Assignment 2

Q6: System Architectures

a. Distributed applications targeting centralized architectures (client/server model) are often implemented as multi-tiered systems (e.g., user interface, application logic, and database). What is the motivation for such a design?

b. How can the scalability of client/server architectures be improved?

c. What is the conceptual difference between a multi-tiered client/server architecture and the peer-to-peer approach with respect to the distribution of the service?

d. Compare client/server architectures and the peer-to-peer approach. What properties should a (distributed) application have to be well-suited for either approach?

e. Compare structured and unstructured peer-to-peer systems. What are the respective strengths and weaknesses?

Q7: Software Architectures

a. Enumerate some of the services typically provided by the middleware.

b. Outline the general HW/SW structure of a distributed operating system.

c. Name the main difference(s) between a distributed operating system (DOS) and a network operating system (NOS).

Q8: Communication Models

a. Depict the basic communication model. What is required for effective communication to take place?

b. Why do many models introduce layers (protocol stack) in the sender and the receiver?

c. Enumerate the basic network topologies—both static and dynamic. List interesting properties such as max. path length and compute them for one non-trivial topology.

Q9: Unreliable Multi-Hop Communication

a. What types of errors can occur when sending data through a packet switched network? Can we detect or correct them? Explain your answers.

b. How can TCP establish reliable communication on top of the unreliable IP protocol?

c. Sketch the principal ideas of link state and/or of distance vector routing.
Q10: Sockets

a. Describe the client- and server-side actions required to establish, use, and dispose a communication channel via sockets.

b. Each socket links at most one server with one client. How can a server still handle requests from multiple clients simultaneously?

Q11: Programming Assignment: Fun with Sockets
Implement the game “Battleships”\(^1\) as a distributed application, using sockets for communication. Some ideas and design suggestions:

- It is possible to either realize the game using a dedicated server process to which both players have to connect, or to have both client programs communicate directly with each other. Choose whatever approach you favor.
- The game is intended as a pure multiplayer game, so there is no need for AI-based opponents.
- Focus on the communication mechanisms. You might need to develop some sort of protocol that allows one player’s client to notify the server/other client of the actions the player performed.
- If you do not want to implement a GUI, you can e.g. read the initial configuration (ship placement) from a text file.
- There is no need for authentication mechanism, cheating protection, 3D visualization ...
- Choose whatever programming language you like, but if you want to send me your sources, you should use something like C(++) or Java.

This assignment is intended to help you to become acquainted with socket programming. Therefore, you should not use any existing RPC libraries. The assignment is completely voluntary, and there is no deadline. You can send your results (i.e., the sources) to pk@ibds.uka.de. If you have a working version until end of term, you can demonstrate it in the tutorial (but please contact me beforehand).

\(^1\)Schiffe versenken