

# 電子情報工学専門実験 C1

## 「通信ネットワーク工学の基礎実験」

【実施場所】

オンライン

【担当教員】

井上 文彰 (yoshiaki@comm.eng.osaka-u.ac.jp)

樽谷 優弥 (tarutani@comm.eng.osaka-u.ac.jp)

各課題を実施する様子の動画ならびに各実験課題の実施結果は CLE にアップロードされている。それらを使用してレポートを作成すること。

### 1 目的

パケットキャプチャログの分析を通じて、トランスポート層・アプリケーション層における代表的な通信プロトコルである User Datagram Protocol (UDP), Transmission Control Protocol (TCP)，ならびに Hypertext Transfer Protocol (HTTP) の基本動作を理解する。加えて、Web アプリケーションにおける輻輳現象の観察を通じて、通信トラヒック工学の基礎を学習する。

### 2 第一週：パケットキャプチャによる通信ログ分析

第一週はパケットキャプチャを用いて、HTTP メッセージの送受信、ならびにそれに用いられるトランスポート層プロトコル TCP の動作を確認する。また、トランスポート層プロトコル UDP が、Domain Name System (DNS) により URL を IP アドレスに変換する際に使用されることを確認する。本実験ではトランスポート層ならびにアプリケーション層における通信に対する分析のみを必須課題とし、これらより下位の層（リンク層およびネットワーク層）における通信の分析は任意とする。

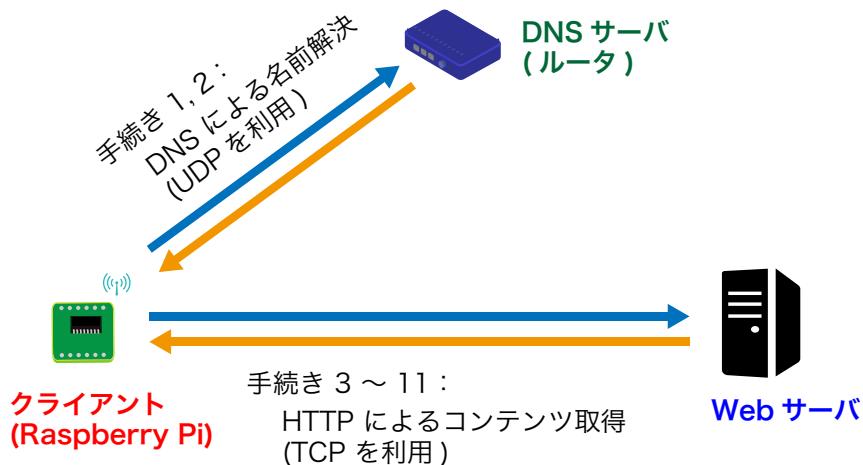


図 1: 第一週で用いるシステムの模式図

## 2.1 実験環境

ハードウェア:

- Raspberry Pi 3 Model B+

ソフトウェア:

- Raspberry Pi OS 5.10 (オペレーティング・システム)
- Wireshark 2.6.20 (パケットキャプチャソフトウェア)
- Curl 7.64.0 (HTTP をサポートするコマンドラインツール)
- Mozilla Firefox 78.9.0esr (Web ブラウザ)

## 2.2 実験内容・手順

「Web ブラウザに URL を入力してから、Web ページが表示されるまで」に行われる一連の通信をパケットキャプチャによって記録し、その内容を分析する。以下では、手元で操作する端末 (Raspberry Pi) を、単に**クライアント**と呼ぶ。

### 2.2.1 Curl を用いた Web ページ閲覧

実際の Web ブラウザの動作は多少複雑であるため、最初はコマンドラインツール「Curl」を使用して Web ページを読み込むことにする。

**【実験課題 1】** クライアントでターミナルを起動し、以下のコマンドを実行せよ。

```
$ curl http://www2b.comm.eng.osaka-u.ac.jp/~yoshiaki/C1/page1.html
```

(左端の \$ はターミナルであらかじめ表示されている記号のため入力不要)

上記のコマンドを実行すると、ターミナルの標準出力に当該 Web ページのソースコード、すなわち HTML ファイルの中身が表示される。実行結果は CLE に **kadai1.txt** としてアップロードされている。これを実験課題 1 の結果としてレポートに記載すること。

このとき、次のような一連の通信が行われている（図 1 も参照のこと）。以下では、角括弧 [ ] を [IP アドレス] または [IP アドレス: ポート番号] という形式で、IP アドレスや、IP アドレスとポート番号の組を示すのに用いる。

※以下に示す IP アドレスは一例であり、実際の実験結果と異なる可能性があることに注意。

### ドメイン名を IP アドレスに変換

1. **クライアントは**、入力された URL のコンテンツを保持する Web サーバの IP アドレスを DNS サーバ [10.144.0.1:53] に問い合わせる（**名前解決**）。この通信には UDP が使用される。
2. **DNS サーバ [10.144.0.1:53] は**、1 の問い合わせ結果をクライアントに返す。この通信も、1 と同様に UDP が使用される。

### TCP コネクションの確立

3. **クライアントは**、2 で通知された IP アドレス [192.168.13.80] と、HTTP におけるサーバ側ポート番号 80 の組 [192.168.13.80:80] を宛先として、TCP コネクション開始要求 (SYN) を送信する。
4. **Web サーバ [192.168.13.80:80] は**、3 で送信された SYN を受信し、それに対する確認応答 (ACK) に加え、コネクション開始要求 (SYN) をクライアントに送信する。
5. **クライアントは**、4 で送信された SYN/ACK を受信し、それに対する ACK を Web サーバ [192.168.13.80:80] に送信する。これによりクライアントと Web サーバの間に TCP コネクションが確立される。

### HTTP メッセージの送受信

6. **クライアントは**、5 で確立された TCP コネクションを通じて、Web サーバに「HTTP GET メッセージ」を格納したパケットを送信する。
7. **Web サーバは**、6 で送信されたパケットを受信し、ACK をクライアントに返す。また、このパケットに含まれる「HTTP GET メッセージ」を読み込み、「HTTP Response メッセージ」を生成する。

セージ」をクライアントに返す。この「HTTP Response メッセージ」には、クライアントが要求した Web ページのソースコード (HTML ファイル) が含まれている。

8. **クライアントは**, 7 で送信された「HTTP Response メッセージ」を受信し, ACK を返す。

## TCP コネクションの終了

9. **クライアントは**, Web サーバにコネクション終了通知 (FIN) を送信する。
10. **Web サーバは**, 9 で送信された FIN を受信し, それに対する ACK に加え, コネクション終了通知 (FIN) をクライアントに送信する。
11. **クライアントは**, 10 で送信された FIN を受信し, それに対する ACK を Web サーバに送信する。

**【実験課題 2】**以下の手順で、上記の一連の通信に関するログを取得せよ。

- (i) クライアントで Wireshark を起動し、パケットキャプチャを開始。
- (ii) 実験課題 1 と同じコマンドを実行。
- (iii) Web ページのソースコードが標準出力に表示されたら、パケットキャプチャを停止。

さらに、上記の 1 から 11 に相当するパケットを特定し、それらの詳細を分析せよ。具体的には、以下の各項目に関する情報をまとめよ。

#### UDP パケット

- 送信元 IP アドレス、受信先 IP アドレス (ネットワーク層)
- 送信元ポート番号、受信先ポート番号 (トランスポート層)
- UDP セグメント長 (トランスポート層)
- アプリケーション層メッセージの内容

#### TCP パケット

- 送信元 IP アドレス、受信先 IP アドレス (ネットワーク層)
- 送信元ポート番号、受信先ポート番号 (トランスポート層)
- TCP セグメント長 (トランスポート層)
- TCP フラグ (トランスポート層)。何が ON で何が OFF になっているか。
- シーケンス番号と ACK 番号 (トランスポート層)。
- アプリケーション層メッセージの内容 (アプリケーション層)

パケットキャプチャログは `kadai2.txt` ならびに `kadai2_detail.txt` として CLE にアップロードされている。

## 2.2.2 Firefox を用いた Web ページ閲覧

次に、普通の Web ブラウザ (Firefox) から Web ページを閲覧する際の通信を観察する。

**注: Firefox にキャッシュが残っていると、通信が見られないことがある。その場合は一度 Firefox の履歴を削除すること。**

**【実験課題 3】** 実験課題 2 と同様の手順で、Firefox から

<http://www2b.comm.eng.osaka-u.ac.jp/~yoshiaki/C1/page1.html>

にアクセスしたときの通信ログを分析せよ。また、実験課題 2 の場合との相違点を検証せよ。

**パケットキャプチャログは kadai3.txt ならびに kadai3\_detail.txt として CLE にアップロードされている。**

page1.html はテキストのみが書かれたページであり、単独の HTML ファイルだけで完結している。一方、大半の Web ページでは、HTML ファイル内から別の画像ファイルなどを読み込んで表示するように作られている。この場合、Web ブラウザは最初に取得した HTML ファイルに続けて、さらに HTTP GET メッセージを Web サーバに送信することで、HTML ファイル内で参照されている画像ファイルなどを取得する。

**【実験課題 4】** 実験課題 3 と同様の手順で、Firefox から

<http://www2b.comm.eng.osaka-u.ac.jp/~yoshiaki/C1/page2.html>

にアクセスしたときの通信ログを分析せよ。具体的には、2.2.1 節で述べた 1 から 11 の通信の流れを参考に、この場合の通信の流れを記述せよ。さらに、実験課題 2 と同様に、各パケットに関する情報をまとめよ。

**パケットキャプチャログは kadai4.txt ならびに kadai4\_detail.txt として CLE にアップロードされている。**

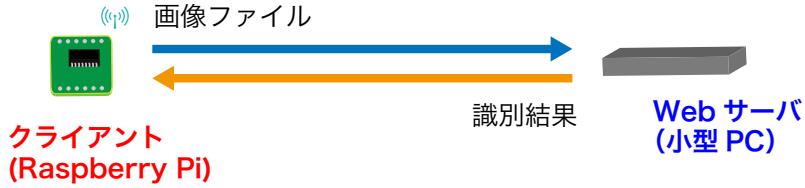


図 2: 第二週で用いるシステムの模式図

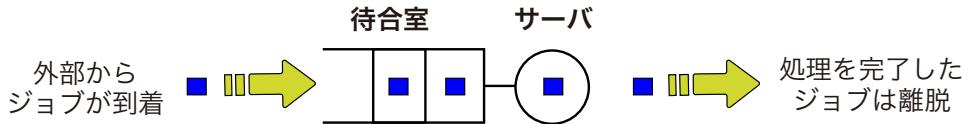


図 3: 単一サーバ待ち行列モデル

### 3 第二週: Web アプリケーションにおける輻輳現象の分析

第二週は、画像識別を行う Web アプリケーションを題材として、情報通信システムにおいて生じる輻輳現象を観察する。実験対象とするシステムの概略を図 2 に示す。クライアント (Raspberry Pi) と Web サーバ (小型 PC) は LAN ケーブルとスイッチングハブを介して接続しており、Web サーバ上では画像識別を行う Web アプリケーションが動作している。クライアントから Web サーバへ HTTP POST メッセージとして画像ファイルを送信すると、Web サーバは「ResNet V2 101」と呼ばれる深層ニューラルネットワークを使用して画像のラベルを識別する。さらに、識別結果は HTTP Response メッセージとしてクライアントに返される。

このシステムでは、クライアントが画像を送信可能である最小間隔と比べて、Web サーバが画像識別に要する時間の方がはるかに大きいという特徴がある。したがって、クライアントが最大の送信頻度で画像を送り続けると Web サーバの処理能力を超過してしまい、遅延時間が極めて増大するという結果を招く。本実験では、このような状況において、送信頻度が遅延時間に与える影響、ならびに遅延時間との適切な兼ね合いを取ることのできる適切な送信頻度の決定方法について、実測に基づいた考察を行う。

#### 3.1 理論

本実験のシステムを **待ち行列モデル** を用いてモデル化する。待ち行列モデルは、さまざまな資源競合問題を数学的に表現するモデルであり、特に、情報通信システムにおける混雑(輻輳)の発生を数学的に解析するのによく用いられる。本実験では、**単一サーバ待ち行列モデル**を考える(図 3)。单一サーバ待ち行列は、**待合室**と単一の**サーバ**で構成され、外部から次々に到着する**ジョブ**をサーバが順次処理していく状況を表現する。以降では特に断りの無い限り、サーバによるジョブの処理順序は先着順であるとする。

单一サーバ待ち行列モデルは、具体的には以下の手続きに従って動作する。

- (a) ある頻度で、外部からジョブが待合室に到着する。

- (a-1) ジョブの到着時点においてサーバが空いていた場合,  
到着ジョブは即座にサーバで処理を受け始める.
- (a-2) ジョブの到着時点において他のジョブが処理中であった場合,  
到着ジョブは待合室で待機する.

- (b) ジョブの処理が完了した時点において,
  - (b-1) 待合室にジョブが存在する場合, サーバは次のジョブの処理を開始する.
  - (b-2) 待合室にジョブが存在しない場合, サーバは次のジョブの到着まで待機する.

上記の手続きから明らかのように, 本実験においては, 待ち行列モデルにおける「待合室」および「サーバ」は, Web サーバ上に存在する「バッファ」および「演算装置 (CPU)」に相当する(サーバという言葉が紛らわしいため注意を要する). また, 「外部」とは Web サーバの外側のことを表し, 実際には到着ジョブはクライアントが生成してサーバに送出したものである.

時刻 0 に最初のジョブが到着し, その後,  $N - 1$  個のジョブが到着するものとする.  $n$  番目 ( $n = 0, 1, \dots, N - 1$ ) のジョブの到着時刻を  $\alpha_n$  ( $\alpha_{n-1} < \alpha_n$ ) とする. ただし, 便宜上, 時刻 0 に到着したジョブを「0 番目」のジョブと呼ぶことにし, したがって  $\alpha_0 = 0$  とする.  $G_n$  を  $n - 1$  番目と  $n$  番目のジョブの到着間隔と定義する.

$$G_n = \alpha_n - \alpha_{n-1}, \quad n = 1, 2, \dots, N - 1$$

次に,  $n$  ( $n = 0, 1, \dots$ ) 番目のジョブがサービスを完了する時刻を  $\beta_n$  とし, そのジョブの遅延時間  $D_n$  と定義する.

$$D_n = \beta_n - \alpha_n, \quad n = 0, 1, \dots, N - 1$$

すなわち, 遅延時間とは, ジョブが到着してからサービスを完了するまでにかかる時間を表す. さらに, 遅延時間は二種類の要素の和として表現することができる.

$$D_n = W_n + H_n, \quad n = 0, 1, \dots, N - 1 \tag{1}$$

上式の  $W_n$  は待ち時間,  $H_n$  は処理時間を表す. すなわち,  $W_n$  は他のジョブのサービスが完了するまで待合室で待機する時間の長さを表し,  $H_n$  は自身が実際にサーバで処理される時間の長さを表す.

**定理 1.** 先着順処理の单一サーバ待ち行列では, 次の漸化式 (**Lindley 方程式**) が成立する.

$$D_n = \max(0, D_{n-1} - G_n) + H_n, \quad n = 1, 2, \dots, N - 1 \tag{2}$$

**【演習課題 1】** 定理 1 を証明せよ.

ヒント: 式 (1) と式 (2) を比較するところから始めると良い.

式 (2) を少し変形すると、次式が得られる。

$$D_n = \max(G_n, D_{n-1}) + H_n - G_n, \quad n = 1, 2, \dots, N-1$$

この式は次のように解釈できる。まず、 $\max(G_n, D_{n-1}) = G_n$  すなわち  $G_n \geq D_{n-1}$  となるとき、 $n$  番目のジョブは待ち時間無しに処理が開始するため次式が成り立つ。

$$D_n = H_n$$

一方、 $G_n < D_{n-1}$  となるとき、 $n$  番目のジョブは前のジョブの処理完了を待つ必要があり、

$$D_n = D_{n-1} + (H_n - G_n)$$

である。すなわち、このとき、一つ前のジョブの遅延時間  $D_{n-1}$  と比べて、遅延時間  $D_n$  は  $H_n - G_n$  だけ加算される。

したがって、ジョブの処理時間  $H_1, H_2, \dots$  と比べて到着間隔  $G_1, G_2, \dots$  が小さい状況が継続した場合、**遅延時間  $D_n$  は限無く増加し続ける**。以下ではさらに、この事実を別の観点から捉えることとする。 $A(t)$  ( $t \geq 0$ ) を、時刻  $t$  までに到着したジョブの総数と定義する。

$$A(t) = \sum_{n=0}^{N-1} \mathbb{1}\{\alpha_n \leq t\}$$

ただし、 $\mathbb{1}\{\mathcal{A}\}$  は、 $\mathcal{A}$  が真であるとき 1、偽であるとき 0 を取る関数（指示関数）を表す。同様に、 $D(t)$  ( $t \geq 0$ ) を時刻  $t$  までにサービスを完了した離脱したジョブの総数と定義する。

$$D(t) = \sum_{n=0}^{N-1} \mathbb{1}\{\beta_n \leq t\}$$

時刻  $t$  において待合室またはサーバに滞在しているジョブの総数  $L(t)$  は次式で与えられる。

$$L(t) = A(t) - D(t), \quad t \geq 0 \tag{3}$$

$T$  ( $0 \leq T \leq \beta_N$ ) を、最後のジョブが離脱するまでのある一時刻とする。時間  $(0, T]$  における**平均到着率**  $\bar{\lambda}(T)$  ならびに**平均離脱率**  $\bar{\mu}(T)$  を次式で定義する。

$$\bar{\lambda}(T) = \frac{A(T) - 1}{T}, \quad \bar{\mu}(T) = \frac{D(T)}{T},$$

すなわち、 $\bar{\lambda}(T)$  は単位時間当たりの到着数、 $\bar{\mu}(T)$  は単位時間当たりの離脱数を表す。

以下のように、平均到着率  $\bar{\lambda}(T)$  は到着間隔  $G_n$  ( $n = 1, 2, \dots, N-1$ ) の平均と関係付けられる。無用な複雑さを避けるため、以下の議論では  $T \geq \alpha_1$  を仮定する。まず、 $\bar{\tau}(T)$  を時間  $(0, T]$  に発生した到着の平均間隔とする。

$$\bar{\tau}(T) = \frac{1}{A(T) - 1} \sum_{n=1}^{A(T)-1} G_n$$

**補題 1.**  $A(T) \leq N - 2$  のとき, 次の不等式が成立する.

$$\sum_{n=1}^{A(T)-1} G_n \leq T < \sum_{n=1}^{A(T)-1} G_n + G_{A(T)}$$

**定理 2.**  $A(T) \leq N - 2$  のとき, 次の不等式が成立する.

$$\bar{\tau}(T) \leq \frac{1}{\bar{\lambda}(T)} < \bar{\tau}(T) + \frac{G_{A(T)}}{A(T) - 1} \quad (4)$$

**【演習課題 2】** 補題 1 および定理 2 を証明せよ.

$A(T)$  が十分大きいとき, 式 (4) の最右辺における第二項は無視できるほど小さいとみなして良い. 実際, 分子の  $G_{A(T)}$  は到着間隔一つ分の長さであり, 非常に特異な場合を除けば  $A(T)$  にほとんど依存しない. したがって, 式 (4) から次の関係を得る.

$$\bar{\lambda}(T) \simeq \frac{1}{\bar{\tau}(T)} \quad (5)$$

すなわち, 平均到着率は平均到着間隔の逆数に (ほぼ) 等しい.

一方, 平均離脱率は到着間隔  $G_n$  と処理時間  $H_n$  の両方に依存する. 厳密な解析は煩雑であるため, ここでは直観的な議論のみを行う. 到着ジョブの負荷に対してサーバの処理能力が十分高いとき, すなわち, 処理時間が到着間隔に比べて小さいとき, 時刻  $T$  での滞留ジョブ数  $L(T)$  は 0 に近い値を取る. このとき, 式 (3) より  $A(T) \simeq D(T)$  となるため,

$$\bar{\mu}(T) \simeq \bar{\lambda}(T), \quad \text{サーバの処理能力} > \text{到着ジョブの負荷} \text{ のとき}$$

が成り立つ. 一方, 到着ジョブの負荷に対してサーバの処理能力が不十分であるとき, すなわち, 処理時間が到着間隔に比べて大きいとき, 時刻  $t \in [0, T]$  において滞留ジョブ数  $L(t)$  は 1 以上の値を取り, サーバは常にジョブを処理している状態となる. このとき, ジョブの離脱間隔は処理時間  $H_n$  と等しくなるため, 式 (5) と同様にして

$$\bar{\mu}(T) \simeq \frac{1}{\bar{B}(T)}, \quad \text{サーバの処理能力} < \text{到着ジョブの負荷} \text{ のとき}$$

となる. ただし,  $\bar{B}(T)$  は時間  $[0, T]$  における平均処理時間を表す ( $D(T) \geq 1$  を仮定する).

$$\bar{B}(T) = \frac{1}{D(T)} \sum_{n=0}^{D(T)-1} H_n$$

以上より、平均離脱率は次のように特徴付けられる。

- サーバの処理能力が到着負荷と比べて**高い**  
⇒ 平均離脱率は平均到着率に(ほぼ)等しい
- サーバの処理能力が到着負荷と比べて**低い**  
⇒ 平均離脱率は平均処理時間の逆数に(ほぼ)等しい。

式(3)および平均到着率と平均離脱率の定義より、時刻  $T$  における滞留ジョブ数は

$$L(T) = (\bar{\lambda}(T) - \bar{\mu}(T)) \times T$$

で与えられる。したがって、平均離脱率に関する上記の議論から、次の結論を得る。

- サーバの処理能力が到着負荷と比べて**高い**  
⇒ 滞留ジョブ数は比較的小さな値にとどまる
- サーバの処理能力が到着負荷と比べて**低い**  
⇒ 滞留ジョブ数は時間経過とともに際限無く増加する。

特に後者の結論は、Lindley 方程式を用いた前述の議論とも整合する。

なお、サーバの処理能力が到着負荷と比べて高い場合であっても、滞留ジョブ数は無視できるほど小さいとは限らない。これは主に、到着間隔や処理時間のばらつきの影響によるものである。**通信トラヒック理論(待ち行列理論)**では、確率論を用いて遅延時間や滞留ジョブ数をより緻密に定式化し、その統計的性質を数学的に議論する。通信トラヒック理論の詳細は、3年次秋冬学期配当科目「通信ネットワーク工学」において学習する。

## 3.2 実験環境

クライアント側ハードウェア:

- Raspberry Pi 3 Model B+

クライアント側ソフトウェア:

- Ubuntu 20.04 LTS
- Python 3.8.2

サーバ側ハードウェア:

- HP ProDesk 400 G4 DM/CT

## サーバ側ソフトウェア:

- Ubuntu 20.04 LTS
- TensorFlow 2.3.1
- TFServer 0.3, ResNet V2 101
- Python 3.8.2

### 3.3 実験内容・手順

本実験でも一週目と同様にクライアント (Raspberry Pi) のみを操作する。まず、ターミナルを開き、ホームディレクトリ直下の client ディレクトリに移動する。

```
$ cd /home/ubuntu/client
```

#### 【実験課題 5】

client.py を実行し、その結果を確認せよ。

```
$ python3 client.py
```

サーバから返された識別結果と画像ファイルの内容を見比べよ。なお、このプログラムでは、どの画像をサーバに送るかは実行時ごとに変化する。

**実行結果は CLE に `kadai5.txt` としてアップロードされている。**

**【実験課題 6】** 第一週の実験課題 2 と同様に、Wireshark でパケットキャプチャを開始した状態で実験課題 5 のコマンドを実行し、クライアントとサーバの間でどのような通信が行われているかを分析せよ。

**パケットキャプチャログは `kadai6.txt` ならびに `kadai6_detail.txt` として CLE にアップロードされている。**

client.py には複数のコマンドライン引数を渡すことができる。コマンドライン引数で指定できるオプションの一覧は次のコマンドで確認できる。

```
$ python3 client.py --help
```

特に、以下の実験では次のオプションを使用する。

- `--num_images N`: サーバへ送信する総画像枚数を  $N$  に指定
- `--interval T`: サーバへの画像の送出間隔を  $T$  に指定
- `--show_statistics`: 遅延時間などの測定値を表示

### 【実験課題 7】送出間隔 $T$ を様々に変化させながら

```
$ python3 client.py --num_images 100 --interval T --show_statistics
```

を実行することで、以下の四つの課題を実施せよ。

- (a) 横軸に画像のインデックス、縦軸に遅延時間を取りグラフを、複数の  $T$  の値に関して重ねてプロットせよ。  $T$  の最大値を 1.0 [秒]、最小値を 0.05 [秒] とし、その中間の値も複数選んで測定すること。
- (b) 横軸に画像送出頻度  $\lambda := 1/T$ 、縦軸に平均遅延時間を取りグラフをプロットせよ。  $T$  の値は、遅延時間が小さい場合から大きい場合まで、まんべんなくグラフにプロットできるように選択せよ。
- (c) 3.1 節の理論を参考に、サーバの最大処理能力 [枚/秒] を推定せよ。
- (d) 平均遅延時間と送出間隔  $T$  の和が最小となるような画像送出頻度を特定せよ。

実験結果は CLE に (送出間隔)\_kadai7.txt という名前でアップロードされている。

## 4 レポートについて

提出期限 (2026年1月22日 23:59) までに担当教員宛 (yoshiaki@comm.eng.osaka-u.ac.jp) にメールで提出すること。期限を過ぎたレポートは、原則として受け付けない。やむを得ない事情がある場合は、事前に連絡すること。

レポートは、この実験を受講していない人が読んでも内容が分かるように書かれてていなければならない。最低限、以下の内容を含めること。

- 実験の目的と理論、および演習課題の解答
- 実験内容および手順
- 実験結果
- 実験結果に対する考察
- 感想

なお、読みやすく書かれてさえいれば、具体的な記述順序や構成方法は問わない。