Syllabus **CS 603 Advanced Topics in Distributed Systems** **Spring 1997**

Instructor : Kihong Park
Class : MWF 3:30–4:20 (CS G066)
Office Hours : MW 2:30–3:30, R 2–3 (CS 126)
E-mail : `park@cs.purdue.edu` (Tel. : 494–7821)

Teaching Assistant : Niaz Pavel
Office Hours : MW 11:30–12:30, F 2:30–3:30 (CS G016)
E-mail : `nrp@cs.purdue.edu` (Tel. : 494–6002)

Course Content : Advanced graduate course in distributed system design and analysis.

Prerequisites : Operating systems (CS 503); computer networks (CS 536); solid background in C, C++ and UNIX (experience with projects and system programming); sound mathematical preparation (calculus, discrete math, differential equations, probability and statistics).

Text Books : Required - *Distributed Operating Systems.* Tanenbaum, Prentice-Hall, 1995. Recommended - *Distributed Algorithms.* Lynch, Morgan Kaufmann Publ., 1996; *Internetworking with TCP/IP*, Vol. III. Comer & Stevens, Prentice-Hall, 1996. The textbooks, including Tanenbaum, are to be viewed as reference materials; the lectures are the main source of information. Attendance is crucial for the successful completion of the course.

Work Load and Grading Policy : Homework assignments (40 %), midterm (30 %), research project (30 %). The research project is due at the end of the semester; project topics will be determined by the halfway point (in the semester). Although several topics will be suggested, students are encouraged to seek out and explore interesting subject matters on their own.

Homework assignments are given out weekly; they are due in class the following week. Late homeworks will not be accepted. Discussion of initial ideas and approaches regarding homework assignments is fine. *Collaboration* on homeworks (i.e., jointly exploring the specifics of a solution), however, is not allowed. Academic dishonesty is a serious matter and is dealt with in accordance with University policies.

Assignments involving code must be made available *one day after the due date* in a directory named ~login-name/cs603/subdirectory-name for possible inspection by the TA.

Lab Space : Our main programming and implementation platform will be the Xinu Lab. Accounts, if not already set up, should be applied for with Candy Walters (CS 210).

Goal of the Course :

The primary goal of this course is to study the design principles underlying distributed systems—distributed operating systems, network of workstations (NOWs), network computing (e.g., PVM, Java), and other distributed computing environments. The first half of the semester will be devoted to understanding the fundamentals of distributed systems design—synchronization/concurrency control, network communication,

scheduling/load balancing, threads, real-time issues, naming, replication/cache consistency, fault-tolerance, distributed shared memory, and security. Emphasis will be placed on performance issues including trade-offs in design (e.g., computation/communication ratio, user space vs. kernel space implementation), and their analysis.

In the second part of the semester, we will study several state-of-the-art research topics including parallel distributed computing, microeconomic approaches to resource allocation (e.g., CPU, bandwidth), on-line algorithms for load balancing and communication management, O.S. support for high-speed networking, adaptive algorithms for server selection, fault-tolerance in Internet-like large-scale network environments, and case studies of distributed systems.

Course Outline :

- Introduction
- Network communication and client/server design (e.g., socket programming, RPC, group communication)
- Synchronization and concurrency control
- Load balancing and distributed scheduling (e.g., threads, real-time scheduling)
- Naming, replication, and cache consistency (e.g., distributed file system structure & design)
- Distributed shared memory, fault-tolerance, and security
- ——*Midterm*——
- On-line adaptive resource control
- O.S. support for high-speed networking and multi-media communication
- Microeconomic approaches to resource allocation (e.g., load balancing, end-to-end quality of service)
- Parallel distributed computing
- Case studies (e.g., Amoeba, MOSIX, PVM)
- ——*Project due*——