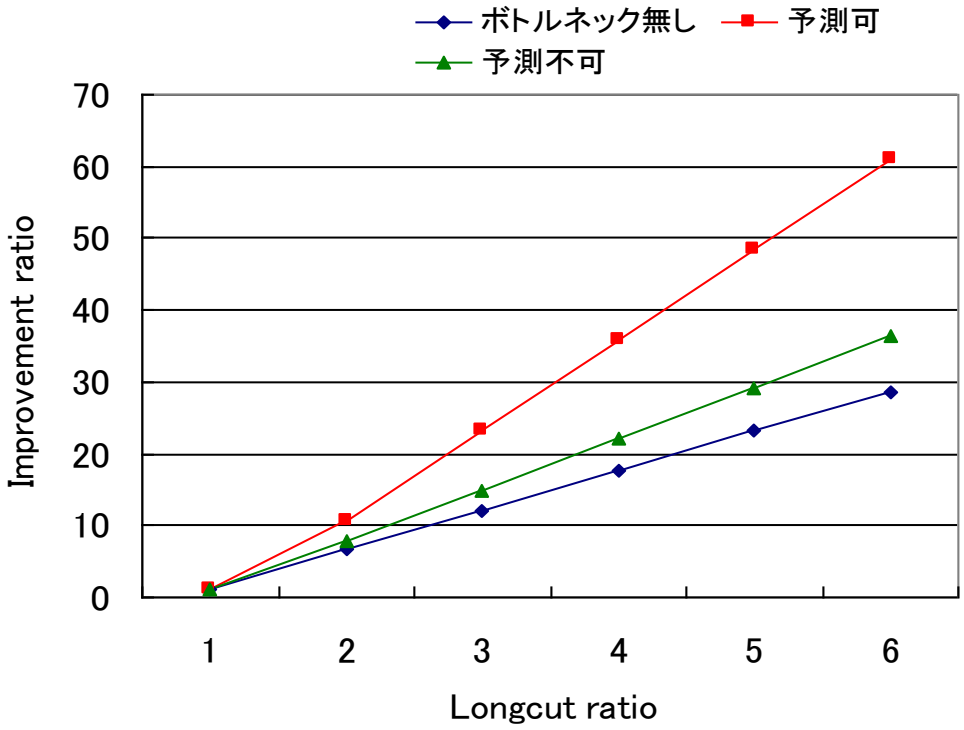


図 4.3　Longcut ratio と Improvement ratio の関係

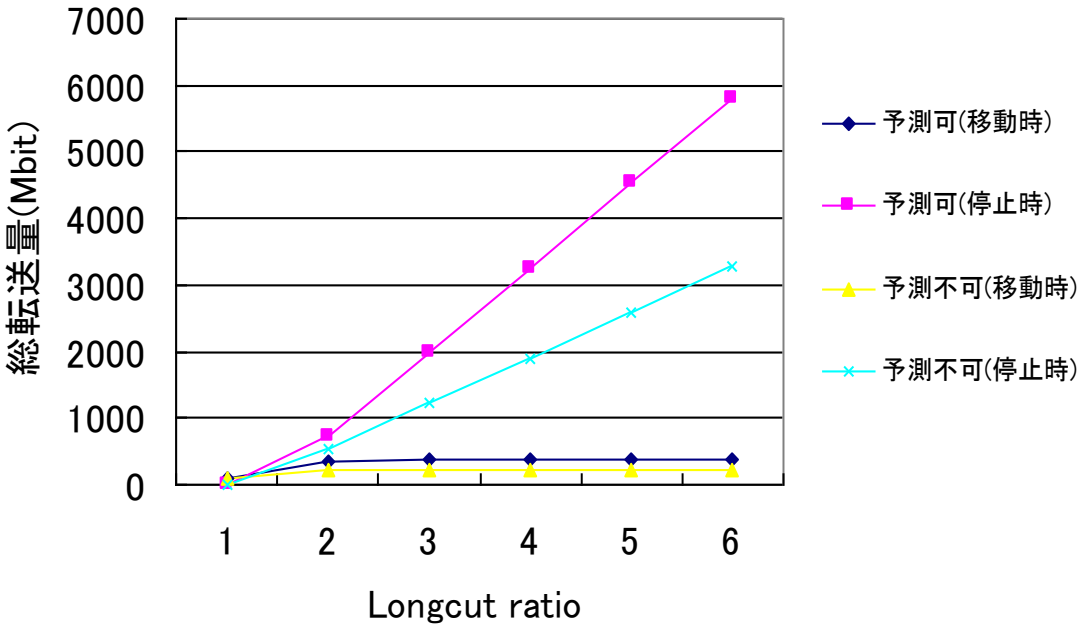


図 4.4　停止時と移動時の総転送量の関係

4.3.3　Base station number と Improvement ratio の関係

ボトルネックが存在する場合に、基地局数を変動させたときの最適寄り道経路の特性を明らかにする。各パラメータは、Longcut raio =2、Bottleneck node fraction =70%、Bottleneck bandwidth =2Mbps とする。基地局は各頂点に重複を避けて一様ランダムに配置する。Base station number と Improvement ratio の関係を図 4.5 に示す。

基地局数が少ないとき、最短経路に基地局が存在する確率は非常に低い。そのため、最適寄り道経路が基地局の近くを通過できれば、非常に大きな寄り道効果が得られるはずである。基地局数が多いと、最短経路に基地局が存在する確率が高くなるため、寄り道効果は小さくなるはずである。

図 4.5 より、ボトルネックの有無に関わらず、寄り道効果は基地局の少ない場合に大きく、基地局が増加するほど小さくなる傾向が得られている。この傾向は従来評価の結果と同様である。

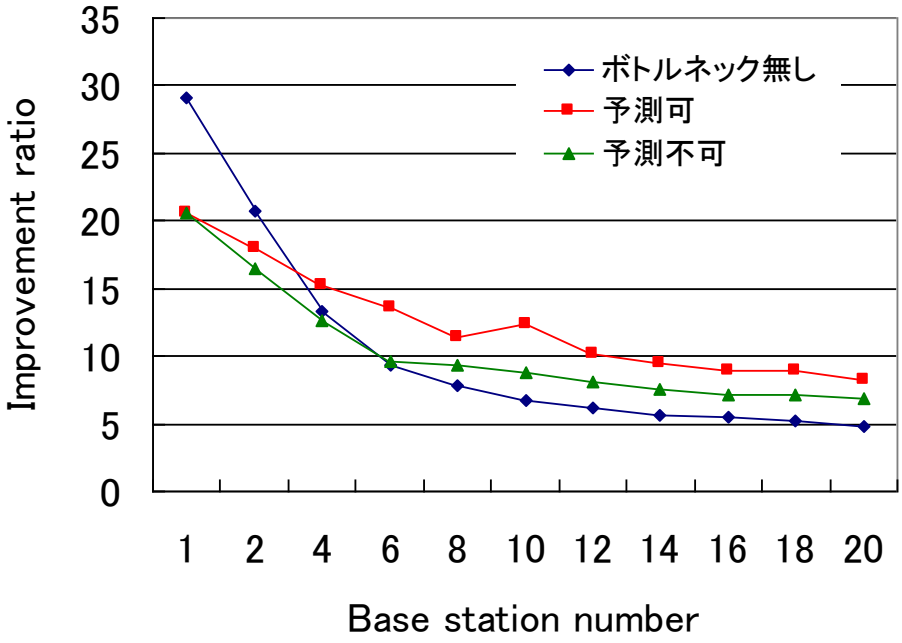


図 4.5　Base station number と Improvement ratio の関係

4.3.4　Bottleneck node fraction と Improvement ratio の関係

有線ネットワークにボトルネックが発生する確率を変動させた場合の最適寄り道経路の特性を明らかにする。各パラメータは、Base station number =10、Bottleneck bandwidth =2Mbps とし、Longcut raito が 2、6 の場合それぞれについて評価を行った。基地局は各頂点に重複を避けて一様ランダムに配置する。Bottleneck node fraction と Improvement ratio の関係を図 4.6、4.7 に示す。

Bottleneck node fraction を増加させると、最短経路上にボトルネックの影響を受ける基地局が存在する確率が増加する。ここで、予測可は Bottleneck node fraction がある値までは、寄り道することでボトルネックの影響を極力受けずに通信が行えるため、寄り道効果が大きくなるはずである。しかし、Bottleneck node fraction が過度に大きくなると、最適寄り道経路もボトルネックの影響を受ける確率が高くなるため、寄り道効果は小さくなると考えられる。予測不可は、最適寄り道経路でもボトルネックの影響を受ける可能性があるため、予測可とは違い、大きな寄り道効果とはならないはずである。

図 4.6 より、Longcut ratio = 2 のとき、予測可の場合は Bottleneck node fraction が 70%のときが Improvement ratio のピークで 10 倍強の改善率を示している。Bottleneck node fraction が 90%のときは、Improvement ratio の値が下がっていることが確認できる。これは、Bottleneck node fraction が過度に大きい状態であるため、上記で述べた通り、最適

寄り道経路もボトルネックの影響を受ける確率が高くなっていることが原因である。しかし、Longcut ratio = 6 と大きくした図 4.7 では、Bottleneck node fraction が 90%のとき、70%のときよりも Improvement ratio の値が大きくなっていることが確認できる。これは、Bottleneck node fraction が大きいことによる最適寄り道経路の転送量の減少率よりも、最大通信速度の地点で停止したときの転送量の増加率の方が大きくなっているからだと考えられる。

図 4.6,4.7 より、予測不可の場合、全体的に予測可よりも Improvement ratio が小さいことが確認できる。Bottleneck node fraction が 90%のとき、Bottleneck node fraction が 0%（ボトルネック無し）の場合よりも、Improvement ratio が若干小さいことが確認できる。これは、ボトルネックの発生率が高いため、最短経路も最適寄り道経路も総転送量を稼ぐことができていないと考えられる。

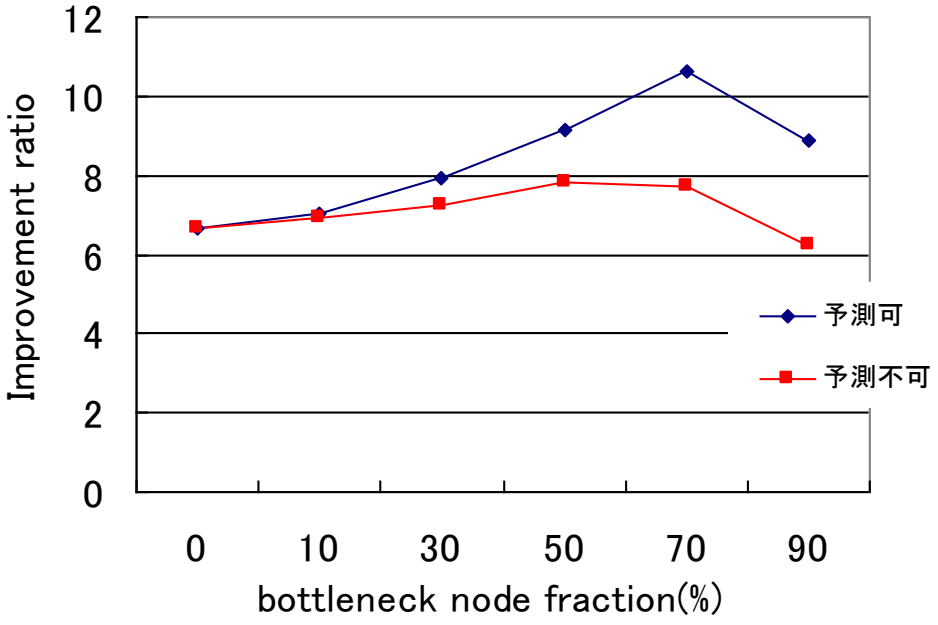


図 4.6　Bottleneck node fraction と Improvement ratio の関係

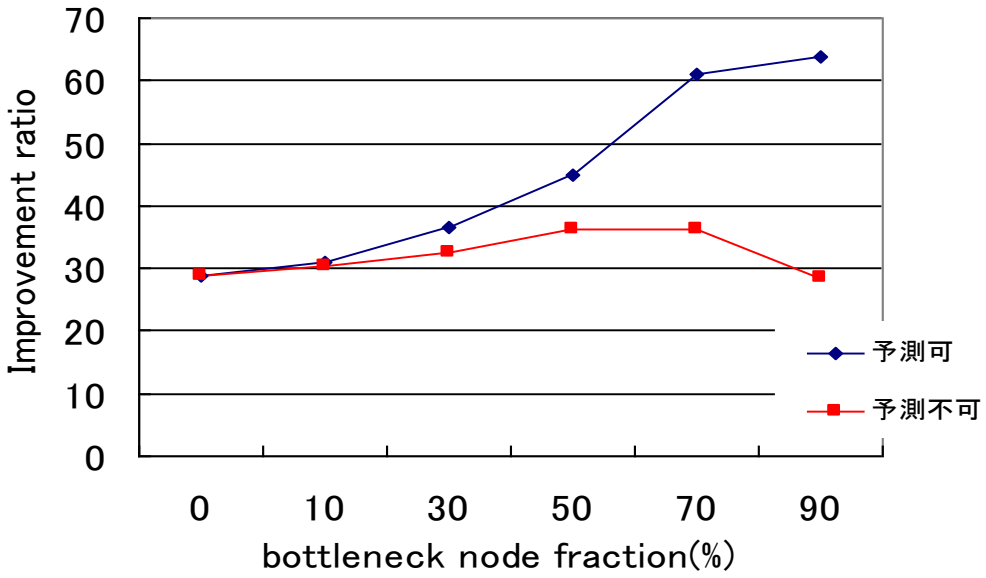


図 4.7　Bottleneck node fraction と Improvement ratio の関係

4.3.5　Bottleneck bandwidth と Improvement ratio の関係

ボトルネックとなったときの有線ネットワークの帯域を変動させた場合の最適寄り道経路の特性を明らかにする。各パラメータは、Base station number =10、Longcut ratio= 2 、Bottleneck node fraction =70%とする。基地局は各頂点に重複を避けて一様ランダムに配置する。Bottleneck bandwidth と Improvement ratio の関係を図 4.8 に示す。

Bottleneck bandwidth を変化させたとき、最短経路がボトルネックの影響を受けるならば、Bottleneck bandwidth が小さいほど、最短経路の総転送量は小さくなる。予測可は、寄り道により、ボトルネックの影響を極力受けないため、最短経路の総転送量が小さい分、Improvement ratio は大きくなるはずである。反対に、Bottleneck bandwidth を大きくすると、ボトルネックの影響を受けても最短経路の総転送量はそれほど小さくはならないため、予測可の Improvement ratio は小さくなるはずである。予測不可は、寄り道しても、ボトルネックの影響を受ける可能性が高いため、大きな寄り道効果は望めない。

図 4.8 より、Bottleneck bandwidth が小さいほど、Improvement ratio が大きく、Bottleneck bandwidth=2[Mbps]で、予測可は 10 倍強、予測不可は約 8 倍の改善となっていることが確認できる。また、Bottleneck bandwidth が大きくなるにつれて、予測可と予測不可の差が小さくなっていることがわかる。図 4.8 の Bottleneck bandwidth を 2～22[Mbps]としたのは、今後の予定として、ns-2 でのシミュレーションを行うことを考えており、ns-2 において、IEEE802.11g を用いた TCP のスループットが約 22[Mbps]であったからである。

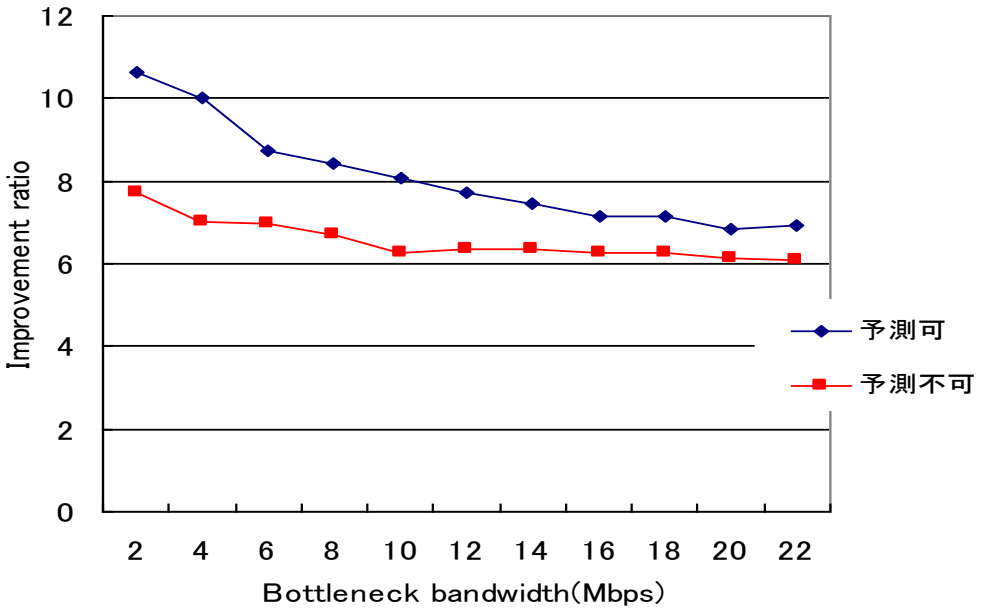


図 4.8　Bottleneck bandwidth と Improvement ratio の関係

第5章　まとめと今後の検討

2 地点間を移動するユーザが、移動時間が多少長くとも通信品質の高い経路を移動するときに、最も効率の良い経路を選択する方法が既に知られている。これに対して、本編では、有線ネットワークにボトルネックが発生するといったより実環境に近い状況を想定し、モデル化を行い、ボトルネックが存在することによる寄り道の通信特性を解明した。

ボトルネックが予測可能である場合、ボトルネックが存在しない場合よりも約 1.5 倍～2 倍、寄り道効果が大きくなる。また、ボトルネックを予測できる場合は、予測できない場合よりも約 1.4 倍～1.7 倍、寄り道効果が大きくなることが確認できた。

今後は、提案評価モデルにおける ns-2 によるシミュレーションを行い、どの程度の寄り道効果が得られるかを試行する必要がある。また、提案評価モデル以外にも様々な状況が考えられる。ボトルネックの発生箇所がダイナミックに変化するといったさらに実環境に近いモデルやユーザではなく AP がユーザに近づいてくるモデルなどが例として挙げられる。

実環境を考慮するならば、ユーザによって必要な通信量が異なることを想定し、ある総転送量を得るにはどれだけの寄り道をすればよいかといった評価を行うべきである。また、基地局をランダムに配置するのではなく、現実のマップを想定した寄り道の特性評価も行う必要があると考えられる。評価対象としては、総転送量だけでなく、電力消費も考慮した評価も行う必要があるだろう。

謝辞

本論文の作成にあたり日頃より助言、御指導を頂いた甲藤二郎教授に深く感謝致します。また、研究を進める上で貴重なアドバイス、ご協力をして頂いた小倉一峰先輩、日本電気株式会社(NEC)の村瀬勉様、本吉 彦様に深く感謝致します。そして、研究だけでなく様々な面においてお世話になった甲藤研究室の皆様に深く御礼申し上げます。

2012 年 2 月 6 日

園田　和秀

参考文献

[1] V. Jacobson, “Congestion avoidance and control,” in Proc. ACM SIGCOMM, Stanford, CA, Aug. 1988.

[2] 兼子和巳, “高速回線のための TCP 輻輳制御方式”,早稲田大学修士論文, 2007

[3] R Braden. Requirements for Internet Hosts - communication Layers. Oct 1989. RFC 1122

[4] W. Richard Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and FastRecovery Algorithms” Jan. 1997. RFC 2001.

[5] Janey C, Hoe, “Improving the Start-up Behavior of a Congestion Control Scheme for TCP” proceedings of ACM SIGCOMM ’96 Aug. 1996.

[6] L. Brakmo, S. O'Malley, and L. Peterson. “TCP Vegas: New techniques for congestion detection and avoidance”. In Proceedings of the SIGCOMM '94 Symposium (Aug. 1994)pages 24-35.

[7] L. Ding, X. Wang, Y. Xu, W. Zhang, Y. Liu, “Improve throughput of TCP-Vegas in multihop ad hoc networks”, IEEE ICC 2008, May 2008.

[8] I. Rhee and L. Xu, “CUBIC: A New TCP-Friendly High-speed TCP Variant”, in Proc. of PFLDnet 2005.

[9] L. Xu, K. Harfoush, I. Rhee, “Binary Increase Congestion Control for Fast, Long Distance Networks”, in Proc. of INFOCOM 2004.

[10] 根本洋平, “CUBIC-TCP と Hybrid-TCP の各種特性比較実験”, 早稲田大学学士論文, 2010

[11] Kun Tan Jingmin Song, Qian zhang, Murari Sridharan, “A Compound TCP Approach For High-Speed and Long Distance Networks”, RFLDnet 2006, Fev 2006.

[12] K.Kaneko, T.Fujikawa, Z.Su and J.Katto, " TCP-Fusion: A Hybrid Congestion Control Algorithm for High-speed Networks", PFLDNet 2007, Feb.2007.

[13] K.Ogura, Z.Su and J.Katto, "Congestion Control with Two Fair Allocation Modes to Achieve RTT-Fairness", ICMU 2010, Apr.2010.

[14] 守倉正博,久保田周治,”802.11 高速無線 LAN 教科書”

[15] Qiang Ni, Lamia Romdhani, Thierry Turletti,” A Survey of QoS Enhancements for IEEE 802.11 Wireless LAN”,Journal of Wireless Communications and Mobile Computing, Wiley. 2004: Volume 4, Issue 5: pp.547-566.

[16] IEEE Standard for Information technology - Telecommunications and information

exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), IEEE Std 802.11e-2005.

[17] B. A. Hirantha Sithira Abeysekera, 松田崇弘, 滝根哲哉, "IEEE 802.11e 無線 LAN における動的ウインド制御",IEICE,2008.

[18] Jitendera Pahdye, Sally Floyd, "On inferring TCP behavior", ACM SIGCOMM'01, August 27-31, 2001.

[19] J.Padhye and S.Floyd. The TBIT Web Page. http://www.aciri.org/tbit/.

S. Feyzabadi and J. Schonwalder, "Identifying TCP congestion control algorithms using active probing," in *Passive and Active Measurement Conference (PAM), Poster*, Switzerland, April 2010.

[20] S. Feyzabadi and J. Schonwalder, "Identifying TCP congestion control algorithms using active probing," in *Passive and Active Measurement Conference (PAM), Poster*, Switzerland, April 2010.

[21] P. Yang, W. Luo, and L. Xu, "TCP Congestion Avoidance Algorithm Identification," IEEE ICDCS2011

[22] Junpei OSHIO, Shingo ATA, Ikuo OKA, "Identification of Different TCP Versions Based on Cluster Analysis", in Proceedings of IEEE 18th International Conference on Computer Communications and Networks (ICCCN 2009), pp.1-6, San Francisco CA, August 2009.

[23] Srinivas Shakkottai, R.Srikant, Nevil Brownlee, Andre Broido, kc claffy, "The RTT Distribution of TCP Flows in the and its Impact on TCP-based Flow Control", CAIDA Technical Report number tr-2004-02.

[24]"The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/ .

[25] "the madwifi project" madwifi-project.org.

[26] Vikram Chandrasekhar, Jeffrey G. Andrews, and Alan Gatherer, "Femto cell Networks: A Survey," IEEE Communications Magazine, Vol.46, No.9, pp. 59-67, Sep.2008

[27] Mitola, J., III and Maguire, G.Q., Jr., "Cognitive radio: making software radios more personal," Personal Communications, IEEE, Volume 6, Issue 4, Aug. 1999

[28] 首藤 裕一, 本吉 彦, 村瀬 勉, 増澤 利光, "無線モバイルユーザのための最適「寄り道」経路の特性," 電子情報通信学会 RCS2009-269, 2010-03

[29] Gen Motoyoshi, Yuichi Sudo, Tutomu Murase and Toshimitsu Masuzawa, "Advantages of Optimal Longcut Route for Wireless Mobile Users," IEEE International Conference on Communications ICC 2011

[30] Canfeng Chen, Jian Ma Ke Yu, “Designing Energy-Efficient Wireless Sensor Networks with Mobile Sinks,” WSW’ 06 at Sensys ‘ 06,2006
[31] Z. Maria Wang, Stefano Basagni, Emanuel Melachrinoudis, Chiara Petrioli, “Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime," Proceedings of the 38th Hawaii International Conference on System Sciences 2005
[32] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” IEEE J.Sel. Areas Commun., vol.14, no.7, pp.1228.1334, jul. 1996
R.Guerin, A. Orda, and D. Williams, “QoS routing mechanisms and OSPF extensions,” IETF RFC 2676, 1999
[33] 寒川 知生, 吉野 信, 新熊 亮一, 高橋 達郎, “電波環境に基づくユーザ位置制御のための効用とコストのモデル化とその応用,” 電子情報通信学会論文誌 B, vol.J90-B, no.12, pp.1263-1273,Dec.2007
[34] 吉野 信, 寒川 知生, 新熊 亮一, 佐々木 純, 高橋 達郎, “電波環境に基づくポイント誘導型ユーザ位置制御の設計,” 電子情報通信学会総合大会, B-15-9, Mar. 2007
[35] Lenin Ravindranath, Calvin Newport, Hari Balakrishnan, and Sam Madden, “Improving Wireless Network Performance Using Sensor Hints,” Proceeding NSDI 2011 Proceedings of the 8th USENIX conference on Networked systems design and implementation

発表文献リスト

[1] 園田 和秀, 飯窪 尚也, 甲藤 二郎, “無線 LAN 環境における 802.11e とウィンドウ制御を併用した TCP-Vegas の特性改善”, 電子情報通信学会 総合大会, B-6-145, 2009 年 3 月.
[2] 園田 和秀, 飯窪 尚也, 甲藤 二郎, “無線 LAN 環境における 802.11e とウィンドウ制御を併用した TCP-Vegas の特性改善”, 電子情報通信学会技術研究報告, vol.109, no.448, NS2009-179, pp.101-106, 2009 年 3 月.
[3] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP バージョン識別を想定した優先アクセス制御の一検討”, 電子情報通信学会 ソサイエティ大会, B-6-41, 2010 年 9 月.
[4] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP 差別化のための TCP バージョン識別手法”, 電子情報通信学会 総合大会, B-6-29, 2011 年 3 月.
[5] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP 差別化のための TCP バージョン識別手法”, 電子情報通信学会技術研究報告, vol.110, no.448, NS2010-272, pp. 615-620, 2011 年 3 月.
[6] 園田 和秀, 本吉 彦, 村瀬 勉, 甲藤 二郎, “確率的にリソースが変動する場合の最適寄り道経路特性”, 電子情報通信学会技術研究報告, vol.111, no.202, CQ2011-34, pp.19-24, 2011 年 9 月.
[7] 園田 和秀, 本吉 彦, 村瀬 勉, 甲藤 二郎, “確率的に変動するリソースを考慮した最適寄り道経路の特性評価”, 電子情報通信学会 ソサイエティ大会, B-6-48, 2011 年 9 月.
[8] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP バージョン識別と EDCA 制御を併用した TCP 差別化手法”, 電子情報通信学会 総合大会, 2012 年 3 月, (投稿中).
[9] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP バージョン識別と EDCA 制御を併用した TCP 差別化手法”, 電子情報通信学会技術研究報告, 2012 年 3 月, (投稿中).
[10] 園田 和秀, 小倉 一峰, 甲藤 二郎, “TCP バージョン識別と EDCA 制御を併用した TCP 差別化手法”, 電子情報通信学会論文誌,(投稿予定)