



Fall 2009

Dr. Kevin Hamlen



# CS 4485: CS PROJECT

# Schedule

- Next week: Stage 4 description and requirements
- Stage 3 demo in two weeks
  - At this point you should have fixed your Stage 1+2 bugs
    - If you don't yet have a **fully working** Stage 1+2, you are in danger of not being able to make the Stage 3 deadline!
  - Goal: Complete base implementation of Stage 3 by next Friday
  - Debug Stage 3 between next Friday and the demo
    - Leave significant time for debugging! You will need it!
    - Aggressive test suite is critical for a successful Stage 3
- Today: How to make maximal use of this class
  - job interview advice
  - grad school application advice



# Stage 3 Questions?



Today: How to use this class to  
give a killer job interview



# Beyond cs4485

- Scenarios
  - Getting a (better) job: interview advice
  - Getting into grad school: application advice
- Know what (prospective) employers will value from this class and what they won't.
  - which experiences from this class?
  - which skills from this class?
  - which knowledge from this class?

# This class proves that you...

- had a semi-realistic product development experience
  - team management and coordination
  - design decisions (interface, code structure, etc.)
  - implementation decisions (language, tools, etc.)
  - non-trivial testing and debugging
- have applied established software development techniques
  - networking (sockets, streaming, etc.)
  - concurrency (threads, processes, channels, etc.)
  - GUI, API, object-oriented design
- have exposure to aggressive, cutting-edge techniques
  - distributed hash table algorithms (Chord)
  - security analysis and protection systems for p2p networks
- have created a finished product
  - fully functional stand-alone application
  - complete documentation

# What we didn't do

- Formal design
  - class hierarchy diagrams
  - requirements specifications
  - code review
  - take SE4485 for this
- Formal networking protocol design & analysis
  - protocol diagrams
  - proofs of correctness
  - statistic analysis of network latencies, etc.
  - take CS4390 for this
- Formal security analysis
  - formal security policy specification
  - attacker modeling
  - proofs of correctness
  - take CS4393 for this

# Have a story to tell

- Interviewers want to hear stories that relate your experiences and showcase your skills.
- Plan a collection of stories in advance:
  - a team management issue and how you handled it
  - a difficult bug and how you solved it
  - a large project you completed (this one!)
- Find a way to tell your prepared stories no matter what the interviewer asks.
  - They don't really want the answer to their question anyway.
- When telling stories about this class, you'll need to...
  - be able to clearly define p2p and its various varieties
  - argue why p2p is an important and useful technology
  - explain how your work fits into the landscape of past work

# Defining p2p

- Define p2p in 2 sentences or less...
  - Network with NO centralized servers
  - Every node in the network is both a server and a client
- Two main kinds of p2p
  - Unstructured
    - new nodes link wherever convenient
    - topology is usually a giant web of interconnected nodes
  - Structured (our project)
    - new nodes positioned according to some function (e.g., hash function)
    - topology is a ring, or more sophisticated multidimensional structure
- Hybrid networks: some elements of both

# Pitching Peer-to-Peer

- Convincing the interviewer of the merits of p2p
  - (Why is it important to be able to offer a good argument?)
- Potential Applications
  - It's NOT just for file-sharing. It's a general framework for sharing ANY electronic service.
  - Examples: distributed computations, sensor networks, large data sets (think Google), wireless sensor networks, military networks in the battlefield
- Advantages over traditional server-client approach
  - high fault-tolerance
  - naturally load-balancing
  - low-cost deployment, heterogeneous hardware
- Anonymity is NOT a clear advantage—true anonymity is quite hard to achieve in p2p networks!

# Structured vs. Unstructured

- Unstructured
  - more common (Gnutella, etc.)
  - easier to locate geographically close peers near one another in the network topology
  - but query results are not convergent (different querier = different result)
- Structured
  - more experimental (research prototypes)
  - adjacent peers potentially far geographically
  - main advantage: query results convergent

# Why Chord?

- Four main structured p2p network topologies
  - Chord (MIT) [2001]
  - CAN (Intel) [2001]
  - Pastry (Microsoft) [2001]
  - Tapestry (U.C. Santa Barbara) [2001]
- Advantages of Chord
  - formally proved worst-case hop bound of  $O(\log_2 n)$ 
    - (but empirically they're all about the same)
  - very simple core algorithm

# Learn the terminology!

- The terminology you use when talking about your experiences and skills makes a HUGE difference in your interview!
- Compare:
  - “I made a program that lets you download and upload files, and it had security and stuff.”
  - “I designed and implemented a fully **decentralized, structured**, peer-to-peer networking client. It enforced **access control policies** using an **authentication** protocol based on **asymmetric key cryptography**.”

# Important Terms

- (un)structured p2p
- fault tolerance
- load-balancing
- distributed hash table
- query convergence
- data replication
- overlay
- topology
- trust management system
- message-dropping attack
- freeriding / freeloading
- integrity policies
- confidentiality policies
- availability policies
- access control policies
- denial of service attacks
- distributed DoS attacks
- authorization
- man-in-the-middle attack
- privacy
- flood attacks
- (a)symmetric key cryptography

# Find the right focus

- Job interviews for industry positions
  - Interviewer is looking for relevancy of your skills to his particular problem.
  - You cannot be a one-trick pony (e.g., “I only code in C++ on linux with AMD processors and Nvidia graphics cards”).
  - Usually no way to find out in advance exactly what they need. Come prepared for anything.
  - Argue that your experiences demonstrate a versatile range of abilities.
  - Stage 3 is most useful for this kind of interview.
- Applying to graduate schools
  - Professors want students who can think outside the box.
  - Find a way to teach the professor something new.
  - Example: Why is message-dropping such a hard problem compared to centralized networks?
  - Stage 4 is most useful stage for this kind of interview.



# More than just a programmer

- Good programmers...
  - can reliably implement what they're told
  - write scalable, modular, efficient code
  - work well in a team
  - have a comprehensive collection of tools (data structures, algorithms, etc.)
- Good managers...
  - have a high-level understanding of the impact of various design decisions
  - know the past work related to the current project, its pros, its cons, and how it fits together
  - can identify and isolate design flaws in *someone else's* code
  - can *manage* a team effectively
  - know when to take the (rare) leap of creating a *new* technique, and can analyze its performance and ramifications



# Questions and Feedback

- Questions about course material
    - Things about p2p networks / computer security that you've been wanting to ask?
    - Questions about design decisions I made for you?
    - Questions about things we didn't implement?
  - Feedback for me
    - Did you find this course useful?
    - Was the project interesting?
    - Too easy / too hard (so far)?
    - Reasonable / unreasonable work load?
    - What advice should I give future instructors of this course?
    - What advice should I give to the department about this course?
- 