Kurs rozszerzony języka Python

Wykład 7.

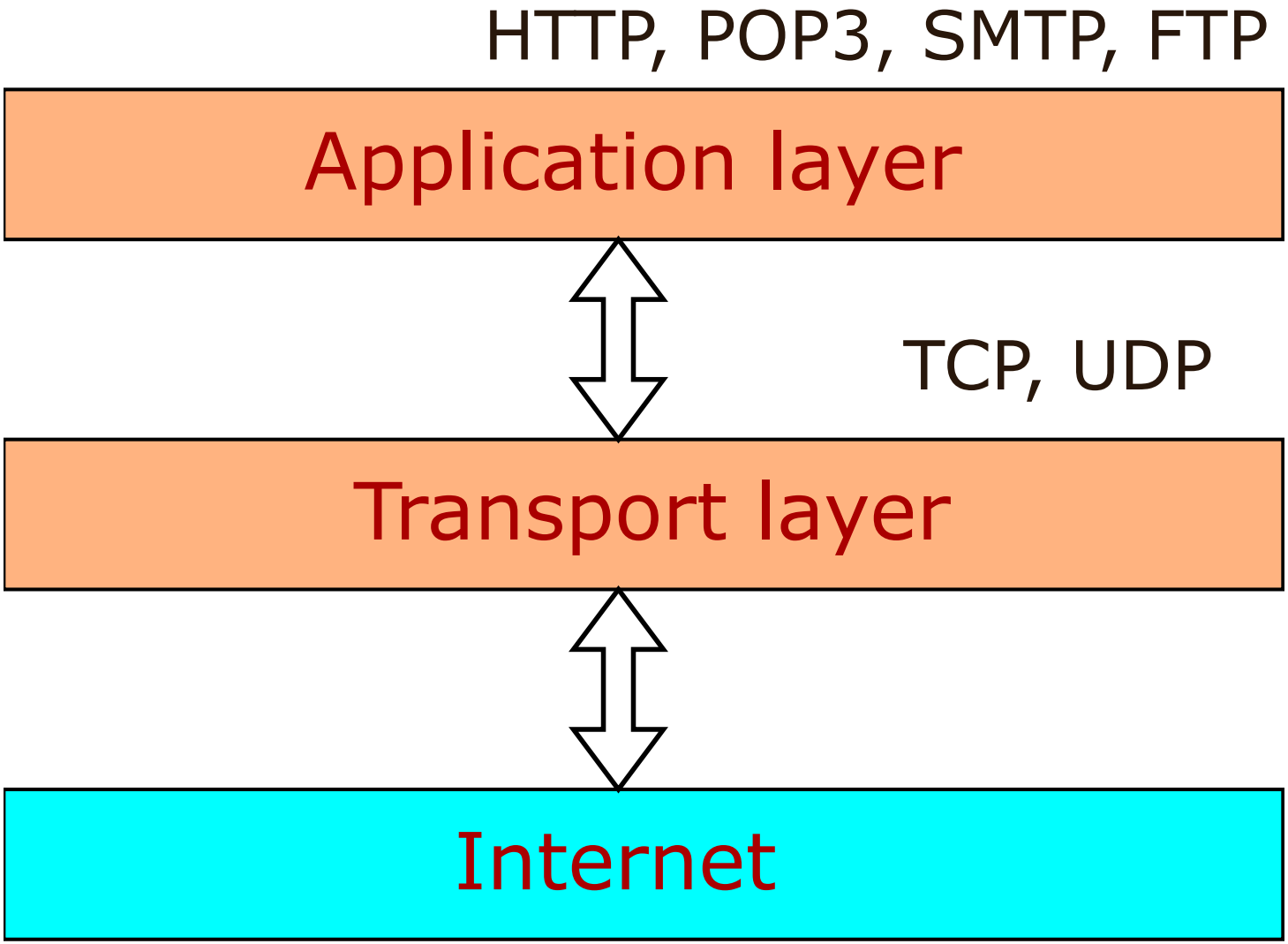
Marcin Młotkowski

30 listopada 2022

Plan wykładu

Plan wykładu

Schemat sieci I



Obsługa gniazd

```
import socket

port = 8081
host = "localhost"
s = socket.socket(socket.AF_INET,
                  socket.SOCK_DGRAM)
s.sendto("Hello, Python!!!", (host, port))
```

Dla porównania wersja w C

```c
#define _XOPEN_SOURCE 700

#include <assert.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <arpa/inet.h>
#include <netdb.h> /* getprotobyname */
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>

int main(int argc, char **argv) {
    char buffer[BUFSIZ];
    char protoname[] = "tcp";
    struct protoent *protoent;
    char *server_hostname = "127.0.0.1";
    char *user_input = NULL;
    in_addr_t in_addr;
    in_addr_t server_addr;
    int sockfd;
    size_t getline_buffer = 0;
    ssize_t nbytes_read, i, user_input_len;
    struct hostent *hostent;
    /* This is the struct used by INet addresses. */
    struct sockaddr_in sockaddr_in;
    unsigned short server_port = 12345;

    if (argc > 1) {
        server_hostname = argv[1];
        if (argc > 2) {
            server_port = strtol(argv[2], NULL, 10);
        }
    }
```

Dla porównania wersja w C

```c
/* Get socket. */
protoent = getprotobyname(protoname);
if (protoent == NULL) {
    perror("getprotobyname");
    exit(EXIT_FAILURE);
}
sockfd = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
if (sockfd == -1) {
    perror("socket");
    exit(EXIT_FAILURE);
}

/* Prepare sockaddr_in. */
hostent = gethostbyname(server_hostname);
if (hostent == NULL) {
    fprintf(stderr, "error: gethostbyname(\"%s\")\n", server_hostname);
    exit(EXIT_FAILURE);
}
in_addr = inet_addr(inet_ntoa(*(struct in_addr*)*(hostent->h_addr_list)));
if (in_addr == (in_addr_t)-1) {
    fprintf(stderr, "error: inet_addr(\"%s\")\n", *(hostent->h_addr_list));
    exit(EXIT_FAILURE);
}
sockaddr_in.sin_addr.s_addr = in_addr;
sockaddr_in.sin_family = AF_INET;
sockaddr_in.sin_port = htons(server_port);

/* Do the actual connection. */
if (connect(sockfd, (struct sockaddr*)&sockaddr_in, sizeof(sockaddr_in)) == -1) {
    perror("connect");
    return EXIT_FAILURE;
}
while (1) {
    fprintf(stderr, "enter string (empty to quit):\n");
    user_input_len = getline(&user_input, &getline_buffer, stdin);
    if (user_input_len == -1) {
```

Protokoły sieciowe

- Web Services (SOAP/WSDL)
- XML RPC
- JSON RPC
- REST
- ...

Publiczne serwisy

- Google
- Amazon
- Allegro
- GUS
- Geodezja
- NASA
- Zapisy
- ...

Plan wykładu

Logowanie i sesja

Logowanie do Systemu Zapisy i pobranie wiadomości.

Protokół http/https, żądanie GET

```
GET / HTTP/1.1
Host: zapisy.ii.uni.wroc.pl
User-Agent: Mozilla/5.0
```

Protokół http, odpowiedź serwera

```
HTTP/1.1 200 OK
Date: Mon, 21 Nov 2022 09:14:01 GMT
Server: Apache/2.0.54 (Debian GNU/Linux)
Content-Length: 37402
dane
```

Protokół http/https, żądanie POST

```
POST /login.php HTTP/1.1
Host: zapisy.ii.uni.wroc.pl
User-Agent: Mozilla/5.0

username=ja&password=hasło
```

Podstawowe narzędzie

```
import requests
```

Podstawowe narzędzie

```
import requests
```

Z dokumentacji

Requests: HTTP dla ludzi

Sesja

Za obsługę sesji (ciasteczek etc) odpowiada obiekt `requests.Session()`.

Kod

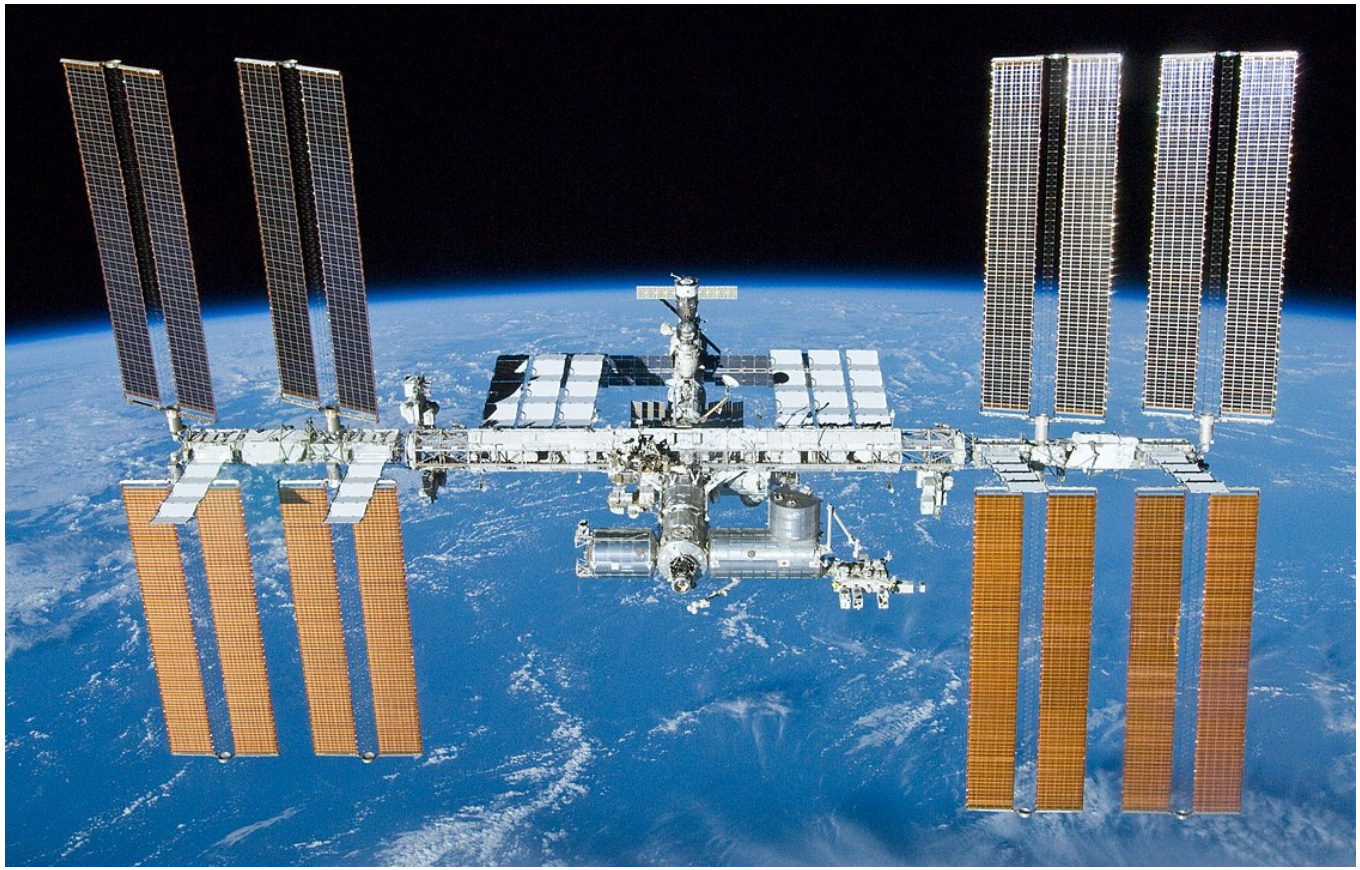
```
import requests
import private

url = "https://zapisy.ii.uni.wroc.pl/"

cred = {"username": "usrname", "password": "kdfjaskd"}
with requests.Session() as s:
    s.get(url)
    s.post(url + "users/login", data=cred)
    odp = s.get(url + "news/")
    print(odp.text)
```

Plan wykładu

International Space Station



Źródło: Wikipedia

Zapytanie i odpowiedź

```
GET /iss-now.json
```

Zapytanie i odpowiedź

```
GET /iss-now.json
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Gdzie jest ISS, kto tam jest

```
import requests
res = requests.get("http://api.open-notify.org/astros.json")
print(res.json())
res = requests.get("http://api.open-notify.org/iss-now.json")
print(res.json())
```

Jaka będzie pogoda

- `http://api.openweathermap.org`
- uzyskanie własnego API key

Postać żądania

```
"http://api.openweathermap.org/data/2.5/forecast?q=W
↪  roclaw&units=metrics&mode=json&APPID=..."

url = "http://api.openweathermap.org/data/2.5/forecast"
params = {"q": "Wroclaw", "mode": "json",
          "units": "metric", "APPID": private.KEY}
res = requests.get(url, params=params)
with open(prognoza.json, 'w') as fh:
    fh.write(json.dumps(res.json()))
```

Plan wykładu

Przykład: TERYT

Oficjalny rejest jednostek terytorialnych: województw, powiatów, gmin, miast, dzielnic, ulic etc. Administratorem danych jest Główny Urząd Statystyczny udostępniający dane m.in. przez API.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Dobra

Ale są biblioteki, np. zeep.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Dobra

Ale są biblioteki, np. zeep.

```
pip3 install zeep
```

Czemu nie lubię SOAP :(

```
<client>
    <endpoint address="https://uslugaterytws1.stat.gov.pl/T
        binding="customBinding" bindingConfiguration="custo
        contract="ServiceReference1.ITerytWs1" name="custom
</client>
<bindings>
  <customBinding>
    <binding name="custom">
    <security defaultAlgorithmSuite="Default"
        authenticationMode="UserNameOverTransport"
        requireDerivedKeys="true" includeTimestamp="true"
        messageSecurityVersion="WSSecurity11WSTrustFebruary
    <localClientSettings detectReplays="false" />
    <localServiceSettings detectReplays="false" />
    </security>
```

Kod

```
from zeep import Client
from zeep.wsse.username import UsernameToken

CREDENTIALS = {
    'wsdl': 'https://uslugaterytws1test.stat.gov.pl/wsdl/te
    'username': 'TestPubliczny',
    'password': '1234abcd'
}
token = UsernameToken(
    username=CREDENTIALS['username'],
    password=CREDENTIALS['password']
)
```

Pobranie danych o Wrocławiu

```
client = Client(wsdl=CREDENTIALS['wsdl'], wsse=token)

print(client.service.CzyZalogowany())

for jpt in client.service.WyszukajJPT(nazwa='Wrocław'):
    print(jpt)
```

Plan wykładu

Przykład: Allegro

- Link

Klient P2P

```
import sys, os, SimpleXMLRPCServer, xmlrpclib, re, hmac # (C) 2004, E.W. Felten
ar,pw,res = (sys.argv,lambda u:hmac.new(sys.argv[1],u).hexdigest(),re.search)
pxy,xs = (xmlrpclib.ServerProxy,SimpleXMLRPCServer.SimpleXMLRPCServer)
def ls(p=""):return filter(lambda n:(p=="")or res(p,n),os.listdir(os.getcwd()))
if ar[2]!="client": # license: http://creativecommons.org/licenses/by-nc-sa/2.0
  myU,prs,srv = ("http://"+ar[3]+":"+ar[4], ar[5:],lambda x:x.serve_forever())
  def pr(x=[]): return ([(y in prs) or prs.append(y) for y in x] or 1) and prs
  def c(n): return ((lambda f: (f.read(), f.close()))(file(n)))[0]
  f=lambda p,n,a:(p==pw(myU))and(((n==0)and pr(a))or((n==1)and [ls(a)])or c(a))
  def aug(u): return ((u==myU) and pr()) or pr(pxy(u).f(pw(u),0,pr([myU])))
  pr() and [aug(s) for s in aug(pr()[0])]
  (lambda sv:sv.register_function(f,"f") or srv(sv))(xs((ar[3],int(ar[4]))))
for url in pxy(ar[3]).f(pw(ar[3]),0,[]):
  for fn in filter(lambda n:not n in ls(), (pxy(url).f(pw(url),1,ar[4]))[0]):
    (lambda fi:fi.write(pxy(url).f(pw(url),2,fn)) or fi.close())(file(fn,"wc"))
```