

The Final Lecture

Databases
Daemons and the Internet

Prof. Steven Ludtke
N420, sludtke@bcm.edu

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...

Spreadsheet !

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...

Surname	GivenName	Email	Program	Institution
Anderson	Alex	alex@bcm.edu	Biochemistry	BCM
Brown	Brad	<u>bradb@bcm.edu</u>	Genetics	BCM
Chen	Claudia	<u>claudiac@bcm.edu</u>	Neuroscience	BCM
Davis	Alice	<u>adavis@rice.edu</u>	Physics	Rice
Ferris	Zed	<u>zedf@bcm.edu</u>	Biochemistry	BCM
Su	Jin	<u>jins@rice.edu</u>	Math	Rice

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...
- We now want to record all of the students grades in all of his/her classes. How to approach?

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...

Surname	GivenName	Email	Program	Institution	Course	Year	Term	Grade
Anderson	Alex	alexa@bcm.edu	Biochemistry	BCM				
Brown	Brad	bradb@bcm.edu	Genetics	BCM				
Chen	Claudia	claudiac@bcm.edu	Neuroscience	BCM				
Davis	Alice	adavis@rice.edu	Physics	Rice				
Ferris	Zed	zedf@bcm.edu	Biochemistry	BCM				
Su	Jin	jins@rice.edu	Math	Rice				

How to Store Complex Data ? (again)

- Students
 - name, email, program, institution ...
- Grades
 - class, year, student, grade ...
- How do we represent "student"?
- How about "class"?

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...

Surname	GivenName	Email	Program	Institution
Anderson	Alex	alexa@bcm.edu	Biochemistry	BCM
Brown	Brad	<u>bradb@bcm.edu</u>	Genetics	BCM
Chen	Claudia	<u>claudiac@bcm.edu</u>	Neuroscience	BCM
Davis	Alice	<u>adavis@rice.edu</u>	Physics	Rice
Ferris	Zed	<u>zedf@bcm.edu</u>	Biochemistry	BCM
Su	Jin	<u>jins@rice.edu</u>	Math	Rice

How to Store Complex Data ? (again)

- Students
 - Add a 'accession number' or 'record number'

Row	Surname	GivenName	Email	Program	Institution
1	Anderson	Alex	alexa@bcm.edu	Biochemistry	BCM
2	Brown	Brad	<u>bradb@bcm.edu</u>	Genetics	BCM
3	Chen	Claudia	<u>claudiac@bcm.edu</u>	Neuroscience	BCM
4	Davis	Alice	<u>adavis@rice.edu</u>	Physics	Rice
5	Ferris	Zed	<u>zedf@bcm.edu</u>	Biochemistry	BCM
6	Su	Jin	<u>jins@rice.edu</u>	Math	Rice

How to Store Complex Data ? (again)

- Students
 - name, email, program, institution ...
- Grades
 - class, year, student, grade ...

Class	Year	Student	Grade
Ph101	2018	1	A
Bioc37	2018	1	B-
Eng234	2017	1	A-
Ph101	2018	3	C
Chem22	2016	5	B

How to Store Complex Data ? (again)

- Students
 - Name, email, program, institution ...
- Classes
 - Course #, Description, Instructor, term, room, hours, ...
- Class Years
 - Which class, year, students
- Grades
 - Class Year, student, grade

How to Store Complex Data ? (again)

- Protein
 - accession code, name, sequence, organism, pathway
- organism
 - accession code, common name, formal name
- pathway
 - ...

How to Store Complex Data ? (again)

- Dictionary
- Classes (object oriented programming)
- XML

Databases

- ◉ Simple (embedded) Database
 - ◉ dbm, BerkeleyDB (bsddb), SQLite*
- ◉ Relational Database
 - ◉ MySQL, Oracle, DB2, SQLite*
- ◉ NoSQL Database
 - ◉ Object Oriented Database (OODB)
 - ◉ Zope, Databeans
 - ◉ XML Database
 - ◉ Sedna, Oracle, BerkeleyDB-XML
 - ◉ MonqoDB (document-based)

Simple/Embedded Database

- Key/Value pairs
- Python
 - Shelve — Persistent dictionary
 - anydbm — Generic access to DBM-style databases
 - whichdb — Guess which DBM module created a database
 - dbm — Simple “database” interface
 - gdbm — GNU’s reinterpretation of dbm
 - dbhash — DBM-style interface to the BSD database library
 - dumbdbm — Portable DBM implementation
 - * bsddb (bsddb3) — Interface to Berkeley DB library

pickle

- 'Serialization' - converting a complex object to a stream of data

```
from pickle import dump,load,dumps,loads
```

```
dump(obj,file[,protocol]) # stores 'obj' in file
```

```
obj=load(file) # restores 'obj' from file
```

```
str=dumps(obj[,protocol]) # pickled representation of obj
```

```
obj=loads(str) # restore representation of obj
```

shelve

```
import shelve      # dictionary-like object on disk
dct=shelve.open(filename[,protocol])
dct=shelve.open(filename,writeback=True)
dct.close()
```

Databases

- Simple (embedded) Database
 - dbm, BerkeleyDB (bsddb), SQLite*
- Relational Database
 - MySQL, Oracle, DB2, SQLite*
- NoSQL Database
 - Object Oriented Database (OODB)
 - Zope, Databeans
 - XML Database
 - Sedna, Oracle, BerkeleyDB-XML
 - MonqoDB (document-based)

(Relational) Database Terms

- Table - equivalent to a single spreadsheet page
- Database - a set of tables
- Column - one specific type of data for all records in a table
- Row - one record in a table
- Tuple - another name for one row in the database
- Schema - The titles and datatypes of all the rows in all of the tables
- RDMS - Relational Database Management System

Relational Database

Tables (Relations)

First Name	Last Name	Program	Grade
John	Barnes	1	A
Frank	Smith	1	B+
Carol	Franklin	2	A-
Steve	Black	3	C+
Charles	Baker	1	B+

For many-→many we use a 'junction table'.

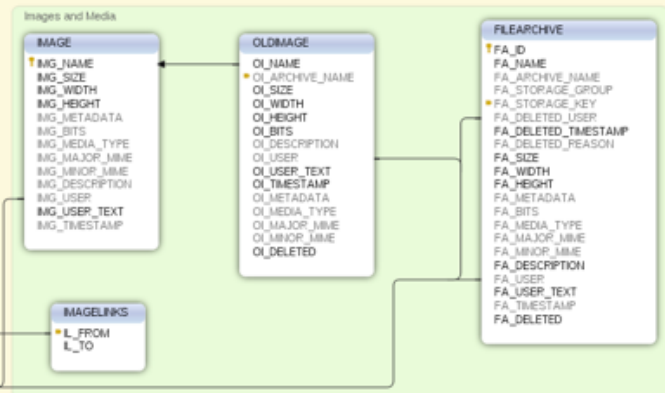
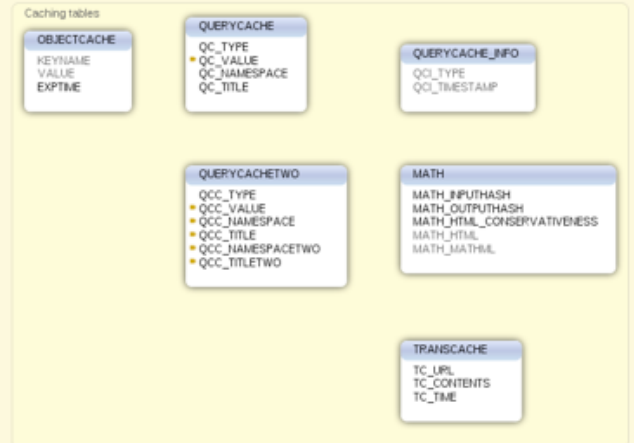
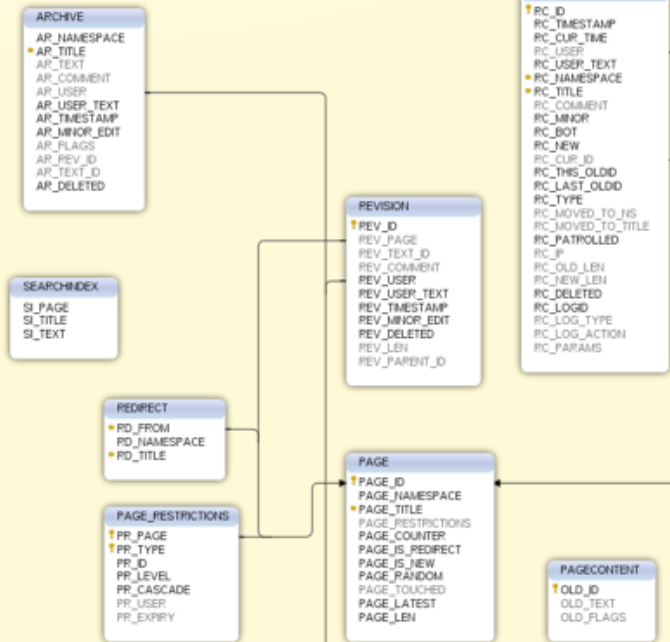
1 to 1

id	Name	Institution
1	SCBMB	BCM
2	Biochemistry	BCM
3	Biochemistry	Rice

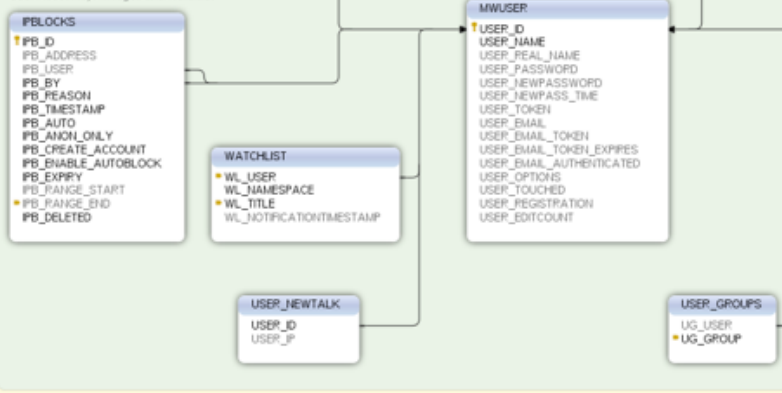
Database Schema

Media Wiki database schema
Move the mouse cursor over the table header and columns to view the comments
generated using DbSchema

Article text and association information



User Accounts, Privileges and Watchlist



ACID

- Atomicity

- Transactions are 'all or none', no partial modifications

- Consistency

- Contents of the database are always valid and self-consistent

- Isolation

- Readers will not see partially completed transactions of writers

- Durability

- Completed transactions can survive any possible system crash

SQL

- Structured Query Language
- Developed at IBM in the 70s
- ANSI Standard in 1986
- Now fairly ubiquitous for Relational Databases
- (see Wikipedia for more trivia)

SQLite3

Python type	SQLite type
<u>None</u>	NULL
<u>int</u>	INTEGER
<u>long</u>	INTEGER
<u>float</u>	REAL
<u>str</u> (UTF8-encoded)	TEXT
<u>unicode</u>	TEXT
<u>buffer</u>	BLOB

SQLite3

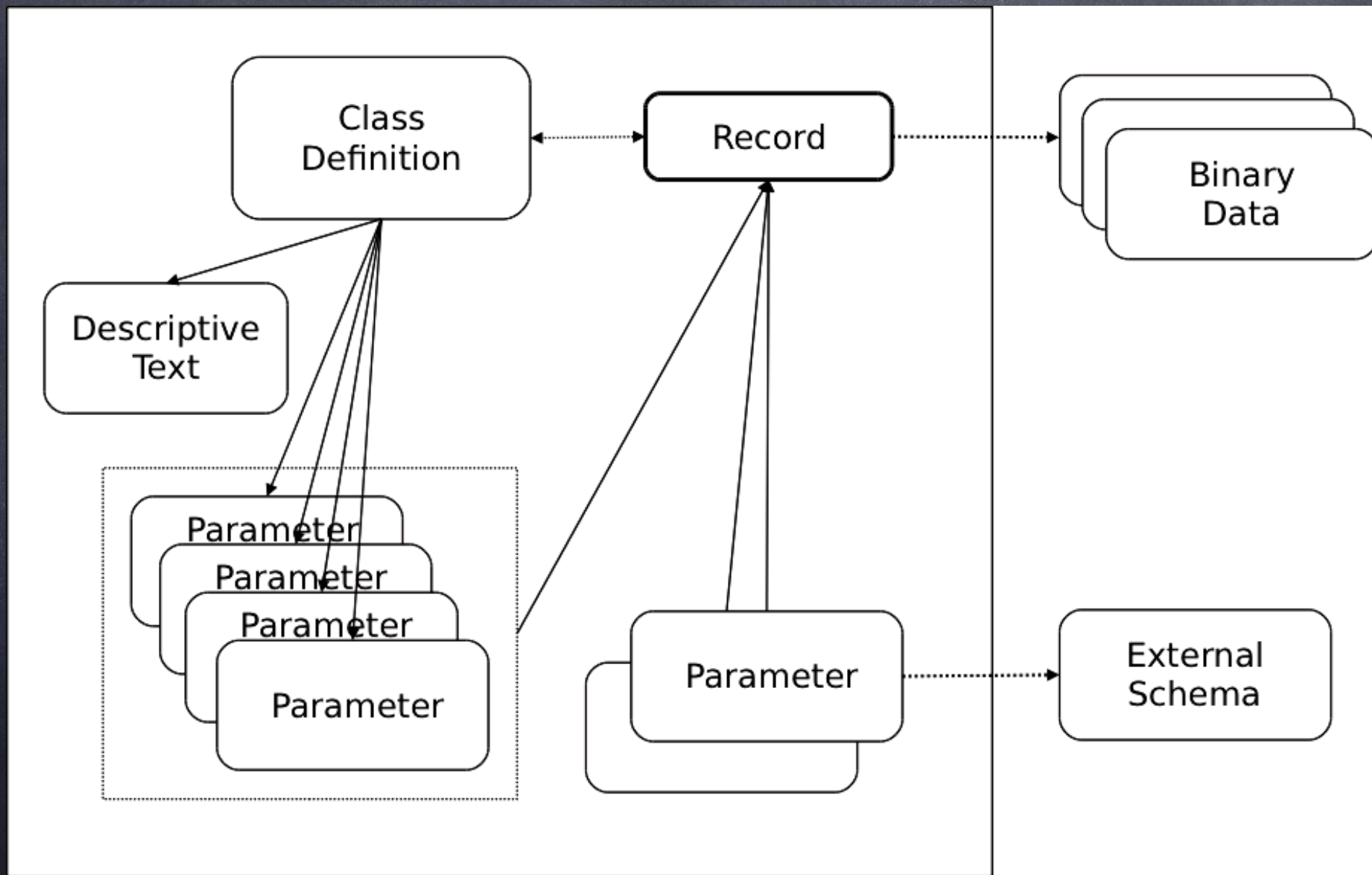
- `import sqlite3`
- `conn=sqlite3.connect("mydb.db")`
- `cur=conn.cursor()`
- create table
 - `cur.execute("CREATE TABLE Person(Id INT, Name TEXT, Zip INT);")`
- insert
 - `cur.execute("INSERT INTO Person VALUES (1,'Steve',77884)")`
- select (search)
 - `cur.execute("SELECT * FROM Person")`
 - `print(cur.fetchall())`
- update
 - `cur.execute("UPDATE Person SET zip=77584 WHERE Name='Steve'")`
- drop (delete)
 - `cur.execute("DROP TABLE Person")`

Databases

- Simple (embedded) Database
 - dbm, BerkeleyDB (bsddb), SQLite*
- Relational Database
 - MySQL, Oracle, DB2, SQLite*
- NoSQL Database
 - Object Oriented Database (OODB)
 - Zope, Databeans
 - XML Database
 - Sedna, Oracle, BerkeleyDB-XML
 - MonqoDB (document-based)

OODB

(Object Oriented DataBase)



XML Database

File0001.xml:

```
<colony name="HD122">
  <cage number="7">
    <mouse>
      <tag>19-834</tag>
      <strain>YAC128</strain>
      <gender>M</gender>
      <measurement date="1/12/15">
        <mass units="g">10.3</mass>
        <rotarod units="s">45</rotarod>
      </measurement>
      <measurement date="1/13/15">
        <rotarod>53</rotarod>
        <mass>10.4</mass>
      </measurement>
    </mouse>
    <mouse>
      ...
    </mouse>
    ...
  </cage>
  <cage number="8"> <mouse> ... </mouse> ... </cage>
</colony>
```

File0002.xml:

...

The Internet

- When I enter www.google.com into my web browser, what happens?

The Internet

- My machine opens a connection to the server at Google and requests the main HTML page
- Google server sends the page
- Browser displays the page

The Internet

- My machine opens a connection to the server at Google and requests the main HTML page
- Google server asks who I am (cookie)
- My browser sends my credentials
- Google server sends the (personalized) page
- Browser displays the page

The Internet

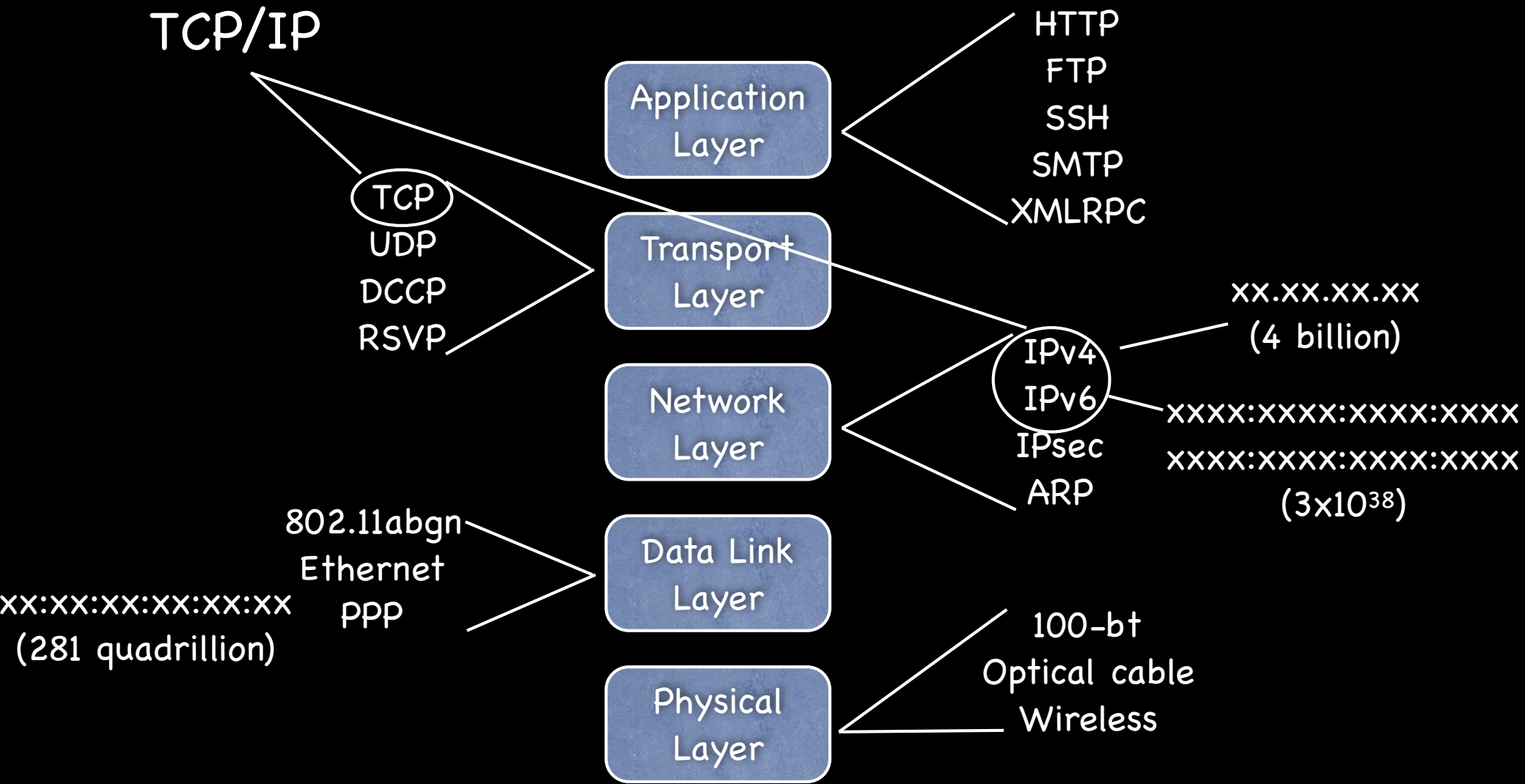
- My machine opens a connection to the server at Google and requests the main HTML page
 - How does my machine connect to www.google.com?
 - Where is this machine?
 - How does it connect?
 - What does "connect" mean?

The Internet

- My machine asks my local DNS server for the IP address of the name "www.google.com"
- DNS server either has the information or gets it by asking other DNS servers. Sends my machine the address.
- My machine opens a TCP connection to this address on port 80, and sends a HTTP request for "/" the root page.
- Google looks at the information in my request, including the browser I claim to be using, sends back its request to my browser for a "cookie" it stored in my browser the last time I visited the site.
- My browser returns the "cookie" if it has one, or replies that it doesn't have one.
- Google assembles a customized HTML page based on everything it knows and sends it back to me.
- My browser renders the HTML page, and starts running any embedded JavaScript programs.

Networking

TCP/IP

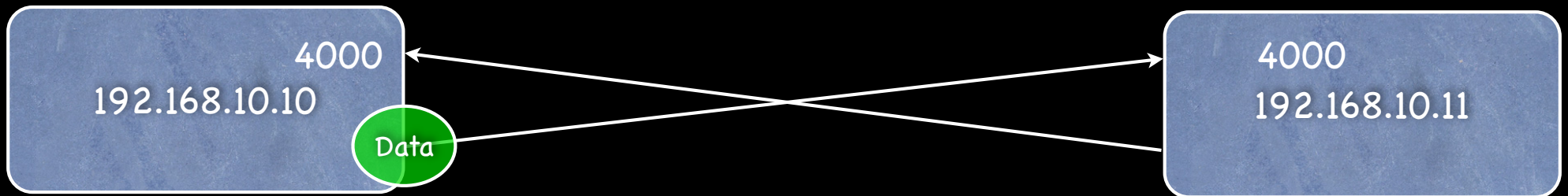


Common Services

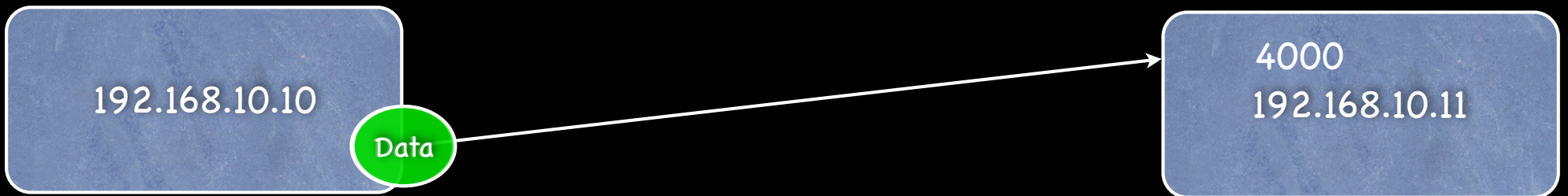
port	service
21	ftp
22	ssh
23	telnet
25	smtp (mail)
79	finger
80	http (web)
110	pop3 (email retrieval)
123	ntp (time)
137-139	Windows file sharing
143	imap (email retrieval)
443	https (secure http)

Sockets (TCP/UDP)

UDP

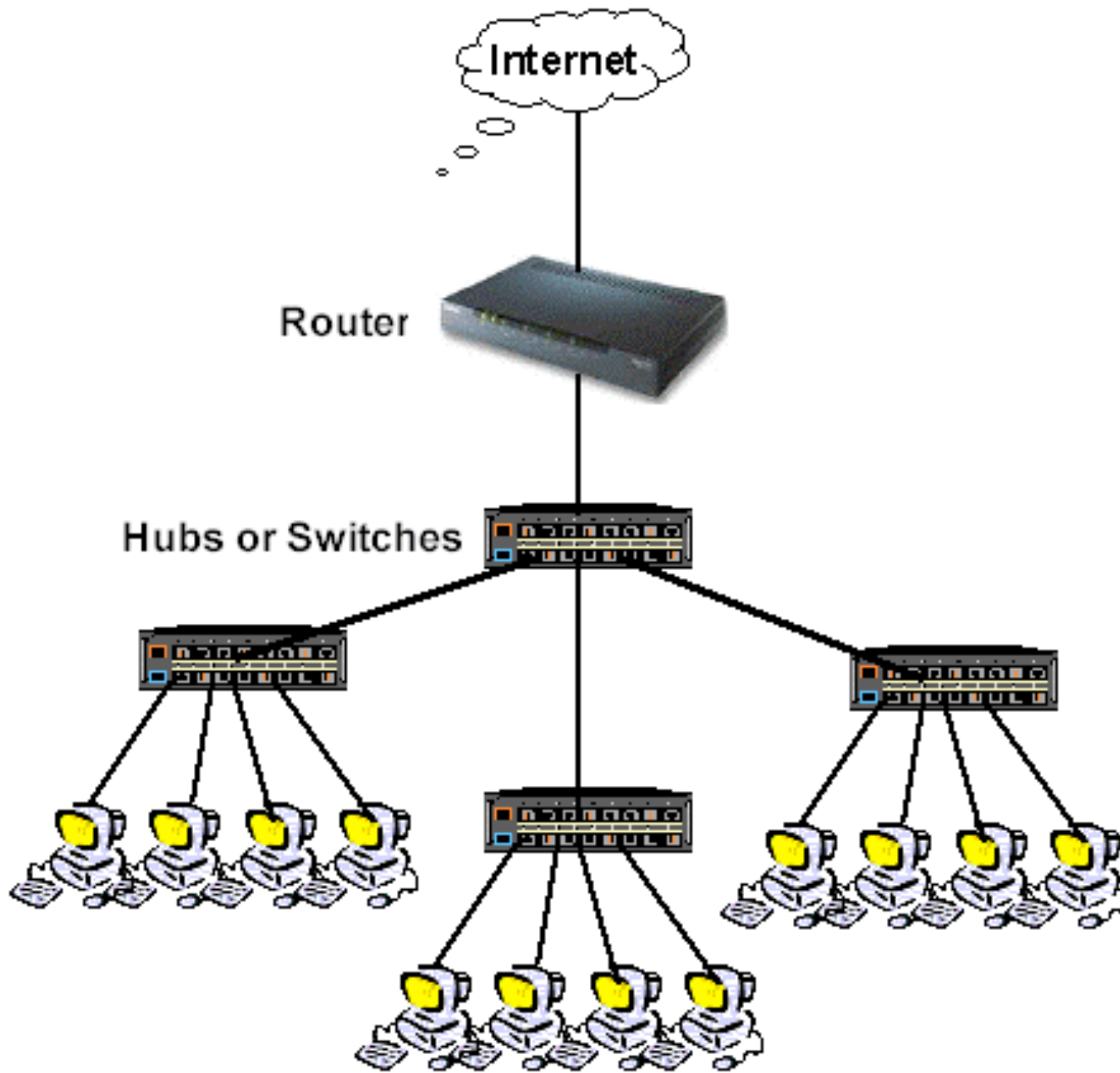


TCP



IPv4 Network Parameters

- IP Address - Computer's unique* address (x.x.x.x)
- Netmask - defines local 'subnet', machines the computer can speak to 'directly'
- Router - Address used to contact machines outside subnet
- DNS Server - Address where names can be mapped to addresses
- Port - For a specific connection 0-65535, 0-1023 reserved for system services



Socket Module

- `gethostname()` - name of the current machine
- `gethostbyname()` - given a name, returns an IP address
- `gethostbyaddr()` - given an address, returns names and other info
- `socket()` - create a new socket
 - `listen(n)` - waits for incoming connections on a socket $n > 1$
 - `accept()` - accepts an incoming connection, returns tuple with socket
 - `connect((host,port))` - connects to a remote socket
 - `send()`, `recv()` - send and receive data
 - `sendall()` - send until success or error
 - `makefile()` - make a file-like object for the socket
- `bind()` - bind a socket to an address
- `select()` - from select module, lets you wait for activity on set of sockets

UDP

```
#receiver
```

```
import socket
```

```
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

```
s.bind(("",40000))
```

```
print(s.recv(1000)) # up to 1000 bytes
```

```
#sender
```

```
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

```
s.bind(("",40000))
```

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
```

```
s.sendto("Hello there",("<broadcast>",40000))
```

Making Connections

```
# Receiver/server
import socket
print("I am: ",socket.gethostbyname(socket.gethostname()))

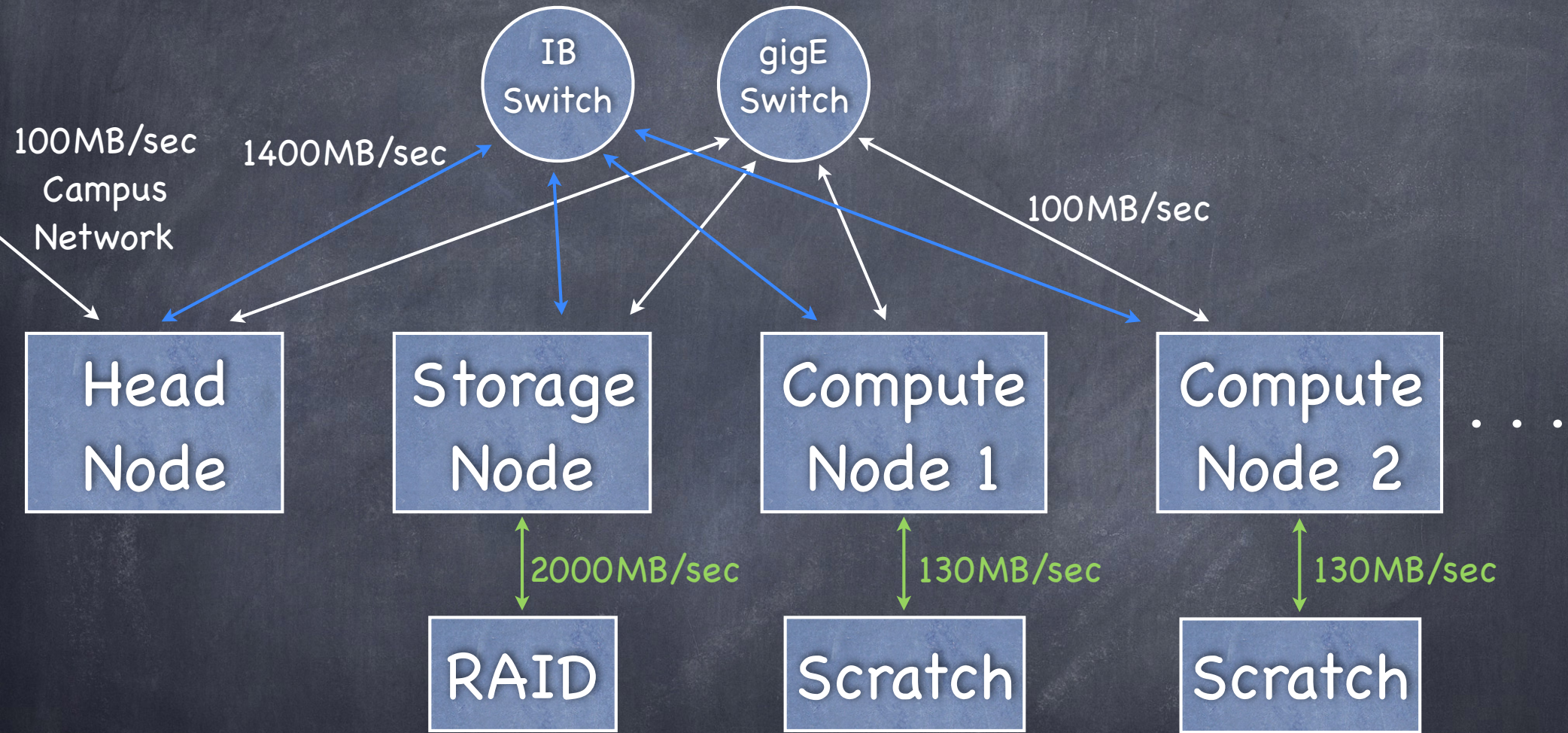
sock=socket.socket()    # default is to make a normal internet socket
sock.bind(("",40000))    # Nothing magic about 40000
sock.listen(3)          # Wait for 'connect' requests
while True:
    sock2=sock.accept()    # accept the connection (new socket)
    msg=sock2[0].recv(1024)    # receive up to 1024 bytes
    if msg==b"exit": break
    print(msg)
print("The remote user told me to exit!")
sock.close()

# Sender
import socket
sock=socket.socket()    # default socket
sock.connect(("
```

Socket vs File Obj.

- Socket:
 - send - will transmit data immediately
 - recv - specifies maximum amount to receive. May not get all sent data in one call. Call multiple times until you have what you expected
- File:
 - write - buffers. Need to call flush() for immediate transmission.
 - read/readline - Reads the full amount expected. Multiple calls not required.

Hypothetical Cluster



The Internet

- What happens when I go to <http://maps.google.com> ?

Simple Python Webserver

```
# This will serve files from the current directory
# we use port 8080 because port 80 is restricted

from http.server import *

httpd = HTTPServer(("", 8080), SimpleHTTPRequestHandler)
httpd.serve_forever()
```

Security?

How Does Email Work?

- When I send an email to xxx@yyy.edu, what happens?
- Do multimedia emails actually contain all of those images and movies?

CLASS PROJECTS

- Must do something useful in some specific context
- Not be trivial, or readily available with the same functionality (without approval)
- If you have past programming experience I will expect more
- **Please follow these instructions exactly:**
- Your class project **MUST** be submitted by 11:59 PM on Sat, Feb 24. No revisions will be accepted after this time. You can use Sunday to prepare your oral presentation
- Your submission should consist of:
 - one or more .py files (no .ipynb! Should have comments '#' explaining the code)
 - include necessary additional files to demonstrate that the program works
 - A 1-2 page PDF file with a brief description of your program, what inputs the program takes, what outputs the program produces, and what it is supposed to do.
 - The final item in the PDF should be a command-line to use in running the program, and any necessary instructions to demonstrate that it works.
- Combine all files into a .zip or .tgz file named:
Familyname_Givenname_project_2018.zip (or .tgz)
- Email **sludtke42@gmail.com** with the subject "Class project submission", and attach the .zip file. If the file is too large for email, feel free to transfer it via Box, DropBox, etc.

CLASS PROJECT PRESENTATIONS

- Monday, Feb 26
- **8 AM** - 10:30
- **First slide:** Program Title, Your Name, Department/Graduate Program
- You will have 6 minutes total:
 - Set up your presentation (1 minute) - TEST LAPTOP BEFORE FEB 26!!!
 - Give your talk (4 minutes. practice!)
 - What does your software do, and why did you write it
 - Inputs and outputs
 - Demonstration (mock demonstration if it takes too long)
 - Questions (1 minute)
 - Class projects are 1/2 of your final grade in the class.
- The program **MUST WORK** to get a good grade. Better to turn in something that doesn't do everything you wanted, but works, than something broken

CLASS PROJECT RUBRIC

- Project (2/3 project grade):
 - Program works (2 pt)
 - Does something useful (1 pt)
 - Programming style (1/2 pt)
 - Followed all instructions (1/2 pt)
- Presentation (1/3 project grade):
 - Presentation Clarity (2 pt)
 - Program Demonstrated Effectively (1 pt)
 - Presentation Organization (1/2 pt)
 - Followed all instructions, including projector issues (1/2 pt)