UNIVERSITY OF TARTU
Institute of Computer Science

Distributed Systems (Spring 2021)

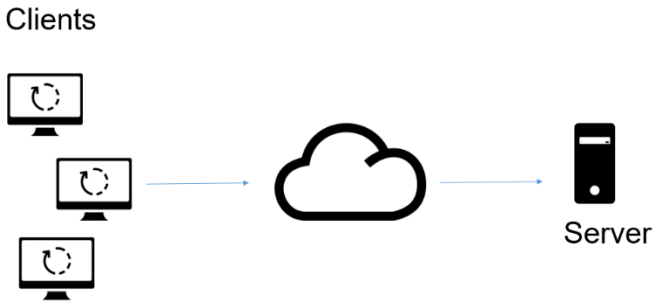
Task 1: system architectures and performance modelling

Practical information:
Due date: Monday, March 15, midnight (FIRM DEADLINE)
– This task can be performed in a team of max 3 persons ONLY.
– Total amount of pts that that can be collected = 100 + Bonus pts = 115 pts. **Bonus pts are optional**

Submit your solution to farooq.ayoub.dar@ut.ee, mohan.liyanage@ut.ee, (CC) huber.flores@ut.ee

Instructions: Please respond to the following questions. **Be clear and concise in your answers.** Provide enough explanation to support your arguments. Ambiguous answers to fill up space are considered wrong and no points are granted. For questions that involve implementation, be sure to include everything needed to execute your program (Re-submissions are not allowed).

1. One key characteristic of a distributed system is that its components can be added and removed transparently with ease. Assuming that an existing centralized Client-Server system (in the Figure below) has new policies, which indicate that that user authentication is required to access the server for the upcoming six months, and after six months, the authentication will be removed again. Please enumerate at least two solutions about how to handle this issue. Notice that the service demand of the server should be preserved. Use illustrations to support your ideas (7 points)



2. An end-to-end Client-Server system for Web applications has 100 clients and the client's think time is equal to 3 seconds. The utilization in one disk of system is estimated to be 60%. Another available measurement is the service time at the disk, which is equal to 30 milliseconds. Each user interaction requires, on average, four I/Os on this disk.
    a. What is the average response time of the interactive system? (6 points)
    b. What are the implications of this result? (4 points)

3. A Web server is deployed to handle a shopping Web page. Due to the increasing number of sold items, the Web page has become very unresponsive. To fix the problem, some performance measurements are collected from the server hosting the page (using logs). Measurements taken during one hour from the Web server shows that the CPU utilization and the utilization of the two disks are: $U_{CPU} = 0.25$, $U_{disk1} = 0.35$, and $U_{disk2} = 0.30$. The Web server log shows that 141,600

requests were processed during the measurement interval. Please elaborate and use illustrations to respond the following questions/statements.

a. Draw the QN model that is most appropriate to model this situation (qualitative aspects). (2pts)
b. What are the service demands at the CPU and both disks? (3pts)
c. What is the maximum throughput? (2pts)
d. What was the response time of the Web server during the measurement interval? (2pts)
e. What would be your recommendations in this situation? (1pt)

4. Explain the key differences between stateful and stateless servers. Use illustrations to support your ideas (8 points)

5. An e-commerce Web system recoded the periods of time in which the service was down during the last three days and obtained the results shown in the Table below. What was the availability of the site during the three days? (10 points)

Day	Start of Down time	Duration of Down time (min)
1	13:25 PM	10
1	9:00 AM	3
2	17:30 PM	5
3	6:00 AM	5
3	15:00 PM	7
3	21:30 PM	1

Table: Site Down Periods

6. Consider a single Web server with three identical storage disks. The service demand in the server on these disks are, 75 milliseconds (disk 1), 120 milliseconds (disk 3) and 140 milliseconds (disk 3), respectively. In parallel to this, the server is constantly receiving workload from clients requesting services. The workload of the Web server is divided into four types of requests: simple (R1), medium (R2), complex (R3), and internal jobs (R4). The Table below shows the arrival rates and service demands for each class.

	Class (request type)			
	R1	R2	R3	R4
Arrival rate (requests per second – rps)	0.15	0.25	0.40	0.12
Service demand (seconds)				
CPU	0.20	0.20	0.45	0.7
Disk 1	0.25	0.25	0.35	0.3
Disk 2	0.15	0.30	0.25	0.2
Disk 3	0.05	0.15	0.40	0.1

Analyze how these service demands would change under the following scenarios:

- Disk 1 is replaced by a disk that is 50% slower. (2pts)
- Disk 1, Disk 2 and Disk 3 are replaced by disks that are 50% faster. (2pts)
- What is the effect on response time of R1 requests if their arrival rate increases by 70%? (2pts)
- What is the effect on response time of R3 requests if their arrival rate is increased by 45%? (2pts)
- What is the effect on the response time of R1 requests if R2 requests are run on a different computer? (2pts)

7. Implement a Distributed Hash Table (DHT): Follow the specifications below (45 pts). You can use any language of your preference for performing this task. Implement the logic described below. Do not use solutions that just simulate the required behavior, e.g., directory solution.

Program specifications:
A DHT is typically deployed over a decentralized system architecture (peer-to-peer). By sorting the collection of nodes into a ring, the main purpose of a DHT is to facilitate the location of nodes (containing data) giving a key value (lookup).

Your DHT program is required to provide a command line interface, and has to consider the following input file below for bootstrapping.

Input-file.txt: (# depicts a comment)
#---here starts the file--

#key-space
1, 100

#nodes
5, 56, 22, 17, 89, 71, 92, 110

#shortcuts
5:56, 5:71, 22:89

#---here end the file--

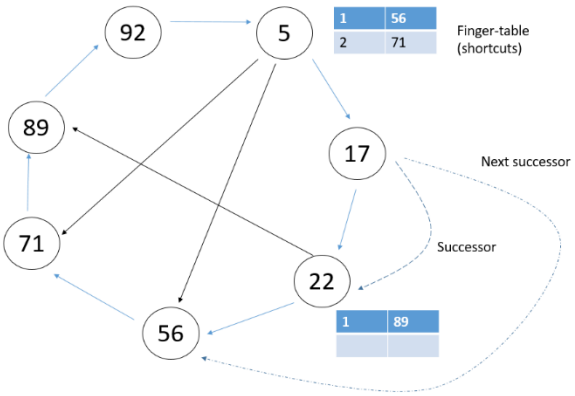
Where,

- #**key-space** defines the range of values that the ring stores.
- **#nodes** describe the list of actual nodes in the ring storing key data. Nodes are assigned with values within the #key-space. If nodes with values outside that range are introduced, then the nodes are discarded and a warning message should be triggered, e.g., 110 is not a valid node. At least one node should be defined in the file. If there is just a single node, this node has a reference to itself.
- **#shortcuts** describe direct connections between nodes that can be used to reduce lookup time.

The DHT program is required to have the following commands (Wrong syntax/commands or out of range key values should be reported as error warnings)

1) Your program should start by loading the initial ring based on the input file **(10pts)**

$ Your_DTH_program *Inputfile.txt*



(You do not have to draw the ring, but this is how the ring looks like. Notice that each node has two references by default – **successor and next successor.** Remember also that each successor node stores all the key values before it, for instance, node 17 stores key values in the range of 6 to 17). Nodes can just send requests to successor nodes (neighbors) or nodes in their finger table (if any).

After the initial ring is loaded, your DHT program has to support the following commands through command line

2) **List:** List the nodes in the ring (sorted from lower to higher). It also shows the references that each node is storing. (: Shortcuts, S: successor, NS: next successor) **(10pts)**

$ List
5:56,71, S-17, NS-22
17:, S-22, NS-56
22:89, S-56, NS-71
56:, S-71, NS-89
71:, S-89, NS-92
89:, S-92, NS-5
92:, S-5, NS-17

3) **Lookup:** It is used to lookup for a node containing a particular key. The lookup also defines the node that receives the lookup request. If just a key value is passed, then the lookup starts from the node with the lowest value. A node lookup for key data to itself has a weight of zero. A node lookup for key data to its finger table (shortcuts) or another node has a weight of 1. Lookups are accumulated until the key value is found. **(10pts)**

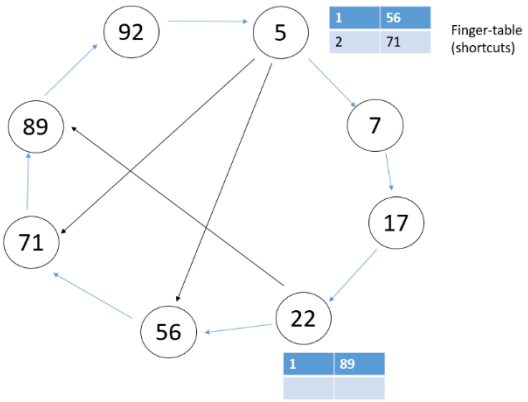
(node 17 receives a lookup request for key 69)
$ Lookup 69:17
Result: Data stored in node 71 – 3 requests sent

(by default node 5 receives a lookup request for key 87)

$ Lookup 87
Result: Data stored in node 89 – 2 requests sent

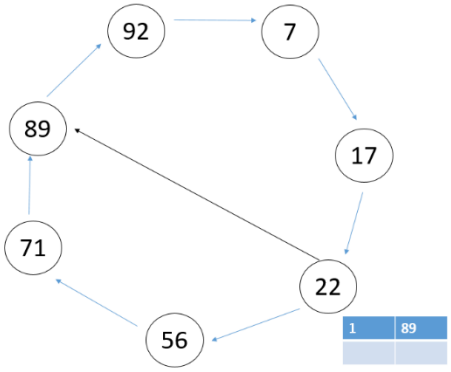
4) **Join:** this commands allows a new node to join the ring. Notice that successor and next successor references should be updated. **(5pts)**

$ Join 7



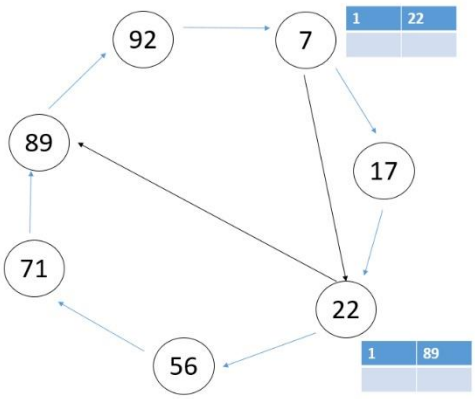
5) **Leave:** this commands allows a node to leave the ring. We will handle the simple resilience case in which a single node leaves at the time. Leave cannot be applied when there is just a single node remaining. **(5pts)**

$ Leave 5



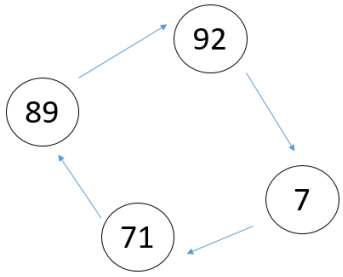
6) **Shortcut:** this command adds a shortcut from one node to another. **(5pts)**

$ Shortcut 7:22



(10 pts) BONUS POINTS: Explore how to handle resilience when more than one node leaves at the same time. Implement your strategy using the following command and explain it in detail.

$ Remove 17, 22, 56



Deliverables for your DHT program:

1- A short 1 min video showing your program in execution (showing all the commands)
2- Source code and everything needed for executing your program (binaries and input file). We will evaluate your work with multiple file configurations. Thus, do not hard code your solution just to work with the input file described here.

8. **(5 pts) BONUS POINTS:** Implement the dispatcher/worker model for handling client-server requests using threads. You can use any language of your preference for performing this task. Use a single class (of your choice) as a service. Generate a diagram that shows how server performance degrades as the load of requests increases. Verify the minimum workload that can be handled with a response time below 500ms. Write down the specifications of your deployment environment (CPU, memory, etc.), and reflect on how the response time of the system can be preserved below the specified threshold.