School of Computing
UNIVERSITY OF GEORGIA

# *Course Information Sheet*
# CSCI 4780
Distributed Computing Systems

| | |
|---|---|
| **Brief Course Description** (50-words or less) | The fundamental concepts in distributed computing and the practical techniques for building distributed systems. Topics include distributed computing models, naming, synchronization, replication and consistency, fault tolerance, and security. Widely deployed distributed systems are used as case studies. Students design, implement, and analyze prototype systems. |
| **Extended Course Description / Comments** | This course is targeted for undergraduates in their junior/senior years. |
| **Pre-Requisites and/or Co-Requisites** | CSCI 2720: Data Structures<br>And either<br>CSCI 4730: Operating Systems OR CSCI 4760: Computer Networks |
| **Required, Elective or Selected Elective** | Selected Elective Course |
| **Approved Textbook** | Author(s): Andrew S. Tanenbaum and Maarten Van Steen<br>Title: Distributed Systems: Principles and Paradigms<br>Edition: Second Edition<br>ISBN-13: 0-13-239227-5 |

**Specific Learning Outcomes (Performance Indicators)**

This course presents the fundamental concepts in distributed computing systems. At the end of the semester, all students will be able to do the following:

1. Define, identify and distinguish various types of transparencies, and analyze their importance for various distributed applications
2. Develop (in multi-student team) software prototypes applying variety of distributed system architectures.
3. Define, recognize and distinguish various types of communication (synchronous, asynchronous, persistent, transient).
4. Explain the steps of a remote procedure call (RPC)
5. Develop (in multi-student team) a multi-threaded software that applies thread synchronization functionality (locks and barriers).
6. Outline the steps involved in resolving flat/structured names using recursive and iterative approaches.
7. Develop (in multi-student team) a distributed software for resolving flat/structured names
8. Hand-simulate operations of logical clocks (Lamport and Vector clocks) and present the outcomes to a technical audience.
9. Define and distinguish various types of data consistency models

10. Analyze and present the design and internal workings of a distributed computing application to a professional audience.

**ABET Learning Outcomes**

A. Graduates of the program will have an ability to: Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
B. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
C. Communicate effectively in a variety of professional contexts.
D. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
E. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
F. Apply computer science theory and software development fundamentals to produce computing-based solutions.

**Relationship Between Student Outcomes and Learning Outcomes**

| | ABET Learning Outcomes | | | | | | |
|---|---|---|---|---|---|---|---|
| **Specific Learning Outcomes** | | A | B | C | D | E | F |
| | 1 | ● | ● | | | | ● |
| | 2 | | ● | | | ● | ● |
| | 3 | ● | | | | | ● |
| | 4 | ● | ● | | | | ● |
| | 5 | ● | ● | | | ● | ● |
| | 6 | ● | | ● | | | ● |
| | 7 | | ● | | | ● | ● |
| | 8 | ● | | ● | | | ● |
| | 9 | ● | | | | | ● |
| | 10 | ● | | ● | | | ● |

**Major Topics Covered**

1. Goal and Types of Distributed Systems (Usage)
2. Distributed System Architectures (Assessment)
3. Threads and processes in distributed systems (Assessment)
4. Communication models (Usage)
5. Remote procedure calls and sockets (Assessment)
6. Naming and name resolution (Assessment)
7. Distributed hash tables and consistent hashing (Assessment)
8. Physical and Logical Clocks (Usage)
9. Mutual Exclusion and Leader Election (Usage)
10. Data Centric Consistency Models (Familiarity)
11. User Centric Consistency Models (Familiarity)
12. Replica Management (Usage)
13. Reliability in Distributed Systems (Usage)
14. Distributed Commit Protocols (Familiarity)
15. Distributed File Systems (Familiarity)

16. World Wide Web-based Systems (Familiarity)

**Knowledge Levels**

The following is the ACM's categorization of different levels of mastery: Assessment, Usage, and Familiarity. Note that Assessment encompasses both Usage and Familiarity, and Usage encompasses Familiarity.

**Familiarity:** The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept as opposed to expecting real facility with its application. It provides an answer to the question "What do you know about this?"

**Usage:** The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question "What do you know how to do?"

**Assessment:** The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question "Why would you do that?"

**Course Master** Dr. Lakshmish Ramaswamy

**Modified** 3/13/2024 by Dr. Ramaswamy