Modis Exam Fall 2019

Contents

[Introduction 2](#_Toc26788162)

[Protocol 2](#_Toc26788163)

[TCP 2](#_Toc26788164)

[Ring Network Topology 2](#_Toc26788165)

[Client-Server 2](#_Toc26788166)

[Correctness 1 3](#_Toc26788167)

[Correctness 2 3](#_Toc26788168)

15% Description of your protocol.

25% Analysis arguing for correctness of your protocol in the absence of failures.

25% Analysis arguing for correctness of your protocol in the presence of failures.

20% Other questions

15% Implementation of the design

# Introduction

The distributed auction-system is built with a client-server structure. Any number of network-nodes can add themselves to the network if they know of at least one existing node’s socket. The network periodically monitors itself and, in the case of an unresponsive node, is quickly able to reconstruct itself. Additional nodes, although not needed, function as additional containers of the current auction’s state.

The client does not need to be aware of the current structure of the network. It only needs a single node’s socket to send and receive auction-related messages. The results of client requests do not vary depending on the number of nodes in the network.

# Protocol

## TCP

The communication between nodes, other nodes and clients follows the Transmission Control Protocol. This is to ensure that no unsuccessful transfer of a message is assumed to be successful. If a handshake is not responded to, the nodes act accordingly. They assume the unresponsive node to be “*dead*” and reconstruct the network without that node.

In the implementation, handshakes are performed as “*pings*” between network-nodes. When a node wishes to send a message to another node, a ping and its response represent the handshake-mechanism. Within the ping-message an ID is included in the header. This ID functions as a representation of the message-to-come and lets the receiver know what to expect and what they are accepting when responding with an acknowledgement.

As additional stability insurance, the nodes ping their neighbor-nodes twice every second to assure their well-being. Holes in the network-structure are then closed automatically. The network-topology will be explained thoroughly in an upcoming section.

To support transferring of multiple types of messages, the messages are all an extension of the same *TCPMessage*-class. This way, any message can be treated in the same way until they are determined to be an instance of any subclass of *TCPMessage*and then treated accordingly. These message objects are serialized, sent as byte-arrays and then deserialized on the receiving end.

## Ring Network Topology

The node-network is structured according to the ring network typology. This is characterized by having each node know of exactly two other nodes. In the implementation, each node has two *neighbors*. They have a *next-neighbor* right next to them and a *nextnext-neighbor* after their *next-neighbor*. The ring is unidirectional, since the nodes point to two neighbors in one direction. Although a unidirectional network on paper is less flexible, since this network is still aware of two neighbors and can therefore reconstruct itself in the case unresponsive nodes.

The ring-restoration process starts when a node finds it’s *next-neighbor* to be unresponsive and a *NodeLostMessage* is sent around the ring. During the traversing, sockets of nodes next to the missing link, are saved in the message header. That information is used to close the gap when the initiator receives the message after it has come around the ring, completing the restoration.

Each node in the network carry equal workloads. They all contain all the necessary data for the current auction. This makes any node dispensable unless of course it is the last one in the network.

## Client-Server

# Correctness 1

Df

# Correctness 2

df