CS263 - Modern Distributed Systems

September 16, 2003

Matt Welsh
mdw@eecs.harvard.edu

http://www.eecs.harvard.edu/~mdw/course/cs263
What is this course about?

- Modern renaissance in “systems research”
- “Systems” is no longer about the kernel running on your desktop
- Networking vast numbers of computers together yields a new kind of systems science

Harness large amounts of resources for novel applications

- Enormous clusters as scalable Internet services (e.g., Google)
- The Grid: Global network of supercomputers allowing a user virtually anywhere to submit jobs to run on

Napster/Gnutella/KaZaa - pool resources to “share” music

- Napster spawned a new research field
- “Peer-to-peer computing”
- Try to build legitimate applications on this model

Embedding networked systems in the physical world

- Sensor networks allow us to monitor and compute over real physical space at high resolution
The goals of this class

Survey the brave new world of systems research
Read a bunch of papers on exciting new topics
Experiment with sensor networks
Do a research project on your favorite topic
Hopefully publish a paper on your work
Other Systems Classes this Term

CS261 - Advanced Operating Systems (M. Seltzer)
- Focus on historical context and “operating systems”
- Here we focus on networked, scalable, decentralized systems

CS246 - Power-Aware Computing (D. Brooks)
- Really “Advanced Computer Architecture”
- Both hardware and software approaches to energy management

CS222 - Algorithms at the End of the Wire (M. Mitzenmacher)
- An algorithms class, but focuses on great ideas that one can apply to real systems
Running themes

Massive scale
- Internet consists of hundreds of millions of nodes
- Potential number of users is incredibly large
- CNN.com on Sept 11, 2001: 30,000 hits a second

Self-organization and decentralization
- No central authority managing, organizing, or deploying system
- e.g., Gnutella nodes discover each other through broadcasting advertisements
- Any part of the network can be taken down and the rest will survive

Robustness and fault tolerance
- Novel systems not deployed on well-maintained, well-configured hardware in an “engineered environment”
- Systems must tolerate unprecedented degrees of heterogeneity and rate of failure
Large-scale Internet Services

Complex, multi-tiered, clustered systems

- Front-end web server pool
- Cluster of “middle tier” application logic servers
- Back-end database

Great deal of work on scalability, performance, and reliability

- Load balancing across nodes to avoid bottlenecks
- Rapid failover in case of node failure
- Novel directions in data storage and retrieval
Overload in the Internet

Overload is an inevitable aspect of systems connected to the Internet

- (Approximately) infinite user populations
- Large correlation of user demand (e.g., flash crowds)
- Peak load can be orders of magnitude greater than average

Modern Internet services as highly dynamic

- Web servers do much more than serve up static pages
- e.g., server-side scripts (CGI, PHP), SSL, database access
- Requests have highly unpredictable CPU, memory, and I/O demands
- Makes overload very difficult to predict and manage
Massive Overload is Sudden and Unpredictable

September 11 - unprecedented web traffic

- CNN: 20x over *expected peak* - 30,000+ hits/sec
- Grew server farm by 5x by scrounging machines, but still no service for 2.5 hours

USGS site load after M7.1 earthquake

- 3 orders of magnitude increase in 10 minutes, disk log filled up

![USGS Web server load graph](graph.png)
Paper Topics

Clustered search engines
- Inktomi and Google
- “Lessons from massive scale services”

Storage systems
- Distributed Data Structures - novel approach to cluster-based storage
- Porcupine - A clustered, replicated e-mail server

Managing concurrency and load
- SEDA - Event-driven server design for managing massive concurrency
- Capriccio - lightweight threads to simplify server design

Resource management in clustered systems
- Allocation of resources to different competing apps on a cluster
- Develop models of application resource requirements and load
Peer-to-peer computing and the Grid

Huge, decentralized systems formed out of heterogeneous machines scattered across the Internet

- Napster and Gnutella among the first examples
- Lots of recent research to build useful systems on this paradigm

Distributed, massively replicated data storage

- Access your data from anywhere, efficiently, and robustly
- Replicate data across many nodes on the Internet, encode and encrypt to ensure security
- Question - Are wide-area distributed filesystems really compelling?

“The Grid”

- Pool the CPU/memory/disk resources of the world’s supercomputers
- Allow anyone to get access to vast amounts of computing power

Distributed Hash Tables

- All data represented as \((key, value)\) pairs
- Each node in the system associated with a range of keys
- Route messages to the node with the appropriate key
Chord approach

- Nodes associated with address between $0 \ldots 2^N$
- Data stored on node with nearest address
- Nodes maintain set of “fingers” to other nodes
- Route request to nearest predecessor to requested key
Research Challenges

Data caching and replication
- Replicate data to improve resilience to node failure
- Cache frequently requested data items along lookup path

Security and resilience to attacks
- e.g., “Sybil attack” – attacker owns enough nodes in keyspace to foil lookups

Locality and performance
- Reduce number of hops for each lookup, avoid long-distance hops
Paper Topics

P2P storage systems

- Focus on complete, vertical systems: Pond and PAST

Indexing and search in P2P systems

- Going beyond just lookup of documents with keys
- PIER: Run a database on top of a P2P DHT!

Fun and funky P2P Applications

- SplitStream: High-bandwidth content distribution
- Palimpsest: Soft-capacity, best-effort storage for wide area services

Mobile and Pervasive Computing Environments

- Active Proxies: In-network processing to adapt content
- Intentional Naming System: Flexible naming for resources in a mobile environment

The Grid Perspective

- Grid Background by Ian Foster
- The Grid Meets P2P
Sensor networks


Exciting emerging domain of deeply networked systems
- Low-power, wireless “motes” with tiny amount of CPU/memory
- Large federated networks for high-resolution sensing of environment

Drive towards miniaturization and low power
- Eventual goal - complete systems in 1 mm³, MEMS sensors
- Family of Berkeley motes as COTS experimental platform

The Berkeley Mica mote

- ATMEGA 128L (4 MHz 8-bit CPU)
- 128KB code, 4 KB data SRAM
- 512 KB flash for logging
- 916 MHz 40 Kbps radio (100’ max)
- Sandwich-on sensor boards
- Powered by 2AA batteries

Several thousand produced, used by over 150 research groups worldwide

- Get yours at www.xbow.com (or www.ebay.com)
- About $100 a pop (maybe more)

Great platform for experimentation (though not particularly small)

- Easy to integrate new sensors/actuators
- 15-20 mA active (5-6 days), 15 µA sleeping (21 years, but limited by shelf life)
Typical applications

Object tracking

- Sensors take magnetometer readings, locate object using centroid of readings
- Communicate using geographic routing to base station
- Robust against node and link failures

Great Duck Island - habitat monitoring

- Gather temp, humidity, IR readings from petrel nests
- Determine occupancy of nests to understand breeding/migration behavior
- Live readings at www.greatduckisland.net
Vital Dust: Emergency Medical Triage

with S. Moulton, M.D., Boston Medical Center and
M. Gaynor, Boston University

Motes attached to patients collect vital signs (pulse ox, heart rate, etc.)

Ambulance system makes triage decisions, relays to EMTs

PDAs carried by EMTs receive vital signs and enter into field report

Correlate with patient records at hospital

- Patient motes form ad-hoc wireless network with EMT PDAs
- Enables rapid, continuous survey of patients in field
- Requires secure, reliable communications
Sensor network programming challenges

Driven by interaction with environment

- Data collection and control, not general purpose computation
- Reactive, event-driven programming model
- Exploit locality of communication in network

Extremely limited resources

- Very low cost, size, and power consumption
- Typical embedded OSs consume hundreds of KB of memory
- Battery lifetime is the critical resource

Many nodes with sparse and error-prone connectivity

- Reliable communication too expensive
- Ad-hoc formation of communication paths
- Applications must be robust to individual node failure
Very small “operating system” for sensor networks

- Core OS requires 396 bytes of memory

Component-oriented architecture

- Set of reusable system components: sensing, communication, timers, etc.
- No binary kernel - build *app specific* OS from components

Concurrency based on **tasks** and **events**

- **Task**: deferred computation, runs to completion, no preemption
- **Event**: Invoked by module (upcall) or interrupt, may preempt tasks or other events
- Very low overhead, no threads

Split-phase operations

- No blocking operations
- Long-latency ops (sensing, comm, etc.) are **split phase**
- Request to execute an operation returns immediately
- Event signals completion of operation
Web-based interface to building-wide sensor network

- Currently deploying 40 motes throughout Maxwell Dworkin
- Web interface allows job upload and database access to all received messages
- You will prototype your code on a handful of motes, then test on MoteLab
Paper Topics

Overview and Applications
- Part of Jason Hill’s thesis
- Habitat Monitoring on Great Duck Island
- Structural health monitoring

Wireless networking and routing
- Efficient, energy-aware routing protocols
- Directed diffusion: data dissemination and naming
- Reliable transport protocols, ad hoc mobile networking

Localization and Time Synchronization

Storage and Querying
- TinyDB and TAG: Expose sensor network data through a high level query interface
- COUGAR: Another database interface to sensor nets
- Hashtables, GHT, and multiresolution search and storage
Course staff and administrivia

Main contact - cs263@eeecs

Instructor: Matt Welsh (mdw@eeecs)
- Office: 233 Maxwell Dworkin
- Office hours: Thursdays, 10am - 12pm

TF: Geoff Werner-Allen (werner@eeecs)
- Office: 209 Maxwell Dworkin (Systems lab)
- Office hours: Wednesdays, 4 - 6pm
- General course consulting and MoteLab development

All papers, due dates, etc. on course web page:
- http://www.eecs.harvard.edu/~mdw/course/cs263
Readings and reviews

Read and review two papers for each lecture

- Write a 4-bullet summary of each paper
- E-mail to cs263@eecs before class begins
- Main “take away” point of the paper, and 3-4 other main points, either strengths or weaknesses
- Be concise, critical, and thoughtful

Lead the discussion for one (half) lecture

- E-mail us (cs263@eecs) which lectures you prefer to cover
- We will assign lectures to students and e-mail you by end of the week
- If enough students, will run the lectures in pairs
- Prepare a short (20-minute) presentation/discussion notes on the papers for that week
- Focus on raising interesting questions, not summarizing the papers
Paper Reviews

Th 9/18/03 - Overlay and Content Delivery Networks

- (I will lead this discussion)
- Resilient Overlay Networks
- Internet Indirection Infrastructure

Tu 9/23/03 - Scalable Internet Services Overview

- (Need volunteers)
- Lessons from Giant-Scale Services
- Scalable Network Services

Th 9/25/03 - Concurrency Models

- (Need volunteers)
- SEDA - Event driven server design
- Capriccio - Thread based server design
Course projects

Main goal of the course - do some research

- Work either individually or in pairs (pairs preferred)
- Select a juicy research problem that fits in the theme of the course
- CS261 and CS243 students may be able to share the project
- Use the course as a vehicle to further your own research goals

Project proposals (due 9/30)

- Short (2 pages max) summary of what you propose to do, why the problem is interesting, and how you will get started
- Set milestones along the way
- Research project update by 11/20 - short (1-2 page email) telling us where you are at with the project and how you will wrap it up
Course projects continued

Research presentations (last day of class)

- Give a short, fun talk telling us what you did
- Learn from each other’s experiences
- May combine this into a joint session with CS261

Research papers

- Conference-style paper (10 pages max) detailing your project
- Goal is to get used to writing these things - it’s important
- I will work with you afterwards to turn it into a conference/journal submission
Project ideas

- Add resource management and admission control to a thread-based Internet services programming model
- Steganographic networking - hide messages in an innocent looking TCP stream by artificially delaying packets
- Evaluate locality metrics for choosing neighbors in an overlay network
- Market-based programming model for sensor networks
- Extend TinyOS and nesC to support real-time scheduling on motes
- Develop a low-power, probabilistically reliable wireless routing protocol
- Apply great ideas from algorithms to robust routing in P2P or sensor nets
Keys to success

Lots of reading - get used to it!
- (I’ll have to do it too)
- Read critically but efficiently
- Spend more time on the meaty parts of the paper
- **Save your summaries** - start a “paper summary file” and keep adding to it

Don’t get bogged down by trivialities
- Your grade is not (very) important
- “I have never let my schooling interfere with my education” – Mark Twain

Start your research project *early*
- The end-of-course deadline is arbitrary
- Treat this as a forcing function to start a new project, or wrap up an old one
- Combine your project with the research you want to be doing anyway