Department of Computer Science	Exam **Distributed Systems**
Vrije Universiteit	Exam code: X_400130
Dr. T. Kielmann	04 – 02 – 2015

Please make sure that your handwriting is readable!

This is a *"closed book"* exam.
No printed materials or electronic devices are admitted for use during the exam.
You are supposed to answer the questions **in English**.

Wishing you lots of success with the exam!

Grading: The final grade is calculated by accumulating the scores per question (maximum: 90 points), and adding 10 bonus points. The maximum total is therefore 100 points. To pass the exam, it is sufficient to get at least 55 points.

1. Failures

1.a: Explain the differences among an *asynchronous system*, a *synchronous system*, and a *partially synchronous system* and which kinds of failures can be detected with such systems. 6pt

1.b: How large must a k-fault tolerant group be for halting failures and for arbitrary failures? 4pt

1.c: Consider a k-fault tolerant group, with $k > 1$. Assume that one process fails. Do we still have a k-fault tolerant group? Explain your answer! 4pt

1.d: What is being achieved using the Paxos algorithm? Which kind(s) of failures can be handled by Paxos? 4pt

2. RPC/RMI

2.a: How does RPC achieve location transparency? What are the limitations in this respect? 3pt

2.b: How does Java RMI overcome some of these limitations? 3pt

2.c: If your middleware only supports synchronous RPC, what can you do to improve geographical scalability? 3pt

2.d: When objects are replicated across multiple servers, which additional problem is caused with remote method invocation on such replicated objects? How can this problem be solved? 6pt

3. Naming

3.a: Explain (briefly!) the concepts of *name*, *address*, and *identifier*. 3pt

3.b: In a *Hierarchical Location Service*, the underlying network is divided into hierarchical domains; each domain is represented by a separate directory node. How are lookups performed in such a system? What is the scalability problem and how can it be resolved? 5pt

3.c: Using the global DNS system, there are issues with *size scalability* and *geographical scalability*. Explain how these problems are caused and how they are addressed. 4pt

4. Causality

4.a: Show that logical clocks do not necessarily capture potentially causal relationships. 4pt

4.b: When using vector clocks for enforcing causally ordered multicasting, $VC_i[i]$ is incremented only when process P_i sends a message, and sends VC_i as a timestamp $ts(m)$ with message m. How should we interpret the following two conditions for delivering m when received by process P_j: 4pt

1. $ts(m)[i] = VC_j[i] + 1$.
2. $ts(m)[k] \leq VC_j[k]$ for $k \neq i$.

4.c: Take $VC_2 = [0, 2, 2]$ and a message m being sent by process P_0 with $ts(m) = [1, 3, 0]$. What information does P_2 have, and what will it do when receiving m (from P_0)? 4pt

4.d: We always carefully talk about tracking "potentially" causal relationships by middleware. Why "potential?" 2pt

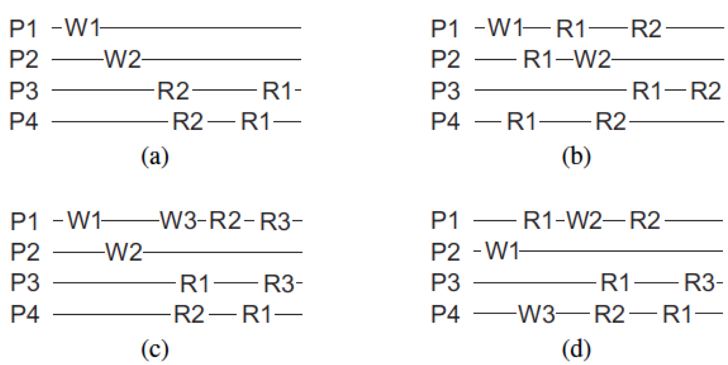
5. Gossiping

5.a: Explain the basics how gossiping algorithms work. 4pt

5.b: How can one delete an item that has been distributed with gossiping? 6pt

6. Consistency

6.a: Consider the following four execution. We write $W1$ as an abbreviation for $W(x)1$, and likewise $R2$ for $R(x)2$, where x is the variable shared by the processes $P_1, \ldots, P_4$. Which of these four executions are sequentially consistent, and which ones are not? Explain your answer. Hint: think twice in cases (c) and (d). 6pt



6.b: Give an eample where using only client-centric consistency will lead to a conflict between update operations. 5pt

7. Unstructured peer-to-peer (P2P) systems

7.a: Describe the graph structure of an unstructured P2P system. 3pt

7.b: Which two approaches can be applied to *searching* in an unstructured P2P system? How do these approaches compare w.r.t. completion time and communication effort (no formulas needed)? 7pt