Student’s name: Nguyễn Đức Dũng

Class: ICT.02-K61

**Class Exercises**

**Module: Distributed Systems**

**Chapter 2: Architectures**

**Theoretical Exercises:**

Question 1: If a client and a server are placed far apart, we may see network latency dominating overall performance. How can we tackle this problem?

It really depends on how the client is organized. It may be possible to divide the client-side code into smaller parts that can run separately. In that case, when one part is waiting for the server to respond, we can schedule another part. Alternatively, we may be able to rearrange the client so that it can do other work after having sent a request to the server. This last solution effec- tively replaces the synchronous client-server communication with asyn- chronous one-way communication.

Question 2: What is a three-tiered client-server architecture?

A three-tiered client-server architecture consists of three logical layers, where each layer is, in principle, implemented at a separate machine. The highest layer consists of a client user interface, the middle layer contains the actual application, and the lowest layer implements the data that are being used.

Question 3: What is the difference between a vertical distribution and a horizontal distribution?

Vertical distribution refers to the distribution of the different layers in a multitiered architectures across multiple machines. In principle, each layer is implemented on a different machine. Horizontal distribution deals with the distribution of a single layer across multiple machines, such as distributing a single database.

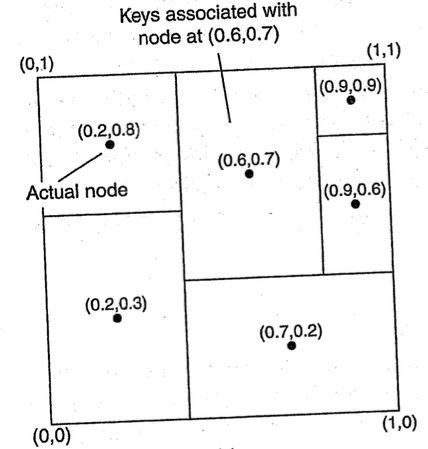
Question 4: In a structured overlay network, messages are routed according to the topology of the overlay. What is an important disadvantage of this approach?

The problem is that we are dealing only with *logical* paths. It may very well be the case that two nodes *A* and *B* which are neighbors in the overlay network are physically placed far apart. As a consequence, the logically short path between *A* and *B* may require routing a message along a very long path in the underlying physical network.

Question 5: Consider a chain of processes P1 , P2 , ..., P n implementing a multitiered  client-server architecture. Process Pi is client of process Pi+1, and Pi will  return a reply to Pi−1 only after receiving a reply from Pi+1 . What are the main problems with this organization when taking a look at the request-reply performance at process P1?

Performance can be expected to be bad for large *n*. The problem is that each communication between two successive layers is, in principle, between two different machines. Consequently, the performance between *P*1 and *P*2 may also be determined by *n* − 2 request-reply interactions between the other layers. Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the perfor- mance at the highest level.

Question 6: Considering that a node in CAN knows the coordinates of its immediate neighbors, a reasonable routing policy would be to forward a message to the closest node toward the destination. How good is this policy? (see the figure in the attached file)



In our example from the previous question, it can already be seen that it need not lead to the best route. If node (0.2,0.3) follows this policy for the message destined for node (0.9,0.6), it would send it off to node (0.7,0.2).

Question 7: What are the benefits of Microservices architecture compared to monolithic architecture?

Microservice capabilities are expressed formally with business-oriented APIs. They encapsulate a core business capability, and as such are valuable assets to the business. The implementation of the service, which may involve integrations with systems of record, is completely hidden as the interface is defined purely in business terms. The positioning of services as valuable assets to the business implicitly promotes them as adaptable for use in multiple contexts. The same service can be reused in more than one business process or over different business channels or digital touchpoints, depending on need. Dependencies between services and their consumer are minimized by applying the principle of loose coupling. By standardizing on contracts expressed through business-oriented APIs, consumers are not impacted by changes in the implementation of the service. This allows service owners to change the implementation and modify the systems of record or service compositions which may lie behind the interface and replace them without any downstream impact.

Question 8: Design yourself an e-commerce system using Microservices architecture.

A picture containing table

Description automatically generated