Computer Science 162
Section 0

CS162 Teaching Staff

Welcome

- Computer systems are exciting!
- Thousands of engineers in industry building huge systems at scale
- You get to make real stuff that people can use

Projects Overview

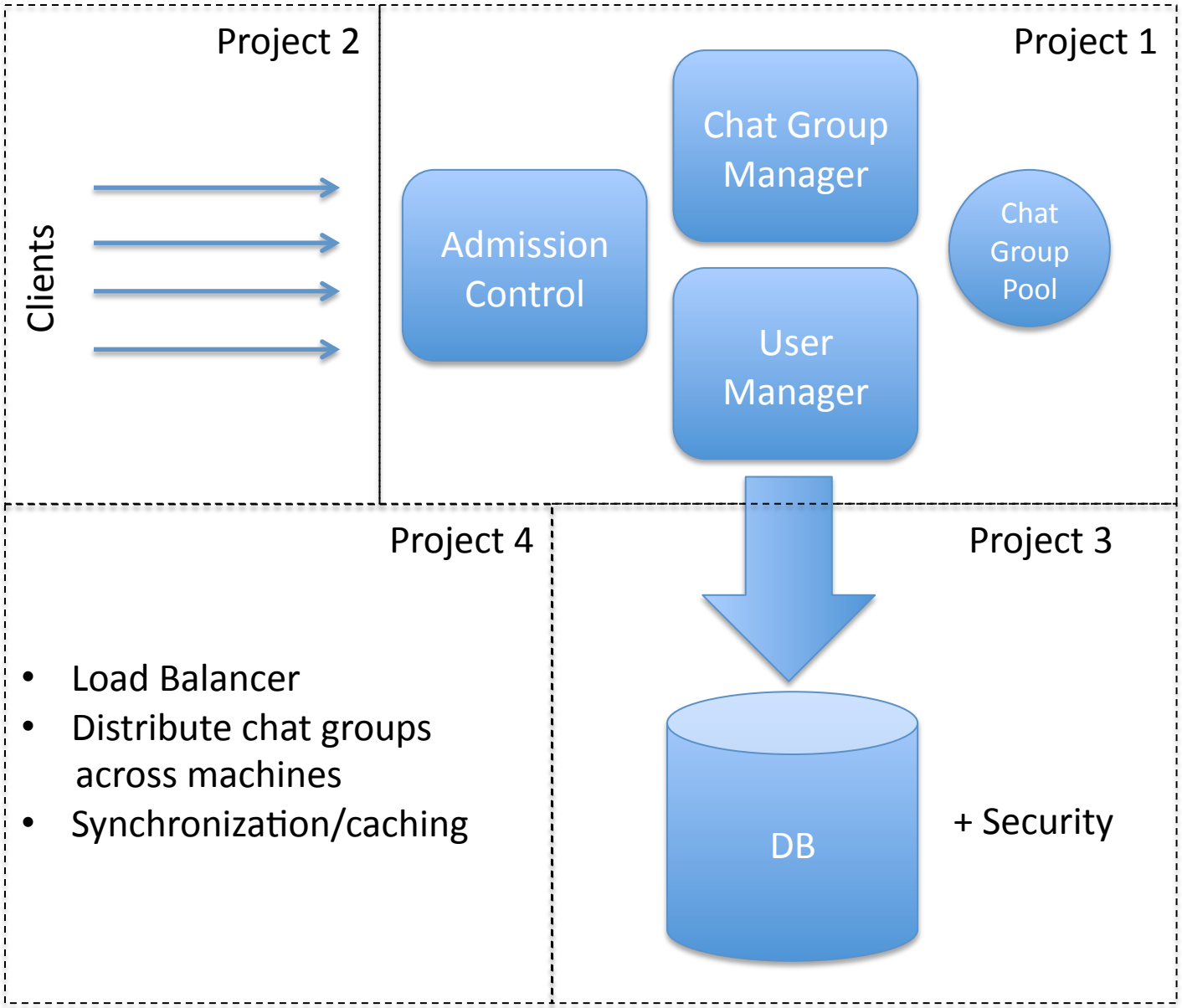
- Goals of projects:
  - Learn to work in teams
  - Use good engineering practices
    - Requirements specification
    - Design Document
    - Implementation
    - Testing
    - [Performance, reliability, ...] analysis
  - Understand lecture concepts at the implementation level
- Your TA will be your guide through this process

A Good Project Lifetime

- Day 0: Project released on course webpage
- Day 1-13: Team meets, discusses and breaks up work on design and necessary prototyping
- Day 14: Final design document due
  - Team reviews the document with the TA
- Day 15: Implementation begins
- Day 20: Implementation is finished. Team switches to writing test cases. Design doc has been updated to reflect the implementation.
- Day 21: Iteration and performance analysis.
- Day 23: Team puts finishing touches on writeup and gets to bed early.

Way to Not have fun

- Day 0: Project design
- Day 13: Oops, we were busy. Throw something together and meet with the TA.
- Day 14: Your TA is “concerned” in your design review.
- Day 21: The l33t hacker in your group makes a pass on the implementation. It mostly works.
- Day 22: Someone else fixes some bugs
- Day 24: Run out of time debugging; don’t make it to analysis or updating design.


Project 2
Project 1
Clients
Admission Control
Chat Group Manager
User Manager
Chat Group Pool
Project 4
Project 3
• Load Balancer
• Distribute chat groups across machines
• Synchronization/caching
DB
+ Security

Project 1: Concurrency

- Topic: concurrency
  - People have wildly different experience with this
  - Most people cannot write correct concurrent code
    - That's just a fact, sorry ☺
- Individual portion
  - Questions about POSIX API
  - Solve a toy synchronization problem
- Group portion
  - Design and implement the core of a parallel chat server
  - Define users, groups, messages
  - Implement message delivery respecting the desired semantics
  - Test your work

Project 2: Network Communication

- Allow clients to connect over the network
  - Design the protocol state machine
    - Connection and assignment to thread
    - Admission control and authentication
    - Steady state
    - Disconnection
  - Always “have an answer” – deal with failure – it can occur ANYWHERE in the protocol
- Implement most of a server, and just enough of a client to fit the testing harness

Project 3: Cloud Technologies

- Deploy your chat server on an EC2 instance
- Add user authentication and logging via MySQL
- Add offline message support
- This might be a “catch up” assignment if you have fallen behind: you will need your codebase to be relatively stable before working on #4.

Project 4: Distributed Systems

- Make your system scale by allowing your system to run on multiple servers
- Design and test a failure protocol for your chat server
  - This is tricky
- Evaluate your design using appropriate metrics and tools.
  - Are there any performance anomalies? Try to explain them and fix the bottlenecks
  - Make sure you drive your system off a cliff

Optional Extra Credit

- Can be added to any of assignment 2, 3, and 4
  - Not worth all that much, we're still working it out
- File transfer
- Management console
- Additional desktop client support
- Connection encryption
- Something else fun...

Tools

- You will use appropriate tools
  - Revision control: svn, git, etc.
  - Build systems: make
  - Test harness: (we will provide)
    - In some cases we will ask you to implement to our interfaces so we can autotest your work
  - Piazzza.com
  - Feel free to use other tools you find useful
    - static analysis
    - debuggers
    - ci tools
    - etc...
- You could also consider using a software engineering methodology to structure your work, *e.g.* agile development; only if it helps you.

Questions about Projects?