TDDE35/TEN2 and TDDD93/TEN2 – Large-scale distributed systems and networks
Final Examination: 8:00-12:00, Friday, June 1, 2018
Time: 240 minutes
Total Marks: 40
Grade Requirements: Three (20/40); four (28/40); and five (36/40).
Assistance: None (closed book, closed notes, and no electronics)
Instructor: Niklas Carlsson

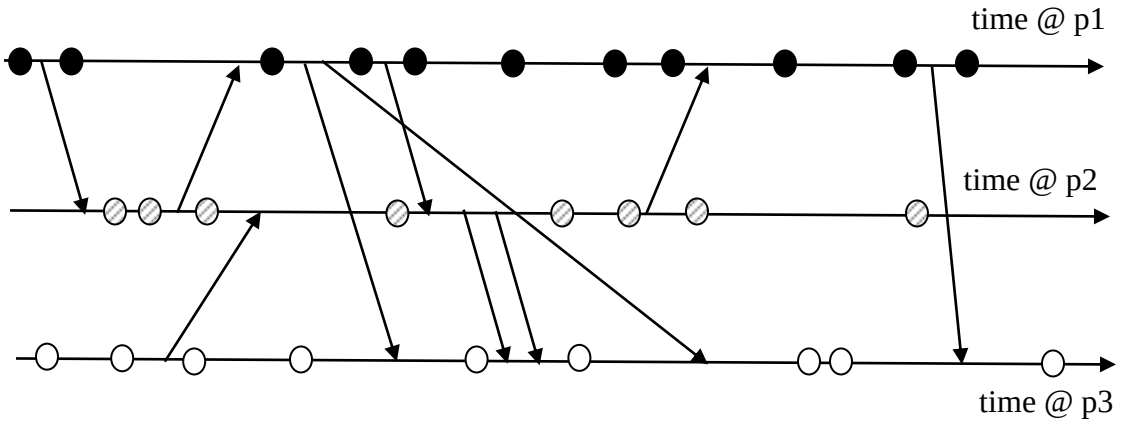
Instructions:

- Read all instructions carefully (including these)!!!! Some questions have multiple tasks/parts. Please make sure to address *all* of these.
- The total possible marks granted for each question are given in parentheses. The entire test will be graded out of 40. This gives you 10 marks per hour, or six minutes per mark, plan your time accordingly.
- This examination consists of a total of 13+1=14 questions. Check to ensure that this exam is complete.
- When applicable, please explain how you derived your answers. Your final answers should be clearly stated.
- Write answers legibly; no marks will be given for answers that cannot be read easily.
- Where a discourse or discussion is called for, be concise and precise.
- If necessary, state any assumptions you made in answering a question. However, remember to read the instructions for each question carefully and answer the questions as precisely as possible. Solving the *wrong* question may result in deductions! It is better to solve the *right* question incorrectly, than the *wrong* question correctly.
- Please write your AID number, exam code, page numbers (even if the questions indicate numbers as well), etc. at the top/header of each page. (This ensures that marks always can be accredited to the correct individual, while ensuring that the exam is anonymous.)
- Please answer in English and utilize figures and tables to the largest extent.
- If needed, feel free to bring a dictionary from an official publisher. Hardcopy, not electronic!! Also, your dictionary is not allowed to contain any notes; only the printed text by the publisher.
- Good luck with the exam.

Part A: Distributed Systems

Question 1 (4 points)

Assume that you have three processes p1, p2, and p3 which are implementing Lamport's clocks. There are many events that take place at these processes, including some messages being sent between the processes. In the figure below we use circles and arrows to specify in-processor events and messages being sent between processes, respectively. Please provide the logical timestamps associated with each event. You can assume that all three clocks start at zero, at the left-most point in time. (Also, explain how the processes would adjust their clocks if using Lamport's logical clocks.)



Question 2 (4 points)

Transparency plays a central role in some distributed systems. Consider a simple multi-tier system with three levels: a user interface, an application server, and two replicated database servers. Assume these layers are implemented as a distributed cloud service at different geographic locations and that the average round trip time (RTT) between the machines used in the consecutive layers (starting with the top-tier layer) is 20ms and 15ms, respectively. Consider a workload (set of calls) with two different types of "jobs" (call types). The first type results in fully synchronized calls in which the application server requires 30ms total processing and the database requires 60ms processing to satisfy the request. The second type does not require any database access, is fully synchronized and requires 50ms processing at the application server.

(a) For each of the two types of jobs, how much time is the client process locked from the moment it makes the request to the application server? You can assume that no large data is transferred between the layers such that the call and responses fits within a single package, and that messages do not need to be acknowledged. Please explain your answer and illustrate with a figure.

(b) Assuming 75% of the clients make each type of requests, what is the average response time (assuming no competing jobs or other reasons for queuing).

Please give concrete examples of two types of transparency that are provided in the above example. Remember to explain your answers.

Question 3 (2 points)

In the context of remote procedure call (RPC), please describe and compare at least two potential actions that a server orphan can take after the client has crashed while the server was computing.

Part B: Methodology

Question 4 (4 points)

When designing experiments, it is important to carefully identify the most appropriate factors, levels, and metrics to consider. Consider a researcher wanting to assess the performance of a webserver. The researcher has identified three factors of interest: (i) the request rate, (ii) the job size, and (iii) the processor speed. For each of these factors, the researcher has identified 9, 8 and 7 levels of interest, respectively, including identified a default request rate, job size, and processor speed. Let us call the request rate levels R1, R2, ..., R9; the job size levels S1, S2, ..., S8; and the processor speed levels P1, P2, …, P7. Please estimate the number of experiments that the researcher would need to perform if performing (a) one factor experiments with the default scenario as baseline, (b) two factor experiments with the default scenario as baseline, and (c) full factor experiments. Also, please explain which experiments would be performed in each case.

Question 5 (3 points)

Assume that you are interested in understanding the Autonomous Systems (ASes) that your internet packets take between two end points. For example, let us assume that you have access to PlanetLab and wants to estimate the path between a node located in California (USA) and a node in Italy. For this scenario, please describe a methodology for how you can estimate the AS path packets takes as they go from the PlanetLab node in California to that in Italy. Note that direction is important here. Please provide a figure of your experimental setup, describe the tools you would use, and how you would use them to solve the task at hand. (Also, as a semi-bonus questions, please give "guestimates" of the number of ASes and routers along such an example path. Note that in practice, having reasonable "guestimates" help quickly sanity check your results.)

Question 6 (3 points)

Consider a long duration video streaming session between a client and a server, for which it was observed that the average round trip time (RTT) between the client and server was 200ms and the average TCP window size was 40 packets, each of which is 1.5kB. It was also measured that each video frame is buffered on average 10s at the player. Please estimate the average video encoding and buffer occupancy measured in bytes? (**Hint:** You may want to use Little's law twice.)

Part C: Multicore and Parallel Programming

Question 7 (4 points)

Questions on parallel computer architecture.

a) Sketch the network topology (graph structure) for the 2D-mesh network. (Hint: annotated drawing). Analyze how its (A) average communication distance, (B) overall network cost (#links) and (C) maximum node degree grow with the number N of nodes connected by the network (give commented formulas in N). (Hint: 3 commented formulas in total. Assume here for simplicity that N is square number and the torus is organized accordingly.) (2p)

   If you do not recall the 2D-torus network, you may instead do it for any other interconnection network discussed in the lecture, with half the amount of points.

b) Since about 2004, all major processor manufacturers switched to multi-core designs. What was the main driving (physical) factor behind this technology shift? Explain your answer. (1p)

c) What does Moore's Law say? (0.5p)

d) Explain how "hardware multithreading" can improve the utilization of a CPU core. (0.5p)

Question 8 (1 points)

Question in thread programming. Mutex locks are often implemented using atomic test-and-set instructions. Why can so-called spinlocks using busy waiting, as presented in the lecture, become a performance problem in multithreaded program execution on a multicore processor, and what (short answer) could be done about it? (1p)

Question 9 (2 points)

Questions on MPI/algorithm design. You have a cluster computer running P (P>1) MPI processes. Process 0 has an array A of N elements, e.g., float values. Each process also allocates in its main memory an array B of N/P elements.

i) Design a parallel message-passing program (using all P nodes), using MPI-like explicit *send()* and *receive()* operations only, that scatters the array A across all P processes so that afterwards each process i, $0 \leq i \leq P$-1, shall hold in its array B copy of the elements at positions iN/P ... $(i+1)N/P$-1 of array A. (MPI or pseudocode is fine, explain your code.) Hint: Keep it simple, no optimality is required here. (1p)

ii) Assume that the time for sending and receiving a block of K elements is $aK+b$ for constant system parameters $a > 1$ and $b > 1$, that each node can only send to or receive from one node at a time, and that there are direct network links between all nodes. Assume further that copying an element and other local operations on an element cost 1 unit of time. Derive the asymptotic time complexity for completing the entire scatter operation according to your algorithm of (i) (i.e.,

derive a formula in *N*, *P*, *a* and *b*; use big-O notation where appropriate, and explain your calculation). (1p)

iii) How does your algorithm perform if *N* = 1 while *P* grows very large? Describe the main idea of an alternative algorithm that is asymptotically faster, and explain why. (1p)

Question 10 (3 points)

Question on theory.

a) Explain Amdahl's Law (including its assumptions) and give its proof, and explain its implications for where to focus one's efforts when parallelizing a sequential program. (2p)

b) Give an example of a (parallel) speedup anomaly. (1p)

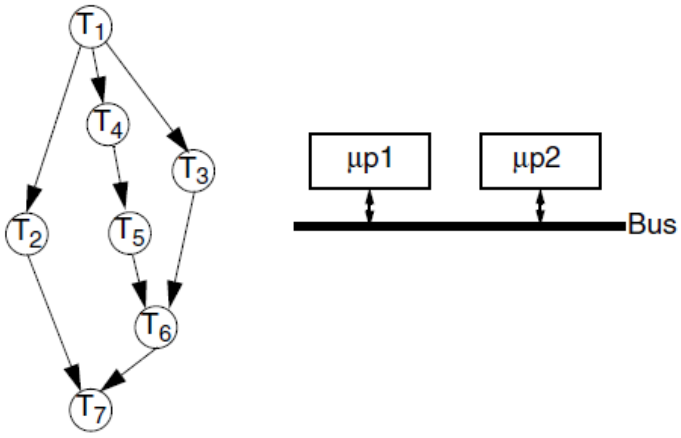
Part D: Embedded Systems

Question 11 (4 points)

Consider an application modelled as the task graph below. Each task, when activated, consumes one message on each input edge and generates, at termination, one message on each output edge. The task graph is executed on the architecture shown in the figure. Execution times of the tasks, when executed on the corresponding processor, are shown in the table. All messages transmitted over the bus, between tasks mapped on different processors, consume 2 time units to reach the destination. Communication between tasks mapped to the same processor is considered to not consume any time.

Propose an efficient task mapping (indicate on which processor each task is executed) and a corresponding static (nonpreemptive) schedule for the application. Illustrate your schedule as a Gantt chart (similar to the way we captured schedules in Lecture 1&2).

Try to achieve a maximum delay (the time interval between the start of T1 and the finishing of T7) of 46 time units!



Task	WCET	
	µp1	µp2
T_1	5	6
T_2	12	15
T_3	10	11
T_4	5	6
T_5	3	4
T_6	17	21
T_7	10	14

Question 12 (3 points)
What is an Embedded System? What makes it different from other applications? Why is it difficult to design?

Question 13 (3 points)
Why is power consumption an important issue in today's computer systems?

Part E: Bonus Part

Question 14 (4 points)
Peer-to-peer vs. client server comparison. Please derive expressions for how much time it takes to distribute file from one server to N clients/peers when using the client-server model and peer-to-peer, respectively. You can assume that the server has an upload bandwidth U, that each peer i has an upload bandwidth of u_i, that each peer i has a download bandwidth d_i, and that the file is of size F. Then use the expressions to and plot the distribution time as a function of the number of clients/peers in the system for the two delivery models, for the case when using the normalized file size $F = 1$ (i.e., the file size is measured in units of the file size itself) and $U = u_i = 1$ (i.e., the upload rate is measured in the time units that it takes the server to upload one copy of the file and the clients have the same upload rate) and $d_i = 2$ (i.e., the maximum download rate of a client is twice its upload rate). For simplicity, assume that the file can be broken into infinitesimally small chunks and that a client can start help out as soon as it has obtained such small chunk.

Good luck!!