応用プログラミング

ex3

本日のレシピ

1. HTTPプロトコル
2. httpServer.py（簡易HTTPサーバ）
3. ファイル処理（前半の復習）
4. エラー処理（try ～ except）

プロトコル

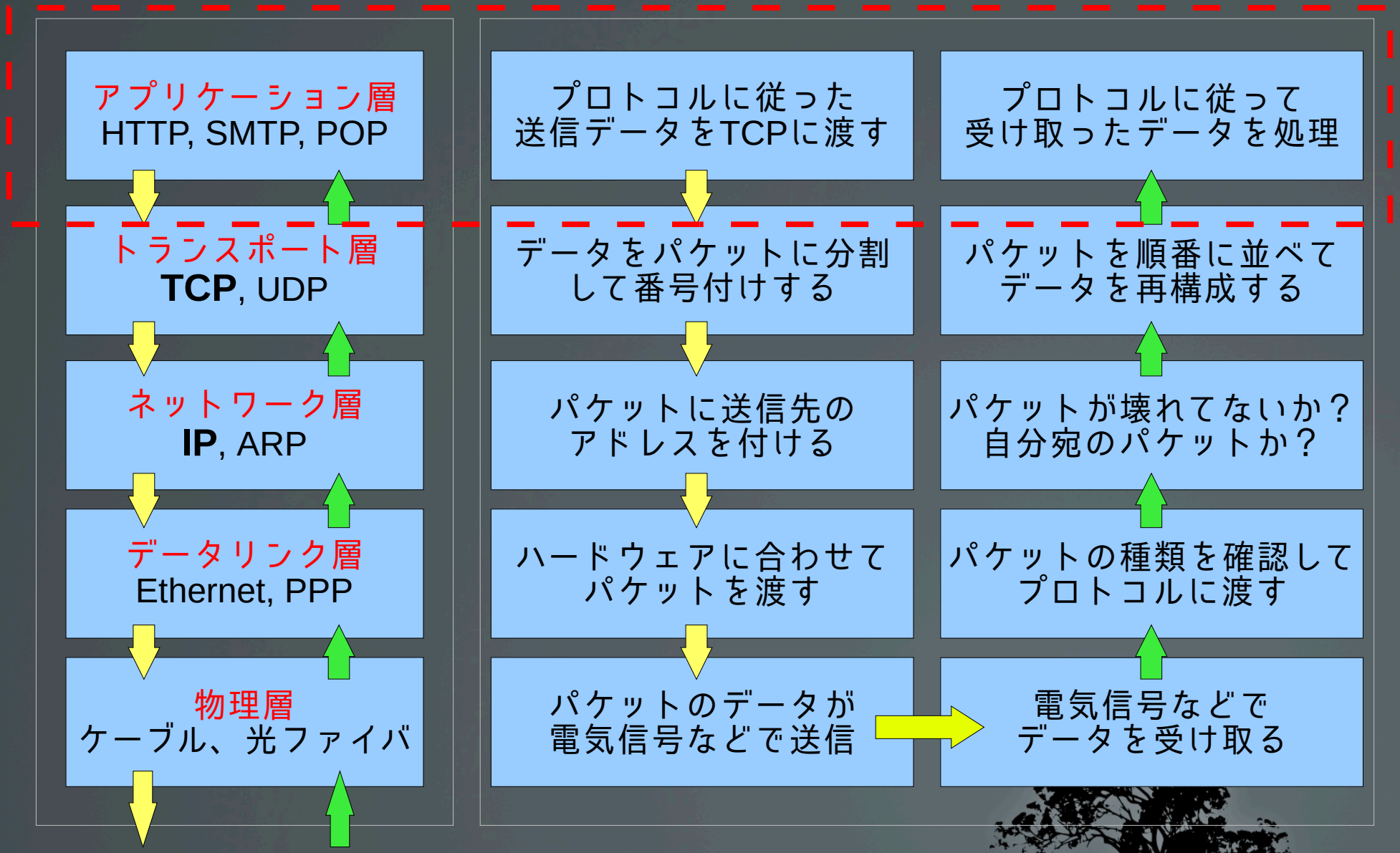
- プロトコル = 約束事、（通信）規約
  - 情報伝達における曖昧さを無くすための手続き方法
  - 情報伝達や手続きの方法・通信手順など
  - 使われる言葉や記号の意味、それの伴う動作など
  - TCP/IP プロトコル、京都議定書(*Kyoto Protocol*)

- この応用プログラミング（後半戦）では、

  TCP/IP プロトコルを用いたネットワークプログラミング

  の HTTPプロトコル（簡易版）を扱う。

プロトコル階層


アプリケーション層
HTTP, SMTP, POP
トランスポート層
TCP, UDP
ネットワーク層
IP, ARP
データリンク層
Ethernet, PPP
物理層
ケーブル、光ファイバ
プロトコルに従った
送信データをTCPに渡す
データをパケットに分割
して番号付けする
パケットに送信先の
アドレスを付ける
ハードウェアに合わせて
パケットを渡す
パケットのデータが
電気信号などで送信
プロトコルに従って
受け取ったデータを処理
パケットを順番に並べて
データを再構成する
パケットが壊れてないか？
自分宛のパケットか？
パケットの種類を確認して
プロトコルに渡す
電気信号などで
データを受け取る

HTTP プロトコル

- HyperText Transfer Protocol
  - HTTP/0.9 (1991)
  - HTTP/1.0 RFC1945(1996)
  - HTTP/1.1 RFC2616(1999)
  - HTTP/2 RFC7540(2015)
  - HTTP/3 (HTTP-over-QUIC) (2019?)
- 80番ポート
- URL/URI (Uniform Resource Locator/Identifier)

  http://www.math.ryukoku.ac.jp/index.html

  <プロトコル>://<ホスト名>/<リソース名>

HTTPリクエスト（クライアント → サーバ）

HTTPリクエストの例：

```
GET / HTTP/1.1
If-Modified-Since: Thu, 29 Nov 2018 05:01:38 GMT
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.1
Safari/605.1.15
Accept-Language: ja-jp
DNT: 1
Accept-Encoding: gzip, deflate
Host: 10.10.112.161:5000
Cache-Control: max-age=259200
Connection: keep-alive
<<空行>>
<<HTTPメッセージ>>
      :
```

1. HTTPリクエスト：リクエストの種類、対象リソース、HTTPプロトコルのバージョン

   メソッド␣リソース␣HTTPバージョン

2. HTTPヘッダ：必要に応じてサーバに送付する情報
3. HTTPメッセージ：1つの空行を挟んでメッセージ本体（GET メソッドでは使わない）

HTTPレスポンス（クライアント ← サーバ）

HTTPレスポンスの例：

```
HTTP/1.1 302 Found
Date: Fri, 30 Nov 2018 01:48:44 GMT
Server: Apache
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Location: https://www.ryukoku.ac.jp/
Content-Length: 210
Content-Type: text/html; charset=iso-8859-1
<<空行>>
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="https://www.ryukoku.ac.jp/">here</a>.</p>
</body></html>
```

1. HTTPステータス：リクエスト処理の結果

   HTTPバージョン␣ステータスコード␣ステータスの説明

2. HTTPヘッダ：必要に応じてクライアントに送付する情報
3. HTTPメッセージ：1つの空行を挟んでメッセージ本体

本日のレシピ

1. HTTPプロトコル
2. httpServer.py（簡易HTTPサーバ）
3. ファイル処理（前半の復習）
4. エラー処理（try ～ except）

httpServer.py

```
with conn:
    print("Connected by,", addr)
    data = b""
    while True:                                  # 受信：空行 b"\r\n\r\n" があれば受信を終了する
        data += conn.recv(1024)
        if b"\r\n\r\n" in data:
            break
    print("Received:", data.decode(), end="")  # 改行しない

    rData = b"HTTP/1.1 "
    cType = b"Content-Type: text/html; charset=utf-8\r\n\r\n"
    htHead = b"<html><body>"
    htTail = b"</html></body>"

    if data.startswith(b"GET "):                # 受信データが GET メソッド(b"GET "で始まっている)なら
        try:
            with open(file) as f:               # file を開いて HTTPレスポンスと file の中身を
                rData += b"200 OK\r\n" + cType
                rData += f.read().encode() + b"\r\n"
        except OSError:
            rData += b"404 Not Found\r\n" + cType
            rData += htHead + file.encode() + b" is not found" + htTail

    conn.sendall(rData)                          # 送信
    print("Sent:", rData.decode())
```

本日のレシピ

1. HTTPプロトコル
2. httpServer.py（簡易HTTPサーバ）
3. ファイル処理（前半の復習）
4. エラー処理（try ～ except）

ファイル処理とwith文

```
if data.startswith(b"GET "):                # 受信データが GET メソッド(b"GET "で始まっている)なら
   try:
      with open(file) as f:                 # file を開いて HTTPレスポンスと file の中身を
         rData += b"200 OK\r\n" + cType
         rData += f.read().encode() + b"\r\n"
   except OSError:
      rData += b"404 Not Found\r\n" + cType
      rData += htHead + file.encode() + b" is not found" + htTail
```

ファイルの入出力

1. ファイルを開く: open()
2. ファイルの読み書き：read(), write(), readline() など
3. ファイルを閉じる：close()

with 文：

- open(file) が成功したら返り値（ファイルオブジェクト）を f としてブロックを実行
- ブロックが終了したら自動的に close()
- 失敗したら例外（エラー）を発生させてブロックは実行しない

本日のレシピ

1. HTTPプロトコル
2. httpServer.py（簡易HTTPサーバ）
3. ファイル処理（前半の復習）
4. エラー処理（try ～ except）

エラー処理（try ～ except）

```
if data.startswith(b"GET "):                    # 受信データが GET メソッド(b"GET "で始まっている)なら
   try:
      with open(file) as f:                     # file を開いて HTTPレスポンスと file の中身を
         rData += b"200 OK\r\n" + cType
         rData += f.read().encode() + b"\r\n"
   except OSError:
      rData += b"404 Not Found\r\n" + cType
      rData += htHead + file.encode() + b" is not found" + htTail
```

with 文：

- open(file) が成功したら返り値（ファイルオブジェクト）を f としてブロックを実行
- ブロックが終了したら自動的に close()
- 失敗したら例外（エラー）を発生させてブロックは実行しない

try ～ except 文による例外処理：

- try: ブロックを実行
- try:ブロックで例外が発生したら、except: ブロックを実行してエラー処理
- except 例外クラス名: で、発生した例外によってエラー処理を複数記述できる
- その他、例外が発生しなかった場合の処理(else:)、例外が発生してもしなくても必ず最後に実行する後処理(finally:)など、詳しくはテキストのchap.10 例外処理(p.335～)

httpServer.py を起動してWebブラウザと通信