open.michigan

You assume all responsibility for use and potential liability associated with any use of the material. Material contains copyrighted content, used in accordance with U.S. law. Copyright holders of content included in this material should contact open.michigan@umich.edu with any questions, corrections, or clarifications regarding the use of content. The Regents of the University of Michigan do not license the use of third party content posted to this site unless such a license is specifically granted in connection with particular content. Users of content are responsible for their compliance with applicable law. Mention of specific products in this material solely represents the opinion of the speaker and does not represent an endorsement by the University of Michigan. For more information about how to cite these materials visit http://michigan.educommons.net/about/terms-of-use.

Any medical information in this material is intended to inform and educate and is not a tool for self-diagnosis or a replacement for medical evaluation, advice, diagnosis or treatment by a healthcare professional. You should speak to your physician or make an appointment to be seen if you have questions or concerns about this information or your medical condition. Viewer discretion is advised: Material may contain medical images that may be disturbing to some viewers.

UNIVERSITY OF MICHIGAN

cc BY

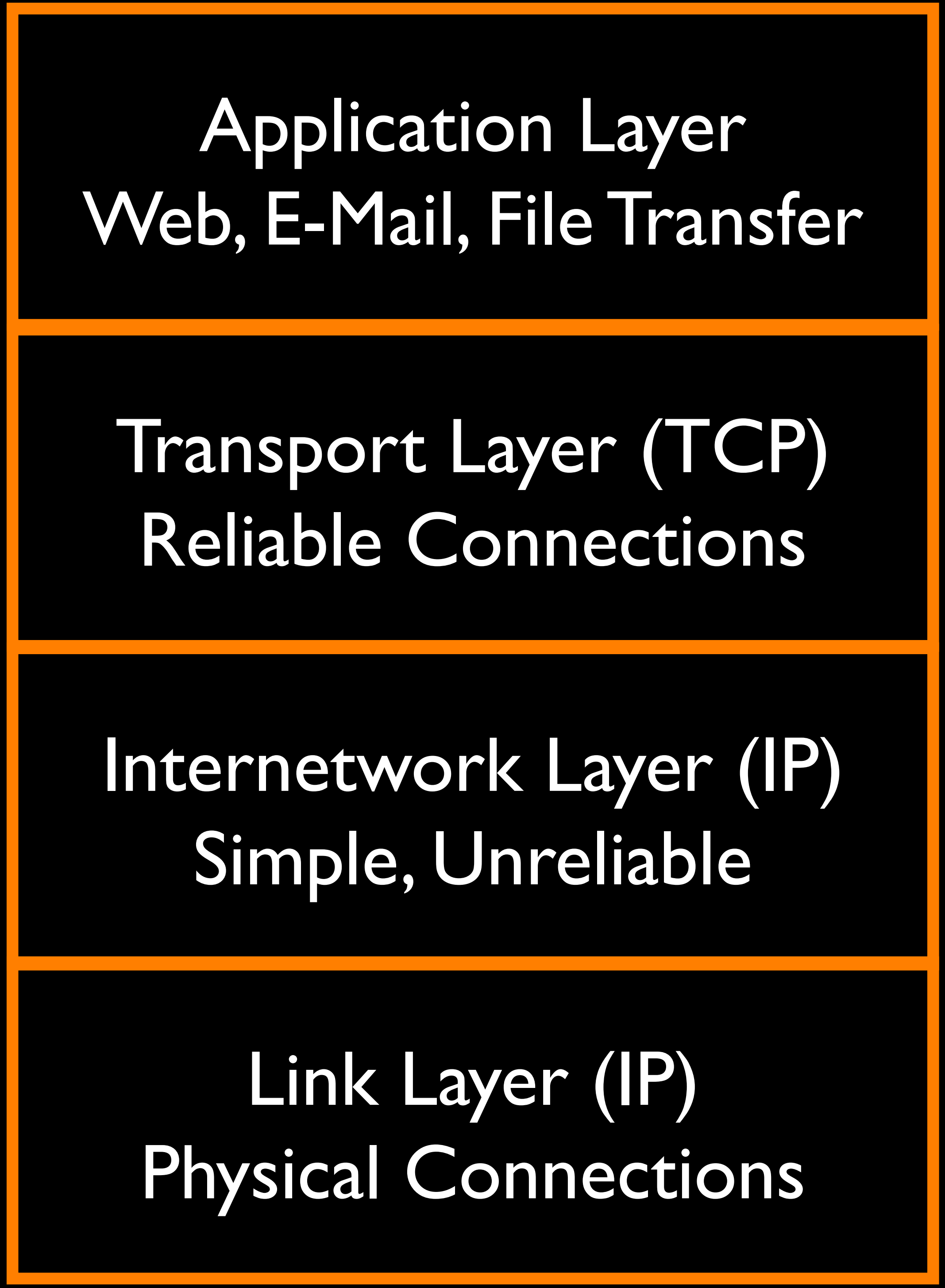
Networking Part 2

Charles Severance

Review...

Layered Network Model

- A layered approach allows the problem of implementing a network to be broken into more manageable sub problems
- For example the IP layer is allowed to lose a packet if things go bad
- It is TCP’s Responsibility to store and retransmit data.



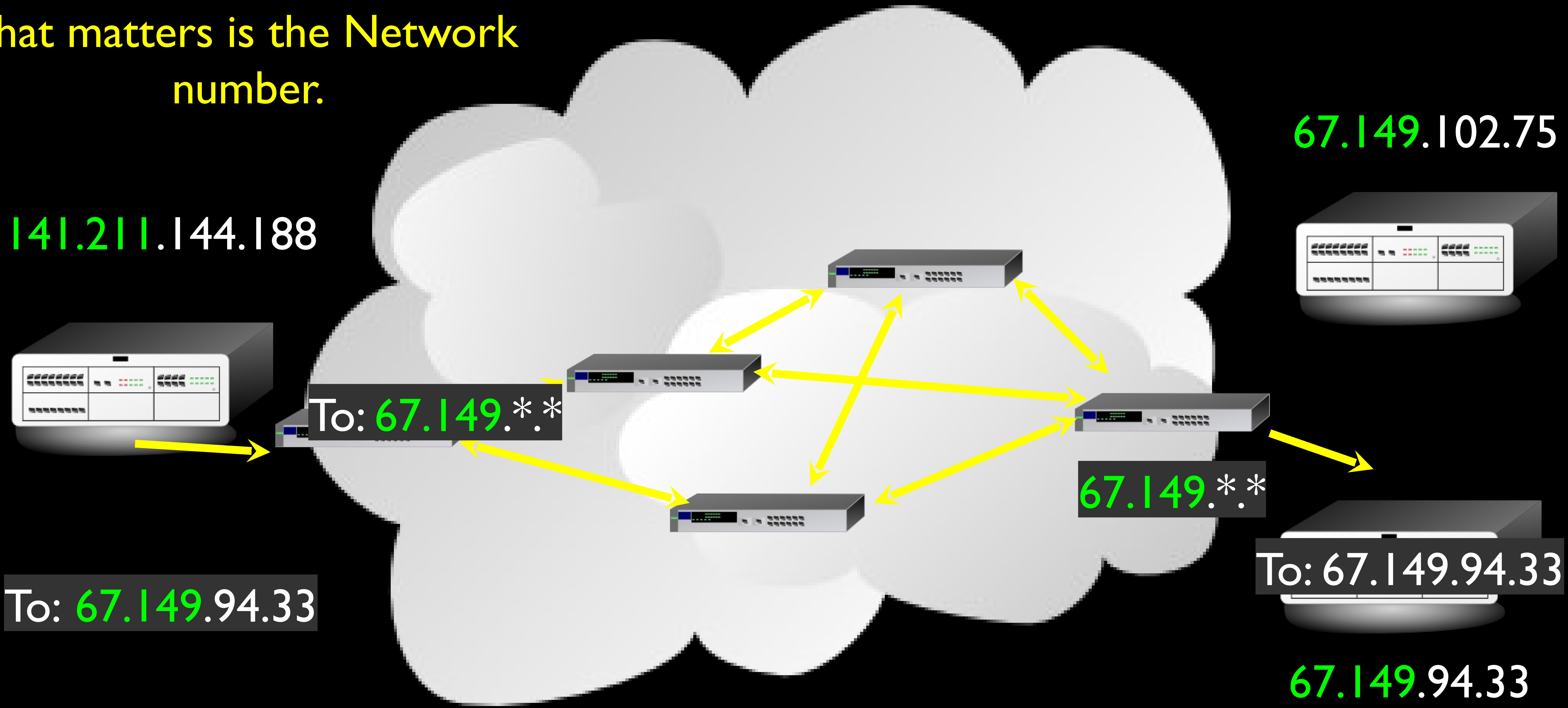
Application Layer
Web, E-Mail, File Transfer

Transport Layer (TCP)
Reliable Connections

Internetwork Layer (IP)
Simple, Unreliable

Link Layer (IP)
Physical Connections

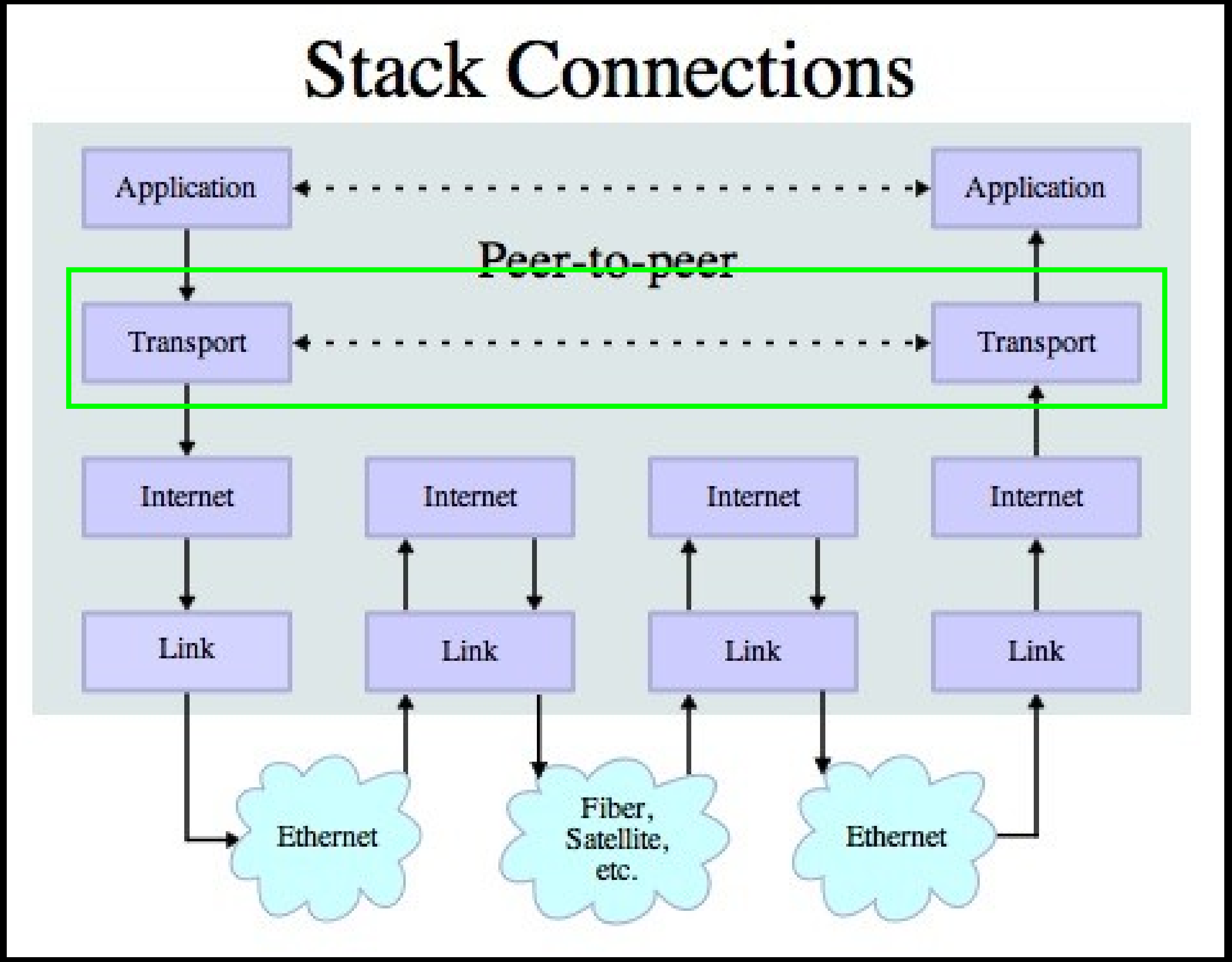
While in the network, all that matters is the Network number.



Clipart: http://www.clker.com/search/networksym/1

Transport Protocol (TCP)

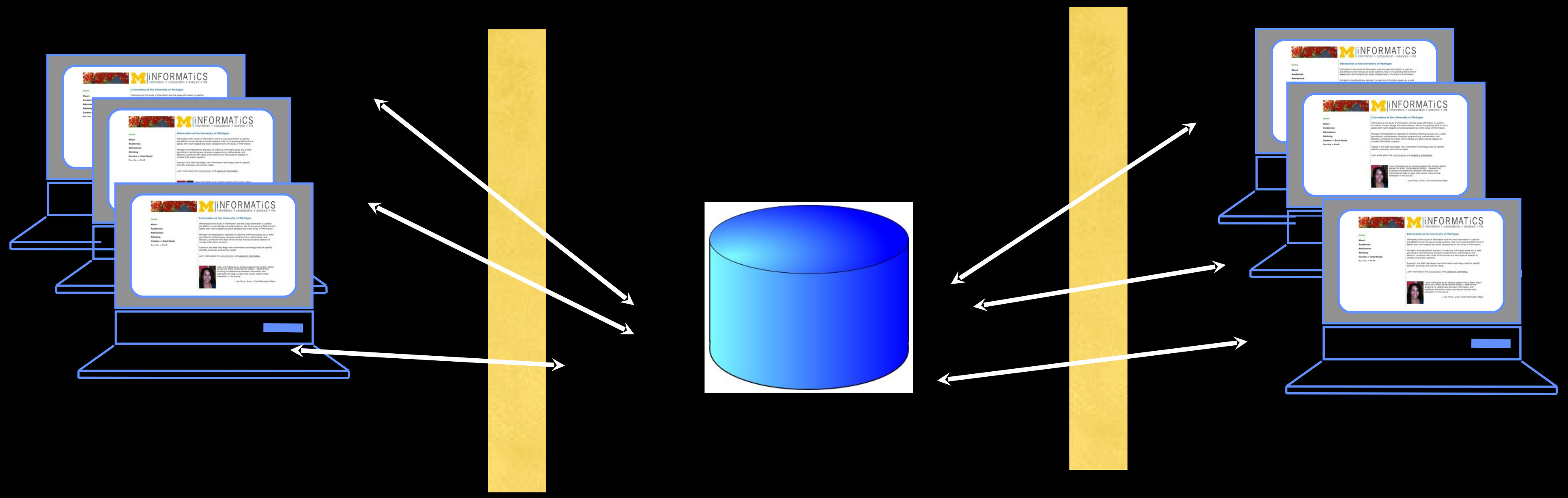
- Built on top of IP
- Assumes IP might lose some data
- In case data gets lost - we keep a copy of the data a we send until we get an acknowledgement
- If it takes "too long" - just send it again



Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite

System to System IP

- Regardless of the number of connections between two systems, the traffic is transported across the internet as a single IP address - It is the responsibility of TCP to separate (de-multiplex) each stream on each system



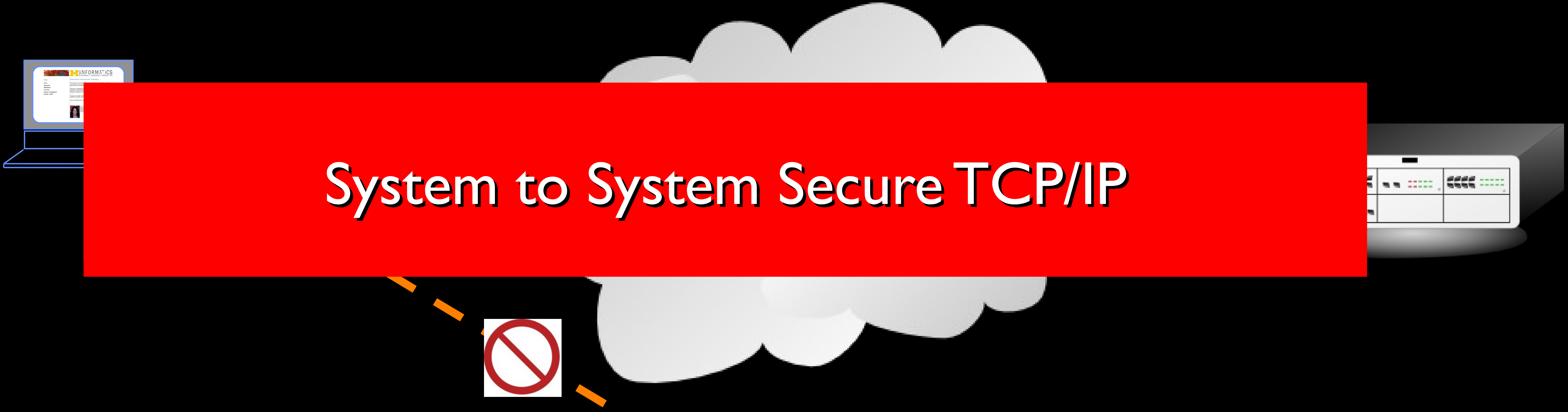
Transport Protocol (TCP)

- The responsibility of the transport layer is to present a reliable end-to-end pipe to the application
- Data either arrives in the proper order or the connection is closed
- TCP keeps buffers in the sending and destination system to keep data which has arrived out of order or to retransmit if necessary
- TCP provides individual connections between applications

Security for TCP

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

System to System Secure TCP/IP



Your local connection (particularly when wireless) is your greatest exposure.

Generally, the backbone of the Internet is pretty secure to prying eyes from generic baddies...

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

Clipart: http://www.clker.com/search/networksym/1
Photo CC BY: karindalziel (flickr)
http://creativecommons.org/licenses/by/2.0/

Secure Sockets
Transport Layer Security (TLS)

- When Secure Sockets Layer (SSL) is used, all of the data in the TCP is encrypted before it leaves your machine and decrypted in the destination machine

- It is very difficult but not impossible to break this security - normal people do not have the necessary compute resources to break TLS

- Encrypting sockets takes resources - so we use it for things when it is needed

- The IP and link layers are unaware when the contents of a TCP connections are encrypted (Abstraction)

Secure Sockets

- SSL is best thought of as a "sub" layer
- Like a thin shim between the application layer and transport layer
- Hides data from prying eyes

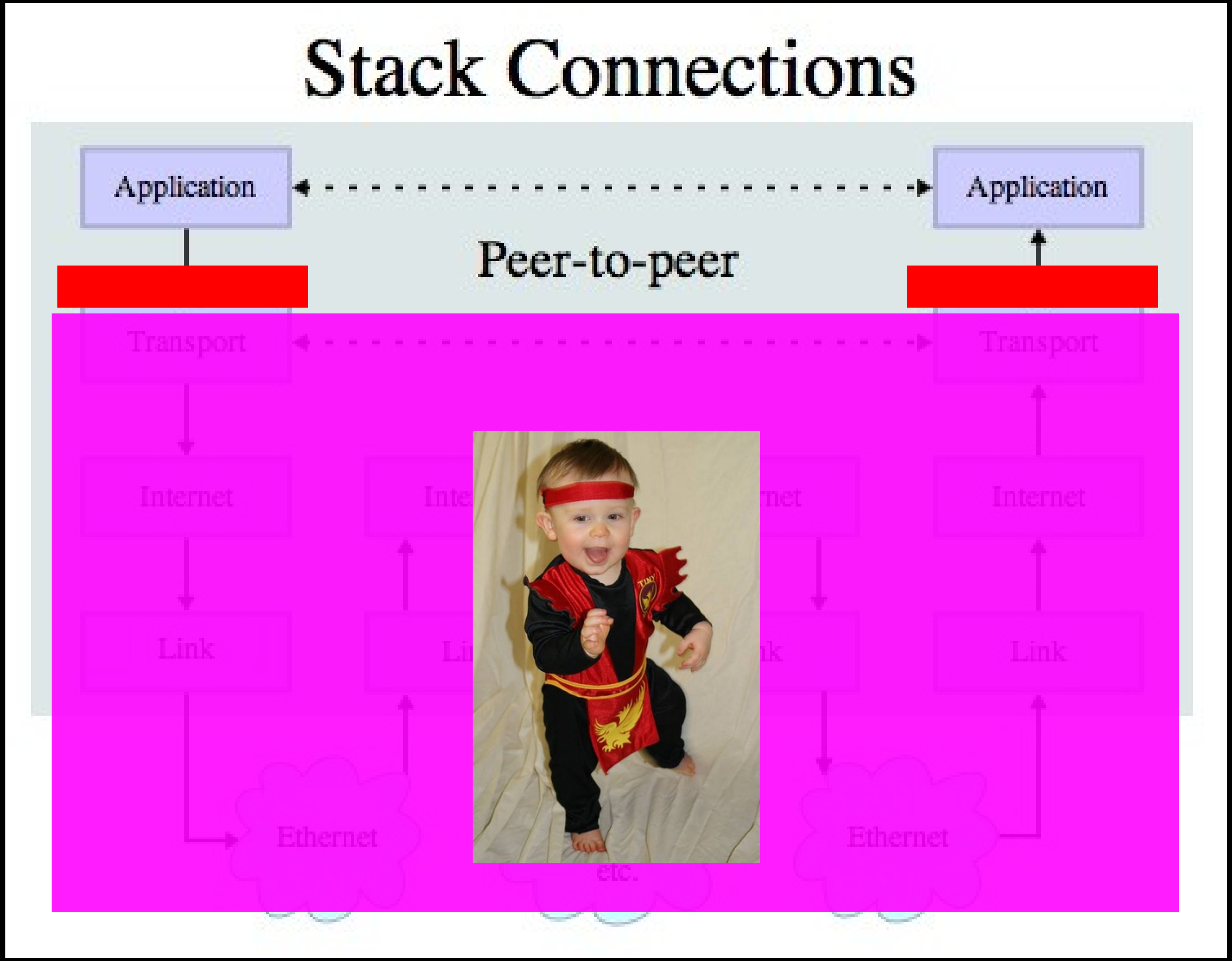
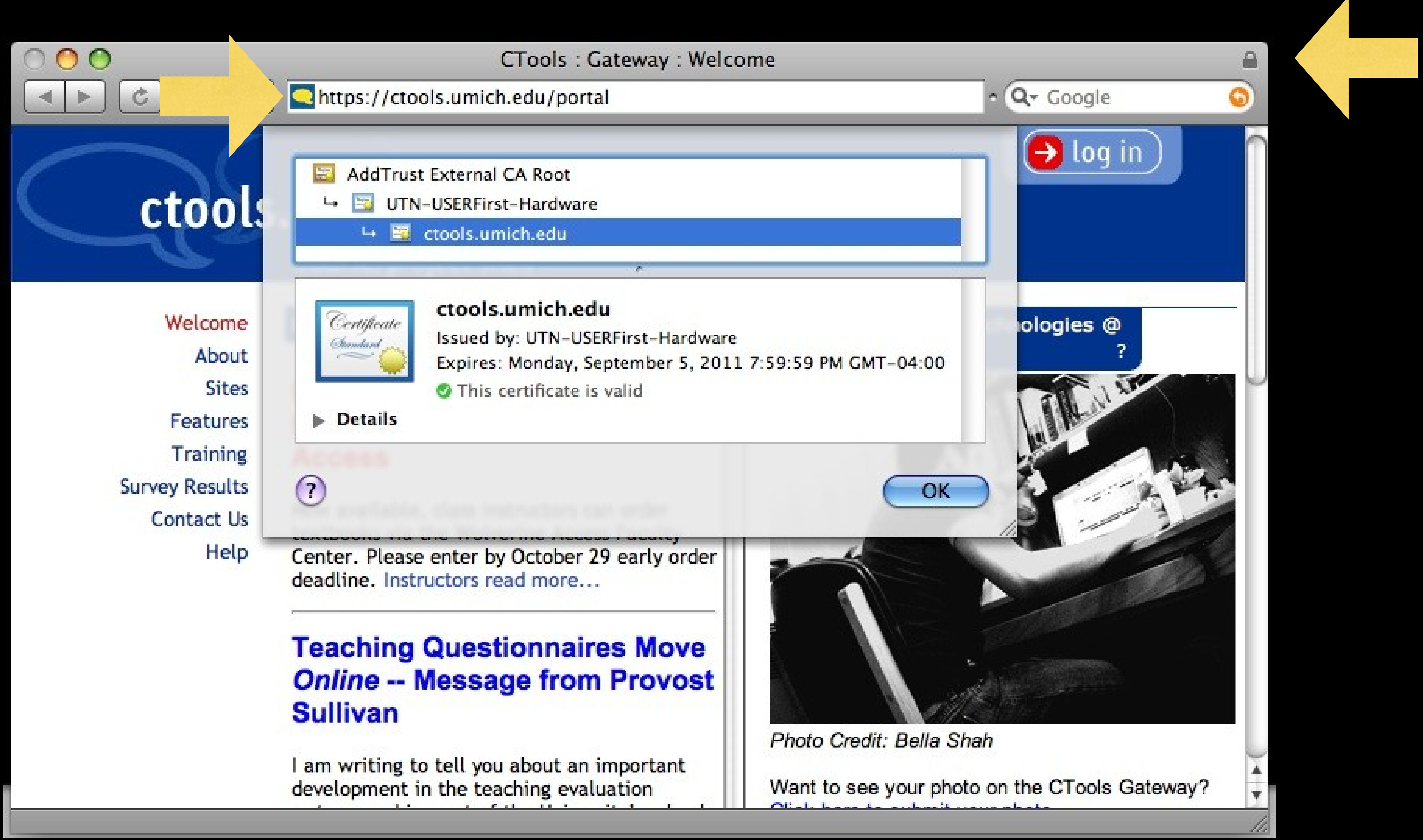


Chart: http://en.wikipedia.org/wiki/Internet_Protocol_Suite
Photo CC BY: karindalziel (flickr)
http://creativecommons.org/licenses/by/2.0/

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

Secure Application Protocols

- There are often secure and unencrypted application protocols
    - http://ctools.umich.edu
    - https://ctools.umich.edu
- Your browser tells you when using a secure connection - you should never type passwords into a non-secure connection
- Especially over wireless - especially at a security conference...



CTools : Gateway : Welcome

https://ctools.umich.edu/portal

AddTrust External CA Root
↳ UTN-USERFirst-Hardware
↳ ctools.umich.edu

ctools.umich.edu
Issued by: UTN-USERFirst-Hardware
Expires: Monday, September 5, 2011 7:59:59 PM GMT-04:00
✓ This certificate is valid

▶ Details

Center. Please enter by October 29 early order deadline. Instructors read more...

Teaching Questionnaires Move *Online* -- Message from Provost Sullivan

I am writing to tell you about an important development in the teaching evaluation

Photo Credit: Bella Shah

Want to see your photo on the CTools Gateway?

Source: https://ctools.umich.edu/portal

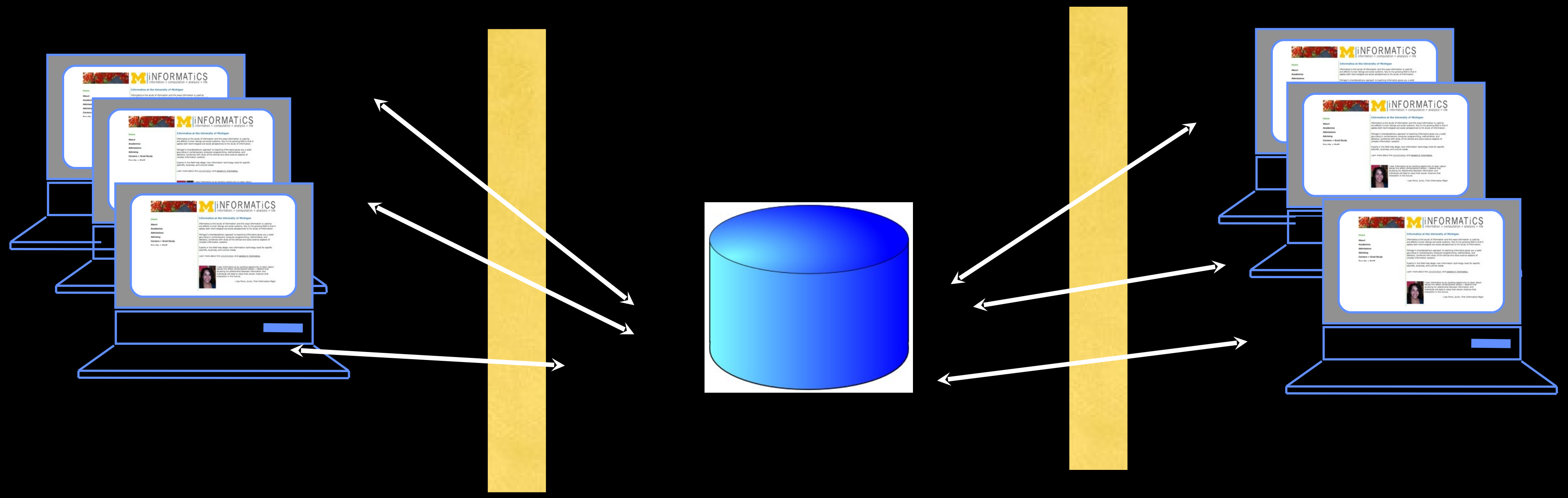
TCP, Ports, and Connections

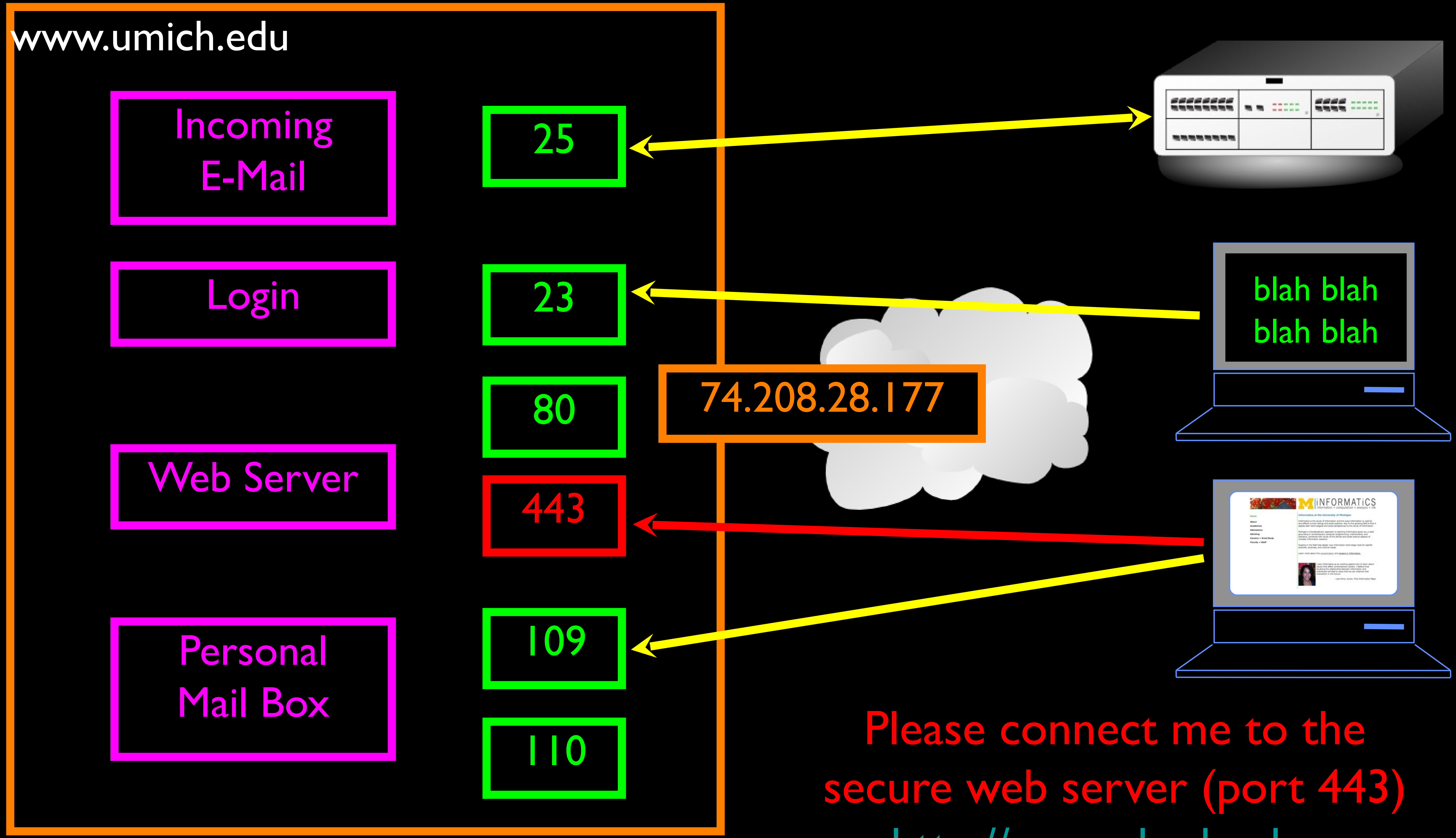
http://en.wikipedia.org/wiki/TCP_and_UDP_port

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

System to System IP

- Regardless of the number of connections between two systems, the traffic is transported across the internet as a single IP address - It is the responsibility of TCP to separate (de-multiplex) each stream on each system





www.umich.edu

Incoming E-Mail

Login

Web Server

Personal Mail Box

25

23

80

443

109

110

74.208.28.177

blah blah blah blah

Please connect me to the secure web server (port 443) on http://www.dr-chuck.com

Clipart: http://www.clker.com/search/networksym/1

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

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

Application Protocols

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

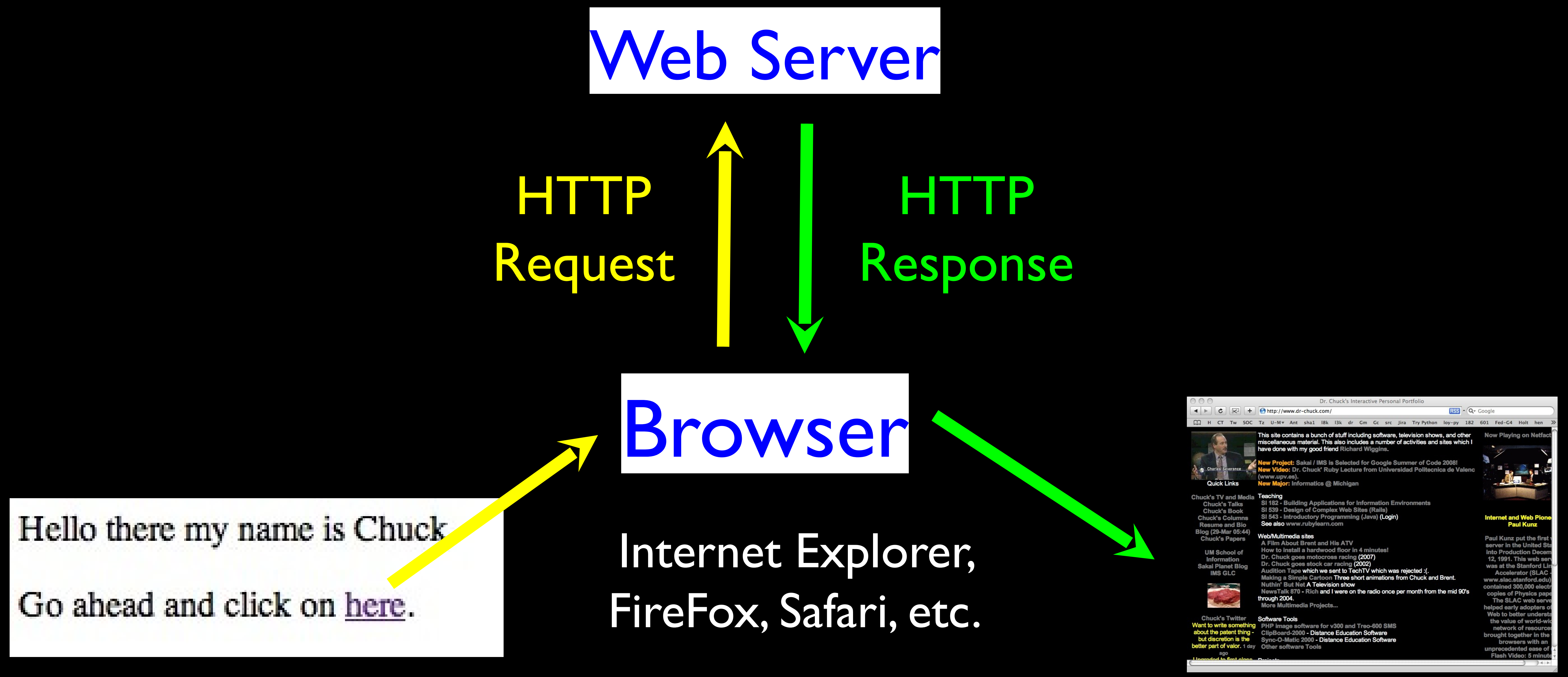
http://en.wikipedia.org/wiki/Pop3

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

HTTP Request / Response Cycle



Web Server

HTTP Request

HTTP Response

Browser

Hello there my name is Chuck

Go ahead and click on here.

Internet Explorer, FireFox, Safari, etc.

http://www.oreilly.com/openbook/cgi/ch04_02.html

Source: http://www.dr-chuck.com/

HTTP Request / Response Cycle

Web Server

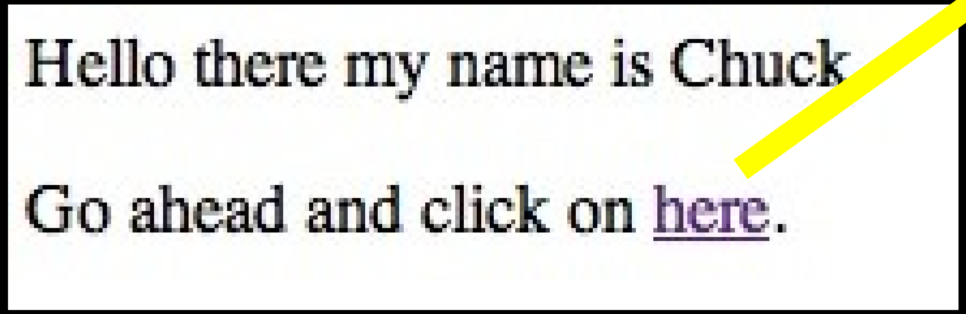
HTTP Request

GET /index.html

HTTP Response

```
<head> .. </head>
<body>
<h1>Welcome to my
   application</h1>
 ....
</body>
```

Browser

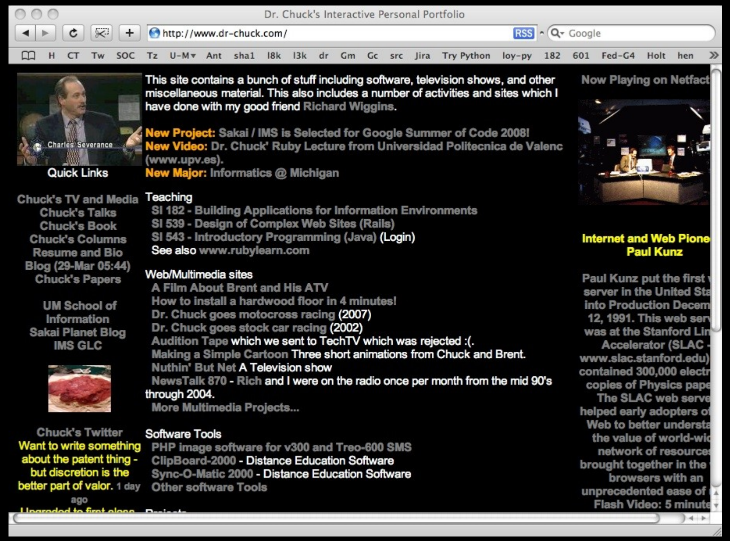


Hello there my name is Chuck

Go ahead and click on here.

Internet Explorer, FireFox, Safari, etc.

http://www.oreilly.com/openbook/cgi/ch04_02.html


Dr. Chuck's Interactive Personal Portfolio
http://www.dr-chuck.com/
RSS
Google
H CT Tw SOC Tz U-M Ant sha1 I8k I3k dr Gm Gc src Jira Try Python loy-py 182 601 Fed-G4 Holt hen
Charles Severance

This site contains a bunch of stuff including software, television shows, and other miscellaneous material. This also includes a number of activities and sites which I have done with my good friend Richard Wiggins.

New Project: Sakai / IMS is Selected for Google Summer of Code 2008!
New Video: Dr. Chuck' Ruby Lecture from Universidad Politecnica de Valenc (www.upv.es).
New Major: Informatics @ Michigan

Quick Links

Chuck's TV and Media
Chuck's Talks
Chuck's Book
Chuck's Columns
Resume and Bio
Blog (29-Mar 05:44)
Chuck's Papers

UM School of Information
Sakai Planet Blog
IMS GLC

Chuck's Twitter
Want to write something about the patent thing - but discretion is the better part of valor. 1 day ago

Teaching
SI 182 - Building Applications for Information Environments
SI 539 - Design of Complex Web Sites (Rails)
SI 543 - Introductory Programming (Java) (Login)
See also www.rubylearn.com

Web/Multimedia sites
A Film About Brent and His ATV
How to install a hardwood floor in 4 minutes!
Dr. Chuck goes motocross racing (2007)
Dr. Chuck goes stock car racing (2002)
Audition Tape which we sent to TechTV which was rejected :(.
Making a Simple Cartoon Three short animations from Chuck and Brent.
Nuthin' But Net A Television show
NewsTalk 870 - Rich and I were on the radio once per month from the mid 90's through 2004.
More Multimedia Projects...

Software Tools
PHP image software for v300 and Treo-600 SMS
ClipBoard-2000 - Distance Education Software
Sync-O-Matic 2000 - Distance Education Software
Other software Tools

Now Playing on Netfact

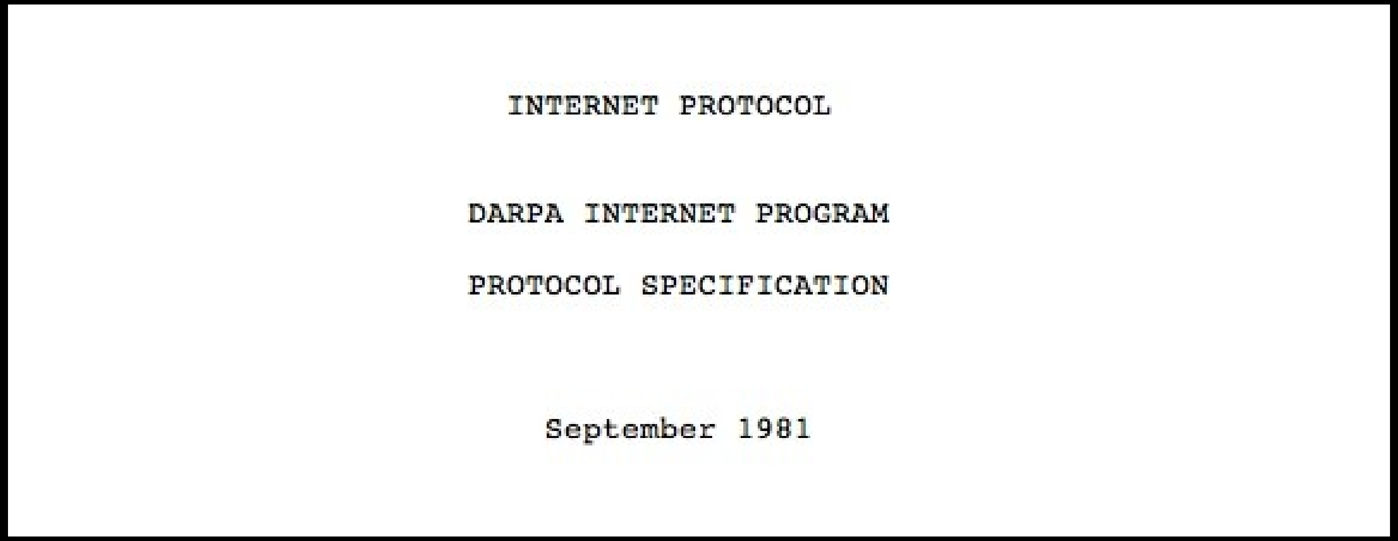
Internet and Web Pione
Paul Kunz

Paul Kunz put the first
server in the United Sta
into Production Decem
12, 1991. This web serv
was at the Stanford Lin
Accelerator (SLAC -
www.slac.stanford.edu)
contained 300,000 electr
copies of Physics pape
The SLAC web serve
helped early adopters of
Web to better understa
the value of world-wid
network of resources
brought together in the
browsers with an
unprecedented ease of
Flash Video: 5 minute

Source: http://www.dr-chuck.com/

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”

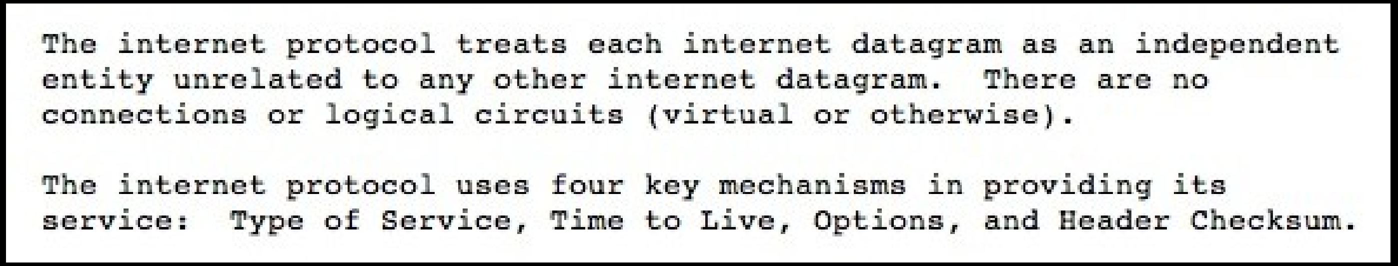


INTERNET PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

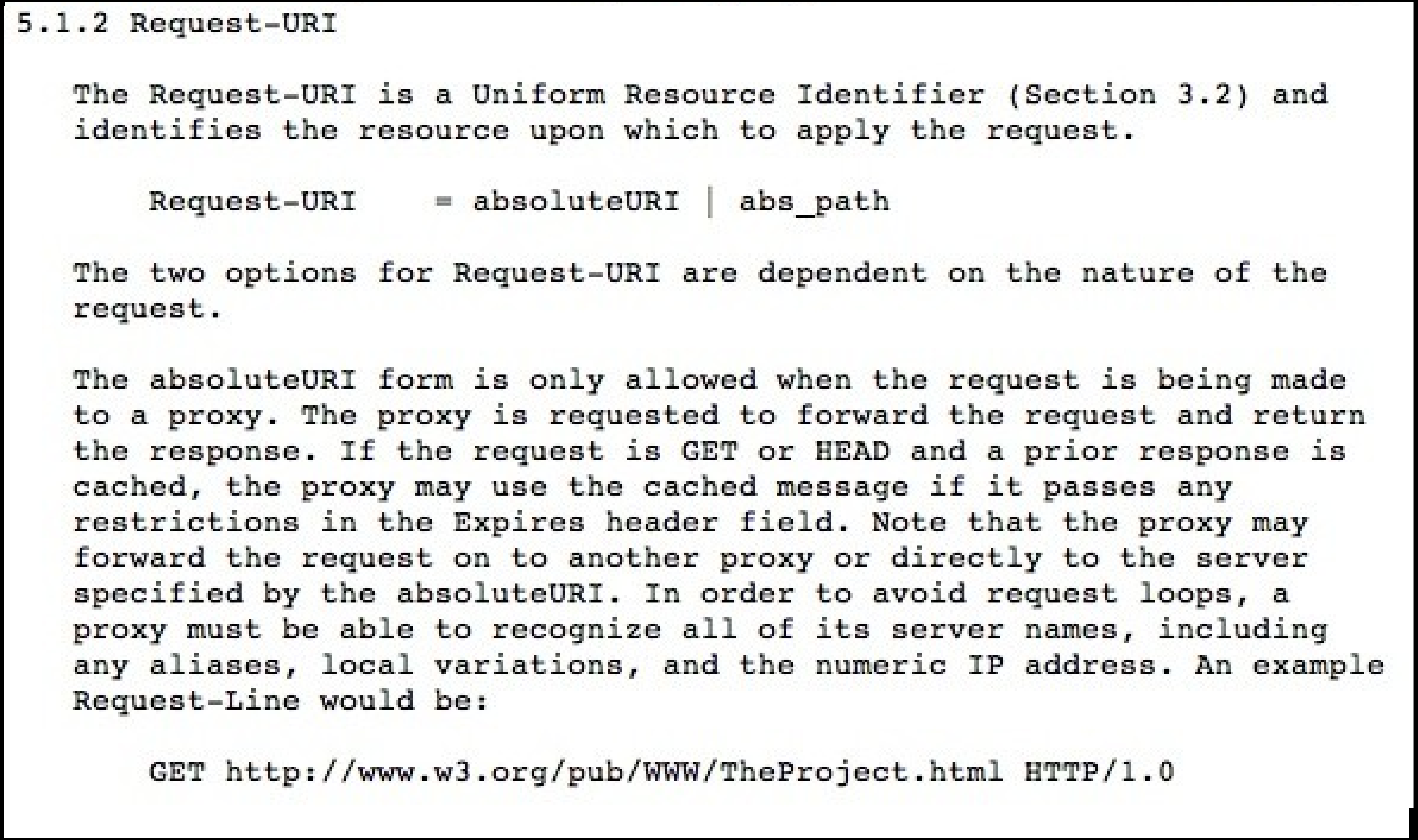
September 1981



The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

Source: http://tools.ietf.org/html/rfc791



5.1.2 Request-URI

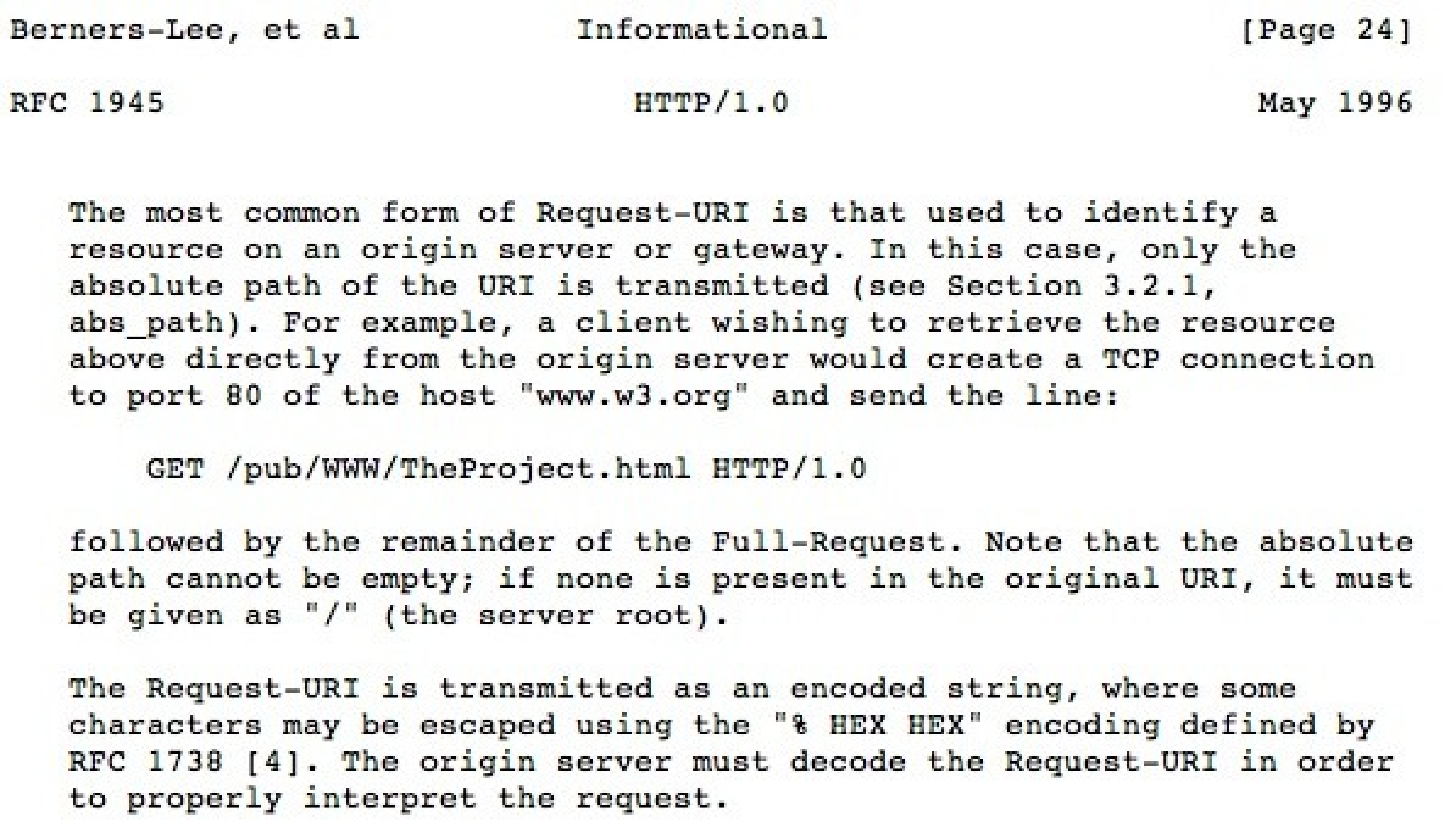
The Request-URI is a Uniform Resource Identifier (Section 3.2) and identifies the resource upon which to apply the request.

```
    Request-URI    = absoluteURI | abs_path
```

The two options for Request-URI are dependent on the nature of the request.

The absoluteURI form is only allowed when the request is being made to a proxy. The proxy is requested to forward the request and return the response. If the request is GET or HEAD and a prior response is cached, the proxy may use the cached message if it passes any restrictions in the Expires header field. Note that the proxy may forward the request on to another proxy or directly to the server specified by the absoluteURI. In order to avoid request loops, a proxy must be able to recognize all of its server names, including any aliases, local variations, and the numeric IP address. An example Request-Line would be:

```
    GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.0
```



Berners-Lee, et al Informational [Page 24]

RFC 1945 HTTP/1.0 May 1996

The most common form of Request-URI is that used to identify a resource on an origin server or gateway. In this case, only the absolute path of the URI is transmitted (see Section 3.2.1, abs_path). For example, a client wishing to retrieve the resource above directly from the origin server would create a TCP connection to port 80 of the host "www.w3.org" and send the line:

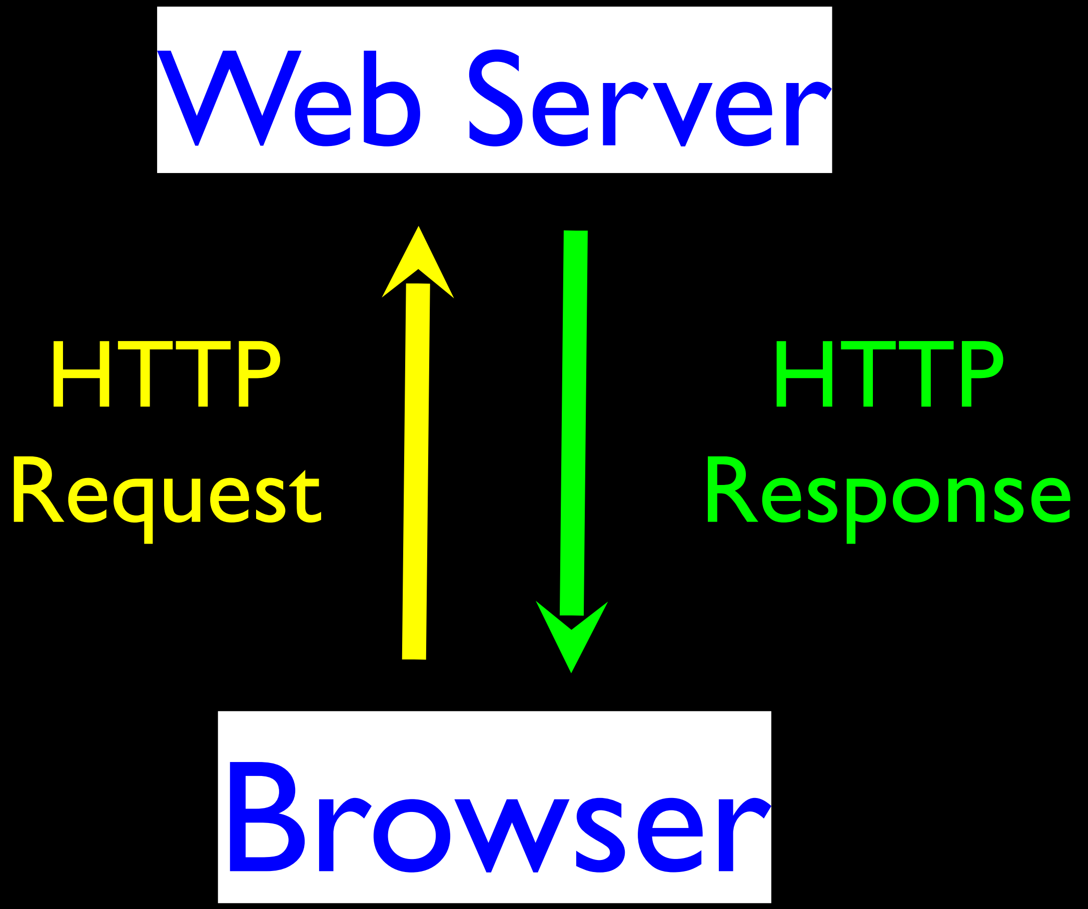
```
GET /pub/WWW/TheProject.html HTTP/1.0
```

followed by the remainder of the Full-Request. Note that the absolute path cannot be empty; if none is present in the original URI, it must be given as "/" (the server root).

The Request-URI is transmitted as an encoded string, where some characters may be escaped using the "% HEX HEX" encoding defined by RFC 1738 [4]. The origin server must decode the Request-URI in order to properly interpret the request.

Source: http://tools.ietf.org/html/rfc791

“Hacking” HTTP



```
Last login: Wed Oct 10 04:20:19 on ttyp2
si-csev-mbp:~ csev$ telnet www.umich.edu 80
Trying 141.211.144.188...
Connected to www.umich.edu.
Escape character is '^]'.
GET /
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
 ....
```

Port 80 is the non-encrypted HTTP port

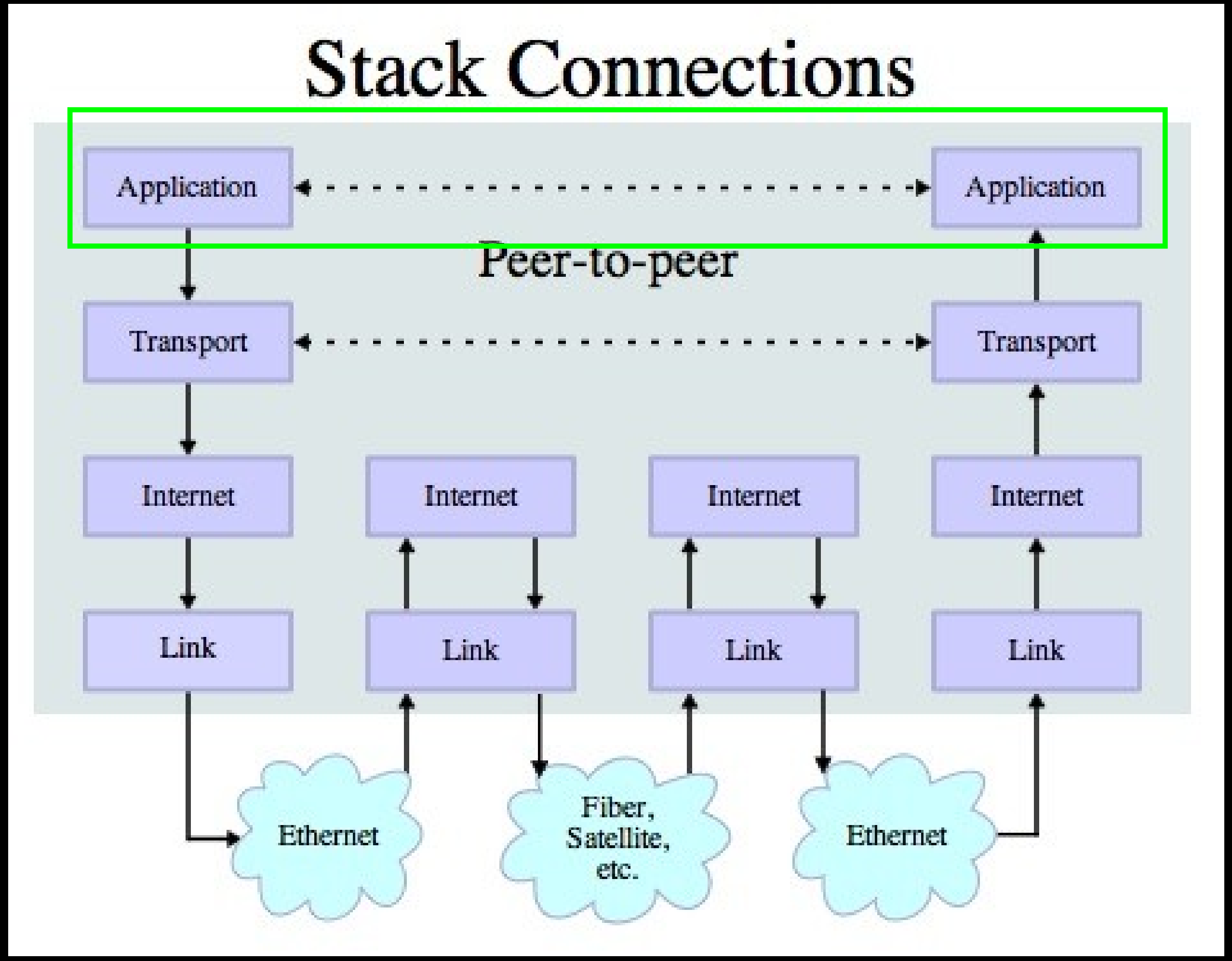
A Bit of Python Network Software

http://en.wikipedia.org/wiki/Abstraction_(computer_science)

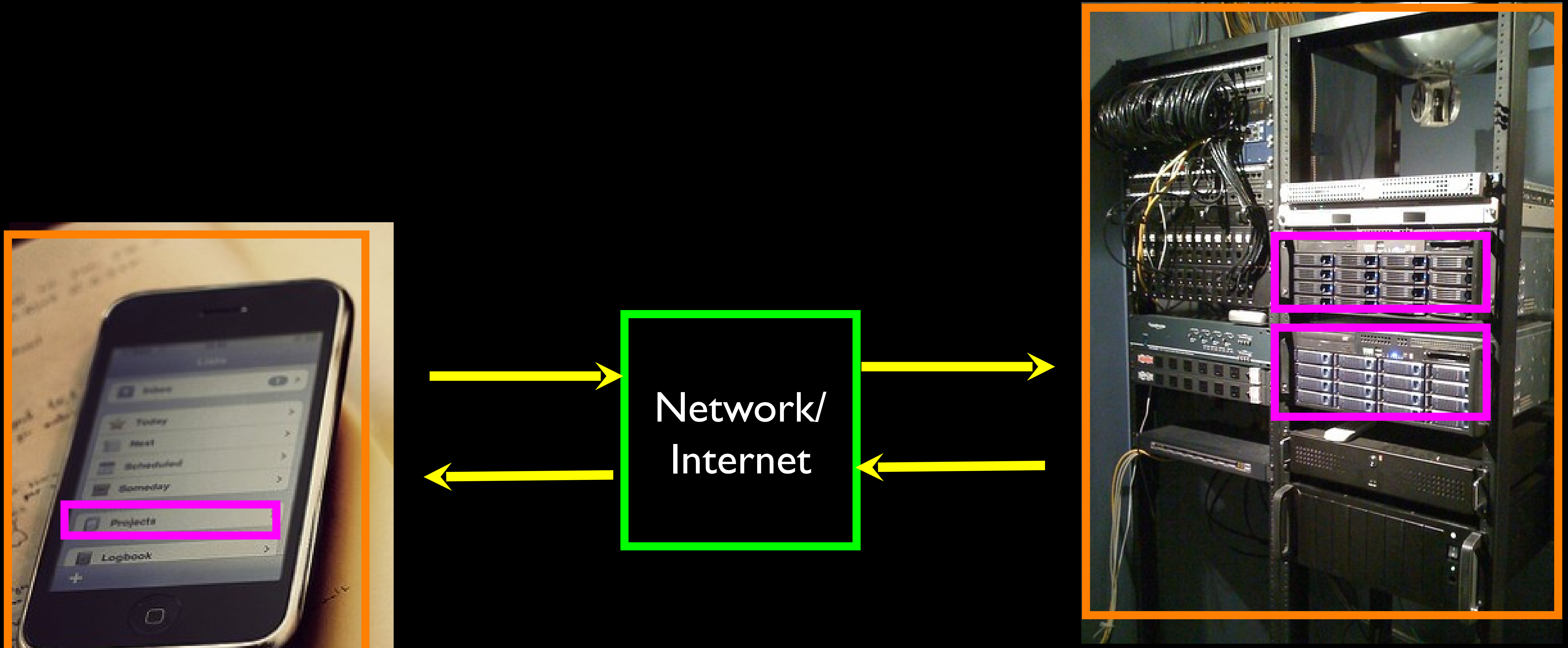
A Simple Web Browser

- We will write a Python application that connects to web server and retrieves the top level page

- Our software will run in a client (our desktop) and talk to a server - far away in the network "cloud"



Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite

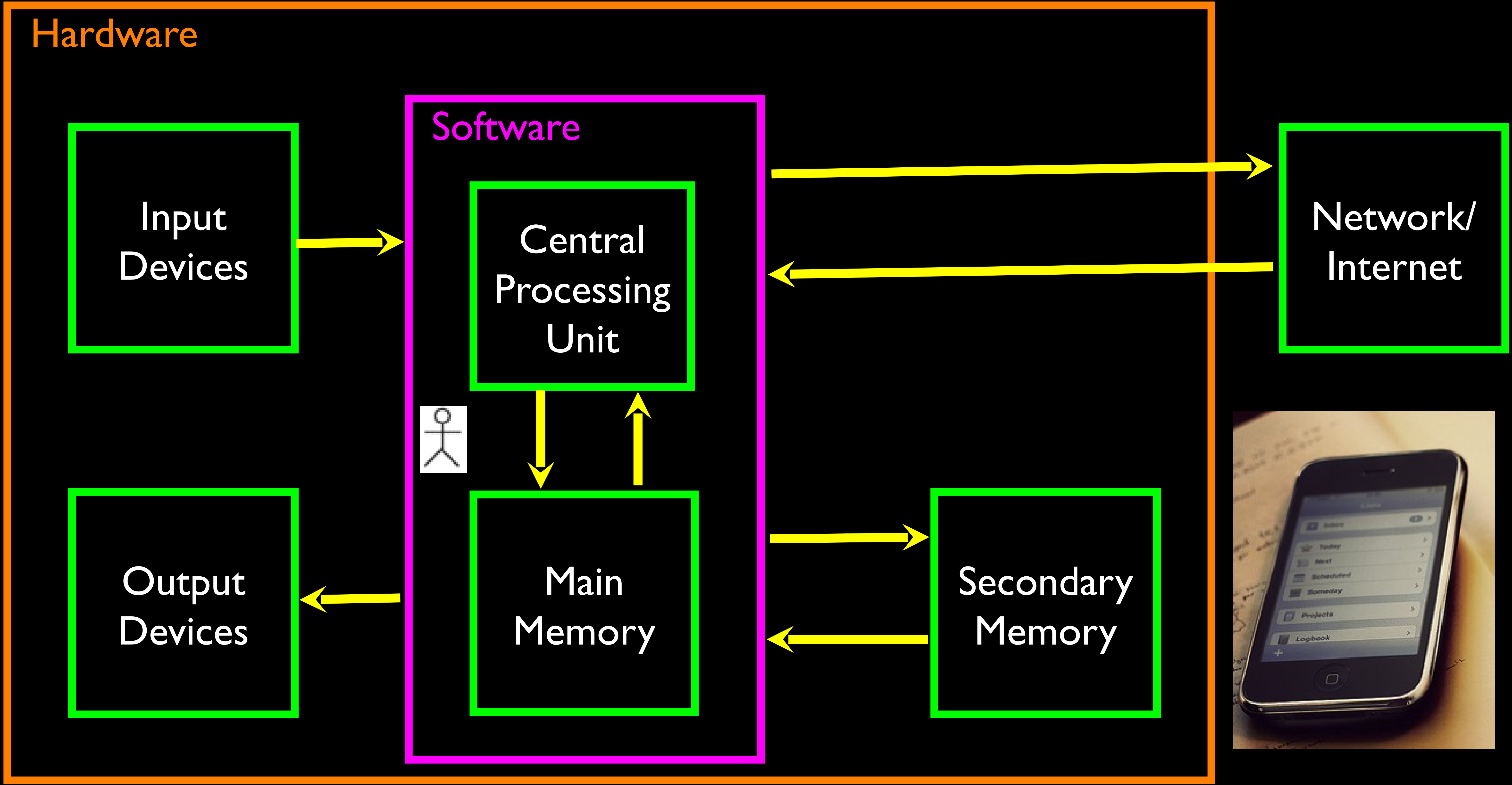


Network/ Internet

Hardware

Software

Phone CC BY: Johan Larsson (flickr)
Servers CC BY: Jesse Wagstaff (flickr)
http://creativecommons.org/licenses/by/2.0/



Hardware

Software

Input Devices

Central Processing Unit

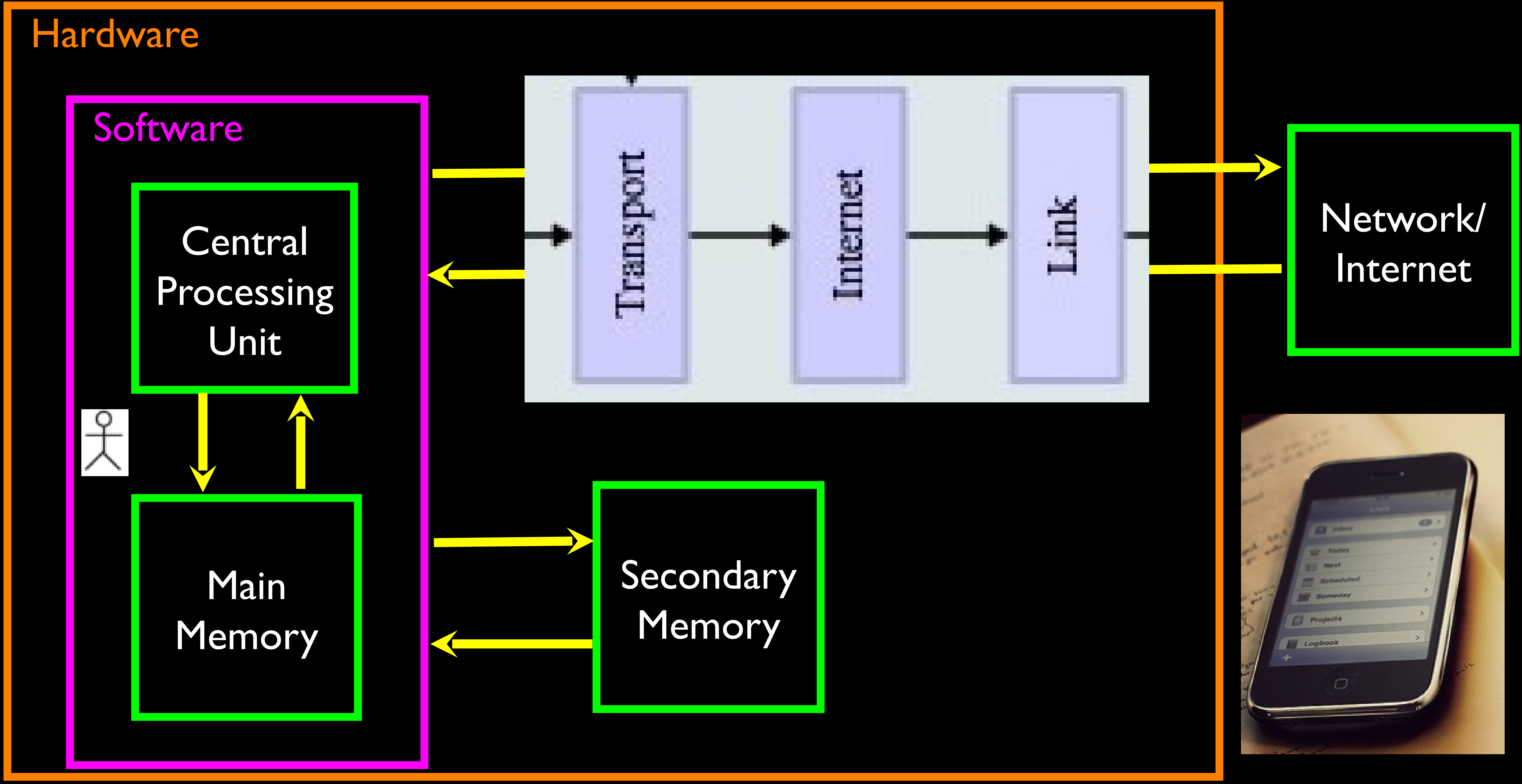
Main Memory

Output Devices

Network/ Internet

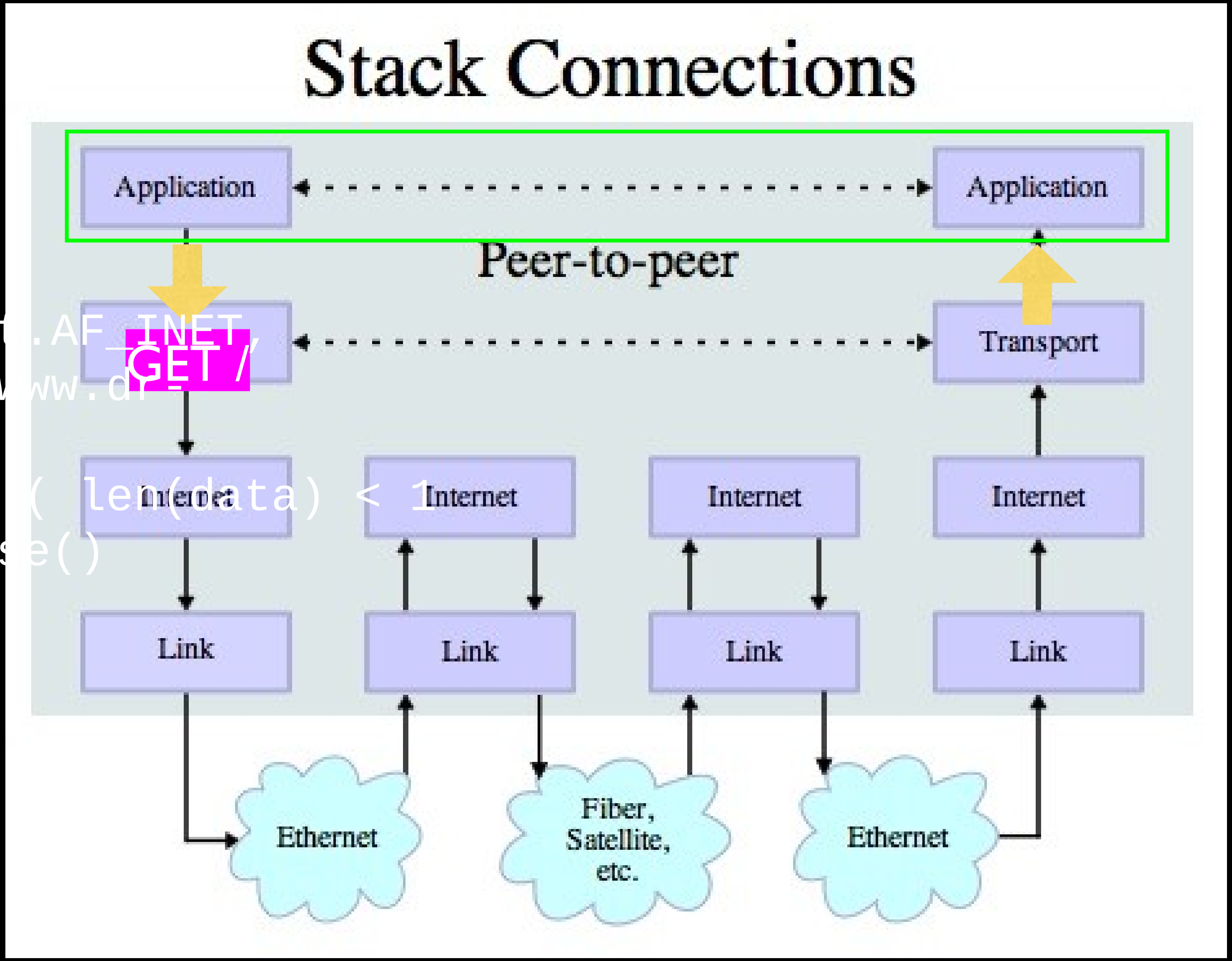
Secondary Memory

Phone CC BY: Johan Larsson (flickr)
http://creativecommons.org/licenses/by/2.0/



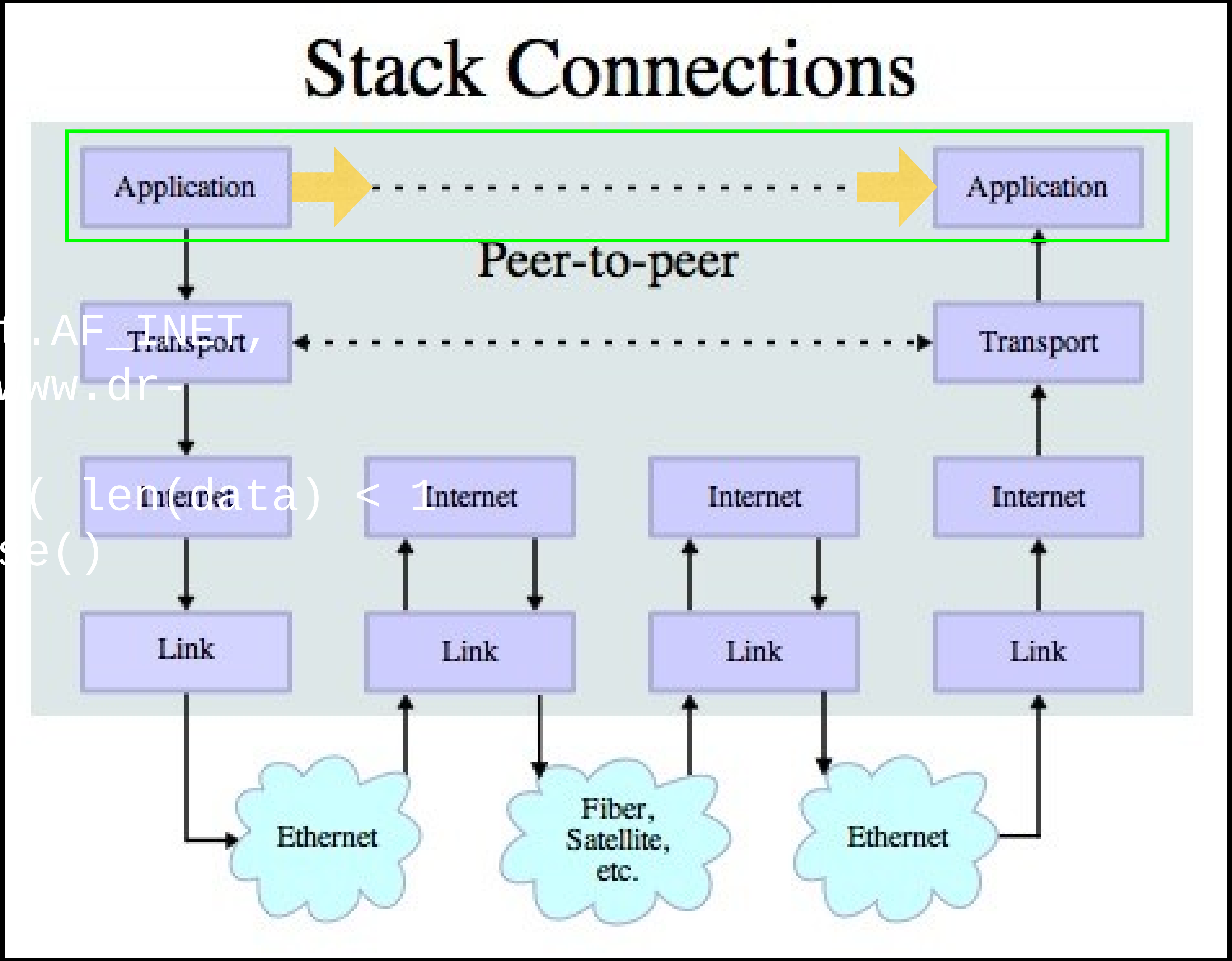
IP Suite: http://en.wikipedia.org/wiki/Internet_Protocol_Suite
Phone CC BY: Johan Larsson (flickr) http://creativecommons.org/licenses/by/2.0/

```
import socketmysock = socket.socket(socket.AF_INET,
     socket.SOCK_STREAM)mysock.connect((“www.dr-
chuck.com”, 80))mysock.send("GET /\n")
while 1:    data = mysock.recv(512)    if ( len(data) < 1
) :        break    print data;mysock.close()
```



Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite

Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite

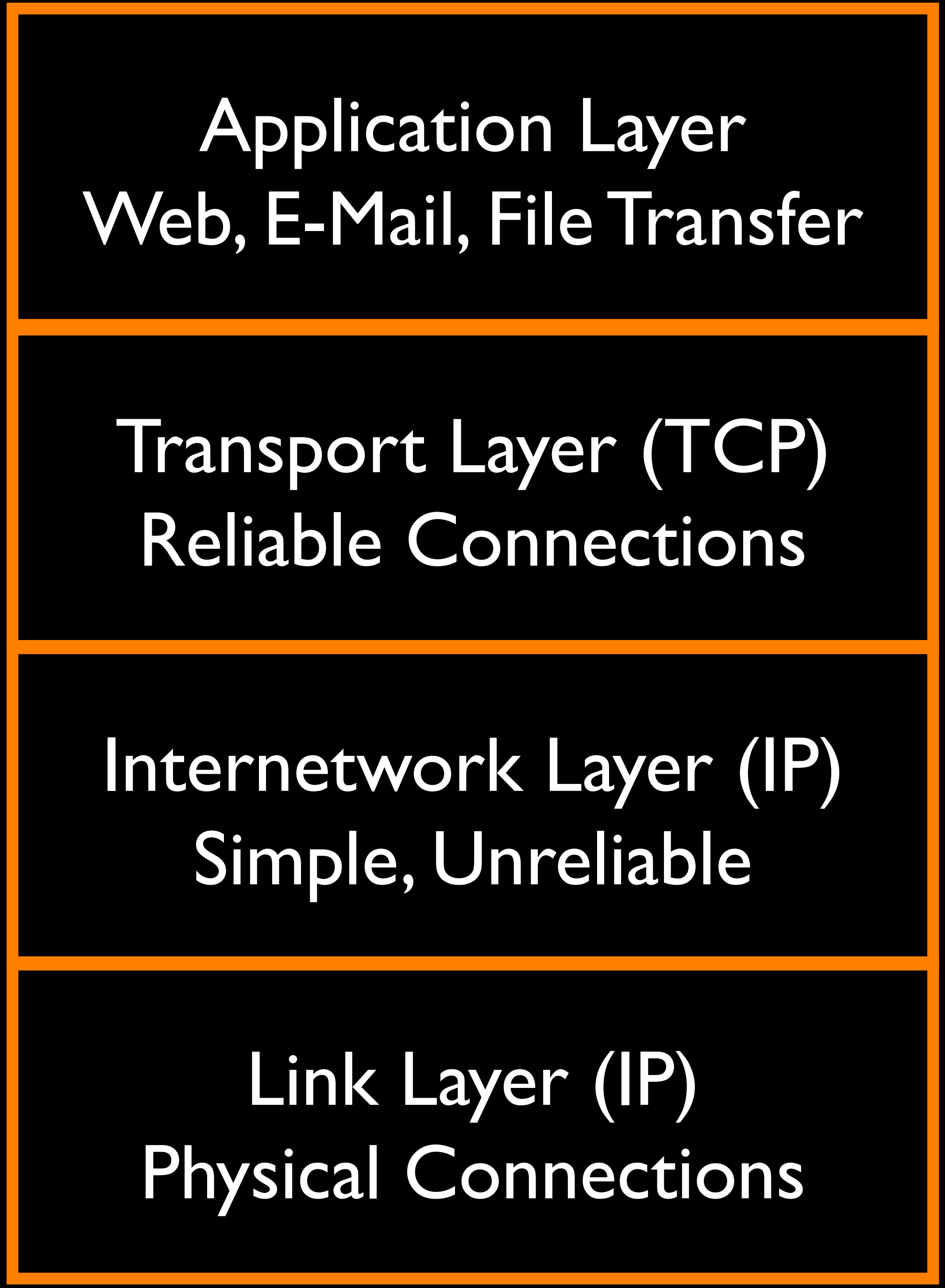


```
import socketmysock = socket.socket(socket.AF_INET,
      socket.SOCK_STREAM)mysock.connect((“www.dr-
chuck.com”, 80))mysock.send("GET /\n")
while 1:    data = mysock.recv(512)    if ( len(data) < 1
) :        break    print data;mysock.close()
```

http://en.wikipedia.org/wiki/Abstraction_(computer_science)

Layered Network Model

- A layered approach allows the problem of implementing a network to be broken into more manageable sub problems
- The layers provide abstraction - each layer can focus on one problem and assume the other layers do their jobs



Application Layer
Web, E-Mail, File Transfer

Transport Layer (TCP)
Reliable Connections

Internetwork Layer (IP)
Simple, Unreliable

Link Layer (IP)
Physical Connections

http://en.wikipedia.org/wiki/Abstraction_(computer_science))

```
import socketmysock =
socket.socket(socket.AF_INET,

socket.SOCK_STREAM)mysock.connect((“www.dr-
chuck.com”, 80))mysock.send("GET /\n")
while 1:    data = mysock.recv(512)    if
( len(data) < 1 ) :        break    print
data;mysock.close()
```

A Simple Web Browser

```
$ python http.py
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
        <title>Error 404 - Not found</title>
</head>

....
```

```
import socketmysock = socket.socket(sock
     socket.SOCK_STREAM)mysock.connect((
chuck.com”, 80))mysock.send("GET /\n")
while 1:    data = mysock.recv(512)    i
( len(data) < 1 ) :        break    prin
data;mysock.close()
```

```
import socket
while 1:
    host = raw_input("Enter host: ");
    if ( host == "quit") :
        break
    mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    mysock.connect((host, 80))
    mysock.send("GET /\n")
   data = ""     while 1:        chunk = mysock.recv(512)        if ( len(chunk)
< 1 ) :            break        # print data;        data = data + chunk     print
"Page size:", len(data)     mysock.close()
```

```
$ python browser.py
Enter host: www.dr-chuck.com
Page size: 1996
Enter host:  www.umich.edu
Page size: 25404
Enter host: www.google.com
Page size: 6152
Enter host:  www.yahoo.com
Page size: 9554
Enter host: ww.dr-chuck.com
Traceback (most recent call last):
  File "browser.py", line 10, in <module>
    mysock.connect((host, 80))
  File "<string>", line 1, in connect
socket.gaierror: (8, 'nodename nor servname provided, or not known')
```

```
import socket
while 1:
    host = raw_input("Enter host: ");
    if ( host == "quit") :
        break
    mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    mysock.connect((host, 80))
    mysock.send("GET /\n")
   data = ""     while 1:        chunk = mysock.recv(512)        if ( len(chunk)
< 1 ) :            break        # print data;        data = data + chunk     print
"Page size:", len(data)     mysock.close()
```

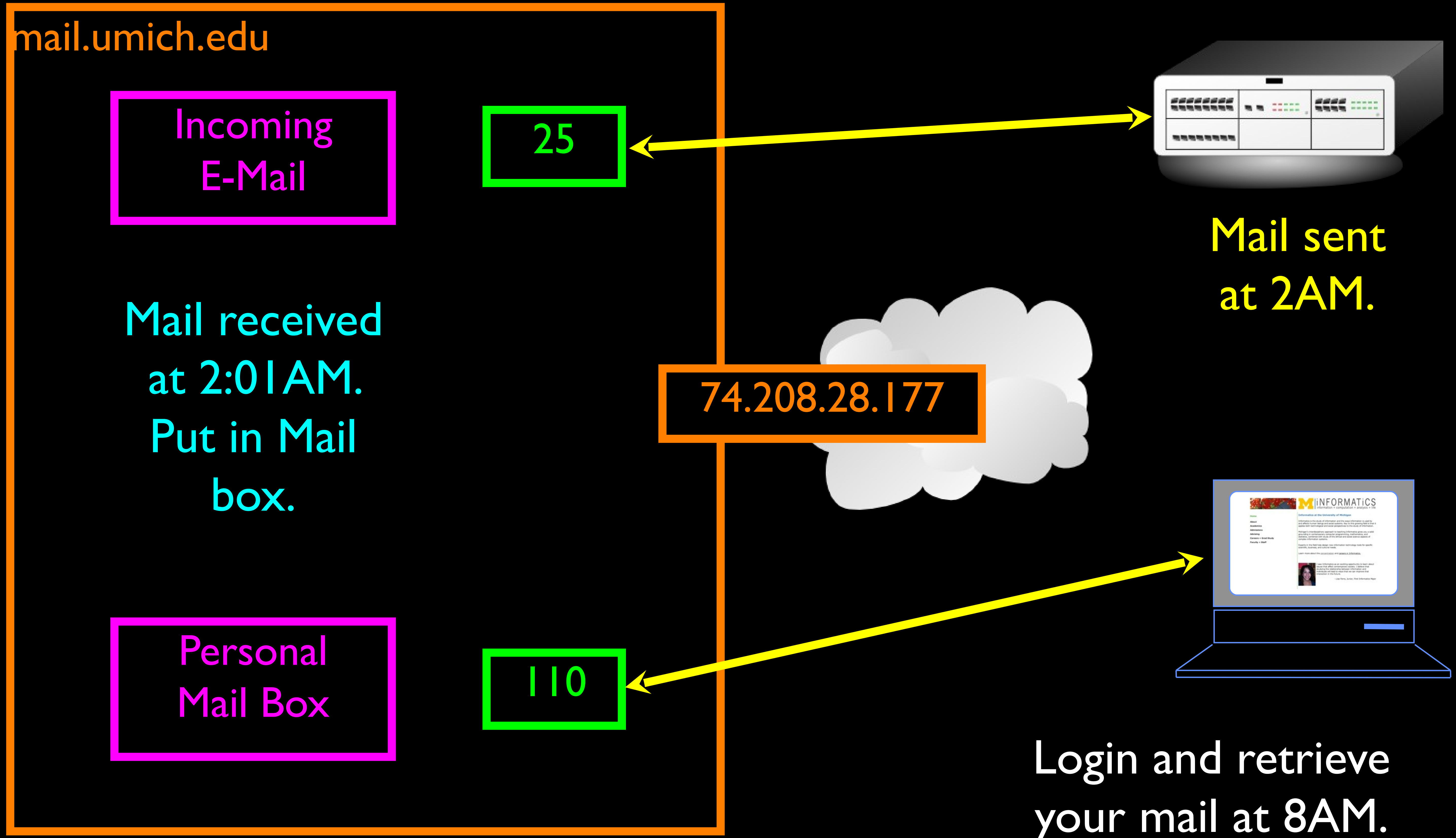
How might we recover from this error in Python more gracefully?

POP3 Protocol

CC BY: Faeryan (flickr)
http://creativecommons.org/licenses/by/2.0/

- Post Office Protocol
- Your laptop/PDA is not always connected to the network
- Your mail is delivered to a "Post office Box" - which is always up and waiting for your mail
- From time to time - you connect to the "Post Office Box" and pull down your new mail

http://www.ietf.org/rfc/rfc1939.txt



mail.umich.edu

Incoming
E-Mail

25

Mail received
at 2:01AM.
Put in Mail
box.

Personal
Mail Box

110

74.208.28.177

Mail sent
at 2AM.

Login and retrieve
your mail at 8AM.

Clipart: http://www.clker.com/search/networksym/1

The Written Specification for the POP3 Protocol

USER name Arguments: a string identifying a mailbox (required), which is of significance ONLY to the server Restrictions: may only be given in the AUTHORIZATION state after the POP3 greeting or after an unsuccessful USER or PASS command Discussion: To authenticate using the USER and PASS command combination, the client must first issue the USER command. If the POP3 server responds with a positive status indicator ("+OK"), then the client may issue either the PASS command to complete the authentication, or the QUIT command to terminate the POP3 session. If the POP3 server responds with a negative status indicator ("-ERR") to the USER command, then the client may either issue a new authentication command or may issue the QUIT command.

The server may return a positive response even though no such mailbox exists. The server may return a negative response if mailbox exists, but does not permit plaintext password authentication. Possible Responses: +OK name is a valid mailbox -ERR never heard of mailbox name Examples: C: USER frated S: -ERR sorry, no mailbox for frated here ... C: USER mrose S: +OK mrose is a real hoopy frood

http://www.ietf.org/rfc/rfc1939.txt

```
$ telnet mail.comcast.net 110
Trying 76.96.30.119...
Connected to mail.g.comcast.net.
Escape character is '^]'.
+OK POP3 ready
USER csev
+OK
PASS ***********
+OK ready
LIST
+OK 32 messages (2611298)
1 3433
2 4009
3 45565
4 8540
.
```

```
RETR 6
+OK
Received: from imta11.westchester.pa.mail.comcast.net
([76.96.62.22])
X-Originating-IP: [76.96.62.22]
Received: by 10.150.57.18 with HTTP; Tue, 10 Jun 2008
13:33:13 -0700 (PDT)
Date: Tue, 10 Jun 2008 16:33:13 -0400
From: "Bob S." <***********@gmail.com>
To: c.severance@ieee.org
Subject: Blast from the past

Hi Chuck,

I want to comment on your router problem at home.
Are you being bad and using something other than
Unix/Linux to combine your network connection?

Bob S
.
```

Summary

- We start with a “pipe” abstraction - we can send and receive data on the same “socket”
- We can optionally add a security layer to TCP using SSL - Secure Socket Layer (aka TLS - Transport Layer Security)
- We use well known “port numbers” so that applications can find a particular application *within* a server such as a mail server, web service, etc

Summary

- When doing network programming we make use of a library that hide all the detail of how the Internet is put together - we simple open and use a TCP connection (Abstraction)
- Each application defines a set of rules of interaction between client and server (a protocol)
- Knowing the protocol - we can write an application to talk that protocol