The Project, in short

- You will work in teams
- to build a distributed system
- doing something **awesome**
- and expanding your knowledge base

The Project

- At least 4 nodes with DS technologies:
  - Communications
    - e.g. protocol transparency, location transparency, source independence, and destination independence
  - Coordination
    - Sharing of some state information to make distributed decisions

The Project: DS Features

- Scope your project to include some of these features:
- Failure handling (node, process, network)
- Multimedia data (audio / video)
  - learn about realtime features
- Bootstrap configuration, but not from a bootstrap server
  - Okay to use DNS solutions

The Project: DS Features (2)

- QoS and Resource management
  - reservations to provide guarantees of performance
- Security
  - online commerce techniques
- Middleware
  - CORBA, Enterprise Java Beans: The power of abstraction in taming DS complexity

The Project: MUST

- Project MUST include some sort of **skills challenge**
  - Something that pushes your group to learn and employ better technology
  - **New programming language (Erlang, Go, Reia, ?)**
    - In general: No Java, C, C++
  - New platform / middleware (CORBA, ?)
  - Research techniques (Content-centric Networking, ?)
  - New hardware platform (Embedded sensor nodes, ?)
    - Android not considered “New”
  - Management/SW process techniques (Agile, ?)

The Project: MUST NOT

- Your project MUST NOT be designed to centralize decision making
  - Don’t use a distributed filesystem / database to let all nodes know all information
  - Don’t send all updates to a central server where decisions are made
- Also, must not use *a. priori.* knowledge of a bootstrap server's IP address

Scaling Analysis

- Must analyze how big N can be
  - Measure your bandwidth usage and patterns
  - Understand your critical path / section
  - Properly use queuing theory, appropriate analytic (prob/stat?) techniques
- Can you state how much better you are than a centralized solution?
- Simulation can be your friend
  - For P2P projects: PlanetSim, P2PSim, PeerSim, ...

How to get an ‘A’ on your Project

- Your team built something that:
  - Works great
    - Perhaps has a bug or two, but you have good workarounds in place
    - and you’ve tested it well enough to be confident in it
  - Fulfills the goals described in your specification
    - You didn’t cut out major parts because you couldn’t get them working
  - Is challenging
    - Challenged the technology skills of all team members
  - Applies DS principles well
    - Implements fundamental algorithms

How to get an ‘A’ on your Project

- You:
  - Were a good team member
    - Contributed often
    - Did lots of work
    - Focused on the mission
    - No team member ever complained about you

How to get a ‘B’ on your Project

- Your team built something that:
  - Mostly works
  - Fulfills some of the goals described in your specification
    - But, you threw out the hard stuff when it didn’t work
  - Is somewhat challenging
    - Written in java, using std libraries or stuff grabbed from the net
    - Connects your 4 laptops together, nothing else
      - And you aren’t sure how it would scale to other machines
  - Applies networking principles well
    - to send all information to a central server / database for action

How to get a ‘B’ on your Project

- You:
  - Were an undistinguished team member
  - Contributed
    - Did lots of work
      - But didn’t quite get it to execute well
      - But not when your teammates needed you

Some Basic Project Rules

- You can continue to refine specifications during the course of the semester
  - Typical: requirements change all the time!
  - A good design can deal with changes (or enhancements to supported requirements)
  - Use layering
    - lawyers are not so good; but layers are!
  - Talk to TA before making major changes

Some Basic Project Rules

- You can propose your own project
  - get credit for doing that project you have been itching to do!
  - talk to faculty members you know: will likely give you a close guide to work with
- Don't panic …
  - … but do start early!
  - We will work with you to get a reasonable project description
  - Reasonable reuse of software from other sources is allowed (foundation classes, algorithms, commercial back-ends, etc.)
    - Re-use of platforms / problems from your research is encouraged

Teams

- Each project team consists of 4 students
- Teams will be assigned
  - Method to this (madness): force you to learn how to talk to other tech folks (yes, just like the real world)
- Manage your group dynamics
  - It's your problem....
  - ...unless you tell me early enough
- Each team will work with a particular TA
  - Meet at least bi-weekly

Team Skills

- Planning
- Expectation-setting
  - Team contracts, ground rules
  - Establish expectations and commitments
- Delegating
  - Know the skills of your teammates

Team Skills (2)

- Leadership
  - Choose leadership style and roles
  - Can be tough in a group of peers
- Teamwork
  - There are a range of roles you will play, all of which contribute towards your team's success

Team Skills

- Conflict Resolution
  - Friction sometimes can be creative
  - Disagreements over technical approach
  - Conflict often comes from communication barriers, so make sure you are listening
  - Can’t get personal, or everyone loses

Handling conflict

- Direct approach: confront the issue
- Bargain: find a compromise
  - Especially good for handling conflicting solutions to a problem or design
- Enforcement of team rules: Only as a last resort, as it causes lots of hard feelings
- Retreat: Avoid or work around the problem
- De-emphasis: Move in a new direction, towards areas of agreement

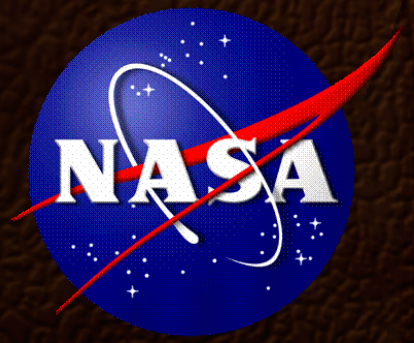
Project Ideas

- **Automotive applications**
  - OpenXC has tools and APIs to develop vehicular applications and access data sources in the car
  - I have a former-student contact at Ford who has offered a development kit to an 842 team
  - openxcplatform.com

Video Management Platforms

- **"Video on Demand" Services**
- **Streaming platforms, esp P2P**
  - Timely delivery of audio/video streams
  - Cached near the point-of-use to minimize/eliminate congestion
- Can you outdo these guys?

- **Exploit Skype Infrastructure**
  - Skype is a very successful P2P system
  - Has an open API. Leveragable?
  - Could you backup your computer to your Skype contacts?
- **P2P Data Analysis**
  - To what degree could “big data” benefit from P2P computation?
    - Ex: http://nssdc.gsfc.nasa.gov/ for Astronomy data



- **Distributed Collaboration**
  - Audio / video conferencing: Many-to-many
  - Collaborative whiteboarding w/audio support
  - Distributed chat: support text, graphics, audio, video?
  - Minimize BW needs with smart video and audio
- **Distributed "Active Network" Services (SDN)**
  - We inject postscript into printers as computer programs
  - Similarly, send new routing, transformation, filtering policies to routers and switches within networks
  - The whole internet becomes programmable!
  - Quality adapts to available BW, display size, etc.

Network and Resource Management

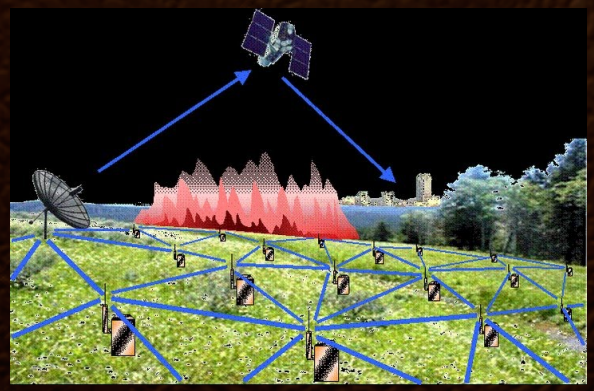
- **Distributed Quality of Service Management**
  - QoS tradeoffs have to be made all the time!
  - infrastructural support for delivering variable quality of service based on BW availability, fees charged, etc.
  - billing and tracking support
- **QoS aware End to End Applications**
  - build applications like videoconferencing “automatically” adapt to congestion conditions
  - Treat premier customers with higher QoS
  - Time-based/bit-based charging services

Infrastructure Support

- **Visualization/Control of Distributed Execution**
  - A debugging/monitoring environment for distributed systems
  - Log events and resource usage locally
  - Log userlevel events and kernellevel events (by emulation if needed)
  - Batch and ship events to a remote console to visualize them on a single screen
  - Add 2way communications to each machine from monitoring machine to stop/start processes, add/filter events
  - Aids performance tuning and visualization

- **Multicast Support**
  - Multicast IP and RealTime Protocol (RTP)
  - Session Initiation Protocol (SIP)
  - synchronize multiple streams, merge multiple streams
  - split a merged stream into components
- **A "PushPull" communications architecture**
  - may or may not be modeled after the publisher/subscriber
  - server can “push” data to subscribed clients on regular basis
  - a client can “pull” data from server on demand
  - support for different types of data

- **Wireless Sensor Networks**
  - Large numbers of physically distributed sensors
    - Including cameras, microphones, bio-sensors, chemical sensors, etc.
  - Distributed and centralized/replicated processing of audio/video for “special” events, faces, etc.
  - Persistent logging of all events with easy retrieval
  - Flexible querying of monitored data (“where is Bill now?”)
  - Levels of access control
  - Flexible front-ends
  - Remote administration



- **Smart Electronics for the Home**
  - Wireless/wired extremely reconfigurable devices for the home
  - Any-to-any communications and control
  - Standard pluggable interfaces
- **Bioengineering Data Manipulations**
  - Folding@Home
  - Mummer (Genome sequence alignment)
    - mummer.sourceforge.net

Other Possibilities

- Distributed OS-kernel coordination
- Projects from research groups
- Projects from potential employers
- P2P systems
- Social networking
  - FaceBook & Mobility
- Vehicular networks
  - V2V and V2I

Other Possibilities

- Project based on OpenFlow switch
  - see openflowswitch.org
- Project using DCCP (UDP + congestion control)
- DSM where memory pages are shared using a P2P network (challenging, potentially publishable)
- Secure, decentralized, name-system
  - See www.aaronsw.com/weblog/squarezooko