CS343 - Distributed Computing

Instructor:
Christine Bassem
cbassem@wellesley.edu

Lecture 1: Introduction

With a bit of Networks
According to Wikipedia:

- Distributed Computing is a field of computer science that studies distributed systems.

Distributed System

- A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal.

A better definition*:

Also,

- A distributed computing application consists of multiple programs running on multiple machines that coordinate together to perform a single task.

Specifically,

- The computers in a distributed system are independent and do not physically share memory or processors.
- They communicate with each other using messages, pieces of information transferred from one computer to another over a network.
Class Exercise - Candy Count

Get ready to stand up, and move around!

Let’s take a closer look

- How did you assign the work to each of you?
- What if we have more work units that you (machines)?
- How did everyone know what to do?
- How did you know that one of you is done?
- How did you aggregate the data at the end?
- What if one of you made a mistake counting?
- What is one of you fell asleep?

- Who made all of these decisions?
What will we learn in this course?

- The components needed in a distributed system to coordinate the distributed resources to perform a computation.

- These components include:
  - Network architectures
  - Message passing
  - Distributed algorithms
  - Fault-tolerant mechanisms
  - Distributed file systems

Our textbooks

We’ll denote this one as KS
And this as CDKB
How will we learn?

1. Before the lecture:
   - We’ll study existing systems
   - We’ll try to analyze how they should work

2. In the lecture:
   - We’ll discuss how they really work
     - Architecture
     - Algorithms

3. After the lecture:
   - You will build some components

   **By the end of the course**
   - We’ll coordinate our own distributed mini-computation

1) Before the lecture

- You will read about existing systems.
  - Or related algorithms.
  - Or current challenges.

- It depends on the lecture topic.

- The reading is required, because all class interactions will depend on it.

- Whenever time permits, we’ll do a live demo of the topic that you will be reading about in the previous lecture.
2) During the lecture

- You will forms groups of 4.
- In a typical setting, each of you will have one of these roles:
  - **Manager:** The person in charge of managing the group.
  - **Reflector:** The person in charge of analyzing the group dynamics.
  - **Recorder:** The person in charge of taking notes.
  - **Speaker:** The person in charge of speaking for the group.
- However, I will let you decide whether you prefer to have less/zero roles defined in your groups. You are the ones learning here 😊
- **Think about it, and let me know by next lecture.**
- **Group details will be finalized in the second week of classes.**

3) After the lecture

- There will be at least 4 programming assignments throughout the semester.
  - Work in pairs.
- And at least 2 written assignments.
  - Work individually.
- Starting from the Assignment 2, we’ll work on real virtual resources 😊
- All assignment management will be done through our GitHub course repository.
- Don’t forget to check out Assignment 0!
  - Due this Friday.
  - It’s simply to set up your accounts on GitHub.
Grading policy

- Assignments: 40%
- Final Project: 20%
- Exam 1 (in-class): 15%
- Exam 2 (take-home): 15%
- Class Participation: 10%

Final project

- After the second exam, you’ll start working on the final project.
- Projects are done in pairs.
- It is a compilation of all the components that you have built in your assignments, to build a complete system.
- You also have the option to pick your own research / exploration project idea, if you’d prefer.
Intermediate nodes such as switches or routers allow the hosts to share the infrastructure.
Sharing bandwidth

FDMA

Example: 4 users

TDMA

Circuit switching

- It’s the method used by the telephone network
- A call has three phases:
  1. Establish circuit from end-to-end (“dialing”),
  2. Communicate,
  3. Close circuit (“tear down”).
- If circuit not available: “busy signal”
But, Internet traffic is bursty!

Daily traffic at an MIT-CSAIL router

Max In: 12.2 Mb/s  Avg. In:  2.5 Mb/s
Max Out: 12.8 Mb/s  Avg. Out:  3.4 Mb/s
Packet switching

- Messages are broken into packets each of which travel from the source to destination through a maze of routers and links.

- Most routers are store-and-forward, meaning the switch must receive the entire packet before it can transmit it outbound link.
Packet Switching

- Messages are broken into packets each of which travel from the source to destination through a maze of routers and links.
- Data is sent in these packets (header contains control info, e.g., source and destination addresses)
- Per-packet routing
- At each node the entire packet is received, stored, and then forwarded (store-and-forward networks)

Efficient, but queues introduce ...

- Variable Delay
  - Delay = sum of
    - Processing delay: Time required to examine the packet’s header and determine where it goes.
    - Queuing delay: Time packet cools its heels waiting to be served.
    - Transmission delay: Store-and-forward delay.
    - Propagation delay: Time to push a bit through a link.

- Losses
  - When packets arrive to a full queue/buffer they are dropped
Packet switching may also suffer from reordering

- Each packet is individually routed
- No time guarantee for delivery
- No guarantee of delivery in sequence
- No guarantee of delivery at all!
  - Things get lost
  - Acknowledgements
  - Retransmission
    - How to determine when to retransmit? Timeout?
- If packet is re-transmitted too soon → duplicate

Packets in a flow may not follow the same path → packets may be reordered
Best Effort

- No Guarantees!
- Variable Delay (jitter)
- Variable rate
- Packet loss
- Duplicates
- Reordering

Differences Between Circuit & Packet Switching

Circuit-switching
- Guaranteed capacity
- Capacity is wasted if data is bursty
- Before sending data establishes a path
- All data in a single flow follow one path
- No reordering; constant delay; no pkt drops

Packet-Switching
- No guarantees (best effort)
- More efficient
- Send data immediately
- Different packets might follow different paths
- Packets may be reordered, delayed, or dropped