Modis Exam Fall 2019

Contents

[1. Introduction 2](#_Toc26808231)

[2. Protocol 2](#_Toc26808232)

[TCP 2](#_Toc26808233)

[Ring Network Topology 2](#_Toc26808234)

[Client-Server 3](#_Toc26808235)

[3. Correctness in the absence of failures 3](#_Toc26808236)

[4. Correctness in the presence of failures 4](#_Toc26808237)

[5. Other Questions 4](#_Toc26808238)

[P2P (structured or unstructured), replication, or something else 4](#_Toc26808239)

[Analysis of the security aspects of your distributed auction system 4](#_Toc26808240)

# 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 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. The auction starts when the first node is created. Any new nodes will have the auction information transferred to them.

# 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. Any node being dispensable is of course a plus, however there are a few disadvantages. Firstly, the insertion of new auction data, requires that each node is sent the same query. In the case of a large circuit of say, 20 nodes, the placement of a new bid takes at least twenty times the amount of traffic as it would if only one node had to be given the data. And 20 times the amount of memory is also used.

New bids are registered when they enter the system. There is no timestamp on the bids and no message queue ensuring that they are treated in order. Since two bids could potentially traverse the ring at the same time, the outcome seems a bit unsure at first glance. However, the larger bid will always, in the end, override the smaller one, since it will eventually have reached all nodes that the small bid has reached. Even if a smaller bid entered the ring through a node ahead of the larger bid, the small bid will eventually reach a node that has been traversed by the large bit and then be marked with the error: “*too small*”.

The moment an auction ends depends on the server-nodes’ perception of global time. It simply compares the local clock with the given end time of the auction. There is no consideration of the time a bid message is sent or the amount of time it takes for it to reach the receiver. The auction-system is built upon the assumption that all server nodes already have synchronized clocks. That is a fatal flaw. There is never a guarantee that two different computers run their clocks synchronously. With a network spanning different time zones, the auction would not function at all.

## Client-Server

An auction is run entirely by the server-network. Clients send requests and get responses accordingly. A peer-to-peer system could also have worked. The main advantage of P2P is that the workload is distributed among users of the service. In the case of this auction system however, the total workload is never larger than just a few variables. Besides that, an auction should be allowed to take place without any active attendees.

Letting the client keep track of the current highest bid might also be a cause of concern when it comes to safety. There is also no requirement for users the be able to start an auction. Therefore, it was found to be ideal for some type of central network to be responsible for the auction.

The client and server utilize request-reply communication. The client expects each invoke to result in a reply from the node it contacted. When placing a bid for instance, a reply is returned a status. That status is either marked with *success*, *fail*, or *exception* including a supplementary description of the outcome.

# Correctness in the absence of failures

Upon initialization, a node with no neighbors starts the auction, marking the time that it ends and setting the highest bid to 0. Any new node introduced to the initial node is put into the ring and has the current auction status transferred to it. That way, even nodes added after the beginning of an auction, can be used as an entry-point for *bids* and *result*-invokes. Since the relations of nodes in the network are the same no matter their position in the ring, new nodes can be introduced through any of them.

When receiving a bid, the message is forwarded through all nodes until it once again reaches the original entry-node. The message is then returned to the client along with the latest *status* given by the node before the last one. That status depends on the bid and the current auction’s state. The status informs the client whether the bid was successful, too low or too late. Since all nodes at any point hold the same auction-status, the resulting response given to the client will be the same as if the entry-node was the only one in the network. Result-invokes from the client is sent to a single node and that node handles the request by itself. Since all nodes in the network at this point are identical, the response is correct.

The auction is expired if the set end-time (demonstrated by a millisecond time of the computer) is less that the current time. The expiration of an auction is not observed until a client sends a *bid*- or *result*-invoke. Since that doesn’t change the logic of the current time having past another time, that is a functional solution.If all nodes run on synchronized time, the auction will have run out at the same time on all of them. However, as discussed in the *Ring Topology Network*-section, if the clocks are not perfectly in sync, which they most likely are not, the timing will fail.

# Correctness in the presence of failures

Each node, once every 500ms, pings it’s *next-neighbor*. If that neighbor does not respond, it is assumed to be dead. The node then uses the neighbor after the dead one to reconstruct the ring. Since each node contains the same auction-data, one or more failures will not be a problem.

If a handshake fails however, the node simply cancels the transfer and does not reconstruct the ring. Since the network reconstruct itself often and quickly, this will most likely not happen. But if a node happens to die and a client sends it a request before the network is reconstructed, the message will never be received. That can easily be considered a flaw in the system. If for instance, the badly timed message is a bid, only the nodes before the dead one will be notified of the bid and the user will never receive a response because the ring couldn’t be traversed fully. The network also won’t be able to reconstruct itself if two nodes die at times close enough to each other for the network not to recover between the failures.

This could be fixed by using a message queue. Instead of cancelling a not-received message, it would be better if nodes saved the failed message, attempted to reconstruct the network and then again attempted to send the message. This insurance of delivery is a common part of the TCP-protocol that has not been implemented in this system.

# Other Questions

## P2P (structured or unstructured), replication, or something else

The auction only contains a few objects that the network allows clients to access. Those are the end-time of the auction and the highest bid. The network nodes all contain replicas of that data and allow multiple entry points for multiple clients as well as extra stability in case one node breaks down. Thus, the system uses replication.

## Analysis of the security aspects of your distributed auction system

Messages between nodes and the client are serialized objects. These byte-arrays are not as easy to tamper with as pure strings, but they could still potentially be manipulated by someone with bad intentions. Especially if that someone has the source-code.

The system uses TCP for transferring messages. The source of a message is included in the ping, which could be involved in some type of filtering of senders. Using certificates for instance. However, this is not included in the implementation. This means that anyone, provided they have the source-code, could impersonate a trusted node and add themselves to the network or send node-related messages. For a message to be accepted however, it must be preceded by a PingMessage with a handshake-key matching that in the message. TCP can be utilized for extra security measures but in this implementation it is, to put it bluntly, only used to ensure that the receiver is responsive.

An advantage of the ring-topology and replication of auction-data in this network is that no phony node can allow a too small bid to become the highest bidder. Every node receives the bid-invoke and any if of those nodes find the bid to be too low, their stored highest bid is not overwritten. The last node in the recursion dictates the response that the user received but no node can overwrite another node’s decisions. Since the network can survive one or more nodes dying, denial-of-service attacks that only take out one node is not a fatal risk.