信州大学
大学院総合理工学研究科

修士論文

情報鮮度の観点に基づくバッファで割り込みがある通信システムの実験的評価

指導教員　西新 幹彦 准教授

専攻　工学専攻
分野　電子情報システム学分野
学籍番号　19W2075D
氏名　千葉 直紀

2021 年 2 月 28 日

目次

1 はじめに 1

2 情報鮮度の定義 1
2.1 情報鮮度 1
2.2 時間平均 AoI 3

3 従来研究の紹介 4
3.1 サーバで割り込みが発生する場合 4
3.2 バッファで割り込みが発生する場合 6

4 符号器を追加した通信路の情報鮮度 7
4.1 通信路モデル 7
4.2 クラフトの不等式と符号語長 8
4.3 パラメータ設定について 8

5 結果と考察 9
5.1 情報源の分布と最小時間平均 AoI 9
5.2 情報源の分布と棄却率 11
5.3 情報源の分布と最小時間平均 AoI となるときの棄却率 12
5.4 情報源の分布と最小棄却率となるときの時間平均 AoI 14
5.5 比較 14
5.6 最小 AoI，最小棄却率時の q_1 の比較 15

6 まとめと今後の課題 16

謝辞 17

参考文献 17

付録 A ソースコード 18
A.1 発生確率を変化させたときの時間平均 AoI を測定するプログラム 18

1　はじめに

河川の水位をカメラで観測し遠隔地のモニタへ映し出すといった場合などでは，異常水位や氾濫が発生した際に素早く伝えるためモニタは最新の情報を映す必要がある．しかし，モニタに表示されている情報は伝送路を通して送信されるためセンサでサンプリングしてから時間が経過している．一般に，情報の新しさを情報鮮度 (AoI, Age of Information) という．具体的には，情報の AoI はその情報が発生してからの経過時間で測られる．したがって AoI が小さいほど情報が新しいことを意味する．この用語を用いれば，モニタに表示されている情報の AoI は小さいことが望ましいと言える．この問題に対する従来研究 [4] では，センサでサンプリングした情報が送信機に到着した順番に処理する FIFO(First In First Out) のシステムを考え，モニタに表示されている情報の AoI の時間平均は到着間隔とサービス時間を用いて数理的に求まることが明らかになっている．また別の研究 [1] では，送信機の処理中に新しい情報が到着したら古い情報の処理を中断し割り込みをして優先的に処理する LIFO(Last In First Out) を用いたシステムを考え，モニタに表示中の情報の AoI が FIFO のシステムに比べてより小さくなることが明らかにされている．これらの研究で想定されたシステムは，情報が符号語に変換されておらず，かつサーバのサービス時間は指数分布に従っている．しかし，情報を符号語に変換し符号語長ごとに最適なサービス時間を設定すれば，AoI を小さくできると考えた．そこで，本研究では 2 種類の情報を送信したときのシステムに符号器を追加し情報を符号語に変換，符号語長に比例したサービス時間で処理される場合について検証を行った．

本論文は次のように構成される．2 章では，AoI の時間変化と時間平均 AoI の計測方法について述べる．3 章では，時間平均 AoI を小さくする割り込みがあるシステムに関する従来研究 [3] を紹介する．4 章では，本研究で扱う符号器を追加したシステムの解説と符号語の生成に用いたクラフトの不等式について説明する．5 章では，システムの検証結果として情報源の分布を変化させたときの時間平均 AoI と，割り込みが発生しどれだけの情報が棄却されたのかという棄却率について検証結果示す．6 章で本稿のまとめと今後の課題について述べる．

2　情報鮮度の定義

本章では情報鮮度について説明する．

2.1　情報鮮度

情報鮮度を説明するために図 1 の通信システムを考える．このシステムでは情報源からサンプリングされた情報はパケットに格納され送信機のバッファに到着順に保管される．サーバ

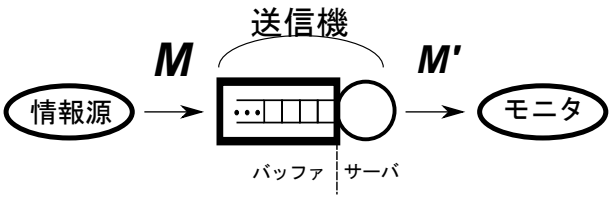

図 1　通信システムの例

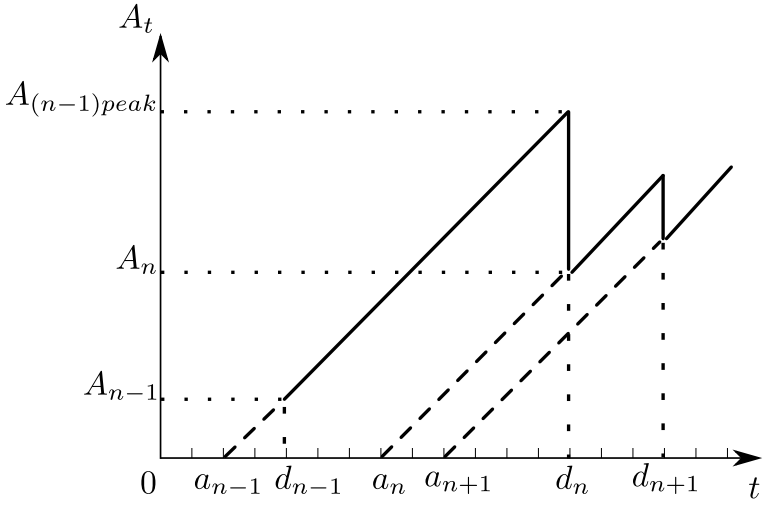

図 2　AoI の遷移図

ではバッファに到着した順番でパケットの処理が行われネットワークを通じて遠隔地のモニタに表示される．こうしてモニタにはサンプリング時の情報が表示される．モニタに表示された直後の情報は最新のため年齢は小さいが，時間経過とともにモニタに表示中の情報の年齢は斜め 45° で増加していく．そしてモニタの表示が新しい情報に更新されるとき新しい情報の年齢までジャンプする．この情報の鮮度を定量化するための指標として Age of Information(AoI) を用いる．時刻 t にモニタに表示されている情報の AoI A_t はその情報のサンプリング時刻を a とおくと，

$$A_t = t - a \tag{1}$$

と書ける．図 2 に AoI の時間変化の例を示す．$n-1$ 番目にサンプリングされた情報 X_{n-1} がサンプリング時刻 a_{n-1} でバッファに保持される．a_{n-1} から時間が経過すると情報 X_{n-1} の AoI が増加していき，情報は古くなる．時刻 d_{n-1} にサーバの処理が終了するとモニタに情報 X_{n-1} が表示される．情報 X_{n-1} が表示された瞬間の AoI の値は，

$$A_{n-1} = d_{n-1} - a_{n-1} \tag{2}$$

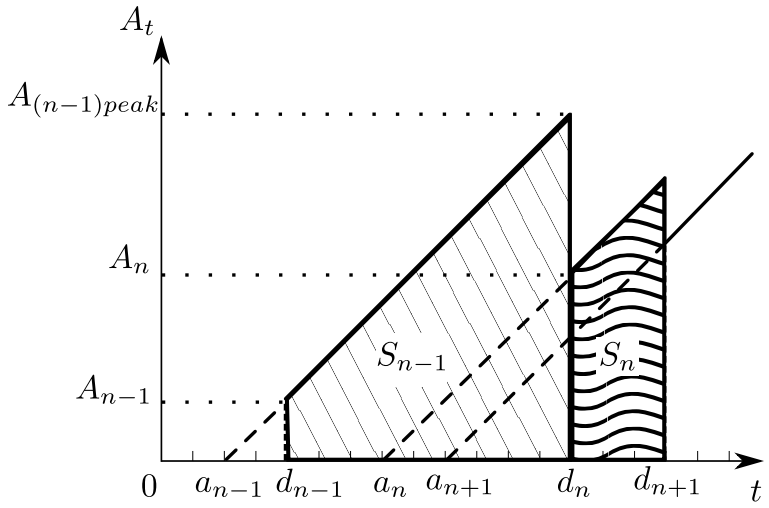

図 3　AoI の時間平均 導出

と書ける．さらに，時刻 a_n で n 番目の情報 X_n がサンプリングされる．ここで時刻 d_n に注目する．この時，直前までモニタに表示されていた情報は更新されて，情報 X_n がモニタに表示される．モニタの表示内容が更新されることで，AoI の値が $A_{(n-1)peak}$ から A_n と小さくなる．このように，モニタの表示内容は更新を繰り返すことから，図 2 のノコギリ状の実線グラフが得られる．本研究の目的はこのノコギリ状の AoI のグラフの時間平均を小さくすることである．次節で AoI を時間平均で評価する方法について述べる．

2.2　時間平均 AoI

AoI の評価方法には，更新される直前のピーク AoI がある目標値以下にとどまっている時間割合で評価する方法や，AoI を計測時間の時間平均で評価する方法などがある．本研究では AoI を時間平均で評価する方法を採用した．

観測時間 T の AoI の時間平均を A_{ave} とすると，

$$A_{ave} = \frac{1}{T}\int_0^T A_t dt \tag{3}$$

と書ける．本研究では図 3 のグラフから A_{ave} は，

$$A_{ave} = \frac{1}{T}\sum_{n=1}^{l} S_n \tag{4}$$

$$S_n = \frac{1}{2}\left\{(d_{n+1} - a_n)^2 - (d_n - a_n)^2\right\} \tag{5}$$

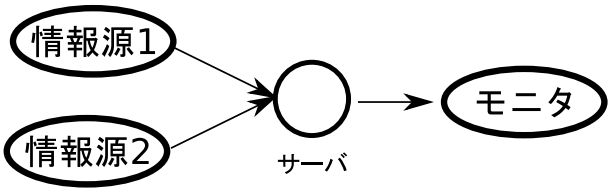

図 4　サーバで割り込みが発生する通信システム

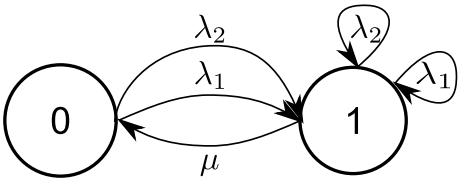

図 5　サーバで割り込みが発生する通信システムの状態遷移図

と書ける．ここで，S_n は情報 x_n の AoI をモニタに表示された時刻 d_n から更新された時刻 d_{n+1} までの時間で囲んだ台形面積であり，l は観測時間 T で最後に表示された情報がサンプリングされた順序 (l 番目) である．

3　従来研究の紹介

モニタの AoI に注目すると，サンプリング時刻と現在時刻の差が小さいほど AoI の値が小さくなりより新しい情報が表示されることになる．また，情報がモニタに表示された時点での AoI はバッファでの待ち時間とサーバでのサービス時間によって決まる．そこで，それらの時間を短くするために古い情報を破棄して新しい情報を優先的に処理する方法が考えられる．以下でそのようなシステムに対する従来研究 [3] を紹介する．

3.1　サーバで割り込みが発生する場合

サーバで割り込みが発生する通信システムを図 4 に示す．情報源は 2 種類あり，着目したい情報は情報源 1 から到着レート λ_1 で出力され，それ以外の情報は情報源 2 から到着レート λ_2 で出力される．情報源から出力された情報は送信機にあるサーバに入れられ送信レート μ で処理され遠隔地のモニタに情報が表示される．ここで，サーバで処理中に新しい情報がサーバに到着すると処理中の情報を棄却し新しい情報を処理することでモニタに表示されている情報の AoI を小さくすることができる．サーバの状態は待機中かパケットの処理中かのどちらかであり，次のパケットの到着や処理の終了によって状態が遷移する．この様子を表した状態遷移図を図 5 に示す．図中の状態 0 と状態 1 は，それぞれサーバの待機中と処理中を表してい

る．待機中のサーバには情報源 1 と 2 からそれぞれ到着レート λ_1 と λ_2 で情報が到着する．いずれかの情報源から情報が到着するとサーバーは処理中の状態へと遷移する．サーバは処理を終えると待機中の状態になる．ただしサーバの処理中にもいずれかの情報源から情報が届く可能性があり，その場合は処理中の情報を破棄して新しい情報の処理が改めて開始される．この状態遷移に基づいて時間平均 AoI の値が従来研究 [2] から次のように求められる．

定理 1. 時間平均 *AoI*
情報源 *1* の時間平均 *AoI* を Δ_1 とすると，

$$\Delta_1 = \sum_{q=0}^{n} v_{q0} \tag{6}$$

である．

ここで情報源 1， 2，全体の稼働率 ρ_1, ρ_2, ρ は，

$$\rho_1 = \frac{\lambda_1}{\mu} \tag{7}$$

$$\rho_2 = \frac{\lambda_2}{\mu} \tag{8}$$

$$\rho = \rho_1 + \rho_2 \tag{9}$$

と書ける．また，

$$\begin{bmatrix} \pi_0 & \pi_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{1+\rho} & \frac{\rho}{1+\rho} \end{bmatrix} \tag{10}$$

とする．状態は，0,1 の 2 種類より $n = 1$．v_{00}, v_{10} は，

$$v_{00} = \frac{1}{\rho_1}\left(\frac{1}{\mu}\pi_0 + \frac{1+\rho_2}{\mu(1+\rho)}\pi_1\right) \tag{11}$$

$$v_{10} = \frac{\pi_1}{\mu} + \frac{\rho}{\rho_1}\left(\frac{1}{\mu}\pi_0 + \frac{1+\rho_2}{\mu(1+\rho)}\pi_1\right) \tag{12}$$

である．時間平均 AoIΔ_1 は代入すると，

$$\begin{aligned} \Delta_1 &= v_{00} + v_{10} \\ &= \frac{1+\rho}{\mu\rho_1} \end{aligned} \tag{13}$$

と書ける．式 (13) より，時間平均 AoI を小さくするためには送信レートを大きくするか ρ_1 を大きくすればよいことが分かる．しかし，サーバが処理中であっても新しい情報が到着したら処理を中断しなければならないため情報源の種類によっては着目したい情報がモニタに表示される頻度も減ってしまうことが考えられる．

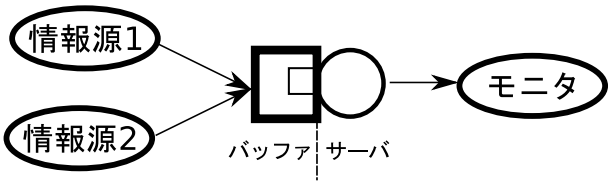

図 6　バッファで割り込みが発生する通信システム

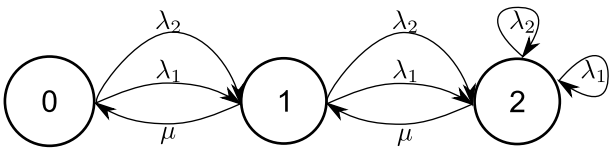

図 7　バッファで割り込みが発生する通信システムの状態遷移図

3.2　バッファで割り込みが発生する場合

バッファで割り込みが発生する通信システムを図 6 に示す．情報源は 2 種類あり，着目したい情報は情報源 1 から到着レート λ_1 で出力され，それ以外の情報は情報源 2 から到着レート λ_2 で出力される．情報源から出力された情報は送信機にあるサーバへ入れられ送信レート μ で処理され遠隔地のモニタに情報が表示される．サーバの前段にはバッファが設置してありサーバが処理中の時に新しい情報が出力されたら 1 つだけ保持することができる．ここで，サーバで処理中かつバッファで情報が保持されているとき新しい情報がバッファに到着するとバッファ内の情報を棄却し新しい情報を保持することで待機時間が短くなり，モニタに表示されている情報の AoI を小さくすることができる．サーバの状態は待機中かパケットの処理中かのどちらかであり，次のパケットの到着や処理の終了によって状態が遷移する．この様子を表した状態遷移図を図 7 に示す．図中の状態 0 と状態 1 と状態 2 は，それぞれバッファが空でサーバ待機中，バッファが空でサーバ処理中，バッファ保持かつサーバ処理中を表している．各状態へ向かう遷移についてはサーバで割り込みが発生する場合と同様である．この状態遷移に基づいて時間平均 AoI の値が次のように求められる．

$$\begin{bmatrix} \pi_0 & \pi_1 & \pi_2 \end{bmatrix} = C_\pi \begin{bmatrix} 1 & \rho & \rho^2 \end{bmatrix} \tag{14}$$

$$C_\pi = (1 + \rho + \rho^2)^{-1} \tag{15}$$

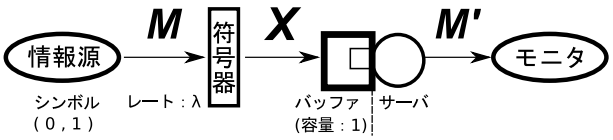

図 8　通信モデル

とする．状態は，0,1,2 の 3 種類より $n = 2$．v_{00}, v_{10}, v_{20} は，

$$\rho v_{00} = \frac{\pi_0}{\mu} + v_{11} \tag{16}$$

$$v_{10} = \frac{1}{\mu(1+\rho)} + v_{11} \tag{17}$$

$$v_{20} = \frac{\pi_2}{\mu} + \rho v_{10} \tag{18}$$

と書ける．ここで $v_{11} = \frac{\rho}{\mu(1+\rho)}\frac{1}{\rho_1} - \frac{C_\pi(1+\rho+\rho^3)}{\mu(1+\rho)^2}$ である．定理 6 に代入すると，

$$\begin{aligned}\Delta_1 &= v_{00} + v_{10} + v_{20} \\ &= \frac{1}{\mu} + \frac{\pi_0 + \pi_2}{\rho} + \frac{1+\rho+\rho^2}{\rho} v_{11}\end{aligned} \tag{19}$$

が得られる．このシステムではバッファで割り込みが発生するためバッファでの待ち時間が発生してしまうがサーバに到着した情報は確実にモニタに表示することができる．サーバで割り込みが発生するシステムと比較すると AoI は大きくなってしまうが，着目したい情報の表示される頻度は増えると考えられる．

4　符号器を追加した通信路の情報鮮度

4.1　通信路モデル

3 章では，従来研究として 2 種類のシステムについて説明した．しかし，どちらのシステムも送信された情報の AoI を測定することはできるが，AoI を小さくするためにエンジニアが調整できる部分はない．そこで，本研究では，3.2 節のバッファで割り込みが発生する通信システムを用いて．システムに符号器を追加し，符号語長に比例したサービス時間を調整することで AoI を小さくすることができると考えた．本研究で扱う通信システムを図 8 に示す．このシステムは，情報源からサンプリングされたシンボルは符号器で符号語に変換され送信機にあるサーバで処理され，ネットワークを通じて遠隔地のモニタに送信される．情報源について，シンボルがポアソン過程で到着すると考える．符号語はバッファを通過しサーバで処理される．サーバでのサービス時間は符号語長に比例する．モニタでは符号語が復号され表示される．ここでサーバに処理中の符号語がある際新しい符号語はバッファで待機する．バッファの

容量は 1 つであり，待機してるところにさらに新しい符号語がきたらバッファで待機している符号語は棄却され新しい符号語が待機するバッファで割り込みが発生する通信システムを考える．

4.2　クラフトの不等式と符号語長

サーバは符号器で生成する符号語の長さに比例した時間で処理をする．すると，符号語長が短いほどより新しい情報が伝達されるため AoI は小さくなる．本研究では，符号語長をクラフトの不等式を満たすように生成する．

定理 2. クラフトの不等式
符号語長が $d_1, d_2 ... d_n$ である瞬時復号可能な符号が存在するための必要十分条件は，

$$\sum_{i=1}^{n} 2^{-d_i} \leq 1 \tag{20}$$

を満たすことである．

本研究では，簡単のため，シンボルの種類は'0,1' の 2 種類とする．シンボル'0' のサービス時間 d_0，シンボル'1' のサービス時間 d_1 を用いて符号語長の条件を，

$$q_0 + q_1 = 1 \tag{21}$$
$$q_0 = 2^{-d_0} \tag{22}$$
$$q_1 = 2^{-d_1} \tag{23}$$

と定義する．式 (21) の条件の下でサービス時間を決定する．また，実際には符号語長は整数でなければならないが本研究ではサービス時間を実数で表現するため符号語長を実数として扱う．

4.3　パラメータ設定について

通信システムをプログラムで実験するためにパラメータとして情報源の分布，サーバのサービス時間を変更し，情報源の種類と到着レート，バッファの個数，観測期間は固定した．表 1 にパラメータ設定をのせる．また，バッファで割り込みのあるシステムであり棄却率も計測した．本研究における棄却率は，情報源から発生したシンボルの個数に対する割りこまれたシンボルの個数の割合で定義している．

表 1　パラメータ設定

条件	値	備考
シンボルの種類	0　1	確率 p_1 で 1 が送信
到着レート λ	1.0	ポアソン過程で到着
サービス時間		クラフトの不等式で決まる
合計到着数	100000	計測開始後シンボルが合計到着数に達したら計測終了
バッファの容量	1	サービス中に新しいシンボルが到着したら保持
バッファでの割り込み	あり	保持中に新しいシンボルが到着したらバッファで割り込み
サーバでの割り込み	なし	サーバに入ったら処理は中断されない

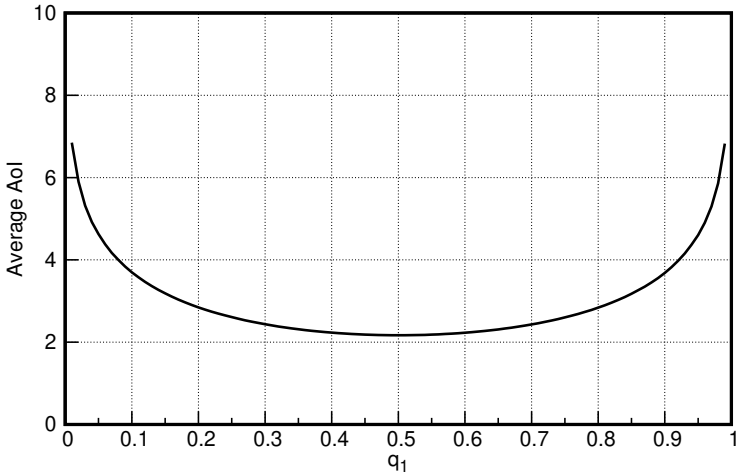

図 9　$p_1 = 0.5$ の時間平均 AoI

5　結果と考察

5.1　情報源の分布と最小時間平均 AoI

シンボル'1' の発生確率 $p_1 = 0.5$ の時間平均 AoI の結果を図 9 に示す．横軸 $q1$ はクラフトの不等式で求まる値であり，横軸の値が小さいほどシンボル'1' のサービス時間は大きくなる．縦軸は時間平均 AoI の値であり，縦軸の値が小さいほどモニタに表示される情報は新しいことになる．最小値は $(q_1, AoI) = (0.5, 2.168)$ の点をとりそこから AoI の値が大きくなる

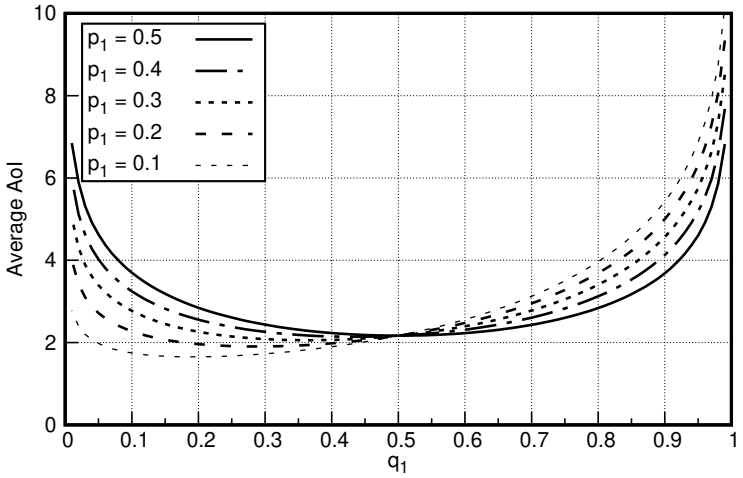

図 10　分布を変更したときの時間平均 AoI

下に凸のグラフが得られた．まず，モニタに表示されている情報の AoI を小さくするためには，シンボル'0'，'1' のサービス時間を限りなく小さくすればよい．しかし，クラフトの不等式 $q0 + q1 = 1$ の制限があるため一方のサービス時間を小さくするともう一方のサービス時間が大きくなる関係にある．次に，状態遷移を考える．本研究ではバッファで割り込みが発生する通信システムを利用しているため状態遷移図は図 7 と同じになる．待機中の状態 0，バッファが空でサービス中の状態 1，シンボル'0','1' が割り込む状態 2 に分けることができる．状態 0 から状態 1，状態 1 から状態 2 の遷移はシンボルの種類によらず一定の到着レート λ である．状態 2 のループでも割り込むシンボルは種類によらず到着レートは λ である．状態 2 から状態 1，状態 1 から状態 0 の遷移はクラフトの不等式に基づいたサービス時間でサービスされるためシンボルの種類によって異なる．モニタに表示される AoI を小さくするためには各状態に滞在する時間を短くする必要がある．そのためにはサービス時間を短くする必要がある．シンボル'1' の発生確率 $p_1 = 0.5$ のときは，'0'，'1' が等確率で発生するためサービス時間を同一にしたときが一番時間平均 AoI は小さくなったと考えられる．

次に，シンボル'1' の発生確率 $p_1 = 0.1$ から 0.5 と変化したときの時間平均 AoI の結果を図 10 に示す．結果から各グラフの最小値は $p_1 = 0.1$ のとき $(q_1, AoI) = (0.19, 1.653)$，$p_1 = 0.2$ のとき $(q_1, AoI) = (0.29, 1.904)$，$p_1 = 0.3$ のとき $(q_1, AoI) = (0.36, 2.057)$，$p_1 = 0.4$ のとき $(q_1, AoI) = (0.43, 2.141)$，$p_1 = 0.5$ のとき $(q_1, AoI) = (0.5, 2.168)$ である．p_1 が小さくなるほど最小値はグラフの左下に向かう傾向が得られた．この結果について，情報源の分布が偏っているため頻繁に出力されるシンボルについてはサービス時間を短くし，稀に出力される

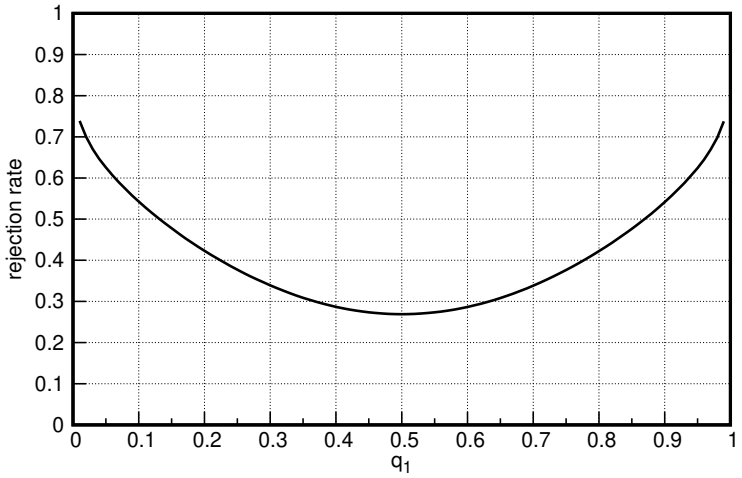

図 11　$p_1 = 0.5$ の棄却率

シンボルのサービス時間は大きくすることで全体としてモニタには新しい情報が表示され時間平均 AoI は小さくなる．分布の偏りが大きいほどサービス時間の配分も偏り，頻繁に出力されるシンボルを短いサービス時間で処理することとなりグラフの最小値が左下に向かうと考えられる．

5.2　情報源の分布と棄却率

前節では時間平均 AoI についての結果を示した．本研究では割り込みが発生するシステムを考えているため，どれだけ割り込みが発生して情報が棄却されたかについても興味が出た．そこで，シンボル'1' の発生確率 $p_1 = 0.5$ としたときの棄却率の結果を図 11 に示す．横軸は $q1$，縦軸は棄却率 (rejection rate:rr) であり，値が 0 に近づくほど割り込みが少なくなり情報源から出力されたシンボルが棄却されずにモニタに表示されることになる．最小値は $(q_1, rr) = (0.5, 0.268)$ の点をとりそこから棄却率が大きくなる下に凸のグラフが得られた．この結果について状態遷移を考えると，状態 2 から状態 1，状態 1 から状態 0 の遷移はクラフトの不等式に基づいたサービス時間で処理されるためシンボルの種類によって異なる．サービス中に新しいシンボルが到着するとバッファで棄却が発生する．新しいシンボルが到着する時間間隔よりもサービス時間が長いほど棄却も多くなってしまう．したがって棄却率を小さくするためには到着間隔に対してサービス時間を短くし，各状態に滞在する時間を短くする必要がある．シンボル'1' の発生確率 $p_1 = 0.5$ のときは，'0'，'1' が等確率で発生するためサービス時間を同一にしたときが一番棄却率は小さくなると考えられる．

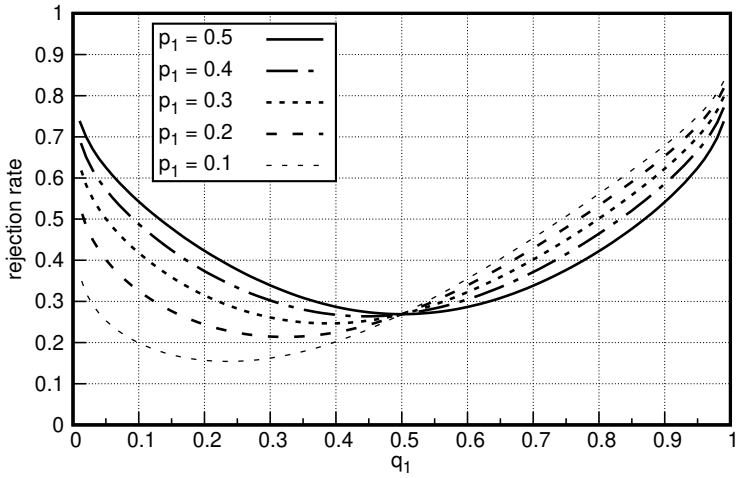

図 12　分布を変更したときの棄却率

次に，シンボル'1' の発生確率 $p_1 = 0.1$ から 0.5 と変化したときの棄却率の結果を図 12 に示す 結果から各グラフの最小値は $p_1 = 0.1$ のとき $(q_1, rr) = (0.24, 0.154)$，$p_1 = 0.2$ のとき $(q_1, rr) = (0.32, 0.213)$，$p_1 = 0.3$ のとき $(q_1, rr) = (0.39, 0.246)$，$p_1 = 0.4$ のとき $(q_1, rr) = (0.45, 0.263)$，$p_1 = 0.5$ のとき $(q_1, rr) = (0.5, 0.268)$ である．p_1 が小さくなるほど最小値はグラフの左下に向かう傾向が得られた．この結果について，情報源の分布が偏っているため頻繁に出力されるシンボルについてはサービス時間を短くし，稀に出力されるシンボルのサービス時間は大きくすることで全体として到着間隔よりもサービス時間が短くなりに棄却が発生しづらくなる．分布の偏りが大きいほどサービス時間の配分も偏り，頻繁に出力されるシンボルを短いサービス時間で処理することとなりグラフの最小値が左下に向かうと考えられる．

5.3　情報源の分布と最小時間平均 AoI となるときの棄却率

エンジニアはシステムの符号器を調整することでモニタの時間平均 AoI を最小にし，最新の情報をモニタに表示することができる．そこで，情報源の分布ごとに時間平均 AoI の最小値を測定し，分布を変化させて繰り返し測定を行った．さらに，最小時間平均 AoI を取るときの棄却率も同時に測定した．まず，シンボル'1' の発生確率 $p_1 = 0.01$ から 0.49 と変化させたときの確率ごとの最小時間平均 AoI の結果を図 13 に示す．横軸は p_1，縦軸は AoI である．以降の結果は横軸 p_1 の範囲を 0.0 から 0.5 としている．これは，シンボルの種類が 2 種類のため 0.5 から 1.0 の範囲は対称なグラフが得られることが自明である．結果からシンボル'1' の

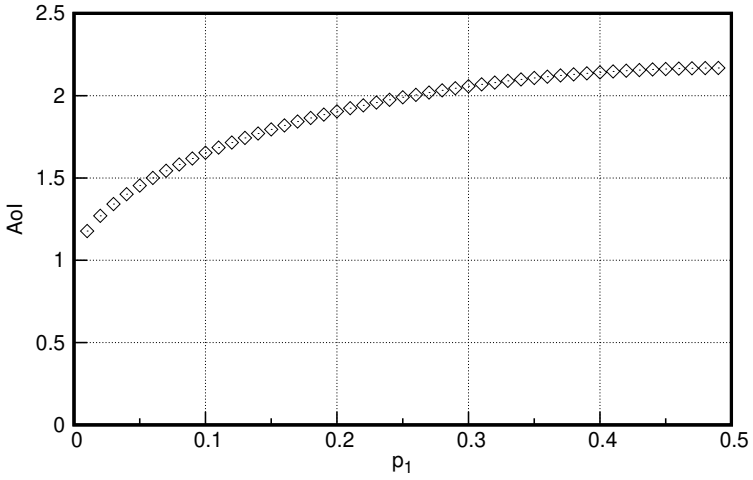

図 13　発生確率ごとの最小時間平均 AoI

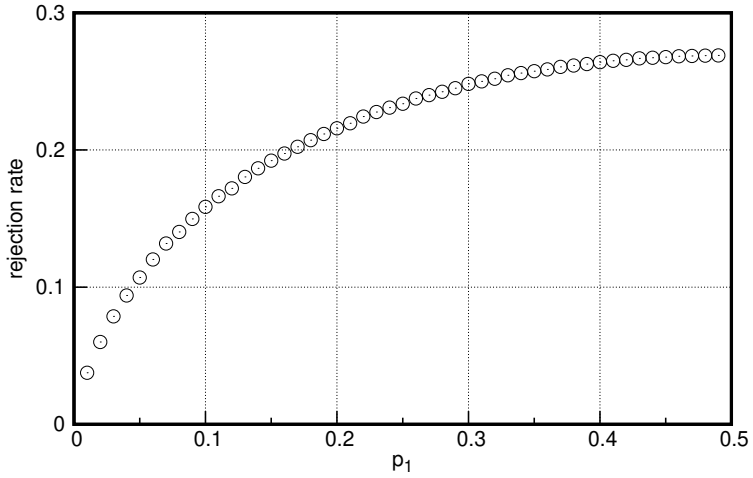

図 14　棄却率

発生確率が等確率に近づくと AoI が増加し，情報源の分布に偏りが大きいほど AoI は小さくなることが分かる．次に，最小時間平均 AoI となるときの棄却率を図 14 に示す．横軸は p_1，縦軸は rr である．結果からシンボル'1' の発生確率が等確率に近づくと rr が増加し，情報源の分布に偏りが大きいほど棄却率は小さくなることが分かる．図 13 と図 14 から，情報源の分布が分かればエンジニアは AoI が最小になるシステムを設計した際に棄却率の許容値を示す

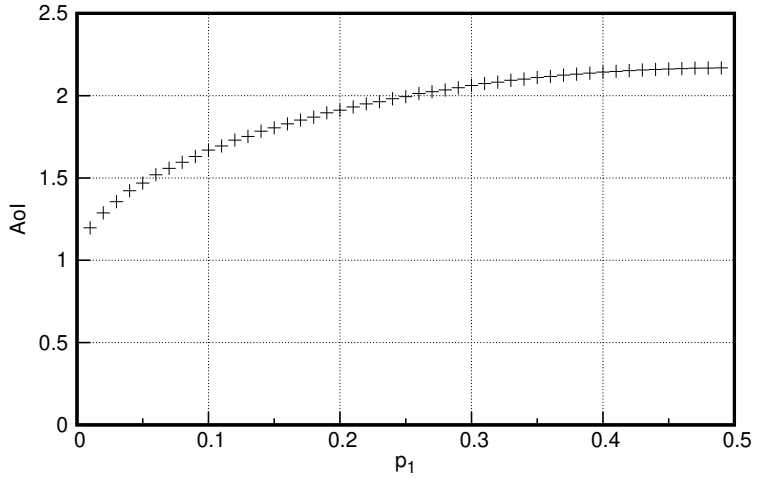

図 15　時間平均 AoI

ことができると考えられる．

5.4　情報源の分布と最小棄却率となるときの時間平均 AoI

なるべく情報を棄却しないで送信したいといった場合の符号器の条件にも興味がある．そこでシステムの分布を変化させたときの最小棄却率と時間平均 AoI を測定した．まず，情報源の分布ごとの最小棄却率となるときの AoI の結果を図 15，次に，最小棄却率の結果を図 16 に示す． AoI，棄却率ともに p_1 の増加に従って値が増加する傾向が得られた．図 15 と図 16 から，情報源の分布が分かればエンジニアは棄却率が最小になるシステムを設計した際に時間平均 AoI の許容値を示すことができると考えられる．

5.5　比較

前節までに情報源の分布を変化させたときの最小時間平均 AoI と最小棄却率となるときで AoI と棄却率にどのような関係があるかを比較し，どれだけ異なるかを確認した．上記の結果を比較したものをそれぞれ AoI について図 17，棄却率について図 18 に示す． 両者ほぼ一致するという結果が得られた．結果から，シンボル’1’ の発生確率が等確率に近づくと最小時間平均 AoI にすれば棄却率も最小にすることができるが，分布に偏りが発生すると最小時間平均 AoI 時の棄却率と最小棄却率には差が発生することが分かった．

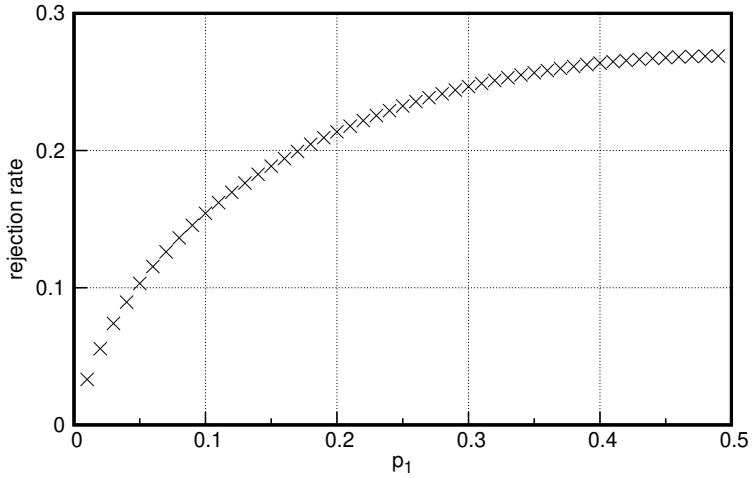

図 16　発生確率ごとの最小棄却率

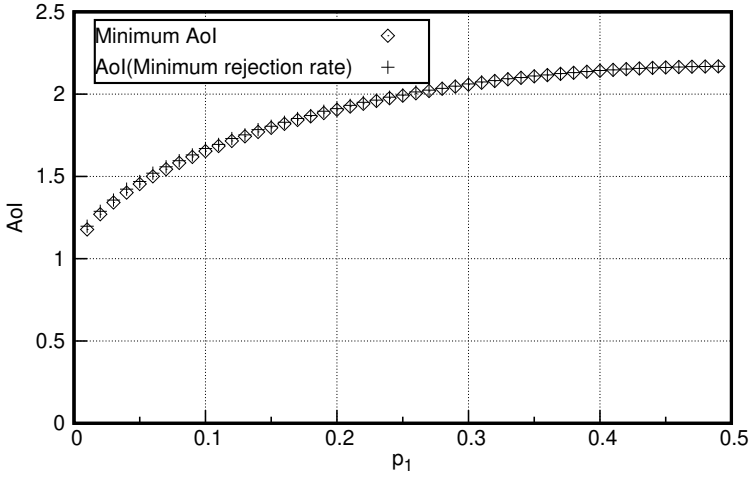

図 17　時間平均 AoI の比較

5.6　最小 AoI，最小棄却率時の q_1 の比較

5.1 節で情報源の分布 p_1 と最小時間平均 AoI の関係について，5.2 節で分布と最小棄却率についてそれぞれ検証した．また，最小 AoI 時の q_1 の値と最小棄却率時の q_1 の値は分布 p_1 が

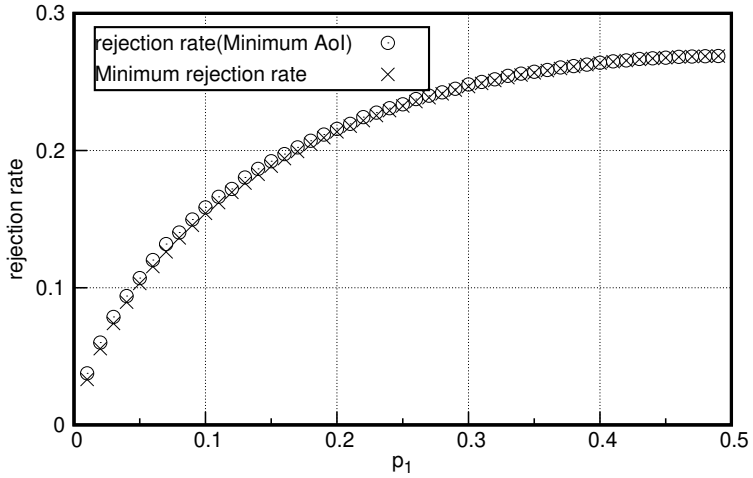

図 18　棄却率の比較

等確率に近づくほど増加するのではないかと考えた．そこで，分布と最小 AoI 時の q_1 の関係について興味が出たため検証を行った．分布を変化させたときの最小 AoI 時の q_1，最小棄却率時の q_1 を図 19 に示す．グラフから両者 p_1 の増加とともに q_1 も増加する傾向が得られた．また，分布の偏りが大きいほど最小 AoI の q_1 と最小棄却率時の q_1 の差が大きくなった．ここで，両者の q_1 の違いによって時間平均 AoI と棄却率にどれだけ影響しているかを考える．図 18 を見ると分布の偏りが大きくなっても最小棄却率と最小 AoI 時の棄却率はほぼ一致することが分かる．従って，符号器で時間平均 AoI を最小化するようにサービス時間を設定すれば棄却率もほぼ最小にできると考えられる．

6　まとめと今後の課題

本研究では AoI を小さくするためのサービス時間を調べるため，バッファで割り込みが発生する通信システムに符号器を追加しサービス時間が符号語長に比例するモデルについて検証を行った．モニタに表示される時間平均 AoI を小さくするための符号器の条件に重点を置き，割り込みの頻度である棄却率についても検証を行った．時間平均 AoI と棄却率はシンボル'1' の発生確率が等確率に近づくと増加し，分布に偏りがあると減少する傾向が得られた．また，最小時間平均 AoI，最小棄却率となるときの時間平均 AoI を見るとシンボル'1' の発生確率が等確率に近いと両者ほぼ一致し，AoI を最小にすれば棄却率も最小になることが確認できた．今後の課題として，通信システムを数学的に解析することができないか検討していきたい．ま

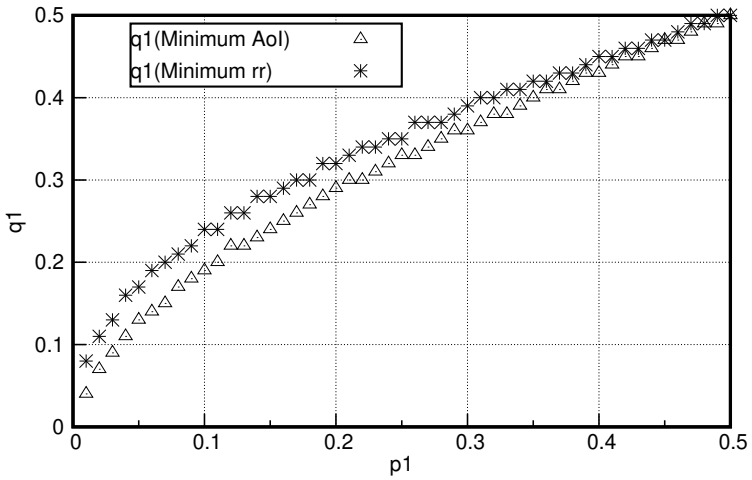

図 19　最小 AoI. 最小棄却率時の q_1

た，情報源のシンボルを増やした場合についても検証をしたい．

謝辞

本研究を行うにあたり，数多くの助言，指導してくださった指導教員西新幹彦准教授，また西新研究室の皆様に感謝の意を表する．

参考文献

[1] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka, “A general formula for the stationary distribution of the age of information and its application to single-server queues,” IEEE Trans. Inf. Theory, vol. 65, no. 12, pp. 8305–8324, Dec. 2019.

[2] S. Kaul, R. Yates, and M. Gruteser, “Real-time status: How often should one update?”, in Proc. of IEEE INFOCOM, Mar. 2012, pp. 2731–2735.

[3] R. D. Yates and S. K. Kaul, “The age of information: Real-time status updating by multiple sources,” IEEE Trans. Inf. Theory, vol. 64, no. 3, pp. 1807–1827, Mar. 2019.

[4] 井上文彰，滝根哲哉，「Age of Information (AoI) 基本概念と研究動向」http://www.ieice.org/ess/sita/forum/article/2018/201810111910.pdf, 2020 年 12 月閲覧．

付録 A　ソースコード

A.1　発生確率を変化させたときの時間平均 AoI を測定するプログラム

```
#pragma warning(disable: 4996) //txt ファイル出力用
#include <stdio.h>
#include<stdlib.h>
#include <math.h>
#include"MT.cpp"
#include"MT.h"
#define _CRT_SECURE_NO_WARNINGS

//指数分布で乱数生成 (パラメータ:para)
double rnd_exp(double para){
   double X;
   X = (-1.0) / para * log(1.0 - genrand_real2()); //[0,1)
   return X;
}

//確率 p で 1 を乱数生成
int rnd_bin(double p) {
   int S;
   if ((double)genrand_real2() >= (1.0 - p)) { //(0,1)
   S = 1;
   }
   else {
   S = 0;
   }
   return S;
}

// 初期設定
// サーバー停止中
// バッファ空

int main(void) {
   FILE *file;
   file = fopen("q1_p kankei.txt", "w");
   /***************ループ関係 [サービス時間可変]*********/
   double service_time[2]; // 各シンボルのサービス時間

   fprintf(file, "最小時間平均 AoI となる q1\t1 の発生確率 p1\n");
   double SRC_PROB = 0.01; //シンボル"1"が生成する確率
   while (SRC_PROB <= 0.5) {
      double q0 = 0.3;   //最小値探索だけだから q1=1-0.3 から開始
      double q1min = 0.0;   //AoI が最小になるときの q1
      double AoImin = 0.0;
      double q1;
      double kraft = 0.0; //クラフト和
      while (q0 < 1.0) {
      q1 = 1.0 - q0;
      service_time[0] = -log2(q0);
      service_time[1] = -log2(q1);
      kraft = pow(2.0, (-1.0) * service_time[0]) + pow(2.0, (-1.0) * service_time[1]);
      /*===シンボル 1 専用の AoI 計算関連=================*/ //if(src_s or svr_s == 1){処理};
      double t1_arv = 0.0;   //シンボルの到着時刻
      double t1_dep = -1.0;    //シンボルの出発時刻 (-1:サービス終了)
      double svr1_a;    //サーバーの中のシンボルの到着時刻
      double pre1_arv = 0.0;   //ひとつ前の到着
      double pre1_dep = 0.0;   //ひとつ前の出発
      double area1 = 0.0;   //各時間の AoI(シンボル 1 について)
      int num1_arv = 0;   //0 の到着数
      int num1_svr = 0;  //サーバに入った 0 の数
      int num_svr = 0;  //サーバに入った 0 の数
      double rejection_rate;  //棄却率 (= 1.0-サーバに入った 1 の割合)
```

```
      /*===========計測用初期値=========================*/
#define ARRIVAL_RATE 1.0  //←←←←←←←←←←←←到着レート初期値
      double t = 0.0;    //現在時刻
      double t_arv = 0.0;   //シンボルの到着時刻
      double t_dep = -1.0;    //シンボルの出発時刻 (-1:サービス終了)
      int count_arrival = 0; //到着シンボル数
#define NUM_ARRIVAL 1000000 //到着数の上限
      int src_s;   //到着するシンボルの種類
      int svr_s;    //サーバーの中のシンボル種類
      double svr_a;    //サーバーの中のシンボルの到着時刻
      int buf_s = -1;    //バッファの中のシンボル種類 (空だったら-1)
      double buf_a;    //バッファの中のシンボルの到着時刻
      int count_dep = 0;   //サーバーを出発した数
      double area = 0.0;    //各時間での AoI(2 つのシンボル)
      double aoi_ave;   //全体の時間平均 AOI
      double pre_arv = 0.0;   //ひとつ前の到着
      double pre_dep = 0.0;   //ひとつ前の出発
      double start;   //計測開始時刻 (最初にサーバー出発した時刻)
      double last;   //計測終了時刻 (最後にサーバーを出発した時刻)

      /*============本文================*/
      init_genrand(19650218UL);
      t_arv = rnd_exp(ARRIVAL_RATE);  //最初の到着時刻
      src_s = rnd_bin(SRC_PROB);   //最初に到着するシンボルの種類
      if (src_s == 1) {
      num1_arv++;
      t1_arv = t_arv;
      //printf("最初のシンボルは 1．時刻は %lf\n", t1_arv);
      }
      while (t_arv >= 0 || t_dep >= 0) {
      if ((t_arv >= 0 && t_dep >= 0 && t_dep < t_arv) || (t_arv < 0 && t_dep >= 0)) {
         // サービス終了
                        //printf("サービス終了処理イン\n");
         count_dep++;
         t = t_dep;
         //printf("\t 現在時刻 %lf\t シンボルの出発数 %d\n", t, count_dep);
         if (count_dep == 1) {
         //計測開始
         start = t;
         pre_arv = svr_a;
         pre_dep = t_dep;
         if (src_s == 1) {
            pre1_arv = pre_arv;
            pre1_dep = pre_dep;
         }
         //printf("最初の到着時刻 %lf\t 最初の出発時刻 %lf\n", pre_arv, pre_dep);
         }
         else {
         //AOI 計算
         //area += 1.0 / 2.0 * (pow(t_dep - pre_arv , 2) - pow(pre_dep - pre_arv , 2));//三角形
         area += 0.5 * ((t_dep - pre_arv) + (pre_dep - pre_arv)) * (t_dep - pre_dep);//台形

         pre_arv = svr_a;//モニタにシンボルの到着時刻
         pre_dep = t_dep;//直前に出発したシンボルの出発時刻
                  //printf("表示されているシンボル 1 の出発時刻 %lf\t 一つ前の 1 の到着時刻 %lf\t 一つ前の 1 の出発時
刻 %lf\n", t1_dep, pre1_arv, pre1_dep);
         /*if(svr_s == 1){
            area1 += 0.5 * ((t1_dep - pre1_arv) + (pre1_dep - pre1_arv)) * (t1_dep - pre1_dep);//台形
            pre1_arv = svr1_a;
            pre1_dep = t1_dep;
         }    */          //printf("aoi1\t%lf\n",area1);
         }
         if (buf_s < 0) {
         // バッファが空
         t_dep = -1; // サーバー停止
         }
         else {
         // バッファ満
         svr_s = buf_s;
```

```
        svr_a = buf_a;
        t_dep = t + service_time[svr_s];
        buf_s = -1; // バッファを空にする
        num_svr++;
        if (svr_s == 1) {
           num1_svr++;
        }
        // printf("バッファからサーバに移動\t 現在時刻 %lf\t バッファ内のサービス時間 %lf\n", t, service_time[svr_s]);
        }
        //printf("サービス終了処理アウト\n\n");
     }
     else {
        // シンボル到着
                  //printf("シンボル到着処理イン\n");
        count_arrival++;
        t = t_arv;
        //printf("\t 現在時刻 %lf\t シンボルの到着数 %d\n", t, count_arrival);
        if (t_dep < 0) {
        // サーバーに入る
        num_svr++;      //printf("サーバーイン処理\n");
        svr_s = src_s;
        svr_a = t;
        t_dep = t + service_time[svr_s];
        if (svr_s == 1) {
           t1_dep = t_dep;
           svr1_a = svr_a;
           num1_svr++;
        }
        //printf("サーバイン処理\t 現在時刻 %lf\t サーバーのシンボル種類 %d\t 出発予定時刻 %lf\t サーバを出た 1 の数
(分子) %d\n", t, svr_s, t_dep, num1_svr);
        }
        else {
        // バッファに入る (割り込みかも)
                  //printf("バッファイン処理 (割り込み) \n");
        buf_s = src_s;
        buf_a = t;
        //printf("\t バッファのシンボル種類 %d\n", buf_s);
        }
        // 次の到着シンボルの準備
        if (count_arrival < NUM_ARRIVAL) {
        t_arv = t + rnd_exp(ARRIVAL_RATE);
        src_s = rnd_bin(SRC_PROB);
        if (src_s == 1) {
           num1_arv++;
           t1_arv = t_arv;
        }                       // printf("(準備\t 次の到着シンボル %d\t 次の到着時刻 %lf)1 の到着する数 (分
母) %d\n", src_s, t_arv, num1_arv);
        }
        else {
        t_arv = -1;
        }
        //printf("シンボル到着処理アウト\n\n");
     }
     }
     last = t;
     //時間平均 AOI
     aoi_ave = area / (last - start);

     rejection_rate = 1.0 - (double)num_svr / NUM_ARRIVAL;
     if (AoImin > aoi_ave || AoImin == 0.0) {
     AoImin = aoi_ave;
     q1min = q1;
     //printf("%lf\t%lf\n", AoImin, q1min);
     }
     q0 = q0 +0.01;
  }

  fprintf(file, "%lf\t%lf\t%lf\n", q1min, SRC_PROB, AoImin);
  SRC_PROB += 0.01;
```

```
    }
    fclose(file);
}
```