信州大学工学部

学士論文

サービス時間を固定した待ち行列システムの出発間隔と遅延について

指導教員　西新 幹彦 准教授

学科　電気電子工学科
学籍番号　13T2032B
氏名　小林 輝

2017 年 9 月 29 日

目次

1　序章

1.1　研究の背景

スマートフォンや携帯電話における緊急地震速報などでは情報を早く送受信する必要がある．しかし，伝送システムを用いて情報を送受信する場合，必ず遅延が生じる．そのため，伝送システムにおける遅延を小さくすることによって情報を早く送ることができる．一般に世の中で用いられている伝送システムは符号器と復号器が含まれているが，本研究では伝送システムの符号器と復号器の間で起きている現象を調べるため送信機からの出発間隔と遅延を測定し，その結果を考察する．従来研究 [1] では符号器と復号器を含む伝送システムにおいて符号器と復号器に分節木を用い，分節木の中間ノードに語頭符号を配置することで遅延を小さくできると示されている．

1.2　論文の構成

本論文では次のような構成をとる．第 2 章では本研究で想定した伝送システムの概要と出発間隔と遅延の定義を記す．第 3 章では出発間隔の分布に関する実験結果とその考察を述べる．第 4 章では遅延の分布に関する実験結果とその考察を述べる．第 5 章ではまとめと今後の課題を述べる．付録には，情報源の到着過程で使用したポアソン過程についての説明と実験で使用したプログラムのソースコードを掲載した．

2　伝送システムと出発間隔と遅延

図 1 のような情報源から符号器に情報が送られ，符号化された情報がバッファに到着し送信機によって送信され，復号器によって復号化された情報が出力される伝送システムを考えることができる．本研究ではこの伝送システムの符号器と復号器の間で起きている現象を調べるため図 2 のような伝送システムを考える．到着レート λ をもつポアソン過程にしたがってシンボルがバッファに到着する．シンボルはバッファに蓄えられ，送信機によって 1 シンボルずつ送信される．簡単のため，1 シンボルを送信するのにかかる時間を 1 秒とする．バッファが溢れないようにバッファのサイズは十分大きいとする．以上が本研究で考える伝送システムの概要である．

送信機からシンボルが送信され，次のシンボルが送信されるまでの時間を出発間隔と定義する．また，シンボルが情報源から出力されてから，そのシンボルが送信機から送信されるまでにかかる時間を遅延と定義する．

図 1　一般的な伝送システム

図 2　本研究で想定した伝送システム

3　出発間隔の分布

図 2 で記した伝送システムに基づいて出発間隔を測定するシミュレーション実験を行った．情報源シンボルはポアソン過程で 100 万回到着させた．送信機の送信レートを 1[シンボル/秒] と固定し、ポアソン過程の到着レート λ を 0.9，0.7，0.5，0.3，0.1 と変えて測定した．出発間隔の分布を図 3 に記す．

測定結果を見ると，出発間隔が 0[秒] 以上 1[秒] 未満となる事象は起こっておらず，出発間隔

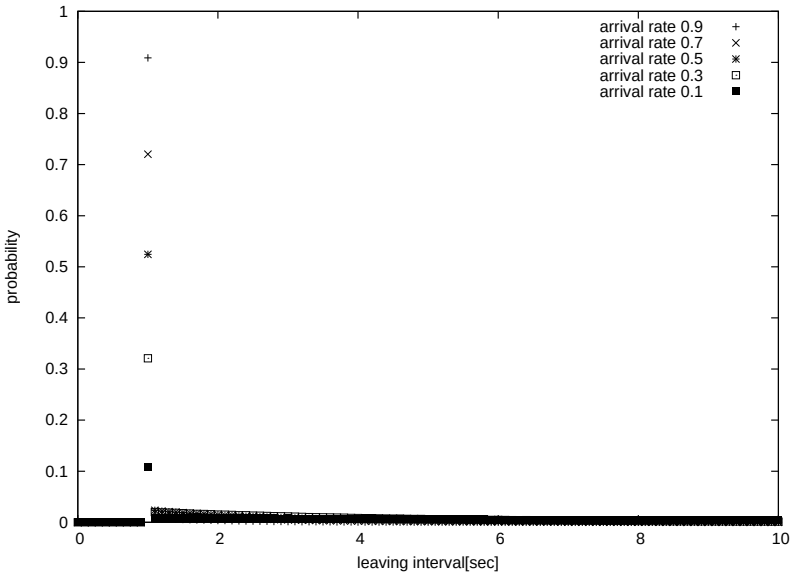

図 3　出発間隔の分布

が正確に 1[秒] となる確率はいずれのレートでも突出して大きい値となり，レートが大きいほど出発間隔が 1[秒] となる確率が大きくなっている．

ここで出発間隔がなぜ図 3 のように分布しているかを考察する．本研究では送信機の送信レートを 1[シンボル/秒] としているため出発間隔が 1[秒] 未満となる確率は 0 である．また，シンボルがバッファに到着したときバッファや送信機に先に到着したシンボルがある場合は今到着したシンボルの送信時間のみを考慮すればよいため，到着したシンボルとその前に送信されたシンボルの時間間隔すなわち出発間隔は 1[秒] となる．そのため，到着レート λ が大きくなるにつれてシンボルがバッファに到着する間隔が短くなり，出発間隔が 1[秒] になる確率が大きくなっている．

また，シンボルがバッファに到着したときバッファや送信機にシンボルがない場合は，1 個前に到着したシンボルが送信完了となり送信機が止まっていた時間と到着シンボルの送信時間 1[秒] を足したものが出発間隔となるため出発間隔が 1[秒] よりも大きくなる．そのため，出発間隔からシンボルの送信時間 1[秒] を引いた値は送信機の休止時間である．この分布を到着レート λ ごとに図 4 にまとめた．図 4 を見ると送信機の休止時間の分布は指数分布のようなグラフとなっている．そこで実際に指数分布であるか検証していく．送信機の休止時間がレート η の指数分布に従うと仮定する．送信機から送信されるシンボルの時間間隔が $1/\lambda$ に等しいため，出発間隔が 1[秒] となる確率を Q とすると

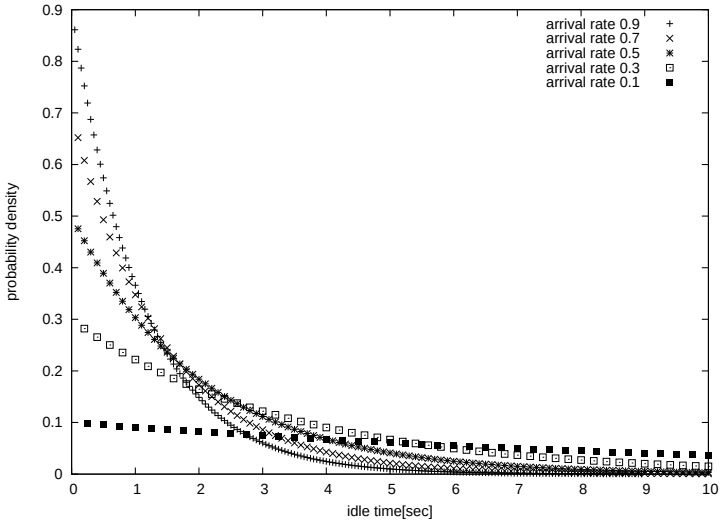

図 4　バッファの休止時間の分布

$$1/\lambda = Q \times 1 + (1-Q)(1/\eta + 1) \tag{1}$$

が成り立つ．式 (1) より

$$\eta = \frac{(1-Q)\lambda}{1-\lambda} \tag{2}$$

が求められる．シミュレーション実験によりそれぞれの到着レート λ における Q を測定し，到着レート λ における η の値を求め，図 5〜9 に送信機の休止時間の分布とレート η の指数分布のグラフをまとめた．どの到着レート λ においても 2 つのグラフは一致したため，送信機の休止時間の分布はレート η の指数分布となることが分かった．そのため，出発間隔が 1[秒] より大きい範囲においては出発間隔の分布は送信機の休止時間の分布にシンボルの送信時間 1[秒] を足したものとなることが分かった．

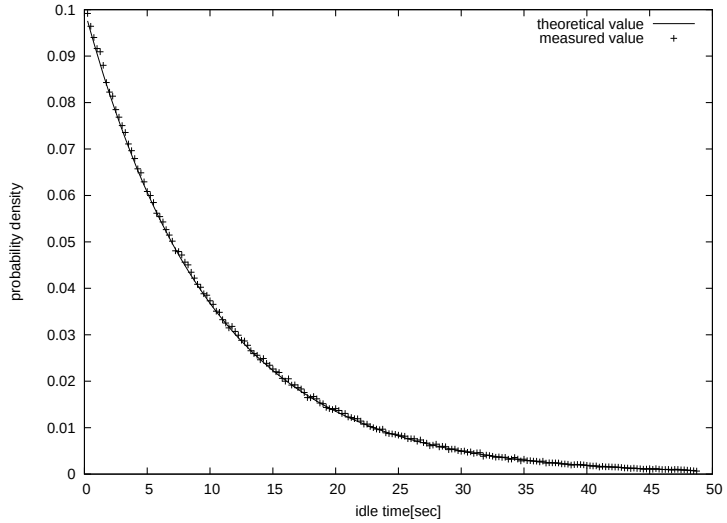

図 5　λ=0.1 のときのバッファの休止時間の分布

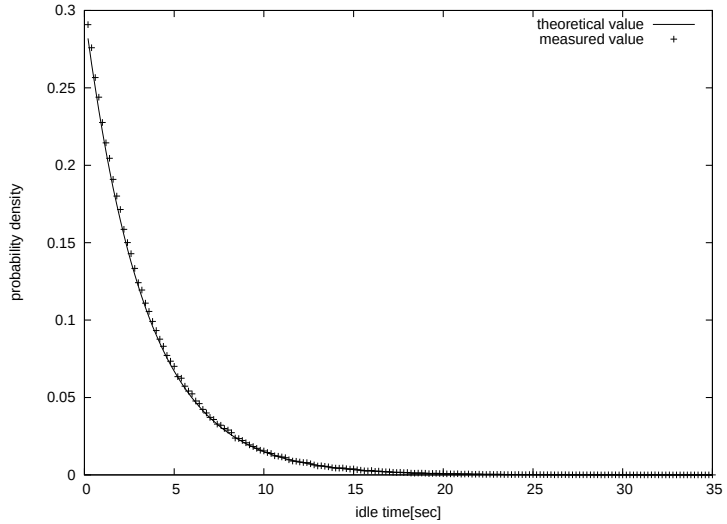

図 6　λ=0.3 のときのバッファの休止時間の分布

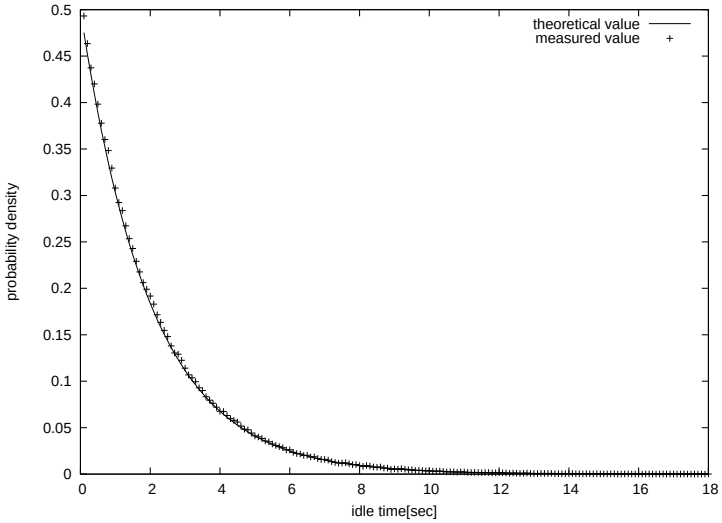

図 7　λ=0.5 のときのバッファの休止時間の分布

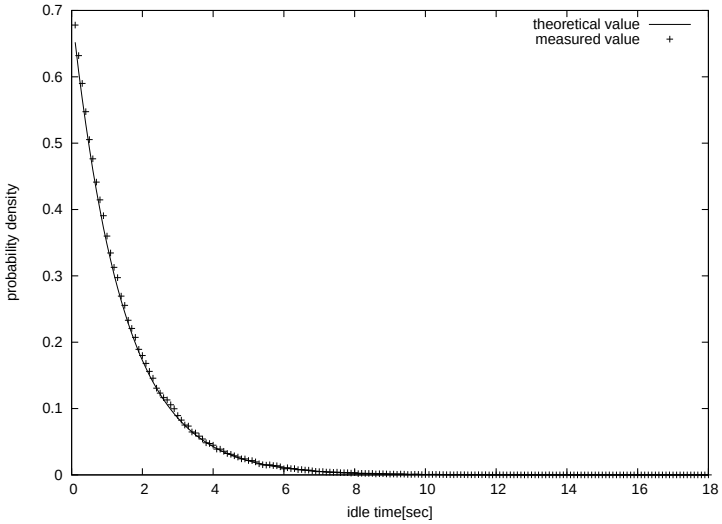

図 8　λ=0.7 のときのバッファの休止時間の分布

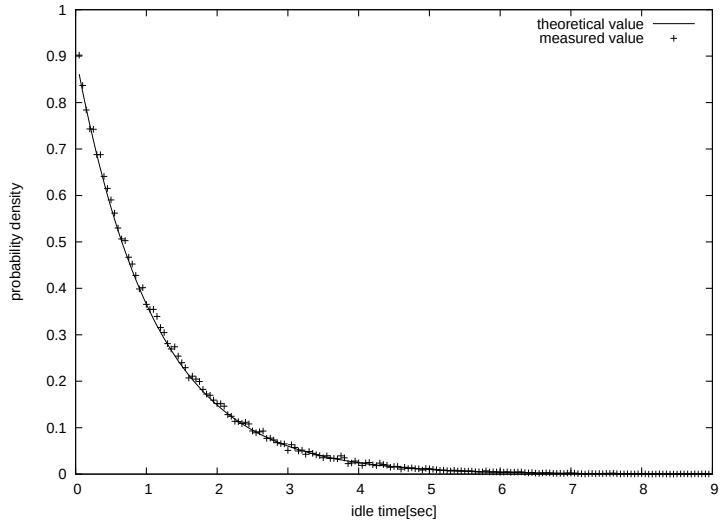

図 9　λ=0.9 のときのバッファの休止時間の分布

4　遅延の分布

図 2 で記した伝送システムに基づいて遅延を測定するシミュレーション実験を行った．情報源シンボルはポアソン過程で 100 万回到着させた．送信機の送信レートを 1[シンボル/秒] と固定し、ポアソン過程の到着レート λ を 0.9，0.7，0.5，0.3，0.1 と変えて測定した．遅延の分布を図 10 に記す．測定結果を見ると，遅延が 0[秒] 以上 1[秒] 未満となる事象は起こっておらず，遅延が正確に 1[秒] となる確率はいずれのレートでも大きい値となり，レートが小さいほど遅延が 1 [秒] となる確率が大きくなっている．

ここで遅延がなぜ図 10 のように分布しているかを考察する．本研究では送信機の送信レートを 1[シンボル/秒] としているため遅延が 1[秒] 未満となる確率は 0 である．また，シンボルがバッファに到着したときバッファや送信機にシンボルがない場合は到着したシンボルの送信時間のみを考慮すればよいため，シンボルがバッファに到着してから送信完了となるまでにかかる時間すなわち遅延は 1[秒] となる．そのため，到着レート λ が小さくなるにつれてシンボルがバッファに到着する間隔が長くなり，バッファや送信機が空の状態のときにシンボルが到着する事象が起こりにくくなり，遅延が 1[秒] になる確率が大きくなっている．

また，図 10 の遅延が 1[秒] より大きい範囲だけ表したものを図 11 に記す．図 11 より，遅延が 1[秒] より大きく 2[秒] 未満の範囲では遅延の増加に伴い確率も増加している．また，遅延が 2[秒] のとき確率は不連続となり，遅延が 2[秒] より大きく 3[秒] 未満の範囲では上に凸の曲線となった．遅延が 3[秒] のときでは曲線がなめらかでないように見え，遅延が 3[秒] より

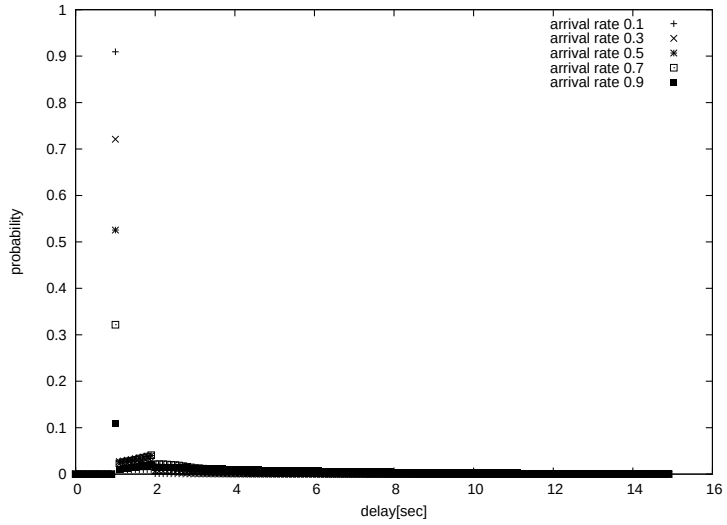

図 10　遅延の分布

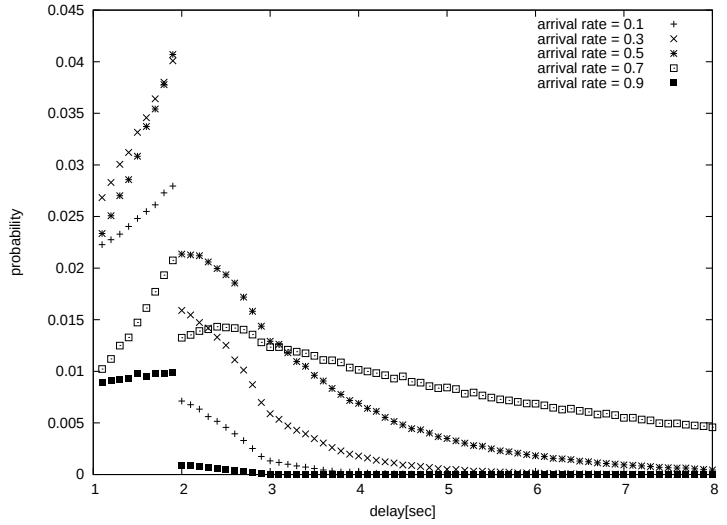

図 11　遅延の分布

大きいの範囲では遅延の増加に伴い確率は減少した.

ここで，遅延が 1[秒] より大きい範囲ではなぜこのような分布になったのか考察していく. シンボルがバッファに到着したときバッファや送信機に先に到着したシンボルがある場合は，今到着したシンボルの送信時間 1[秒] とバッファにあるシンボルの送信時間と送信中のシンボルの送信残り時間を足したものが到着シンボルの遅延となるため遅延が 1[秒] よりも大きくな

る．送信中のシンボルの送信残り時間は 0[秒] より大きく 1[秒] 以下であるから，シンボルがバッファに到着したときバッファが空で送信機にシンボルが 1 個ある場合は，到着したシンボルの送信時間と送信中のシンボルの送信残り時間を足したものが到着したシンボルの遅延となり，この遅延は 1〜2[秒] の間に収まる．また，シンボルがバッファに到着したとき送信中のシンボル 1 個とバッファにシンボルが 1 個ある場合は，到着したシンボルの送信時間とバッファにある 1 個のシンボルの送信時間 1[秒] と送信中のシンボルの送信残り時間を足したものが到着したシンボルの遅延となり，この遅延は 2〜3[秒] の間に収まる．一般に，シンボルがバッファに到着したとき送信中のシンボル 1 個とバッファにシンボルが $n-1$ 個ある場合は，到着したシンボルの送信時間とバッファにある $n-1$ 個のシンボルの送信時間 $n-1$[秒] と送信中のシンボルの送信残り時間を足したものが到着したシンボルの遅延となり，この遅延は $n \sim n+1$[秒] の間に収まる．このように，シンボルがバッファに到着したときのバッファと送信機のシンボルの数によって遅延は互いに素な範囲の値を取り，図 11 のように遅延は整数のときで不連続になったりなめらかでないように見えたりすると考えられる．

図 12 にバッファと送信機にあるシンボルの数の遷移を示す．図 12 の数字はシンボルの数を表している．シンボルの数が 0 個から 1 個になり次のシンボルが到着する場合と 2 個から 1 個になり次のシンボルが到着する場合では送信中のシンボルの送信残り時間の分布は等しいが，シンボルの数が 1 個から 2 個になり次のシンボルが到着する場合と 3 個から 2 個になり次のシンボルが到着する場合では送信中のシンボルの送信残り時間の分布は異なるため 3’，3” として区別している．同様に，シンボルの数が 1 個から 2 個になり 3 個になって次のシンボルが到着する場合と 3 個から 2 個になり 3 個になって次のシンボルが到着する場合と 4 個から 3 個になり次のシンボルが到着する場合とでは送信中のシンボルの送信残り時間の分布はそれぞれ異なるため 4’，4”，4”’ として区別している．ここで，2’ の状態からシンボルが 1 個減ると 1 の状態へ，3’ や 3” の状態からシンボルが 1 個減ると 2 の状態へ，4’ や 4”，4”’ の状態からシンボルが 1 個減ると 3 の状態へ移るが，図を見やすくするため矢印を省略している．また，シンボルの送信中にシンボルが 1 個到着した場合のバッファにシンボルが到着したときの送信中のシンボルの送信残り時間の分布 $f_1(x)$，1 個のシンボルの送信中にシンボルが 2 個到着した場合の送信残り時間の分布 $f_2(x)$，1 個のシンボルの送信中にシンボルが 3 個到着した場合の送信残り時間の分布 $f_3(x)$ とする．

送信残り時間の分布 $f_1(x)$，$f_2(x)$，$f_3(x)$ は次のように求めることができる．

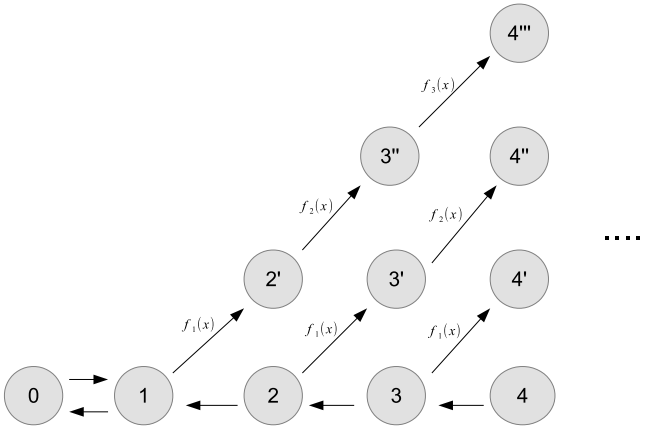

図 12　バッファの中のシンボル数の遷移

4.1　1 個のシンボルの送信中にシンボルが 1 個到着した場合の送信残り時間の分布

まず，1 個のシンボルの送信中にシンボルが 1 個到着した場合を考える．シンボルはポアソン到着であるため情報源からバッファにシンボルが到着する間隔は指数分布

$$f(x) = \lambda e^{-\lambda x} \qquad (0 \leq x) \tag{3}$$

に従う．しかし，送信機の送信時間は 1 秒であるためバッファにシンボルが到着したときの送信中のシンボルが送信開始してから経過した時間の分布 $f_1'(x)$ は $f(x)$ を 0 から 1 の範囲で正規化することで求められ

$$f_1'(x) = \frac{\lambda e^{-\lambda x}}{\int_0^1 \lambda e^{-\lambda x} dx} = \frac{\lambda e^{-\lambda x}}{1 - e^{-\lambda}} (0 \leq x \leq 1) \tag{4}$$

となる．よって，1 個のシンボルの送信中にシンボルが 1 個到着した場合の送信残り時間の分布 $f_1(x)$ は

$$f_1(x) = \frac{\lambda e^{-\lambda(1-x)}}{1 - e^{-\lambda}} (0 \leq x \leq 1) \tag{5}$$

となる．ここで，図 12 よりバッファが空で送信機にシンボルが 1 個あるときに次のシンボルが到着した場合の送信残り時間の分布は $f_1(x)$ の 1 つだけである．図 13〜17 にバッファと送信機にシンボルが 1 個あるときの到着レート λ ごとの送信残り時間の測定値の分布と式 (5) の分布を示す．図 13〜17 より測定値は式 (5) にほぼ従っているため，1 個のシンボルの送信中にシンボルが 1 個到着した場合の送信残り時間の分布は式 (5) と表されることが検証された．

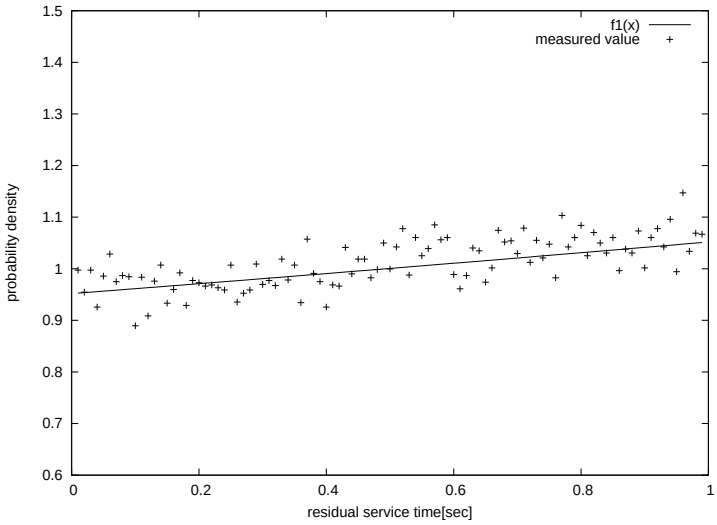

図 13　バッファのシンボル 1 個　λ=0.1 のときの送信残り時間の分布

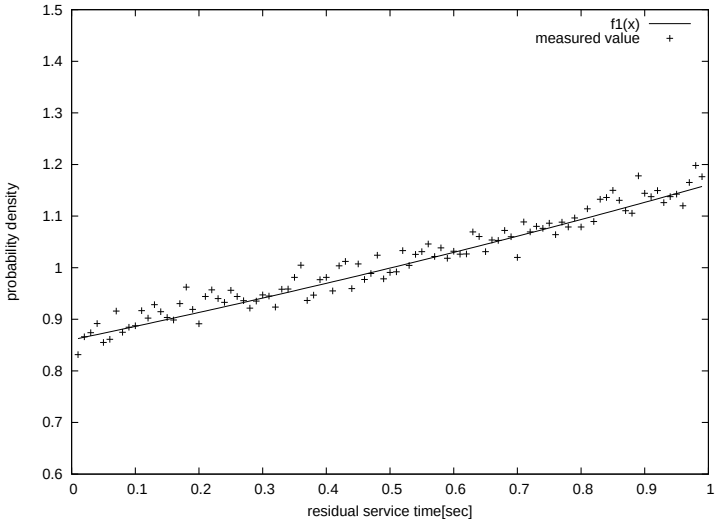

図 14　バッファのシンボル 1 個　λ=0.3 のときの送信残り時間の分布

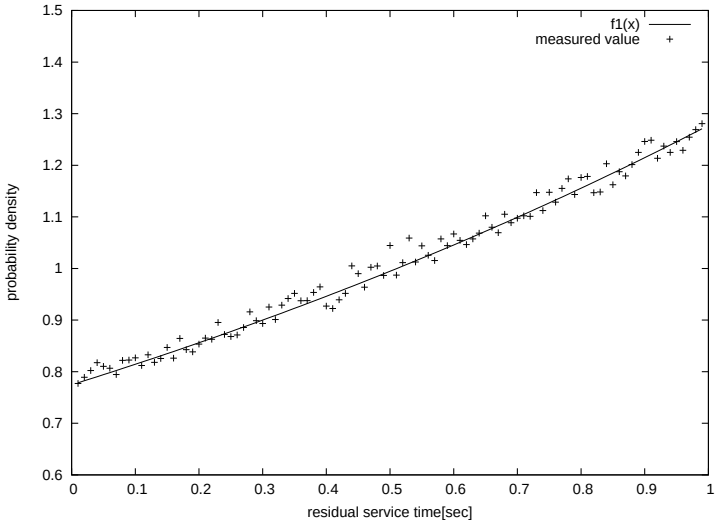

図 15　バッファのシンボル 1 個　λ=0.5 のときの送信残り時間の分布

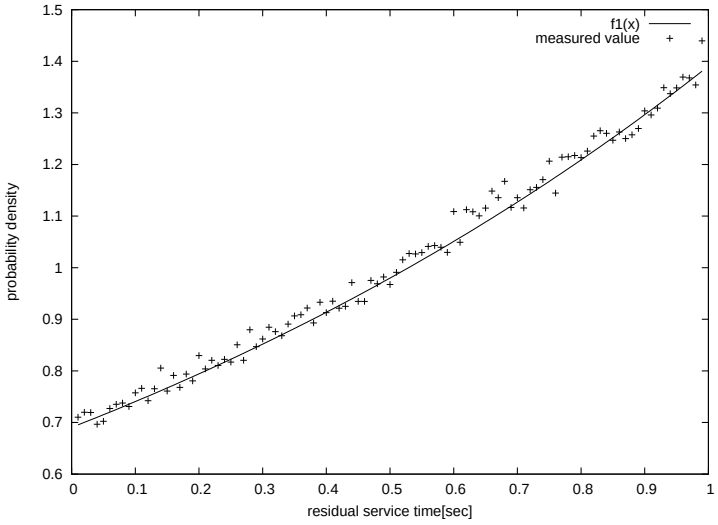

図 16　バッファのシンボル 1 個　λ=0.7 のときの送信残り時間の分布

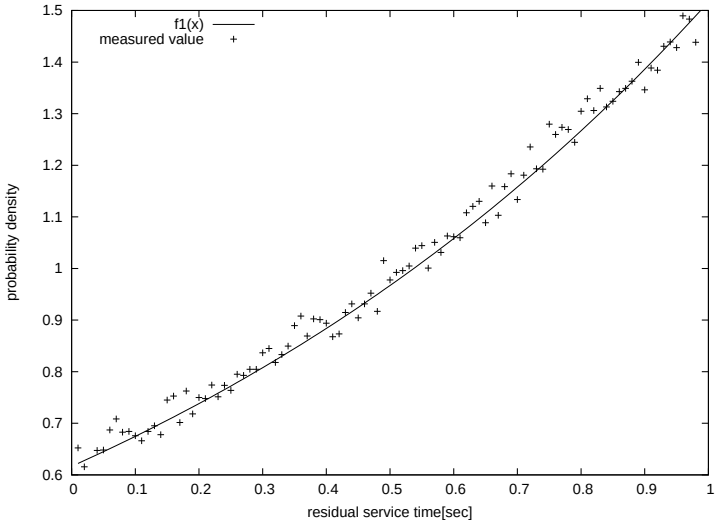

図 17　バッファのシンボル 1 個　λ=0.9 のときの送信残り時間の分布

4.2 1 個のシンボルの送信中にシンボルが 2 個到着した場合の送信残り時間の分布

次に 1 個のシンボルの送信中にシンボルが 2 個到着した場合を考える．まず 1 個のシンボルの送信中にシンボルが 1 個到着した場合の送信残り時間の分布は $f_1(x)$ であるから，1 個のシンボルの送信中にシンボルが 1 個到着した場合の送信残り時間よりも次のシンボルが到着する時間が短くなるように式 (2) と式 (4) を畳み込み積分すると，1 個のシンボルの送信中にシンボルが 2 個到着した場合の送信残り時間の分布 $f_2'(x)$ は

$$f_2'(x) = \int_x^1 f_1(x+u)f(u)du = \frac{(1-x)\lambda^2 e^{-\lambda(1-x)}}{1-e^{-\lambda}}(0 \leq x) \tag{6}$$

となる．ここで送信機の送信時間は 1 秒であるから $f_2'(x)$ を $(0 \leq x \leq 1)$ の範囲で正規化すると 1 個のシンボルの送信中にシンボルが 2 個到着した場合の送信残り時間の分布

$$f_2(x) = \frac{f_2'(x)}{\int_0^1 f_2'(x)dx} = \frac{(1-x)\lambda^2 e^{-\lambda(1-x)}}{1-(1+\lambda)e^{-\lambda}}(0 \leq x \leq 1) \tag{7}$$

が得られる．ここで，図 12 よりバッファと送信機にシンボルが 2 個あるときに次のシンボルが到着した場合の送信残り時間の分布は $f_1(x)$ と $f_2(x)$ の 2 通りあるため，シミュレーション実験により $f_1(x)$ と $f_2(x)$ の比率を求め，$f_1(x)$ と $f_2(x)$ をそれぞれの比で合成することによりバッファと送信機にシンボルが 2 個あるときの送信残り時間の分布を求めることができる．図 18〜22 はバッファと送信機にシンボルが 2 個あるときの到着レート λ ごとの送信残り時間の測定値の分布と計算値の分布を示しており，測定値は計算値に従うことがわかった．

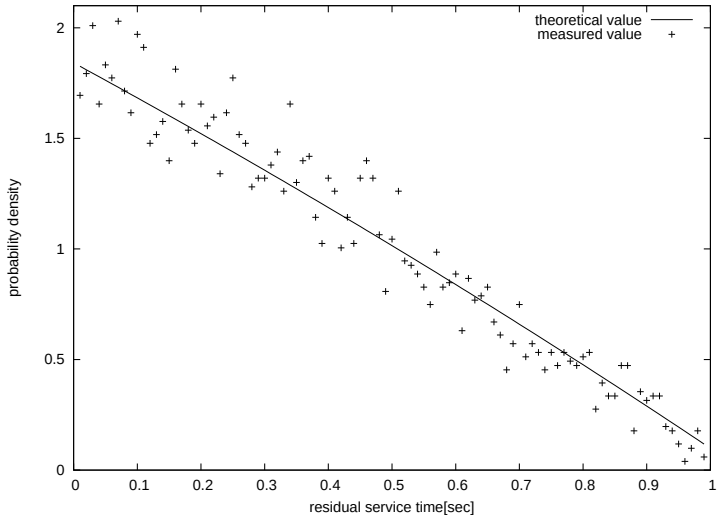

図 18　バッファのシンボル 2 個　λ=0.1 のときの送信残り時間の分布

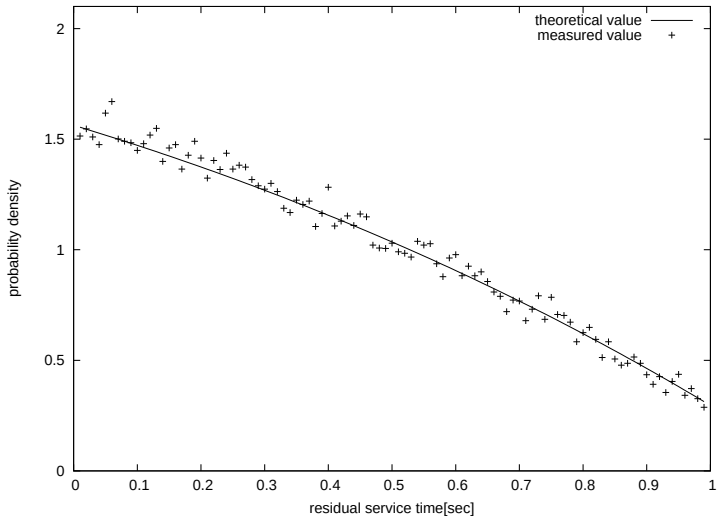

図 19　バッファのシンボル 2 個　λ=0.3 のときの送信残り時間の分布

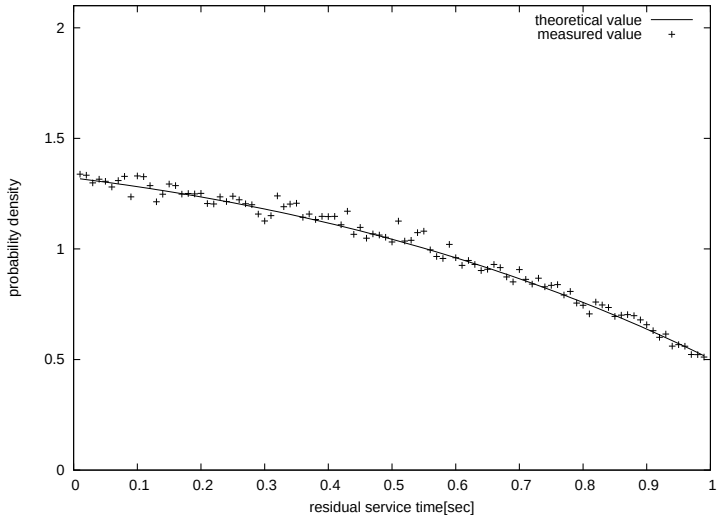

図 20　バッファのシンボル 2 個　λ=0.5 のときの送信残り時間の分布

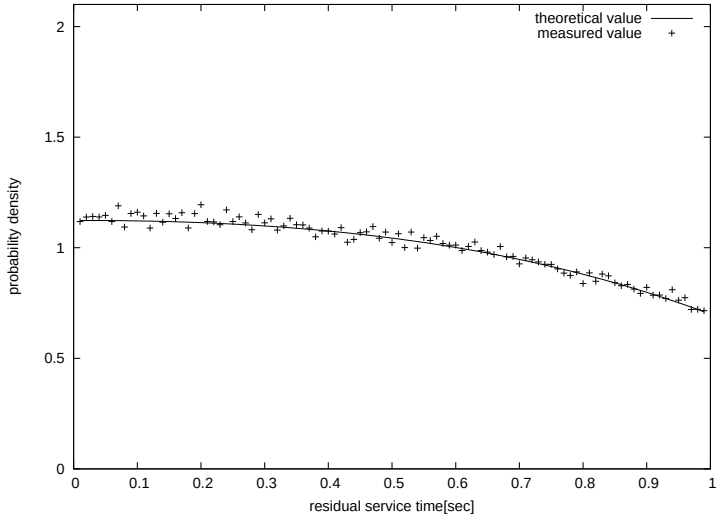

図 21　バッファのシンボル 2 個　λ=0.7 のときの送信残り時間の分布

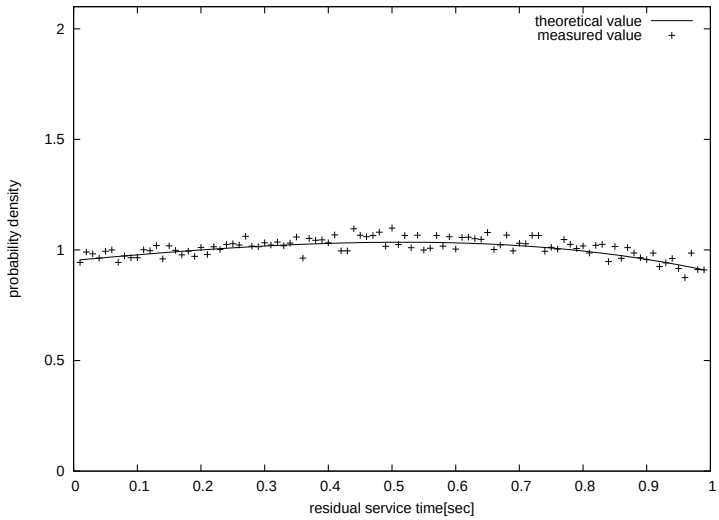

図 22　バッファのシンボル 2 個　λ=0.9 のときの送信残り時間の分布

4.3　1 個のシンボルの送信中にシンボルが 3 個到着した場合の送信残り時間の分布

次に 1 個のシンボルの送信中にシンボルが 3 個到着した場合を考える．まず 1 個のシンボルの送信中にシンボルが 2 個到着した場合の送信残り時間の分布は $f_2(x)$ であるから，1 個のシンボルの送信中にシンボルが 3 個到着した場合の送信残り時間よりも次のシンボルが到着する時間が短くなるように式 (2) と式 (6) を畳み込み積分すると 1 個のシンボルの送信中にシンボルが 3 個到着した場合の送信残り時間の分布 $f_3'(x)$ は

$$f_3'(x) = \int_x^1 f_2(x+u)f(u)du = \frac{(x^2-2x+1)\lambda^3 e^{-\lambda(1-x)}}{2(1-(1+\lambda)e^{-\lambda})}(0 \leq x) \tag{8}$$

となる．ここで送信機の送信時間は 1 秒であるから $f_3'(x)$ を $(0 \leq x \leq 1)$ の範囲で正規化すると 1 個のシンボルの送信中にシンボルが 3 個到着した場合の送信残り時間の分布

$$f_3(x) = \frac{f_3'(x)}{\int_0^1 f_3'(x)dx} = \frac{(x^2-2x+1)\lambda^3 e^{-\lambda(1-x)}}{2-(\lambda^2+2\lambda+2)e^{-\lambda}}(0 \leq x \leq 1) \tag{9}$$

が得られる．ここで，図 12 よりバッファと送信機にシンボルが 3 個あるときに次のシンボルが到着した場合の送信残り時間の分布は $f_1(x)$ と $f_2(x)$ と $f_3(x)$ の 3 通りあるため，シミュレーション実験により $f_1(x)$ と $f_2(x)$ と $f_3(x)$ の比率を求め，$f_1(x)$ と $f_2(x)$ と $f_3(x)$ をそれぞれの比で合成することによりバッファと送信機にシンボルが 3 個あるときの送信残り時間の

分布を求めることができる．図 23〜27 はバッファと送信機にシンボルが 3 個あるときの到着レート λ ごとの送信残り時間の測定値の分布と計算値の分布を示しており，測定値は計算値に従うことがわかった．

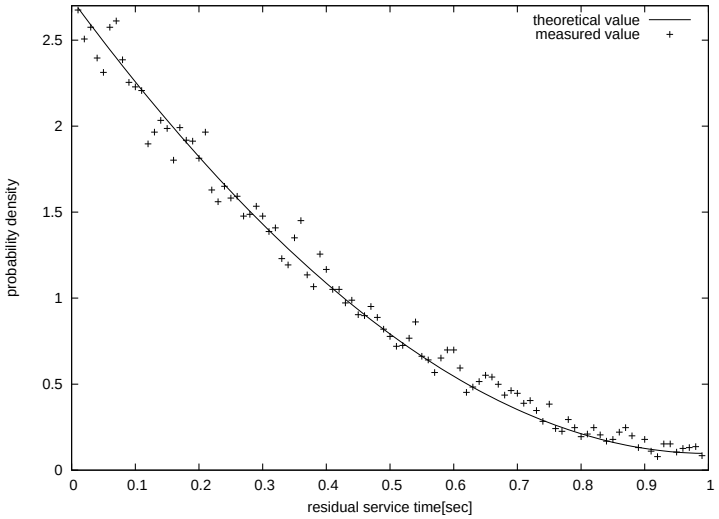

図 23　バッファのシンボル 3 個　λ=0.1 のときの送信残り時間の分布

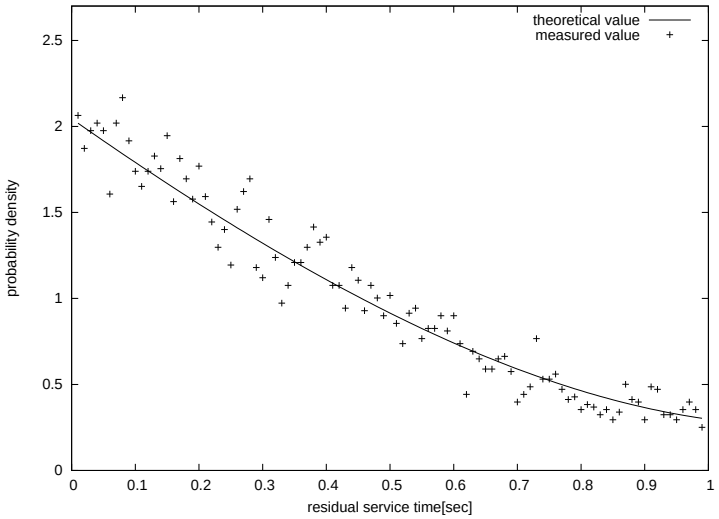

図 24　バッファのシンボル 3 個　λ=0.3 のときの送信残り時間の分布

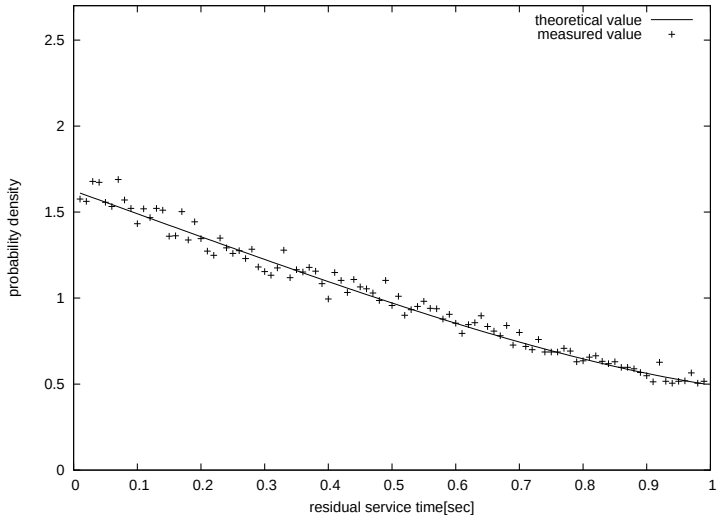

図 25　バッファのシンボル 3 個　λ=0.5 のときの送信残り時間の分布

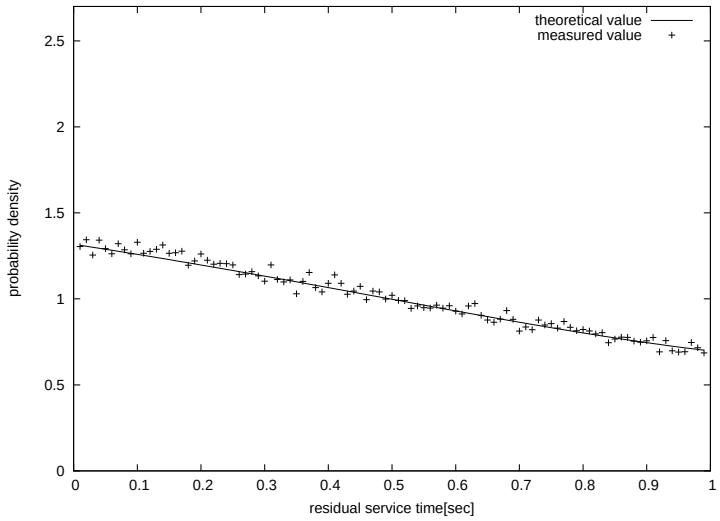

図 26　バッファのシンボル 3 個　λ=0.7 のときの送信残り時間の分布

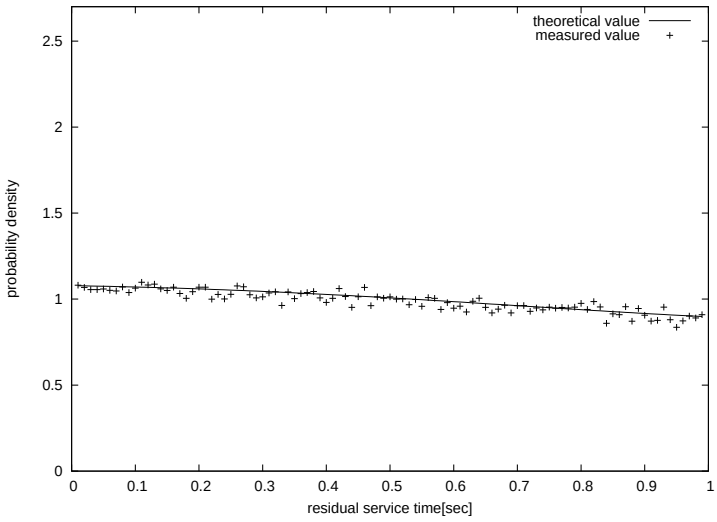

図 27　バッファのシンボル 3 個　λ=0.9 のときの送信残り時間の分布

上記より，シンボルが到着したときにバッファのシンボル数が 1〜3 個となるときの遅延を求め，測定値とともに図 28〜32 に示す．このように，シンボルがバッファに到着したときのバッファと送信機のシンボル数で場合分けして考えることでの送信中のシンボルの送信残り時間が求められ，これにより遅延を求められることが分かった．

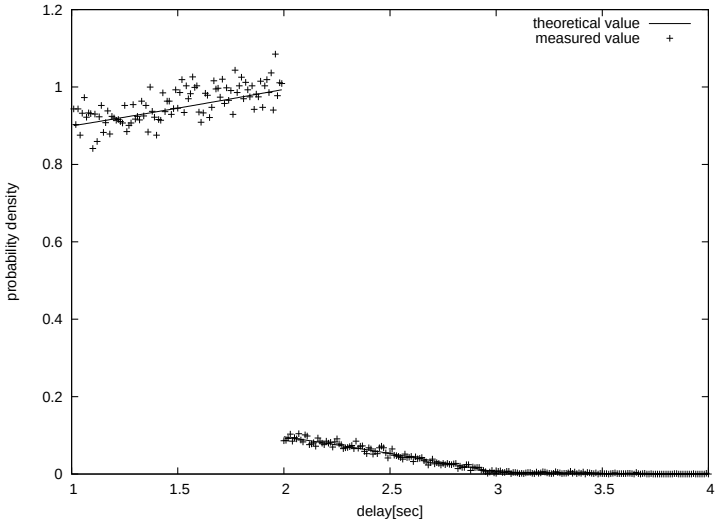

図 28　λ=0.1 のときの遅延の分布

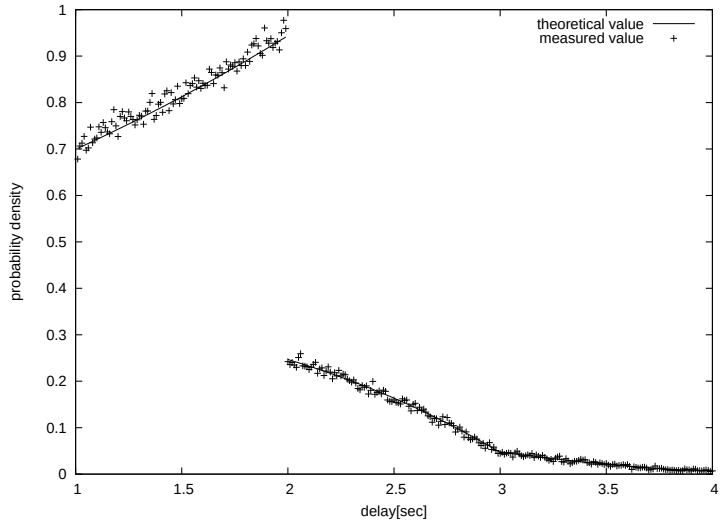

図 29　λ=0.3 のときの遅延の分布

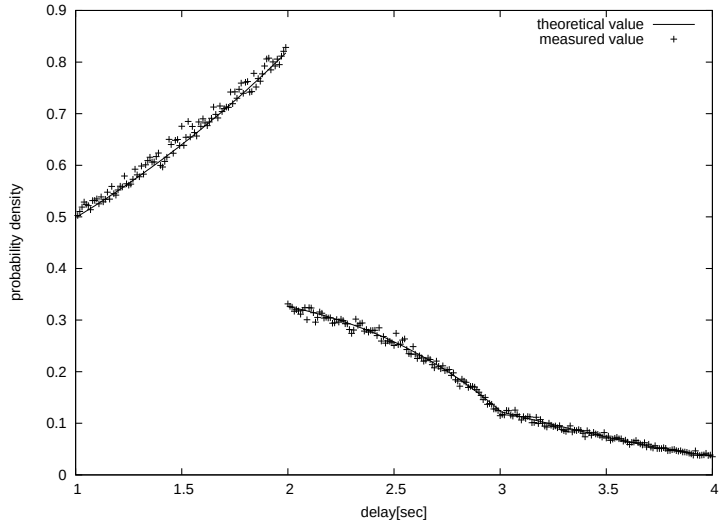

図 30　λ=0.5 のときの遅延の分布

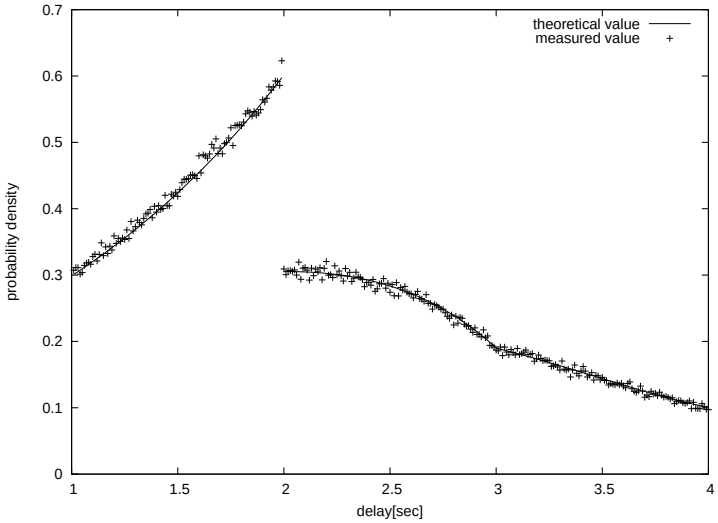

図 31　λ=0.7 のときの遅延の分布

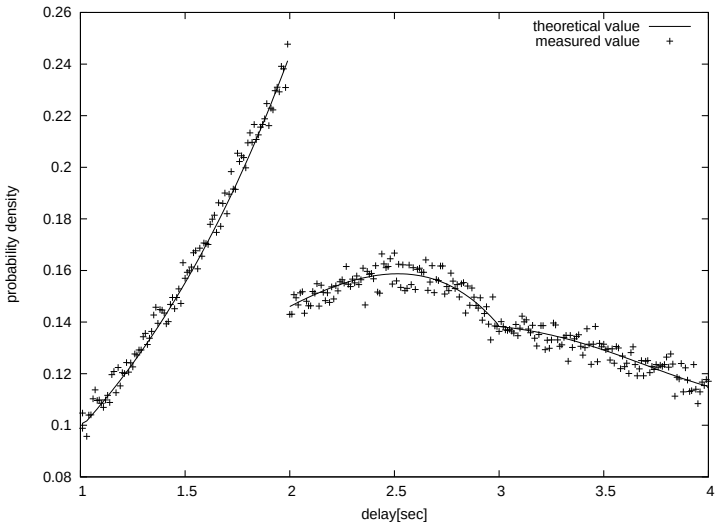

図 32　λ=0.9 のときの遅延の分布

5　まとめと今後の課題

送信機の送信レートを 1[シンボル/秒] に固定した場合の符号器と復号器を除いた伝送システムすなわち待ち行列システムの出発間隔と遅延の分布をプログラムによるシミュレーション実験により測定し，理論的に考察した．出発間隔から送信にかかる時間 1[秒] を引いた値は送信機の休止時間と等しく，送信機の休止時間は指数分布に従うことが分かった．また，シンボルがバッファに到着したときのバッファと送信機のシンボル数で場合分けして考えることで送信中のシンボルの送信残り時間が求められ，これにより遅延を求められることが分かった．また，本研究ではバッファが溢れないようにバッファのサイズを十分に大きくして実験を行ったが，バッファなしの場合とバッファ容量が 1 個の場合でも同様に実験を行い測定結果と考察をそれぞれ付録 B，付録 C に記す．　今後は，本研究で得られた待ち行列システムの特性を基に符号器と復号器を含む伝送システムの遅延について考察することが課題である．

謝辞

本研究を終えるにあたり，終始一貫して丁寧なご指導を賜りました指導教員の西新幹彦先生に心より感謝の意を申し上げます．

参考文献

[1] 荻野真志，「遅延特性改善のための分節木上の符号器の配置に関する一考察」，信州大学工学部，学士論文，2013.

[2] 奥村晴彦，C 言語による最新アルゴリズム事典，技術評論社，1991.

付録 A　ポアソン過程

ポアソン到着とは，ランダムに到着するシンボルを表す確率過程の一つである．ポアソン到着のシンボルの到着間隔は指数分布に従う．ポアソン到着は以下の 3 つの性質をもつ．

・定常性

$t>0$，$x\leq 0$ となる任意の実数 t，任意の整数 x に対し，時間間隔 $(a, a+t)$ の間に x 人到着する確率は，すべての a について同一で，$v_x(t)$ と表すことができる．つまり，時間区間の幅が同じであれば，どこをとってもシステムの確率的状態は同じである．この性質を定常性という．

・独立性

上記に記した $v_x(t)$ は，時刻 a までにどれだけきたか，またいつきたかには無関係である．もし前に到着したシンボルの時刻や個数によってこの確率 $v_x(t)$ が影響を受けるならば，前のシンボルはあとに残る影響，つまり残留効果をもつことになるが，そういうことがないということがこの独立性である．

・希少性

$\psi(t) = \sum_{x=2}^{\infty} v_x(t)$ と置くとき，t が十分小さければ $\psi(t)$ は無視してよい．つまり，

$$\psi(t) = o(t)(t \to 0) \tag{10}$$

が成り立つ．この式は

$$\lim_{t\to\infty} \frac{\psi(t)}{t} = 0 \tag{11}$$

を意味する．$o(t)$ とは t が十分小さければ無視してよい程度であることを示している．つまり，$\psi(t) = o(t)$ は 2 つ以上のシンボルが同時に連れ立ってこないことを意味する．

付録 B　バッファがない場合の出発間隔と遅延について

図 2 で記した伝送システムのバッファを取り除いたシステムにおいて出発間隔と遅延について考察する．情報源シンボルはポアソン過程で 100 万回到着させた．送信機の送信レートを 1[シンボル/秒] と固定する．バッファがないため，送信機が動いているときにシンボルが到着すると到着シンボルは取りこぼされることとなる．まず，出発間隔について考える．シンボルが送信中に次のシンボルが到着した場合は到着シンボルは取りこぼされるためこの場合は出発間隔は考慮しないこととする．送信機が止まっているときに次のシンボルが到着した場合は送信機の休止時間と送信レート 1[秒] を足したものが出発間隔となる．シンボルが送信完了し

たときから見ると次のシンボルの到着はポアソン到着であるから，送信機の休止時間の分布はポアソン分布と等しくなる．図 33〜35 にバッファなしのときの出発間隔の測定値と理論値を示す．図 33〜35 より測定値と理論値がほぼ同値となったため，バッファなしのときの出発間隔は送信機の休止時間と送信レート 1[秒] を足したものとなり，送信機の休止時間はレート λ の指数分布に従うことがわかった．次に遅延について考える．シンボルが送信中に次のシンボルが到着した場合は到着シンボルは取りこぼされるためこの場合は遅延は考慮しないこととする．送信機が止まっているときに次のシンボルが到着した場合は送信レートが 1[秒] であるため遅延は 1[秒] となる．これよりバッファを取り除いたシステムでは遅延は必ず 1[秒] となる．

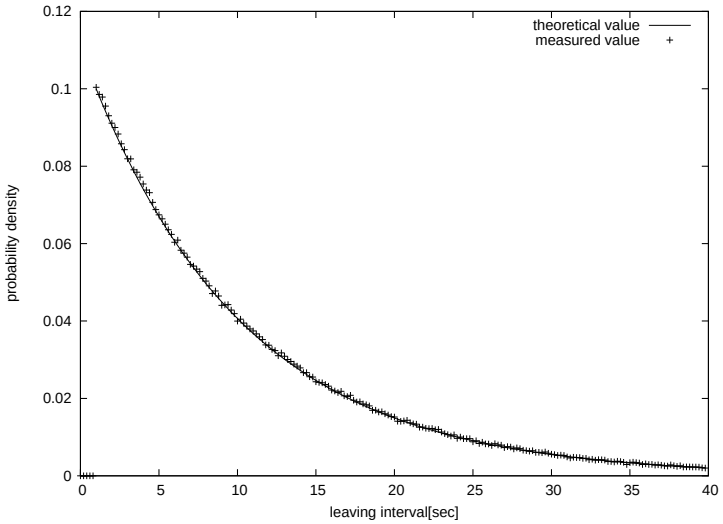

図 33　バッファなし　λ=0.1 のときの出発間隔の分布

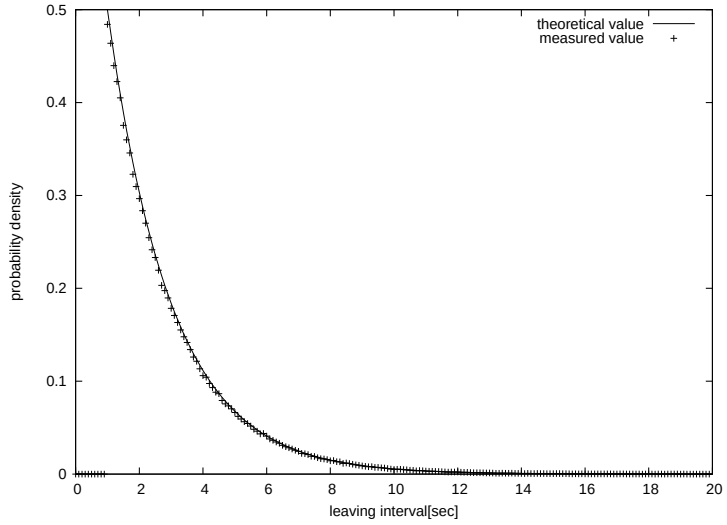

図 34　バッファなし　λ=0.5 のときの出発間隔の分布

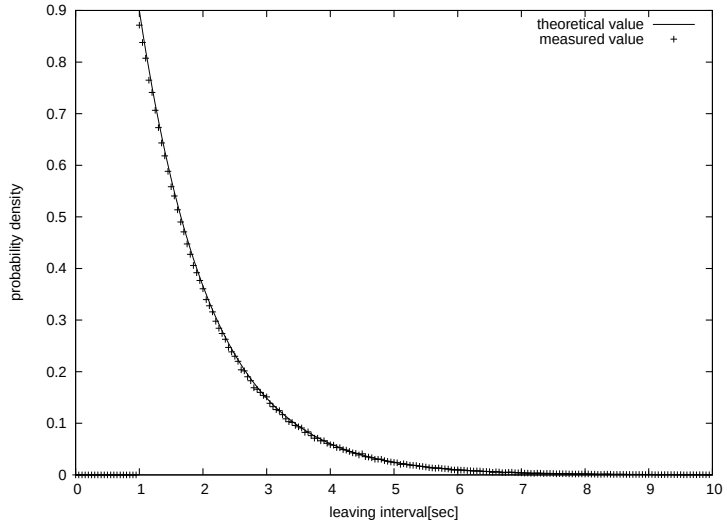

図 35　バッファなし　λ=0.9 のときの出発間隔の分布

付録 C　バッファ容量が 1 個の場合の出発間隔と遅延について

図 2 で記した伝送システムでバッファ容量を 1 個にした場合の出発間隔と遅延について考察する．情報源シンボルはポアソン過程で 100 万回到着させた．送信機の送信レートを 1[シンボル/秒] と固定する．まず，出発間隔について考える．シンボルが送信中に次のシンボルが到着した場合は，送信中のシンボルが送信完了してから到着シンボルが送信完了するまでの時間すなわち出発間隔は 1[秒] となる．送信機が止まっているときに次のシンボルが到着した場合は送信機の休止時間と送信レート 1[秒] をたしたものが出発間隔となる．バッファ容量が 1 個の場合でも送信機の休止時間の分布は指数分布に従い，そのレート η は取りこぼされずに送信されたシンボルで式 (2) を用いることで求められる．図 36〜38 にバッファ容量が 1 個のときのバッファの休止時間の測定値と理論値を示す．図 36〜38 より測定値と理論値はほぼ同値となったため，バッファの休憩時間はレート η の指数分布に従うことが分かった．次に遅延について考える．送信機が止まっているときに次のシンボルが到着した場合は送信レートが 1[秒] であるため遅延は 1[秒] となる．図 39〜41 にバッファ容量 1 個のときの送信中のシンボルの送信残り時間の測定値と理論値を示す．図 39〜41 より測定値と理論値はほぼ同値となったため，バッファ容量 1 個のときの送信中のシンボルの送信残り時間の分布は $f_1(x)$ で数式的に表すことができる．また，シンボルが送信中でバッファにシンボルがないときに次のシンボルが到着した場合は送信中のシンボルの送信残り時間の分布 $f_1(x)$ と送信レート 1[秒] を足したものが到着シンボルの遅延の分布となる．また，シンボルが送信中でバッファにシンボルが 1 個あるときに次のシンボルが到着した場合は到着シンボルは取りこぼされるため，バッファ容量が 1 個の場合の遅延は 1〜2[秒] の間に収まることが分かる．

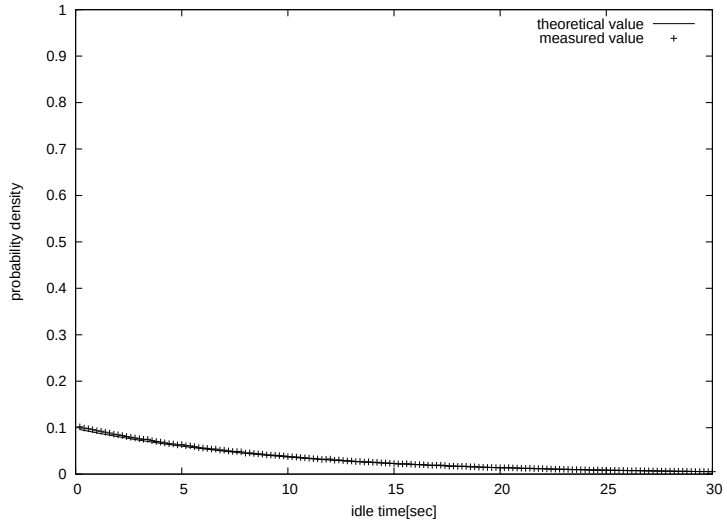

図 36　バッファ容量 1 個　λ=0.1 のときのバッファの休止時間の分布

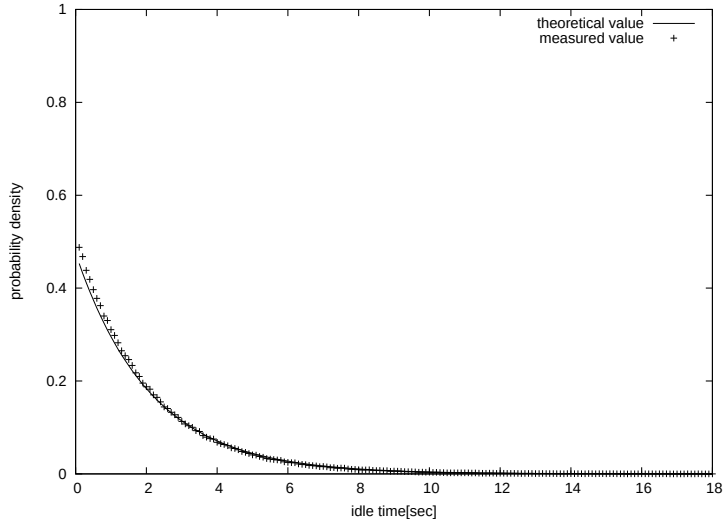

図 37　バッファ容量 1 個　λ=0.5 のときのバッファの休止時間の分布

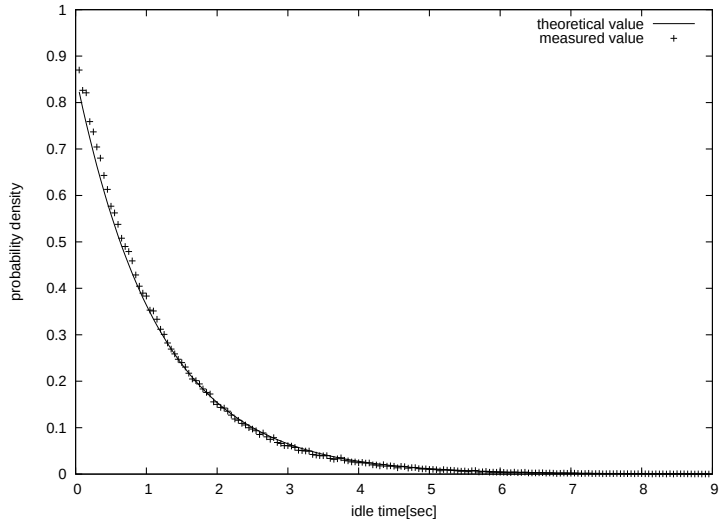

図 38　バッファ容量 1 個　λ=0.9 のときのバッファの休止時間の分布

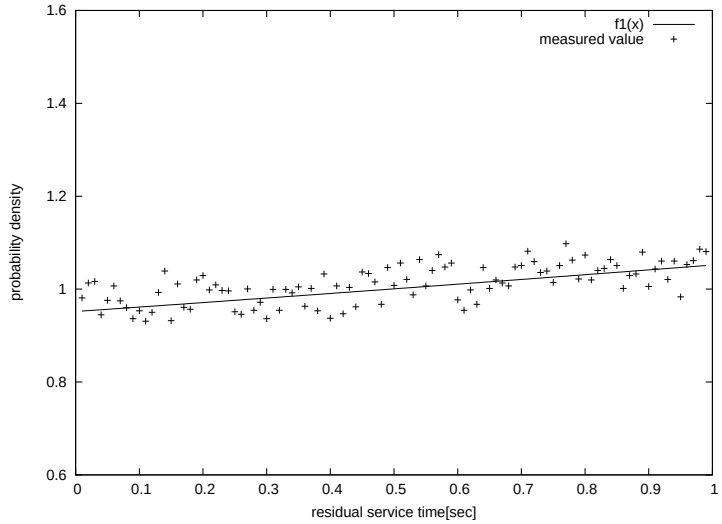

図 39　バッファ容量 1 個　λ=0.1 のときの送信残り時間の分布

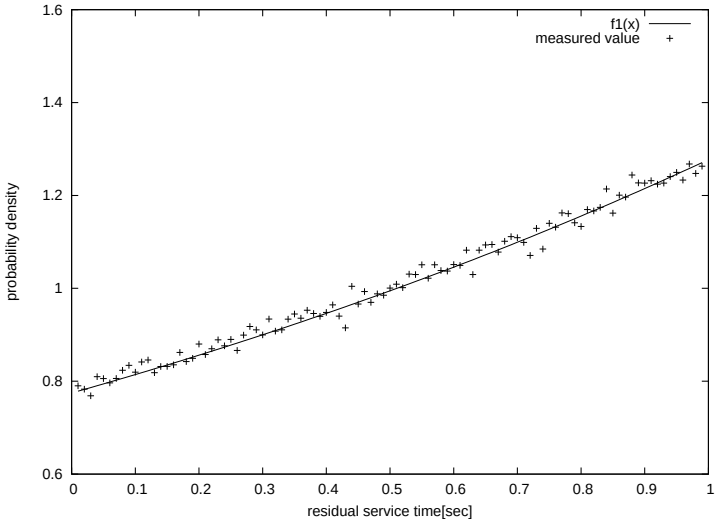

図 40　バッファ容量 1 個　λ=0.5 のときの送信残り時間の分布

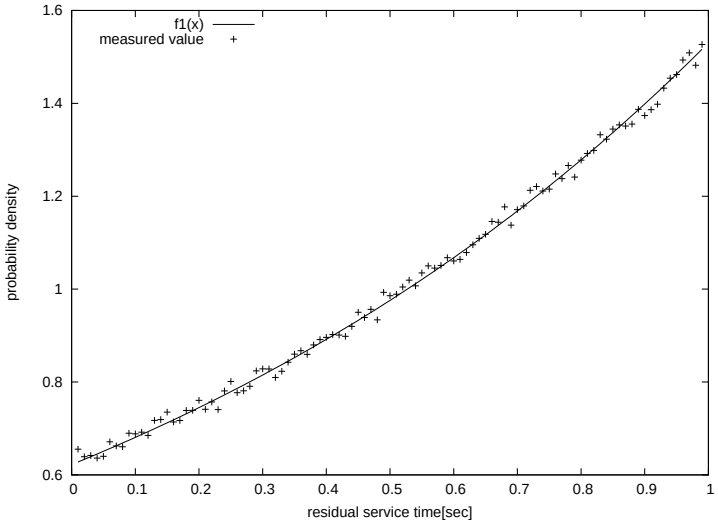

図 41　バッファ容量 1 個　λ=0.9 のときの送信残り時間の分布

付録 D　ソースコード

D.1　出発間隔を測定するプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*******************************指数分布乱数を発生*****************************************/

#define MRND  1000000000L
static int jrand;
static long ia[56]; /* ia[1..55] */

static void irn55(void)
{
        int i;
        long j;

        for (i = 1; i <= 24; i++) {
                j = ia[i] - ia[i + 31];
                if (j < 0)j += MRND;
                ia[i] = j;
        }
        for (i = 25; i <= 55; i++) {
                j = ia[i] - ia[i - 24];
                if (j < 0)j += MRND;
                ia[i] = j;
        }
}

void init_rnd(unsigned long seed)
{
        int i, ii;
        long k;

        ia[55] = seed;
        k = 1;
        for (i = 1; i <= 54; i++) {
                ii = (21 * i) % 55;
                ia[ii] = k;
                k = seed - k;
                if (k < 0) k += MRND;
                seed = ia[ii];
        }
        irn55(); irn55(); irn55(); /* warm up */
        jrand = 55;
}

long irnd(void) /* 0<=irnd()<MRND */
{
        if (++jrand > 55) { irn55(); jrand = 1; }
        return ia[jrand];
}

double rnd(void) /* 0<=rnd()<1 */
{
        return (1.0 / MRND) * irnd();
}

double rand_exp(double lambda)
{
        return (-(1 / lambda)*log(1 - (rnd())));    /*指数分布*/
}
```

```
/*************************************************************/

double present_time;  /*現在時刻*/
double arrival_time;  /*到着時刻*/
double sending_time;  /* 送信終了時刻 */

#define ARRIVING_LIST_MAX 10000
#define SENDING_BUFFER_LIST_MAX 10000
#define INPUT_LENGTH 1000000
#define ARRIVING_RATE 0.9

double arriving_list[ARRIVING_LIST_MAX];
int arriving_count = 0;
int arriving_in = 0;
int arriving_out = 0;

int sending_buffer_list[SENDING_BUFFER_LIST_MAX];
int sending_buffer_count = 0;
int sending_buffer_in = 0;
int sending_buffer_out = 0;

/************************バッファ（エンコーダ側）**********************************/

void add_arriving_list(double time)
{
        if (arriving_count >= ARRIVING_LIST_MAX) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        arriving_list[arriving_in] = time;
        arriving_in = (arriving_in + 1) % ARRIVING_LIST_MAX;
        arriving_count++;
        return;
}

double get_arriving_list()
{
        double time;

        if (arriving_count == 0) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        time = arriving_list[arriving_out];
        arriving_out = (arriving_out + 1) % ARRIVING_LIST_MAX;
        arriving_count--;
        return(time);
}

/*****************************バッファ（送信器側）**********************************/

void add_sending_buffer_list(int id)
{
        if (sending_buffer_count >= SENDING_BUFFER_LIST_MAX) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        sending_buffer_list[sending_buffer_in] = id;
        sending_buffer_in = (sending_buffer_in + 1) % SENDING_BUFFER_LIST_MAX;
        sending_buffer_count++;
        return;
}

int get_sending_buffer_list()
{
        int id;

        if (sending_buffer_count == 0) {
                printf("error on %d\n", __LINE__);
```

```
                exit(0);
        }
        id = sending_buffer_list[sending_buffer_out];
        sending_buffer_out = (sending_buffer_out + 1) % SENDING_BUFFER_LIST_MAX;
        sending_buffer_count--;
        return(id);
}

/**************************シンボルを生成*************************************/

double prob0 = 0.5;

int generate_symbol()
{
        return((rnd() < prob0) ? 0 : 1);
}

/**************************到着処理***********************************/
int input_count = 0;
void arrival()
{
        present_time = arrival_time;
        input_count++;
        if (input_count >= INPUT_LENGTH) {
                arrival_time = -1.0;
        }
        else {
                arrival_time = present_time + rand_exp(ARRIVING_RATE);
        }
        add_sending_buffer_list(generate_symbol());
        add_arriving_list(present_time);
        if (sending_time < 0.0) {
                sending_time = present_time + 1.0;
        }
        return;
}

/**************************送信完了処理*********************************/
double last_arrival;
void sending_finish()
{
        double interval;  /* 到着間隔 */

        present_time = sending_time;
        get_sending_buffer_list();
        get_arriving_list();
        interval = present_time - last_arrival;
        printf("%f\n", interval);
        last_arrival = present_time;
        if (sending_buffer_count > 0) {
                sending_time = present_time + 1.0;
        }
        else {
                sending_time = -1.0;
        }
        return;
}
/*****************************************************************/

int main(void)
{
        init_rnd(12345);
        present_time = 0.0;
        arrival_time = rand_exp(ARRIVING_RATE);
        sending_time = -1.0;
        last_arrival = 0;

        while ((arrival_time >= 0.0) || (sending_time >= 0.0)) {
                if (arrival_time >= 0.0 && sending_time < 0.0) {
```

```
                        arrival();
                }
                else if (arrival_time < 0.0 && sending_time >= 0.0) {
                        sending_finish();
                }
                else if (arrival_time <= sending_time ){
                        arrival();
                }
                else {
                        sending_finish();
                }
        }
        return(0);
}
```

D.2 遅延を測定するプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*******************************指数分布乱数を発生*****************************************/

#define MRND  1000000000L
static int jrand;
static long ia[56]; /* ia[1..55] */

static void irn55(void)
{
        int i;
        long j;

        for (i = 1; i <= 24; i++) {
                j = ia[i] - ia[i + 31];
                if (j < 0)j += MRND;
                ia[i] = j;
        }
        for (i = 25; i <= 55; i++) {
                j = ia[i] - ia[i - 24];
                if (j < 0)j += MRND;
                ia[i] = j;
        }
}

void init_rnd(unsigned long seed)
{
        int i, ii;
        long k;

        ia[55] = seed;
        k = 1;
        for (i = 1; i <= 54; i++) {
                ii = (21 * i) % 55;
                ia[ii] = k;
                k = seed - k;
                if (k < 0) k += MRND;
                seed = ia[ii];
        }
        irn55(); irn55(); irn55(); /* warm up */
        jrand = 55;
}

long irnd(void) /* 0<=irnd()<MRND */
{
```

```
        if (++jrand > 55) { irn55(); jrand = 1; }
        return ia[jrand];
}

double rnd(void) /* 0<=rnd()<1 */
{
        return (1.0 / MRND) * irnd();
}

double rand_exp(double lambda)
{
        return (-(1 / lambda)*log(1 - (rnd())));    /*指数分布*/
}

/*****************************************************************/

double present_time;  /*現在時刻*/
double arrival_time;  /*到着時刻*/
double sending_time;  /* 送信終了時刻 */

#define ARRIVING_LIST_MAX 100000
#define SENDING_BUFFER_LIST_MAX 100000
#define INPUT_LENGTH 1000000
#define ARRIVING_RATE 0.5

double arriving_list[ARRIVING_LIST_MAX];
int arriving_count = 0;
int arriving_in = 0;
int arriving_out = 0;

int sending_buffer_list[SENDING_BUFFER_LIST_MAX];
int sending_buffer_count = 0;
int sending_buffer_in = 0;
int sending_buffer_out = 0;

/*************************バッファ（エンコーダ側）**********************************/

void add_arriving_list(double time)
{
        if (arriving_count >= ARRIVING_LIST_MAX) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        arriving_list[arriving_in] = time;
        arriving_in = (arriving_in + 1) % ARRIVING_LIST_MAX;
        arriving_count++;
        return;
}

double get_arriving_list()
{
        double time;

        if (arriving_count == 0) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        time = arriving_list[arriving_out];
        arriving_out = (arriving_out + 1) % ARRIVING_LIST_MAX;
        arriving_count--;
        return(time);
}

/*****************************バッファ（送信器側）**********************************/

void add_sending_buffer_list(int id)
{
        if (sending_buffer_count >= SENDING_BUFFER_LIST_MAX) {
                printf("error on %d\n", __LINE__);
```

```
                exit(0);
        }
        sending_buffer_list[sending_buffer_in] = id;
        sending_buffer_in = (sending_buffer_in + 1) % SENDING_BUFFER_LIST_MAX;
        sending_buffer_count++;
        return;
}

int get_sending_buffer_list()
{
        int id;

        if (sending_buffer_count == 0) {
                printf("error on %d\n", __LINE__);
                exit(0);
        }
        id = sending_buffer_list[sending_buffer_out];
        sending_buffer_out = (sending_buffer_out + 1) % SENDING_BUFFER_LIST_MAX;
        sending_buffer_count--;
        return(id);
}

/**************************シンボルを生成**************************************/

double prob0 = 0.5;

int generate_symbol()
{
        return((rnd() < prob0) ? 0 : 1);
}

/*************************到着処理**********************************/
int input_count = 0;
void arrival()
{
        present_time = arrival_time;
        input_count++;
        if (input_count >= INPUT_LENGTH) {
                arrival_time = -1.0;
        }
        else {
                arrival_time = present_time + rand_exp(ARRIVING_RATE);
        }
        add_sending_buffer_list(generate_symbol());
        add_arriving_list(present_time);
        if (sending_time < 0.0) {
                sending_time = present_time + 1.0;
        }
        return;
}

/*************************送信完了処理**********************************/

void sending_finish()
{
        double delay;  /* 遅延 */

        present_time = sending_time;
        get_sending_buffer_list();
        delay = (present_time - get_arriving_list());
        printf("%f\n", delay);
        if (sending_buffer_count > 0) {
                sending_time = present_time + 1;
        }
        else {
                sending_time = -1.0;
        }
        return;
}
```

```
/*****************************************************************/

int main(void)
{
        int i;
        init_rnd(12345);
        present_time = 0.0;
        arrival_time = rand_exp(ARRIVING_RATE);
        sending_time = -1.0;

        while ((arrival_time >= 0.0) || (sending_time >= 0.0)) {
                if (arrival_time >= 0.0 && sending_time < 0.0) {
                        arrival();
                }
                else if (arrival_time < 0.0 && sending_time >= 0.0) {
                        sending_finish();
                }
                else if (arrival_time <= sending_time) {
                        arrival();
                }
                else {
                        sending_finish();
                }
        }

return(0);
}
```