修 士 論 文

利用可能帯域を意識した適応転送制御に関する研究

指導教官　日比野靖 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

西田 悦雄

1999 年 8 月 13 日

目 次

第 1 章

はじめに

1.1　本研究の背景と目的

コンピュータネットワークを用いてマルチメディア通信を行うときには、転送が行われるデータ量が大きいことに加えて、リアルタイム性を求められているため、マルチメディア通信を円滑に行うためには、通信に必要な帯域を確保、保証することで通信を行う形態が望ましいとされている。

しかしながら、物理的な通信回線上の帯域には制限があり、制限以上の帯域を必要とした場合には通信回線上での送信待ち列のあふれが生じ、ネットワーク上の輻輳を招くことになり、最悪の場合では通信が行えない状態に陥ってしまうことになる。

回線上に輻輳が生じ通信が行えない状態を抑制するため、物理的な通信帯域を増強することや、通信網での輻輳制御などで対処している場合が多くある。しかし、輻輳が起きていない時、通信可能なときに通信可能な帯域を用いて転送を行なうことができれば、転送はできるはずである。また、通信回線上の通信可能な帯域を把握し、その通信可能帯域での転送ができるのであれば過剰なトラヒックを通信回線上に流すことはない。

利用している通信経路上にあるルータの送信待ち列長を測定することで通信回線上のトラヒック状況を推定でき、トラヒック状況を把握することができる。さらに、その通信回線上の余剰帯域 (以下、利用可能帯域という) を検出でき、この情報を得て利用可能帯域に見合った発呼制御を行えば、利用している通信回線上での転送をスムーズに行え、結果的に通信回線上の輻輳抑制にも役立つ。

これによる利点を示す。

- 利用可能帯域を検出することで能動的にトラヒックを呼発生源で制御が行うため、必要以上に通信回線上へトラヒックを生成しない。

- 利用可能な帯域を用いて転送を行うので、目的地への呼の伝搬遅延時間を押えることができる。

本論文では、通信路上にあるルータの出力ポート待ち列の待ち列長を測定し、出力ポートの平均待ち列長をもとに接続されている回線上のトラヒック状況の推定を行うトラヒック測定機構と、トラヒック発生源において、前にトラヒックトラヒック測定の結果から回線上の利用可能帯域の算出を行い、利用可能帯域に合わせた呼発生を行う呼発生制御法について論ずる。具体的には制御の有無によっての呼の伝播遅延時間に関して比較を行い、シミュレーションによって利用可能帯域に適応させた呼発生制御における有効性を示す。

1.2 本論文の構成

本論文の構成は次の通りである。

第 1 章 本研究の背景と目的および本論文の構成を述べる。

第 2 章 基本機構である、トラヒック測定機構と発呼制御機構について述べ、帯域割り当てのポリシーについて考察を行う。

第 3 章 システムモデルについて述べる。

第 4 章 シミュレーション実験によって得られた結果をもとに、呼の伝播遅延時間の短縮について評価し考察を行う。

第 5 章 本研究での成果と今後の課題について述べる。

第 2 章

基本機構

本論文で提案するシステムは、トラヒック測定機構と発呼制御機構が基本となる。この章では、それぞれの機構についての基本的な事項について述べる。さらに、利用可能帯域の割り当て方法についても考察する。

2.1 発呼制御による輻輳回避

提案の基本は、輻輳をどのように回避させるかではなく、輻輳を起こさないような通信トラヒック発生にするかである。

輻輳が生じた通信網においては、トラヒックの集中による輻輳を解消する方法は発呼を止めること以外方法はない。これはトラヒック理論 [1] からも明らかなことである。一度発呼されてしまった呼を呼び戻すことはできないので、トラヒックを制御することができるのは発呼のみだからである。

この発呼制御を行うためには、通信回線上のトラヒックの増大をできる限り早く検出することが必要となるので、トラヒックの測定が必要となる。

そのために本提案では 2 つの機構を用いる。

1. ルータにおける通信回線トラヒックの測定機構

   ルータの出力ポートの瞬時の待ち列長の測定を行い、その待ち列における平均待ち列長を推定を行い、その結果を接続されている全セグメントにブロードキャストによって、フィードバックさせる。この推定値は、接続されている回線の利用可能帯域の算出とその利用可能帯域に適応した発呼に用いる。

2. 利用可能な帯域に合わせた発呼制御機構

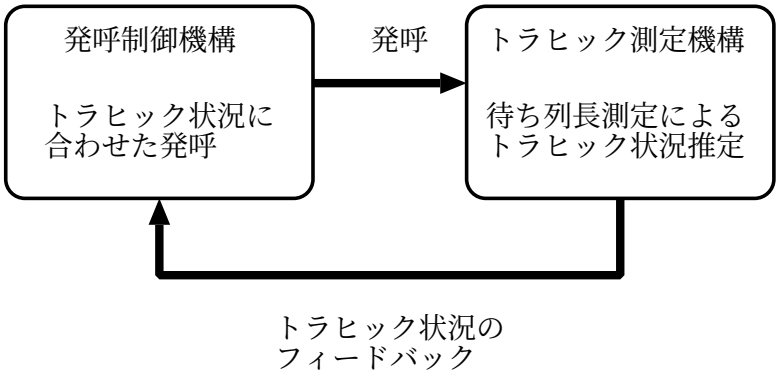

図 2.1: **トラヒック測定機構と発呼制御機構**

ルータからフィードバックされてくる平均待ち列長から接続されている回線使用率ρを算出した後、使用率ρから利用可能帯域の算出を行い、それに適応した発呼制御を行う。

2.2 トラヒックの測定

通信回線上のトラヒック状況は直接使用率ρが求められないが、ルータの出力ポートの待ち行列長により判断が可能である。ルータでの出力ポートの待ち列長が長ければ、回線のサービス率μに比較して到着する呼が多いことであり、そのルータでの回線使用率ρは高いということになる。

通信回線上のトラヒック状況をその回線上のルータのある時刻 t_0における瞬間的な待ち行列長を測定し、ルータの回線使用率を推定する方法は、TCP/IP でのゲートウェイの輻輳回避方法として、1993 年カルフォルニア大学 Lawrence Berkeley Laboratory, Sally Floyd と Van Jacobson によって考案された **Random Early Detection**[2] がある。次節で **Random Early Detection** を簡単にふれる。

2.2.1 Random Early Detection の概要

従来のルータでは、出力ポートの待ち列が溢れた (輻輳が生じている) とき到着した呼に対しては、末尾廃棄 (tail dropping) を行う。この末尾廃棄が複数の TCP コネクションからの呼に対して行われると、これらのコネクションは同時に Window size を小さくし、結果的にルータのスループットを低下させてしまう。この現象は **Global synchronization**

として知られている。

これら Global synchronization の問題と TCP による輻輳制御が通信遅延を生じさせるという問題を解決するために提案された方法が、**Random Early Detection**[2](以下 RED と表記) である。

RED は、end-to-end 間に存在するルータの出力ポートの待ち列長を測定し、その平均待ち列長を用いて、予め決められている最小閾値と最大閾値によって、輻輳の兆候が現れるとデータグラムの廃棄を行う代わりに、TCP の Windows Size を下げるようデータグラムにマーキングを施して、ルータの出力ポートの送信待ち列が溢れてしまう前に、明示的に TCP の WindowSize を下げる要求を行う。

RED の特徴としてあげられるのは、

- 出力ポートの最大待ち列長を越えてデータグラムが末尾廃棄されてしまう輻輳が生じる前に、長期間にわたる輻輳の兆候を発見する。
- 短期間のバースト的トラヒックや過渡的な輻輳は許容する。
- マーキングされるデータグラムの確率をそのデータグラムの属するコネクションがルータにおける使用帯域に比例する仕組みを用いて Global synchronization を防止する。

である。

RED での出力ポートでの待ち列長は、測定待ち列長 Lq_{obs}と加重平均の式で求められる平均待ち列長 Lq_{avg}を用いて制御を行う。

$$Lq_{avg}(t) = (1-w)Lq_{avg}(t-1) + wLq_{obs} \quad w : \text{加重率} \tag{2.1}$$

このときの加重率 wの値は 0.0001 以上に設定するように推奨されている。また RED では測定した待ち列長が 0 の場合には以下の式で平均待ち列長 Lq_{avg}を算出している。

$$m = f(time - q\_time)$$
$$Lq_{avg}(t) = (1-w)^m Lq_{avg}(t-1)$$

本研究においては、RED におけるトラヒック測定の一部を用いる。

2.3 トラヒック測定機構

発呼制御を行うためのトラヒック状況の把握は重要である。特に、トラヒック変動の検出の応答が最も重要なファクタである。この節では、待ち列のモデルとトラヒック測定機構での応答性に関するパラメータについて述べる。

2.3.1 待ち列のサービス

待ち行列モデル $M/M/1$ と仮定すると、待ち列モデルは図 (2.2) である。平衡状態であるとしたときの条件は

$$\rho < 1 \tag{2.2}$$

である。待ち行列での使用率ρと系内数 L のとの関係式は待ち行列理論 [3][4] から式 (2.3) となる。

$$L = \frac{\rho}{1-\rho} = F(\rho) \tag{2.3}$$

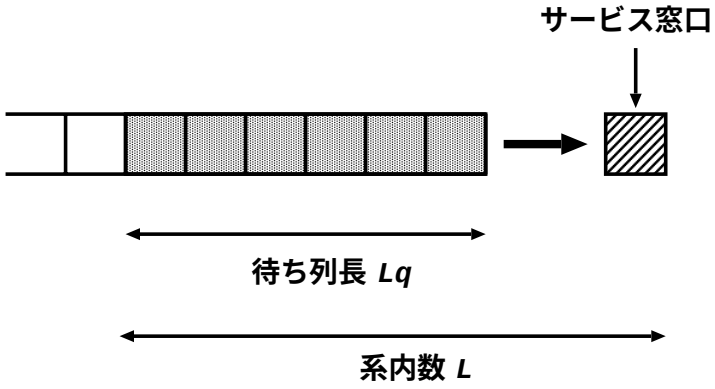

図 2.2: **待ち列モデル**

しかし、実際に測定できるのは、待ち列長 Lqである。待ち列長 Lq は系内数から 1 を引いた $Lq = L - 1$ ではない。サービスを受けている確率ρを系内数 L から引く必要がある。

$$Lq = L - \rho \tag{2.4}$$

そこで使用率ρと待ち列長 Lqの関係式は、式 (2.5) となる。

$$
\begin{aligned}
Lq &= \frac{\rho}{1-\rho} - \rho \\
&= \frac{\rho^2}{1-\rho}
\end{aligned}
\tag{2.5}
$$

使用率ρと待ち列 Lqは関係式は式 (2.5) になり、そのグラフは図 2.3になる。

図 2.3と式 (2.5) から、使用率ρが 1 に漸近するにしたがって待ち列長は急速に増加し、$\rho = 1$ のとき待ち列長 Lqは無限大になることが分かる。

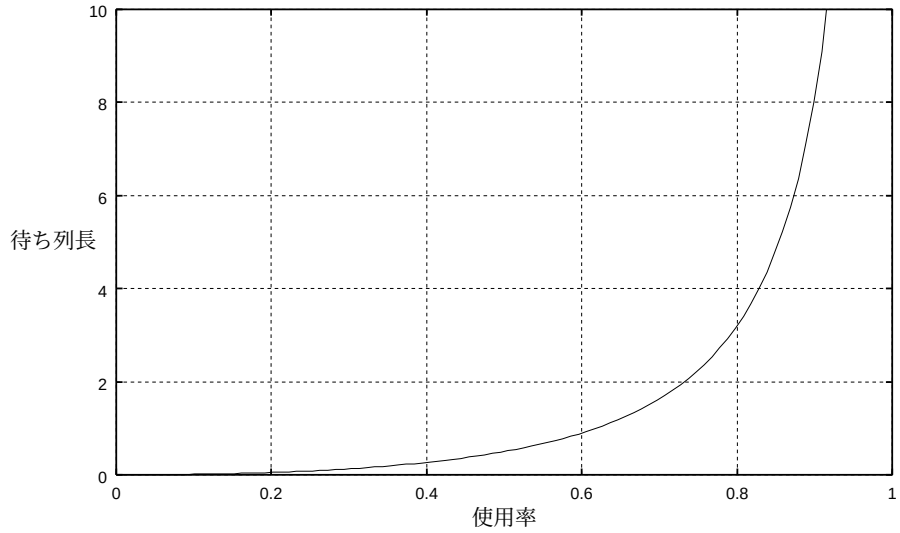

図 2.3: **使用率ρと待ち列長の関係**　($M/M/1$ モデル)

実際にルータで測定できるは回線使用率ρではなく、出力ポートの待ち列長 Lqであるから、式 (2.5) から待ち列長 Lqから使用率ρは $F(\rho)$ の逆関数 $F(\rho)^{-1}$で求めることができる。

$$
\rho = F^{-1}(Lq) \tag{2.6}
$$

でありρについて解くと

$$
\rho = \frac{-Lq + \sqrt{Lq^2 + 4Lq}}{2} \tag{2.7}
$$

となる。

2.3.2 平均待ち列長の加重率

ルータの出力ポートの待ち列から使用率ρを求めるためには、測定時刻 t における待ち列長 $Lq_{obs}(t)$ だけでは、瞬間的な待ち列長である。バースト的なトラヒックや過渡的な輻輳状態においても待ち列長は長くなり、長期的な兆候を検出するには困難である。

したがって、RED[2] で用いられているように、制御に用いるための待ち列長は、平均待ち列長 Lq_{obs}を用い、平均待ち列長の求め方も単純平均ではなく、加重平均で平均待ち列長 Lq_{avg}で長期的に安定した平均待ち列長を求める (式 (2.8))。

$$Lq_{avg}(t) = (1 - w)Lq_{avg}(t - 1) + wLq_{obs}(t) \tag{2.8}$$

加重率 wが小さい場合には、測定した時点での待ち行列長 Lq_{obs}が平均待ち列長 Lq_{avg}に与える影響は少なくなり、長期的なバースト的トラヒックあるいは過渡的輻輳の発生が生じても、平均待ち列長 Lq_{arg}の大きさは急激に増大することはない。逆に加重率 wが大きい場合には、平均待ち列長 Lq_{arg}に与える影響は大きくなり、短期的な変動に対する平均待ち列長 Lq_{arg}の値の大きさは大きくなり、平均待ち列長 Lq_{arg}に反映されることになる。

加重率 wは小さくすると、長期的な変動をするトラヒックに対しての変動を推定するには都合がよいが、短期的な変動に追従できなくなることがある。加重率 wを大きくすると、短期的な変動をするトラヒックに対しての変動を推定するにはよいが、長期的な推測をする場合には変動の影響を及ぼしてしまうトレード オフが存在する。

本研究で用いる加重率 wは、変動に対する応答性と目標値への推測値の収束時間を両立する加重率 wをシミュレーション実験で得た結果より $w = 0.0005$ とした。

2.3.3 測定機構におけるシミュレーション

測定機構シミュレーションは、ルータの出力ポートに対して到着する呼のトラヒックを与えて、そのトラヒックを出力ポートの待ち列長から推定される平均待ち列長 Lq_{avg}の値と与えたトラヒックとの値の差を比較し、測定機構のパラメータと測定値からの推定値の応答を確認する。

さらに、出力ポートの待ち列長の測定からの平均待ち列長 Lq_{avg}の推定値は平均待ち列長 Lq_{avg}推定の式での加重率 wと測定時間間隔に依存するので、シミュレーションに用いるためのパラメータの決定に用いたことは先の 2.3.2 節「平均待ち列長の加重率」で示した。

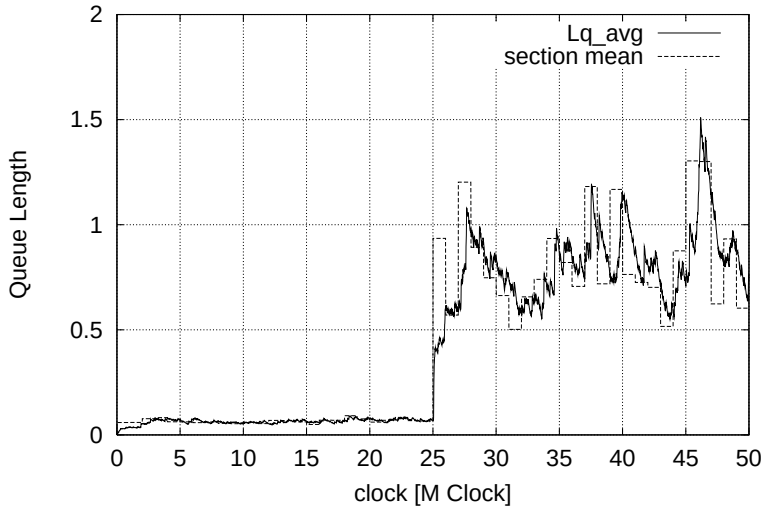

図 2.4: 加重率 $w = 0.0005$ での平均待ち列長 Lq推定値 1

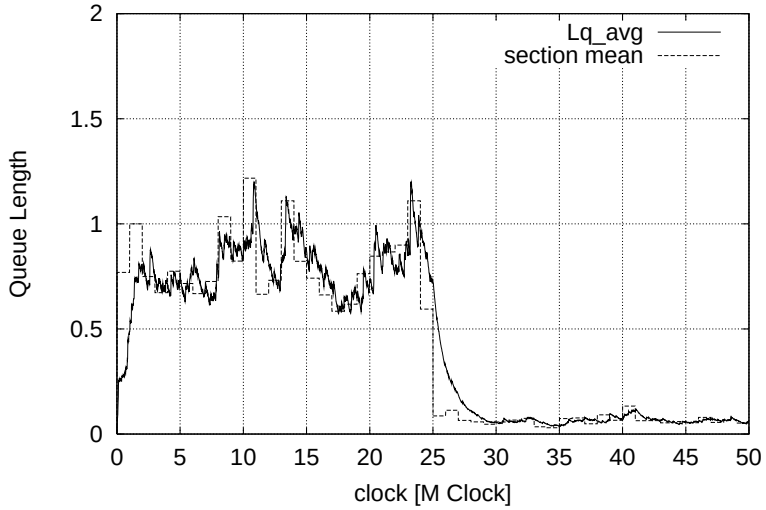

図 2.5: 加重率 $w = 0.0005$ での平均待ち列長 Lq推定値 2

加重率

加重率 wによる平均待ち列長 Lq_{avg}の応答性を確認するので、与えるトラヒック変動はステップ関数を用いた。

加重率 wは RED では 0.0001 以上とあるので、加重率 wは 9 種類に対して発呼率λの変動差を$\lambda = 0.5$ を中心に± 0.05、± 0.1、± 0.2、± 0.3 とし、それぞれ増加傾向と減少傾向の変動を与えて行った。トラヒック測定機構の特性を決定する重要なファクタである応答性を確認する指標として、区間平均待ち列長を同時に表示しその値を目標値とした。

図 2.4は与えたトラヒックが増加傾向の場合、図 2.4は与えたトラヒックが減少傾向の場合である。

式 (2.8) から加重率 wによって、目標値までの応答性が変更されることは明らかである。加重率 wが大きいほど、トラヒック変動に対しての応答性はよくなるが、平均待ち列長 Lq_{avg}に与える影響が大きくなり、平均待ち列長 Lq_{avg}は振動して目標値に収束しない。特に加重率 wが 0.0025 を越えると平均待ち列長 Lq_{avg}は目標値に収束しなくなる。逆に加重率 wが小さい場合には、応答性は悪くなり、目標値である区間平均待ち列長への収束時間は長くなってしまうが、目標値への振幅は小さくなって目標値へ収束する。

先に述べたように、目標値である区間平均待ち列長に収束する時間つまり応答性と、発呼制御に用いるために目標値への収束することの間にはトレードオフが存在することが確認できた。

単位時間当たりの変動量

ある単位時間に対しての変動量が測定機構の平均待ち列長 Lq_{avg}の推定に及ぼす影響を確認を行う。

単位時間の変化率は式 (2.9) のようになる。

$$\text{変動率} = \frac{\Delta(\lambda(t_b) - \lambda(t_a))}{t_b - t_a} \quad a < b \tag{2.9}$$

単位時間の変動量を変更するために、与えるトラヒックの時間変動関数を sigmoid 関数とする。ステップ応答のように急激な変化ではない、緩慢なトラヒック変化に対する応答性と追従性を見る。図 (2.6) と図 (2.6) は与えたトラヒック、図 (2.7) と図 (2.9) は増加傾向の場合、図 (2.10) と図 (2.10) は与えたトラヒック、図 (2.11) から図 (2.13) は減少傾向の場合である。

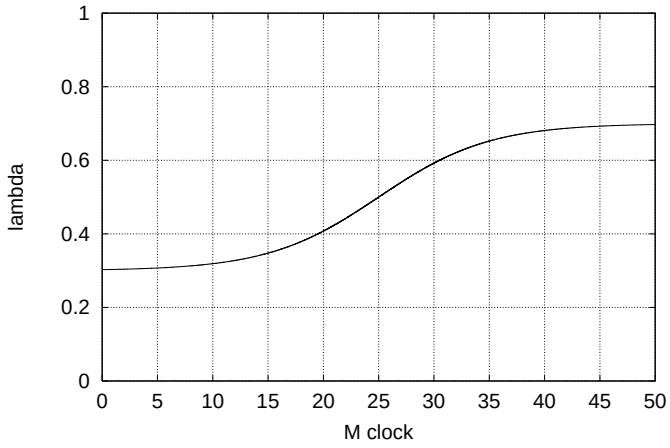

図 2.6: **与えた単位時間の変化率 1**

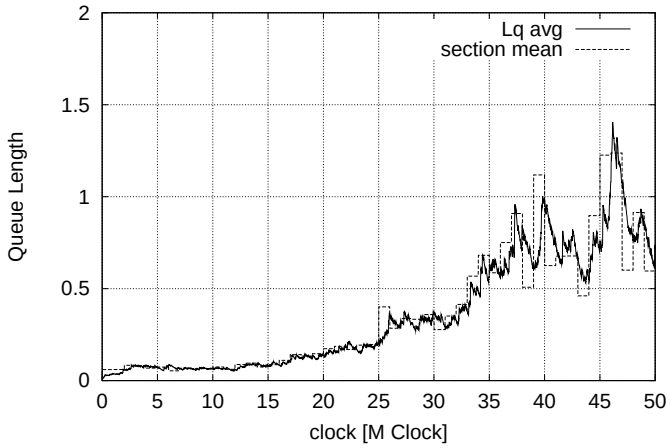

図 2.7: **単位時間の変化率による応答性 1**

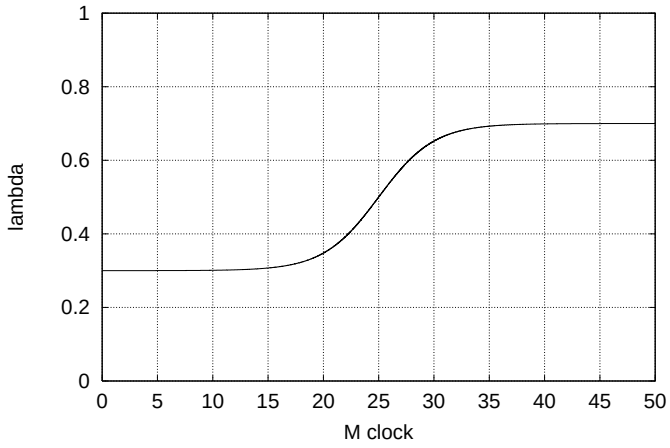

図 2.8: **与えた単位時間の変化率 2**

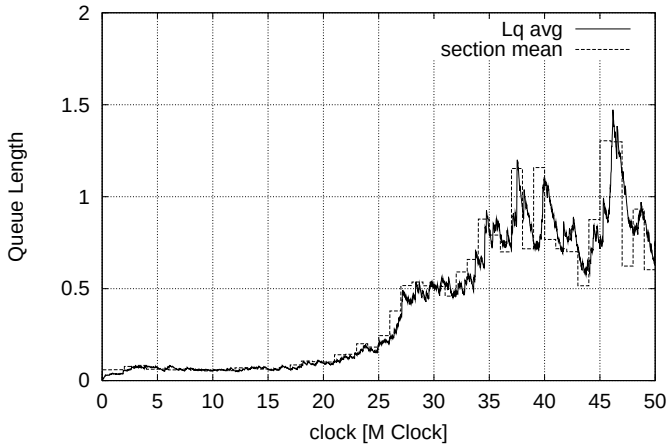

図 2.9: **単位時間の変化率による応答性 2**

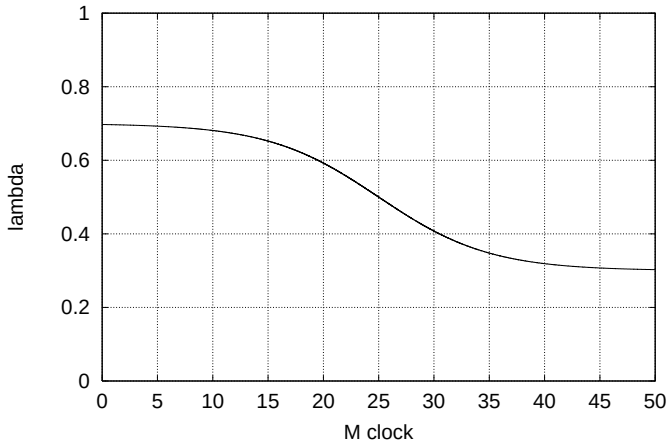

図 2.10: **与えた単位時間の変化率 3**

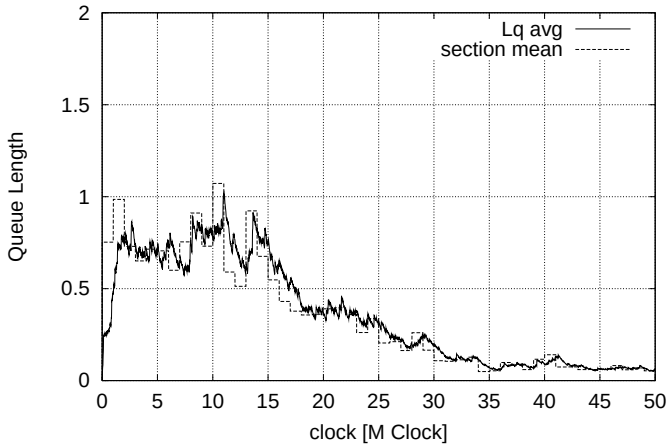

図 2.11: **単位時間の変化率による応答性 3**

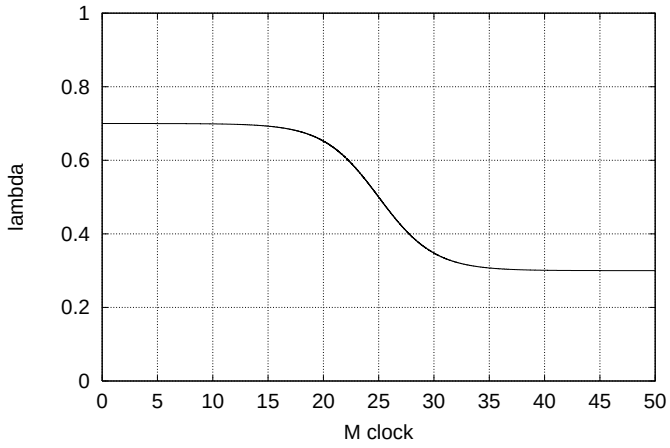

図 2.12: **与えた単位時間の変化率** 4

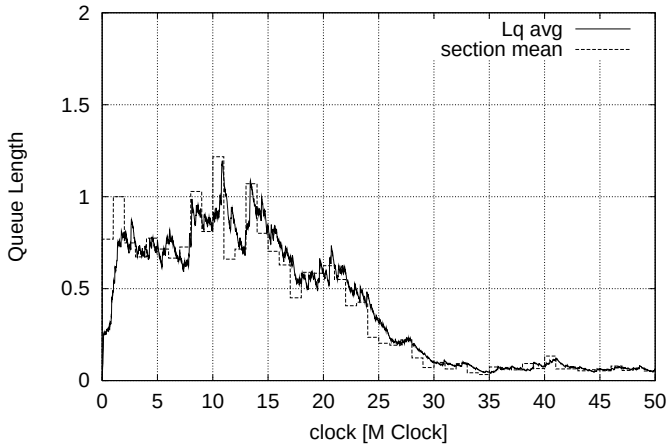

図 2.13: **単位時間の変化率による応答性** 4

2.3.4 測定時間間隔

加重率 wに加えて、測定時間間隔の影響も考慮しなければならない。測定値とルータにかかるトラヒックに追従させるためには、トラヒックの変動を見逃さないことが重要である。

トラヒックの変動が激しい場合に、測定時刻 t_0と次の測定時刻 t_1 の間 Tを短くすることで、短期的なトラヒックの変動に対しても追従できるが、測定時間間隔 Tが小さければ短期的なトラヒックを見逃すことはなくなる。

測定時刻をサンプル点、測定時間間隔をサンプリング間隔、与えられるトラヒックの変動を信号波形として考えると、信号処理で一般的に用いられる標本化定理が使える。

サンプリング間隔は、ナイキスト間隔により最大間隔 $T = 1/2f_m$で行えばよいので、回線伝送時間の 1/2 を測定時間間隔とする (図 2.14)。

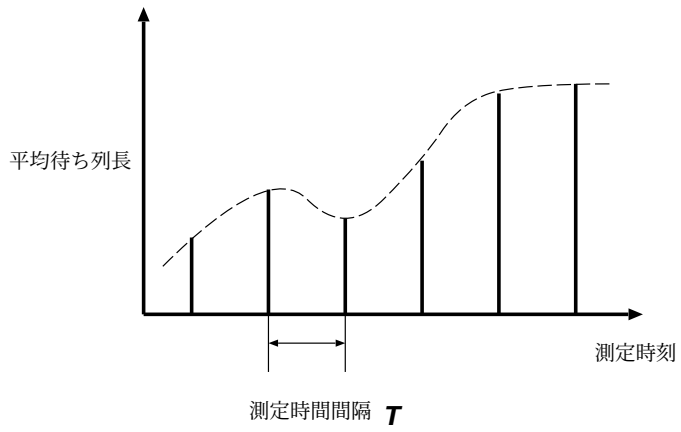

図 2.14: **測定時間間隔**

都合のよいことに、測定した出力ポートの待ち列長を元に算出された平均待ち列長を標本とすることで、遮断周波数ω_mの低域フィルタ (Low Path Filter) を通した信号処理となり、短期的な変動はノイズとして扱われ、短期的な変動を押えた波形を得ることができる。測定機構の影響を及ぼす加重率 wと測定時間間隔の影響について述べたが測定に関しての限界が存在している。次の節で測定に関する限界について述べる。

2.3.5 測定機構の限界

加重平均による平均待ち列長を求めることと、sender がよい測定時間間隔による標本化により測定機構は、2 つの信号フィルターを通ることになる (図 2.15)。

もし、測定時間間隔を前節のようにナイキスト間隔とせず、より長い測定時間間隔とした場合は、このフィルターはどちらも低域フィルターの役割を果たすので、短期的な変動に対しては許容することになる。この 2 つのフィルターの遮断周波数とトラヒックの変化を考えてみると、測定時間間隔による遮断周波数とトラヒック変化が漸近しているときには、検出ができないことになるので測定機構の限界になる。

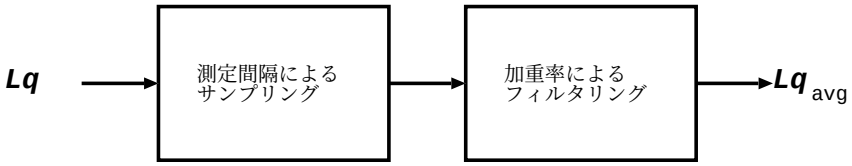

図 2.15: 測定機構のフィルタ

2.3.6 平均待ち列長情報のフィードバック

上で述べた測定機構によって得られた情報は、ルータの出力ポートに接続されているすべてのネットワークセグメントにブロードキャスト方式でフィードバックを行う。これは、同一ルータに接続されている発呼制御を行っているホストが複数ある場合に有効である。

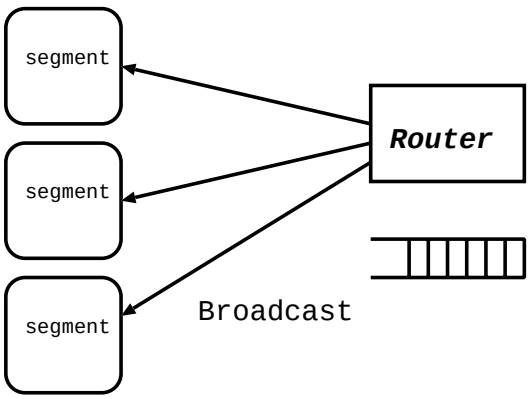

図 2.16: ブロードキャストによるフィードバック

フィードバック時もフィードバック情報の更新する時間間隔が短ければ、逆向きのトラヒックを増大させてしまう。そこで、フィードバックを行う間隔は一定にせず、トラヒッ

クの変動がある閾値を越えたときに情報をフィードバックさせれば、トラヒックを増大させることを抑制できる。end-to-end のパスに複数のルータが存在も同様である。

ルータのトラヒック情報にも伝播遅延が存在するので、発呼制御を行う時刻とフィードバックされた情報には時間差が生じることになる。フィードバックされるルータの出力ポートの平均待ち列長 Lq_{avg}は、短期的な状況ではなく中長期的な状況を推定しているため、発呼時におけるフィードバック情報の時間差の影響はそれほどないと思われる。

2.4 発呼制御機構

トラヒック測定機構により推定された平均待ち列長 Lq_{avg}を用いて、利用可能帯域の算出、利用可能帯域に適応した呼発生レートでの呼発生制御を行う。

発呼制御を行うことで、通信回線上に輻輳を生じないようにできる。また通信回線上に輻輳が生じた場合でも、輻輳している時間を最小に抑えることできる。それにより、通信回線の使用率ρを一定に保つようになり、通信回線の有効な利用ができることと、発生した呼の目的地までの伝播遅延時間を短くできることがあげられる。

発呼制御における過程は次の通りである。

1. 使用率ρの算出
2. 利用可能帯域の算出
3. 利用可能帯域による発呼量の決定

これらについて次節以降で述べる。

2.4.1 使用率ρの算出

呼発生源では、end-to-end 間の通進路上のルータからのブロードキャストされてくる出力ポート平均待ち列長 Lqから回線利用率ρを求め、接続されている回線のサービス率μから物理的に空いているサービス可能率λ_{avg}の算出を行う。

回線使用率ρについては、式 (2.7) で求めることができる。

$$\rho = \frac{-Lq + \sqrt{Lq^2 + 4Lq}}{2} \tag{2.10}$$

式 (2.5) において、回線使用率 $rho = 1$ の場合、平均待ち列長: q_{avg}と使用率ρの関係式 (2.5) より $Lq = \infty$ となり無限大の待ち列長となり機構を破綻させてしまう。この機構の

破綻を回避するためには、$\rho_{max} < 1$ を満たす必要があり、最大利用率ρ_{max}を設定し制限を行うことで、この問題から回避を行う。

2.4.2 利用可能帯域の算出

利用可能帯域を利用可能発呼率 $lambda$ とし算出する。これは、発生させるパケットを一定と考えると、発呼率λによって制御が可能になるからである。

利用可能発呼率λ_{avl}を求めるには、得られた使用率ρより、現時刻での発呼率$\lambda)cur$を式 (??) によって算出する。ここで注意が必要なのは、算出されているトラヒック情報には、呼発生源のトラヒックも含まれていることである。したがって、呼発生源のトラヒックを差し引く必要である。自らのλ_{self}は式 (2.11) より求まるので、

$$\lambda_{self} = f(\lambda_{avl}) \tag{2.11}$$

$$\lambda_{cur} = \rho\mu - \lambda_{self}(t) \tag{2.12}$$

となる。

利用可能発呼率λ_{avl}は、最大利用可能発呼率λ_{max}から

$$\lambda_{avl} = \rho_{max}\mu - \lambda_{cur} \tag{2.13}$$

で求められる。

2.4.3 発呼制御

さらに得られた利用可能発呼率λ_{avl}に合わせて呼発量の制御を行う。先の平均待ち列長情報のフィードバックの節でも触れたが、フィードバックされてくる待ち列長 Lq_{avg}と、発呼時刻には時間差があるが、フィードバックされる情報であるルータの出力ポートの平均待ち列長 Lq_{avg}には中・長期的な推測であるために、時間差による影響は少ないと考える。

利用可能帯域に合わせた発呼をさせるための発呼率の決定は次のように発呼のタイミング毎に行う。

1. 前回の発呼を行った時の発呼率$\lambda_{avl}(t-1)$ と利用可能発呼率$\lambda_{avl}(t)$ を比較する。

2. 前回の発呼制御値 よりも利用可能帯域が大きい場合には、3. のの処理を行い、直前の発呼制御値の方が利用可能帯域よりも大きいか同じである時には、4. の処理を行う。

3. この状況では、利用可能帯域が増加傾向にあるので、利用可能帯域と再度比較を行い、発呼制御値を利用可能帯域を上回らないように増加させる。

4. この状況では、前回の発呼時刻の利用可能帯域と変更はないか、あるいは利用可能帯域が減少傾向にある。従って、発呼制御値を利用可能帯域を上回らないように減少させる。

ここで重要なのは、利用可能帯域が減少傾向にあるとき、つまり回線上のトラヒックが増大傾向時の制御方法である。利用可能帯域が減少傾向にあるとの発呼率は、できるだけ速やかな減少を行わなければ、発呼制御を行っている自ら発するトラヒックによって回線上のトラヒックを増大させることになるので、発呼制御による応答性は重要になる。

一方、利用可能帯域が増加傾向にある場合には、回線上のトラヒックの観点から考えると、減少傾向の時ほどシビアではない。ルータに対するトラヒックの負荷が低いということなので、トラヒックによる、帯域の有効利用という観点では、利用可能帯域に余裕を与えることになるが、発呼された呼の伝播遅延時間に悪影響を与える要因とはならない。

2.4.4 発呼率変動点付近での offset による制御値微調整

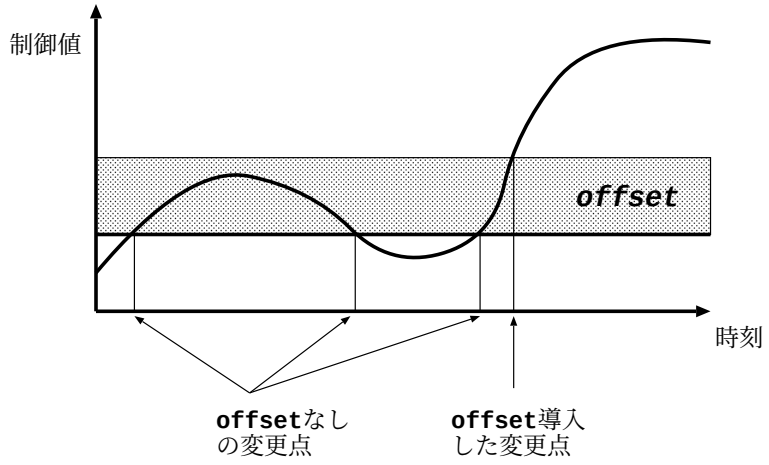

図 2.17: **増加傾向時の offset 導入**

発呼率を変更する点においては、発呼率変更点付近の利用可能帯域の微小な変化による発呼率の変動が考えられる。制御値変更点付近の利用可能帯域の微小な変化によって問題となるのは、その微小な変化によって発呼率が大きく変動することにある。最悪の場合、発呼制御による呼の発生により、回線上のトラヒックを増大させてしまい、発呼率が振動してしまう可能性を含んでいる。不用意な発呼率の変動を抑制し安定させるためには、オフセットを設けて、発呼制御を行うことで、微小な利用可能帯域の変動からの影響を抑えるというものである。オフセット値は、最大利用率ρ_{max}の許容率 (%) を発呼率変動点の値に加減し制御を行う。利用可能帯域が増加傾向にある場合には、offset 値を発呼率変動点に加えて考えることにより、不用意な増加変動を抑制できる (図 2.17)。逆に、利用可能帯域が減少傾向にある場合には、offset 値を発呼率変動点から減じた値を適応することで行う (図 2.18)。

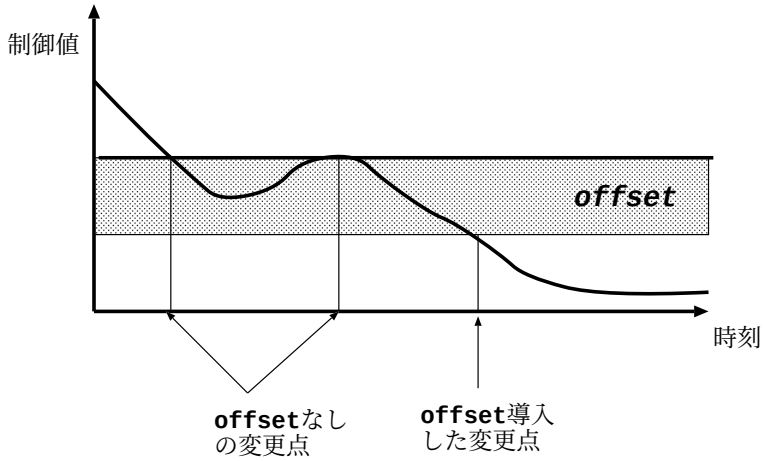

図 2.18: **減少傾向時の offset 導入**

2.5 利用可能帯域の割り当て

単一の発呼制御を行っているアプリケーションが存在する場合と、発呼制御を行っているアプリケーションが存在する場合の、帯域割り当てについて考察しておく。基本的には発呼制御を行っているホストのアプリケーションの性質や優先度といったポリシーによるものである。

表 2.1: 単一発呼制御の場合

利用可能帯域 帯域変動傾向	帯域割り当て
なし	割り当てられない
増加傾向	変動点以上ならば増加
	変動点未満ならば現状維持
減少傾向	変動点未満ならば減少
	変動点以上ならば現状維持

2.5.1 単一アプリケーションでの割り当て

単一のアプリケーションのみが発呼制御を行っている場合には単純である。帯域が制御値変動点でのふるまいは表 1 にある通りである。

利用可能帯域を利用しうるのは、発呼制御を行っているアプリケーションだけなので、算出される利用可能帯域については独占できる状態にあるので、利用可能帯域の変動方向と制御変動点以上、未満で分かれるだけである。

2.5.2 発呼制御アプリケーションが複数の場合

複数の発呼制御アプリケーションが存在するときを考えてみると、各発呼制御アプリケーションにもたらされる情報は、ルータからフィードバックされる出力ポートの平均待ち列長 Lqのみであり、各発呼制御アプリケーションでは、利用可能なλ_{avl}と、自らの発呼率λ_{self} ク値だけが与えられているが、発呼制御を行っているアプリケーションの数は分かり得ない。したがって、算出された利用可能帯域を単一の場合と同様、利用可能帯域を独占しようとすれば、利用可能帯域を複数の発呼制御アプリケーションで争奪することになりかねない。そこで、発呼制御のアプリケーションの要求する帯域が同一の場合と、要求する帯域が異なる場合の 2 つについて考察する。

問題を簡単にするために、以下の考察では発呼制御アプリケーションは A,B として考察を行う。

発呼制御アプリケーションの要求帯域が同一の場合

発呼制御アプリケーションの要求帯域が同一の場合の要求帯域をW_a、利用可能帯域をWとすると、利用可能帯域と要求される帯域との関係には以下の状況が考えられる。

1. $W < W_a$
2. $W_a \leq W < 2W_a$
3. $2W_a \leq W$

また、発呼制御を行っているアプリケーションの性質によっても異なる。

1. 両方のアプリケーションがリアルタイム性を重視する性質の場合
2. 両方のアプリケーションが遅延を許容する性質の場合
3. 一方のアプリケーションがリアルタイム性を重視する性質であり、もう一方が遅延を許容する性質の場合

1. と 2. の場合だと、平等に割り当てを分割させる方法とそれとも優先度をつけるかによる方法が考えられる。平等に分割する方法では、互いに制御値の上下が起こり、制御不能に陥る可能性はないとはいえないので、アプリケーションの性質により優先度を決める方法を考えると、さらに、優先度の付け方にもいくつかの方法がある。

- 先にコネクションを張ったアプリケーションを優先
- 後からコネクションを張ったアプリケーションを優先
- 予め与えられた優先度によって決定

一般的に考えれば、転送を早く終るコネクションを優先させた方がよい。その点から考えると、先にコネクションを張ったアプリケーションの方を優先させることで、転送を終える確率が高いと思われる。

したがって、この場合での利用可能帯域の割り当て方法は、発呼制御アプリケーションの性質による優先度と転送終了の確率を重視した割り当て方法がよいと思われる。

発呼制御アプリケーションの要求帯域が異なるの場合

要求帯域が同一の場合よりも割り当て方法は複雑になるように思われる。発呼制御アプリケーションの要求帯域が異なる場合の要求帯域を各々$W_a, W_b \quad a < b$、利用可能帯域をWとする。

この場合では以下の状況が考えられる。

- $W < W_a$
- $W_a \leq W < W_b$
- $W_b \leq W < W_a + W_b$
- $W_a + W_b \leq W$

また要求帯域が同一の場合と同様に、発呼制御を行っているアプリケーションの性質によっても異なる。

1. A,B がリアルタイム性を重視する性質の場合
2. A,B が遅延を許容する性質の場合
3. A がリアルタイム性を重視する性質であり、B が遅延を許容する性質の場合
4. A が遅延を許容する性質であり、B がリアルタイム性を重視する性質の場合

この発呼制御アプリケーションの性質によって優先度を決定できる 3. と 4. の場合は、要求帯域が同一の場合の優先度と同様であるが、1. と 2. の場合ではさらに考察が必要となる。要求帯域が同一の場合だと、転送終了が早い方を考えたが、要求帯域が低い方を優先させた方が、確率的に考えて転送終了は早いと思われる。なぜならば、転送時間を考えると要求帯域に比例すると推察できる。さらに、利用可能帯域の変動に対しても要求帯域が低い方が適応させやすい。

したがって、この場合での発呼制御アプリケーションの優先度はアプリケーションの性質に加え要求帯域を考慮したポリシーがよいと思われる。

第 3 章

システムモデル

シミュレーションを行うために、システムモデルについて述べる。

3.1 システムモデル概説

シミュレーションモデルとして仮定するのは、図 (3.2) のようなモデルを考える。

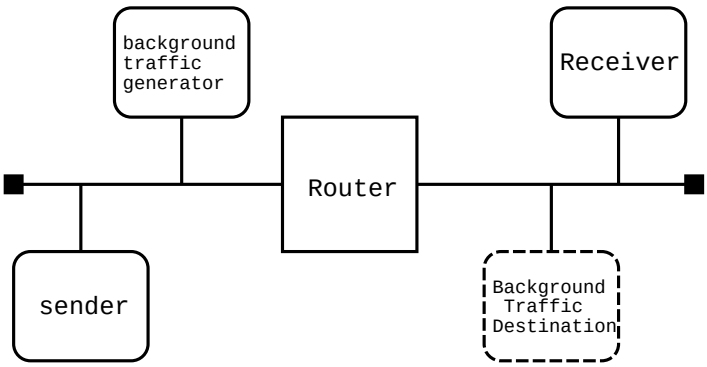

図 3.1: **システムモデル**

セグメント A とセグメント B はルータによって接続されており、セグメント A においては、発呼制御を行う sender と、発呼制御を行う sender 以外のトラヒック (以降バックグラウンドトラヒックと呼ぶ) を発生させる呼源とを置く。また、セグメント B には、sender の目的地である receiver を置き、発呼される呼が receiver に到着するまでの呼の一連の動作をシミュレーションにて確認する。さらに、システム全体のネットワークのト

ポロジーやルータに接続されているリンク数、各リンクの回線速度は既知としたネットワークモデルを考える。

3.2 トラヒックモデル

一般的に、トラヒックモデルは到着分布は指数分布に従う。[5][6]。したがって、バックグラウンドトラヒックの到着分布は指数分布に従うとする。バックグラウンドトラヒックの目的地は、セグメント B 内にある仮想的な host と仮定した。また、パケット長は一定として発生させるものとする。ルータモデルでは、トラヒック測定機構を組み込んだ形態である。

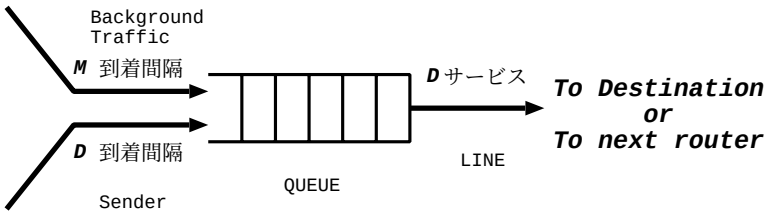

図 3.2: **ルータモデル**

- ルータはコールドスタート
- ルータに到着するすべての呼は一度出力ポートの待ち列の末尾に挿入
- サービスに要する時間は呼長が一定ということからサービス時間も一定
- 呼のヘッダーなどの処理もサービス時間に含まれると仮定
- 出力ポートの待ち列 Lqの測定時間間隔は一定 (サービス時間の 1/2)
- 測定された待ち列長 Lqからの平均待ち列長 Lq_{avg}の情報は遅延時間なしでフィードバックすると仮定。

ある時刻にルータに到着した呼は、無条件に出力ポートの待ち列の末尾に挿入される。接続される通信回線が idle の状態では、待ち列の先頭から呼を取り出し、サービス (通信回線上に転送) を行うと同時に伝送のための時間をタイマーでセットする。回線が busy

の状態では、転送時間のタイマーから減算し、サービス時間を進める。回線も idle 状態で、出力ポートの待ち列に呼が存在しなければ、呼に対するアクションは行わない。

本研究での重要な機構である、出力ポートの待ち列長 Lqについて測定を行う。ここでの測定間隔は、ナイキスト間隔により、回線の転送時間の 1/2 とした。さらに、応答を確認する上で区間平均待ち列長の算出の処理も行う。

ルータモデルにより、システム全体は 2 章で述べた $M/M/1$ モデルから、パケット長を固定することで $M/D/1$ モデルに簡単化した。

$M/D/1$ モデルに簡単化したために 2 章での待ち列長 Lqとρの関係式 (2.7) は次のようになる。

$$Lq = \frac{\rho^2}{2(1-\rho)} \tag{3.1}$$

したがって、ρは

$$\rho = \sqrt{Lq(Lq+2)} - Lq \tag{3.2}$$

となる。

3.3 発呼制御モデル

発呼制御を行う呼の情報源は、画像データとして考えたので、解像度等の変更には 2^n で行われるように、発呼制御に用いる発呼率の変動は、2 の冪乗 2^n とした。さらに、固定長画像データは固定長で転送するものとした。

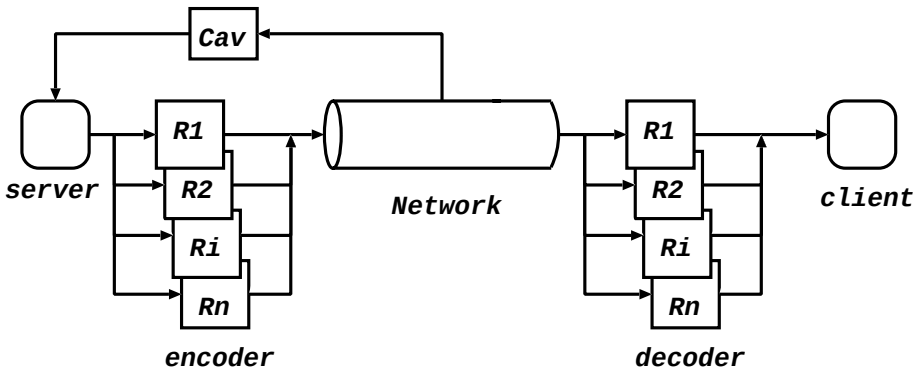

図 3.3: **発呼制御モデルのイメージ**

第 4 章

シミュレーション実験

シミュレーションは、ルータにおいてはコールドスタートで始まる発生からルータを通り、目的地に到着するまでの一連の呼の流れを見るように、疑似的な時刻を刻んでいくイベント駆動方式で行う。

シミュレーションで得られた結果は、呼発生制御機構シミュレーションにおける、呼発生制御の有無による伝播遅延時間の分布と有効性である。

4.1 シミュレーションにおける考察点

本研究での目的は、発呼された呼の伝播遅延時間を短縮することが最終的な目標である。従ってシミュレーション結果の応答性で重要なのは増加傾向の時の場合である。なぜならば、トラヒックが増加傾向にあるときが、帯域利用の効率を悪くし、発呼させた目的地までの伝播遅延時間を増大させるからである。

4.2 バックグラウンド定常トラヒック

バックグラウンドトラヒックを一定としたとき、発呼制御を行った場合と、発呼制御を行わなかった場合を比較する。発呼制御がうまく行われているかどうかを調べるために、まずは発呼制御なしの場合と、発呼制御を行った場合の発呼率λ_{nc} と λ_cとを比較する。

表 4.1 に示すように、発呼制御なしの場合は発呼率上限値 0.8 とバックグラウンドトラヒックλ_{base}との差を固定値として与えている。制御ありの場合、ルータからのフィードバック情報により、発呼可能率λを計算し求めた発呼率 λ_cが安定して得られるかどうかを確認した。

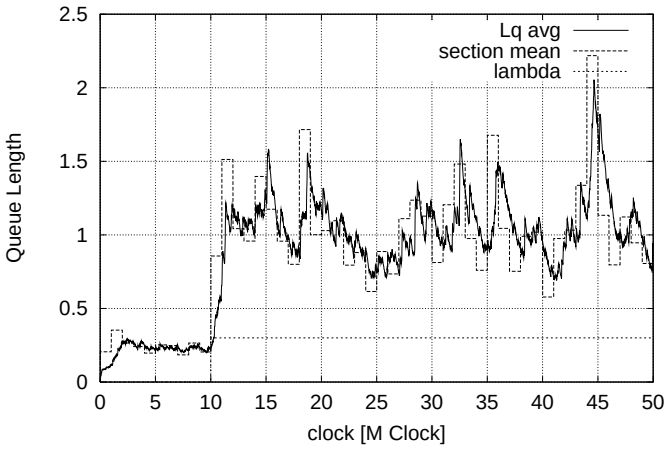

図 4.1: **定常状態制御なし Lq 観測**$\lambda = 0.5$

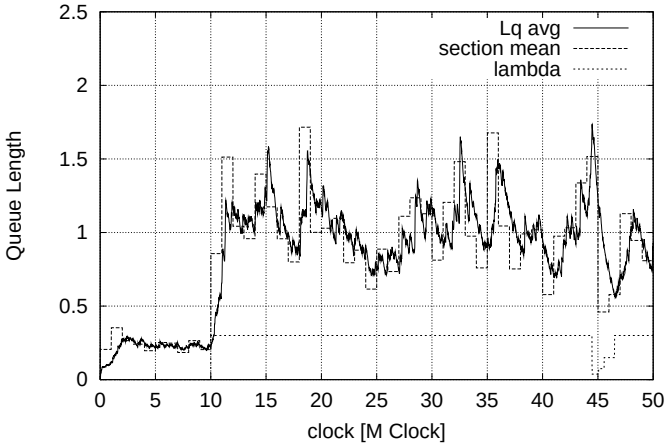

図 4.2: **定常状態制御あり Lq 観測**$\lambda = 0.5$

表 4.1: 発呼時のλの比較

上限値 トラヒックλ	バックグラウンド λ_{nc}	発呼制御なし λ_c	発呼制御 λ_{base} λ_{max}
0.8	0.3	0.5	0.5
0.8	0.5	0.3	0.3
0.8	0.7	0.1	0.1

表 4.1 に示すように、発呼制御ありの場合、発呼可能率λ_cは発呼制御なしの場合とかわらない。詳しい状況をみるために、バックグラウンドトラヒック$\lambda_{base} = 0.5$ の場合について、発呼制御なしと発呼制御ありの平均待ち列長と発呼率の時系列変化を図 4.1,4.2に示す。

40-45 [M clock] の間のベーストラヒックのピークのため、制御なしの場合、sender のトラヒックと重畳し、著しい平均待ち列長の増大が見られる。制御ありの場合、発呼可能率が、その間低く制御され、その結果平均待ち列長の増大が抑えられている。

4.3 バックグラウンド変動トラヒック

与えるバックグラウンドトラヒックに変動を加えてターゲットのトラヒックとなる呼発生制御を確認を行う。実際のネットワークにかかる負荷トラヒックパターンは分からない場合が多くあるが、シミュレーションではトラヒックの変動パターンについては、以下のような時間変動関数で与えることができるトラヒックの変動パターンをモデルとした。変動率の観点から、

- 急激なトラヒックパターン
- 緩やかなトラヒックパターン

を step 関数と sigmoid 関数で時間変動させたシミュレーションを評価、考察する。さらに周期的にトラヒック変動を起こすモデルとして

- 正弦波変動するトラヒックパターン
- 線形のこぎり波変動するトラヒックパターン

についても同様に評価・考察を行う。

それぞれの関数について、増加傾向時の特性と減少傾向時の特性についてシミュレーションを行う。さらに、どの時間変動関数の場合でも、発呼制御についての考察は、増加傾向で行うことにする。また、コールドスタートによる過渡状態の影響を除くために、測定区間は 10[Mclock] から 50[Mclock] までとした。

4.3.1 ステップ型変動

システムの応答性を確認するために適したステップ関数でのシミュレーションについて評価し考察する。

増加傾向

まず、与えるバックグラウンドトラヒックについて図 4.3で示す。

つぎに、発呼側での発呼率λ_cの変動について示す。図 4.4は制御なしの場合、図 4.5 は制御ありの場合である。バックグランドトラヒックは、25 Mclock で λ_{base} は 0.3 から 0.7 にステップ関数状に変化する。ここで、比較する場合にはスケールをそろえるのがよいはずであるが、待ち列長のグラフに関して、40 倍もの差があるのでスケールをそろえると制御なしのグラフの全体が見えないために、あえてスケールを変えた。

制御なしの場合、バックグラウンドトラヒックがステップ関数状に変化した直後から平均待ち列長の著しい増大が見られる。待ち列長の最長は 100 を越えてしまう。

制御ありの場合、発呼開始時の 10Mclock からバックグラウンドトラヒックが急激に変動する 20M clock までの間では待ち列が一時的に増加しており、20Mclock を過ぎると定常状態になる。さらに、バックグラウンドトラヒックがステップ上に変動後は、一時的に待ち列長が増加するが 3Mclock 後には待ち列長は 1 付近にまで減少し、その後も待ち列長 2 を越えることはない。ステップ関数型の増加傾向時での応答は、すばやく応答できている。

制御の効果を詳細に見るために、パケットの発信元からの受信先までの伝播遅延時間の分布を調べる。図 4.6は制御なしの場合、図 4.7は制御ありの場合の 1000clock 毎の伝播遅延時間の度数分布である。この結果を表 4.2にまとめる。

最大伝播遅延時間は、制御ありの場合は、制御なしの場合に比べて、17.7 %に大幅に短縮している。平均伝播遅延時間は 6.6 %に大幅に短縮された。

表 4.2: **ステップ関数増加傾向の度数分布からの平均値の比較**

λ_{base}	最大伝播遅延時間 [度数]	平均値 [clock]	分散	標準偏差
発呼制御なし	141	28,500	178,000	3.85
発呼制御あり	25	1,890	768	0.30

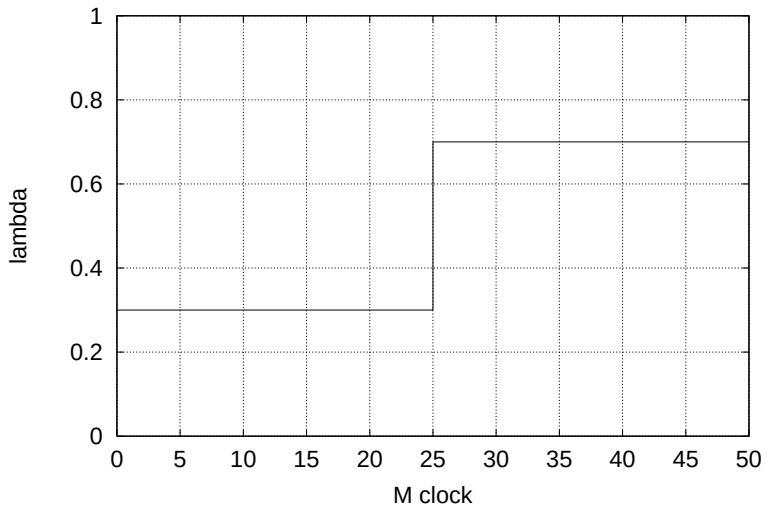

図 4.3: **ステップ関数増加傾向 バックグラウンドトラヒック**

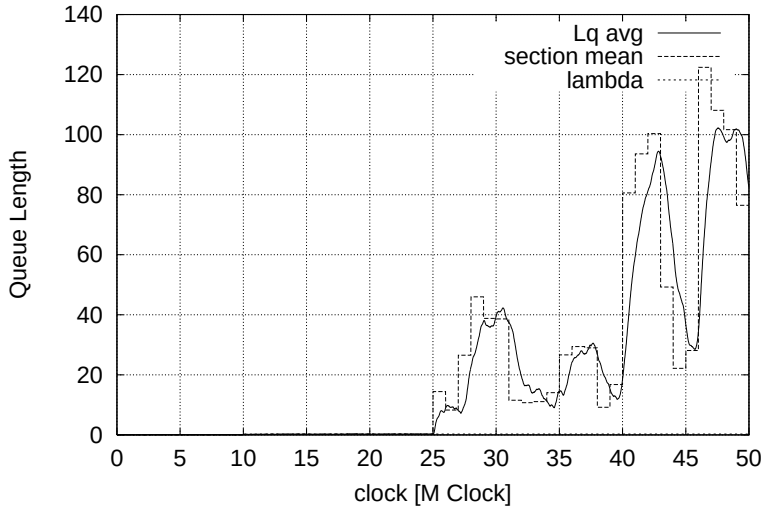

図 4.4: **ステップ関数増加傾向 制御なし Lq 観測**$\lambda_{base} = 0.3 \rightarrow 0.7$

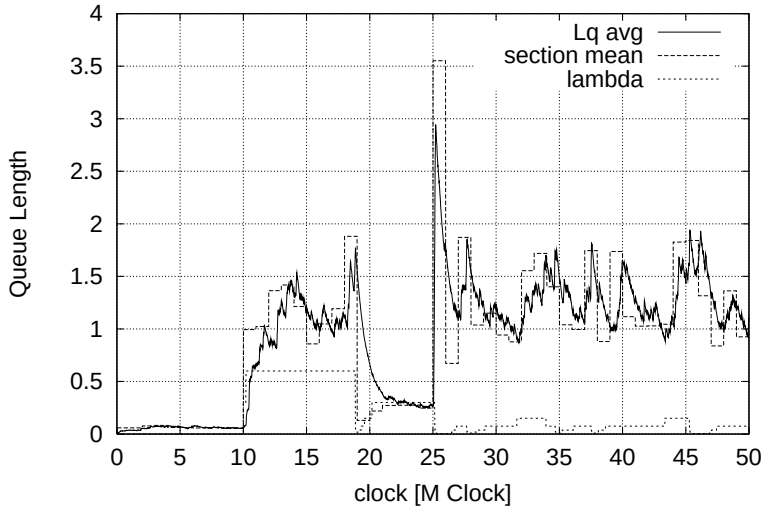

図 4.5: **ステップ関数増加傾向 制御あり Lq 観測**$\lambda_{base} = 0.3 \rightarrow 0.7$

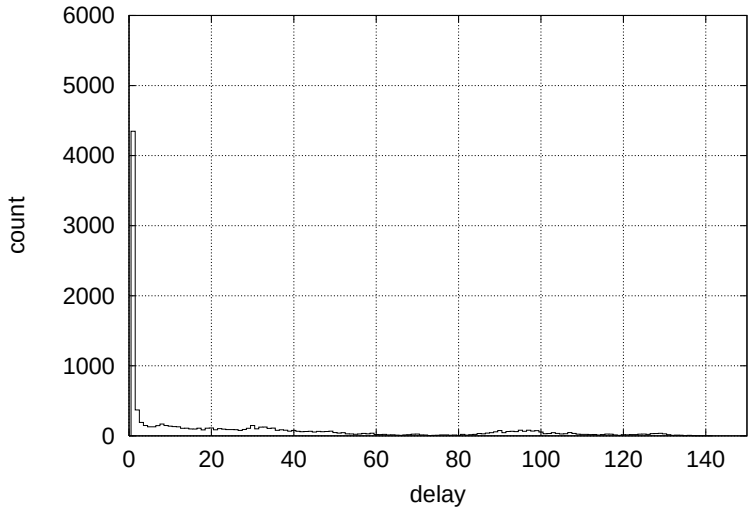

図 4.6: **ステップ関数増加傾向 制御あり伝播遅延時間の度数分布**

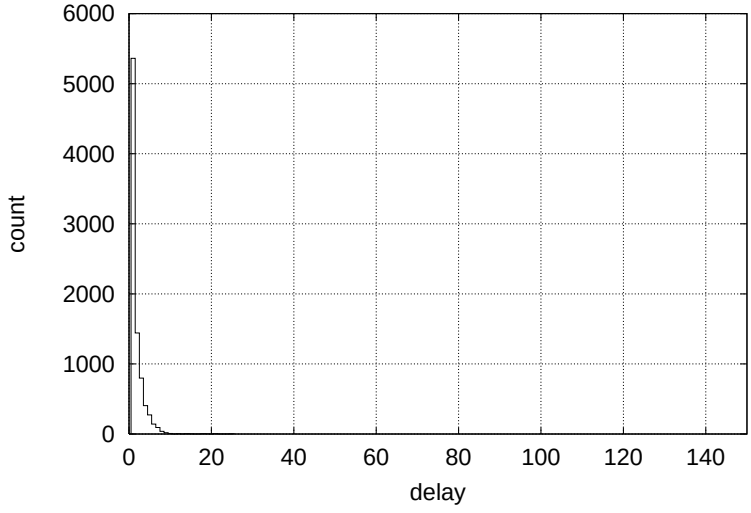

図 4.7: **ステップ関数増加傾向 制御あり伝播遅延時間の度数分布**

表 4.3: ステップ関数減少傾向の度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	60	9,980	25,900	1.47
発呼制御あり	17	1,760	438	0.19

減少傾向

まずは、与えるバックグラウンドトラヒックを図 4.8で示す。

次に、sender での発呼率λ_cの変動について示す。図 4.9は制御なしの場合、図 4.10 は制御ありの場合の図である。バックグラウンドトラヒックは、25Mclock で λ_{base}は 0.7 から 0.3 へと step 関数状に変化する。

制御なしの場合、sernder 側で発呼が行われた直後から著しい平均待ち列長の増大がみられ、ベーストラヒックが低下した後、待ち列は短くなってきており、解消へ向かう。

制御ありの場合は、sender の発呼開始時の 10Mclock で一時的に待ち列は延び 14Mclock 付近では待ち列長は 1 程度になる。さらに、バックグラウンドトラヒックの減少変動が起こる 25Mclock を過ぎると待ち列長は解消しはじめるが、31Mclock 付近で sender 側のトラヒックが増大するために待ち列ができ、バックグラウンドトラヒックの揺れも手伝い待ち列は 1 を越える。45Mclock 付近からの待ち列長の延びもバックグラウンドトラヒックの揺れによるものである。

さらに、制御の効果を詳細に確認するため、伝播遅延時間の分布を調べた。図 4.11は制御なしの場合、図 4.12は制御ありの場合の 1000 clock 毎の伝播遅延時間の度数分布である。

この結果を表 4.3 にまとめる。

最大伝播遅延時間 [度数] を比較すると、制御ありの場合は、制御なしに比べ 28.3 %に短縮している。さらに平均伝播遅延時間は 17.6%まで短縮できていることが分かる。

減少傾向におけるステップ型の変動に対しては、変動前の影響が残っているが、発呼制御は追従できている。

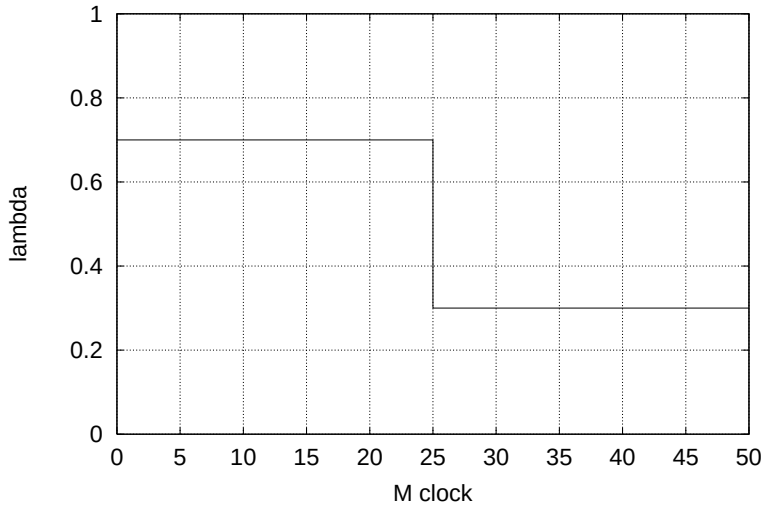

図 4.8: **ステップ関数減少傾向 バックグラウンドトラヒック**

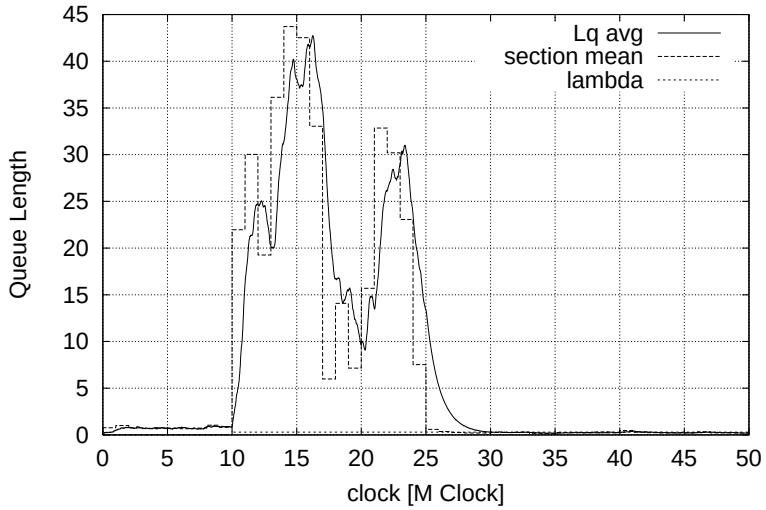

図 4.9: **ステップ関数減少傾向 制御なし平均待ち列長 Lq 観測**$\lambda_{base} = 0.7 \rightarrow 0.3$

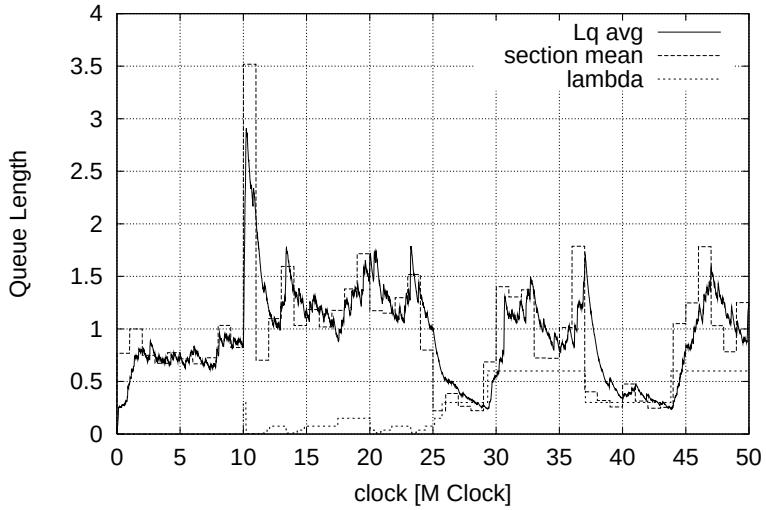

図 4.10: **ステップ関数減少傾向 制御あり平均待ち列長 Lq 観測**$\lambda_{base} = 0.7 \rightarrow 0.3$

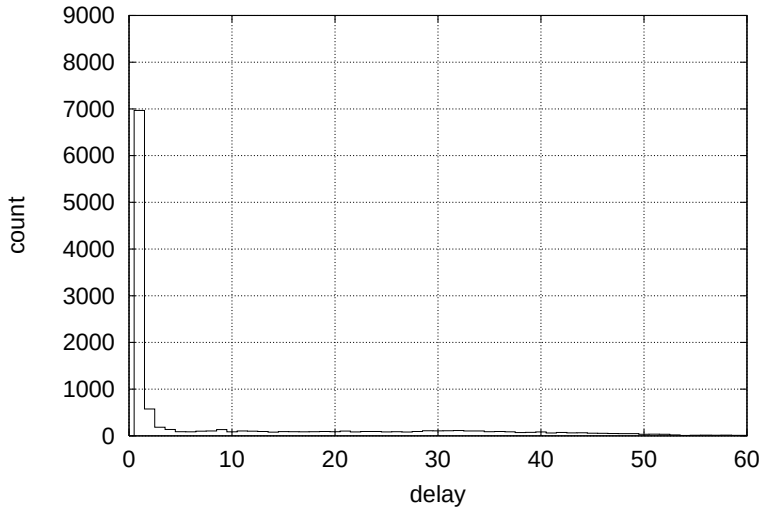

図 4.11: **ステップ関数減少傾向 制御なし伝播遅延時間の度数分布**

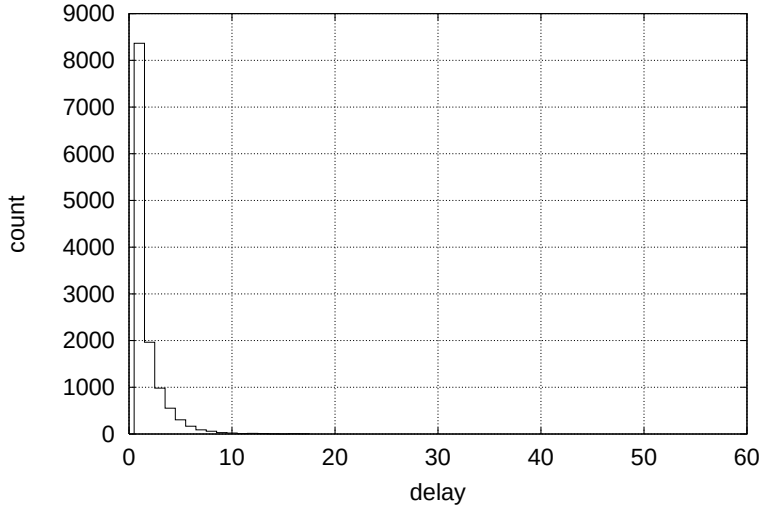

図 4.12: **ステップ関数減少傾向 制御あり伝播遅延時間の度数分布**

4.3.2 sigmoid 変動

急激なバックグラウンドトラヒックの変動型であるステップ関数に対して緩慢なバックグラウンドトラヒックの変動を行う sigmoid 関数について示し。sigmoid 関数でのシミュレーションについて評価し、考察を行う。

増加傾向

まず、与えるバックグラウンドトラヒックを図 4.13で示す。バックグラウンドトラヒックは 15Mclock から 35Mclock までの時間で、λ_{base}は 0.3 から 0.7 へと変動する。次に、sender での発呼率λ_cの変動について示す。図 4.14は制御なしの場合、図 4.15は制御ありの場合の図である。

待ち列長の動向をみるために、図のスケールを変えて表示した。制御なしの場合、25Mclock 付近ですでに待ち列が生じて、一気に増大しはじめ、37Mclcok 付近での待ち列長は 20 を越える。その後、待ち列長は減少に向かうが、40Mclock を過ぎ 43Mclock 付近では待ち列長は 80 を越える。45Mclock 付近で減少しているがさらに待ち列長は増大をはじめてしまう。

制御ありの場合では、発呼開始時の 10Mclock で待ち列を生じ、20Mclock を過ぎた辺りで待ち列は定常状態になり、バックグラウンドトラヒックの緩やかな増加にしたがって待ち列は延び、27Mclock 付近で待ち列は 1.8 程度になる。その後、発呼制御により、待ち列長は減少しはじめるが、29Mclock 付近で若干待ち列長を伸ばしてしまう。その後は 1 から 2 の待ち列長で変動を繰り返している。20Mclock からのバックグラウンドトラヒックがゆっくりと変化するにしたがって、sender の発呼率も段階的に下がり、発呼制御の効果がみてとれる。待ち列は 2 までの値となっており、緩慢な変化をするバックグラウンドトラヒックにおいても発呼制御は効果があることが確認できる。

さらに制御の有効性を詳細にみるために、伝播遅延時間の度数分布グラフを示す。図 4.16は制御なしの場合、図 4.17は制御ありの場合であり、1000 clock 毎の伝播遅延時間の度数分布である。

これを表 4.4にまとめる

最大伝播遅延時間は、制御ありの場合は、制御なしの場合に比べ、9.3 % の最大伝播遅延時間に大幅に短縮されている。また、平均伝播遅延時間も 8.0 %に短縮されている。

sigmoid 関数変動の場合、ステップ型に比べて応答は若干悪くなっているが、トラヒックの変化率が低いため、検出がステップ型に比べて少し遅くなる傾向がみられるが、追従はできていると考えられる。変化率が大きくなる点ではステップ型と同様にすばやく追従

表 4.4: sigmoid 関数増加傾向の度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	140	23,500	15,900	3.63
発呼制御あり	13	1,890	625	0.29

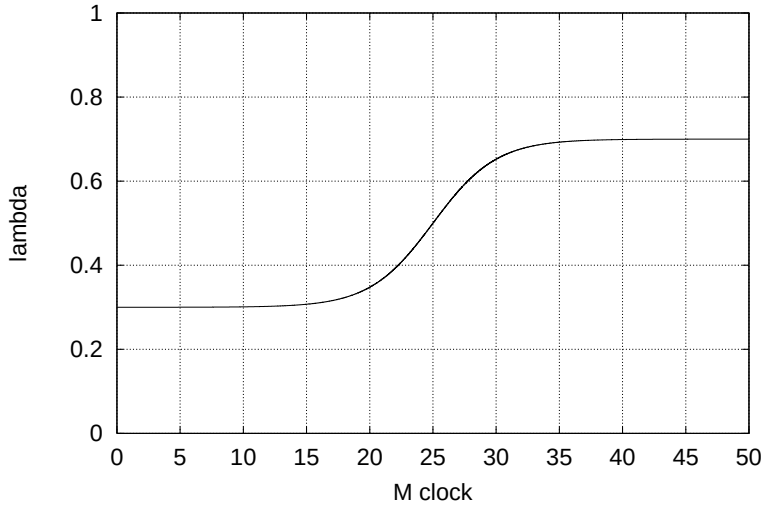

図 4.13: **sigmoid 状増加傾向 バックグランドトラヒック**

できている。

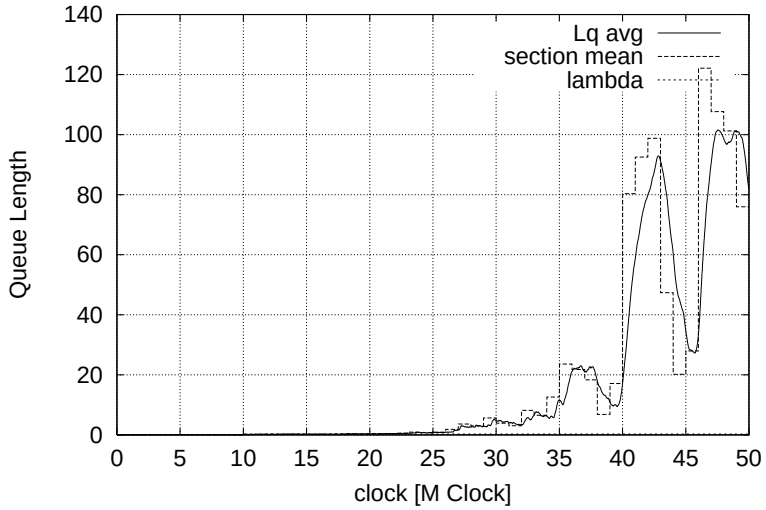

図 4.14: **sigmoid 状増加傾向 制御なし Lq 観測**$\lambda = 0.3 \rightarrow 0.7$

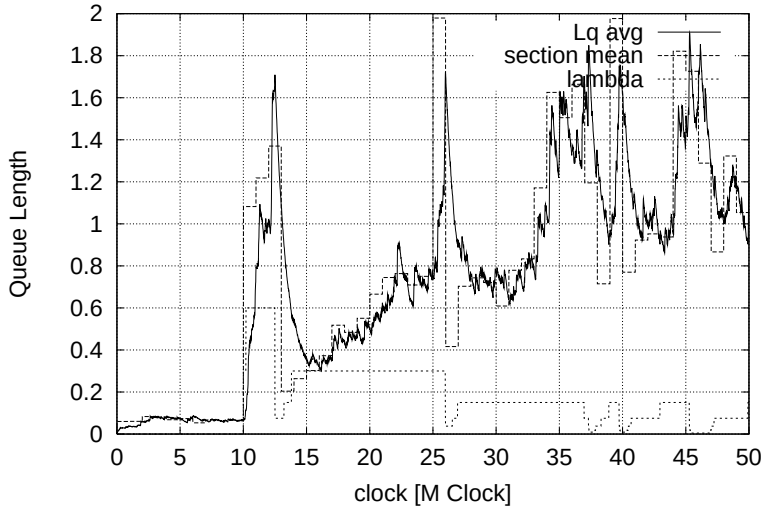

図 4.15: **sigmoid 状増加傾向 制御あり Lq 観測**$\lambda = 0.3 \rightarrow 0.7$

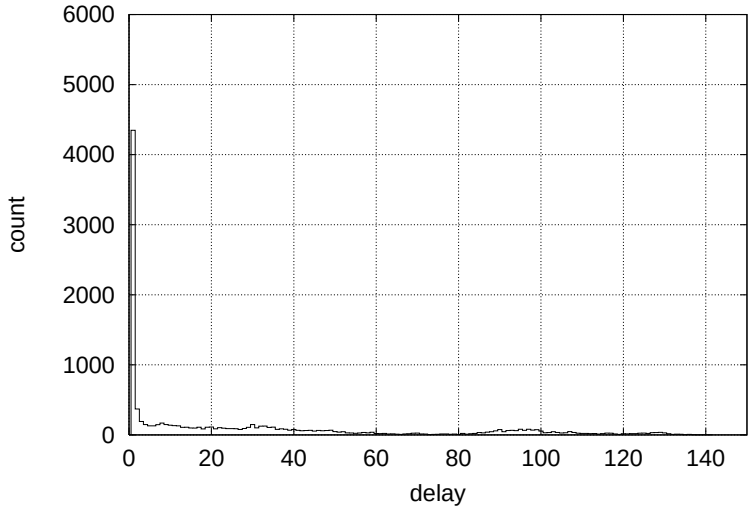

図 4.16: sigmoid **関数増加傾向制御なし伝播遅延時間の度数分布**

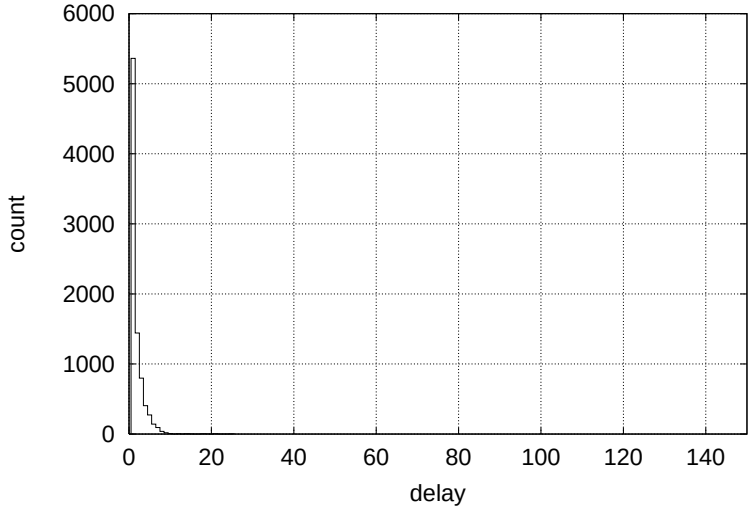

図 4.17: sigmoid **関数増加傾向 制御あり伝播遅延時間の度数分布**

表 4.5: sigmoid 関数減少傾向の度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値 [clock]	分散	標準偏差
発呼制御なし	55	6,230	13,000	1.05
発呼制御あり	18	1,690	444	0.20

減少傾向

まず、与えるバックグラウンドトラヒックを図 4.18で示す。バックグラウンドトラヒックは 15Mclcok から 35Mclock にかけてλ_{base} が 0.7 から 0.3 までの sigmoid 状に減少する。次に、sender での発呼率λ_cの変動について示す。図 4.19は制御なしの場合、図??は制御ありの場合の図である。

待ち列長をみるために、図のスケールは変えて表示している。制御なしの場合、10Mclock を過ぎたところから急激に待ち列長は増大しており最大で 15 を越えている。制御ありの場合では、発呼が開始される 10 M clock で待ち列が若干増えるがその後には待ち列が減少し、解消方向へ向かう。バックグラウンドトラヒックが変動する 15M から 35Mclock での、sender の発呼率が階段状に上昇することがみてとれる。待ち列は 3.5 までの値となっており、緩慢な変化をするバックグラウンドトラヒックにおいても効果があることが確認できる。

さらに制御の効果を詳細に確認するために、伝播遅延時間の分布を調べる。図 4.22は制御なしの場合、図 4.20は制御ありの場合の図である。

この結果を表 4.5にまとめる。最大伝播遅延時間は制御ありの場合は、制御なしの場合に比較し、32.7 %に短縮されている。さらに、平均伝播遅延時間は 27.1 % に短縮されている。

sigmoid 型増加傾向と同様に、待ち列長の変化率が小さいために、ステップ型よりは若干応答は遅れるが、待ち列長の変化に対して追従できていると考えられる。

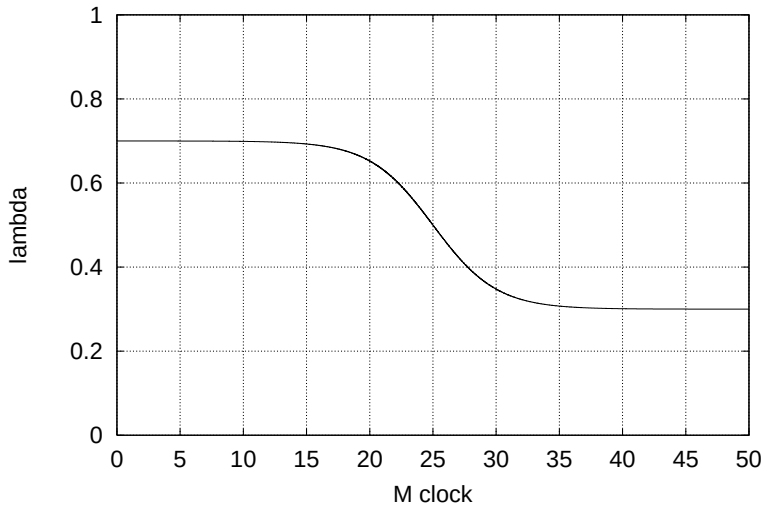

図 4.18: sigmoid 状減少傾向 バックグラウンドトラヒック

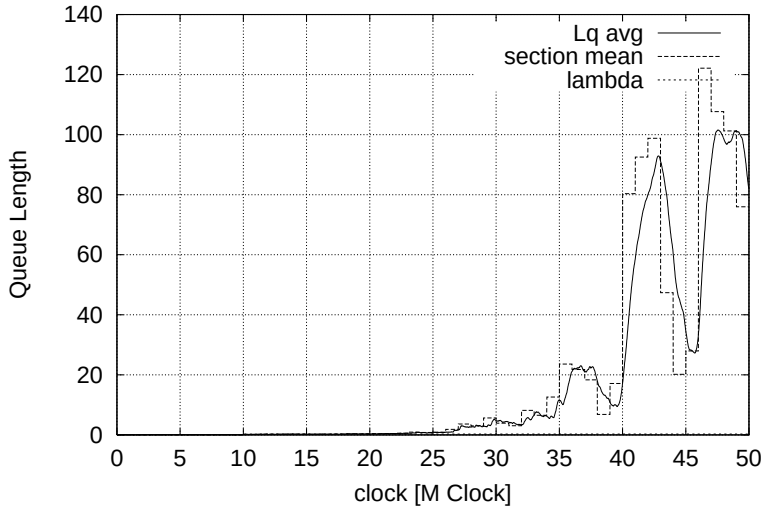

図 4.19: **sigmoid 状減少傾向 制御なし Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

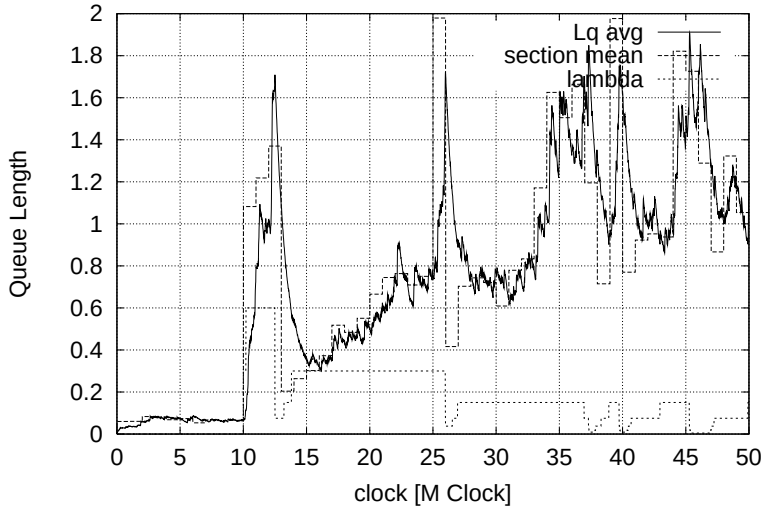

図 4.20: **sigmoid 状減少傾向 制御あり Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

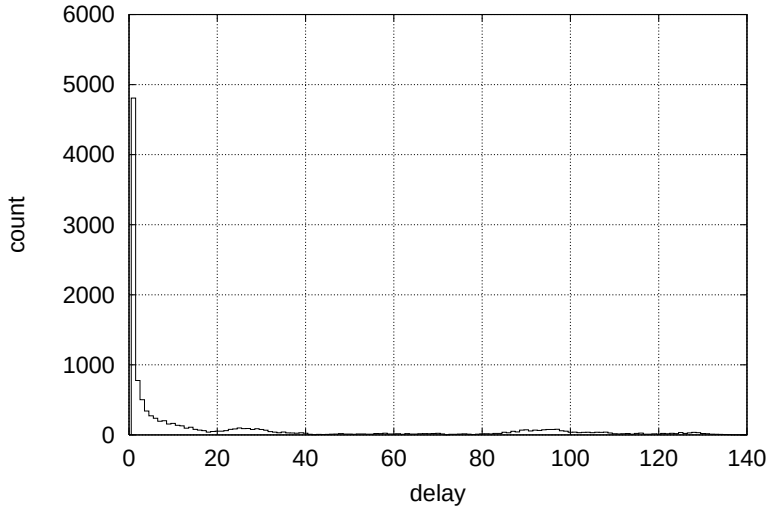

図 4.21: sigmoid 関数減少制御あり伝播遅延時間の度数分布

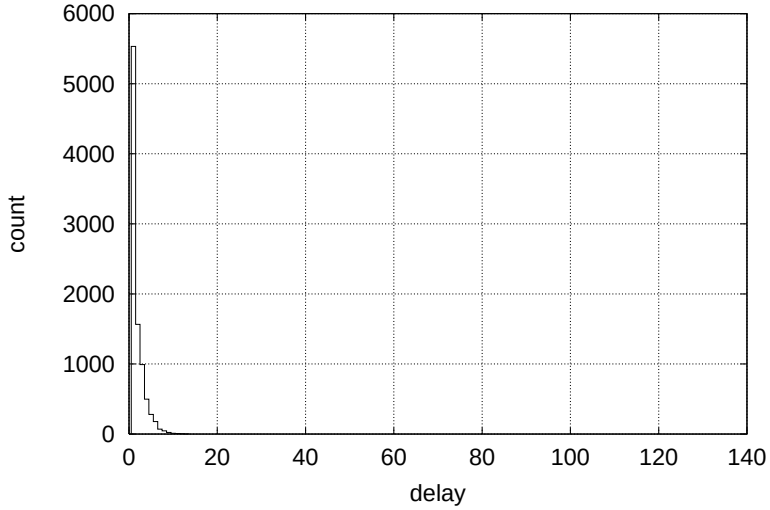

図 4.22: sigmoid 関数減少 制御あり伝搬遅延時間の度数分布

表 4.6: 正弦波関数増加傾向の度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	43	4,480	5,610	0.68
発呼制御あり	13	1,650	500	0.23

4.3.3 正弦波変動

増加傾向

まず、与えるバックグラウンドトラヒックを図 4.23で示す。バックグラウンドトラヒックは sender が発呼を廃止する 10Mclock からシミュレーション終了の 50Mclock の間で λ_{base} は 0.7 と 0.3 の間の正弦波で変化している。

sender の発呼率λ_cの変動について示す。図 4.24は制御なしの場合、図 4.25は制御ありの場合である。

待ち列長の全体の動向をみるために、スケールを変えて表示している。制御なしの場合は、sender の発呼開始から待ち列は延びはじめ、バックグラウンドトラヒックが peek に達する 20Mclock 付近で待ち列は 20 を越える。20Mclcok を過ぎるとバックグラウンドトラヒックが減少傾向になるために待ち列は解消しはじめる。

制御ありの場合は、sender が発呼を開始するまでの 10M clock までは定常状態にあるが、発呼開始した 10Mclock で待ち列長は増加する。20Mclock 付近では、バックグラウンドトラヒックは peek にあるが待ち列長の急激な増大は見られない。バックグラウンドトラヒックが減少傾向にある 20Mclock から 40Mclock では待ち列は解消する方向へ進む。40Mclock からはまた増加傾向に移るが比較的変動が緩慢なために待ち列長への影響はあまりみられない。

制御の効果を詳細に見るために、パケットの発信元からの受信先までの伝搬遅延時間の分布を調べる。図 4.26 は制御なしの場合、図 4.27は制御ありの場合の 1000clock 毎の伝播遅延時間の度数分布である。この結果を表 4.7 にまとめる。

最大伝播遅延時間は、制御ありの場合は、制御なしの場合に比べて、30.2 %に短縮され、平均伝搬遅延時間は 36.8 %に短縮された。

正弦波では、増加傾向および減少傾向の区間内でのバックグラウンドトラヒックの変化率が変わるため、その変化率が小さくなる場合では sigmoid 関数型の場合と同様に利用可能発呼率の制御が遅れる傾向があるが、追従できていると考えられる。

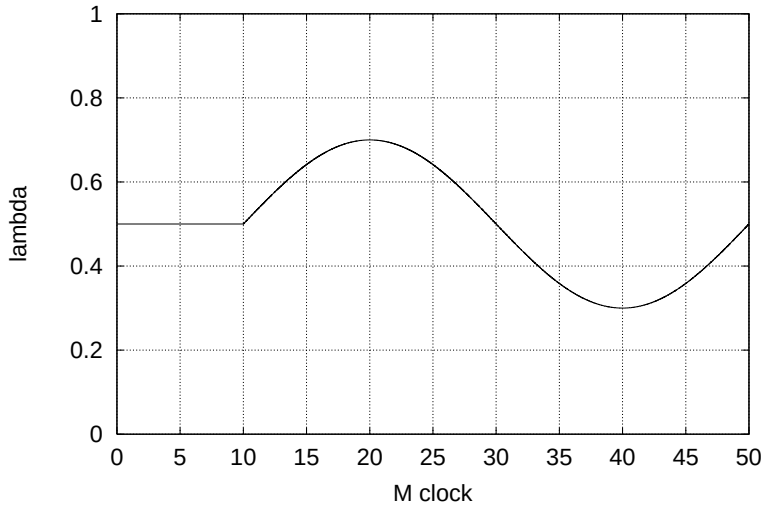

図 4.23: **正弦波関数増加傾向 バックグラウンドトラヒック**

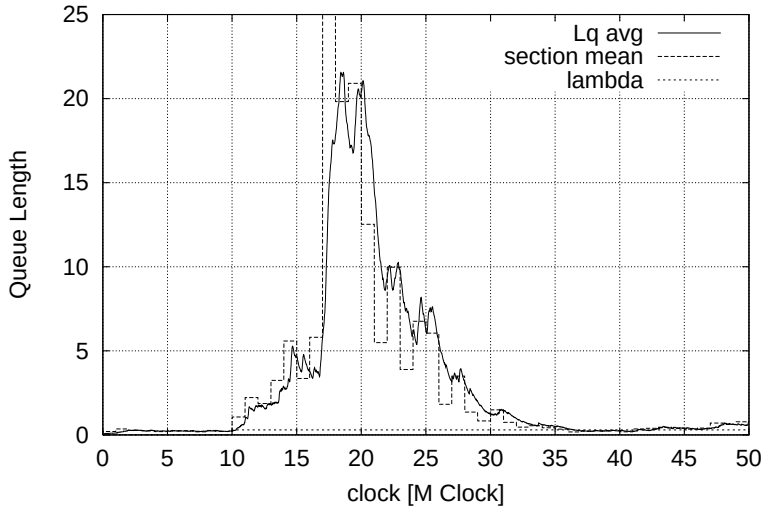

図 4.24: **正弦波関数増加傾向 制御なし Lq 観測**$\lambda_{base} = 0.3 \rightarrow 0.7$

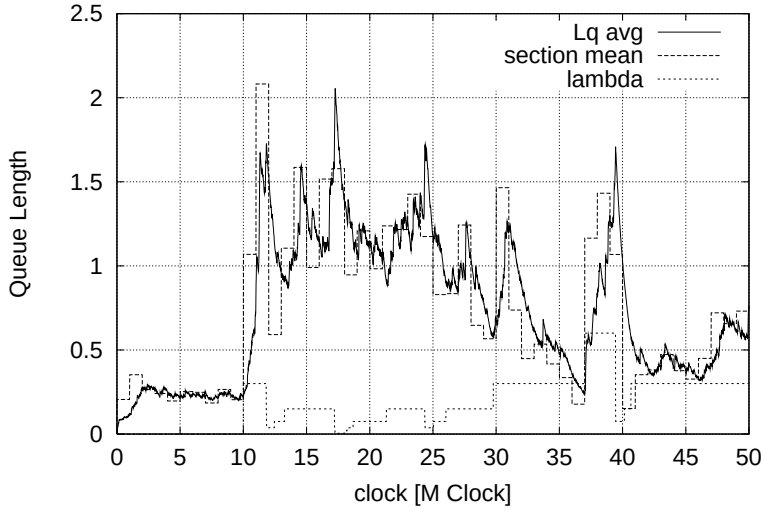

図 4.25: **正弦波関数増加傾向 制御あり Lq 観測**$\lambda_{base} = 0.3 \rightarrow 0.7$

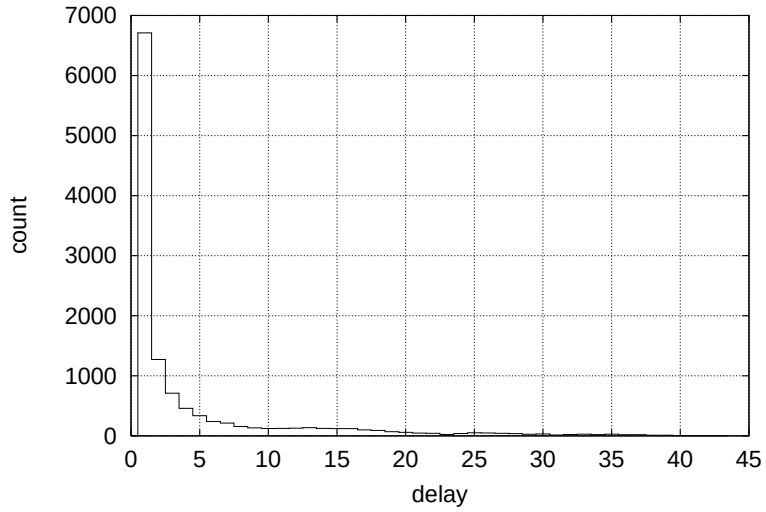

図 4.26: **正弦波関数増加傾向 制御なし伝播遅延時間の度数分布**

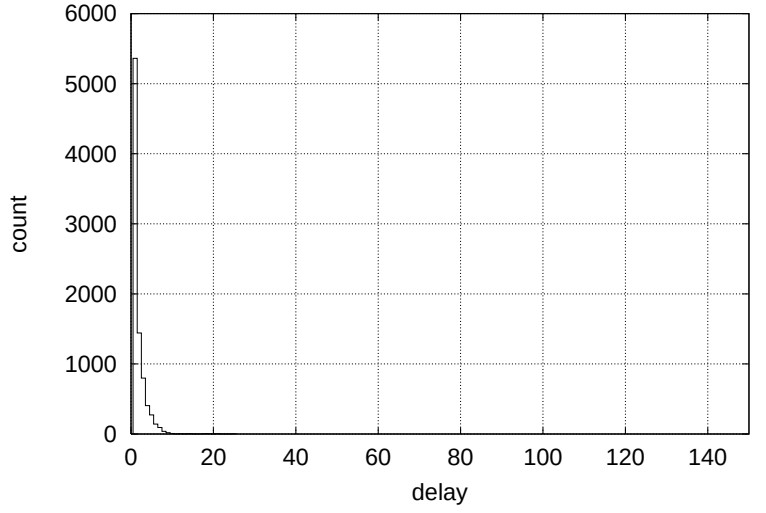

図 4.27: **正弦波関数増加傾向 制御あり伝播遅延時間の度数分布**

表 4.7: 正弦波関数減少傾向の度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	112	11,300	60,900	2.25
発呼制御あり	15	1,620	494	0.26

減少傾向

まず、与えるバックグラウンドトラヒックを図 4.28で示す。バックグラウンドトラヒックの変動は 10Mclock から 50Mclock の間で、λ_{base}0.3 から 0.7 の間の値を半周期ずらした正弦波で変動する。10Mclock から 20Mclcok までは減少傾向であり、20Mclock でバックグラウンドトラヒックは 0.3 になり、その後、40Mclock までは増加傾向となり、40Mclock で 0.7 なったのち 50Mclock で 0.5 になる変動である。次に、sender での発呼率λ_cの変動について示す。図 4.29は制御なしの場合、図 4.30は制御ありの場合である。

待ち列長に関する図は待ち列長の動向をみるためにスケールを変えて表示している。制御なしの場合は、バックグラウンドトラヒックが最大ピークの 0.7 を取る 40Mclock を過ぎたところで待ち列は急激に増大し 90 を越え、その後待ち列は解消に向かう。

制御ありの場合では、sender の発呼開始の 10Mclock から、やや待ち列が延びるが、バックグラウンドトラヒックが減少している区間なので 20Mclock 付近で 0.25 程度落ちつく。バックグラウンドトラヒックが増加傾向に向かう 20Mclock を過ぎると一時的に待ち列は延びる。しかし発呼制御によって sender の発呼率が下がるので 25Mclock 付近では定常状態になっている。さらに 40Mclock まではバックグラウンドトラヒックが増加傾向にあるので若干待ち列はのびるが。45Mclock をすぎるとバックグラウンドトラヒックの減少傾向のために待ち列は解消に向かう。待ち列長が増大しているときには、発呼制御が行われており、35Mclock から 45Mclcok までの間は sender の発呼率は低く抑えられている。

制御の効果を詳細に見るために、パケットの発信元からの受信先までの伝播遅延時間の分布を調べる。図 4.31は制御なしの場合、図 4.32は制御ありの場合の 1000clock 毎の伝播遅延時間の度数分布である。

この結果を表 4.7 にまとめる。

最大伝播遅延時間は、制御ありの場合は、制御なしの場合に比べて、13.4 %に短縮できた。平均伝播遅延時間は 14.3 %にに短縮できた。

増加傾向と同様に、待ち列長の変化率が大きい区間では、発呼制御は速やかに行われているが、変化率が小さい区間では若干遅れる傾向が見られる。

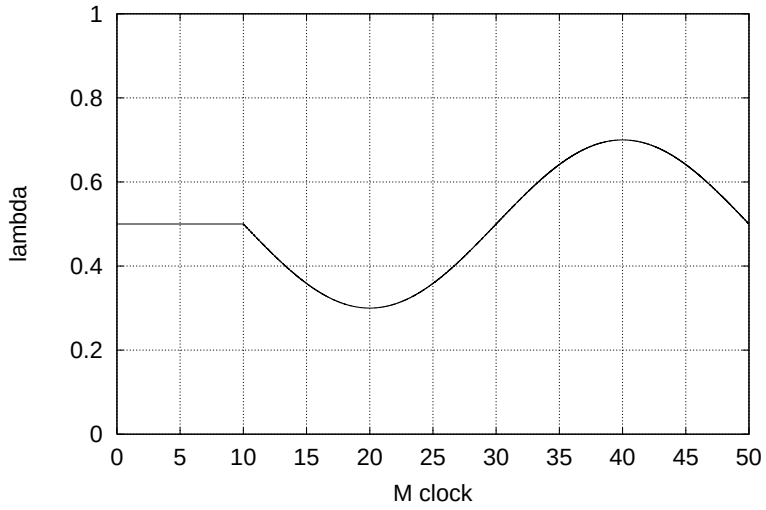

図 4.28: **正弦波関数減少傾向 バックグラウンドトラヒック**

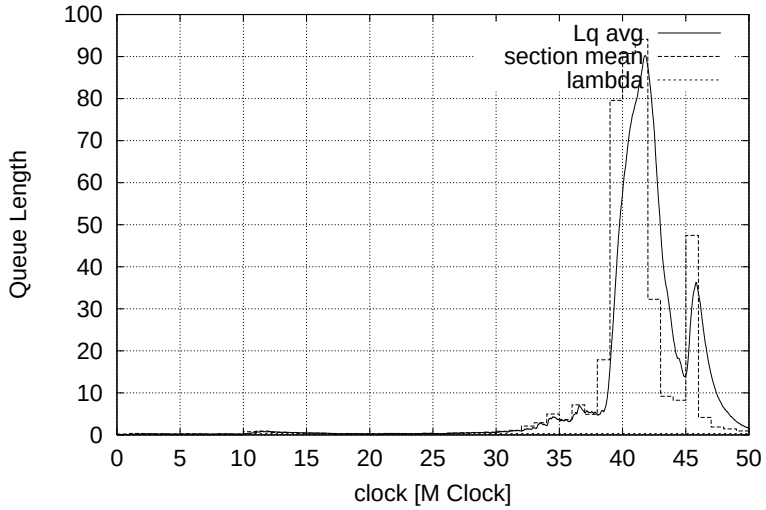

図 4.29: **正弦波関数減少傾向 制御なし Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

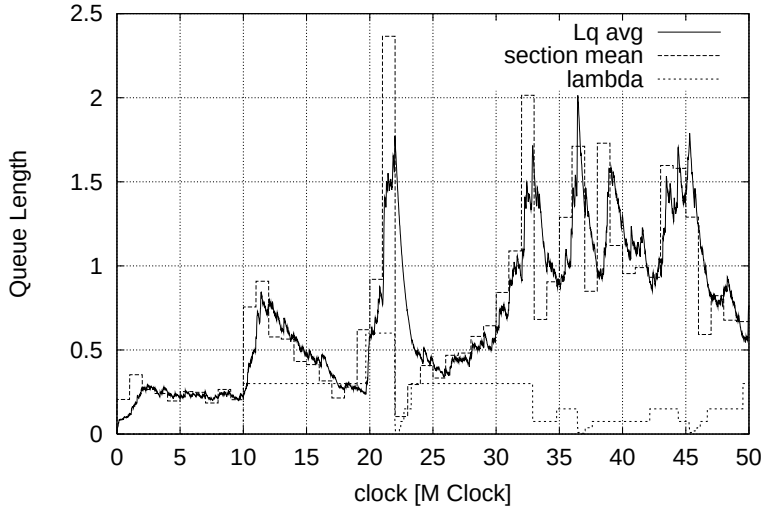

図 4.30: **正弦波関数減少傾向 制御あり Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

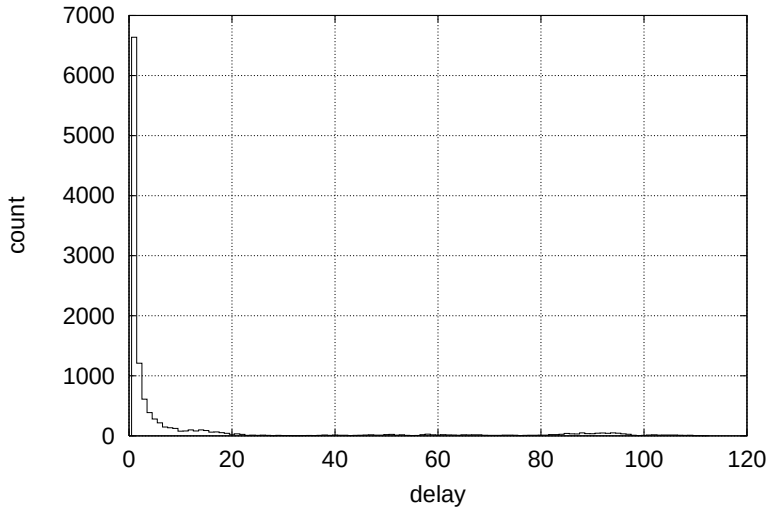

図 4.31: **正弦波関数減少傾向 制御なし伝播遅延時間の度数分布**

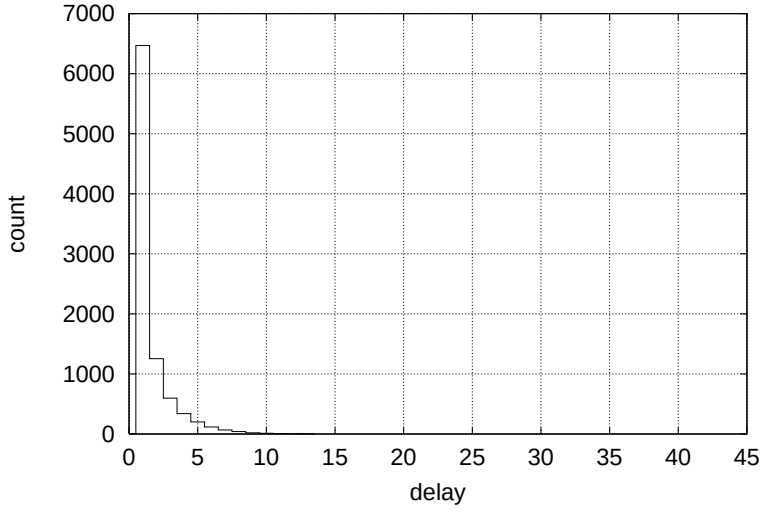

図 4.32: **正弦波関数減少傾向 制御あり伝播遅延時間の度数分布**

4.3.4 線形型変動

増加傾向

まず、与えるバックグラウンドトラヒックを図 4.33で示す。バックグラウンドトラヒックの変動は、sender が発呼を開始する 10Mclock からシミュレーションが終る 50Mclock までの間でλ_{base}は 0.3 から 0.7 の間で変動する。バックグラウンドトラヒックの変動をみると、10Mclock から 20Mclock では 0.5 から 0.7 へ増加し、20Mclcok から 40Mclock までの間は 0.7 から 0.3 まで減少する。さらに 40Mclock から 50Mclock の間では 0.3 から 0.5 までの増加変動となる。次に、sender での発呼率λ_cの変動について示す。図 4.34は制御なしの場合、図 4.35は制御ありの場合である。

図は待ち列長をみるためにグラフのスケールを変えて表示している。制御なしの場合、sernder の発呼が開始された直後から平均待ち列長のぞうだいがみえ、バックグラウンドトラヒックが peek の 0.7 を示す 20Mclock 付近で最大の待ち列になっているが、その後はバックグラウンドトラヒックが減少傾向であるために待ち列は 35Mclock 付近でほぼ解消しており、その後でも著しい増大は見られない。

制御ありの場合は、10M clock で、一時待ち列長が増大するものの、発呼可能率λ_cが制御され、最大の待ち列長は 2 に満たない平均待ち列長に収まっている。バックグラウンドトラヒックが減少しはじめる 20Mclock 以降は待ち列長は解消方向へ進みだす。37Mclock 付近の待ち列は、バックグラウンドトラヒックの揺れによるものであると思われる。発呼制御に関しては 14Mclock 付近と 17Mclock 付近の待ち列の増大に合わせて利用可能発呼率を下げて対応している。

制御の効果を詳細に見るために、パケットの発信元からの受信先までの伝播遅延時間の分布を調べる。図 4.36は制御なしの場合、図 4.37は制御ありの場合の 1000clock 毎の伝播遅延時間の度数分布である。

この結果をまとめた表 4.8を示す。最大伝播遅延時間は、制御ありの場合は、制御なしの場合に比べて、41.9 %に短縮されている。さらに、平均伝播遅延時間は、20.5 % に短縮されている。

バックグラウンドトラヒックが周期変動をする場合の正弦波と比較すると、線形型の場合では、バックグラウンドの増加傾向および減少傾向の区間でのλ_{base}の変化率は一定でとなっており、

表 4.8: 度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	31	2,960	2,020	0.43
発呼制御あり	13	1,530	415	0.24

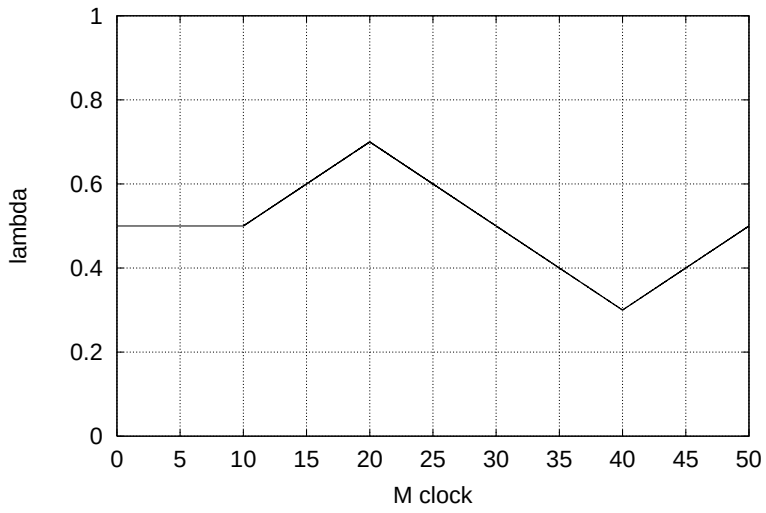

図 4.33: **線形関数増加傾向 バックグラウンドトラヒック**

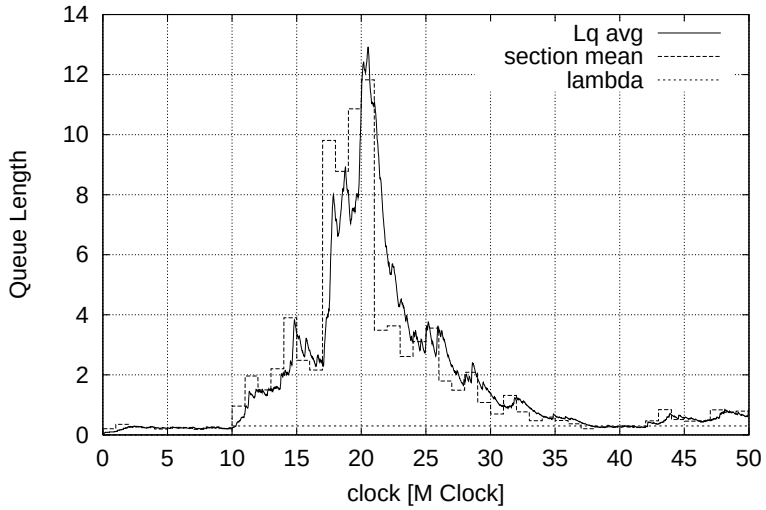

図 4.34: **線形関数増加傾向 制御なし Lq 観測**$\lambda = 0.3 \rightarrow 0.7$

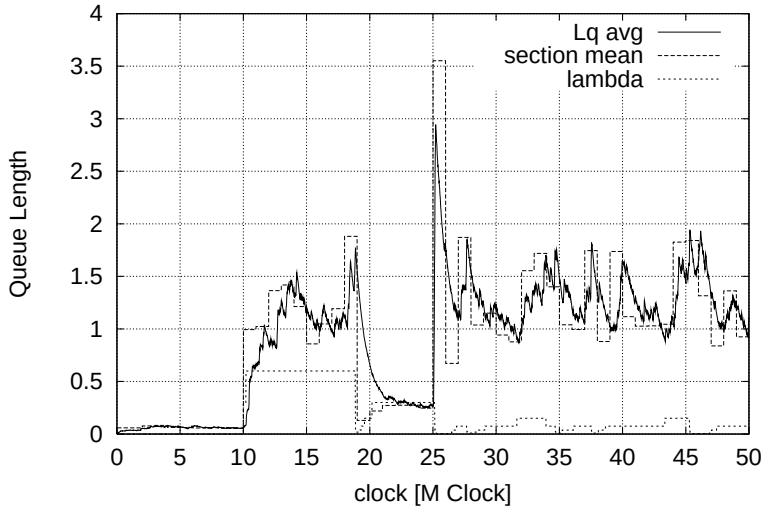

図 4.35: **線形関数増加傾向 制御あり Lq 観測**$\lambda = 0.3 \rightarrow 0.7$

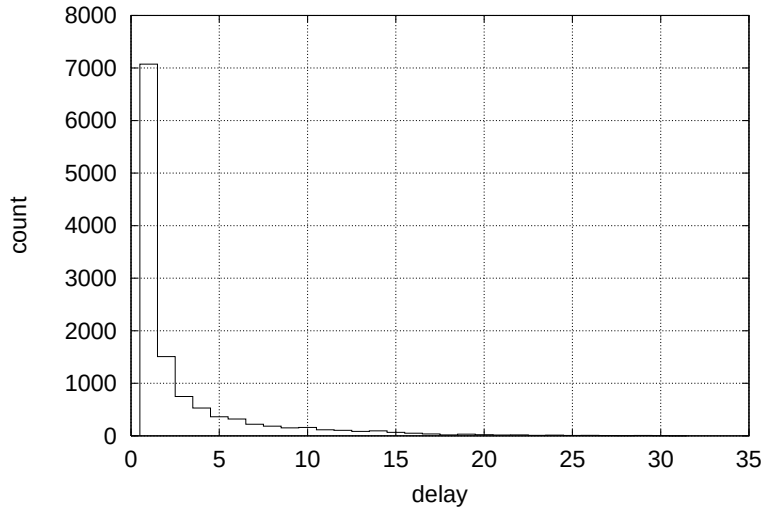

図 4.36: **線形型関数増加傾向 制御なし伝播遅延時間の度数分布**

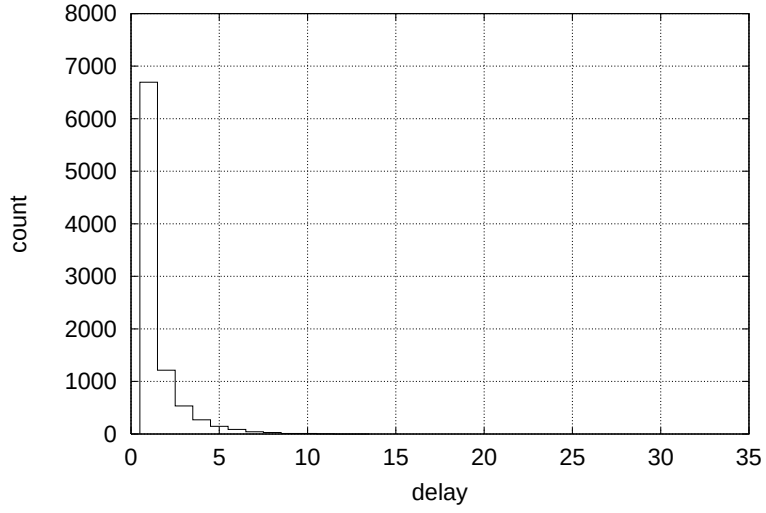

図 4.37: **線形型関数増加傾向 制御あり伝播遅延時間の度数分布**

減少傾向

まず、与えるバックグラウンドトラヒックを図 4.38で示す。変動は、sender が発呼を開始する 10Mclock から 50Mclock の間に、λ_{base}は 0.3 から 0.7 の間で変動する。バックグラウンドトラヒックは、10Mclock から 20Mclock の間は 0.5 から 0.3 へと減少し、20Mclock から 40Mclock は 0.3 から 0.7 へ増加する。さらに、40Mclock から 50Mclock の間では 0.7 から 0.5 へと変動するバックグラウンドトラヒックを与える。

次に sender での発呼率λ_cの変動について示す。図 4.39は制御なしの場合、図 4.40は制御ありの場合である。

待ち列長に対する変化を表示するために、図 4.39と図 4.40はスケールを変えている。制御なしの場合、バックグラウンドトラヒックが peek を向かえた 40Mclock 付近に急激な待ち列の増大が見られ、待ち列長は 50 を越えてしまう。

制御ありの場合には、sender が発呼を開始する 10Mclock を過ぎた辺りで待ち列を一時的に生じるが、バックグラウンドトラヒックが 0.3 となる 20Mclock では定常状態にある。さらに 20Mclock を過ぎて、バックグラウンドトラヒックが増加傾向になると待ち列は少しづつ延びはじめ、33Mclock 付近で待ち列が 1.8 近くになると同時に sender での発呼率は低下し、待ち列の増大を防いでいる。さらにバックグラウンドトラヒックが peek 値の 0.7 に近付いてくる 35Mclock 以降では若干の待ち列長の増大は見られるが、40Mclock 付近では待ち列長は 1 となっており、待ち列の著しい増加は見られない。40Mclock を過ぎて、バックグラウンドトラヒックが減少傾向になると、待ち列は解消方向へ向かう。45Mclock 付近の待ち列の増大は、バックグラウンドトラヒックでの乱数による揺れのための増大である。

制御の効果を詳細に見るために、パケットの発信元からの受信先までの伝播遅延時間の分布を調べる。図 4.41 は制御なしの場合、図 4.42は制御ありの場合の 1000clock 毎の伝播遅延時間の度数分布である。

この結果を表 4.9 にまとめる。最大の伝播遅延時間は制御ありの場合、制御なしの場合に比べて、17.9 % に短縮できている。さらに平均伝播遅延時間では 22.2 %の短縮に短縮できた。

増加傾向と同様に、待ち列長の変化率によって応答が異なることが分かる。さらに正弦波の減少傾向と比べてみると 20Mclock から 25Mclock までの区間での待ち列長の増加がみられない。これはバックグラウンドトラヒックの変化率によるものであると考察できる。

表 4.9: 度数分布からの平均値の比較

λ_{base}	最大伝播遅延時間 [度数]	平均値	分散	標準偏差
発呼制御なし	67	6,750	21,600	1.36
発呼制御あり	12	1,500	400	0.24

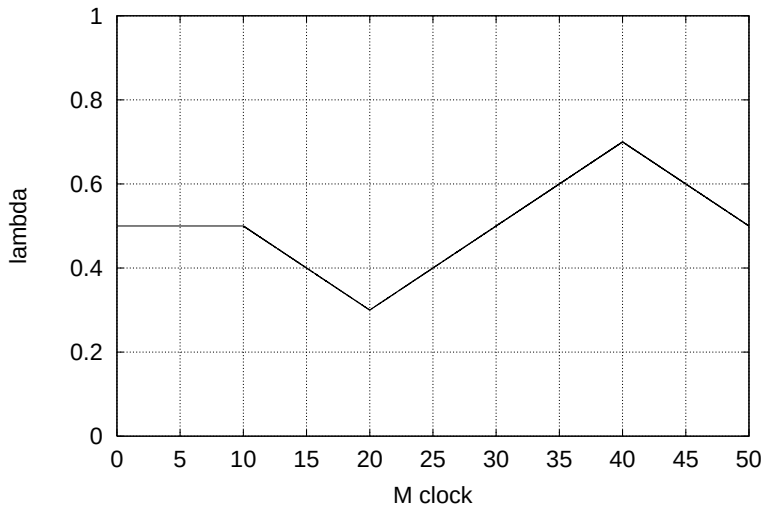

図 4.38: **線形関数減少傾向 バックグラウンドトラヒック**

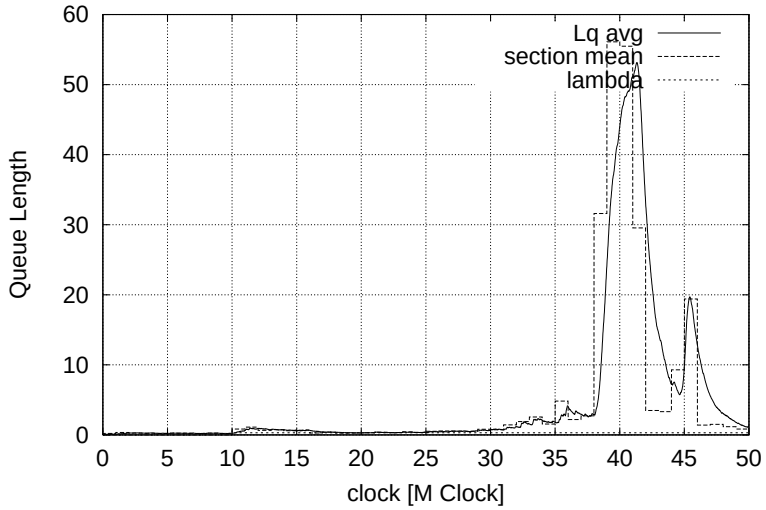

図 4.39: **線形関数減少傾向 制御なし Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

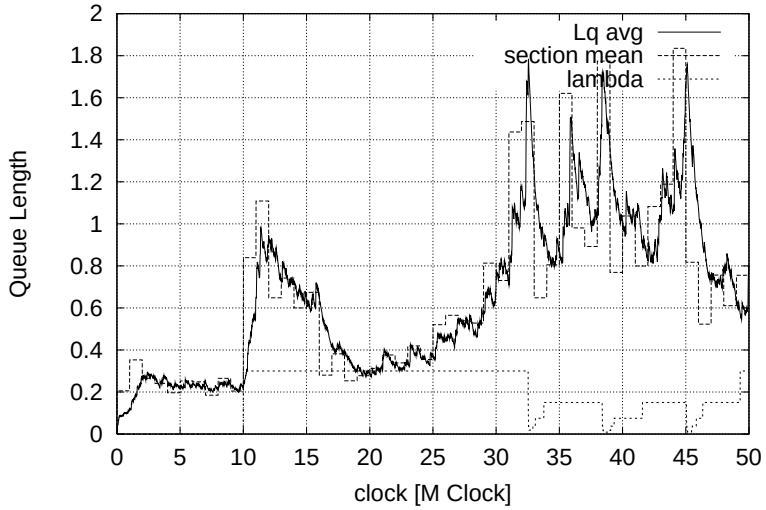

図 4.40: **線形関数減少傾向 制御あり Lq 観測**$\lambda = 0.7 \rightarrow 0.3$

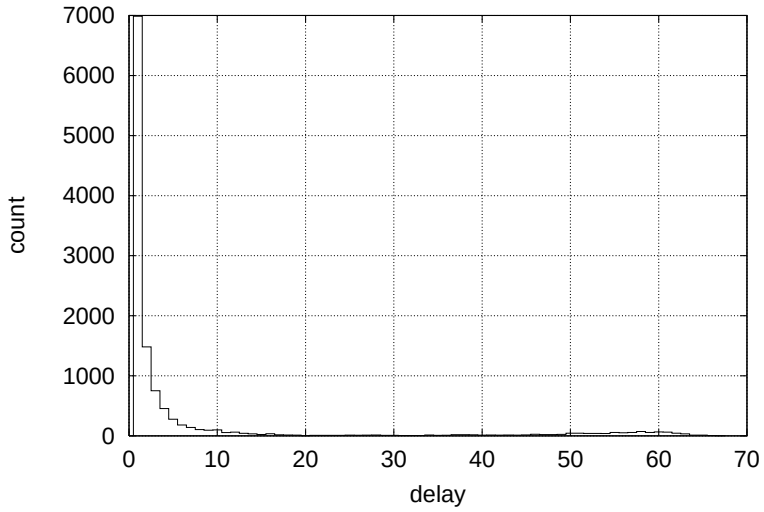

図 4.41: **線形関数減少傾向 制御あり伝播遅延時間の度数分布**

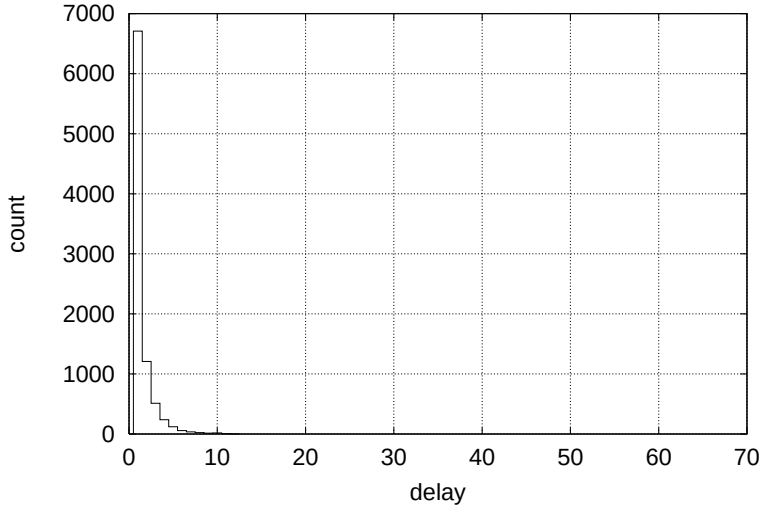

図 4.42: **線形関数減少傾向 制御あり伝播遅延時間の度数分布**

4.4　シミュレーション結果について

トラヒックの変動パターンを変えてシミュレーションを行ったが、どの変動の場合においても、発呼制御を行った場合、制御を行わなかった場合に比べて、伝播遅延時間を縮小できることが確認できた。

変動率に対するシミュレーションとして急激な変動率であるステップ関数型と緩慢な変動を示すsigmoid 関数型の増加傾向と減少傾向について比較すると、得られた結果としては、特に、急激な増加傾向のトラヒック変動に対しての応答性の方が追従性がよいことが分かった。緩慢な変動でも、急激な変動に対する応答性よりは若干悪くなるが、発呼率の増減は階段状に増減し、制御されていることが見て取れる。

さらに周期関数での場合は、正弦波関数と線形関数の結果より考察する。周期的な変動であっても、線形型の変動は、区間内での変化率は一定に対して、正弦波の場合では変化率が異なる。このために、図 4.30と図 4.40を比べると 20Mclock での減少傾向から増加傾向への変更がされた場合には、利用可能発呼率に揺れが起こり、そのために待ち列長を増大させてしまう傾向がみれる。トラヒックの変化率が低い場合には、待ち列長に対する発呼制御の追従性は急激な変化にくらべて悪くなり、そのため発呼制御に遅延を生じるためであると思われる。バックグラウンドトラヒックの変化率における待ち列長の影響は、図 4.30と図 4.40で見られる、20Mclock から 25Mclock までの待ち列長によく現れている。

通信回線上のトラヒック変動が起こる場合でのもっとも重要な問題は、先に述べたとおり、増大するトラヒックに対して行われる発呼制御が重要である。それぞれの結果で示した通り、通信回線上のトラヒック増大が著しければ著しいほど、発呼制御は伝播遅延時間の縮小に有効である。逆に、この制御における、通信回線上のトラヒックが減少している場合での有効性は、通信帯域の有効利用になることである。通信回線上のトラヒック減少は、制御なしの場合でも、待ち列は減少し、その状態が長期間に渡れば、待ち列は解消してしまうからである。トラヒックの減少傾向時でも、発呼率λを増加させて利用できる帯域を利用することも確認できる。

第 5 章

結論

5.1 本研究での成果

本論文では、まず、通進回線上のトラヒック測定の利点について、発呼制御ができ、帯域を有効に利用できることを述べ、機構としてトラヒック測定機構と発呼制御機構についての制御に対する観点から、考察し検討を行った。また、発呼制御時の帯域割り当てのポリシーも検討を行った。

次に、システムをモデル化し、トラヒックパターンをモデル化した上で、トラヒックの変動の特性によるシミュレーションによる実験を行い、トラヒック測定機構の応答性・利用可能帯域に適応させた発呼制御機構の有効性の確認を行った。

この利用可能帯域を意識した発呼制御機構では、トラヒック測定を行った上での発呼制御が行われた呼が、発呼制御を行わない場合に比較して、呼の伝播遅延時間を縮小させ、特に使用率ρが高い時に有効であることが示せた。end-to-end 間のトラヒック状況が把握でき、その状況に適応させた発呼制御ができれば、定常状態時では伝播遅延時間はかなり短くできる。このことは輻輳が生じるまえに制御を行うことで、極力、輻輳を抑えることができ、輻輳状態が生じても復旧が早くなることが考えられる。

5.2 課題

シミュレーションモデルの環境の拡張に関しては、ネットワークトポロジー、ルータに接続される回線数、リンクされた回線の速度などシステム全体が把握できるようなネットワークという仮定が必要となるので小規模なイントラネットまでになると思われる。

トラヒック測定機構での平均待ち列長 Lq_{avg}に関しては、加重平均での平均待ち列長で

あるため、発呼した呼が待ち列に反映されるまでのタイムラグは存在するので、短期的なトラヒック変動と長期的なトラヒック変動を両立させられる、より精度の高い平均待ち列長の算出方法が見つけれられれば、さらに有効な発呼制御ができる可能性がある。

また、発呼制御されている呼に対して優先度を与えた場合に、他のバックグラウンドトラヒックに対しての影響の有無を確認でき示せれば、優先度付きの呼に対する有効性が示すことが可能であると推察できる。この場合の究極的な方法は、帯域の予約ということになる。

発呼制御を行うアプリケーションが複数存在する場合には、それぞれの発呼制御は、使用率ρから算出される利用可能帯域しか算出できないので、同じ利用可能帯域を争奪することになりかねない。この問題を解決するには、綿密なアプリケーションの帯域利用のポリシーが大きくかかわってくることになる。

また、経路上に複数のルータが存在する場合には、end-to-end 間で利用する経路がわかれば、その経路にある情報を集め、その中で最小の利用可能発呼率を計算してやればよいが、ルータ間での同期は取れていないので、情報のフィードバックに関する時間間隔は大きな問題となりうる。増大するトラヒックに対する制御がもっとも重要であることにかわりはない。

ルータの出力ポートが複数ある場合は、それぞれの出力ポート毎に待ち列長測定と情報のフィードバックを行わなければならず、ルータに対する負荷がより高くなってしまう。さらにフィードバックされる情報にも識別できるような工夫をしなければならない。

フィードバックされる情報に関しても課題がある。フィードバック情報の間隔を短くしすぎると、その情報による情報の嵐がおこり、フィードバック方向のトラヒックを増大させる。その結果、sender に届いた情報は余分に時間経過した古い情報となり、発呼制御を行うためには、意味のない情報になりうる。フィードバック情報による情報の嵐を押さえた上で、発呼制御に対して意味のある情報にするように両立させるためには、送られる情報の変動率でフィードバック間隔を調整するのがよいと思われる。ある閾値を設け、それ以上の変動がある場合にのみ送るようにすれば、その情報自身によるトラヒックの増大は防げる。

5.3 提案方法の適用範囲

提案方式での適用範囲は、モデルとして考えた全ネットワークトポロジーやリンク数、リンクされている回線速度等が把握できているような小規模なイントラネットが条件に適応する。しかし、発呼制御が適正に行われるための条件には、経路上のすべてのルータが

トラヒックの測定機構を持ち、待ち列長の情報をフィードバックできることが必要条件となる。さらに、複数の経路がある場合には、経路情報から情報を得られるようなトラヒックエイジェントのような機能を持ったホストを必要するが、各ルータからの情報の同期を取る必要もでてくる。さらにこのような問い合わせ方式を採用すると、問い合わせのためのトラヒック集中が生じてしまい良い方法とは思えない。

したがって、sender による自律的な制御が行える範囲が適応範囲と考えらえれるので、大規模なネットワークには向かない。

参考文献

[1] 川島幸之助, 町原文明, 高橋敬隆, 斎藤洋, “通信トラヒック理論の基礎とマルチメディア通信網”, 電子情報通信学会, 1995 pp.6-69

[2] S.Floyd. and V. Jacobson, “Random Early Detection gateways for Congestion Avoidance.” ftp://ftp.ee.lbl.gov/papers/early.ps.gz, (IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.)

[3] 森村英典, 大前義次,” 応用待ち行列理論”,1975, 日科技連出版社

[4] 牧野都次,” 待ち行列の応用”,1969, 森北出版

[5] 松本浩久,” 異種通式方式ネットワークの相互接続における特性解析”,1998, 北陸先端科学技術大学院大学修士論文.

[6] 雨宮幸雄,” ネットワークスペシャリスト試験待ち行列必勝作成”，1996、オーム社

[7] 山中惣乃助,” 通信方式 情報伝送の基礎”,1977, 朝倉書店

謝辞

本研究を行なうにあたり、日頃御指導を賜わりました日比野靖教授に心より感謝致します。また、コンピュータネットワークの基礎についてや、適切な御示唆、御指導頂きました丹康雄助教授に深く感謝致します。さらに、日頃から有益な助言、シミュレーション環境整備に御助力頂きました宮崎純博士に感謝致します。

最後に、貴重な御意見、御討論を頂きました日比野研究室所属の皆様に厚く御礼申し上げます。

第 A 章

シミュレーション結果

A.1　加重率 w決定シミュレーション

加重率 wを決定するために、Lq_{avg}の区間平均待ち列長への応答性と追従性をグラフで示した。各々、トラヒック増加傾向、トラヒック減少傾向についてシミュレーションを行った結果を示す。

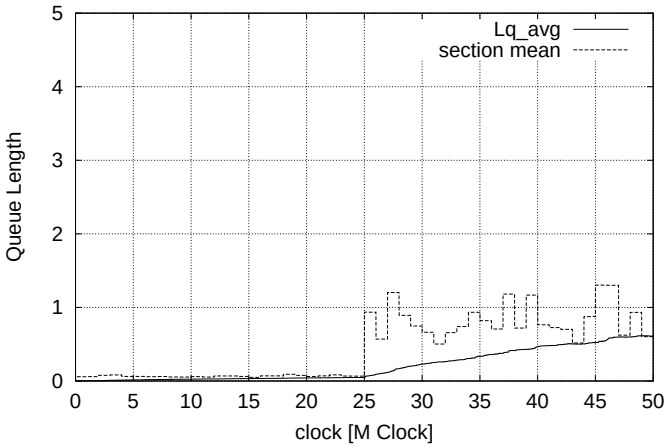

図 A.1: **w=0.000025**

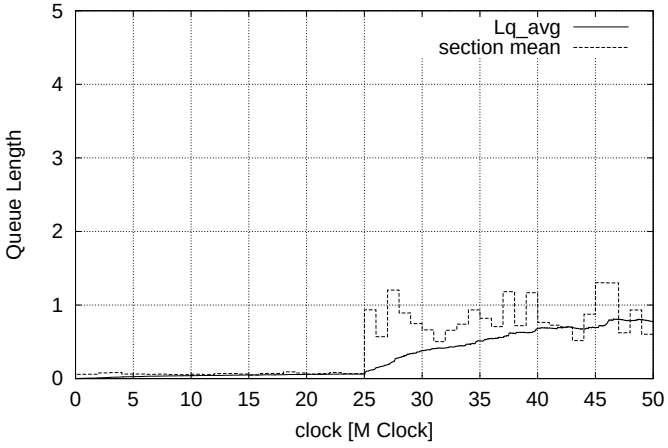

図 A.2: **w=0.00005**

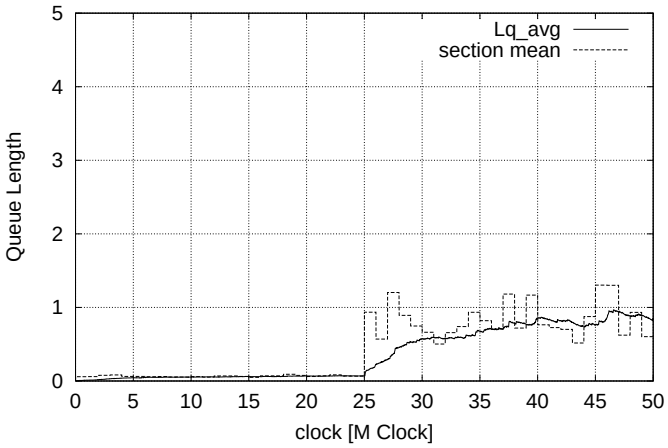

図 A.3: **w=0.0001**

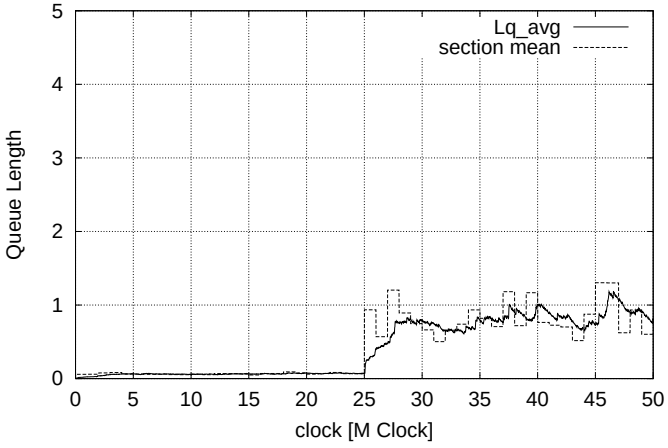

図 A.4: **w=0.00025**

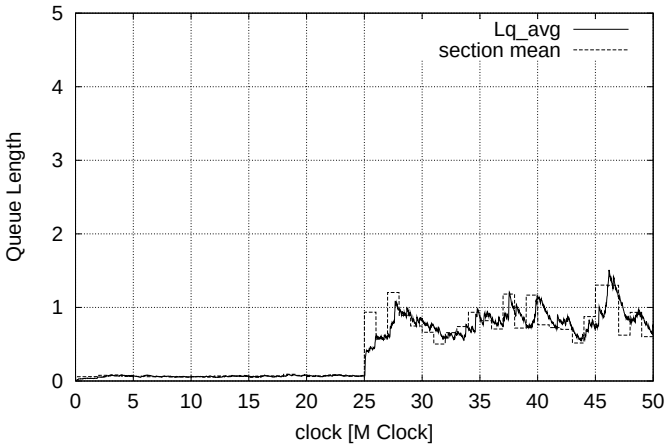

図 A.5: **w=0.0005**

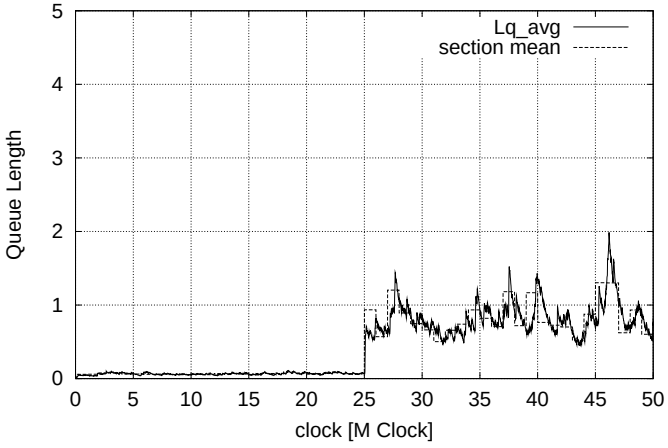

図 A.6: **w=0.001**

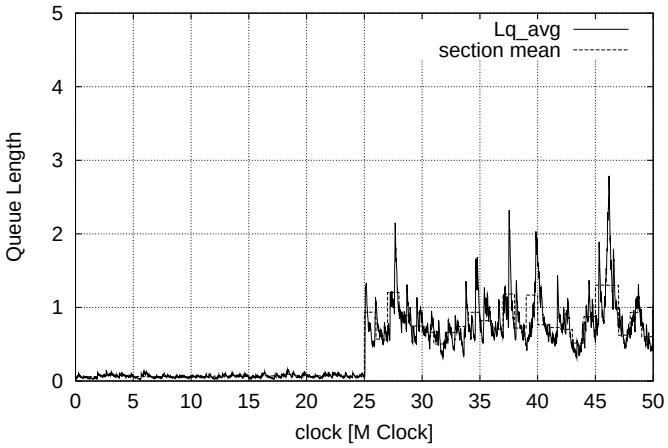

図 A.7: **w=0.0025**

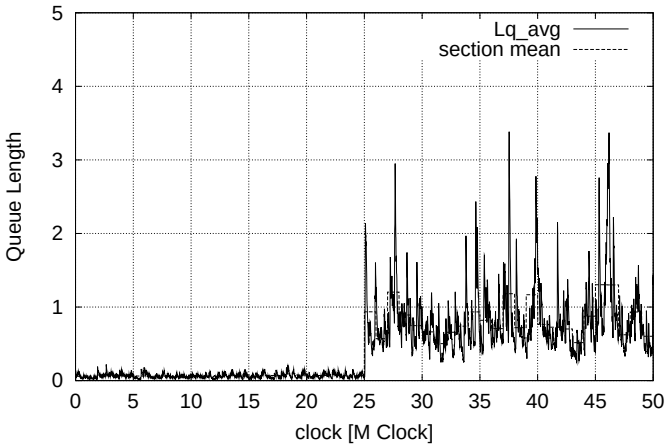

図 A.8: **w=0.005**

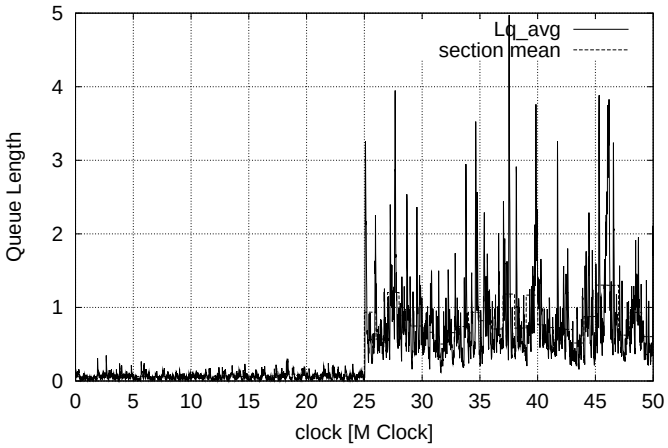

図 A.9: **w=0.01**

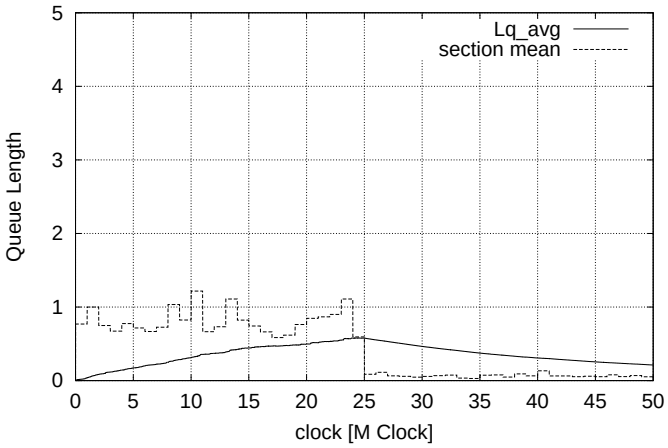

図 A.10: **w=0.000025**

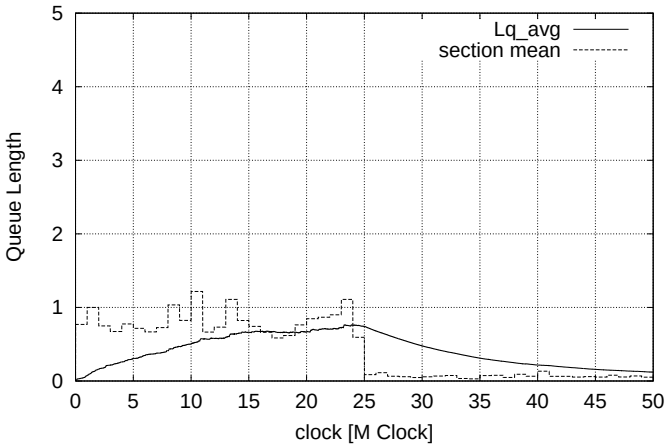

図 A.11: **w=0.00005**

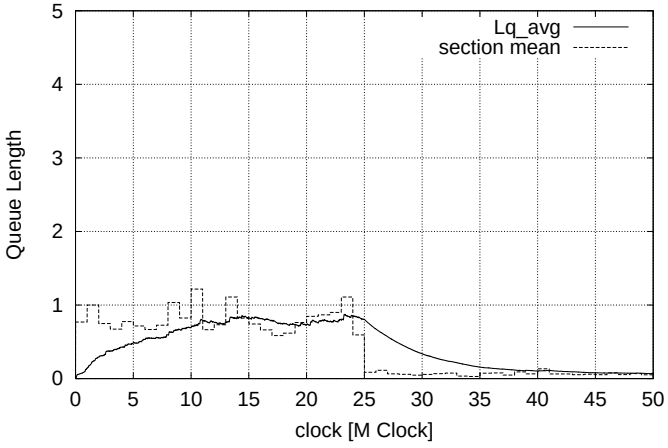

図 A.12: **w=0.0001**

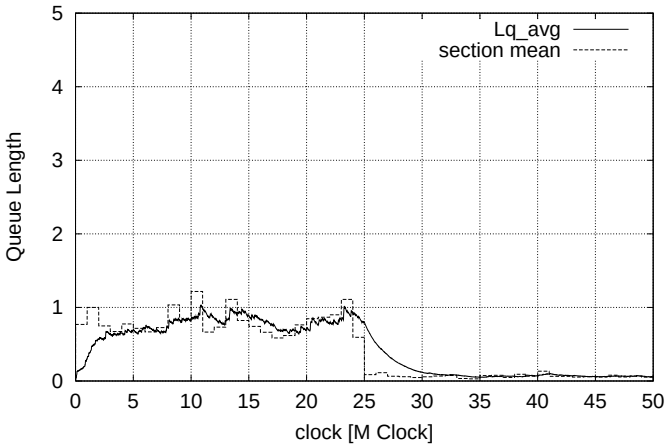

図 A.13: **w=0.00025**

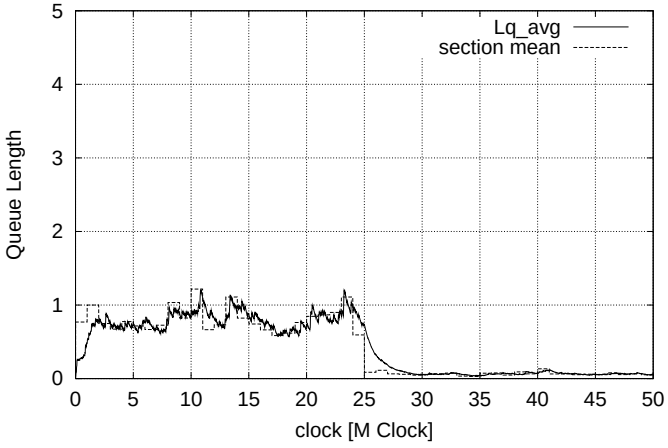

図 A.14: **w=0.0005**

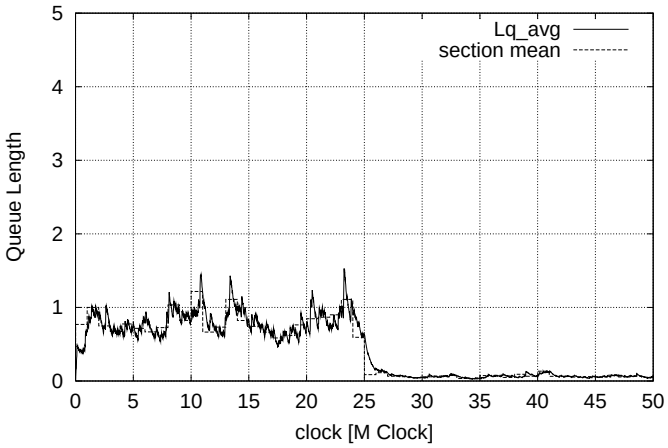

図 A.15: **w=0.001**

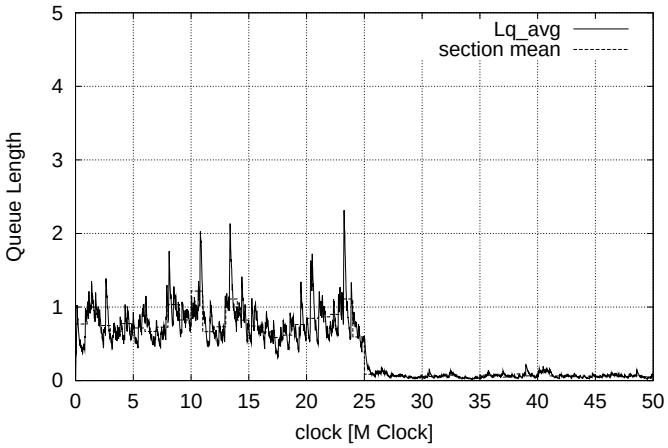

図 A.16: **w=0.0025**

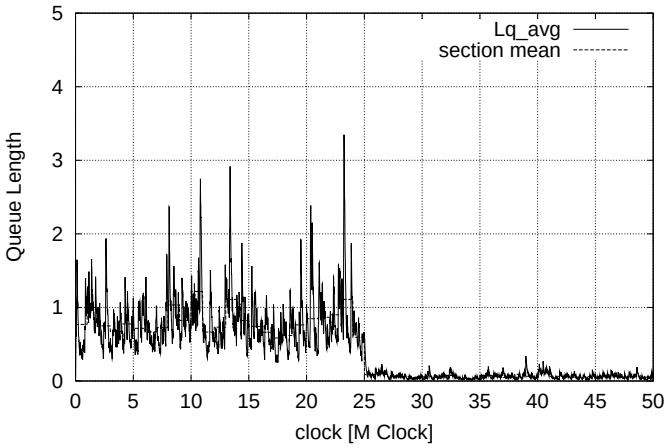

図 A.17: **w=0.005**

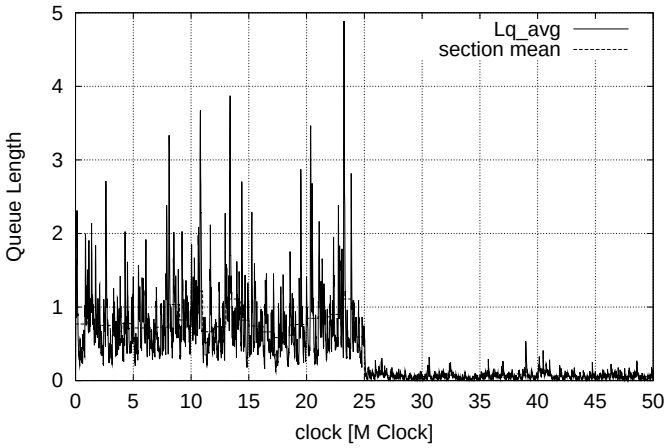

図 A.18: **w=0.01**