Kurs rozszerzony języka Python

Usługi sieciowe

Marcin Młotkowski

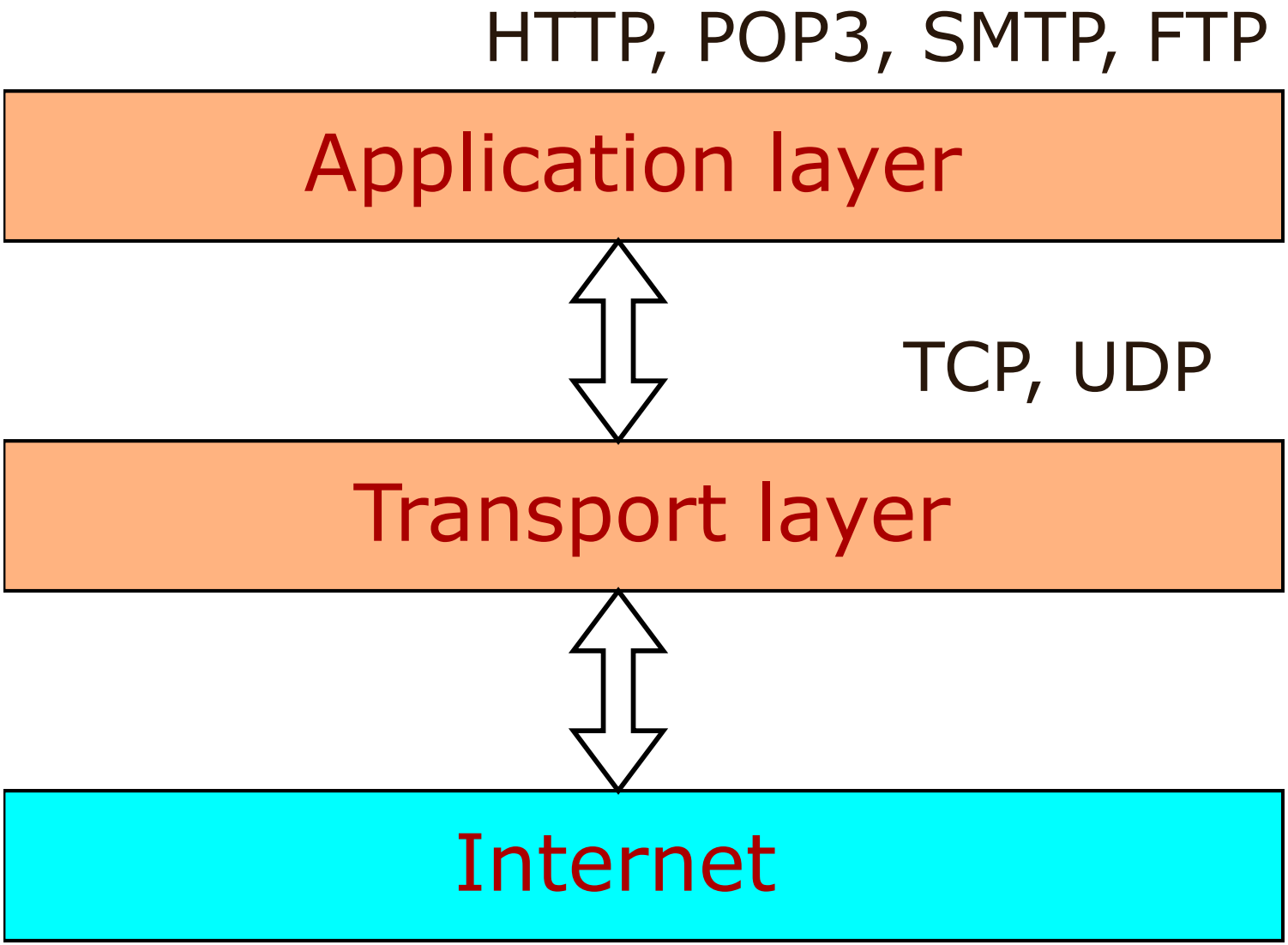
15 grudnia 2017

Plan wykładu

1. Usługi sieciowe
   - Warstwa transportowa
   - Warstwa aplikacji
   - Serwery aplikacyjne
   - Wielowątkowy serwer XML RPC
2. Zope
   - Wersje Zope
   - Cechy Zope

Plan wykładu

Schemat sieci



Protokół UDP

Cechy protokołu

- Protokół jest bardzo prosty
- Brak kontroli dostarczonych komunikatów
- Stosowany tylko w lokalnych (niezawodnych) sieciach
- Komunikacja za pomocą gniazd

Przykład

Zadanie

Przesłać z komputera (nadawca/klient) do komputera (odbiorca/serwer) komunikat "Hello Python".

Klient

Implementacja

```
import socket
port = 8081

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

Odbiorca

Implementacja

```
import socket
port = 8081

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('', port))
print ('Nasłuch na porcie', port)
while True:
    data, addr = s.recvfrom(1024)
    print ('Nadawca', addr, 'dane', data)
```

Gniazda

Inne funkcje modułu `socket`:

- gethostname() — nazwa komputera
- gethostbyname(hostname) — IP hosta
- gethostbyaddr(ip_address)
- Obsługa SSL

'Ambitne' zadanie

Zdalne monitorowanie pracy wielu komputerów za pomocą przeglądarki:

- Na każdym komputerze jest uruchomiony serwer http;
- żądanie jakiejś strony powoduje wykonanie odpowiedniej akcji, np.:

  http://host/uptime

  powoduje wykonanie polecenia `uptime` i zwrócenie outputu polecenia do przeglądarki;
- domyślnie jest wysyłana lista dostępnych funkcji.

Szczegóły protokołu http, żądanie

Klient

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

Szczegóły protokołu http, odpowiedź

Serwer

HTTP/1.1 200 OK
Date: Mon, 21 Dec 2009 09:14:01 GMT
Server: Apache/2.0.54 (Debian GNU/Linux)
Content-Length: 37402

<dane>

Obsługa HTTP

Serwer webowy: obiekt klasy `BaseHTTPServer.HTTPServer`

- Obsługuje protokół http
- Wymaga obiektu do obsługi różnych rodzajów zapytań

Obsługa HTTP

Serwer webowy: obiekt klasy `BaseHTTPServer.HTTPServer`

- Obsługuje protokół http
- Wymaga obiektu do obsługi różnych rodzajów zapytań

Obsługa zapytań: klasa `BaseHTTPRequestHandler`

- Klasa abstrakcyjna
- Metody obsługujące żądania (GET, POST, HEADER,...)
- Metody konstrukcji odpowiedzi

Serwer

Implementacja

```
from BaseHTTPServer import *
import os

class MyHttpHandler(BaseHTTPRequestHandler):
```

Implementacja klasy MyHttpHandler

Obsługa żądania GET

```
def do_GET(self):
    self.send_response(200)
    self.send_header("Content-type", "text/html")
    self.end_headers()
    self.wfile.write("<html><head><head>")
    self.wfile.write("<body>")
    if self.path == "/uptime":
        self.uptime()
    else :
        self.menu()
    self.wfile.write("</body></html>")
    self.wfile.close()
```

Implementacja serwera

Implementacja *uptime*

```
def uptime(self):
    res = os.popen("uptime").read()
    self.wfile.write("<h1>Rezultat</h1>")
    self.wfile.write("<tt>" + res + "</tt>")
```

Implementacja serwera

Menu

```
def menu(self):
    self.wfile.write("<h1>Serwer</h1>")
    self.wfile.write("<ul>")
    self.wfile.write("<li><a href="uptime">uptime</a></li>")
    self.wfile.write("</ul>")
```

Serwer http

Uruchomienie całego serwera

```
address = ('', 8000)
httpd = BaseHTTPServer.HTTPServer(address, MyHttpHandler)
httpd.serve_forever()
```

Klient HTTP

```
import httplib
class przegladarka:
    def __init__(self):
        h = self.request("komputer12", "GET", "/uptime/")
        resp = h.getresponse()
        print ('Kod', resp.status)
        print (resp.read())
```

Klient HTTP

```
import httplib
class przegladarka:
    def __init__(self):
        h = self.request("komputer12", "GET", "/uptime/")
        resp = h.getresponse()
        print ('Kod', resp.status)
        print (resp.read())
```

Przygotowanie żądania

```
    def request(self, host, metoda, strona):
        naglowki = { "Host" : host, "Accept": "text/html" }
        h = httplib.HTTPConnection(host)
        h.request(metoda, strona, , naglowki)
        return h
```

Klient http

Wywołanie

przegladarka()

Batteries Included

Standardowe biblioteki

ftplib, poplib, mmtplib, nntplib, email, mimetools, mimetypes, base64

Batteries Included

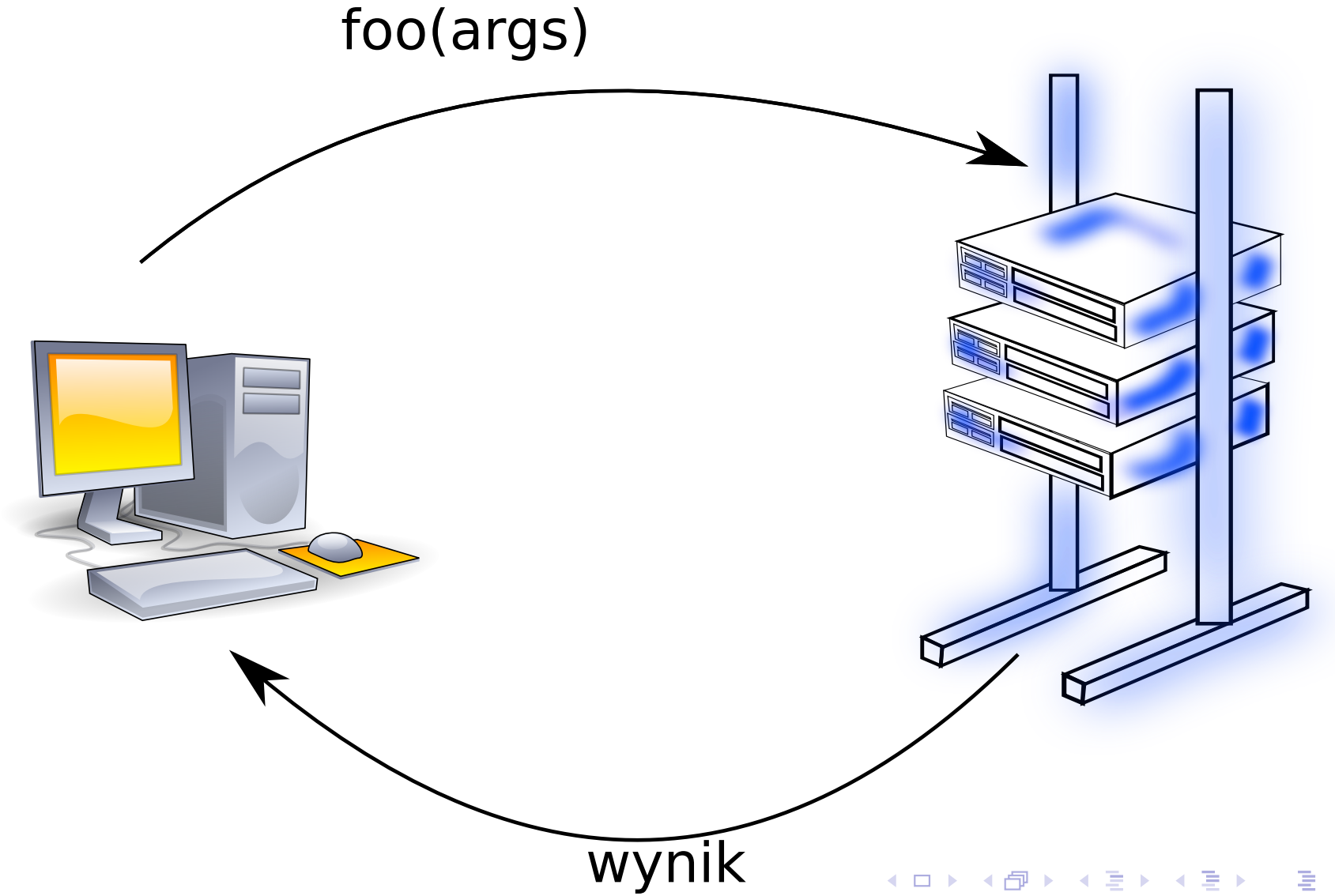
Standardowe biblioteki

ftplib, poplib, mmtplib, nntplib, email, mimetools, mimetypes, base64

Dla 'piratów'

TinyP2P

Serwery aplikacyjne



Wykorzystywane protokoły

- General InterORB Protocol
- Remote Java Invocation
- RPC
- .NET Remoting
- XML RPC
- ...

Zadanie

Serwer obliczający zdalnie *n*-tą liczbę Fibonacciego

Serwer

Implementacja funkcjonalności

```
def fib(n):
    if n < 2: return 1
    return fib(n - 1) + fib(n - 2)
```

Serwer

Implementacja funkcjonalności

```
def fib(n):
    if n < 2: return 1
    return fib(n - 1) + fib(n - 2)
```

Implementacja serwera

```
from SimpleXMLRPCServer import *

server = SimpleXMLRPCServer(("localhost", 8002))
server.register_function(fib)
server.register_function(lambda x, y: x + y, "add")
server.serve_forever()
```

Klient

Implementacja

```
import xmlrpclib
server = xmlrpclib.Server('http://localhost:8002')
print (server.fib(10))
print (server.add(2,3))
```

SOAP

Simple Object Access Protocol

- Rozszerzenie XML-RPC
- Standard W3C
- Brak bibliotek w standardowej dystrybucji Pythona
- Istnieją niezależne implementacje SOAP, np. SOAPPy

Serwisy SOAP

Publiczne serwisy SOAP

- Google
- Amazon
- Allegro

Serwer wielowątkowy

Jak zrobić wielowątkowy serwer XML RPC?

Rozwiązanie

Klasy 'domieszkujące' (*mix–ins*)

SocketServer.ThreadingMixIn
SocketServer.ForkingMixIn

```
import SocketServer
from SimpleXMLRPCServe import SimpleXMLRPCServer
```

```
import SocketServer
from SimpleXMLRPCServe import SimpleXMLRPCServer
```

```
class AsyncServer(SocketServer.ThreadingMixIn,
                  SimpleXMLRPCServer):
    pass

server = AsyncServer(('', 8002))
```

```
import SocketServer
from SimpleXMLRPCServe import SimpleXMLRPCServer
```

```
class AsyncServer(SocketServer.ThreadingMixIn,
                  SimpleXMLRPCServer):
    pass

server = AsyncServer(('', 8002))
```

```
server.register_function(foo)
server.serve_forever()
```

Plan wykładu

Zope

Z Object Publishing Environment

Co to jest

Pierwsza[1] obiektowa platforma webowa.

[1] nie umiem tego zweryfikować ;-)

Co to jest

Pierwsza[1] obiektowa platforma webowa.
Popularność Zope przyczyniła się do popularności Pythona

[1] nie umiem tego zweryfikować ;-)

Początki

Pierwsze wersje

1995: system do zarządzania reklamami w internecie, jako wspólne przedsięwzięcie kilku firm. Utworzenie później Zope Corp.

Początki

Pierwsze wersje

1995: system do zarządzania reklamami w internecie, jako wspólne przedsięwzięcie kilku firm. Utworzenie później Zope Corp.

1998

Powstanie **Zope 2**, przyłączenie PythonLabs (pracował tam Guido van Rossum).

Później

2004

Wydanie Zope 3 (BlueBream): całkowicie przepisany Zope

Zastosowania Zope

Plone

Oparty o Zope *system zarządzania treścią*.

W pełni obiektowy

Wszystko jest obiektem:

- nie ma plików ani katalogów, są obiekty i obiekty zawarte w innych obiektach;

Przykład: zawieranie plików

`Python/index.html` może oznaczać obiekt klasy `Protokol` w obiekcie `Folder` o nazwie `Python`.

W pełni obiektowy

Wszystko jest obiektem:

- nie ma plików ani katalogów, są obiekty i obiekty zawarte w innych obiektach;

Przykład: zawieranie plików

`Python/index.html` może oznaczać obiekt klasy `Protokol` w obiekcie `Folder` o nazwie `Python`.

- jest mechanizm tworzenia podklas i dziedziczenia,

Trwałość obiektów

ZODB: Zope Object Database

Obiektowa baza danych, implementująca transakcje, historię (z undo).

Trwałość obiektów

ZODB: Zope Object Database

Obiektowa baza danych, implementująca transakcje, historię (z undo).

ZEO: Zope Enterprise Objects

Rozszerzenie ZODB o wielodostęp.

URL

URL jest odwzorowywany na obiekty i ew. na metody. Wynikiem działania metody może być np. tekst html'owy.

Bezpieczeństwo

Rozbudowany system uprawnień.

Obiekty prezentacji

Zope Page Templates

Szablony stron — dla projektantów.

Obiekty prezentacji

Zope Page Templates

Szablony stron — dla projektantów.

Document Template Markup Language

"Programowalne" strony webowe — dla programistów.