1. Exam SS 2013
Distributed Systems

Prof. Dr. Katinka Wolter
Benjamin Güldenring
Date: July 9, 2013

Freie Universität Berlin

Name:		Bachelor ☐	Magister ☐
First Name:		Master ☐	Lehramt ☐
Matr.No.:		Diplom ☐	☐

Instructions

- Please write your **name** and **student ID number** (Matrikelnummer) on **every page**.
- Please write your solutions in the textboxes of the exam pages. If you need additional space, blank pages will be provided. **Do not use your own brought along paper**!
- Hand in **every page** at the end of the exam!
- No aids are allowed, except for the course material and other paper work. **Mobile phones** have to be **switched off** and stored in your bag.
- Please use only **permanent pens**, e.g. ballpens, with **blue** or **black** color. Exams written with pencils will not be accepted.
- The exam duration is **90 minutes**.
- Please explain all your answers and at least sketch your lines of thought.
- Please have your **student ID** and **photo ID** ready.
- I agree with publication of my exam result and student ID on the course web page: ☐
- You need **25** of **50** points to pass.

With my signature I attest to have read and understood these instructions.

_____
Signature

Problem	1	2	3	4	5	6	7	∑
Points	8	6	8	8	8	6	6	50
Achieved								

Grade: _____

Problem 1 (Terminology) (2+2+2+2) = 8 Points

Please explain the following

1. What is the difference between *total order* and *logical order*?

2. What is the difference between a *process* and a *thread*?

3. What is *virtualisation*?

4. How does *message-oriented communication* work?

Problem 2 (Chord) (2+2+2) = 6 Points

Assume a ring created by the Chord protocol, as shown in Figure 1.

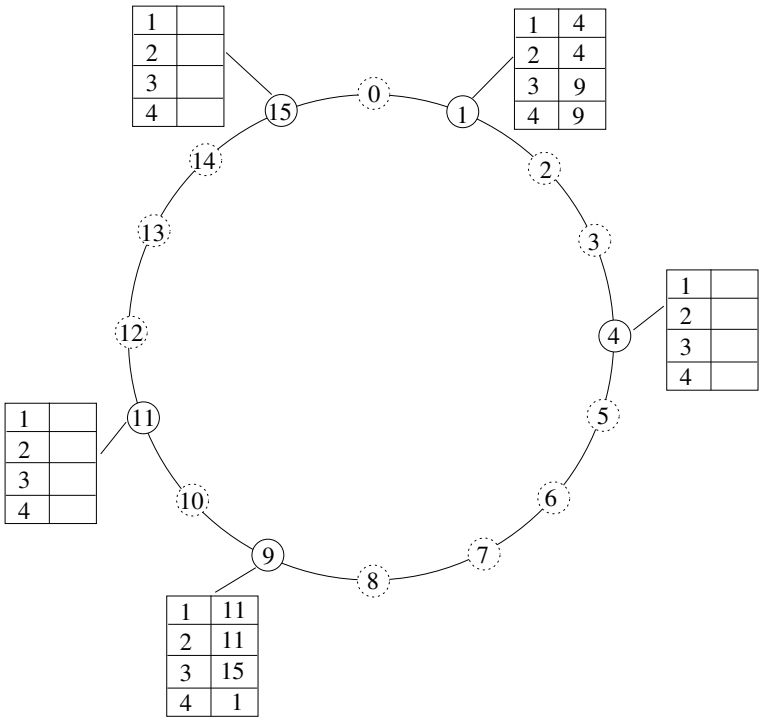

Figure 1: Overlay constructed using Chord

1. Please fill the empty slots in the finger tables

2. Explain the lookup of item 14 when starting from node 1.

3. Assume a node with ID 3 wants to join the network. How do the finger tables change?

Problem 3 (Overlay Tree) (4+4) = 8 Points

The overlay tree in Figure 2 can be used to implement a multicast. The solid links are labelled with the transmission cost of a message. The dashed links connect the overlay. To determine the quality of this application-level

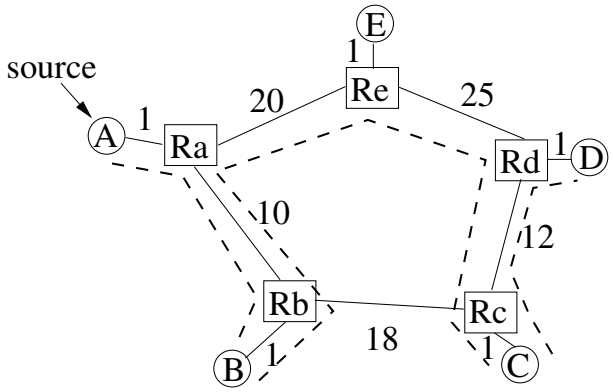

Figure 2: Overlay tree for multicast

tree, analyse only the connection between node A and node D. Please compute and explain

1. the **link stress** and

2. the **stretch** of the overlay.

Problem 4 (Clocks) (2+2+4) = 8 Points

Assume three nodes want to exchange messages for communication. They have the clocks as shown in Figure 3.

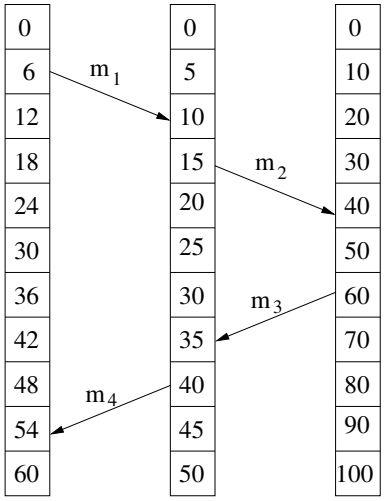

Figure 3: Logical Clocks

1. Please explain the problems that may arise and how they can be solved using logical clocks.

2. Please correct the clocks shown in Figure 3.

3. Please add a message, that happened before m_1 and explain how many options you have to place it.

Problem 5 (Leader Election) (4+4) = 8 Points

Determine the complexity of the Breadth First Algorithm, which we studied as leader election in wireless networks, as defined below.

Algorithm 1 *Assume a set of n nodes, strongly connected by a set E of edges. Each node keeps a list of k neighbours. Nodes exchange information messages in rounds.*
The idea of the algorithm is that it sets up a tree structure and elects the node with the largest parameter value. If several such nodes exist the leader is selected randomly.

1. *Initially a node with ID u_0 sets itself up as root and sends a message to its k neighbours asking them to become child nodes.*
2. *A node with ID u_i that receives one or more requests to become child node, answers with its ID to an arbitrary one and passes the same request on to its other $k-1$ neighbours that did not send a request in the first place.*
3. *If a node has already committed to a parent, it sends a* decline *message back.*
4. *If a node receives* decline *messages from all its neighbours it is a leaf node and sends its parameter value and ID to its parent node.*
5. *A node that receives from all its children parameter value and ID selects the largest and passes it on to its parent.*
6. *The root node selects the largest parameter and ID and transmits this through its children to all nodes of the tree as the* leader.

- Please determine the number of messages that must be sent as a function of n and k. Explain your answer.

- Please determine the runtime complexity of the algorithm as a function of n and k. Explain your answer.

Problem 6 (Commit Protocols) (3+3) = 6 Points

1. Please explain the 2-phase-commit protocol

2. How does the Paxos protocol differ from 2PC?

Problem 7 (PeerSim) (6 Points)

Consider the following distributed algorithm that calculates the global average over all nodes in an overlay network:

- Each node is assigned a value.
- At regular intervals each node (S in the following) sends its value to one randomly chosen neighbor node (R in the following).
  - When R receives the value of S, it first calculates the average of it's value with the one of S,
  - R then sends the difference of its value with the average back to S
  - and R sets it's value to the average afterwards.
- When any node receives a difference, it adds it to it's value.

Unless otherwise noted we assume that messages are never lost and ignore any rounding errors that may occur.

1. Implement the above algorithm as Peersim simulation by filling out the boxes in the source code listing in figure 4. Messages are implemented by the class `AverageMessage`. The `sender` property depicts S in the above description for messages sent form S to R and shall be `null` for messages sent from R to S.

2. Consider the configuration file in figure 5. What do you expect as the final result of the experiment?

3. Now assume that messages can get lost. Does the algorithm still work? If so, present an informal argument. If not, give a counterexample.

```
1  public class AverageDiff extends SingleValueHolder
2                  implements CDProtocol, EDProtocol
3  {
4          public void nextCycle( Node node, int pid ){
5                  Node peern = ... //randomly selected neighbor node
6                  ((Transport)node.getProtocol(FastConfig.getTransport(pid))).
7                          send(
8                                  node,
9                                  peern,
10                                 new AverageMessage( [        ], [        ]),
11                                 pid);
12         }
13
14         public void processEvent( Node node, int pid, Object event ) {
15                 AverageMessage aem = (AverageMessage)event;
16                 if(aem.sender != null)
17                 {
18                                 [                                        ]
19                                 [                                        ]
20                                 [                                        ]
21                                 [                                        ]
22                                 [                                        ]
23                                 [                                        ]
24                                 [                                        ]
25                                 [                                        ]
26                                 [diff =                                  ]
27                                 [value =                                 ]
28                   ((Transport)node.getProtocol(FastConfig.getTransport(pid))).
29                         send(node,
30                         aem.sender,
31                         new AverageMessage( [        ], [        ]),
32                         pid);
33                 }
34                 else
35                   value = [                                    ]
36         }
37 }
38
39 class AverageMessage
40 {
41         final double val;
42         final Node sender;
43         public AverageMessage( double val, Node sender ){
44                 this.val = val;
45                 this.sender = sender;
46         }
47 }
```

Figure 4: Peersim source code for problem 7

```
1  SIZE 100 # network size
2  CYCLES 1000
3  CYCLE SIZE*10000
4  MINDELAY 10
5  MAXDELAY 80
6
7  network.size SIZE
8  simulation.endtime CYCLE*CYCLES
9  simulation.logtime CYCLE
10
11 ################## protocols ===========================
12
13 protocol.link peersim.core.IdleProtocol
14
15 protocol.avg AverageDiff
16 protocol.avg.linkable link
17 protocol.avg.step CYCLE
18 protocol.avg.transport urt
19
20 protocol.urt UniformRandomTransport
21 protocol.urt.mindelay (CYCLE*MINDELAY)/100
22 protocol.urt.maxdelay (CYCLE*MAXDELAY)/100
23
24 ################## initialization ======================
25
26 init.rndlink WireKOut
27 init.rndlink.k 10
28 init.rndlink.protocol link
29
30 init.vals LinearDistribution
31 init.vals.protocol avg
32 init.vals.max SIZE
33 init.vals.min 1
34
35 init.sch CDScheduler
36 init.sch.protocol avg
37 init.sch.randstart
38
39 ############### control ============================
40 # outputs min, max, n, sum/n, variance, countmin, countmax
41 control.0 SingleValueObserver
42 control.0.protocol avg
43 control.0.step CYCLE
```

Figure 5: Peersim config file for problem 7