Social Data Mining and Analysis:

Data Mining

Bruno Gonçalves

www.bgoncalves.com

Python Modules

- urllib2
- posixpath
- requests
- json
- OAuth
- twitter (https://pypi.python.org/pypi/twitter)
- NetworkX

urllib2

http://docs.python.org/2/library/urllib2.html

- urllib2.urlopen(url) opens a url for reading and returns a "file handle"-like object
- Information about the webpage can be obtained with the .info() method in the form of an HTTPMessage
- The .geturl() method returns the final location of the webpage.
  .urlopen() follows redirects until it connects with the final content.
- .getcode() returns the status code of the call
  - 200 OK
  - 404 File Not Found
  - 500 Internal Server Error
- webserver returned headers can be found in the dict field inside the HTTPMessage object

posixpath

http://docs.python.org/2/library/os.path.html

- Manipulate paths in a POSIX operating system
- Also useful to extract information from remote resource paths
- Aliased to os.path if your operating systems is POSIX
- https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f -> Path in remote filesystem
- .basename(path) -> returns the file name (if there is one) 5304b652498e734439d8711f
- .dirname(path) -> return the directory portion /tyayayayaa/checkin

requests

http://requests.readthedocs.org/en/latest/

- .get(url) open the given url for reading (like .urlopen()) and returns a Response
- .get(url, auth=("user", "pass")) open the given url and authenticate with username="user" and password="pass" (Basic Authentication)
- Response.status_code is a field that contains the calls status code
- Response.headers is a dict containing all the returned headers
- Response.text is a field that contains the content of the returned page
- Response.url contains the final url after all redirections
- Response.json() parses a JSON response (throws a JSONDecodeError exception if response is not valid JSON). Check "content-type" header field.

requests

http://requests.readthedocs.org/en/latest/

```
import requests
import sys

url = "http://httpbin.org/basic-auth/user/passwd"

request = requests.get(url, auth=("user", "passwd"))

if request.status_code != 200:
    print >> sys.stderr, "Error found", request.get_code()

content_type = request.headers[“content-type"]

response = request.json()

if response["authenticated"]:
    print "Authentication Successful"
```

- .get(url, headers=headers) open the url for reading but pass “headers” contents to server when establishing a connection
    - Useful to override default values of headers or add custom values to help server decide how to handle our request
    - Most commonly used to spoof the user-agent

requests

http://requests.readthedocs.org/en/latest/

```
import requests
from BeautifulSoup import BeautifulSoup

url = "http://whatsmyuseragent.com/"

headers = {"User-agent" : "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101 Firefox/25.0"}

request_default = requests.get(url)
request_spoofed = requests.get(url, headers=headers)

soup_default = BeautifulSoup(request_default.text)
soup_spoofed = BeautifulSoup(request_spoofed.text)

print "Default:", soup_default.h2.text
print "Spoofed:", soup_spoofed.h2.text
```

- Some servers use the User-agent string to decide how to format the output
  - Correctly handle specific versions of web browsers
  - Provide lighter/simplified versions to users on their mobiles
  - Refusing access to automated tools, etc

json

- JavaScript Object Notation - Serialization format originally developed for Javascript
- Currently widely accepted format for data dissemination
- Most languages have excellent libraries to handle it
- .loads(obj_str) - load JSON data from a string - returns native Python object
- .load(fp) - load JSON data from a file handle - returns native Python object
- .dumps(obj) - convert JSON data to a string
- .dump(obj, fp) - write the string version of obj to the file handle fp

NetworkX

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - MultiGraph - Multi-edged Graph
    - MultiDiGraph - Directed Multigraph
- Similar interface for all types of graphs
- Nodes can be any type of Python object - Practical way to manage relationships

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j
- .add_edges_from() Adds a list of edges. Individual edges are represented by tuples
- .remove_node(node_id)/.remove_nodes_from() Removing a node removes all associated edges
- .remove_edge(node_i, node_j)/.remove_edges_from()

Graph Properties

- .nodes() Returns the list of nodes
- .edges() Returns the list of edges
- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs
- .is_connected() Returns true if the node is connected
- .is_weakly_connected()/.is_strongly_conencted() for DiGraph
- .connected_components() A list of nodes for each connected component

Authentication

OAuth 1

http://hueniverse.com/oauth/

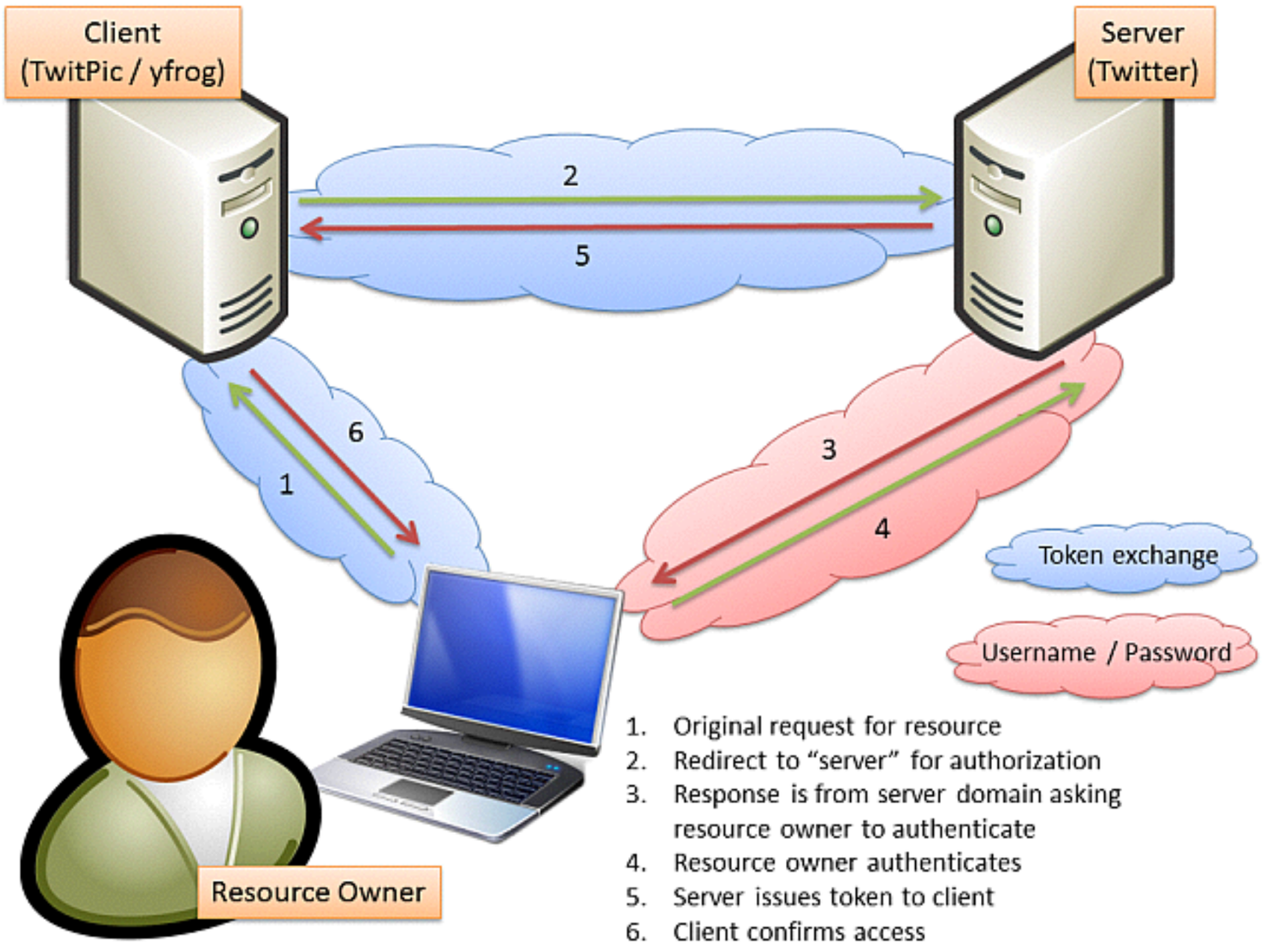
- "An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications."

- The idea is to allow for a safe way to share privileges without divulging private credentials

  - Give XPTO Application permission to post to your Twitter account without having to trust the developers of XPTO with your username/password and while being able to unilaterally revoke privileges.

OAuth 1

http://hueniverse.com/oauth/



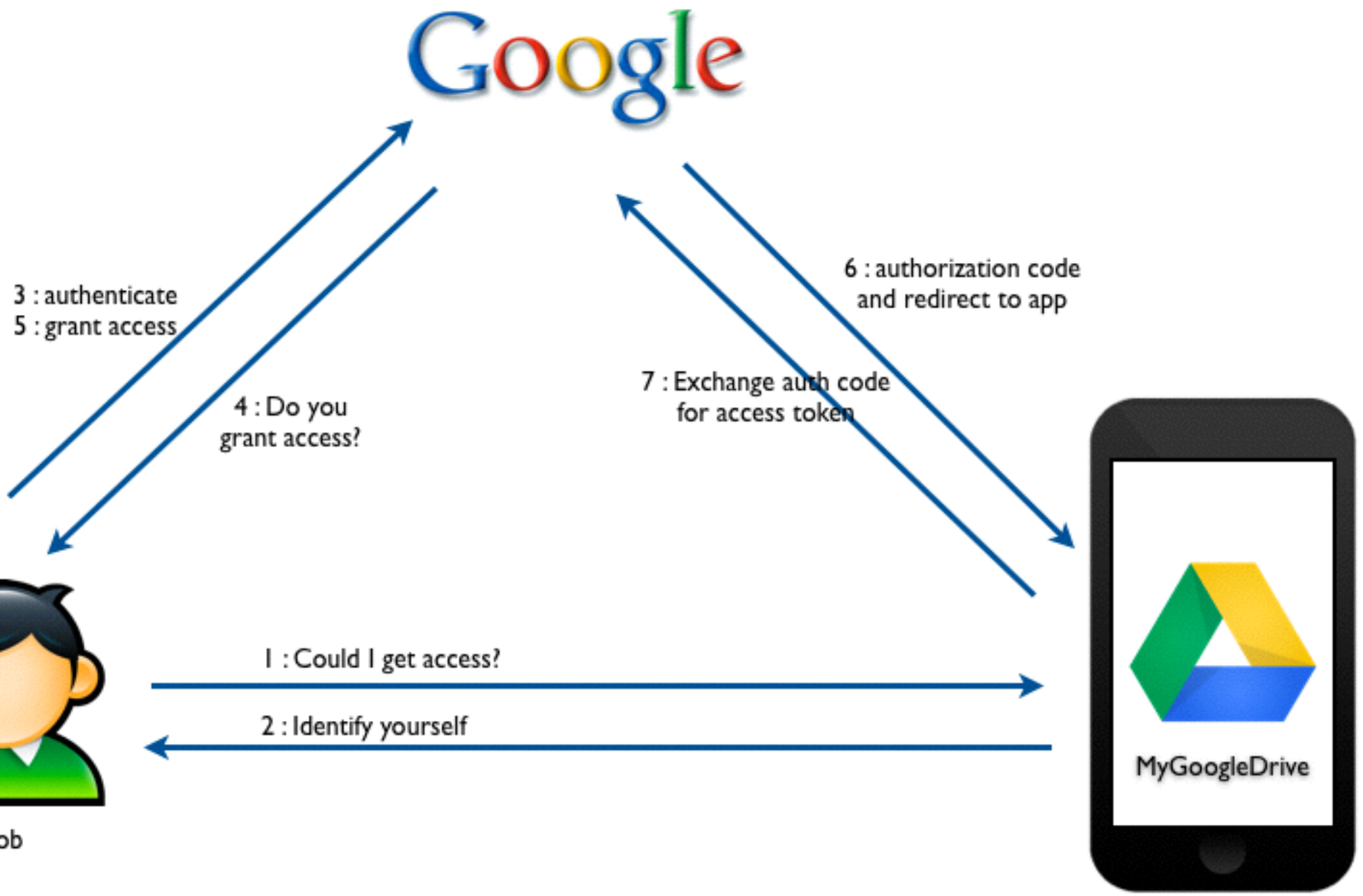
1. Original request for resource
2. Redirect to “server” for authorization
3. Response is from server domain asking resource owner to authenticate
4. Resource owner authenticates
5. Server issues token to client
6. Client confirms access

OAuth 1

http://hueniverse.com/oauth/

- After the "OAuth dance" is concluded, client application has two sets of keys:
  - one that uses to identify itself as a valid application (api_key, api_secret)
  - one that uses to identify the user it wants to access (token, token_secret)
- You can revoke access at any time by letting the token provider that a given app is no long authorized (invalidating token and token_secret).

OAuth 2


Google
3 : authenticate
5 : grant access
4 : Do you
grant access?
6 : authorization code
and redirect to app
7 : Exchange auth code
for access token
1 : Could I get access?
2 : Identify yourself
Bob
MyGoogleDrive

OAuth 2

- Latest version of OAuth protocol
    - Similar "dance" required
    - Allows for "bearer tokens" - access is given to anyone able to provide a valid token without any further restrictions or authentication
        - access tokens are provided along with the request for the resource through a secure connection
        - tokens can expire automatically
- We will use both OAuth and OAuth2 over the next few days