信州大学工学部

学士論文

# サーバで割り込みのある通信システムで多値情報源を符号化したときの情報鮮度の数学的解析

指導教員　西新 幹彦 准教授

学科　電子情報システム工学科
学籍番号　21T2110J
氏名　野田 哲平

2025 年 4 月 24 日

# 目次


1 **はじめに** 1

2 **AoI とは** 1

2.1 AoI の定義とその評価方法 . . . . . 2

2.2 AoI の導出方法 . . . . . 3

3 **二値情報源の符号化** 6

4 **多値情報源の符号化** 10

5 **シミュレーションによる検証** 15

5.1 シミュレーション方法 . . . . . 15

5.2 二値情報源の場合の再検証 . . . . . 16

5.3 多値情報源の場合の検証 . . . . . 17

**謝辞** 19

**参考文献** 19

**付録 A プログラム** 20

# 1　はじめに

遠隔地から観測する防犯カメラ，ネットワークカメラ，あるいは河川の水位監視カメラなどでは，素早く最新の情報を映し出す必要がある．しかし情報は通信路を通じて送信されている間に古くなり，モニタに表示される時には，程度はともかく過去のものとなっている．さらに，表示された情報も次の情報が届くまで時間の経過とともに古くなり続ける．このような情報の新しさを定量的に表現する指標として情報鮮度 (Age of Information: AoI) という概念がある．AoI は数値で表され，その値は小さいほど情報が新しいことを意味する．本論文では単にこれを AoI と呼ぶ.

この問題に対して従来の研究 [1] では，センサでサンプリングされた情報が送信機に到着した順番に処理される FIFO(First In First Out) のシステムが考えられている．また，別の研究 [2] 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）を用いたシステムが考えられている．その後，この問題は，システムの中に符号器が追加され，情報理論的な最適化問題として考えられるようになった．研究 [3] で情報を符号語に変換し，シミュレーション実験によって AoI の変化が調べられている．そして，研究 [4] にて，研究 [3] と同様の通信システムであるシンボルごとにサービスレートが変わる通信モデルを設定し，二値情報源で AoI の解析が行われている．

しかし，現実世界の通信システムにおいて，先行研究 [4] のシンボルが 2 種類というのは，少なすぎるといえる．川に設置された監視カメラの例をあげたが，シンボル 2 種類の小さいアルファベットサイズであるとカメラの解像度が低く，川の濁りや漂流物などの細かい異常を見逃す可能性がある．シンボルを増やして，アルファベットサイズを大きくすることで水質の変化や川岸の状態，漂流物の種類や色の特徴などを詳細に分析できる．よって，シンボルの種類は多い方がよい．そこで，本研究ではシンボルの数が一般に $n$ 個の情報源に対して，サーバで割り込みがある待ち行列通信システムを数学的に解析して，時間平均 AoI を表す数式を導いた．

# 2　AoI とは

本章では，AoI の定義およびその求め方について，先行研究 [5] での内容を基に説明する．

本研究で考える通信システムを図 1 に示す．このシステムでは情報源からポアソン過程に従い出力された情報は，発生と同時に図中の矢印で表された通信路を通りサーバでシンボルの処理が行われる．その後，通信路を通り遠隔地のモニタへ表示される．図 1 のように，サーバにシンボルが保持されている状態で新たなシンボルが情報源から発生し，サーバに到着した場合，既にサーバで処理されているシンボルは棄却され，新しいシンボルがサーバで処理されモニタに表示される．

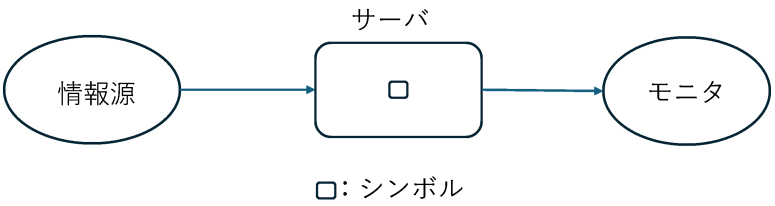


図 1: 通信システム図

## 2.1 AoI の定義とその評価方法

AoI とは，情報が発生してからの経過時間で，情報の年齢と同義である．以降はこれを単に AoI と呼ぶ．本研究では，図 1 のモニタに表示されている情報の AoI を考える．

定義より時刻 $t$ のモニタに表示されている情報の AoI $A_t$ はその情報の発生時刻を $a$ とおくと，

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

と書ける．$A_t$ の具体例を図 2 に示す．実線がモニタの AoI を表し，点線はサーバの AoI を表している．時刻 $a_1$ に情報源から発生したシンボル $X_1$ がモニタに表示された瞬間の AoI は，

$$A_1 = d_1 - a_1 \tag{2}$$

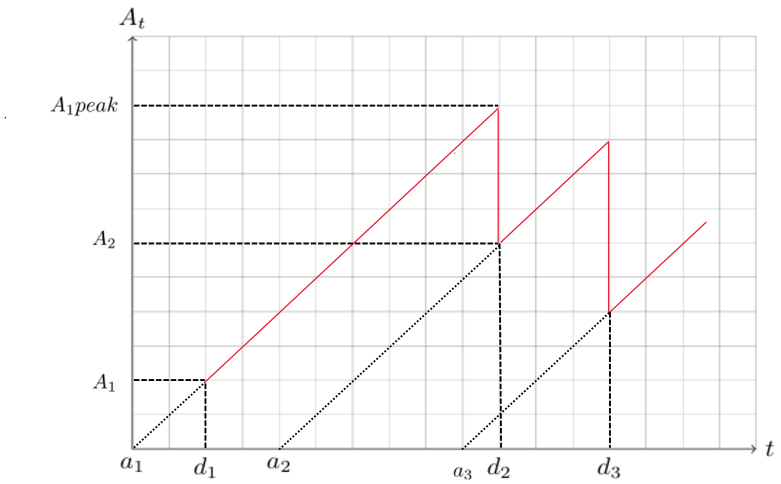


図 2: AoI の遷移

と書ける．また，時刻 $a_2$ で情報源から発生したシンボル $X_2$ は，時刻 $d_2$ でモニタに表示される．その時，モニタの表示内容は古い情報であるシンボル $X_1$ から，シンボル $X_2$ の情報へと更新される．よって，時刻 $d_2$ で，AoI の値は，$A_{1peak}$ から，$A_2$ へ瞬時に下がる．このように次の新しいシンボルがモニタに到着すると，モニタに表示されていた古いシンボルは新しいシンボルへと更新される．このように，AoI の値が更新することを繰り返すためモニタの AoI の値は，図 2 の赤線のようなノコギリ状の実線グラフとなる．これが，AoI の時間的変化である．

本研究ではこの AoI の時間平均を評価する．時間平均 $A_{ave}$ は，観測時間を $T$，時刻 $t$ における AoI を $A_t$ とすると，

$$A_{\text{ave}} = \lim_{T\to\infty} \frac{1}{T} \int_0^T A_t \, dt \tag{3}$$

と書ける．

## 2.2 AoI の導出方法

この節では，情報源からのシンボルが，到着レート $\lambda$ のポアソン過程に従って到着すると仮定する．サーバでのサービス時間は指数分布によって決定し，サービスレートはシンボルの種

表 1: 状態番号と図 1 のサーバの状態

| 状態番号 | サーバの状態 |
|---|---|
| 0 | サーバが空の状態 |
| 1 | サーバでシンボルが処理されている状態 |

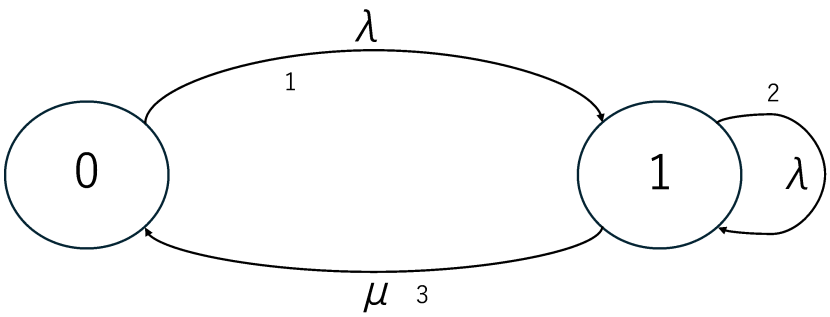


図 3: 図 1 の通信システムの状態遷移図

表 2: 図 1 のマルコフ連鎖の遷移表

| $l$ | $q_l \to q_l'$ | $\lambda^{(l)}$ | $xA_l$ | $A_l$ | $v_{q_l}A_l$ |
|---|---|---|---|---|---|
| 1 | $0 \to 1$ | $\lambda$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{00} & 0 \end{bmatrix}$ |
| 2 | $1 \to 1$ | $\lambda$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{10} & 0 \end{bmatrix}$ |
| 3 | $1 \to 0$ | $\mu$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & * \\ 1 & * \end{bmatrix}$ | $\begin{bmatrix} v_{11} & * \end{bmatrix}$ |

表 3：変数の説明

| **変数** | **説明** |
|---|---|
| $Q$ | 状態の集合 |
| $q$ | 特定の状態 |
| $v_q$ | 状態 $q$ における AoI の期待値 |
| $\sum_{l \in L_q'} \lambda^{(l)}$ | 状態 $q$ から他の状態へ遷移するすべての遷移率の総和 |
| $\sum_{l \in L_q'} \lambda^{(l)} v_{q_l} A_l$ | 他の状態 $q_l$ から状態 $q$ に遷移する期待値と遷移レートの積 |
| $\pi_q$ | 状態 $q$ にいる確率 |
| $b_q$ | 状態 $q$ における モニタとサーバの AoI の傾き |
| $\pi_q b_q$ | 状態 $q$ であり続ける期待値 |

類によらず $\mu$ とする．この設定において，モニタに表示されるシンボルの AoI の導出方法は，先行研究 [5] で明らかにされている．以下で，これを具体的に説明する．

このようなシステムのサーバは表 1 のような 2 つの状態をとることが分かる．状態 0 はサーバが空の状態，状態 1 はサーバでシンボルが処理されている状態を表している．これら状態の互いの遷移関係を，自己遷移も含めて記載したのが図 3 となる．それぞれの遷移を区別するために図 3 のように遷移番号をつける．

遷移番号 1 は，サーバが空の状態で情報源 1 からサーバにシンボルが到着したときの遷移であり，このときの遷移は到着レート $\lambda$ で表される．同様に，遷移番号 2 は，サーバでシンボル処理中に新しく情報源からシンボルが発生する遷移を表しており，このときの遷移は到着レー

ト $\lambda$ で表される．この際，処理中だった古いシンボルは棄却され新しいシンボルが処理される．一方，遷移番号 3 はサーバ内のシンボルが処理されサービス終了したことを示しており，このときのレートはサーバのサービスレート $\mu$ で表される．

以上を基に遷移の詳細を表 2 のように表す．遷移番号 1 を例に各項目を説明する．遷移番号 1 は，状態 $q_l = 0$ から状態 $q_l' = 1$ の遷移を表し，その遷移レートは $\lambda$ である．$xA_l$ はサーバの状態が変化した直後のモニタとサーバの年齢を表しており，遷移する直前のモニタとサーバの年齢をそれぞれ $x_0, x_1$ とおき，横ベクトルとして $[x_0\ x_1]$ と表す．状態 0 の時，モニタではシンボルが表示されており，サーバは空である．ここから状態 1 へと遷移した直後は，情報源から発生したシンボルがサーバに到着し，処理が開始された状態である．この時モニタの表示内容は変化していないため年齢は $x_0$ のままであり，サーバの中には生まれたてのシンボルが保持されているためその年齢は 0 となる．このため状態 0→ 状態 1 への遷移直後のモニタとサーバの年齢は $[x_0\ 0]$ と書ける．ここで，状態 0→ 状態 1 へと変換するための行列 $A_l$ は，

$$A_l \begin{bmatrix} x_0 & x_1 \end{bmatrix} = \begin{bmatrix} x_0 & 0 \end{bmatrix} \tag{4}$$

を満たす行列であり，

$$A_l = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5}$$

となる．また，$v_{q_l}$ は $[v_{q_l 0}\ v_{q_l 1}]$ からなるベクトルであり，遷移番号 1 では $q_l$=0 である．よって，$v_{q_l} A_l$ は

$$v_{q_l} A_l = \begin{bmatrix} v_{00} & 0 \end{bmatrix} \tag{6}$$

となる．他の遷移も同様にして表 2 が得られる．なお，記号 $*$ は値を持たないことを表している．

表 2 で得られた値と未知変数 $v_{q_l}\pi_q$ を用いて連立方程式

$$v_q \sum_{l \in L_q} \lambda^{(l)} = b_q \pi_q + \sum_{l \in L_q'} \lambda^{(l)} v_{q_l} A_l, \quad q \in Q \tag{7}$$

を立てる．この式 (7) に含まれる変数の意味を表 3 に示す．$\pi_q$ はその状態にいる確率である定常確率を表し，

$$\pi_q \sum_{l \in L_q} \lambda^{(l)} = \sum_{l \in L_q'} \lambda^{(l)} \pi_{q_l}, \quad q \in Q \tag{8}$$

と定義される．$\pi_q$ は確率であるため，

$$\sum_{q \in Q} \pi_q = 1 \tag{9}$$

と書ける．

実際に，表 2 から得られた値を式 (7) および式 (8) に代入すると，

$$\begin{bmatrix} v_{00} & v_{01} \end{bmatrix} \lambda = \begin{bmatrix} 1 & 0 \end{bmatrix} \pi_0 + \mu \begin{bmatrix} v_{11} & * \end{bmatrix}, \tag{10}$$

$$\begin{bmatrix} v_{10} & v_{11} \end{bmatrix} (\mu + \lambda) = \begin{bmatrix} 1 & 1 \end{bmatrix} \pi_1 + \lambda \begin{bmatrix} v_{00} & 0 \end{bmatrix} + \lambda \begin{bmatrix} v_{10} & 0 \end{bmatrix}, \tag{11}$$

$$\pi_0 \lambda = \mu \pi_1, \tag{12}$$

$$\pi_1 (\mu + \lambda) = \lambda \pi_0 + \lambda \pi_1 \tag{13}$$

を得る．式 (12), (13) について解くと，

$$\pi_0 = \frac{\mu}{\mu + \lambda} \tag{14}$$

$$\pi_1 = \frac{\lambda}{\mu + \lambda} \tag{15}$$

が得られ，これらを用いて式 (10), (11) について解くと $v_{00}$, $v_{10}$ は，

$$v_{00} = \frac{1}{\mu} \left\{ \frac{\lambda^2 + \mu^2 + 3\lambda\mu}{(\mu + \lambda)^2} \right\} \tag{16}$$

$$v_{10} = \frac{\mu}{\lambda} \left\{ \frac{\mu + 2\lambda}{(\mu + \lambda)^2} \right\} \tag{17}$$

と表される．一般にモニタの時間平均 AoI は $\sum_{q \in Q} v_{q0}$ と表されるため

$$\Delta = \sum_{q \in Q} v_{q0} \tag{18}$$

$$= v_{00} + v_{10} \tag{19}$$

$$= \frac{\mu + \lambda}{\mu \lambda} \tag{20}$$

と求められる．

# 3　二値情報源の符号化

ここまでは，すべてのシンボルが同じサービスレートと到着レートで処理されるシステムを考えた．ここからは，通信システムの中に符号器を追加しサービスレートがシンボルごとに異なる通信システムのモニタ AoI を数理的に導出する．手法そのものは 2 章で説明したものと同じである．本章の内容は先行研究 [4] の内容の紹介である．

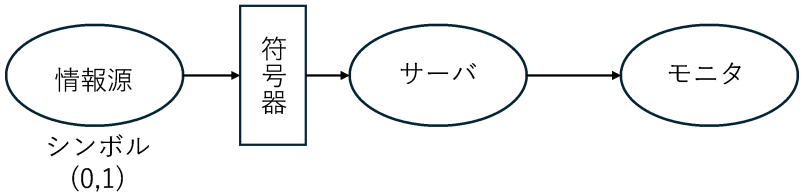


図 4: サーバで割り込みのある符号器を追加した通信システム

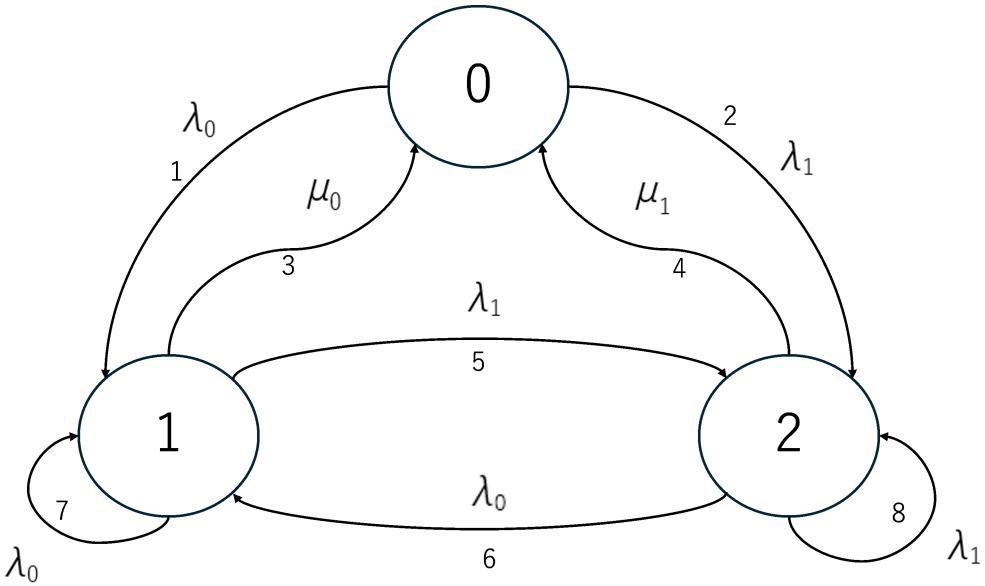


図 5: サーバで割り込みのある符号器を追加した通信システムの状態遷移図

符号器を追加した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．情報源について，シンボルがポアソン過程で発生すると考え，シンボルの種類は‘0’,‘1’ の 2 種類のみである．シンボル 0 の到着レートとサービスレートをそれぞれ $\lambda_0$，$\mu_0$ とする．同様に，シンボル 1 の到着レートとサービスレートをそれぞれ $\lambda_1$，$\mu_1$ とする．シンボルは符号器までポアソン到着し，符号器で符号語に変換されサーバで処理された後，遠隔地のモニタへ表示される．サーバでのサービス時間は符号語長を期待値とした指数分布によって決定する．また，システムでは新しい情報を常に送りたいのでサーバでシンボルが処理されているときに新しいシンボルが到着した場合，処理しているシンボルを棄却し，新しいシンボルが処理される．

この時サーバの考えられる状態は，空の状態，シンボル 0 を処理中，シンボル 1 を処理中の 3 つの状態のいずれかである．このようなシステムの状態遷移図を描くと図 5 のようになる．

状態 0 はサーバが空の状態，状態 1 はサーバでシンボル‘0’が処理されている状態，状態 2 はサーバでシンボル‘1’が処理されている状態を表している．このシステムの状態遷移の詳細は表 4 のようになる．

表 4 から得られた値を式 (7) および式 (8) に代入すると，

$$[v_{00}\ v_{01}](\lambda_0+\lambda_1) = [1\ 0]\pi_0 + \mu_0[v_{11}*] + \mu_1[v_{21}*], \tag{21}$$

$$[v_{10}\ v_{11}](\mu_0+\lambda_0+\lambda_1) = [1\ 1]\pi_1 + \lambda_0[v_{00}\ 0] + \lambda_0[v_{10}\ 0] + \lambda_0[v_{20}\ 0], \tag{22}$$

$$[v_{20}\ v_{21}](\mu_1+\lambda_0+\lambda_1) = [1\ 1]\pi_2 + \lambda_1[v_{00}\ 0] + \lambda_1[v_{10}\ 0] + \lambda_1[v_{20}\ 0], \tag{23}$$

$$\pi_0(\lambda_0+\lambda_1) = \mu_0\pi_1 + \mu_1\pi_2, \tag{24}$$

$$\pi_1(\mu_0+\lambda_0+\lambda_1) = \lambda_0\pi_0 + \lambda_0\pi_1 + \lambda_0\pi_2, \tag{25}$$

$$\pi_2(\mu_1+\lambda_0+\lambda_1) = \lambda_1\pi_0 + \lambda_1\pi_1 + \lambda_1\pi_2. \tag{26}$$

表 4: 図 4 のマルコフ連鎖の遷移表

| $l$ | $q_l \to q_l'$ | $\lambda^{(l)}$ | $xA_l$ | $A_l$ | $v_{q_l}A_l$ |
|---|---|---|---|---|---|
| 1 | $0 \to 1$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{00} & 0 \end{bmatrix}$ |
| 2 | $0 \to 2$ | $\lambda_1$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{00} & 0 \end{bmatrix}$ |
| 3 | $1 \to 0$ | $\mu_0$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & 1 \\ * & * \end{bmatrix}$ | $\begin{bmatrix} v_{11} & * \end{bmatrix}$ |
| 4 | $2 \to 0$ | $\mu_1$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & 1 \\ * & * \end{bmatrix}$ | $\begin{bmatrix} v_{21} & * \end{bmatrix}$ |
| 5 | $1 \to 2$ | $\lambda_1$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{10} & 0 \end{bmatrix}$ |
| 6 | $2 \to 1$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{20} & 0 \end{bmatrix}$ |
| 7 | $1 \to 1$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{10} & 0 \end{bmatrix}$ |
| 8 | $2 \to 2$ | $\lambda_1$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{20} & 0 \end{bmatrix}$ |

という連立方程式が立てられる．式 (24),(25) および式 (26) について解くと，

$$\pi_0 = \frac{1}{\lambda_0+\lambda_1}\left(\frac{\mu_0\lambda_0}{\mu_0+\lambda_0+\lambda_1}+\frac{\mu_1\lambda_1}{\mu_1+\lambda_0+\lambda_1}\right) \tag{27}$$

$$\pi_1 = \frac{\lambda_0}{\mu_0+\lambda_0+\lambda_1} \tag{28}$$

$$\pi_2 = \frac{\lambda_1}{\mu_1+\lambda_0+\lambda_1} \tag{29}$$

が得られ，これらを用いて式 (21),(22) および式 (23) について解くと $v_{00}$,$v_{10}$,$v_{20}$ は，

$$v_{10} = (\lambda_0^2+\mu_1\lambda_0+\lambda_0\lambda_1)v_{00} + \frac{1}{\mu_0\mu_1+\mu_0\lambda_0+\mu_1\lambda_1}\cdot\left\{\frac{\mu_0\lambda_0\lambda_1+\mu_1\lambda_0\lambda_1+\mu_1^2\lambda_0+2\mu_1\lambda_0^2+2\lambda_0^2\lambda_1+\lambda_0^3+\lambda_0\lambda_1^2}{(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)}\right\} \tag{30}$$

$$v_{20} = (\lambda_1^2+\mu_0\lambda_1+\lambda_0\lambda_1)v_{00} + \frac{1}{\mu_0\mu_1+\mu_0\lambda_0+\mu_1\lambda_1}\cdot\left\{\frac{\mu_0\lambda_0\lambda_1+\mu_1\lambda_0\lambda_1+\mu_0^2\lambda_1+2\mu_0\lambda_1^2+2\lambda_0\lambda_1^2+\lambda_0^3+\lambda_0\lambda_1^2}{(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)}\right\} \tag{31}$$

$$v_{00} = \frac{\mu_0^2\lambda_1+2\mu_0\lambda_0(\lambda_0+\lambda_1)}{(\lambda_0+\lambda_1)^2(\mu_0+\lambda_0+\lambda_1)^2}+\frac{\mu_0^2\lambda_1+2\mu_1\lambda_1(\lambda_0+\lambda_1)}{(\lambda_0+\lambda_1)^2(\mu_1+\lambda_0+\lambda_1)^2} \tag{32}$$

と表される．一般に $v_{00}, v_{10}, v_{20}$ の総和がモニタの時間平均 AoI となるため，求めたい値を $\Delta$ とすると，

$$\Delta = v_{00}+v_{10}+v_{20} \tag{33}$$

$$= \frac{1}{\mu_0\mu_1+\mu_0\lambda_0+\mu_1\lambda_1}\cdot\{(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)v_{00} + \frac{(\lambda_0+\lambda_1)^3+2(\mu_1\lambda_0+\mu_0\lambda_1)(\lambda_0+\lambda_1)+\mu_0^2\lambda_1+\mu_1^2\lambda_0}{(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)}\Bigg\} \tag{34}$$

この式に式 (32) を代入して整理すると，

$$\Delta = \frac{1}{\mu_0\mu_1+\mu_0\lambda_0+\mu_1\lambda_1}\cdot\left\{\frac{\{(\lambda_0+\lambda_1)^2+(\mu_0+\mu_1)(\lambda_0+\lambda_1)+\mu_0\mu_1\}}{(\lambda_0+\lambda_1)^2(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)}\right.$$
$$\left.\frac{(\lambda_0+\lambda_1)^3+(\mu_0+\mu_1)(\lambda_0+\lambda_1)^2+\mu_0\mu_1}{(\lambda_0+\lambda_1)^2(\mu_0+\lambda_0+\lambda_1)(\mu_1+\lambda_0+\lambda_1)}\right\} \tag{35}$$

$$= \frac{1}{\mu_0\mu_1+\mu_0\lambda_0+\mu_1\lambda_1}\left(\lambda_0+\lambda_1+\mu_0+\mu_1+\frac{\mu_0\mu_1}{\lambda_0+\lambda_1}\right) \tag{36}$$

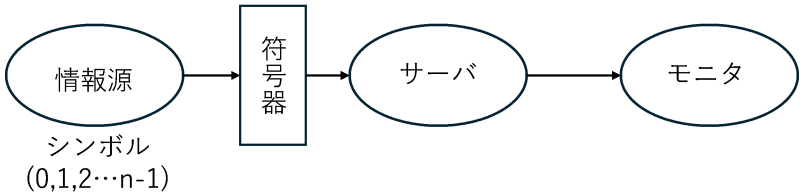


図 6: サーバで割り込みのある符号器を追加した通信システム

と求められる．ただし，(36) は式変形して，

$$\Delta = \frac{1}{\mu_0\mu_1 + \mu_0\lambda_0 + \mu_1\lambda_1}\left(\lambda_0 + \lambda_1 + \mu_0 + \mu_1 + \frac{\mu_0\mu_1}{\lambda_0 + \lambda_1}\right) \tag{37}$$

$$= \frac{(\lambda_0 + \lambda_1)^2 + (\lambda_0 + \lambda_1)(\mu_0 + \mu_1) + \mu_0\mu_1}{(\mu_0\mu_1 + \mu_0\lambda_0 + \mu_1\lambda_1)(\lambda_0 + \lambda_1)} \tag{38}$$

$$= \frac{(\mu_0 + \lambda_0 + \lambda_1)(\mu_1 + \lambda_0 + \lambda_1)}{\mu_0\mu_1\lambda_0 + \mu_0\lambda_0^2 + \mu_0\lambda_0\lambda_1 + \mu_0\mu_1\lambda_1 + \mu_1\lambda_0\lambda_1 + \mu_1\lambda_1^2} \tag{39}$$

$$= \frac{(\mu_0 + \lambda_0 + \lambda_1)(\mu_1 + \lambda_0 + \lambda_1)}{(\mu_0 + \lambda_0 + \lambda_1)\mu_1\lambda_1 + (\mu_1 + \lambda_0 + \lambda_1)\mu_0\lambda_0} \tag{40}$$

とも表すことができる．

## 4　多値情報源の符号化

先行研究では，システムに符号器を追加し，各シンボルに応じてサービスレートが変化する通信モデルを扱い，シンボルが‘0’と‘1’の 2 種類である場合のモニタの時間平均 AoI を解析した．本研究では，同様の通信モデルを基に，シンボルの種類を $n$ 種類に一般化し，モニタ

表 5: 状態番号とサーバの状態

| 状態番号 | サーバの状態 |
|---|---|
| 0 | サーバでシンボル 0 が処理されている状態 |
| 1 | サーバでシンボル 1 が処理されている状態 |
| 2 | サーバでシンボル 2 が処理されている状態 |
| ⋮ | ⋮ |
| $n-1$ | サーバでシンボル $n-1$ が処理されている状態 |
| $n$ | サーバが空の状態 |

AoI を数理的に表現できるかを検討した．すなわち，本研究で扱う通信システムの構造は図 4 と同じであるが，シンボルが $n$ 個である点が拡張されている図 6 のような構造となる．以下から，シンボルの個数は $n$ で固定して考える．導出されるモニタの時間平均 AoI を $\Delta$ としその結果から示すと，

$$\Delta = \frac{1}{\sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{\mu_i + \Lambda}} \tag{41}$$

ただし，

$$\Lambda = \sum_{i=0}^{n-1} \lambda_i \tag{42}$$

である．本研究結果の式 (41) に $n$=1,2 を代入するとそれぞれ，先行研究結果である式 (20),(40) と一致することが確認できる．以下この解の導出過程を詳述する．

図 6 のような通信システムの構造と条件に基づくと，サーバの状態は，サーバが空の状態と特定のシンボルが処理されている状態を取ることが分かり，その詳細は表 5 のようにまとめることができる．この表 5 を基に，遷移図を考察する．各状態の遷移規則を考えると，すべての

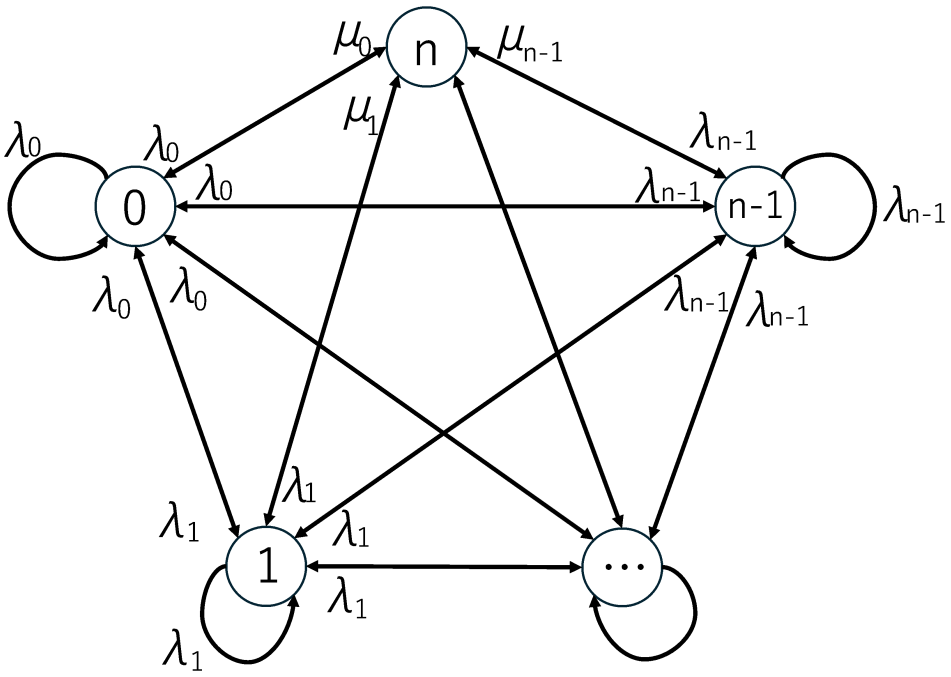


図 7: サーバで割り込みのある符号器を追加した通信システムの状態遷移図 (シンボル $n$ 個)

表 6: 図 6 のマルコフ連鎖の遷移表

| $l$ | $q_l \to q_l'$ | $\lambda^{(l)}$ | $xA_l$ | $A_l$ | $v_{q_l}A_l$ |
|---|---|---|---|---|---|
| $(0,n)$ | $0 \to n$ | $\mu_0$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & * \\ 1 & * \end{bmatrix}$ | $\begin{bmatrix} v_{01} & * \end{bmatrix}$ |
| $(1,n)$ | $1 \to n$ | $\mu_1$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & * \\ 1 & * \end{bmatrix}$ | $\begin{bmatrix} v_{11} & * \end{bmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $(i,n)$ | $i \to n$ | $\mu_i$ | $\begin{bmatrix} x_1 & * \end{bmatrix}$ | $\begin{bmatrix} 0 & * \\ 1 & * \end{bmatrix}$ | $\begin{bmatrix} v_{i1} & * \end{bmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $(1,0)$ | $1 \to 0$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{10} & 0 \end{bmatrix}$ |
| $(2,0)$ | $2 \to 0$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{20} & 0 \end{bmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $(n,0)$ | $n \to 0$ | $\lambda_0$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{n0} & 0 \end{bmatrix}$ |
| $(0,1)$ | $0 \to 1$ | $\lambda_1$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{00} & 0 \end{bmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $(j,i)$ | $j \to i$ | $\lambda_i$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{i0} & 0 \end{bmatrix}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $(n-1,n-1)$ | $n-1 \to n-1$ | $\lambda_{n-1}$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{n-1\,0} & 0 \end{bmatrix}$ |
| $(n,n-1)$ | $n \to n-1$ | $\lambda_{n-1}$ | $\begin{bmatrix} x_0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} v_{n\,0} & 0 \end{bmatrix}$ |

遷移は規則的に定義することができ，その遷移図は図 7 のようになる．任意の 2 つの状態間は互いに遷移が存在し，状態 0 以外の状態には自己遷移が存在する．そのもとで，状態 $j$ から状態 $i$ への遷移には，ラベル $(j,i)$ を付与する．各遷移の詳細をまとめると表 6 のようになる．これらの遷移は，サーバが空の状態に対応するラベル $(i,n)$ の場合と，サーバでいずれかのシンボルが処理中であるラベル $(j,i)$ の場合とで一般化できる．2.2 節での導出方法より，

$$
\begin{aligned}
&[v_{n0}\ v_{n1}](\lambda_0+\lambda_1+\cdots+\lambda_{n-1})\\
&=[1\ 0]\pi_n+\mu_0[v_{01}\ *]+\mu_1[v_{11}\ *]+\mu_2[v_{21}\ *]+\cdots+\mu_{n-1}[v_{n-1\,1}\ *]
\end{aligned}
\tag{43}
$$

$$
\begin{cases}
[v_{00}\ v_{01}](\mu_0+\lambda_0+\cdots+\lambda_{n-1})\\
=[1\ 1]\pi_0+\lambda_0[v_{00}\ 0]+\lambda_0[v_{10}\ 0]+\lambda_0[v_{20}\ 0]+\cdots+\lambda_0[v_{n0}\ 0],\\
[v_{10}\ v_{11}](\mu_1+\lambda_0+\cdots+\lambda_{n-1})\\
=[1\ 1]\pi_1+\lambda_1[v_{00}\ 0]+\lambda_1[v_{10}\ 0]+\lambda_1[v_{20}\ 0]+\cdots+\lambda_1[v_{n0}\ 0],\\
\qquad\qquad\qquad\vdots\\
[v_{n-1\,0}\ v_{n-1\,1}](\mu_{n-1}+\lambda_0+\cdots+\lambda_{n-1})\\
=[1\ 1]\pi_{n-1}+\lambda_{n-1}[v_{00}\ 0]+\lambda_{n-1}[v_{10}\ 0]+\lambda_{n-1}[v_{20}\ 0]+\cdots+\lambda_{n-1}[v_{n0}\ 0].
\end{cases}
\tag{44}
$$

$$
\pi_n(\lambda_0+\lambda_1+\cdots+\lambda_{n-1})=\mu_0\pi_0+\mu_1\pi_1+\mu_2\pi_2+\mu_3\pi_3+\cdots+\mu_{n-1}\pi_{n-1}
\tag{45}
$$

$$
\begin{cases}
\pi_0(\mu_0+\lambda_0+\lambda_1+\lambda_2+\cdots+\lambda_{n-1})=\lambda_0\pi_0+\lambda_0\pi_1+\lambda_0\pi_2+\cdots+\lambda_0\pi_n\\
\pi_1(\mu_1+\lambda_0+\lambda_1+\lambda_2+\cdots+\lambda_{n-1})=\lambda_1\pi_0+\lambda_1\pi_1+\lambda_1\pi_2+\cdots+\lambda_1\pi_n\\
\qquad\qquad\qquad\vdots\\
\pi_{n-1}(\mu_{n-1}+\lambda_0+\lambda_1+\lambda_2+\cdots+\lambda_{n-1})=\lambda_{n-1}\pi_0+\lambda_{n-1}\pi_1+\lambda_{n-1}\pi_2+\cdots+\lambda_{n-1}\pi_n
\end{cases}
\tag{46}
$$

という 4 つの連立方程式が立てられる．ここで,

$$
B_i=\mu_i+\Lambda,\quad i=0,1,\ldots,n-1
\tag{47}
$$

とする．連立方程式を定義 (47) を利用して整理可能な部分をまとめると，式 (43) から,

$$
v_{n0}\sum_{i=0}^{n-1}\lambda_i=\pi_n+\sum_{i=0}^{n-1}\mu_i v_{i1}
\tag{48}
$$

が，式 (44) から

$$
v_{i0}B_i=\pi_i+\lambda_i\sum_{j=0}^{n}v_{j0},\qquad i=0,1,\ldots,n-1
\tag{49}
$$

$$
v_{i1}B_i=\pi_i,\qquad i=0,1,\ldots,n-1
\tag{50}
$$

の 2 式が，式 (45) を総和記号を用いて整理すると,

$$
\pi_n\sum_{i=0}^{n-1}\lambda_i=\sum_{i=0}^{n-1}\mu_i\pi_i
\tag{51}
$$

が，また $\sum_{i=0}^{n}\pi_i=1$ であることから，式 (46) より,

$$
\pi_i B_i=\lambda_i,\quad i=0,1,\ldots,n-1
\tag{52}
$$

が得られる．

以上より，未知数について解いていくと式 (52) から $\pi_i$ について解けて，

$$\pi_i = \frac{\lambda_i}{B_i}, \qquad i = 0, 1, \ldots, n-1 \tag{53}$$

と表される．$\pi_i$ が明らかになると式 (51),(50) から $\pi_n$,$v_{i1}$ がそれぞれ明らかとなりその値は，

$$\pi_n = \frac{\sum_{i=0}^{n-1} \mu_i \pi_i}{\sum_{i=0}^{n-1} \lambda_i} = 1 - \sum_{i=0}^{n-1} \pi_i \tag{54}$$

$$v_{i1} = \frac{\pi_i}{B_i} = \frac{\lambda_i}{B_i^2}, \quad i = 0, 1, \ldots, n-1 \tag{55}$$

となる．$\pi_n$,$v_{i1}$ が明らかになると式 (48) から $v_{n0}$ が明らかとなり，

$$v_{n0} = \left( \pi_n + \sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{B_i^2} \right) \frac{1}{\sum_{i=0}^{n-1} \lambda_i} \tag{56}$$

と表される．ここで，求めたいモニタの時間平均 AoI は $\Delta$ は，

$$\Delta = \sum_{i=0}^{n} v_{i0} \tag{57}$$

と書けるため式 (49) は，

$$v_{i0} = \frac{\pi_i + \lambda_i \Delta}{B_i}, \qquad i = 0, \ldots, n-1 \tag{58}$$

となる．よって，式 (56), 式 (57) より，

$$\Delta = \sum_{i=0}^{n-1} v_{i0} + v_{n0}, \tag{59}$$

$$= \sum_{i=0}^{n-1} \frac{\pi_i}{B_i} + \Delta \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i} + \left( \pi_n + \sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{B_i^2} \right) \frac{1}{\sum_{i=0}^{n-1} \lambda_i} \tag{60}$$

$$\Delta \left( 1 - \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i} \right) = \sum_{i=0}^{n-1} \frac{\pi_i}{B_i} + \left( \pi_n + \sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{B_i^2} \right) \frac{1}{\sum_{i=0}^{n-1} \lambda_i} \tag{61}$$

$$\therefore \Delta = \frac{\sum_{i=0}^{n-1} \lambda_i \sum_{i=0}^{n-1} \frac{\pi_i}{B_i} + \pi_n + \sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{B_i^2}}{\sum_{i=0}^{n-1} \lambda_i \left( 1 - \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i} \right)} \tag{62}$$

となる．ここで定義 (40),(42) と式 (53),(54) より，

$$\Delta = \frac{\Lambda \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i^2} + 1 - \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i} + \sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{B_i^2}}{\Lambda \pi_n} \tag{63}$$

$$= \frac{\sum_{i=0}^{n-1} (\Lambda + \mu_i) \frac{\lambda_i}{B_i^2} + 1 - \sum_{i=0}^{n-1} \frac{\lambda_i}{B_i}}{\Lambda \pi_n} \tag{64}$$

$$= \frac{1}{\Lambda \pi_n} \tag{65}$$

と表される．よって，求めたいモニタの時間平均 AoI は式 (54) より，

$$\Delta = \frac{1}{\sum_{i=0}^{n-1} \frac{\mu_i \lambda_i}{\mu_i + \Lambda}} \tag{66}$$

と求まる．

# 5　シミュレーションによる検証

## 5.1　シミュレーション方法

数理的に求めた値と実験的に求めた値が一致することを確認するためにプログラムで図 6 の通信システムをシミュレートした．具体的には，サーバのサービス時間を指数分布に従わせ，サーバにシンボルが 10,000,000 個到着した時点で終了し時間平均 AoI を計算する．これを 10 回繰り返しその平均が出力されるようにした．

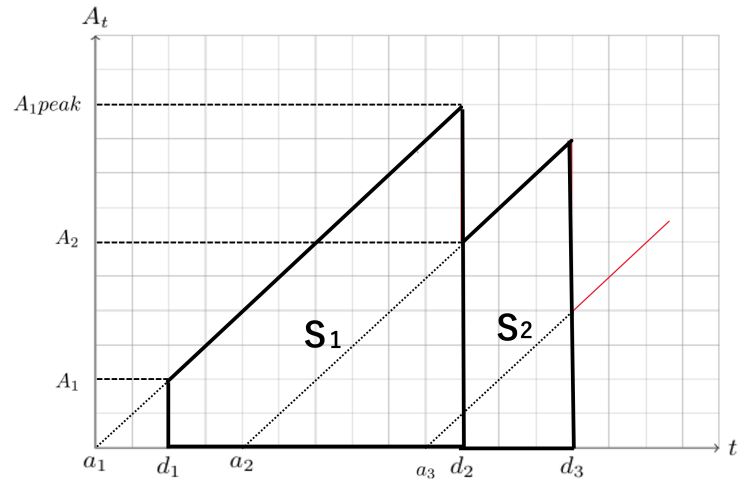


図 8: AoI の時間平均の導出

また，AoI の評価方法は，図 8 のようにモニタ AoI を台形のような面積で評価しそれを観測時間 $T$ で割る時間平均を使用した．この時，時間平均 AoI $A_{ave}$ は，観測時間 $T$ の間に発生したシンボル数を $n$ とすると

$$A_{\mathrm{ave}} = \frac{1}{T}\sum_{k=1}^{n} S_k \tag{67}$$

と書ける．

## 5.2 二値情報源の場合の再検証

シンボル‘0’ の発生確率 $p_0$ を 0.1 から 0.5 まで 0.01 刻みで変更した際の 2 種類の場合の AoI，本研究結果の解に $n$=2 を代入したときの AoI およびシミュレーション実験の AoI の 3 つの値をグラフ形式にし図 10 に示した．この時サービスレートは，$(\mu_0,\mu_1) = (0.2,0.1)$ と，シンボル‘1’の到着レートは $\lambda_1 + \lambda_0 = 1$ に設定しているため $\lambda_0$ が決定することでおのずと決定される．

シンボル‘0’ のサービスレート $\mu_0$ を 0.1 から 0.5 まで 0.01 刻みで変更した際の 2 種類の場合の AoI，本研究結果の解に $n$=2 を代入したときの AoI およびシミュレーション実験の AoI の値 3 つの値をグラフ形式にし図 11 に示した．シンボル‘1’ のサービスレート $\mu_1 = 0.1$ と，シンボル‘0’ ‘1’の到着レートは，$(\lambda_0, \lambda_1) = (0.5,0.5)$ で固定している．

2 つの図から，3 つのプロットはほぼ完全に一致しており，本研究結果が 2 種類のシンボル場合では正しいことが分かった．

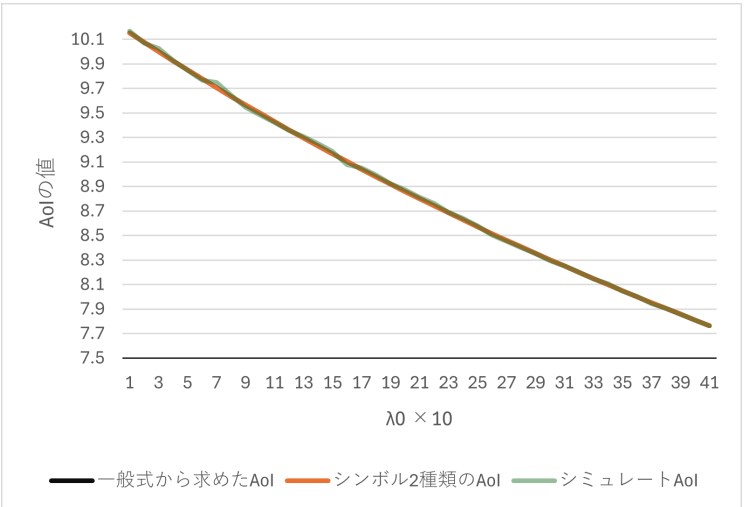


図 9: $\lambda_0$ を 0.1-0.5 で変更した場合

2 つのグラフが全体的に右肩下がりのグラフになっている理由として，サービスレートが関係していると考察できる．図 9 では，サービスレートを $(\mu_0,\mu_1) = (0.2,0.1)$ と固定している．シンボル‘0’の発生確率が増えると，サービスレートが大きい (サービスの効率が良い) シンボルが多く発生するようになる．その結果，より多くのシンボルがモニタに到着するため，時間平均 AoI が減少し，図 9 は右肩下がりのグラフとなる．

一方，図 10 においては，シンボル‘0’のサービスレート $\mu_0$ を徐々に大きくしてる．これに伴い，サービスの効率が向上し，より多くのシンボルがモニタに到着するため，時間平均 AoI が減少する．よって，図 10 も右肩下がりのグラフとなる．

## 5.3　多値情報源の場合の検証

本研究結果の正当性を，2 種類以上のシンボルの場合について検証した．各シンボルについて，本研究で求めた AoI，シミュレーション AoI およびその時のシンボル数 $n$ を表 7 にまとめた．この際のサービスレートおよび到着レートは $n$ シンボルの時，

$$\lambda_{n-i} = \mu_i = 0.1(n-i), \quad i = 1,\ldots,n-1 \tag{68}$$

と設定した．また，本研究で求めた AoI，シミュレーション AoI の比をシンボル数 $n$ ごとにまとめたグラフを図 11 として表示させた．図 11 の縦軸の範囲は，最小値 0.96 最大値 1.04 と 1 付近に注目させたグラフとなっている．

表 7，図 11 から，各シンボルにおいて 2 つの値はほぼ完全に一致しており本研究結果は正しい解であることが確認できた．

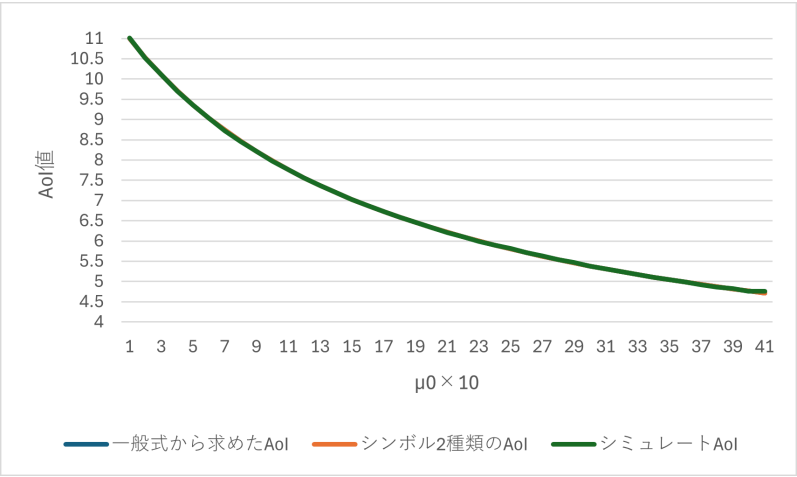


図 10: $\mu_0$ を 0.1-0.5 で変更した場合

表 7: 一般式の AoI とシミュレーション AoI のシンボル $n$ ごとの関係

| $n$ | 本研究結果から求めた AoI | シミュレーション AoI |
|---|---|---|
| 2 | 11.11111111 | 11.109021 |
| 3 | 7.924528302 | 7.923776 |
| 4 | 6.207751938 | 6.207967 |
| 5 | 5.11733835 | 5.114861 |
| 6 | 4.358537689 | 4.357284 |
| 7 | 3.798328639 | 3.79905 |
| 8 | 3.367058222 | 3.368862 |
| 9 | 3.024472697 | 3.02343 |
| 10 | 2.745597097 | 2.74583 |
| 100 | 0.297053002 | 0.297073 |
| 1000 | 0.029970054 | 0.030055 |

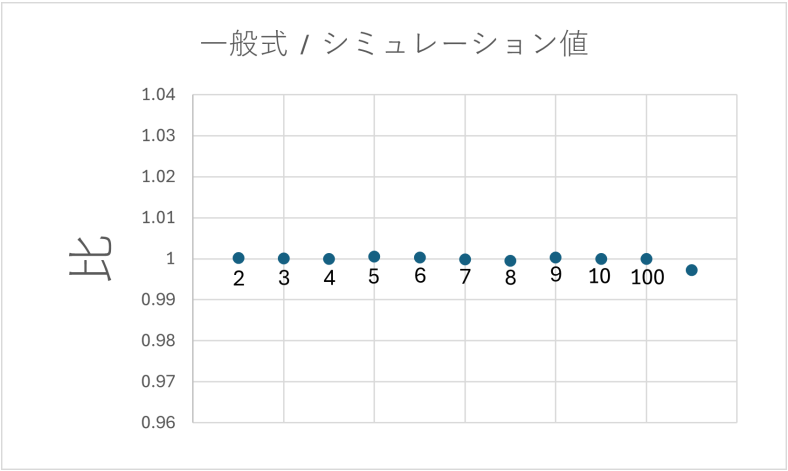


図 11: シミュレーションと本研究結果の比

## まとめ

本研究では AoI という視点から見たサーバで割り込みが発生する符号器を追加した通信システムにおいて，情報源が一般の $n$ 種類の場合のモニタ AoI を求める解を導出した．これにより，あらゆるシンボル数でどれほど情報の鮮度を維持できるか事前に知ることができるよう

になった．また，より現実に近い通信システムの再現ができた．

今後の課題としては，バッファ付きシステムの AoI を求めることが挙げられる．また，シンボルに優先度を設定し，処理の順序を制御するとともに，棄却するシンボルを選択できるようにすることも重要である．

## 謝辞

本研究に際して，様々なご指導をしていただいた指導教員の西新幹彦教授，また助言をしていただいた研究室の先輩方に深く感謝申し上げます．

## 参考文献

[1] 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, 2024 年 11 月閲覧．

[2] 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.

[3] 細海俊介，「情報鮮度の視点から見たバッファで割り込みのある待ち行列通信システムの実験的考察」，信州大学工学部電子情報システム工学科学士論文 (指導教員：西新幹彦) ，2022 年 2 月．

[4] 細海俊介，「情報鮮度の視点から見たサーバで割り込みのある待ち行列通信システムの解析」，信州大学大学院工学専攻電子情報システム工学分野修士論文 (指導教員：西新幹彦)，2024 年 3 月．

[5] R. D. Yates and S. K. Kaul, “The Age of Information: Real-Time Status Updating by Multiple Sources,” IEEE Trans. Inf. Theory, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.

# 付録 A　プログラム

サーバでのサービス時間が指数分布に従う通信モデルにて，多値情報源のモニタ AoI を計算するプログラムを以下に記載する．

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "MT.h"
#define Kaisu 10000000        // 各シミュレーションのイベント回数
#define SYMBOL_COUNT 3       // シンボル数の設定
#define SIMULATION_COUNT 20   // シミュレーション回数

double arrive_rates[SYMBOL_COUNT];  // シンボルごとの到着レート
double service_rates[SYMBOL_COUNT]; // シンボルごとのサービスレート

// 指数分布に従う乱数生成
double Random_Exp(double Lambda) {
    return -log(genrand_real3()) / Lambda;
}

// シンボル決定関数
int symboli() {
    double rnd = genrand_real1();
    double cumulative_probability = 0.0;
    double symbol_probability = 0.0;
    double total_arrive_rate = 0.0;

    // 到着レートの合計を計算
    for (int i = 0; i < SYMBOL_COUNT; i++) {
        total_arrive_rate += arrive_rates[i];
    }

    // 確率に従ってシンボルを選択
    for (int i = 0; i < SYMBOL_COUNT; i++) {
        symbol_probability += arrive_rates[i] / total_arrive_rate;
        if (rnd < symbol_probability) {
            return i;
        }
    }
    return SYMBOL_COUNT - 1;  // 念のため、最後のシンボルを返す
}

// サービスレートに基づきサービス時間を決定
double ui(int service) {
    return Random_Exp(service_rates[service]);
}

// 1 回分の AoI を計算
double simulate_AoI() {
    double current_time = 0.0, current_symbol_service_end_time = INFINITY;
    double server_symbol_arrival_time = 0.0, monitor_update_time = 0.0;
    double monitor_symbol_arrival_time = 0.0, cumulative_AoI = 0.0;
    double first_service_completion_time = 0.0;
    int current_symbol = -1, symbol_arrival_count = 0, service_completion_count = 0;

    double cumulative_probability = 0.0;

    // 到着レートの合計を計算
    for (int i = 0; i < SYMBOL_COUNT; i++) {
        cumulative_probability += arrive_rates[i];
    }
```

```
    // 初回の到着時刻を設定
    current_symbol = symboli();
    double next_symbol_arrival_time = Random_Exp(cumulative_probability);

    while (symbol_arrival_count < Kaisu && service_completion_count < Kaisu) {
        // 到着イベント
        if (next_symbol_arrival_time <= current_symbol_service_end_time) {
            current_time = next_symbol_arrival_time;
            server_symbol_arrival_time = next_symbol_arrival_time;
            current_symbol = symboli();
            next_symbol_arrival_time = current_time + Random_Exp(cumulative_probability);
            current_symbol_service_end_time = current_time + ui(current_symbol);
            symbol_arrival_count += 1;
        }
        // サービス完了イベント
        else if (next_symbol_arrival_time > current_symbol_service_end_time && service_completion_count == 0) {
            first_service_completion_time = current_symbol_service_end_time;
            current_time = current_symbol_service_end_time;
            monitor_update_time = current_symbol_service_end_time;
            monitor_symbol_arrival_time = server_symbol_arrival_time;
            current_symbol = -1;
            current_symbol_service_end_time = INFINITY;
            server_symbol_arrival_time = 0;
            service_completion_count += 1;
        }
        else if (next_symbol_arrival_time > current_symbol_service_end_time && service_completion_count > 0) {
            cumulative_AoI += 0.5 * ((current_symbol_service_end_time - monitor_symbol_arrival_time) *
            (current_symbol_service_end_time - monitor_symbol_arrival_time) -
                                    (monitor_update_time - monitor_symbol_arrival_time) *
                                    (monitor_update_time - monitor_symbol_arrival_time));
            current_time = current_symbol_service_end_time;
            monitor_update_time = current_symbol_service_end_time;
            monitor_symbol_arrival_time = server_symbol_arrival_time;
            current_symbol = -1;
            current_symbol_service_end_time = INFINITY;
            server_symbol_arrival_time = 0;
            service_completion_count += 1;
        }
    }

    return cumulative_AoI / (monitor_update_time - first_service_completion_time);
}

int main() {
    init_genrand((unsigned)time(NULL));

    // ユーザーによる到着レートとサービスレートの設定
    //printf("各シンボルの到着レートを入力してください:\n");
    for (int i = 0; i < SYMBOL_COUNT; i++) {
     arrive_rates[i]=i*0.1 + 0.1;
        printf("シンボル %d の到着レート=%f\n ", i,arrive_rates[i]);
        //scanf("%lf", &arrive_rates[i]);
    }

    //printf("各シンボルのサービスレートを入力してください:\n");
    for (int i = 0; i < SYMBOL_COUNT; i++) {
     service_rates[i]= arrive_rates[SYMBOL_COUNT-(i+1)];
       printf("シンボル %d のサービスレート=%f \n", i,service_rates[i]);
        //scanf("%lf", &service_rates[i]);
    }

    double total_AoI = 0.0;

    // シミュレーションを複数回実行
    for (int sim = 0; sim < SIMULATION_COUNT; sim++) {
        double AoI = simulate_AoI();
        total_AoI += AoI;
        //printf("シミュレーション %d の AoI = %f\n", sim + 1, AoI);
    }
```

```
    // 平均 AoI の計算と出力
    double average_AoI = total_AoI / SIMULATION_COUNT;
    printf("\n10 回分のシミュレーションの平均 AoI = %f\n", average_AoI);

    return 0;
}
```