Final Thoughts

Topics in Distributed Systems

- Communication in distributed systems
  - RPCs, message versus stream-oriented communication
- Processes and scheduling
  - Code and process migration, load balancing, scheduling
  - Key idea: move code or processes for performance
- Naming
  - DNS, recursive v/s iterative name resolution
  - Key idea: use hierarchies to handle large distributed systems

Topics in Distributed Systems

- Canonical problems
  - Clock synchronization, logical clocks, leader election, mutual exclusion,...
- Consistency and replication
  - Consistency semantics, web caching
  - Key idea: replicate and locate data close to where it is accessed
- Fault Tolerance
  - Key idea: use redundancy to increase availability

Topics in Distributed Systems

- Security
  - Encryption, authentication,..
  - Security is hard and shouldn't be an after-throught
- Distributed File Systems
  - NFS, Coda, xFS
  - Key idea: Make remote data access possible and efficient
- Distributed Middleware: CORBA, DCOM, Jini
- Other topics: Video-on-demand, Multimedia OS

Follow-on Courses

- CMPSCI 653: Computer Networks
- CMPSCI 754/654: Multimedia Systems
- CMPSCI 515: Computer and Network Security
- Advanced OS
- Sensor Networks
- Linux kernel programming
- Seminars

Five Sermons in Computer Science

- Courtesy: Tom Anderson, Univ. of Washington
- Sermon 1: Simplicity
- Sermon 2: Performance Tuning
- Sermon 3: Programming Craft
- Sermon 4: Information is Property
- Sermon 5: Stay Broad

Sermon 1: Simplicity

- Keep things simple, stupid (KISS principle)
- Simplicity is absolute good, not a tradeoff
  - Forces against simplicity: marketing, ...
- Reasons:
  - Easier to build and maintain, faster, cheaper,..
- Ways to make things simple:
  - Design, then code
  - Think first, act later

Sermon 2: Building High-Performance Systems

- Options:
  - Make every line of code very fast
  - Tune selectively (better)
- Observations
  - 90-10 rule: 90% of time spent in 10% of code
  - Difficult to predict performance problems in advance
- Solutions
  - Measure existing systems (profiling)
  - Modeling
  - Simulate algorithms ahead of time
  - Tuning: build, get it to run, measure, tune bottlenecks
  - Go top-down

Sermon 3: Programming Style

- Question: how to build large software systems in a reasonable amount of time and make them reliable?
- Programming is a craft (art?)
  - Need lot of discipline and structure to make large systems work (well)
- Rule 0: KISS
- Rule 1: Don't over-generalize
- Rule 2: If it is complex, throw it away and start over
- Rule 3: Modularity (and module testing)
- Rule 4: Adopt consistent style
  - Naming convention, file organization, procedure structuring,...
- Rule 5: Don't go for quick, dirty fixes (do it right the first time)
- Rule 6: Document carefully and well
  - Describe high-level idea and then the code
- Rule 7: Quality not quantity is important for documentation
- Summary: Discipline, craft. Take time upfront to save up time later.

Sermon 4: Information is Property

- Intellectual property (IP) issues are important
  - Who own what? What rights do you have?
- Stealing information is still stealing!

Sermon 5: Stay Broad

- Fast moving field.
- New developments all the time
  - XML, Java did not exist 5 years ago
- Solutions
  - Continuing education
  - Education is a life-long process
  - Explore new areas both inside and outside CS
    - Knowledge is power