CS558: Computer Systems Lab

(January-May 2024)

Assignment - 6

Submission Deadline: 11:55 pm, Wednesday, 24th April, 2024

Instructions:

- Read the questions carefully.
- Each group needs to implement all questions.
- The programs can be written in C/C++.
- Assignments submitted before the deadline will only be considered for evaluation. No extension in submission is allowed. Delay in submission will lead to penalty in marks.
- If something is missing/incorrect in a problem description, clearly mention the assumption with your answer.
- Submit the set of source code files of the application as a zipped file. The ZIP file's name should be the same as your group number - for example, "Group_4.zip", "Group_4.rar", or "Group_4.tar.gz".
- Only one member from a group needs to submit this form by uploading the file: (https://forms.gle/EmCckPZdMAgoeyj5A)
- The assignment will be evaluated offline/through viva voce during your lab session where you will need to explain your answers before the evaluator.
- A plagiarism detection tool will be used and any detection of unfair means will be penalised by awarding NEGATIVE marks (equal to the maximum marks for the assignment).
- NO sharing of code between students, submission of downloaded code (from the Internet, Campus LAN, or anywhere else) is allowed. The first instance of code copying will result in ZERO marks for the assignment. The second instance of code copying will result in a 'F' grade. Students may also be reported to the Students Disciplinary Committee, which can impose additional penalties.

Question 1:

Implement a simplified version of the OSPF (Open Shortest Path First) routing protocol in C/C++. The goal of this assignment is to build a routing table for a network of routers and simulate the routing of packets between them.

Specifications:

1. Implement a Router class in C/C++ with the following attributes:
    - router_id: A unique identifier for the router (an integer).
    - neighbours: A list of neighbouring routers, represented as pointers to other Router objects.
    - routing_table: A data structure to store routing information.
    - add_neighbor(Router* neighbour): A method to add a neighbouring router to the current router's list of neighbours.
    - update_routing_table(): A method to calculate the routing table for the current router using the OSPF algorithm (For demonstration purposes, you can assume static routing here)
    - print_routing_table(): A method to print the routing table for the current router.
2. Implement the OSPF algorithm within the update_routing_table() method to compute the shortest path to each router in the network using Dijkstra's algorithm or a similar approach.
3. Create a network of routers and connect them by adding neighbours to each router.
4. Simulate the routing of packets from one router to another by calling the update_routing_table() method and then using the routing table to determine the path.
5. Print the routing table for each router to demonstrate the routing information.

Question 2:

Implement a simplified MAC layer simulation for a wireless network. Create a program that simulates the behaviour of nodes in a network communicating using the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol.

Requirements:

a. Create a simulation environment with a configurable number of nodes (N) and shared communication channel.
b. Implement a backoff mechanism where nodes choose random backoff intervals before attempting to transmit data. The backoff interval should be adjustable.
c. Simulate the CSMA/CA protocol: Nodes should listen to the channel and only transmit when it's clear (no other nodes transmitting). If the channel is busy, they should back off and retry after the back-off period.
d. Implement a collision detection mechanism: If two or more nodes attempt to transmit simultaneously, a collision occurs, and they should back off for a random period before reattempting.
e. Record statistics: Keep track of successful transmissions, collisions, and the number of times nodes had to back off before successfully transmitting.

Question 3:

Implement a basic HTTP, Web cache that stores recently accessed web pages to improve retrieval speed for subsequent requests. The cache should follow a least-recently-used (LRU) eviction policy. The cache will store web pages retrieved using HTTP GET requests.

Requirements:

- Implement a cache data structure that can store a fixed number of web pages (e.g., 5 pages) in memory.
- Implement a function to retrieve a web page from the cache. If the requested page is present in the cache, return the cached content. If not, fetch the page using an HTTP GET request, store it in the cache (evicting the least recently used page if necessary), and return the content.
- Use sockets to perform the HTTP GET request and receive the web page content.
- Use a linked list to keep track of the order of page accesses. When a page is accessed, move it to the front of the list to represent it as the most recently used page.
- Implement a function to display the contents of the cache, showing the order of page accesses from most to least recently used.

Hints:

- Use socket programming to establish a connection with the web server, send an HTTP GET request, and receive the web page content.
- Use a linked list to represent the cache, where each node contains the URL and content of a web page.
- Example web page url that you can provide for testing: http://quietsilverfreshmelody.neverssl.com/online/

Example Output:

Cache size: 3

Enter URL (or 'exit' to quit): http://www.example.com/page1

Page content:

... (content of page1) ...

Cache Contents (Most to Least Recently Used):

1. http://www.example.com/page1

Enter URL (or 'exit' to quit): http://www.example.com/page2

Page content:

... (content of page2) ...

Cache Contents (Most to Least Recently Used):

1. *http://www.example.com/page2*
2. *http://www.example.com/page1*

Enter URL (or 'exit' to quit): http://www.example.com/page1

Page content:

... (content of page1) ...

Cache Contents (Most to Least Recently Used):

1. *http://www.example.com/page1*
2. *http://www.example.com/page2*

Enter URL (or 'exit' to quit): exit

Provide a well-commented C/C++ code file containing the implementation of the HTTP, Web cache as per the given requirements.