Lecture 22

Chapter 16

16.1 A Network Primer

16.2 Writing a Basic Client

16.1 Network primer

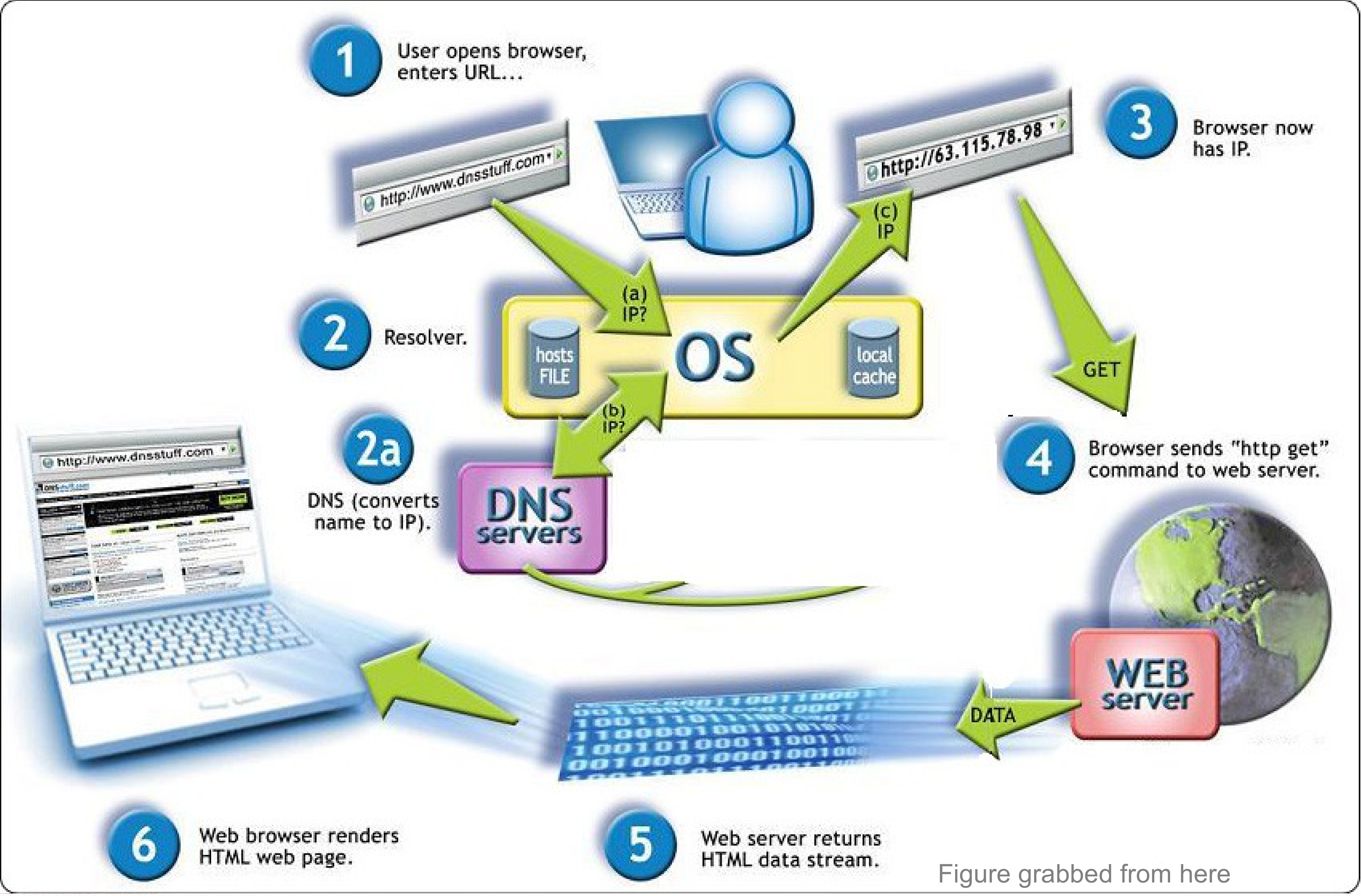
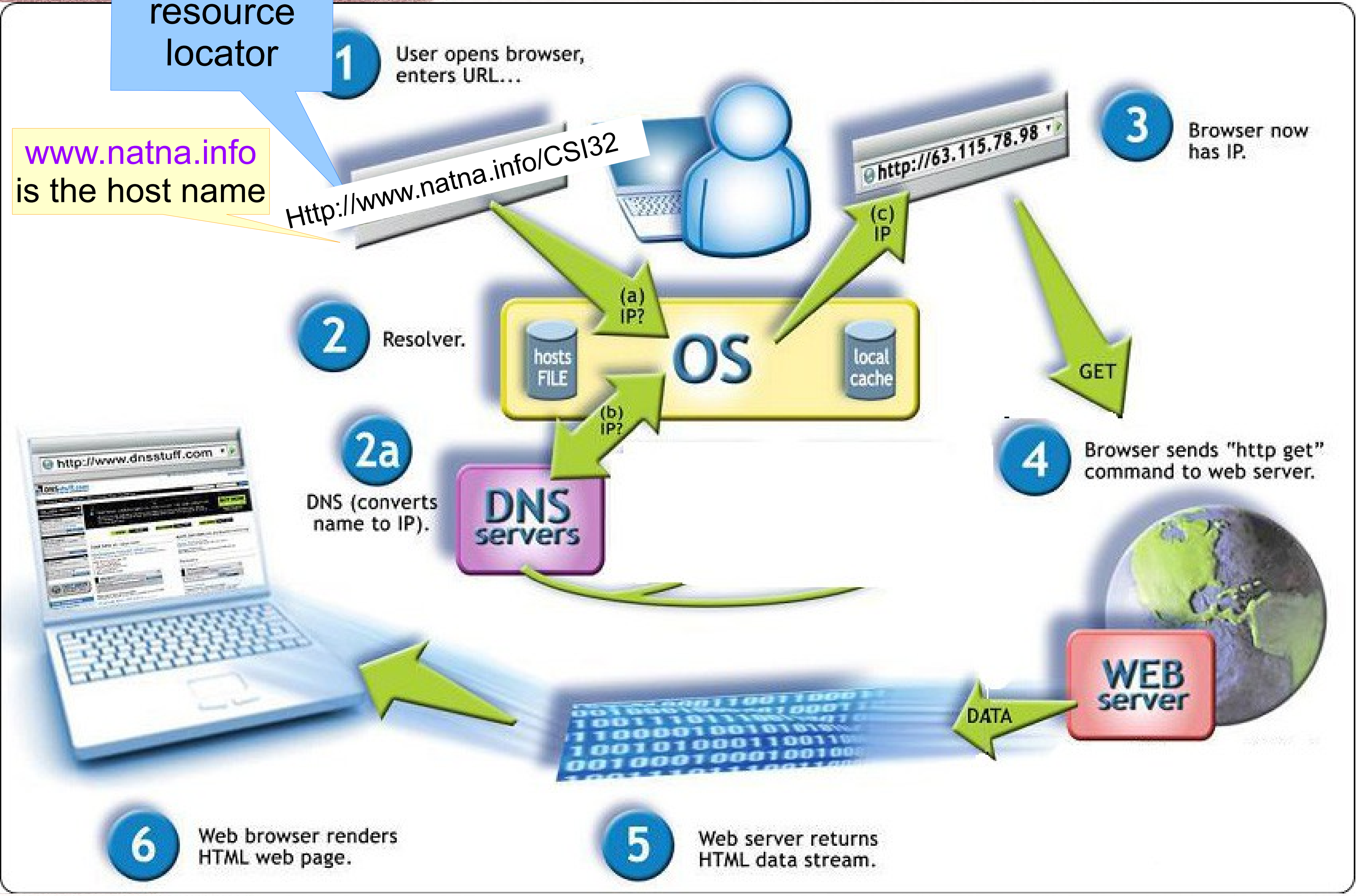

1 User opens browser, enters URL...
http://www.dnsstuff.com
(a) IP?
2 Resolver.
hosts FILE
OS
local cache
(b) IP?
2a DNS (converts name to IP).
DNS servers
(c) IP
http://63.115.78.98
3 Browser now has IP.
GET
4 Browser sends "http get" command to web server.
WEB server
DATA
5 Web server returns HTML data stream.
6 Web browser renders HTML web page.

Figure grabbed from here

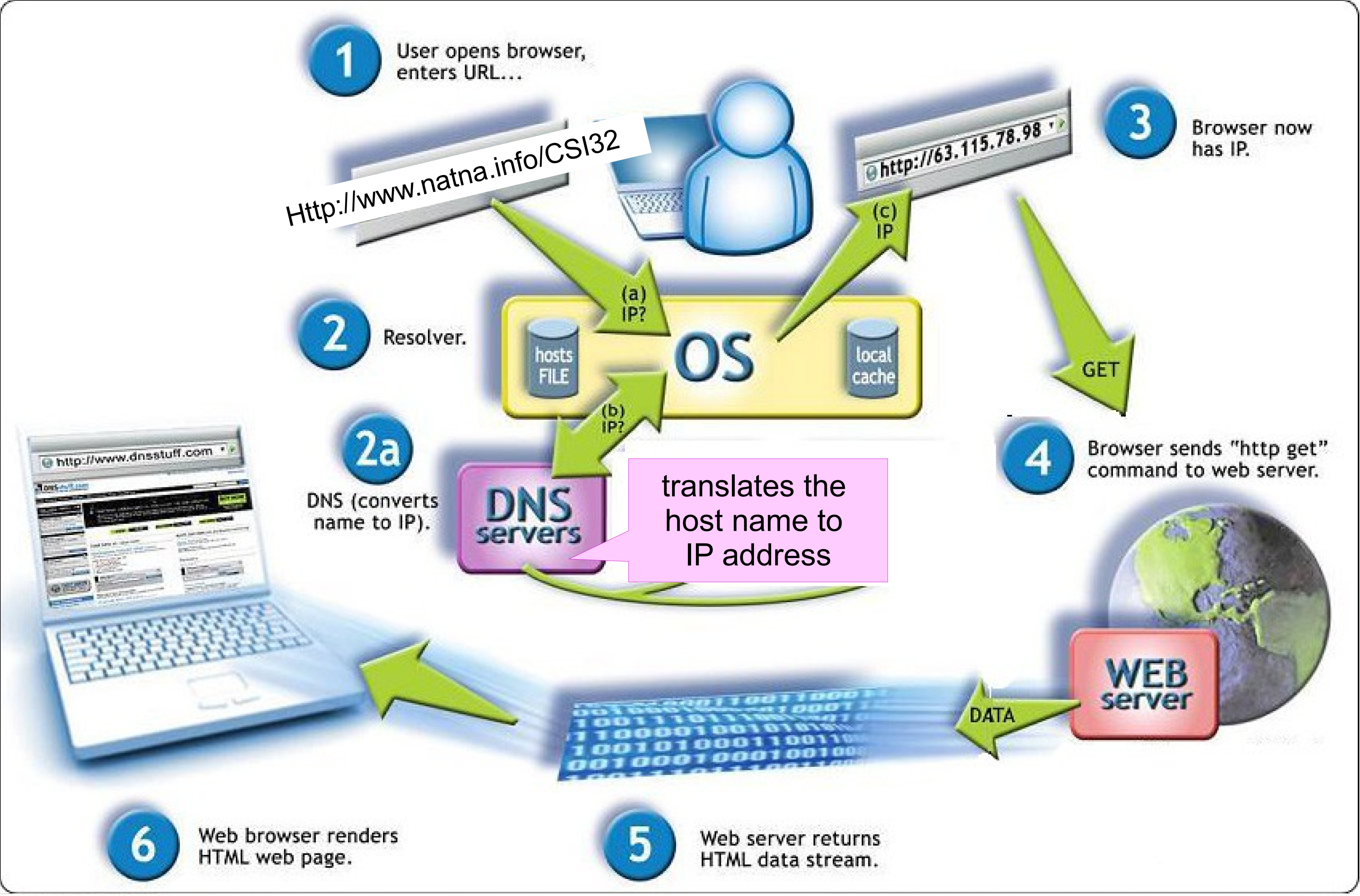
16.1 Network primer

uniform resource locator

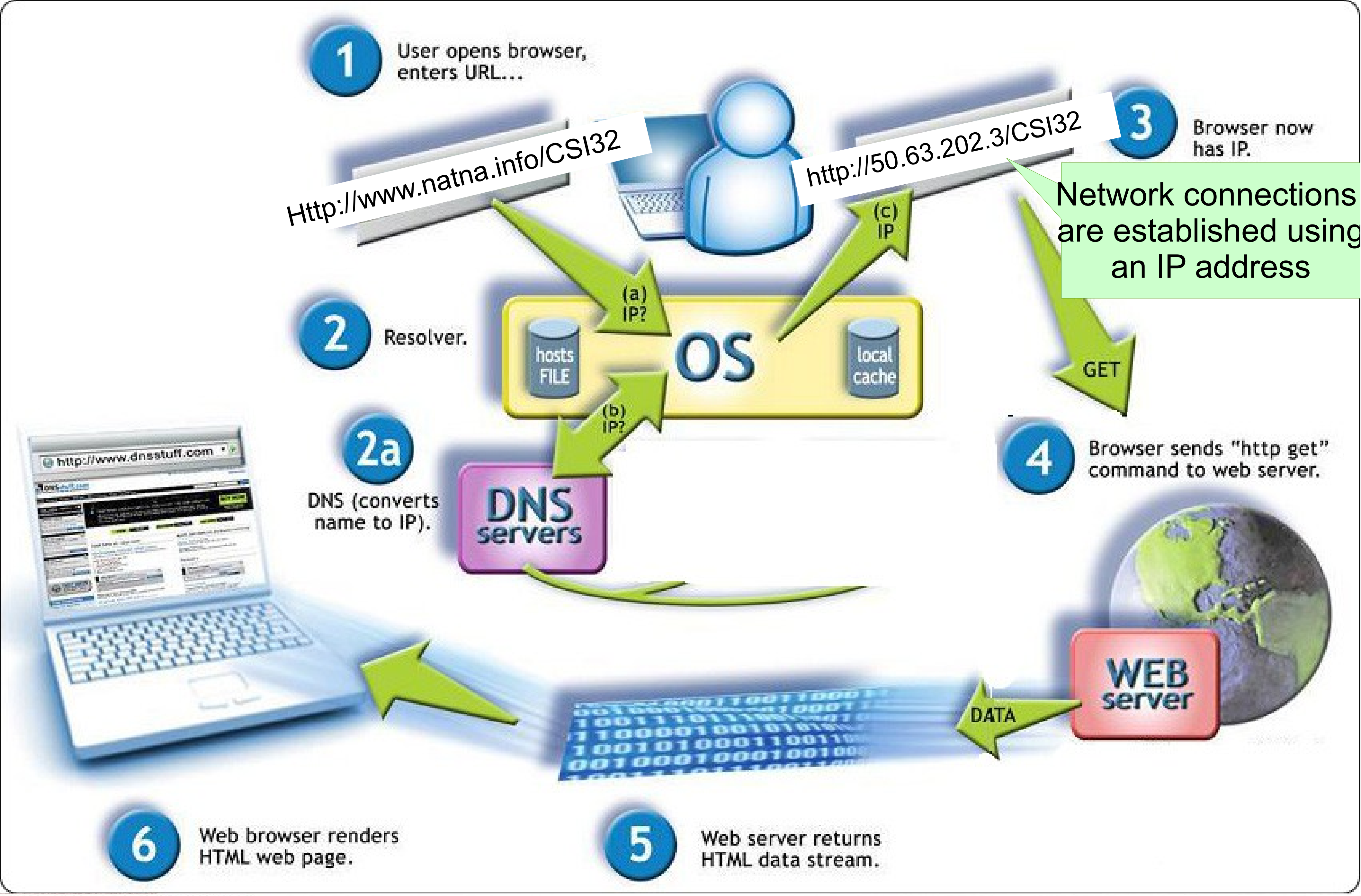
www.natna.info is the host name


Http://www.natna.info/CSI32
1 User opens browser, enters URL...
http://63.115.78.98
3 Browser now has IP.
(a) IP?
(c) IP
2 Resolver.
hosts FILE
OS
local cache
(b) IP?
GET
2a DNS (converts name to IP).
DNS servers
http://www.dnsstuff.com
4 Browser sends "http get" command to web server.
WEB server
DATA
6 Web browser renders HTML web page.
5 Web server returns HTML data stream.

16.1 Network primer


1 User opens browser, enters URL...
Http://www.natna.info/CSI32
(a) IP?
2 Resolver.
hosts FILE
OS
local cache
(b) IP?
2a DNS (converts name to IP).
DNS servers
translates the host name to IP address
(c) IP
http://63.115.78.98
3 Browser now has IP.
GET
4 Browser sends "http get" command to web server.
WEB server
DATA
5 Web server returns HTML data stream.
6 Web browser renders HTML web page.
http://www.dnsstuff.com

16.1 Network primer


1
User opens browser, enters URL...
Http://www.natna.info/CSI32
(a) IP?
2
Resolver.
hosts FILE
OS
local cache
(b) IP?
2a
DNS (converts name to IP).
DNS servers
http://www.dnsstuff.com
(c) IP
3
http://50.63.202.3/CSI32
Browser now has IP.
Network connections are established using an IP address
GET
4
Browser sends "http get" command to web server.
WEB server
DATA
5
Web server returns HTML data stream.
6
Web browser renders HTML web page.

IP address

Internet Protocol Address

________ • ________ • ________ • ________

0 – 255 0 – 255 0 – 255 0 – 255

1 byte 1 byte 1 byte 1 bytes = 4 bytes = 32 bits

256 = 2^8 (256 values possible: from 0 to 255)

IP address

The designers of the *Internet Protocol* defined an IP address as a 32-bit number and this system, known as Internet Protocol Version 4 (IPv4), is still in use today.

However, because of the growth of the Internet and the predicted depletion of available addresses, a new version of IP (IPv6), using 128 bits for the address, was developed in 1995. IPv6 was standardized in 1998, and its deployment has been ongoing since the mid-2000s.

IP addresses:
172.16.254.1 (IPv4), and
2001:db8:0:1234:0:567:8:1 (IPv6).

The Internet *Assigned Numbers Authority* (IANA) manages the IP address space allocations globally and delegates five *regional Internet registries* (RIRs) to allocate IP address blocks to *local Internet registries* (Internet service providers) and other entities.

HTTP

http portion of the URL denotes a particular network protocol that will be used for the transmission of information.

http stands for *Hypertext Transfer Protocol*

The protocol specifies a convention for the data transmission: sender and receiver must have an agreement as to the particular format of data transmission

Port

Different kinds of information can be transmitted:
- requesting web-page
- sending an e-mail
- streaming live video

To help quickly segment incoming network traffic according to a protocol, each network connection to a remote machine must specify a **port**.

Port: 0 – 65535   (note that $65{,}536=2^{16}$)

There are conventions for using certain port numbers for certain types of activity.

Queries to a web server are typically send through port **80**.

See more:
https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_number

Network socket

For a connection between two machines, a separate socket is maintained at each end, used to track the incoming and outgoing data.

Example:

```
from socket import socket
s = socket()
s.connect((“www.google.com”,80))

print(s.recv(1024))
```

- there is no reply at all

The call to recv never returns, because the server doesn't bother to send further information unless we send it a specific query..

Time server

The National Institute of Science and Technology (NIST) runs servers that, when queried, report the current time.

These servers are connected to atomic clocks so their time is precise (modulo network delay)

The time is reported in universal time (UTC) – timezone of Greenwich, England.

This is how the report looks like:

```
b'\n57351 15-11-25 18:17:10 00 0 0 650.8 UTC(NIST) * \n'
```

See the program gettime.py

Downloading a web-page

Let's develop a client program that will download and save a web-page.

HTTP 1.0 specification supports GET, POST and HEAD methods
HTTP 1.1 specification supports 5 more methods: OPTIONS, PUT, DELETE, TRACE and CONNECT

The GET method requests a representation of the specified resource.

Requests using GET should only retrieve data and should have no other effect.

The precise data sent back by the server contains not only the *contents of the web page*, but additional header information such as the *file type*, *modification time*, and *length of the file*.

Downloading a web-page

The precise data sent back by the server contains not only the *contents of the web page*, but additional header information such as the *file type*, *modification time*, and *length of the file*.

```
HTTP/1.1 404 Not Found
Date: Sat, 26 Nov 2016 14:29:13 GMT
Server: Apache/2.4.23
Accept-Ranges: bytes
Vary: Accept-Encoding,User-Agent
Content-Length: 1699
Connection: close
Content-Type: text/html


<!DOCTYPE html>
<html>
<head>
<title>File Not Found</title>
```

Downloading a web-page

We will use header information to get the length of the file's content.

see the program webclient.py

Downloading a web-page

The program webclient.py shows low-level steps to download a web-page, but since this task is such a common practice, Python provides a high-level way to perform it, using urllib.request module.

In particular, it has a method urlopen that takes string (URL) as parameter and returns a file-like object (type **string**) from which we can read the content of the file.

see the program easywebclient.py