信州大学工学部

学士論文

待ち行列システムにおける情報鮮度の時間平均の実験的検証

指導教員　西新 幹彦 准教授

学科　電気電子工学科
学籍番号　15T2063K
氏名　千葉 直紀

2019 年 2 月 21 日

目次

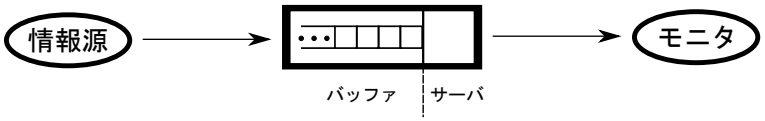

図 1　伝送システム

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，モニタが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カメラでサンプリングしてから時間が経過している．情報が情報源から出力されてからの経過時間をその情報の年齢という．本研究では，ある時点でモニタ表示されている情報の年齢をその時点の AoI(Age of Information) と呼ぶことにする．AoI は小さいことが望ましい．このように，AoI に着目した問題を情報鮮度の問題という．情報鮮度の考え方が提示される前は，情報を多く送受信できるかどうかが着目されていた．しかし，情報を送信しても伝送路で処理しきれず最新の情報を表示できないということも考えられる．情報鮮度の問題では，送信される情報の量ではなく経過時間という尺度で見ることで表示されている情報がどれだけ新しいかということを考察の対象にする．本研究では，ポアソン過程にしたがって情報が出力される情報源に対する AoI の時間平均を実験的に求めた．本論文は次のような章から構成される．2 章では本研究で取り扱う伝送モデル，情報鮮度の定義，情報鮮度の評価方法について述べる．3 章では情報鮮度の時間平均を測定する．また，待ち行列システムを用いて考察を行う．4 章に本研究のまとめと今後の課題を述べる．

2　準備

2.1　伝送モデル

本研究では，モニタに表示されている情報がサンプリングされてからの経過時間がサンプリングレートによりどのように変化するかに着目した．この経過時間は，サービスを待っている情報の数，すなわち伝送システムの混み具合に依存する．実際の伝送システムはネットワークなどの複雑な伝送システムとなっている．本研究では簡単化のためサーバは 1 つとして，情報源からサンプリングされた情報は，サンプリングされた順序でバッファに保持されており，順序通りにサーバで処理された情報がモニタに表示されているとする．このような伝送システムを図 1 に示す．図 1 の伝送システムで考える．情報は指数分布に従う時間間隔で情報源からサンプリングすると仮定しサンプリングレートを λ とする．サーバは指数分布に従う時間

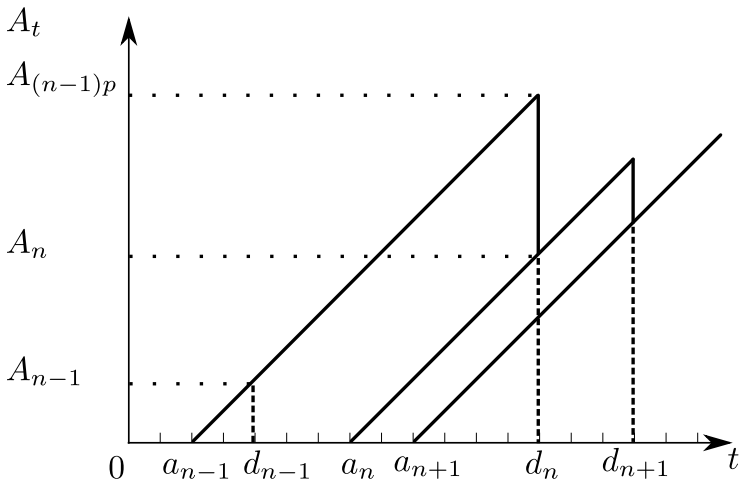

図 2　情報鮮度の遷移図

間隔でサービスを行うと仮定しサービスレートを μ 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．

2.2　情報鮮度の性質

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，

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

と書ける．次に n 番目にサンプリングされた情報 x_n が時刻 a_n，$n+1$ 番目にサンプリングされた情報 x_{n+1} が時刻 a_{n+1} にそれぞれサンプリングされた時刻順でバッファに保持される．情報 x_n と情報 x_{n+1} の AoI は独立に増加していく．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．モニタに表示されている情報の AoI の値が小さくなり，情報が新しくなる．情報がサンプリングされてモニタに表示，新しい情報に更新といった動作を繰り返し，図 2 のノコギリ状のグラフが得られる．

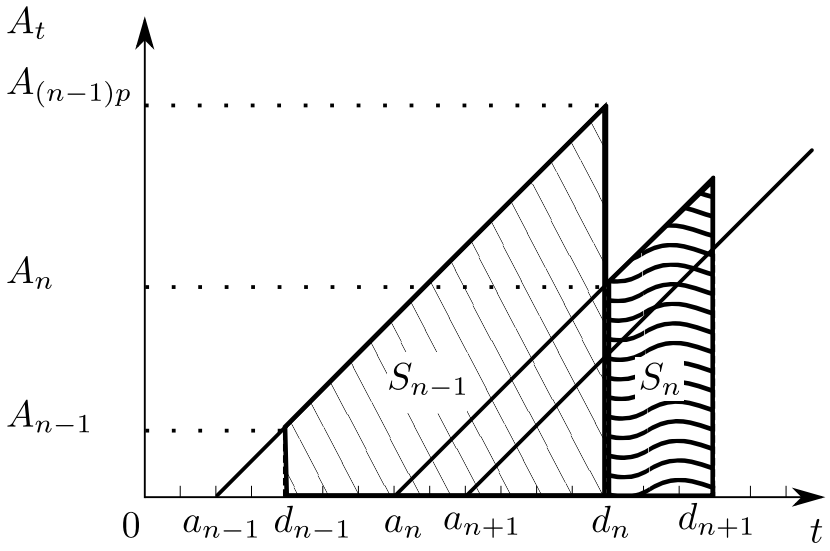

図 3　AoI の時間平均 導出

3　情報鮮度の時間平均の測定

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．

3.1　情報鮮度の時間平均の導出式

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，

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

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

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．

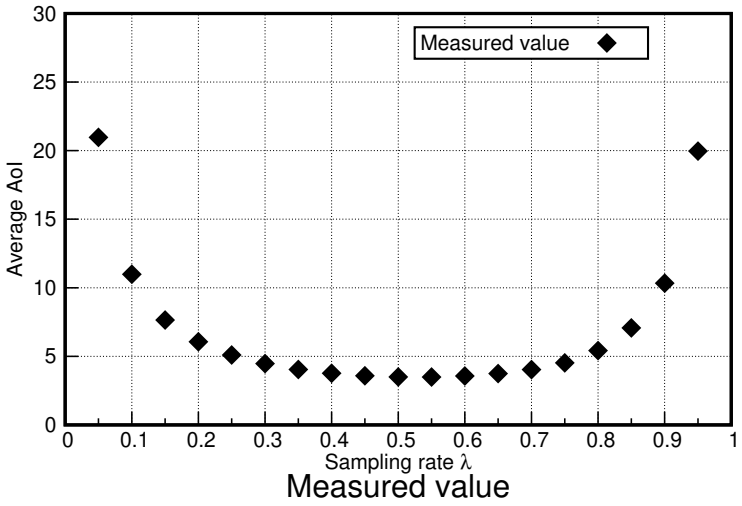

図 4　AoI の時間平均 測定結果

3.2　情報鮮度の測定方法

図 1 で示した伝送システムに基づいて情報源のサンプリングレートを変化させモニタに表示されている情報の AoI の時間平均を測定するシミュレーション実験を行った．測定は時刻 t が十分大きくなるように 100 万となるまで測定を行った．サンプリングはレート λ のポアソン到着として，レート λ を 0.05 から 1.0 まで変化させた．サービス時間はパラメータ μ の指数分布として，パラメータ $\mu = 1.0$ で固定した．観測時間は，最初の情報 x_1 がモニタに表示された時刻 d_1 から最後の情報 x_l が更新される時刻 d_{l+1} までとした．

3.3　測定結果

パラメータ λ を変化させたときの A_{ave} の測定結果を図 4 に示す．全体は，U 字型の下に凸な傾向がみられた．図 4 のような U 字型グラフとなった理由を考察する．ここで，サンプリング間隔の期待値 $E[G]$ は，

$$E[G] = \frac{1}{\lambda} \tag{6}$$

と書ける．また，モニタに表示された情報がサーバでの処理に掛かったサービス時間の期待値 $E[H]$ は，

$$E[H] = \frac{1}{\mu} \tag{7}$$

と書ける．まず，λ が小さい場合を考える．式 (6) からサンプリング間隔は長くなる．そのため情報の更新がなかなか起こらないためモニタに表示されている情報の AoI が大きくなると考えられる．次に，λ が大きい場合を考える．式 (6) からサンプリング間隔は短くなる．バッファに情報が到着するが，サーバの処理が追い付かない場合バッファでの待ち時間が長くなるためモニタに表示される情報の即時性が保てなくなり AoI が大きくなると考えられる．

3.4 考察

測定結果の図 4 となる理由を文献 [1][2][3] に基づいて考察していく．文献 [1] から時間平均 AoI の期待値 $E[A]$ は待ち行列システムにおける遅延時間から導出できることがわかっている．遅延時間 D_n を情報 x_n が時刻 a_n にサンプリングされてから時刻 d_n にモニタに表示されるまでに掛かった時間として，

$$D_n = d_n - a_n \tag{8}$$

と書ける．また，D_n は情報 x_n のバッファでの待ち時間 W_n と情報 x_n のサービス時間 H_n を用いて，

$$D_n = W_n + H_n \tag{9}$$

とも書ける．この遅延時間 D_n を待ち行列システムで考察する．

D_n は情報 x_n のサンプリング時刻 a_n で，直前の情報 x_{n-1} がすでにモニタに表示されているか，バッファで待機しているかで場合分けが必要になる．

まず，情報 x_{n-1} がすでにモニタに表示されている場合を考える．情報 x_{n-1} がモニタに表示される時刻 d_{n-1} が a_n よりも早いといえる．この時情報 x_{n-1} の遅延時間 D_{n-1} と情報 x_{n-1} と情報 x_n のサンプリング間隔 G_{n-1} の関係は，

$$\begin{aligned} &D_{n-1} \leq G_{n-1} \\ &D_{n-1} = d_{n-1} - a_{n-1} \\ &G_{n-1} = a_n - a_{n-1} \end{aligned} \tag{10}$$

と書ける．情報 x_n はサンプリングされてからすぐにサーバで処理される．よってバッファ内の待ち時間 W_n は，

$$W_n = 0 \tag{11}$$

となる．したがって D_n は，

$$D_n = H_n \tag{12}$$

と書ける．

次に，情報 x_{n-1} がバッファで待機している場合を考える．D_{n-1} と G_{n-1} の関係は，

$$D_{n-1} \geq G_{n-1} \tag{13}$$

と書ける．情報 x_n は情報 x_{n-1} のサービスが終わるまでバッファ内で待機しなければならない．よってバッファ内の待ち時間 W_n は

$$W_n = D_{n-1} - G_{n-1} \tag{14}$$

と書ける．したがって D_n は，

$$D_n = D_{n-1} - G_{n-1} + H_n \tag{15}$$

と書ける．式 (12)，式 (15) より D_n は，

$$D_{n+1} = \max(0, D_n - G_n) + H_{n+1} \tag{16}$$

という漸化式で表すことができる．G_nと H_n を独立な確率変数と仮定すると，D_n は式 (16) で定められる確率変数となる．文献 [3] から，一般に D_n と G_n は独立ではない．（G_n が長いと D_n は短くなる傾向がある．）そこで，D_nG_n の期待値 $E[D_nG_n]$ を求める．$E[D_nG_n]$ は，

$$\begin{aligned} E[D_nG_n] &= E[(W_n + H_n)G_n] \\ &= E[W_nG_n] + E[H_n]E[G_n] \end{aligned} \tag{17}$$

と書ける．システムの混み具合 ρ は，

$$\begin{aligned} \rho &= \frac{E[H]}{E[G]} \\ &= \frac{\lambda}{\mu} \end{aligned} \tag{18}$$

と書ける．文献 [2] より $E[W_nG_n]$ は，

$$\begin{aligned} E[W_nG_n] &= \int_0^\infty sE[W_n|G_n = s]f_{G_n}(s)ds \\ &= \frac{\rho}{\mu^2(1-\rho)} \end{aligned} \tag{19}$$

と書ける．したがって $E[D_nG_n]$ は，

$$E[D_nG_n] = \frac{\rho}{\mu^2(1-\rho)} + \frac{1}{\mu\lambda} \tag{20}$$

と書ける．

次に AoI の時間平均の期待値 $E[A]$ は式 (4)，式 (5)，式 (16) を用いて，

$$E[A] = \lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{l} S_n \tag{21}$$

$$\begin{aligned} S_n &= \frac{1}{2}\left\{(d_{n+1}-a_n)^2 - (d_n - a_n)^2\right\} \\ &= \frac{1}{2}\left[\left\{(d_{n+1}-a_{n+1}) + (a_{n+1}-a_n)\right\}^2 - (d_n-a_n)^2\right] \\ &= \frac{1}{2}\left\{(D_{n+1}+G_n)^2 - (D_n)^2\right\} \\ &= \frac{G_n^2}{2} + D_n G_n \end{aligned} \tag{22}$$

と書ける．したがって，文献 [3] から適当な正則条件 (定常性，エルゴード性) の下，$E[A]$ は，

$$E[A] = \frac{\frac{E[G^2]}{2} + E[D_n G_n]}{E[G]} \tag{23}$$

と書ける．ただし $E[D_n G_n]$ は n によらないことに注意されたい．式 (20) と $E[G^2] = \frac{2}{\lambda^2}$ を用いて $E[A]$ は，

$$\begin{aligned} E[A] &= \frac{1}{E[G]}\left(\frac{E[G^2]}{2} + E[D_n G_n]\right) \\ &= \lambda\left(\frac{1}{\lambda^2} + \frac{\rho}{\mu^2(1-\rho)} + \frac{1}{\mu\lambda}\right) \\ &= \frac{\lambda}{\mu^2}\left(\frac{\mu^2}{\lambda^2} + \frac{\rho}{1-\rho} + \frac{1}{\rho}\right) \\ &= \frac{\rho}{\mu}\left(\frac{1}{\rho} + \frac{1}{\rho^2} + \frac{\rho}{1-\rho}\right) \\ &= \left(1 + \frac{1}{\rho} + \frac{\rho^2}{1-\rho}\right) E[H] \end{aligned} \tag{24}$$

と書ける．式 (24) から，$E[A]$ はシステムの混み具合 ρ に依存する関数となることがわかる．

測定結果と期待値 $E[A]$ の比較を図 5 に示す．図 5 より測定結果と期待値が一致していることがわかる．

4　まとめと今後の課題

本研究では，待ち行列システムを用いて情報鮮度の時間平均のシミュレーションを行った．結果はパラメータ λ により変化することが分かった．サンプリング間隔と遅延時間に着目して考察を行いトレードオフの関係となることが実験的に確認できた．

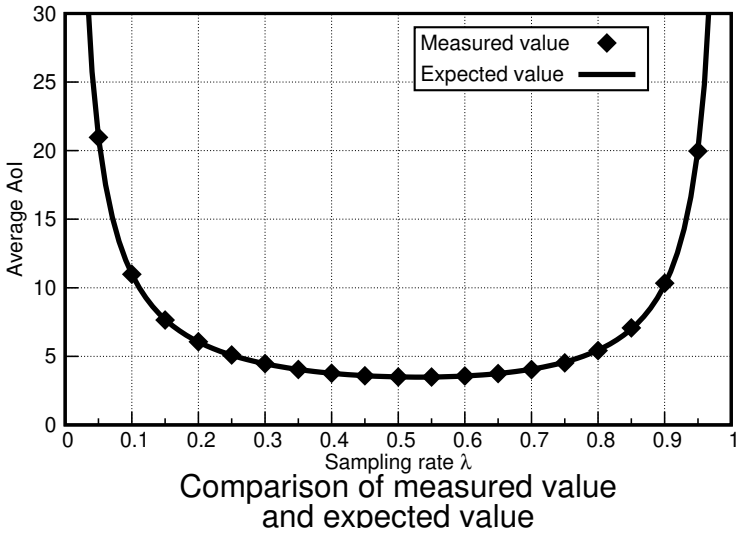

図 5　測定結果と期待値の比較

今後の課題として，情報鮮度の時間平均を小さくするためバッファに制限を設けた場合について検証していきたいと考えている．具体的には，バッファを 1 つにして新しい情報がサンプリングされたら古い情報は破棄する方法や，バッファを無くしてバッファでの待ち時間を 0 にする方法が文献 [1] で提案されている．これらをシミュレーションしていきたい．

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,” arXiv:1804.06139v1, Apr. 2018.

[2] S. Kaul, R. Yates, M. Gruteser, “Real-Time Status: How Often Should One Update?,” in Proc. of IEEE INFOCOM 2012, pp. 2731–2735, Mar. 2012.

[3] 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, 2019 年 1 月閲覧．

付録 A　ソースコード

A.1　情報鮮度の時間平均を測定するプログラム

```
#define _CRT_SECURE_NO_WARNINGS

/************************************************************
    krnd.c -- Knuth の乱数発生法
************************************************************/
#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(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();
}

/************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define QUEUE_SIZE 10000
```

```
#define lambda_e 1.0

int main(void) {
    FILE *fp, *fp2;

    if ((fp = fopen("output3.txt", "w")) == NULL) {
        printf("[%d]open error\n", __LINE__);
        exit(1);
    }
    if ((fp2 = fopen("area_12.txt", "w")) == NULL) {
        printf("[%d]open error\n", __LINE__);
        exit(1);
    }

    fprintf(fp, "パラメータλ v\t 平均値\t 台形の個数\n");

    init_rnd(12345);

    for (int i = 0; i < 20; i++){
        int k = 0, m = 0, c = 0;
        int count = 0; /* サービス終了の数 */
        double t = 0.0; /* 現在時刻 */
        double tv; /* 次の到着時刻 */
        double te = 0.0; /* 次のサービス終了時刻 */
        double Xv, Xe;
        double area = 0.0, d;
        double start, last; /* AoI 計測の開始と最終時刻 */
        double last_dep;

        double lambda_v = (i + 1) / 20.0; /* 到着レート */

        double queue[QUEUE_SIZE];
        int head = 1;
        queue[0] = 0;

        fprintf(fp, "%lf\t", lambda_v);
        printf("%f\t 現在時刻で実行中の動作\n", lambda_v);

        Xv = -1.0 / lambda_v * log(1.0 - rnd()); /* 最初の到着間隔 */
        tv = 0.0 + Xv; /* 最初の到着時刻 */

        while (t < 1.0e+6) {
            if (c >= 1 && tv >= te) { /* サービス終了処理 */
                t = te;
                c = c - 1;
                if (count < 1) {
                    start = t;
                }
                else {
                    fprintf(fp2, "%lf,%lf,%lf\n", queue[(head - 1) % QUEUE_SIZE], te, te - queue[(head - 1) % QUEUE_SIZE])
                    d = last_dep - queue[(head - 1) % QUEUE_SIZE];
                    area += 1.0 / 2.0*(pow(te - (queue[(head - 1) % QUEUE_SIZE]), 2) - pow(d, 2));
                    last = t;
                }
                count++;
                last_dep = t;
                fprintf(fp2, "%lf,%lf,%lf\n", queue[head], te, te - queue[head]);

                head = (head + 1) % QUEUE_SIZE;
                if (c >= 1) {
                    Xe = -1.0 / lambda_e * log(1.0 - rnd());
                    te = t + Xe; /* 次のサービス終了時刻 */
                }
            } else { /* 到着処理 */
                t = tv;
                if (c == QUEUE_SIZE - 1) {
                    printf("[%d] full!\n", __LINE__);
                    exit(1);
                }
```

```
                queue[(head + c) % QUEUE_SIZE] = t;
                c = c + 1;
                Xv = -1.0 / lambda_v * log(1.0 - rnd());
                tv = t + Xv; /* 次の到着時刻 */
                if (c == 1) {
                    Xe = -1.0 / lambda_e * log(1.0 - rnd());
                    te = t + Xe; /* 次のサービス終了時刻 */
                }
            }
        }
        fprintf(fp2, "\t\t\t%lf", area);
        fprintf(fp, "%lf\t%d\t%f\t%f\n", area / (last - start), count, start, last);
    }

    return(0);
}
```