Distributed Systems Group
Department Software Technology
Faculty EEMCS
DELFT UNIVERSITY OF TECHNOLOGY

Examination Distributed Algorithms (IN4150)

4 April 2016, 1:30-4:30 PM

Notes

1. The **number of exercises** in this exam is 3, and the **number of pages** of this exam is 2.
2. Do give **clear, precise, and concise** answers.
3. The **maximum number of points** to be obtained for each part of each exercise is indicated between parentheses.

   New this year These numbers of points add up to 19. The remaining 3 points can be earned for the clarity, preciseness, and conciseness of the answers. Students are strongly encouraged to first write (or at least sketch) their answers on scratch paper before writing them on the sheets they will hand in.
4. The **final grade** for the exam is computed as 12 plus the total number of points obtained for the three exercises and the paper summary divided by 10 and rounded to the nearest integer or integer+0.5.
5. The solutions to the exercises can be either **in Dutch or in English**.

1. (a) (6) Give in words the algorithm of Chandy and Lamport for detecting the global state of a distributed system.

   Consider in (b) and (c) below a system consisting of two bank accounts (processes) A and B with initial value 100 each, and a separate channel each way between the accounts.

   (b) (4) Suppose that A sends two messages with amounts of 50 each to B, and vice versa. Show an execution of the algorithm in which the recorded states of both processes are 50 and the recorded states of both channels consist of a single message with an amount of 50.

   (c) (4) Suppose now that A sends a single message with an amount of 100 to B, and vice versa, and suppose that both processes start the algorithm spontaneously. Is it possible that the global state that will be recorded has not actually occurred in the system? If yes, show an execution of the algorithm in which is the case. If no, argue why this is not posssible.

   (d) (5) Suppose in general that the global state detected by the algorithm has not actually occurred. Explain how the order of events that have taken place can be changed in such a way that the detected state would have occurred.

2. (a) (6) Give definitions of the concepts of *fragment*, the *level* of a fragment, and the *minimum-weight outgoing edge* (MOE) of a fragment in the Minimum-weight Spanning Tree algorithm of Gallager, Humblet, and Spira (GHS).

   (b) (4) Explain the concepts of *merging* two fragments and of *absorbing* one fragment into another in this algorithm.

   (c) (5) If in the GHS algorithm fragment F_1 is absorbed by fragment F_2 and the node in F_2 to which F_1 is connected has already reported in the current search for the MOE of F_2, fragment F_1 is not included in this search anymore. Explain why this is indeed not needed.

   (d) (4) In the situation of (c), the node in F_2 to which F_1 is connected will send an `initiate` message to the connecting node in F_1. Describe the parameters of this message, and explain the actions of this connecting node upon reception of the message.

3. (a) (4) Formulate the Byzantine agreement problem. In particular, state the conditions for *agreement* and *validity*.

   (b) (6) Give in words the algorithm for randomized Byzantine agreement.

   (c) (4) Assume that there are 11 processes, two of which are faulty. Give an assignment of the initial values with which the correct processes start the first round and a scenario for the first round in which none of the correct processes proposes a 0 or a 1.

   (d) (5) Give a scenario in which the correct processes will never reach a decision.

Distributed Systems Group
Department Software Technology
Faculty EEMCS
DELFT UNIVERSITY OF TECHNOLOGY

Distributed Algorithms (IN4150)

List of algorithms

March 15, 2016

Below is a list of the algorithms in the order of treatment in the course, with the names of the original authors, if applicable.

Chapter 3: Synchronization

1. Alpha-, beta-, and gamma-synchronizer: Awerbuch
2. Causal message ordering (broadcast): Birman-Schiper-Stephenson
3. Causal message ordering (point-to-point): Schiper-Eggli-Sandoz
4. Total message ordering
5. Determining global states: Chandy-Lamport
6. Termination detection in a unidirectional ring
7. Termination detection in a general network
8. Deadlock detection for AND requests: Chandy-Misra-Haas
9. Deadlock detection for OR requests: Chandy-Misra-Haas
10. Deadlock detection for M-out-of-N requests (with/without instantaneous communication): Bracha-Toueg

Chapter 4: Coordination

1. Assertion-based mutual exclusion: Lamport
2. Assertion-based mutual exclusion: Ricart-Agrawala
3. Assertion-based mutual exclusion: Maekawa
4. Generalized assertion-based mutual exclusion
5. Token-based mutual exclusion: Suzuki-Kasami
6. Token-based mutual exclusion: Singhal
7. Token-based mutual exclusion (tree-based): Raymond
8. Detection of loss and regeneration of a token
9. Election in a synchronous unidirectional ring (non-comparison-based)
10. Election in a bidirectional ring: Hirschberg-Sinclair
11. Election in a bidirectional ring (enhanced version)
12. Election in a unidirectional ring: Chang-Roberts
13. Election in a unidirectional ring: Peterson
14. Election in a synchronous complete network: Afek-Gafni
15. Election in an asynchronous complete network: Afek-Gafni
16. Minimum-weight spanning trees: Gallager-Humblet-Spira

Chapter 5: Fault Tolerance

1. Agreement with stopping failures
2. Byzantine agreement with oral messages: Lamport-Pease-Shostak
3. Byzantine agreement with authentication: Lamport-Pease-Shostak
4. Randomized Byzantine agreement
5. Stabilizing mutual exclusion: Dijkstra
6. Stabilizing stop-and-wait datalink algorithm
7. Stabilizing sliding-window datalink algorithm