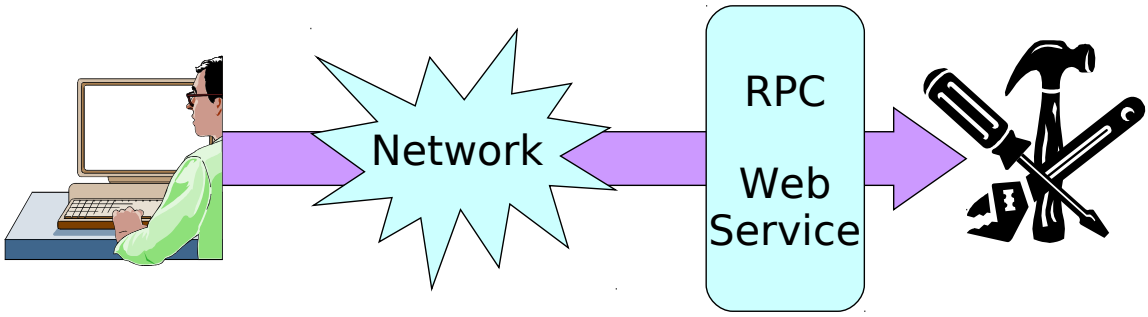
Remote Procedure Calling

Dr. Andrew C.R. Martin
andrew.martin@ucl.ac.uk
http://www.bioinf.org.uk/

Aims and objectives

- Understand the concepts of **remote procedure calling and web services**
- To be able to describe different methods of **remote procedure calls**
- Understand the problems of '**screen scraping**'
- Know how to **write code using urllib.request**

What is RPC?



A network accessible interface to application functionality using standard Internet technologies

Why do RPC?

- distribute the load between computers
- access to other people's methods
- access to the latest data

Ways of performing RPC

- screen scraping
- simple cgi scripts (REST)
- custom code to work across networks
- standardized methods
  - (e.g. CORBA, SOAP, XML-RPC)

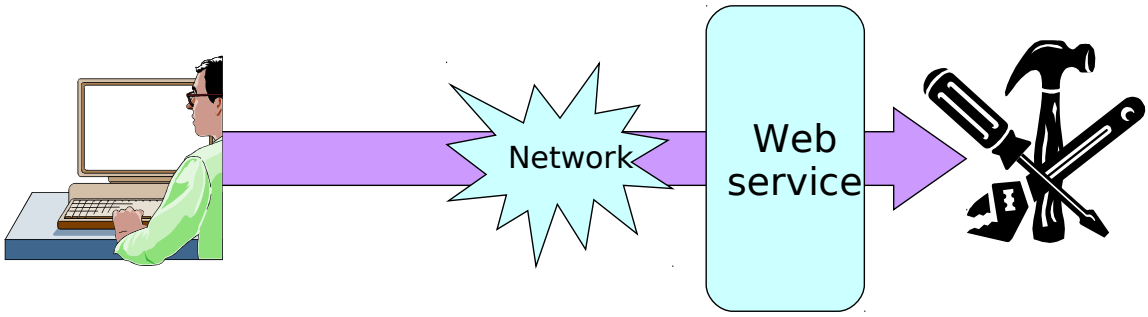
Web services

- RPC methods which work across the internet are often called "**Web Services**"

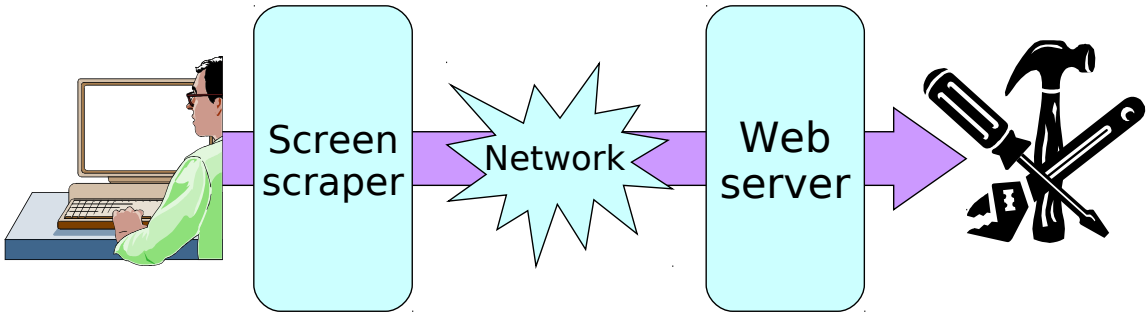
- Web Services can also
  - be self-describing (WSDL)
  - provide methods for discovery (UDDI)

Screen scraping

Web service

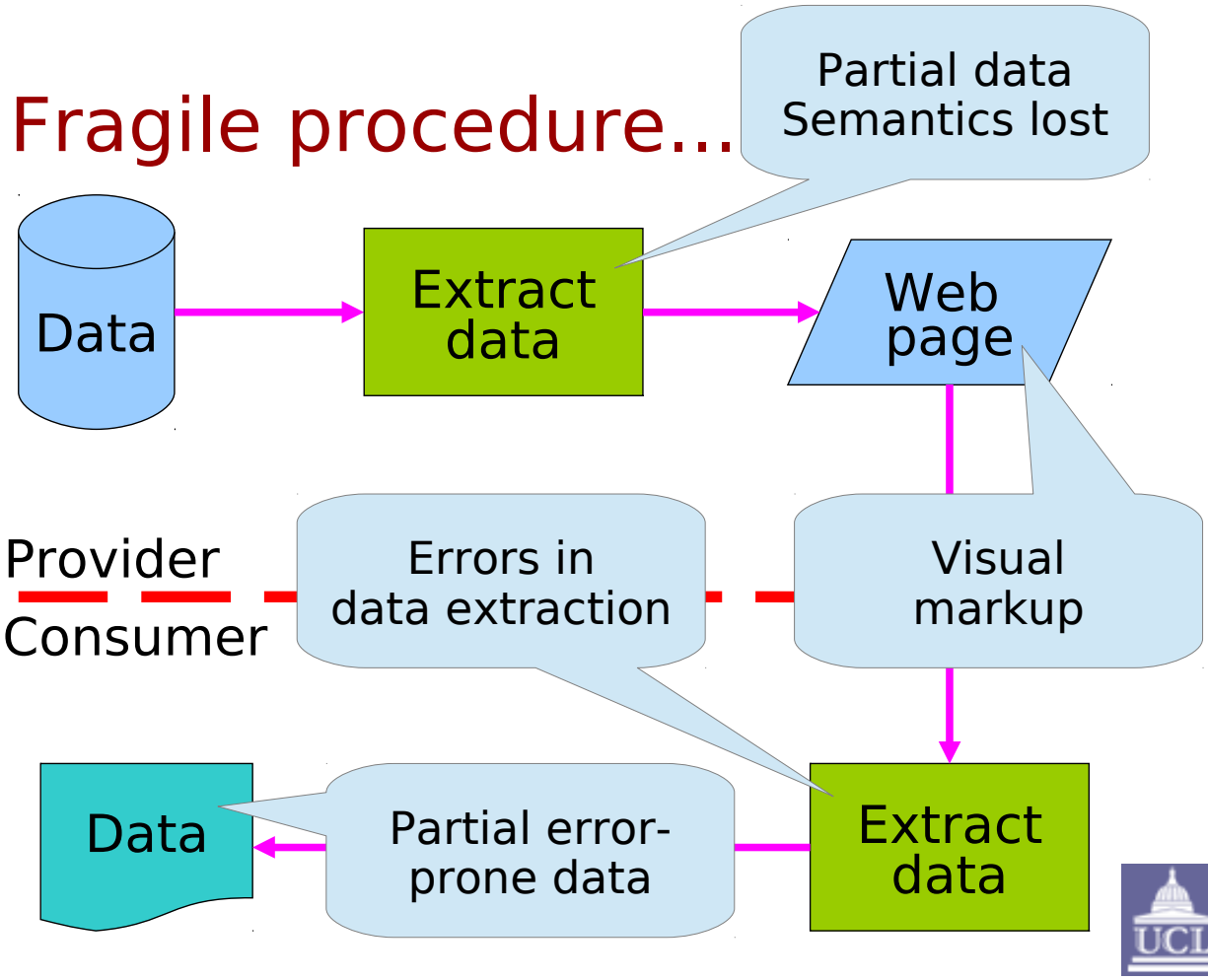


Screen scraping



Extracting content from a web page

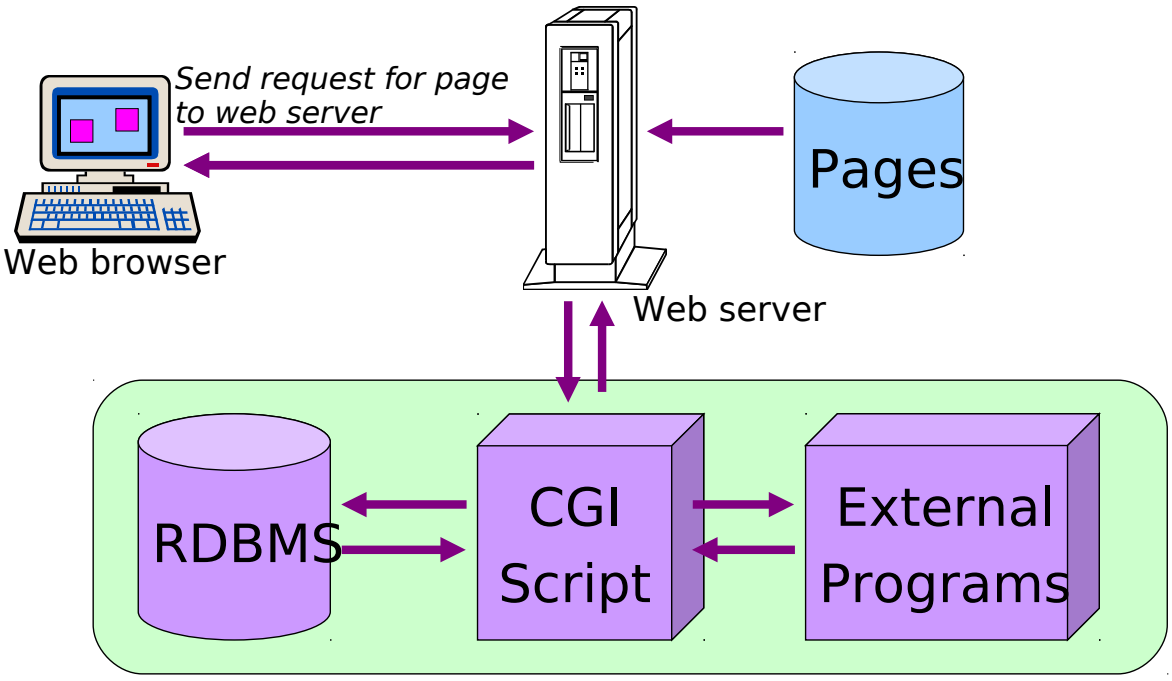
Fragile procedure...



Fragile procedure...

- Trying to **interpret semantics** from display-based markup
- If the **presentation changes**, the screen scraper **breaks**

Web servers...



Screen scraping

Straightforward in Python

- Python **urllib.request** module
  - trivial to write a web client
- **Pattern matching** and **string handling** routines

Example scraper…

- A program for **secondary structure prediction**
- Want a program that:
  - specifies an amino acid sequence
  - provides a secondary structure prediction
    - Using a remote server!

We want to be able to do:

```
#!/usr/bin/env python3

seq = """KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDY
GILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNR
CKGTDVQAWIRGCRL"""

seq = seq.replace('\n', '')

ss = PredictSS(seq)
if(ss != ""):
    print (seq)
    print (ss)
```

Add a dummy test routine

```
#!/usr/bin/env python3

def PredictSS(seq):
    retval = ''
    for i in range(0, len(seq)):
        retval += '?'
    return(retval)


seq = """KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDY
GILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNR
CKGTDVQAWIRGCRL"""

seq = seq.replace('\n', '')

ss = PredictSS(seq)
if(ss != ""):
    print (seq)
    print (ss)
```

Results:

```
KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSR
WWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQA
WIRGCRL
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
???????
```

Obviously we wish to replace PredictSS()

Example scraper...

- **NNPREDICT** web server at

alexander.compbio.ucsf.edu/~nomi/nnpredict.html

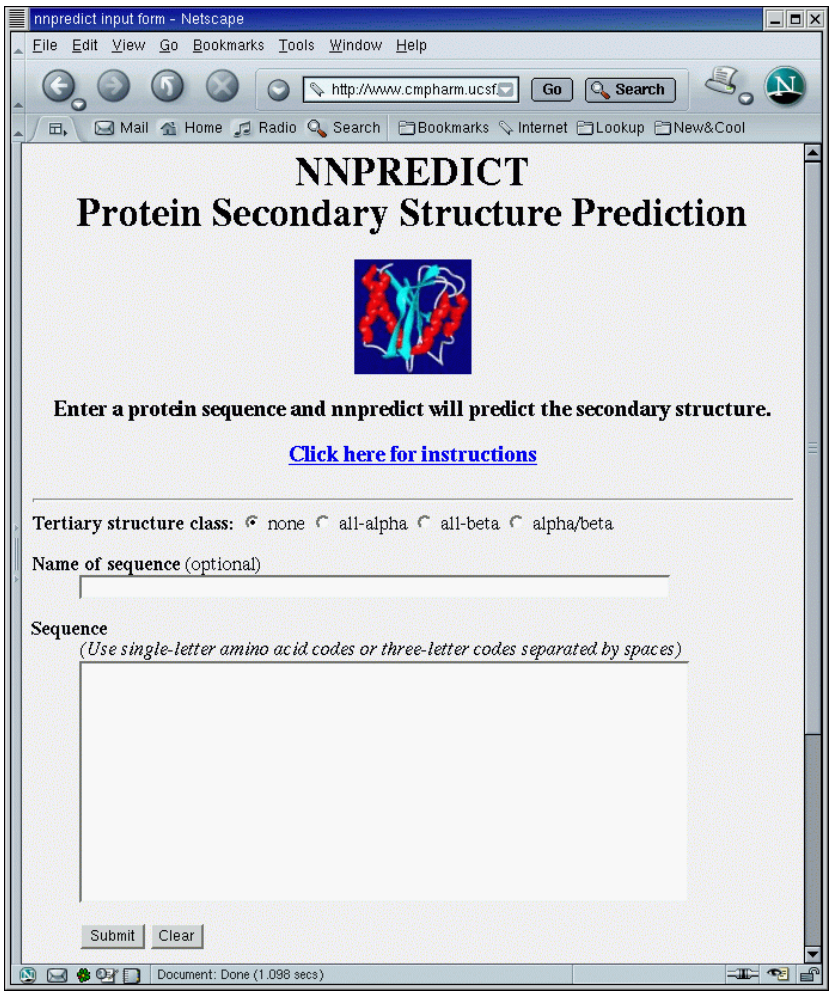
alexander.compbio.ucsf.edu/cgi-bin/nnpredict.pl

Unfortunately no longer available.

Dummy version:

www.bioinf.org.uk/teaching/bbk/biocomp2/rpc/nnpredict/

www.bioinf.org.uk/teaching/bbk/biocomp2/rpc/nnpredict/nnpredict.cgi



Example scraper...

- Program must:
  - **connect** to web server
  - **submit** the sequence
  - obtain the **results** and extract data
- Examine the source for the page...

```
<form method="POST"
action="http://alexander.compbio.ucsf.edu/cgi-bin/nnpredict.pl">

<b>Tertiary structure class:</b>

<input TYPE="radio" NAME="option"
       VALUE="none" CHECKED> none
<input TYPE="radio" NAME="option"
       VALUE="all-alpha"> all-alpha
<input TYPE="radio" NAME="option"
       VALUE="all-beta"> all-beta
<input TYPE="radio" NAME="option"
       VALUE="alpha/beta"> alpha/beta

<b>Name of sequence</b>
<input name="name" size="70">

<b>Sequence</b>
<textarea name="text" rows=14 cols=70></textarea>

</form>
```

Example scraper...

- **option**

'none', 'all-alpha', 'all-beta', or 'alpha/beta'

- **name**

optional name for the sequence

- **text**

the sequence

Example scraper...

```
from urllib import request

def PredictSS(seq):
    url    = "http://www.bioinf.org.uk/teaching/"\
             "bbk/biocomp2/rpc/"\
             "nnpredict/nnpredict.cgi"
    params = "option=none&name=&text=" + seq
    fullurl= url + "?" + params

    result = request.urlopen(fullurl).read()
    result = str(result, encoding='utf-8')

    if(result != ''):
        ss = ParseSS(result)
        return(ss)
    else:
        sys.stderr.write("Nothing was returned\n")

    return("")
```

Example scraper...

```
from urllib import request

def PredictSS(seq):
    url    = "http://www.bioinf.org.uk/teaching/"\
             "bbk/biocomp2/rpc/"\
             "nnpredict/nnpredict.cgi"
    params = "option=none&name=&text=" + seq
    fullurl= url + "?" + params

    result = request.urlopen(fullurl).read()
    result = str(result

    if(result != ''):
        ss = ParseSS(r
        return(ss)
    else:
        sys.stderr.write("Nothing was returned\n")

    return("")
```

Specify URL
and parameters

Example scraper...

```
from urllib import request

def PredictSS(seq):
    url    = "http://                                    '\
             "bbk/bio
             "nnpredi
    params = "option=none                 text=" + seq
    fullurl= url + "?" + p                ms

    result = request.urlopen(fullurl).read()
    result = str(result, encoding='utf-8')

    if(result != ''):
        ss = ParseSS(result)
        return(ss)
    else:
        sys.stderr.write("Nothing was returned\n")

    return("")
```

Retrieve data and convert from byte string

Example scraper...

```
from urllib import request

def PredictSS(seq):
    url    = "http://www.bioinf.org.uk/teaching/"\
             "bbk/biocomp2/rpc/"\
             "nnpredict/nnpredict.cgi"
    params = "option=
    fullurl= url + "?

    result = request.
    result = str(result,                  g='utf-8')

    if(result != ''):
        ss = ParseSS(result)
        return(ss)
    else:
        sys.stderr.write("Nothing was returned\n")

    return("")
```

Extract required info From the HTML

```
<HTML><HEAD>
<TITLE>NNPREDICT RESULTS</TITLE>
</HEAD>
<BODY bgcolor="F0F0F0">
<h1 align=center>Results of nnpredict query</h1>
<p><b>Tertiary structure class:</b>  alpha/beta
<p><b>Sequence</b>:<br>
<tt>
MRSLLILVLCFLPLAALGKVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQA<br>
TNRNTDGSTDYGILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDG<br>
NGMNAWVAWRNRCKGTDVQAWIRGCRL<br>
</tt>
<p><b>Secondary structure prediction
<i>(H = helix, E = strand, - = no prediction)</i>:<br></b>
<tt>
----EEEEEEE-H---H--EE-HHHHHHHHHH-------------HHHHHH--------<br>
-----------HHHHE-------------------------------HH-----EE---<br>
---HHHHHHH--------HHHHHHH--<br>
</tt>
</body></html>
```

Example scraper...

```
import re


def ParseSS(result):
   result = result.replace('\n','')
   result = result.replace('<br>','')

   p = re.compile('^.*<tt>(.*)<\/tt>.*$')
   m = p.match(result)
   result = m.group(1) # Returns the () match

   return(result)
```

Example scraper...

```
import re


def ParseSS(result):
   result = result.replace('\n','')
   result = result.replace('<br>','')

   p = re.compile('^.*<tt>(.*)<\/tt>.*$')
   m = p.match(result)
   result = m.group(1) # Returns the () match

   return(result)
```

Remove returns and
 tags

Example scraper...

```
import re


def ParseSS(result):
   result = result.replace('\
   result = result.replace('<br>'
                                  )

   p = re.compile('^.*<tt>(.*)<\/tt>.*$')
   m = p.match(result)
   result = m.group(1) # Returns the () match

   return(result)
```

Create and match regular expression

Example scraper...

```
import re


def ParseSS(result):
   result = result.replace('\n','')
   result = result.replace('<br>','')

   p = re.compile('^.*<tt>(.*)<\/tt>.*$')
   m = p.match(result)
   result = m.group(1) # Returns the () match

   return(result)
```

Extract the () group match

If authors changed presentation of results, this may well break!

Pros and cons

Advantages

- 'service provider' doesn't do anything special

Disadvantages

- screen scraper will break if format changes
- may be difficult to determine semantic content

Simple CGI scripts

REST: Representational State Transfer

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

Simple CGI scripts

Extension of screen scraping

- relies on service provider to provide a script designed specifically for remote access

Client identical to screen scraper

- but guaranteed that the data will be parsable (plain text or XML)

Simple CGI scripts

Server's point of view

- provide a modified CGI script which returns plain text
- May be an option given to the CGI script

Simple CGI scripts

- '**Entrez** programming utilities'
  http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html
- Search using EUtils is performed in **2 stages**:
  - specified search string returns a set of PubMed Ids
  - fetch the results for each of these PubMed IDs in turn.

Simple CGI scripts

- I have provided a script you can try to extract papers from PubMed
  (Note this is in Perl not Python)

- You can try Web Services that we have provided at:
  - www.bioinf.org.uk/pdbsws/cgi.html
  - www.bioinf.org.uk/abs/abdb/ws.cgi

Custom code

Custom code

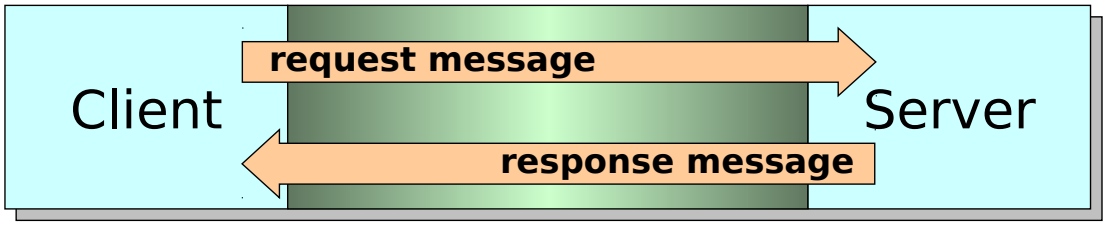
- Generally used to distribute tasks on a **local network**
- Code is **complex**
  - low-level OS calls
  - sample on the web

Custom code

- Link relies on **IP address** and a '**port**'
- Ports tied to a particular service
  - port 80 : HTTP
  - port 22 : ssh
  - See /etc/services

Custom code

- Generally a client/server model:
  - server listens for messages
  - client makes requests of the server



Custom code: server

- Server creates a '**socket**' and '**binds**' it to a port

- **Listens** for a connection from clients
- **Responds** to messages received
- **Replies** with information sent back to the client

Custom code: client

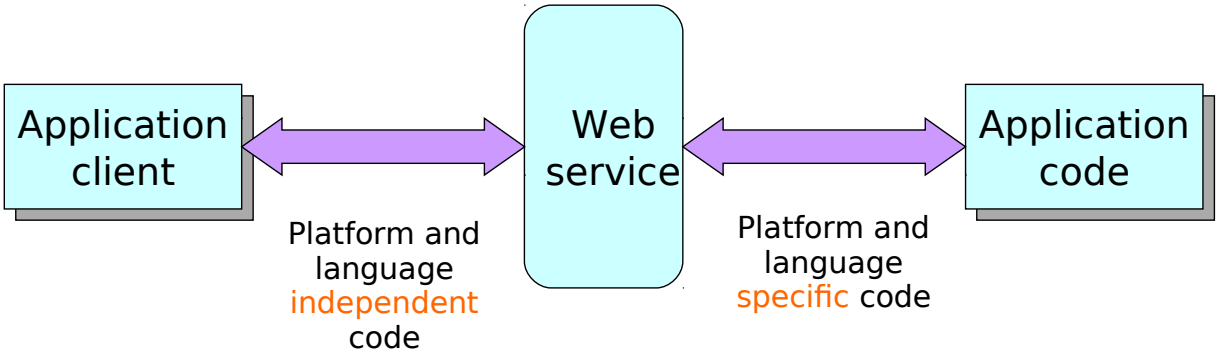
- Client creates a **socket**
- **Binds** it to a **port** and the **IP address** of the server
- **Sends** data to the server
- **Waits** for a response
- Does something with the returned data

Standardized Methods

Standardized methods

- Various methods. e.g.
  - CORBA
  - XML-RPC
  - **SOAP**
- Will now concentrate on SOAP...

Advantages of SOAP

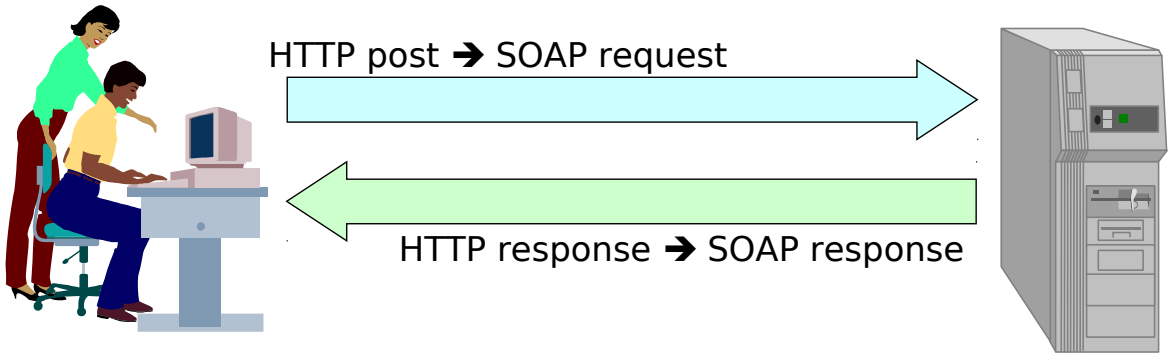


Advantages of SOAP

Information encoded in XML

- Language independent
- All data are transmitted as simple text

Advantages of SOAP

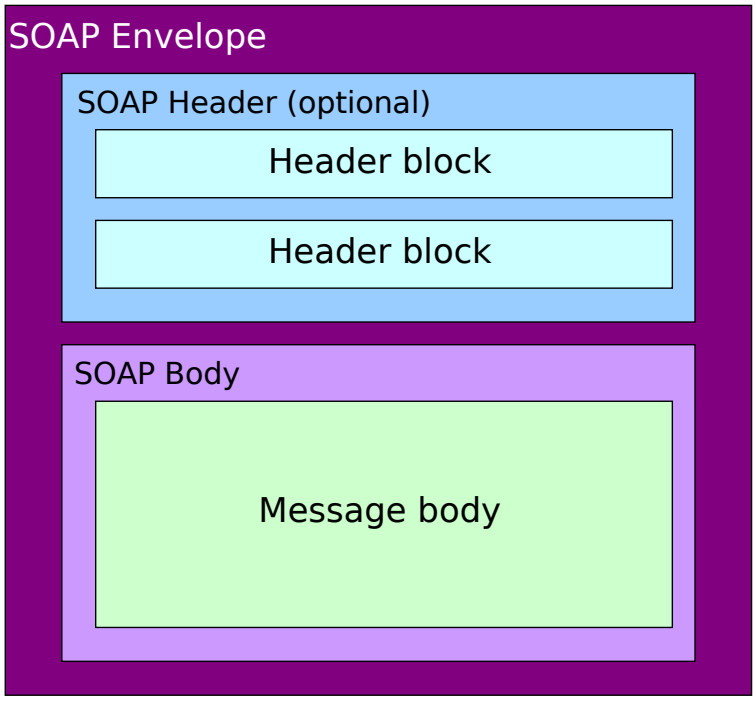


- Normally uses **HTTP** for transport
  - Firewalls allow access to the HTTP protocol
  - Same systems will allow SOAP access

Advantages of SOAP

- W3C standard
- Libraries available for many programming languages

SOAP messages



Example SOAP message

Header block

- Specifies data must be handled as a single 'transaction’

Message body

- contains a sequence simply encoded in XML
- Perfectly legal, but more common to use special RPC encoding

Subroutine calls

Only important factors

- the type of the variables
- the order in which they are handed to the subroutine

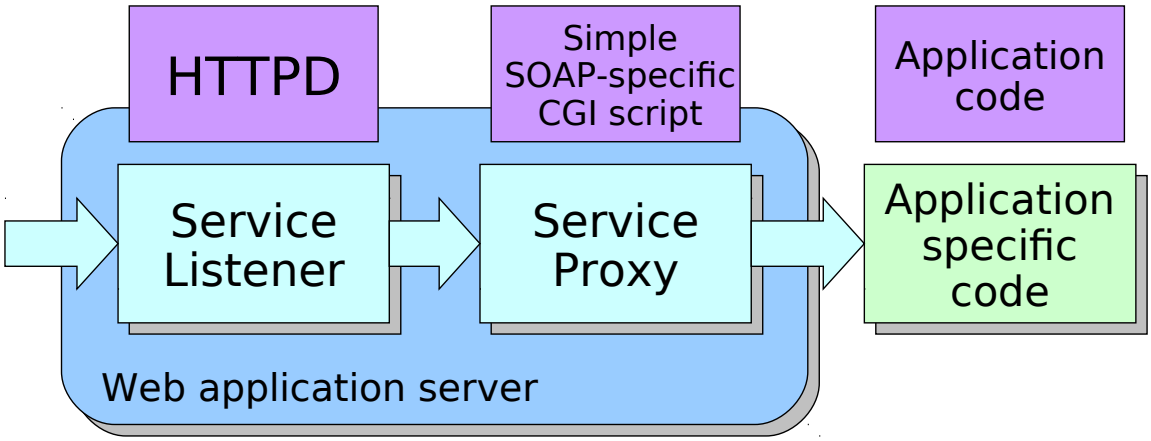
SOAP transport

- SOAP is a **packaging** protocol which provides **standard encoding** for variables:
  - Integers, floats, strings, arrays, hashes, structures
- Layered on **networking** and **transport**
  - SOAP doesn't care what these are

SOAP transport

- Generally uses **HTTP**, but may also use:
  - FTP, raw TCP, SMTP, POP3, Jabber, etc.
- HTTP is **pervasive** across the Internet.
- **Request-response** model of RPC matches HTTP

A simple SOAP server



Summary

- RPC allows access to methods and data on **remote computers**
- **Four main ways** of achieving this
  - Screen scraping
  - Special CGI scripts
  - Custom code
  - Standardized methods (SOAP, etc.)

Summary

- **SOAP** has not caught on as much as was hoped
- **REST** is much simpler to implement and to understand
- REST is simply a **CGI script**

BOOKS:

- Mitchell (2015) *Web Scraping with Python – Collecting Data from the Modern Web.* O'Reilly
- Richardson *et al.* (2013) *RESTful Web APIs.* O'Reilly
- Amundsen (2016) *RESTful Web Clients*. O'Reilly