Computer Science Curriculum and Teaching

Andrew Rosen

May 18th, 2016

Table of Contents

Research

- Fault Tolerant Systems
- Distributed Hash Tables
- Non-traditional systems and problems
  - DHT Computing
  - VANETS
  - Delay Tolerant Networks

Teaching

Taught:

- Principles of Computer Science
- System Level and C Programming (eg Unix and C)
- Data Structures

TA'd the above and:

- Networks
- Security
- Principles of Computer Science (with robots)

Personal Bias

- Draw on history and other disciplines whenever possible

Personal Bias

- Draw on history and other disciplines whenever possible
  - Creates real world examples
  - Creates better stories

Personal Bias

- Draw on history and other disciplines whenever possible
  - Creates real world examples
  - Creates better stories
- I limit the use of slides
  - I often create my own materials and lecture notes for students
  - I live code when I can

Table of Contents

Principles of Computer Science

- First real programming class
- Arguably the hardest to teach
- Taught 100 class, twice
- Four lab sections
- Uses Java

Curriculum

- History and Definitions
- Syntax
- Methods
- Loops and Conditions
- OO Basics and Strings
- Binary
- CS Breadth

System Level Programming

- Taught once, TA'd once
- 51 students
- Covers proficiency in Unix
- Intermediate C
- Huge amount of subject matter

Unix Content

- Bash basics and commands
- Permissions
- Regex
- awk/sed/grep and other scripting tools
- Malfeasance
- A tad bit of Python

C Content

- Differences between C and Java
- Pointers and pointer arithmetic
- Memory management
- Some brief exposure to compilers and interpreters

My Favorite Course: Data Structures

- Taught Twice
- 25 students
- Best class for first time instructors due to :
  - Class makeup and experience
  - Modular content
- Our course had an emphasis on documentation.

Content

- Review and teach last bits of Java
  - Sometimes grad students w/ different background
  - Try/Catch
  - File I/O

Content

- Review and teach last bits of Java
  - Sometimes grad students w/ different background
  - Try/Catch
  - File I/O
- Lists
  - ArrayList
  - Linked Lists

Content

- Review and teach last bits of Java
  - Sometimes grad students w/ different background
  - Try/Catch
  - File I/O
- Lists
  - ArrayList
  - Linked Lists
- Stacks and Queues
  - Good time to introduce Java's Generics

Content

- Review and teach last bits of Java
  - Sometimes grad students w/ different background
  - Try/Catch
  - File I/O
- Lists
  - ArrayList
  - Linked Lists
- Stacks and Queues
  - Good time to introduce Java's Generics
- Trees

Content

- Review and teach last bits of Java
  - Sometimes grad students w/ different background
  - Try/Catch
  - File I/O
- Lists
  - ArrayList
  - Linked Lists
- Stacks and Queues
  - Good time to introduce Java's Generics
- Trees
- Graphs (if time remains)

Table of Contents

The Skill Gap

- Skill gap has many sources
- Inform experienced students
- Address skill gap; students feel less intimidated
- Target and gauge the middle row

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row
- I use a story about cold reading
    - People remember hits more than misses.

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row
- I use a story about cold reading
  - People remember hits more than misses.
- Make yourself available

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row
- I use a story about cold reading
  - People remember hits more than misses.
- Make yourself available
- Talk about your experiences

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row
- I use a story about cold reading
  - People remember hits more than misses.
- Make yourself available
- Talk about your experiences
- Clickers and short, anonymous, in-class exercises

[1] Sweeping Generalization

Fear and Shyness

Students hate being wrong and are shy[1]. How do we encourage interaction?

- Focus on the middle row
- I use a story about cold reading
  - People remember hits more than misses.
- Make yourself available
- Talk about your experiences
- Clickers and short, anonymous, in-class exercises

These seem obvious, but they require repetition in class.

[1] Sweeping Generalization

Unfamiliarity

- CS courses are different.
  - Most similar to Math
  - Expectations most similar to Music
- Relate unfamiliar concepts to:
  - The real world
  - Other domains
  - Previous knowledge
- Build on what students know
- Revisit old material
- Make it cool

Example Problem

Recently, NASA demonstrated a laser communication system which was able to transmit data from the Moon to Earth over a link with a bandwidth of 622 megabits/second. How long would it take an astronaut to send a 500 megabyte video from the Moon to Earth?

- The average distance from Earth to its moon is 384,400 kilometers
- Speed of light $\approx 300,000,000\frac{m}{s}$

Beware of the Pitfalls

Know where students begin to fall behind.

- Consistent model of assignment in Intro
- Functional Decomposition (writing methods)
- Pointers and arrays in C
- Everything is a file in Unix
- Linked Lists and pointers in Data Structures

Table of Contents

"There are only two kinds of languages: the ones people complain about and the ones nobody uses."

– Bjarne Stroustrup, creator of C++

Why Do We Use Java

- Universal
- A whole lot like C and C++ (and anything based off them)
- Good teaching resources
- References a good enough starting point for pointers
- No segfaults or memory leaks

I Prefer Python

- No “black magic”
  - (well less of it is immediately apparent)
- Easy to teach concepts and pseudocode
- Syntax is easy and forgiving
- Dictionaries

Other Suggestions

- Hard concepts earlier
- More complete examples of code before we make students code.
- Experiment
    - Active Learning
    - Flipped Classroom

Live Coding

- Learn from mistakes
- Demonstrate thought process
- Socratic Approach
- Experimentation
- Provides worked examples

Table of Contents

Challenges of Distributed Computing

Distributed Computing platforms experience these challenges:

Scalability As the network grows, more resources are spent on maintaining and organizing the network.

Challenges of Distributed Computing

Distributed Computing platforms experience these challenges:

Scalability As the network grows, more resources are spent on maintaining and organizing the network.

Fault-Tolerance As more machines join the network, there is an increased risk of failure.

Challenges of Distributed Computing

Distributed Computing platforms experience these challenges:

Scalability As the network grows, more resources are spent on maintaining and organizing the network.

Fault-Tolerance As more machines join the network, there is an increased risk of failure.

Load-Balancing Tasks need to be evenly distributed among all the workers.

Distributed Key/Value Stores

Distributed Hash Tables are mechanisms for storing values associated with certain keys.

- Values, such as filenames, data, or IP/port combinations are associated with keys.
- These keys are generated by taking the hash of the value.
- We can get the value for a certain key by asking any node in the network.

How Does It Work?

- DHTs organize a set of nodes, each identified by an **ID**.
- Nodes are responsible for the keys that are closest to their IDs.
- Nodes maintain a small list of other peers in the network.
  - Typically a size $\log(n)$ subset of all nodes in the network.
- Each node uses a very simple routing algorithm to find a node responsible for any given key.

Current Applications

Applications that use or incorporate DHTs:

- P2P File Sharing applications, such as BitTorrent.
- Distributed File Storage.
- Distributed Machine Learning.
- Name resolution in a large distributed database.

Strengths of DHTs

DHTs are designed for large P2P applications, which means they need to be (and are):

- Scalable
- Fault-Tolerant
- Load-Balancing

Research Projects

- ChordReduce
- UrDHT

Other Research

- Mapped DHTs to Voronoi/Delaunay
- Created a greedy heuristic for calculating Voronoi regions
- Sybil Attack Analysis

Table of Contents

P2P

- Networking typically glazes over P2P applications
- Presents another way of discussing fault tolerance
- DHTs are the backbone for most P2P networks
- Generalized model for distributed computing
- Could be used as part of a Special Topics for distributed and decentralized systems.

Cryptographic Hashes

- Essential part of DHTs
- Ties into:
    - Message Authentication
    - Signatures

Security

- Nothing is secure
- Human elements are the weakest
- CIA triad
    - Little focus paid to "A"
- Security of distributed systems is an aside.
- Systems are not more secure just because they are decentralized

Attacks on Distributed Systems

- Sybil Attack
  - Attacker aggressively creates and maintains multiple identities
  - Goal is to become at least 33% of network
  - Prefer majority to have complete control.
- Eclipse Attack is like Sybil but:
  - Goal is to prevent "good" nodes from communicating directly
  - Occlude or eclipse all connections
  - Attack can poison routing tables and force bad updates
- How do you prevent these attacks?