COMP 734
Fall 2008

Project Information
October 14, 2008

All projects must meet ***all the following criteria***:

- includes the design and implementation of nontrivial distributed protocols and/or algorithms that are related to the course content,
- includes multiple distributed processes that execute concurrently, maintain some system state, and use the protocols or algorithms above,
- provides some form of highly-available operations, and
- the amount of work required is roughly the same per person for all projects.

Often students are able to meet these criteria by proposing new distributed functions for systems they are working on for their RA jobs or by extending projects from other courses. If you want to work on a project that involves topics that will not be covered in the lectures and readings until late in the course, I will provide you with all the reference material you need.

Read carefully – instructions with deadlines

Projects will be done by teams of two students (other arrangements will also be considered). The first step is to develop a new project idea or define a project based on ideas similar to the ones from prior classes. Each project should be scaled so it can be completed in the allotted time. The second step is to send me email identifying your proposed project topic (a 1-2 sentence brief description) and names of team members. I should receive your email no later than **6:00 PM on Thursday, October 30, 2008**.

Each project should also schedule a 30 minute (or so) meeting with me. At that meeting we can discuss the goals, scope, ambitions, degree of difficulty, etc., of your project. **You should come to this meeting with an initial design sketch described in a 2-3 page (approximately) document**; this means you should do some investigation and planning for defining and implementing your project goals. The purpose of this meeting is to ensure that your project can meet all the four criteria listed above and be done in the time available. These meetings can take place any time between **now** and **6:00 PM on Thursday, November 6, 2008**. The goal is to have final project definitions completed no later than that date. Obviously it is in everyone's interest to complete this process ***even sooner*** so programming can begin. Your project demonstration will be scheduled sometime during exam week of December 8-12, but you should plan on having your implementation completed by December 8.

Past COMP 734 (243) Projects (A new design and implementation providing similar functions is allowed)

- A distributed simulation of a BotNet to investigate issues such as robustness to node failures and anti-infection measures, as well as determine rates of compromise.

- A Web Proxy service that automatically balances load among a set of servers that logically implement a single proxy.

- A collaborative whiteboard application using two-phase locking to implement synchronization for access to objects on the whiteboard.

- A peer-to-peer backup system that allows users to take advantage of spare disk capacity on peer systems for backing up their files; based on the Napster system model.

- A music file sharing service based on combining a search model similar to that of FastTrack (KaZaA) with a multi-source downloading model similar to that of BitTorrent.

- An access control system based on Chord that maintains a reliable mapping of users, objects, and the permissions users have on those objects.

- A system for supporting multiple agents that migrate from network node to network node and perform services for program written in Perl.

- A Peer-to-Peer file-sharing service for large content objects in which the files are fragmented and distributed over multiple nodes in a distributed hash table.

- A highly-available web proxy server that uses replication of web objects in a primary-backup design.

- An implementation of the Chord protocols suitable for use in a DNS-like name resolution system.

- An implementation of a distributed file storage system using Chord protocols..

- An extended version of the FastTrack (KaZaA) protocols to improve performance and availability.

- A highly-available (replicated data) system for storing and retrieving text annotations for web pages based on Chord where the key is the page URL.

- A highly-available server-driven “spidering” system for retrieving, collecting, and organizing information available in the *Friendster* network.

- A peer-to-peer system that brokers CPU time, allowing users to “sell” idle time on their machines and purchase (and use) idle time on remote machines.

- A network of cooperative web proxy caches that use the Gnutella peer-to-peer protocols for self-organizing and handling location queries.

- A service to support collaborations in a classroom setting with functions such as note sharing, test administration, dynamic demonstrations, chat rooms, etc.

- A database query proxy that caches results from queries and reuses them as all or part of the response to new queries.

- A peer-to-peer music file-sharing service based on distributed hash tables that are replicated on multiple nodes for high availability.

- A distributed highway traffic reporting system for providing real-time traffic-flow maps to drivers of cars equipped with GPS receivers and wireless Internet connections.

- Several multi-player distributed board games (Blackjack, Let it Ride, Evolutionary War, Monopoly, Texas Hold’em, Gobang (a Japanese game)).

- Networked multi-player real-time arcade games such as “Astroids”, “Air Combat” and “Dueling Teapots” (graphics students will appreciate the whimsy in this one).

- A service for distributing graphics processing loads over multiple processors in a tele-immersion environment.

- A service providing voice teleconferencing for a group of users that have computers equipped with sound cards.

- An Internet voice mail server that allows users with a computer equipped with a sound card to login over the Internet and leave or listen to voice messages.

- A Web-based lecture delivery system that allows a teacher to present lecture slides (expressed as web pages) to multiple students concurrently over the Internet and to interact with the students through “chat” windows.

- A collaborative spreadsheet application that allows multiple users to concurrently view and update data in a shared spreadsheet.

- A collaborative data-storage browsing application that allows multiple users to concurrently view and update state information in an object storage system.