Chalmers University of Technology

Computer Science and Engineering
Elad Michael Schiller

2021/08/26

Written exam in EDA387/DIT663 Computer Networks 2021-08-26. Exam time: 4 hours.

A remote exam via Canvas.

Examiner: Elad Michael Schiller, phone: 073-6439754

Credits:	30-38	39-47	48-Max
Grade:	3	4	5
Grade (GU)	G	G	VG

1. Exam problems that require text-based answers should be written and submitted in a text document, e.g. Word or PDF.
   - Use the answer templates Question_01.docx, Question_02.docx, and Question_03.docx --- one text document for each such exam problem. Please see the 'exam assignment' on the Canvas page for these files.
   - Name your text document Question_YY. *Example:* Question_01.pdf
   - Submit your answers by uploading the text documents via Canvas before the due date/time.
   - Exam problem solutions involving calculations, figures, diagrams, etc. can also be solved on paper as in a normal exam. Make sure that each paper is clearly marked with your name, exam problem number, and the page number. Scan or photograph your solutions. Make sure to have good lighting and preferably use a document scanning app, e.g. even CamScanner or Genius Scan. Name your image files Question_YY_Page_XX. *Example:* Question_01_Page_02.jpg.
2. The best is if you can combine images for the same problem into a single document (e.g. Word or PDF) named Question_YY.
3. Submit your solutions by uploading the image files or documents via Canvas before the due date/time.
4. The answer must be written in English (also for Swedish students). Use proper grammar and punctuation.
5. All answers need to be motivated, unless otherwise stated. Correct answers without motivation or with wrong motivation will not be given full credit. Answer concisely but explain all reasoning. Draw figures and diagrams when appropriate. Write clearly. Unreadable or hard-to-read handwriting will not be given any credit.
6. Do not hand in anything that is not part of the solution.
7. Always write your answers in your own words, using your own diagram, etc. In case you have to use the words of others, e.g., a formal definition, please make sure to use quotation marks and clearly identify the source. Aboutsltly do not collaborate in any way or form with other people during the entire exam time.

Question 1: Socket programming (14 points in total)

Each of the following parts of a program contains a flaw. Identify and describe the flaw in a few short sentences or points. You do not have to correct the flaw; you should just find and describe it! (Note: you're not looking for, e.g., syntax errors. Find conceptual flaws in the program.)

(1.a) **(3 points)** The following program has two sockets, sockA and sockB, from which the system expects to receive 24-byte messages. Each message is to be stored in a variable of type message_t, which is sufficiently large to contain the message. The messages are processed using the process_message() method. It is assumed that this method is capable of authenticating the message and checking that the message has the correct format. You can also assume that the handle_*_error() methods do something sensible.

Please write your answer in your own words, and only your own words. Please clearly state the flaw, the exact lines numbers, and why it might occur. Also, you need to say in detail what the programmer forgot to consider when writing the code. Please focus on just one flaw.

```
/* includes, declarations, etc. */

static bool receive_message_from_socket( int sock )
{
        message_t msg;
        ssize_t receivedBytes = 0;

        // make sure that we receive the whole message
        while( receivedBytes < sizeof(msg) )
        {
                ssize_t ret = recv( sock,
                        ((char*)&msg) + receivedBytes,
                        sizeof(msg) - receivedBytes,
                        0
                );

                if( 0 == ret )
                {
                        // peer disconnected; close socket and return false to
                        // indicated that the connection has closed.
                        close( sock );
                        return false;
                }

                if( -1 == ret ) handle_recv_error();

                receivedBytes += ret;
        }

        // OK, process message and return success
        process_message( msg );
        return true;
}

int main()
{
        /* initialization code has been omitted */

        while( sockA != -1 || sockB != -1 )
```

```
		{
			// initialize read set
			fd_set readfds;

			FD_ZERO( &readfds );
			if( sockA != -1 ) FD_SET( sockA, &readfds );
			if( sockB != -1 ) FD_SET( sockB, &readfds );

			// call select(). parameters that are NULL (=0) are ignored by
			// select() - i.e. this is not an error!
			int maxfd = sockA;
			if( sockB > maxfd ) maxfd = sockB;

			int ret = select( maxfd+1, &readfds, 0, 0, 0 );

			if( -1 == ret ) handle_select_error();

			// check sockA for messages
			if( FD_ISSET( sockA, &readfds ) )
			{
				bool stillOpen = receive_message_from_socket( sockA );

				if( !stillOpen )
					sockA = -1;
			}

			// check sockB for messages
			if( FD_ISSET( sockB, &readfds ) )
			{
				bool stillOpen = receive_message_from_socket( sockB );

				if( !stillOpen )
					sockB = -1;
			}
		}

		/* de-initialization code has been omitted */

		return 0;
	}
```

(1.b) **(3 points)** The following program accepts new connections using the listenfd socket. The first byte sent by a client is expected to be an 8-bit ID.

- You may assume that the handle_*_error() methods do something sensible.
- The helper method register_client(client, id) verifies the client ID is acceptable and if that is the case, enters the client into a global list. Otherwise, it closes the connection.
- The method add_client_sockets_to_readfds() properly adds all active clients in the global list to the readfds. It returns the largest socket number it encounters.
- handle_registered_clients() handles clients that are ready to send data according to readfds, and removes clients that close their associated connections from the global list. No data is ever sent to the clients, the program only receives, and processes data sent to it.

Please write your answer in your own words, and only your own words. Please clearly state the flaw, the exact line numbers, and why it might occur. Also, you need to say in detail what the programmer forgot to consider when writing the code. Please focus on just one flaw.

```
/* includes, declarations, etc. */

int main()
{
        int listenfd = -1;

        /* initialization code, such as setting up a listening socket on
                listenfd, has been omitted – this is not the error you're looking for */


        while( 1 )
        {
                // initialize read set
                fd_set readfds;

                FD_ZERO( &readfds );
                int maxfd = add_client_sockets_to_readfds( &readfds );

                FD_SET( listenfd, &readfds );
                if( listenfd > maxfd ) maxfd = listenfd;

                // call select
                int ret = select( maxfd+1, &readfds, 0, 0, 0 );

                if( -1 == ret ) handle_select_error();

                // is there a new client waiting?
                if( FD_ISSET( listenfd, &readfds ) )
                {
                        sockaddr_in clientAddr;
                        socklen_t clientAddrLen = sizeof(clientAddr);

                        int client = accept( listenfd,
                                (sockaddr*)&clientAddr,
                                &clientAddrLen
                        );

                        if( -1 == client ) handle_accept_error();

                        // receive 8bit client ID
                        unsigned char id;
                        int ret = recv( client, &id, sizeof(id), 0 );

                        if( 0 == ret )
                        {
                                close( client );
                                continue;
                        }

                        if( -1 == ret ) handle_recv_error();

                        // register client
                        register_client( client, id );
                }

                // handle registered clients
                handle_registered_clients( &readfds );
        }
        return 0;
        }
```

(1.c) **(8 points)** Consider the socket API studied in class, i.e., socket(), bind(), listen(), accept(), connect(), accept(), select(), close(), and shutdown() as well as the libary functions inet_pton() and inet_ntop(). In your own words (and only using your own words) as well as using the relevant code fragments, please explain how to do the following.

(1.c.i) **(2 pt)** Using this API, can you implement a lock (mutex)?

(1.c.ii) **(2 pt)** Using this API, can you implement a timer?

(1.c.iii) **(2 pt)** Using this API, can you extend the amount of memory available for the process?

(1.c.iv) **(2 pt)** Using this API, can you find out the IP address of the host of the process?

……….

Question 2: self-stabilizing construction of minimum spanning tree (28 points in total)

Consider a computer network whose topology is of a general graph, *G*:=(*P*, *E*), where E⊆PxP is a set of weighted edges, such that each connect a pair of nodes in *P*. Assume that the network includes a set, *T*⊆*E*, of selected edges, such that *T* defines a spanning tree for which there is no other tree, *T'*⊆*E,* where the sum of weights for *T'* is smaller than *T*. In this case, we say *T* is a minimum spanning tree in *E*. The studied problem is to construct a minimum spanning tree (MST) for graph *G*. Your task is to provide, your own, self-stabilizing solution to the problem.

2.a (2 points) Please specify the set of legal execution for the given task.

2.b (12 points) In case you provide an original algorithm (or parts of your solution use an original algorithm), please provide the code of the algorithm. In case your solution uses algorithms that we learned in class (or parts of your solution use algorithms that we learned in class), please specify the exact names of these algorithms as well as the page numbers (or sub-chapter) in the book of these algorithms. In case your solution stacks several such building blocks, please explain how you put them all to work together. (A simple figure can help.) Please use your own words (and only using your own words) --- there is no need to copy code from the book, slides or any other sources.

2.c (8 points) Please state exactly the asymptotic cost of memory for your solution. Specifically, what is the maximum about of memory that any node would need? What is the largest message size that any node will send? Also, what is the maximum number of messages until the MST is guaranteed to be constructed? What is the highest number of communication rounds it would take to construct the MST?

2.d (6 points) In your own words (and only using your own words), please prove the correctness of your proposed solution. That is, show that when starting from any system state (configuration), the algorithm constructs a minimum spanning tree.

………

Question 3: self-stabilizing α-maximal-partitioning of a tree (18 points in total)

Consider a network, $T:=(P, E)$, whose topology is of a tree that has a single-center, cf. lab 2.4. <u>The nodes in the network are anonymous</u>, i.e., they have no globally unique identifiers and they are run the same algorithm. Let $\alpha>0$ be a positive integer. Let $P_1,P_2,\ldots, P_k$ be a partitioning of P, such that $\cup_x P_x = P$ as well as each $P_x \neq \emptyset, |P| \leq \alpha$ and $T_x:=(P_x, E_x)$ is a (connected) tree, where $x \in \{1,\ldots,k\}$ and $E_x \subseteq E \cap (P_x \times P_x)$. In this case, we say that $P_1,P_2,\ldots, P_k$ is an α-partitioning of *T*. Suppose that there are no $x, y \in \{1,\ldots,k\} : x \neq y$, such that $P_1, P_2,\ldots, (P_x \cup P_y),\ldots, P_k$ is also an α-partitioning of T. In this case, we say that $P_1,P_2,\ldots, P_k$ is an α-maximal-partitioning of *T*. Your task is to design a self-stabilizing algorithm for constructing an α-maximal-partitioning of *T*.

3.a (4 points) In your own words (and only using your own words), please suggest at least one application to the problem of α-maximal-partitioning of *T*. In what way can the parameter α play a role in your application? In what way does the maximality requirement plays a role? In what way does the assumption that the network is anonymous helps to simplify the requirements about the system? Please provide clear arguments to support your claims.

3.b (2 points) In your own words (and only using your own words), please define the set of legal executions for the given task.

3.c (6 points) In your own words (and only using your own words), please provide clear and exact pseudocode for the entire protocol that you propose. State all of your assumptions clearly, and in case you use solutions learned in class, please name them clearly. You also have to explain how to combine them with your solution. There is no need to copy code from the book, slides or any other sources.

3.d (6 points) In your own words (and only using your own words), please prove the correctness of your proposed solution. That is, show that when starting from any system state (configuration), the algorithm constructs an α-maximal-partitioning of *T*.