Networked Programs

Chapter 12

Python for Informatics: Exploring Information
www.py4inf.com
Internet

Client

Server

Wikipedia

Internet
Network Architecture....
Transport Control Protocol (TCP)

- Built on top of IP (Internet Protocol)
- Assumes IP might lose some data - stores and retransmits data if it seems to be lost
- Handles “flow control” using a transmit window
- Provides a nice reliable pipe

http://en.wikipedia.org/wiki/Tin_can_telephone

http://www.flickr.com/photos/kitcowan/2103850699/
"In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an Internet Protocol-based computer network, such as the Internet."
TCP Port Numbers

- A port is an application-specific or process-specific software communications endpoint
- It allows multiple networked applications to coexist on the same server.
- There is a list of well-known TCP port numbers

http://en.wikipedia.org/wiki/TCP_and_UDP_port
Please connect me to the web server (port 80) on http://www.dr-chuck.com
Common TCP Ports

- Telnet (23) - Login
- SSH (22) - Secure Login
- HTTP (80)
- HTTPS (443) - Secure
- SMTP (25) (Mail)
- IMAP (143/220/993) - Mail Retrieval
- POP (109/110) - Mail Retrieval
- DNS (53) - Domain Name
- FTP (21) - File Transfer

Sometimes we see the port number in the URL if the web server is running on a "non-standard" port.
Sockets in Python

- Python has built-in support for TCP Sockets

```python
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect( ('www.py4inf.com', 80) )
```

http://docs.python.org/library/socket.html
I learned it last night! Everything is so simple!
Hello world is just print "Hello, world!"

I dunno... dynamic typing? whitespace?
Come join us! Programming is fun again!
It's a whole new world up here!
But how are you flying?

I just typed import antigavity
That's it?
... I also sampled everything in the medicine cabinet for comparison.
But I think this is the Python.
Application Protocol

- Since TCP (and Python) gives us a reliable socket, what do we want to do with the socket? What problem do we want to solve?
- Application Protocols
  - Mail
  - World Wide Web

HTTP - Hypertext Transport Protocol

• The dominant Application Layer Protocol on the Internet
• Invented for the Web - to Retrieve HTML, Images, Documents etc
• Extended to be data in addition to documents - RSS, Web Services, etc..
• Basic Concept - Make a Connection - Request a document - Retrieve the Document - Close the Connection

http://en.wikipedia.org/wiki/Http
• The HyperText Transport Protocol is the set of rules to allow browsers to retrieve web documents from servers over the Internet
What is a Protocol?

- A set of rules that all parties follow so we can predict each other's behavior
- And not bump into each other
- On two-way roads in USA, drive on the right hand side of the road
- On two-way roads in the UK, drive on the left hand side of the road
http://www.dr-chuck.com/page1.htm

protocol

document

http://www.youtube.com/watch?v=x2GylLq59rI

1:17 - 2:19
Getting Data From The Server

• Each the user clicks on an anchor tag with an href= value to switch to a new page, the browser makes a connection to the web server and issues a “GET” request - to GET the content of the page at the specified URL.

• The server returns the HTML document to the Browser which formats and displays the document to the user.
Making an HTTP request

- Connect to the server like www.dr-chuck.com
- a "hand shake"
- Request a document (or the default document)
  - GET http://www.dr-chuck.com/page1.htm
  - GET http://www.mlive.com/ann-arbor/
  - GET http://www.facebook.com
The First Page

If you like, you can switch to the Second Page.
The First Page

If you like, you can switch to the Second Page.
The First Page

If you like, you can switch to the Second Page.
GET http://www.dr-chuck.com/page2.htm

Web Server

80

Browser

The First Page

If you like, you can switch to the Second Page.
GET http://www.dr-chuck.com/page2.htm

<h1>The Second Page</h1>

If you like, you can switch back to the <a href="page1.htm">First Page</a>.

<p></p>
GET http://www.dr-chuck.com/page2.htm

<h1>The Second Page</h1>
<p>If you like, you can switch back to the <a href="page1.htm">First Page</a>.</p>

If you like, you can switch to the <a href="page1.htm">Second Page</a>.
Let's Write a Web Browser!
Internet Standards

- The standards for all of the Internet protocols (inner workings) are developed by an organization
- Internet Engineering Task Force (IETF)
- www.ietf.org
- Standards are called “RFCs” - “Request for Comments”

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information...
5 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

\[
\text{Request} \quad = \quad \text{Request-Line} \quad ; \quad \text{Section 5.1} \\
\quad \quad \ast(( \text{general-header} \quad ; \quad \text{Section 4.5} \\
\quad \quad \quad | \quad \text{request-header} \quad ; \quad \text{Section 5.3} \\
\quad \quad \quad \quad | \quad \text{entity-header} \quad ) \quad \text{CRLF}) \quad ; \quad \text{Section 7.1} \\
\quad \quad \text{CRLF} \\
\quad \quad [ \quad \text{message-body} \quad ] \quad ; \quad \text{Section 4.3}
\]

5.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

\[
\text{Request-Line} \quad = \quad \text{Method SP Request-URI SP HTTP-Version CRLF}
\]
Making an HTTP request

• Connect to the server like www.dr-chuck.com
  • a "hand shake"

• Request a document (or the default document)
  • GET http://www.dr-chuck.com/page1.htm
  • GET http://www.mlive.com/ann-arbor/
  • GET http://www.facebook.com
"Hacking" HTTP

$ telnet www.dr-chuck.com 80
Trying 74.208.28.177...
Escape character is '^]'.
GET http://www.dr-chuck.com/page1.htm
<h1>The First Page</h1>
<p>
If you like, you can switch to the
</p>

Port 80 is the non-encrypted HTTP port
Accurate Hacking in the Movies

- Matrix Reloaded
- Bourne Ultimatum
- Die Hard 4
- ...

http://nmap.org/movies.html
http://www.youtube.com/watch?v=Zy5_gYu_isg
$ telnet www.dr-chuck.com 80
Trying 74.208.28.177...
Escape character is '^]'.
GET http://www.dr-chuck.com/page1.htm
<h1>The First Page</h1>
<p>If you like, you can switch to the
Connection closed by foreign host.
Hmmm - This looks kind of complex.. Lots of GET commands
si-csev-mbp:tex csev$ telnet www.umich.edu 80
Trying 141.211.144.190...
Connected to www.umich.edu.
Escape character is '^]'.

GET /
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>University of Michigan</title>
<meta name="description" content="University of Michigan is one of the top universities of the world, a diverse public institution of higher learning, fostering excellence in research. U-M provides outstanding undergraduate, graduate and professional education, serving the local, regional, national and international communities." />
</head>
As the browser reads the document, it finds other URLs that must be retrieved to produce the document.
Firebug reveals the detail...

- If you haven't already installed the **Firebug** FireFox extension you need it now
- It can help explore the HTTP request-response cycle
- Some simple-looking pages involve *lots of requests*:
  - HTML page(s)
  - Image files
  - CSS Style Sheets
  - Javascript files
AppEngineLearn

This site provides materials to help learn the Google Application Engine. Before you start to learn the Google Application Engine you should be basically familiar with the Python programming language.

New: You can take a look at the draft book chapters for my upcoming O'Reilly AppEngine book titled, "Building Cloud Applications with Google AppEngine".

- Installing Python and JEdit - We recommend using JEdit as your programmer editor and it will be used throughout the Podcasts.
- Installing the Application Engine and writing your first Application.
  - Macintosh: (Handout, Source Code, Screencast, YouTube)
  - Windows Vista: (Handout, Source Code, High Quality Screencast, YouTube)

Console panel is disabled

Use this page to enable or disable following panels. Enabling these panels will reduce performance and will cause a page reload.

- [ ] Console Support for Console logging. Disabled Always
- [x] Script Support for JavaScript debugging. Enabled for www.appenginelearn.com

Apply settings for www.appenginelearn.com
AppEngineLearn

This site provides materials to help learn the Google Application Engine. Before you start to learn the Google Application Engine you should be basically familiar with the Python programming language.

New: You can take a look at the draft book chapters for my upcoming O'Reilly AppEngine book titled, "Building Cloud Applications with Google AppEngine".

- **Installing Python and JEdit** - We recommend using JEdit as your programmer editor and it will be used throughout the Podcasts.
- **Installing the Application Engine and writing your first Application.**
  - Macintosh: (Handout, Source Code, Screencast, YouTube)
  - Windows Vista: (Handout, Source Code, High Quality Screencast, YouTube)
An HTTP Request in Python

```python
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('www.py4inf.com', 80))
mysock.send('GET http://www.py4inf.com/code/romeo.txt HTTP/1.0\n\n')

while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    print data

mysock.close()
```
HTTP/1.1 200 OK
Date: Sun, 14 Mar 2010 23:52:41 GMT
Server: Apache
Last-Modified: Tue, 29 Dec 2009 01:31:22 GMT
ETag: "143c1b33-a7-4b395bea"
Accept-Ranges: bytes
Content-Length: 167
Connection: close
Content-Type: text/plain

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    print data

HTTP Header

HTTP Body
Making HTTP Easier With urllib
Using `urllib` in Python

- Since HTTP is so common, we have a library that does all the socket work for us and makes web pages look like a file

```python
import urllib

fhand = urllib.urlopen('http://www.py4inf.com/code/romeo.txt')

for line in fhand:
    print line.strip()
```

http://docs.python.org/library/urllib.html

`urllib1.py`
import urllib

fhand = urllib.urlopen('http://www.py4inf.com/code/romeo.txt')

for line in fhand:
    print line.strip()

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief
import urllib

fhand = urllib.urlopen('http://www.py4inf.com/code/romeo.txt')

counts = dict()
for line in fhand:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

print counts
import urllib

fhand = urllib.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print line.strip()

<h1>The First Page</h1>
<p>
If you like, you can switch to the
<a href="http://www.dr-chuck.com/page2.htm">
Second Page</a>.
</p>
import urllib

fhand = urllib.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print line.strip()

<h1>The First Page</h1>
<p>
</p>
import urllib

fhand = urllib.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print line.strip()
Parsing HTML
(a.k.a Web Scraping)
What is Web Scraping?

- When a program or script pretends to be a browser and retrieves web pages, looks at those web pages, extracts information and then looks at more web pages.

- Search engines scrape web pages - we call this “spidering the web” or “web crawling”

http://en.wikipedia.org/wiki/Web_scraping
http://en.wikipedia.org/wiki/Web_crawler
charles-severances-macbook-air:Scraping csev$ python
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import urllib
>>> f = urllib.urlopen("http://www.dr-chuck.com/")
>>> contents = f.read()
>>> f.close()
>>> print len(contents)
95328
>>> print contents[0:30]
<html>
<head>
<title>Dr. C
>>>
Why Scrape?

• Pull data - particularly social data - who links to who?

• Get your own data back out of some system that has no “export capability”

• Monitor a site for new information

• Spider the web to make a database for a search engine
Scraping Web Pages

- There is some controversy about web page scraping and some sites are a bit snippy about it.
  - Google: facebook scraping block
- Republishing copyrighted information is not allowed
- Violating terms of service is not allowed
User Conduct

You understand that except for advertising programs offered by us on the Site (e.g., Facebook Flyers, Facebook Marketplace), the Service and the Site are available for your personal, non-commercial use only. You represent, warrant and agree that no materials of any kind submitted through your account or otherwise posted, transmitted, or shared by you on or through the Service will violate or infringe upon the rights of any third party, including copyright, trademark, privacy, publicity or other personal or proprietary rights; or contain libelous, defamatory or otherwise unlawful material.

In addition, you agree not to use the Service or the Site to:

- harvest or collect email addresses or other contact information of other users from the Service or the Site by electronic or other means for the purposes of sending unsolicited emails or other unsolicited communications;
- use the Service or the Site in any unlawful manner or in any other manner that could damage, disable, overburden or impair the Site;
- use automated scripts to collect information from or otherwise interact with the Service or the Site;
The Easy Way - Beautiful Soup

- You could do string searches the hard way
- Or use the free software called BeautifulSoup from www.crummy.com

http://www.crummy.com/software/BeautifulSoup/

Place the BeautifulSoup.py file in the same folder as your Python code...
import urllib
from BeautifulSoup import *

url = raw_input('Enter - ')
html = urllib.urlopen(url).read()
soup = BeautifulSoup(html)

# Retrieve a list of the anchor tags
# Each tag is like a dictionary of HTML attributes

tags = soup('a')

for tag in tags:
    print tag.get('href', None)
<h1>The First Page</h1>
<p>If you like, you can switch to the <a href="http://www.dr-chuck.com/page2.htm">Second Page</a>.</p>

```python
html = urllib.urlopen(url).read()
soup = BeautifulSoup(html)
tags = soup('a')
for tag in tags:
    print tag.get('href', None)
```

default:

```
python urllinks.py
Enter - http://www.dr-chuck.com/page1.htm
http://www.dr-chuck.com/page2.htm
```
Summary

• The TCP/IP gives us pipes / sockets between applications
• We designed application protocols to make use of these pipes
• HyperText Transport Protocol (HTTP) is a simple yet powerful protocol
• Python has good support for sockets, HTTP, and HTML parsing