Group Project 1: A Bite of Distributed Communication

CECS 327 – Intro to Networks and Distributed Computing

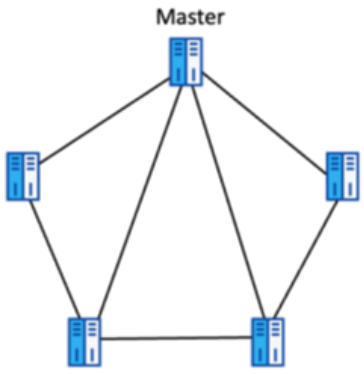
You should submit the required deliverable materials on Canvas by **11:55pm, March 3rd (Sunday), 2024. Only one submission for each group. Please put all your files into one folder for submission.**

1. Description

In this project, you will embark on the exciting journey of building a distributed system using Java/Python, Docker, and the Linux terminal. The project is divided into three main steps, each focusing on different aspects of system architecture, communication protocols, and network monitoring.

Step 1. Build a small distributed system.

You will start by setting up a distributed system consisting of a master server and four nodes using Python code and Docker containers. The master server and nodes are configured to communicate over ports, creating a network named "Proj1-distributed-network" within Docker. Through this setup, you will gain hands-on experience in deploying and managing distributed systems in a controlled environment. As shown in the figure, the servers should be managed into one cluster, and you can manually define one serves as the master server.



Step 2: Implement Protocols

Once the distributed system is up and running, students will delve into implementing protocols for communication between the master server and nodes.

(Choose two protocols in your design, you can also complete all three protocols if you want 😊)

1) Broadcast Protocol Design: Collaboratively design a custom broadcast protocol with a focus on minimizing unnecessary traffic and optimizing message delivery.
2) Unicast Protocol Design: Develop a unicast protocol for one-to-one communication.
3) Multicast Protocol: Develop a multicast protocol that for group communications.

Implement the designed protocols in a simulation environment to evaluate their performance under various network conditions. With the design of protocols, run different test cases to allow server and clients existing messages or deliver data (test all broadcast, multicast, and/or unicast cases). You can define the test cases by yourself.

Note: this project did not ask you to design all protocols from scratch, you should look at existing design and implement them into your cluster. For example. A useful example for multicast: https://pymotw.com/2/socket/multicast.html But remember identifying the source protocol design when you cite it. Otherwise, it will be taken as cheating.
Be careful: you need to do more research work to figure out how to design or implement the protocols, which is the most important point of this project.

Step 3. Monitor the communication.

You should track the network activities by monitoring all communications. Please add a network analysis tool/function inside your design, which can display the detailed information of all communications. You can follow the below tables of such info. (Can be in a CSV file or txt file)

Type	Time(s)	Source_Ip	Destination_Ip	Souce_Port	Destination_Port	Protocol	Length (bytes)	Flags (hex)
Broadcast	0.0000000	192.168.1.202	172.224.42.53	42992	60002	TCP	1314	0x010
Anycast	0.1111111	172.224.42.53	192.168.1.202	3478	46692	UDP	1214	0x011

Prerequisites:

- Basic knowledge of programming language.
- Familiarity with Docker containers and Linux terminal commands.
- Understanding of networking concepts such as IP addresses, ports, and communication protocols is beneficial but not mandatory.

Recommended Resources:

- Java/Python documentation and tutorials for socket programming.
- Docker documentation for containerization and networking.
- Online tutorials and forums for troubleshooting and learning about distributed systems and communication protocols.

3. The Required Deliverable Materials

1. A README file, which describes how we can compile and run your code.
2. Your source code, you may use a Makefile and be submitted in the required format.
3. Your short report, which discusses the design of your program.
4. A recorded video shows the output and runtime

4. Submission Requirements

You need to strictly follow the instructions listed below:

1) This is a **group project**, please submit a .zip/.rar file that contains all files, only one submission from one group.
2) Make a **video** to record your code execution and outputs. The video should present your name or time as identification (You are suggested to upload the video to YouTube and put the link into your report).
3) The submission should include your **source code** and **project report**. Do not submit your binary code. Project report should contain your groupmates name and ID.
4) Your code must **be able to compile**; otherwise, you will receive a grade of zero.
5) Your code should not produce anything else other than the required information in the output file.
6) If you code is **partially completed**, please explain the details in the report what has been completed and the status of the missing parts, we will grade it based on the entire performance.
7) Provide **sufficient comments** in your code to help the TA understand your code. This is important for you to get at least partial credit in case your submitted code does not work properly.

Grading criteria:

Details	Points
Have a README file shows how to compile and test your submission	5 pts
Submitted code has proper comments to show the design	15 pts
Screen a ***video*** to record code execution and outputs	20 pts
Have a **report** (pdf or word) file explains the details of your entire design	20 pts
Report contains clearly individual contributions of your group mates	5 pts
Code can be compiled and shows correct outputs	35 pts

5. Policies

1) Late submissions will be graded based on our policy discussed in the course syllabus.
2) Code-level discussion is **prohibited**. We will use anti-plagiarism tools to detect violations of this policy.
3) AI-tools such as ChatGPT, GPT4, etc. are prohibited in both code and contents.