# 15-440: Distributed Systems
## Syllabus

School of Computer Science
Carnegie Mellon University, Qatar
Fall 2014

# 1 Overview

**Title:** Distributed Systems
**Units:** 12 units
**Pre-requisites:** A grade of "C" or better in 15-213, Introduction to Computer Systems
**Lectures**: Monday and Wednesday, 9:00 – 10:20 AM, Room 1030
**Recitation**: Thursday, 9:00 – 10:20 AM, Room 1030
**Webpage**: http://www.qatar.cmu.edu/~mhhammou/15440-f14/

**Description:**

15-440 is an introductory course in distributed systems. The emphasis will be on the techniques for creating functional, usable, and high-performing distributed systems. To make the issues more concrete, the class includes several multi-week projects requiring significant design and implementation.

The goals of this course are twofold: First, students will gain an understanding of the principles and paradigms that underlay distributed systems, such as communication across networks, concurrency, synchronization, consistency and fault-tolerance. Second, students will gain practical experience in designing, implementing, and debugging real distributed systems.

The major themes this course will teach include process distribution, communication, naming, abstraction and modularity, concurrency, scheduling, resource sharing, locking, consistency and replication, failure handling, distributed programming models, distributed file systems, protection from accidental and malicious harms, virtualization, and the use of instrumentation, monitoring and debugging tools to solve problems at large-scale. As the creation and management of software systems are fundamental goals of any undergraduate systems course, students will design, implement, and debug large programming projects. Students will learn some of today's most popular distributed systems, such as Google File System, MapReduce and GraphLab.

**Instructor:**

Mohammad Hammoud
mhhammou@qatar.cmu.edu, Room 1006, 4454-8506, Office Hours: Wednesday, 4:30PM – 5:30PM

**Teaching Assistant:**

Dania Abed Rabbou
dabedrab@qatar.cmu.edu, Room 1005, 4454-8590, Office Hours: Tuesday, 9:30AM – 11:59AM; Thursday, 10:30AM – 11:59AM

# 2 Objectives

Distributed systems combine the computational power of multiple computers to solve complex problems. The individual computers in a distributed system are typically spread over wide geographies, and possess heterogeneous architectures and operating systems. Hence, an important challenge in distributed systems is to design system models, algorithms and protocols that allow computers to communicate and coordinate their actions over heterogeneous networked computers so as to solve large-scale problems.

Our aim in this course is to introduce you to the area of distributed systems. You will examine and analyze how a set of networked computers can form a functional, usable and high-performing distributed system.

The course has three major goals:

- To learn the principles, architectures, algorithms and programming models used in constructing distributed systems.
- To examine state-of-the-art distributed systems, such as Google File System.
- To design, implement and debug real distributed systems.

Through these objectives, the course will transform your computational thinking from designing applications for a single computer system, towards designing distributed applications that run on networked computers.

# 3 Learning Outcomes

The course encompasses two main learning outcomes:

1. Students will identify the core concepts of distributed systems; that is, the way in which several machines can be orchestrated to **correctly** solve complex problems in an **efficient**, **reliable** and **scalable** manner.
2. Students will examine how existing systems have **applied the core concepts** of distributed systems, and will additionally apply such concepts in developing sample systems.

## 3.1 Understanding the Core Concepts of Distributed Systems

Students will learn the core concepts that comprise any distributed system. They will recognize the system constraints, trade-offs and appropriate techniques for building distributed systems that best serve the computing needs of different classes of applications. In particular, students will learn the following concepts:

- Access and location transparency
- Task parallelization
- Fault-tolerance
- Security

### 3.1.1 Access and location transparency

Exposing the capabilities of machines, yet hiding their details is one of the first steps in designing distributed systems. Such systems penetrate economies and masses which transparently leverage their powers. For instance, in the Internet, which is a successful distributed system, a simple browser interface will allow you to explore information

scattered over wide-geographies. In this course, students will examine how to **abstract** data and machine locations (which may reside at different physical places) as well as data and machine replications.

Specifically, students will study the following topics:

- **Processes and Communication:** Students will explain and contrast the communication mechanisms between different processes and systems.
- **Naming:** Students will identify why entities and resources in distributed systems should be named, and examine the naming conventions as well as some naming resolution mechanisms.

### 3.1.2 Task parallelization

Traditional algorithms that work on a single processor are inefficient – or even fail to work – in a system where multiple machines are working in parallel. In distributed systems, problems/jobs can be solved using parallelization. Generally a job is split into multiple tasks, and all tasks are executed in parallel on different machines. The tasks may access common resources, such as data contained in a shared file. Consequently, two main challenges emerge. First, we ought to ensure that the concurrently running tasks are coordinated and synchronized in a manner that correctly achieves the job's goal. Second, we can potentially replicate and place resources across multiple computers in a way that allows tasks to access them more effectively.

Specifically, students will study the following topics:

- **Concurrency and Synchronization:** Students will identify issues on how to coordinate and synchronize multiple tasks in a distributed system.
- **Consistency and Replication:** Students will understand how replication of resources can optimize performance and scalability, as well as examine various models that allow maintaining consistency of replicated data.

### 3.1.3 Fault-tolerance

In distributed systems, a failure of a single or a part of a computer (or what is known as *partial failure*) is very likely. If such a failure is not tolerated, the whole system might come to a grinding halt or result in a random and unpredictable behavior. Students will learn how to avoid and recover from partial failures, a concept referred to as fault-tolerance.

### 3.1.4 Security

In distributed systems, computers that serve in solving your problem may not be under your administrative control. This makes a distributed system vulnerable to security and privacy issues. Students will learn the common security issues in distributed systems and some mechanisms that can be used to secure distributed systems.

## 3.2 Practical Application of the State-of-the-Art Distributed Systems

Students will also learn how to apply principles of distributed systems in a real-world setting. In particular, they will learn the following topics:

- **Programming Models:** Students will learn some of the programming models which can be adopted in distributed systems such as MapReduce, GraphLab and Pregel. These programming models allow developers to easily program distributed jobs, while ensuring correctness, fault-tolerance and efficiency.

- **Distributed File Systems:** Students will learn how a file can be striped and placed anywhere in a distributed system (or what is referred to as *distributed file system*), yet be accessed transparently- as if it is a local file. They will examine how to apply distributed system principles to ensure transparency, consistency and fault-tolerance in distributed file systems.

- **Virtualization:** Students will learn the concept of *system* virtualization, where a state of a computer is abstracted from the underlying hardware. This allows masking the heterogeneity of the machines that comprise a distributed system, besides increasing overall system utilization and security.

# 4 Textbooks

The primary textbooks for this course are:

- Andrew S. Tannenbaum and Maarten Van Steen, "***Distributed Systems: Principles and Paradigms"***, Second Edition, Pearson, 2007.
- George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair, "***Distributed Systems: Concepts and Design"***, Fifth Edition, Addison Wesley, 2011.

In addition, we recommend the following text books:

- Randal E. Bryant and David R. O'Hallaron, "***Computer Systems: A Programmer's Perspective"***, Prentice Hall, 2003.
- Tom White, "***Hadoop: The Definitive Guide"***, Second Edition, O'Reilly Media, 2010.

We have several reference books in the library covering most of the course topics. We will also be reading book chapters (written by the Instructor), tutorials, and research papers on the covered topics.

# 5 Course Organization

The participation of students in the course will involve five forms of activities:

- Attending lectures and recitations
- Solving assignments (including writing and reading assignments)
- Solving large programming projects
- Taking exams and quizzes
- Participating in class discussions

# 6 Assessment

Each student will receive a numeric score with a corresponding letter grade, based on a weighted average of the following:

1. **Projects:** The projects will count for a total of 45% of your final score. There will be **4** projects throughout the course. All projects are individual projects (i.e., no teams can work on the same project). The first project is worth 15% and the last three projects are worth 10% each.

You are encouraged to submit the projects on time. For all projects except the final one, the following rules apply. If you submit one day late, we will deduct 25% of the project score as a penalty. If you submit two days late, 50% will be deducted. The project will not be graded (and you will receive a zero score on the project) if you are more than two days late. However, there is a **grace-days quota** for projects. In particular, you will be given **3 grace days** for all projects, except for the final one. You can use the grace days as needed. For instance, you can submit your first project three days late and still not receive any penalty. In this case, you will be penalized starting from the $4^{th}$ day after the deadline. Furthermore, when you consume all your grace days, you will be left with no grace days for the rest of the projects.

Note that the final project is unique in two aspects. First, you cannot use grace days for the final project. As such, if you are left with some grace days before the final project, you will lose them all. Hence, plan how to utilize your grace-days quota judiciously. Second, there will not be any penalty system for this project either. That is, if you are one day late in submitting the project, it will not be graded and you will receive a zero score on it.

2. **Exams:** There will be two in-class exams – midterm and final – which combined will count for 25% of your final score. The midterm is worth 10% and the final is worth 15%.

3. **Problem Solving Assignments:** There will be 5 assignments that will test you on problem analysis and solving skills. These assignments will altogether carry 15% of your final score.

4. **Quizzes:** There will be 2 quizzes, which combined will count for 10% of your final score. These quizzes are meant to test your understanding and preparation for the concepts covered throughout the course.

5. **Class/Recitation Participation and Attendance:** Your attendance of both, classes and recitations, as well as your participation in discussions during presentations will count for 5% of your final score.

To this end, Table 1 shows the breakdown of the five forms of activities that the course involves, alongside the quantity and the overall weight of each activity. Take into account that small differences in scores can make the difference between two letter grades. Letter grades will be determined by absolute standards. The total score will be plotted as a histogram. Cutoff points are determined by examining the quality of students' work on the borderlines. Individual cases, especially those near the cutoff points may be adjusted upward or downward based on factors such as attendance, class participation, improvement observed throughout the course, exam performance, and special circumstances.

| Type | # | Weight |
|---|---|---|
| Projects | 4 | 45% |
| Exams | 2 | 25% |
| Problem Solving Assignments | 5 | 15% |
| Quizzes | 2 | 10% |
| Class/Recitation Participation and Attendance | 42 | 5% |

**Table 1**

# 7 Getting Help

For urgent communication with the instructor and the teaching assistant, it is best to send an email (preferred) or give a phone call. If you want to talk to any of them in person, remember that their posted office hours are merely nominal times when they guarantee that they will be in their offices. You are always welcome to visit them outside of their office hours if you need help or want to talk about the course.

We ask that you follow a few simple guidelines. The instructor normally works with his office door being open. Whenever the office door is open, he welcomes visits from students. However, if his office door is closed, this means that he is busy with meetings or phone calls, thus prefers not to be disturbed.

We will use the course webpage as the central repository for all information about the class. Through the webpage, you can:

1. Obtain copies of any handouts or assignments. This is especially useful if you miss a class or lose a document.
2. View announcements that relate to the course.
3. Find links to any electronic data you need for your assignments.
4. Read clarifications and changes made to any assignments, schedules, or policies.
5. Provide healthy feedback about the course.

Lastly, you can use **Piazza** for asking questions and receiving answers without all the emails! Posting your questions on Piazza will help the whole class benefit and will certainly avoid redundancy. Find our class Piazza page at:
https://piazza.com/qatar.cmu/fall2014/15440/home

# 8 Policies

### Working Alone on Assignments/Projects

Assignments/projects that are assigned to students should be performed individually. This course does not include any team project or assignment.

### Handing in Assignments/Projects

All assignments/projects are due at 11:59PM (one minute before midnight) on the specified due date. All hand-ins are electronic and should be submitted using the AFS file system:

/afs/qatar.cmu.edu/usr10/mhhammou/www/15440-f14/handin/*userid*/, where *userid* is your andrew user id.

### Making up Exams, Assignments and Projects

Missed exams, assignments and projects can be made up on a case by case basis, but only if you make prior arrangements with the instructor. However, you should have a good reason for doing so. You need a written consent from the instructor for making up exams, assignments or projects. It is your responsibility to get your projects and assignments done on time. Be sure to work far enough in advance to avoid unexpected problems, such as illness, unreliable or overloaded computer systems, etc.

### Appealing Grades

After each exam, assignment, and/or project is graded, you have **7** calendar days to appeal your grade. All your appeals should be provided in writing. If after appealing you are still not satisfied, please visit the instructor. If you have questions about an exam, an assignment or a project grade, please visit the instructor directly.

## 9 Cheating

Each project or assignment must be the sole work of the student turning it in. Projects and assignments will be closely monitored, and students may be asked to explain suspicious similarities with any write-up or piece of code available. The following are guidelines on what cheating is and is not:

**What is cheating?**

1. Sharing code or other electronic files: either by copying, retyping, looking at, or supplying a copy of a file.
2. Sharing written assignments: either by re-writing, looking at, or supplying a copy of an assignment.

**What is NOT cheating?**

1. Clarifying ambiguities or vague points in class handouts.
2. Helping others use the computer systems, networks, compilers, debuggers, profilers, or other system facilities.
3. Helping others with high-level design issues.
4. Helping others debug their codes.

Consequently, be aware of what constitutes cheating (and what does not) when interacting with your colleague students. Same rules of cheating as above apply when collaborating with other students. In short, you cannot share written assignments, code, and/or other electronic files with other students. If you are unsure, ask the teaching staff.

Finally, be sure to store your work in protected directories. The penalty for cheating is severe, and might jeopardize your whole career as a student – cheating is not worth the trouble. By cheating in the course, you are cheating yourself; the worst outcome of cheating is missing an opportunity to learn. Besides, you will be removed from the course and assigned a failing grade. We also place a record of the incident in your permanent university profile.

## 10 Class Schedule

Table 2 demonstrates the tentative schedule of the class. The schedule indicates the project

and the assignment activities as well. Any changes will be announced and reflected on the course webpage. An updated schedule will be always maintained on the course webpage.

| Week | Session | Date | Topic | Teaching Method | Reading list | Projects | Prob. Solving Assignment |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 25 Aug | Logistics and Introduction | Lecture | Syllabus | | |
| | 2 | 27 Aug | Introduction to Distributed Systems | Lecture | C1, T1 | | |
| | 3 | 28 Aug | Advanced Object-Oriented Java Programming | Recitation | | | Start PS1 |
| 2 | 4 | 1 Sep | Physical and Architectural Models of Distributed Systems & Introduction to Networking | Lecture | C.2.1, C2.2, C2.3 (except 2.3.3), C3.1, & C3.2 | Start P1 | |
| | 5 | 3 Sep | Networking – Layering, Switching, Routing, and Congestion Control | Lecture | C3.3, C3.4 | | |
| | 6 | 4 Sep | Case study: Java Socket Programming & RMI | Recitation | P1 Write-up | | End PS1 |
| 3 | 7 | 8 Sep | Inter-Process Communication – Sockets, RPC, RMI, and Indirect Communication | Lecture | T4.2 – T4.6, C4.1 – C4.4, C5.1 – C5.4, & C6.1 – 6.4 | | Start PS2 |
| | 8 | 10 Sep | Naming – Flat, Structured and Attribute-Based Naming | Lecture | T5.1, T5.2, T5.3, T5.4 .1, & T5.4.2 | | |
| | 9 | 11 Sep | Case study: Naming & Storage Server Programming | Recitation | C5.5; Java API for UDP datagrams and Java API for TCP streams | Design Report P1 | |
| 4 | 10 | 15 Sep | Synchronization – Motivation and Physical Clocks | Lecture | T6.1 | | |
| | 11 | 17 Sep | Synchronization – Logical Clocks and Vector Clocks | Lecture | T6.2 | | End PS2 |
| | 12 | 18 Sep | Case study: Google Protocol Buffers and Publish-Subscribe | Recitation | C21.4 | | Start PS3 |
| 5 | 13 | 22 Sep | Synchronization – Mutual Exclusion and Election Algorithms | Lecture | T6.3 & T6.5 | | |
| | 14 | 24 Sep | Consistency & Replication – Motivation and Data-Centric Consistency Models | Lecture | T7.1 & T7.2 | End P1/Start P2 | |
| | 15 | 25 Sep | Design of Project 2 | Recitation | P2 Write-up | | |
| 6 | 16 | 29 Sep | Consistency & Replication – Client-Centric Consistency Models and Replica Management | Lecture | T7.3 & T7.4 | | |
| | | 1 Oct | **Midterm** | **Exam 1** | | | |
| | 17 | 2 Oct | TBA | Recitation | | | End PS3 |
| | | 5 Oct | Eid Al-Adha Break; NO CLASSES | Break | | | |
| 7 | 18 | 13 Oct | Consistency & Replication – Consistency Protocols | Lecture | T7.5 & T7.6 | Design Report P2 | |
| | 19 | 15 Oct | Fault Tolerance – Basic Concepts, Failure Models, Failure Masking by Redundancy & Process Resilience | Lecture | T8.1 & T8.2 | | |
| | | 16 Oct | Case study: Consistency in GFS and Google Chubby | Recitation | C21.5.1 & C21.5.2 | | |
| 8 | 20 | 20 Oct | Fault Tolerance – Reliable Request-Reply Communication | Lecture | C5.2 & T8.3 | | Start PS4 |
| | 21 | 22 Oct | Fault Tolerance – Reliable Group Communication, Distributed Commit and Recovery | Lecture | T8.4 – T8.7 | End P2/Start P3 | |
| | 22 | 23 Oct | Design of Project 3 | Recitation | P3 Write-up | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | 23 | 27 Oct | Programming Models – Traditional Models on Parallel Programming & Introduction to MPI | Lecture | Notes from Instructor | | |
| | 24 | 29 Oct | Programming Models – MPI | Lecture | Notes from Instructor | | |
| | 25 | 30 Oct | Applying MPI to Program a Classical Problem | Recitation | | | |
| 10 | 26 | 3 Nov | Programming Models – MapReduce | Lecture | Notes from Instructor | | |
| | 27 | 6 Nov | Programming Models – GraphLab and Pregel | Lecture | Notes from Instructor | End P3/Start P4 | |
| | 28 | 7 Nov | Design of P4 | Recitation | P4 Write-up | | End PS4 |
| 11 | 29 | 10 Nov | Distributed File Systems – Architectures, Data Striping, Processes and Communication | Lecture | T11.1 – T11.3 | | Start PS5 |
| | 30 | 12 Nov | Distributed File Systems – Naming, Synchronization, Consistency and Replication & Fault-Tolerance | Lecture | T11.4 – 11.7 | | |
| | 31 | 13 Nov | Applying MapReduce to Program a Classical Problem – Part I | Recitation | Notes from Instructor and the Yahoo Tutorial on Hadoop | | |
| 12 | 32 | 17 Nov | Big Table: A Distributed Structured Storage System | Recorded Video Lecture | The BigTable paper | | |
| | 33 | 19 Nov | Security – Threats, Policies and Mechanisms, Cryptograph, Authentication & Message Integrity and Confidentiality | Lecture | T9.1 & T9.2 | | |
| | 34 | 20 Nov | Applying MapReduce to Program a Classical Problem – Part II | Recitation | Notes from Instructor and the Yahoo Tutorial on Hadoop | | |
| 13 | 35 | 24 Nov | Security – Access Control Matrix, Protection Domains, Key Management & Authorization Management | Lecture | T9.3 – T9.5 | | |
| | 36 | 26 Nov | Virtualization – Motivation, Process and System Virtualization & Logical and Physical Partitioning | Lecture | Notes from the Instructor | | |
| | 37 | 27 Nov | TBA | Recitation | | | |
| 14 | 38 | 1 Dec | Virtualization – CPU Virtualization | Lecture | Notes from the Instructor | | |
| | 39 | 3 Dec | Virtualization – Memory and I/O Virtualization | Lecture | Notes from the Instructor | End P4 | |
| | 40 | 4 Dec | Overview | Recitation | | | End PS5 |
| **15** | | **7 Dec** | **Final** | **Exam 2** | | | |

**Table 2:** Tentative Time-Line of the Course.

Notations used in Table 1 are as given below:

- **PS** : Problem Solving Assignments
- **Cx(.y.z)** : Chapter x (Section y, Subsection z) from the Colouris *et al.* textbook
- **Tx(.y.z)** : Chapter x (Section y, Subsection z) from the Tannenbaum and Van Steen textbook
- **TBA** :To Be Announced