Department and Course Number	**CEG 435**	Course Coordinator	Thomas C. Hartrum
Course Title	**Distributed Computing and Systems**	Total Credits	4

BS CE: Elective; BS CS: Elective.

This document was prepared by: Thomas C. Hartrum Date: May 09, 2005

Catalog Data

Study of process coordination, client-server computing, network and distributed operating systems, network and distributed file systems, concurrency control, recovery of distributed transactions, and fault-tolerant computing. **Prerequisite**: CEG 434 or equivalent.

Text Books and Other Source Materials

1. Andrew S. Tanenbaum and Maarten van Steen, ***Distributed Systems Principles and Paradigms***, 2002: Prentice-Hall, ISBN 0-13-088893-1.

2. David Reilly and Michael Reilly, ***Java Network Programming and Distributed Computing***, 2002: Addisno-Wesley, ISBN 0-201-71037-4.

Home Page: http://www.cs.wright.edu/~thartrum/CEG435SP05/intro635.html

Learning Objectives

The student should have learned the following:
1. Different architectures of distributed systems.
2. The basic communications mechanisms supporting distributed systems.
3. Client-Server architectures and software design issues
4. Remote procedure calls, Java RMI, and underlying design issues
5. Synchronization issues in a distributed context.
6. Use of threads in a distributed context.
7. Models for code migration.
8. Concepts of name spaces and name resolution methods.
9. Different consistency models and the need for them.
10. Different models for replication and distributed update

Prerequisites by Topic
1. Memory Management.
2. File Systems.
3. CPU Scheduling.
4. Synchronization.
5. Shared Memory.
6. Interprocess Communication with Sockets.
7. Threads.
8. Deadlock.

Course Content

1. Introduction, Communications
2. Java, Client-Server
3. Remote Procedure Call, Remote Method Invocation
4. Processes and Threads
5. Process Migration, Software Agents
6. Midterm Exam, Naming
7. Application Protocols, Synchronization
8. Consistency and Replication
9. Advanced Topics

Class/Laboratory Schedule

Each week has two lectures of 75-minutes each. There is no scheduled lab. Students are expected to work in open labs for no less than 2 hours a week.

Laboratory Projects

The laboratory portion, which counts 35% of the course grade, consists of two lab exercises and one project. The project is done in three pieces, each of which builds on the previous one. The lab exercises are worth 10 points each, and the project sections are worth 30 points, 35 points, and 15 points, respectively, for a laboratory total of 100 points.

The lab exercises and project are evaluated on the following criteria: (1) execution (the code must compile and execute); (2) correctness (meets the stated specification); (3) documentation; and (4) programming style. The lab exercises are individual efforts, done only by the student (except for code provided by the instructor). The project is done individually or by teams of two or three persons.

Contribution to Professional Component

CEG 435 contributes 4 hours to Criterion 4(b), including engineering design.

Course Contribution to Program Educational Objectives

CEG 435 contributes to Objectives 1 and 2. By studying various design concepts and tradeoffs in distributed systems, it develops design skills in the students. Detailed examination of distributed architectures and algorithms, along with the projects, extends the students' implementation abilities. The students acquire design and programming skills applicable to other, more advanced courses.

Course Contribution to Program Outcomes and Assessment

a	b	c	d	e	f	g	h	i	j	k
PXX	PX	PXX	0	PX	0	PX	PX	PX	PX	PXX

Estimate ABET CAC Category Content

	Core	Advanced			Core	Advanced
Data Structures				Concepts of PL		0.5
Algorithms		1.0		Comp Organization + Architecture		0.5
Software Design		1.0		Other		1.0

Oral and Written Communications

There are no oral presentations. Students submit documentation in all phases of their project. Although documentation is part of the basis for grading, we do not claim that it constitutes written communications.

Social and Ethical Issues

None.

Theoretical Content

Although for the most part the course is taught at a more pragmatic, design level, there is some theoretical content. Mathematical analysis is done is the context of clock synchronization. Graph-theoretic algorithms for process partitioning are discussed. Overall such mathematical analysis makes up around 10% of the course.

Problem Analysis

The project involves a distributed simulation. It is scoped in size and sophistication to fit a 10-week course. Detailed analyses of the user requirements are performed by the students before implementing

Solution Design

Much of Tananbaum's text deals with design alternatives for distributed systems, and evaluation criteria for making design decisions. The project involves a distributed simulation. It is scoped in size and sophistication to fit a 10-week course. While the primary goal of the project is to provide experience in working with distributed systems, and the requirements are relatively well structured, there is some latitude for the students to make design decisions and to apply alternative approaches to the design of parts of the system.

Desired Outcomes

The student should be able to:

1. Understand the benefits and some of the problems with distributed systems.
2. Design and implement a client-server program.
3. Demonstrate an understanding of message-passing issues in a distributed system.
4. State and resolve the problems of synchronization of distributed processes.
5. Describe how name space resolution works.
6. Deploy a specific consistency model as needed in the application.

Outcome Measures and Assessment

Students are assessed by several homework problems (10%) and two exams (25% and 30% respectively). In addition, application of the concepts covered in class is assessed by programming labs and the project design and implementation (35%). There is a self-assessment conducted at the beginning of the course, and another at the end. These forms are included below.