Distributed Algorithms

Consensus and Related Problems

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

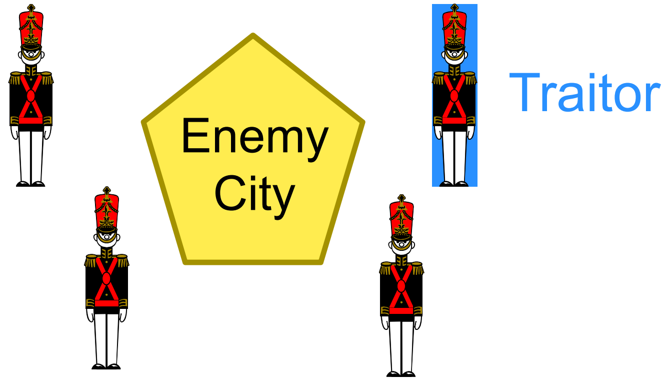
Architecture of Application Systems
Faculty for Computer Science and Electrical Engineering
University of Rostock

Overview

- Introduction (previous lecture)
- Masking fault tolerance (this lecture)
  - Byzantines general problem

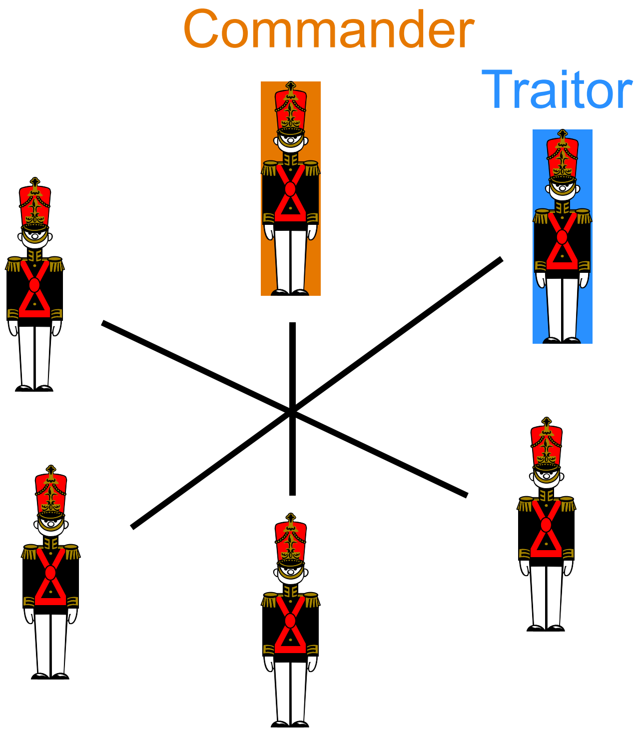
Motivation

- Several divisions of the Byzantines army camp outside an enemy city
- Each division is commanded by its own general
- The generals can only communicate by messengers
- After observing the enemy, they must decide upon a common plan of action (i.e., to attack or to retreat)
- However, some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement
- The generals need an algorithm that guarantees agreement despite some of them are traitors



Byzantine Generals (Lamport et al., 1982)

- $n > 3$ generals, m of them are traitors (cause byzantine errors)
- One of the generals is the commander and proposes a value $v \in \{0, 1\}$
- The other generals (lieutenants) shall execute the order of the commander
- At least one lieutenant is fault-free
- Commander can be a traitor, too
- Question to be answered: Attack together ($v = 1$) or wait ($v = 0$)?



Byzantine Generals – Assumptions

- Synchronous system model
- Each process is directly connected to every other process → completely meshed topology
- Messages
  - Arrive as sent
  - Do not get lost
  - Are not duplicated
  - Cannot be signed forgery-proof
  - Allow to determine the sender's identity

    → oral messages

Byzantine Generals – Algorithm for *m* = 1

1. Commander sends its value (0 or 1) to the others
2. Each lieutenant tells each other lieutenant the value it received from the commander
3. Each lieutenant makes a majority decision according to the received values

> Note: A faulty commander or a faulty lieutenant can send an arbitrary value or no value at all!

Byzantine Generals – Impossibility

> For m traitors and n generals, no algorithm exists that solves the byzantine generals problem for $n \leq 3m$

> Simplest not solvable special case: $n = 3$ and $m = 1$

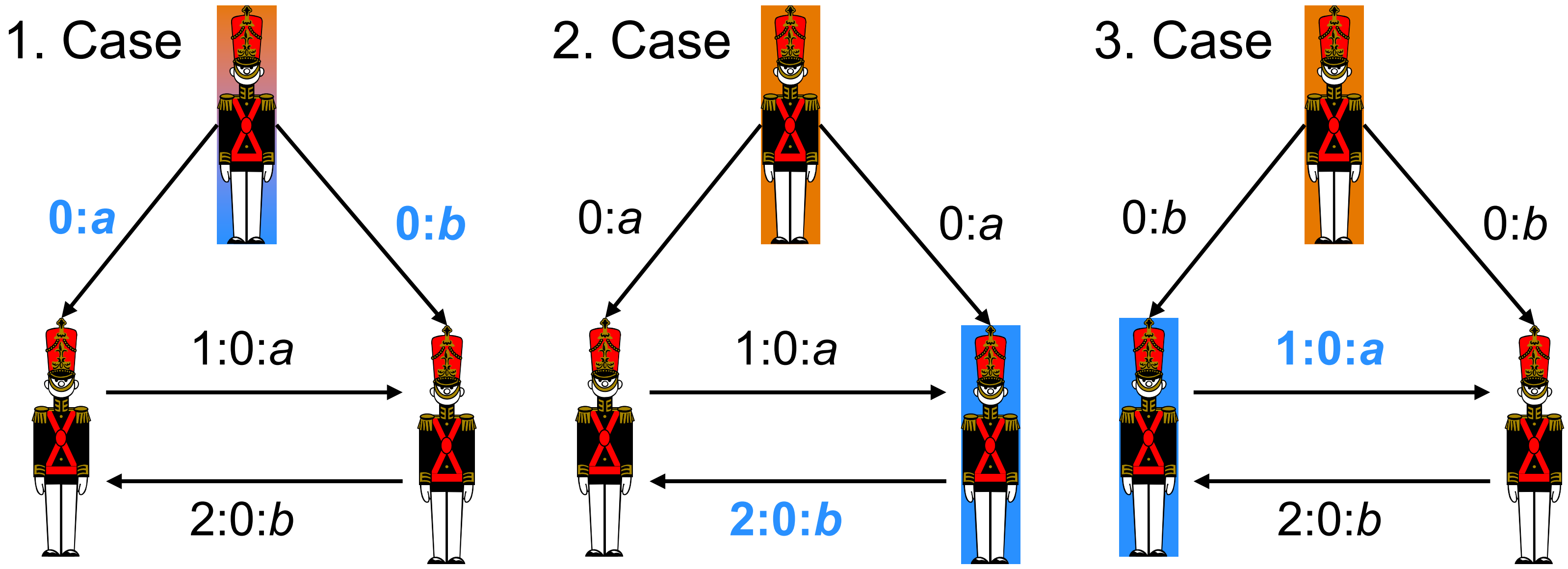
> Intuitive argument: How should a loyal general communicating with a loyal general and a traitor figure out who is who if they blame each other?

Impossibility for n = 3 and m = 1

$a \neq b$

- Case 1: Commander is erroneous → Correct generals receive {a, b}
- Case 2: Right general is erroneous → Left general receives {a, b}
- Case 3: Left general is erroneous → Right general receives {a, b}

Commander: 0
Lieutenants: 1, 2



Impossibility for $n = 3$ and $m = 1$

> Since the left general cannot distinguish case 1 from case 2, it has to choose value *a* given by the commander to fulfill C2

> Since the right general cannot distinguish case 1 from case 3, it has to choose value *b* given by the commander to fulfill C2

> But that means that both generals choose different values in the 1st case. Thus, C1 is violated

> Similar arguments apply for different approaches, e.g., for choosing a default value in case of different opinions

> Then, sending of a non-default value by a correct general leads to a contradiction

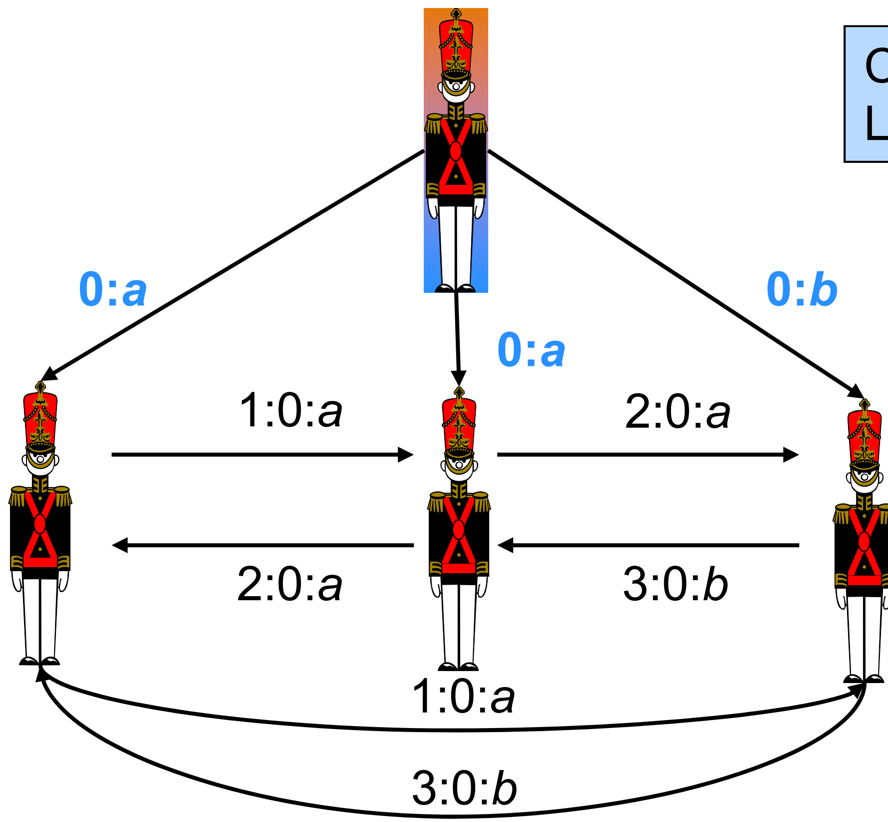
Byzantine Generals for $n = 4$ and $m = 1$

Most simple case with $n > 3m$

- Case 1: Commander is erroneous
  - Each lieutenant receives $\{a, a, b\}$ and, thus, decides for a

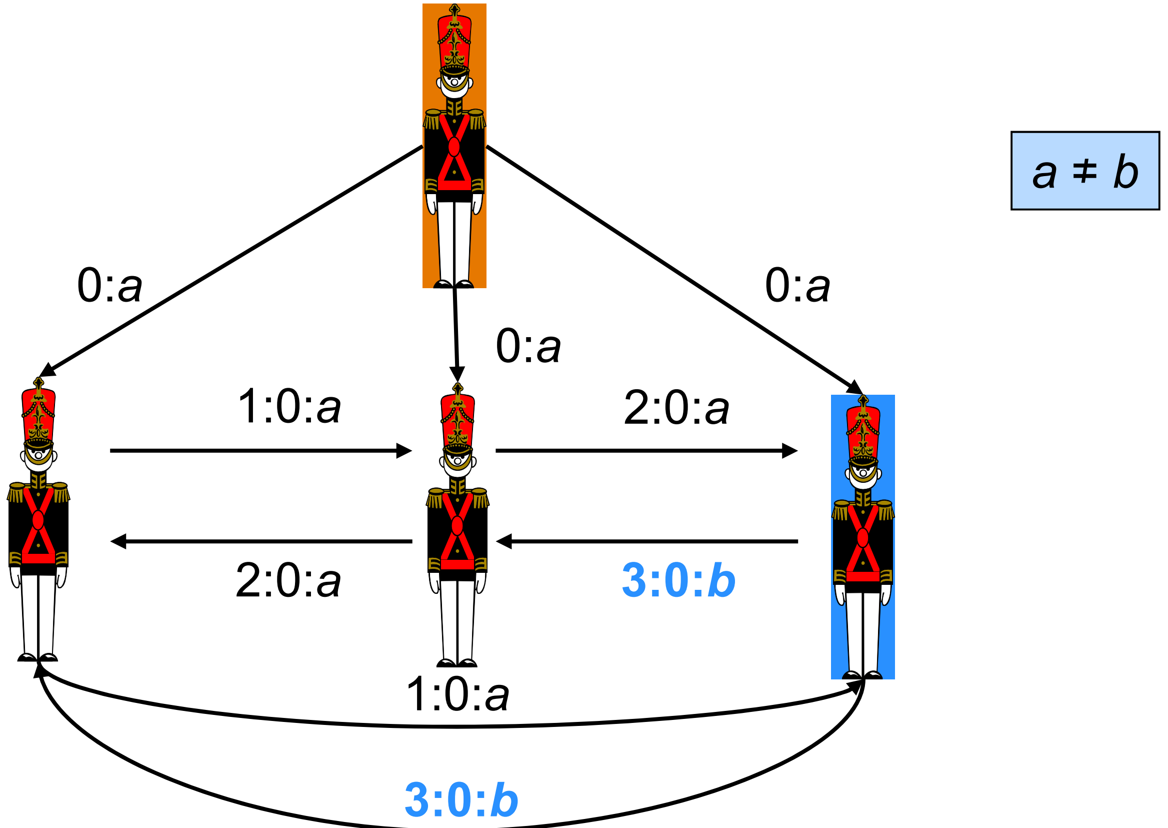
Commander: 0
Lieutenants: 1, 2, 3

$a \neq b$



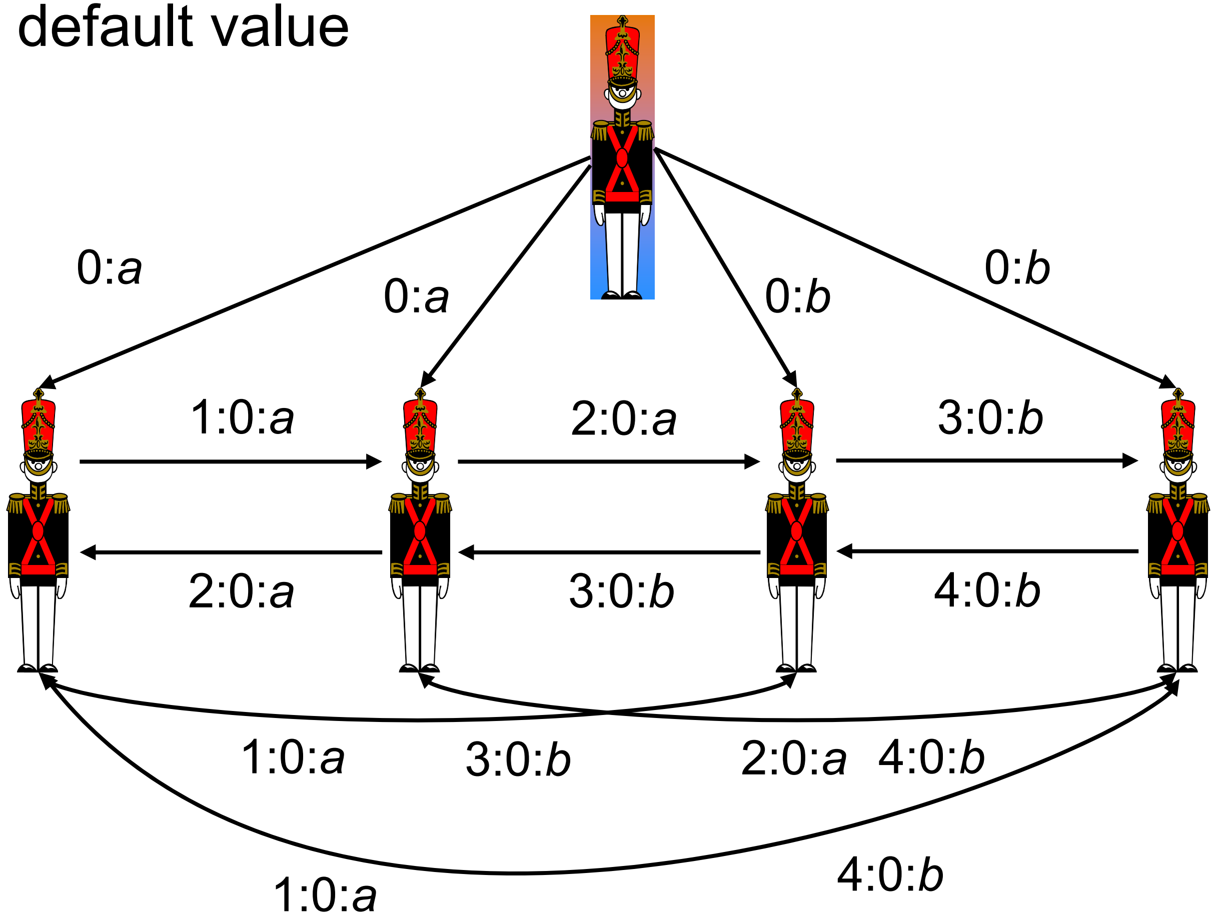
Byzantine Generals for n = 4 and m = 1

- Case 2: A lieutenant is erroneous
  - Left lieutenant receives {*a*, *a*, ***b***} and decides for *a*
  - Middle lieutenant receives {*a*, *a*, ***b***} and decides for *a*



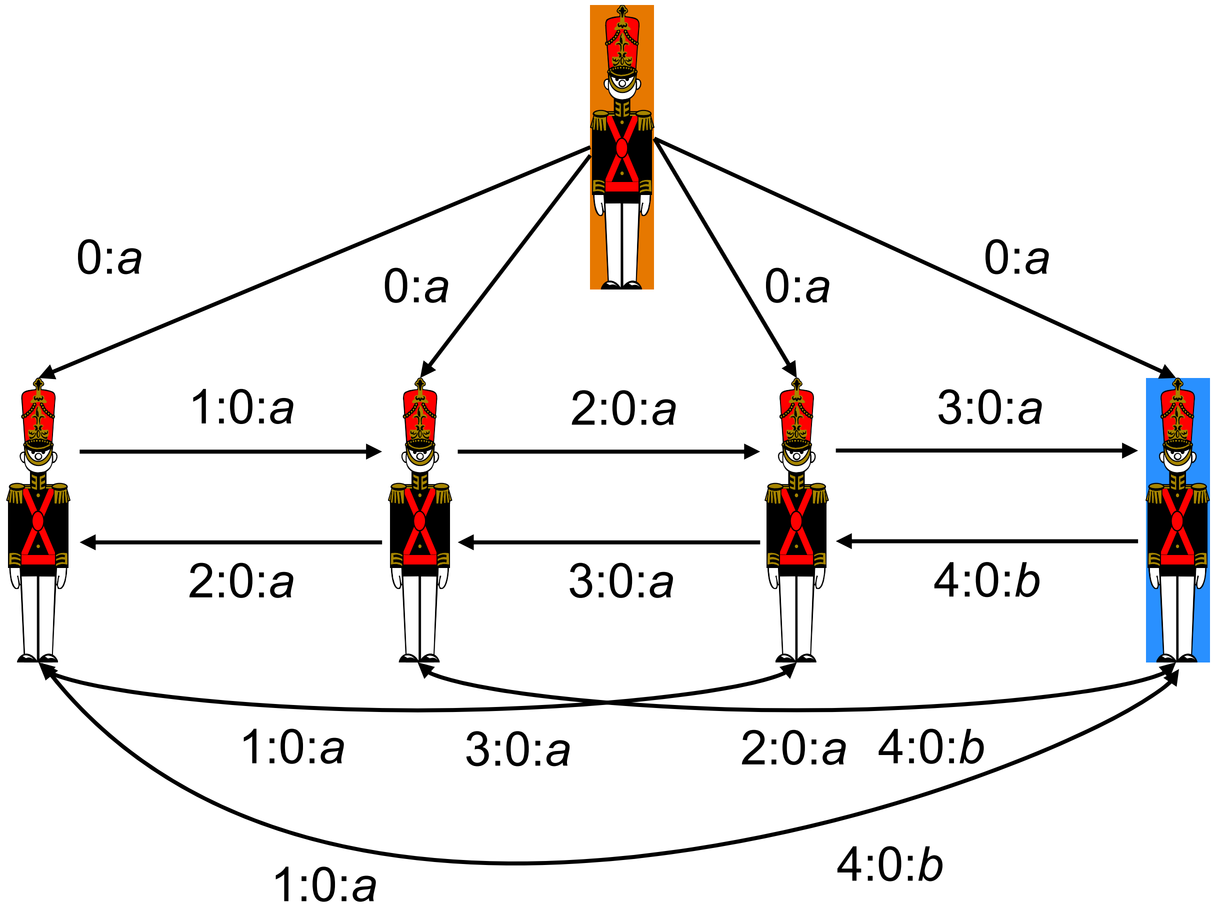
Byzantine Generals for *n* = 5 and *m* = 1

- Case 1: Commander is erroneous
  - Each lieutenant receives {*a*, *a*, *b*, *b*} and decides for the default value



Byzantine Generals for $n = 5$ and $m = 1$

- Case 2: A lieutenant is erroneous
  - Each lieutenant receives {*a*, *a*, *a*, *b*} and decides for *a*



Byzantine Generals

- Algorithm can be generalized to larger m through a recursive execution of the algorithm
  - The algorithm needs $m + 1$ rounds
  - There can be no algorithm with fewer rounds
  - ⇒ Unit time complexity $m + 1$
- With m erroneous processes, agreement is possible if there are at least $2m + 1$ correct processes
- ⇒ Since the barrier is hard, $n > 3m$ must hold
- ⇒ More than 2 / 3 of all processes must work correctly

Recursive algorithm *OM* for Oral Messages

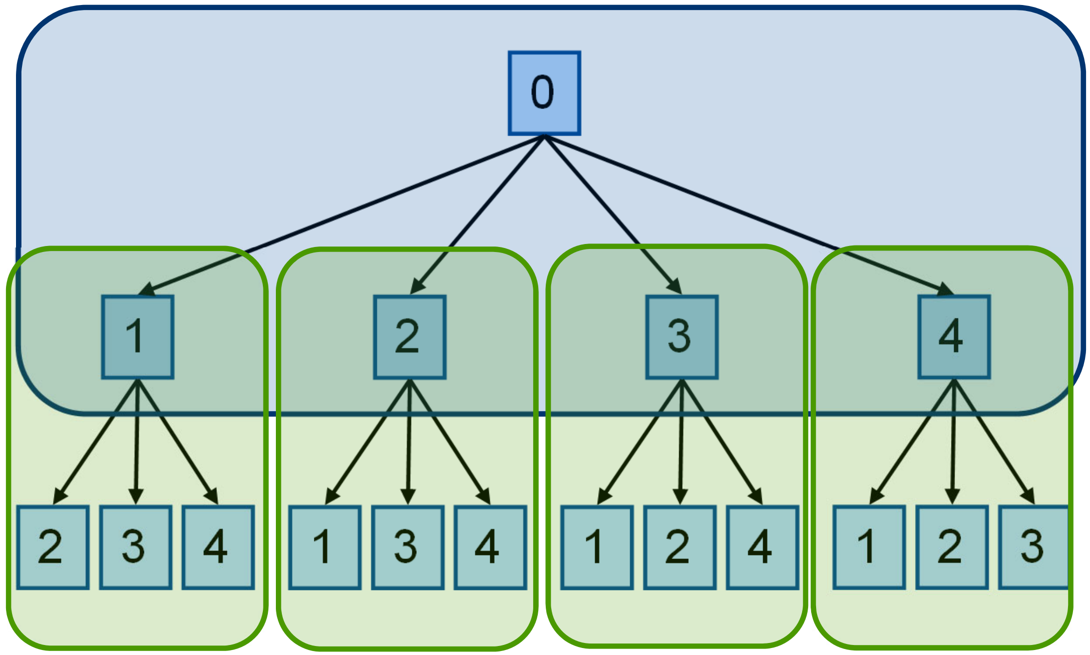
Commander: 0
Lieutenants: 1 to (n – 1)

```
Initial action at commander:
   OM(m, 0, {1, …, n - 1}, v)
Initial action at lieutenant L:
   M_L = {}

PROC OM(m, C, G, v) {
   FOREACH L in G DO
      SEND (m, G, C + ":" + v ) TO L;
   END
}

{Message (m, G, v) is received by lieutenant L}:
   IF <message is expected> THEN
      M_L := M_L ∪ v;
      IF m > 0 THEN
         OM(m - 1, L, G \ {L}, v); // recursive call
      FI
   FI

{Lieutenant L has received all messages}:
   v:=tree_majority(M_L);
```

Example for $n = 5$ and $m = 1$

1st round

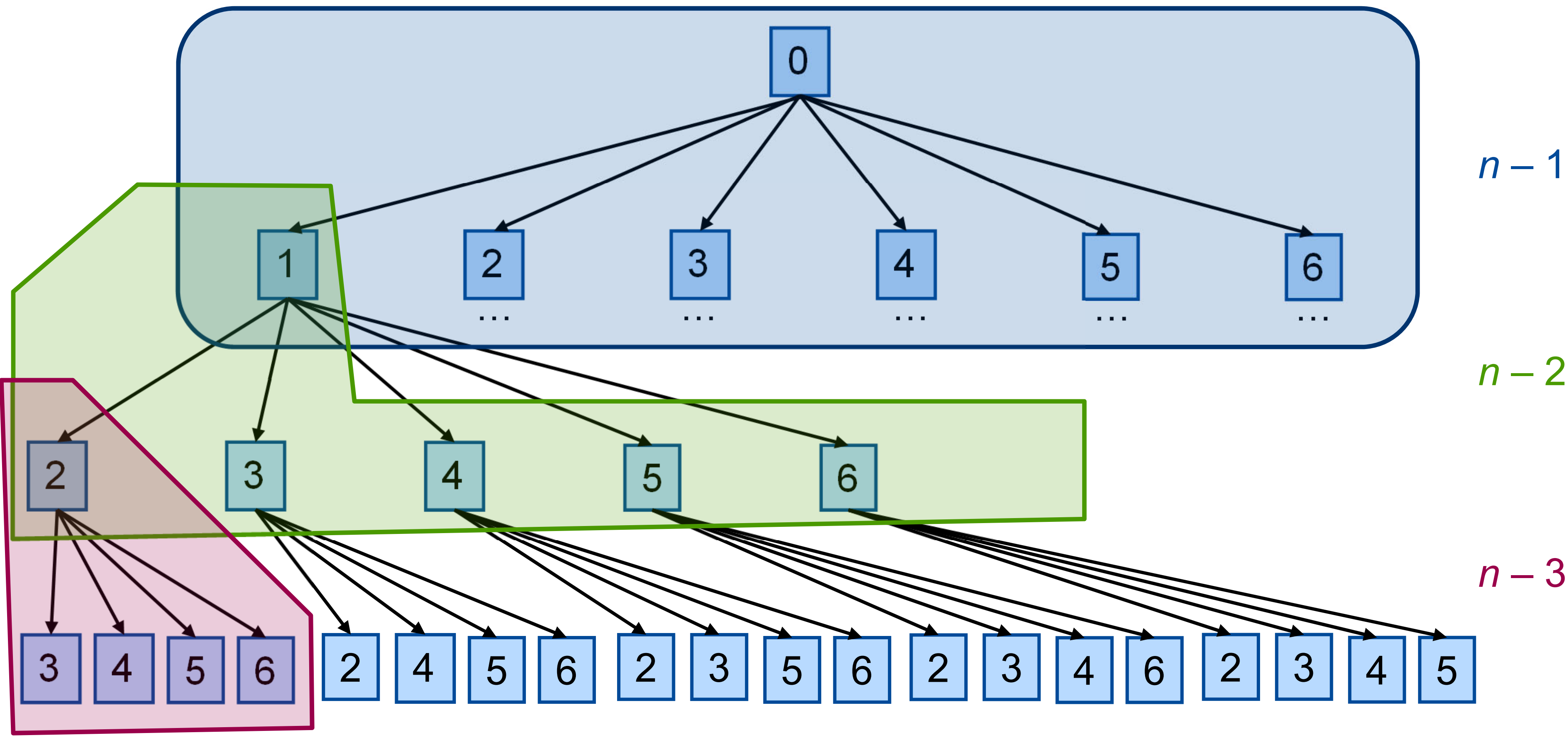
2nd round



$n - 1$

$n - 2$

Example for *n* = 7 and *m* = 2



Message Complexity

> One instance of $OM(m)$ starts $(n - 1)$ instances of $OM(m - 1)$
> Each instance of $OM(m - 1)$ starts $(n - 2)$ instances of $OM(m - 2)$
> Each instance of $OM(m - 2)$ starts $(n - 3)$ instances of $OM(m - 3)$
> …
> Each instance of $OM(1)$ starts $(n - m)$ instances of $OM(0)$

> Each instance of $OM(m)$ sends $(n - 1)$ messages
> Each instance of $OM(m - 1)$ sends $(n - 2)$ messages
> Each instance of $OM(m - 2)$ sends $(n - 3)$ messages
> …
> Each instance of $OM(1)$ sends $(n - m)$ messages
> Each instance of $OM(0)$ sends $(n - 1 - m)$ messages

Message Complexity

- 1st round: 1 instance with $n - 1$ messages
- 2nd round: $(n - 1)$ instances with $n - 2$ messages each
- 3rd round: $(n - 1)(n - 2)$ instances with $n - 3$ messages each
- …
- $(m + 1)$-th R.: $(n - 1)! / (n - 1 - m)!$ instances
  with $n - 1 - m$ messages each

- Derivation of the message complexity

$$\sum_{i=0}^{m}(n-1-i)\frac{(n-1)!}{(n-1-i)!} = \sum_{i=0}^{m}\frac{(n-1)!}{(n-2-i)!}$$
$$= \sum_{i=0}^{m}\prod_{j=0}^{i}(n-1-j) = n^{m+1} + c_m n^m + \cdots + c_0$$
$$= O(n^{m+1})$$

Assumption: Faulty generals do not send more messages than given by the algorithm.

Message Complexity

- $n = 4$, $m = 1$
  - $3 + 3 \cdot 2 = 3 + 6 = 9$ messages
- $n = 7$, $m = 2$
  - $6 + 6 \cdot 5 + 6 \cdot 5 \cdot 4 = 156$ messages
- $n = 10$, $m = 3$
  - $9 + 9 \cdot 8 + 9 \cdot 8 \cdot 7 + 9 \cdot 8 \cdot 7 \cdot 6 = 3{,}609$ messages
- $n = 13$, $m = 4$
  - $13 + 13 \cdot 12 + 13 \cdot 12 \cdot 11 + 13 \cdot 12 \cdot 11 \cdot 10 + 13 \cdot 12 \cdot 11 \cdot 10 \cdot 9 = 108{,}384$ messages

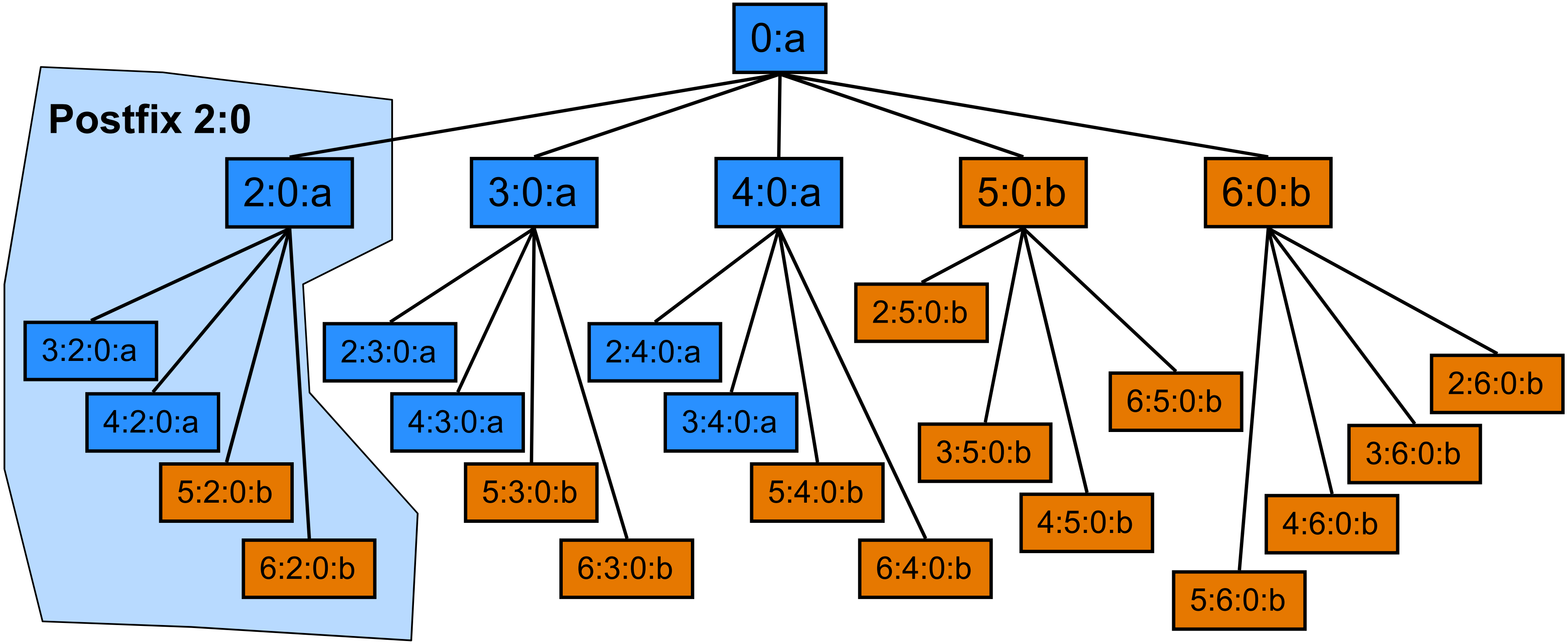
Message Tree / Tree Majority Function

- Each general builds a message tree from the messages it has received
- In this tree, the received messages are arranged according to the postfix of their message path

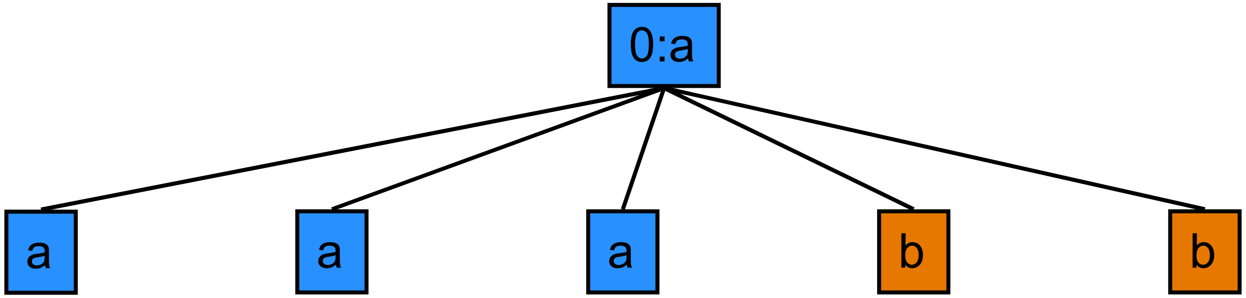
- Majority function is applied repeatedly, until a single value is derived, using the following method
  - Majority is formed for each node directly above the leafs from its value and the values of the leafs below it
  - If there is no majority, a (predefined) default value is used

Message Tree for General 1 ($n = 7$, $m = 2$)

> General 5 and 6 faulty, commander sends *a* to the nodes 1 to 6
> In the 2nd round, 5 and 6 send arbitrary values to the others; here *b* to create maximum confusion



Message Tree for General 1 ($n = 7$, $m = 2$)



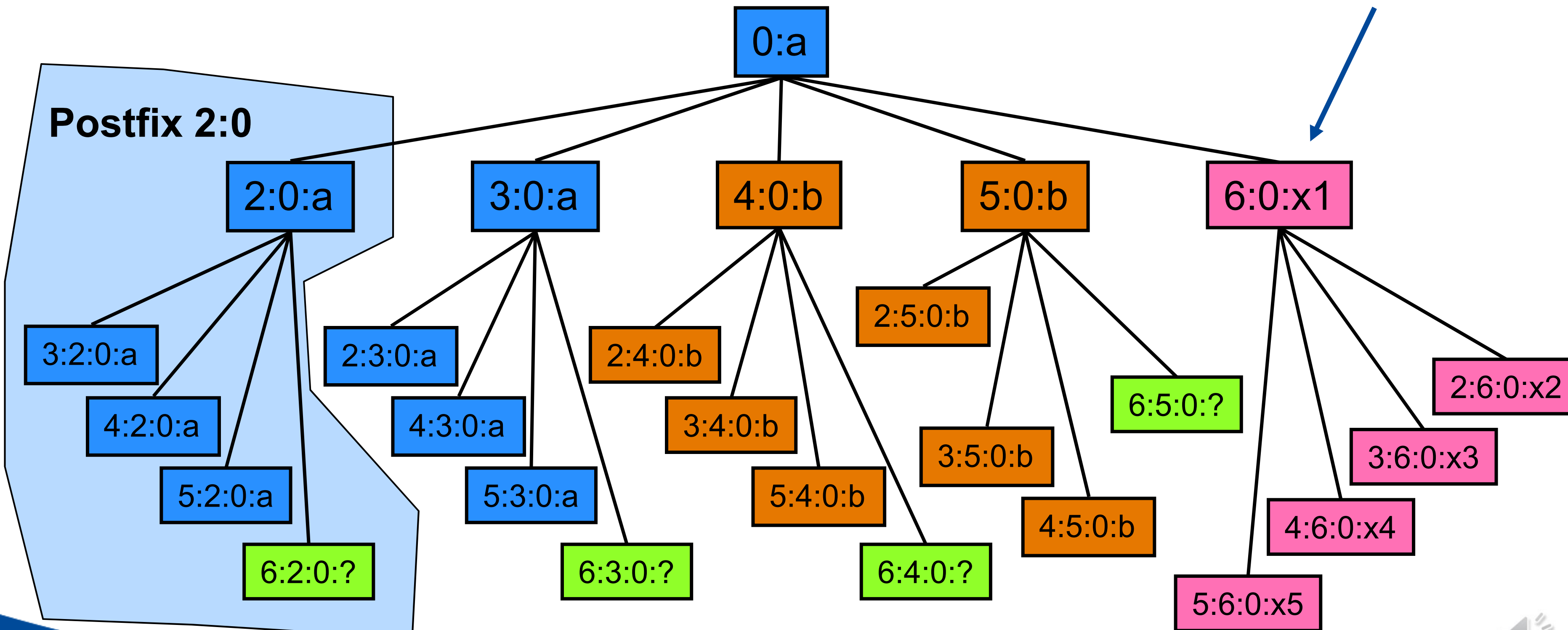
This figure shows the message tree of the previous slide after the first majority formation.

Message Tree for General 1 (n = 7, m = 2)

This figure shows the message tree of the previous slide after the second majority formation.

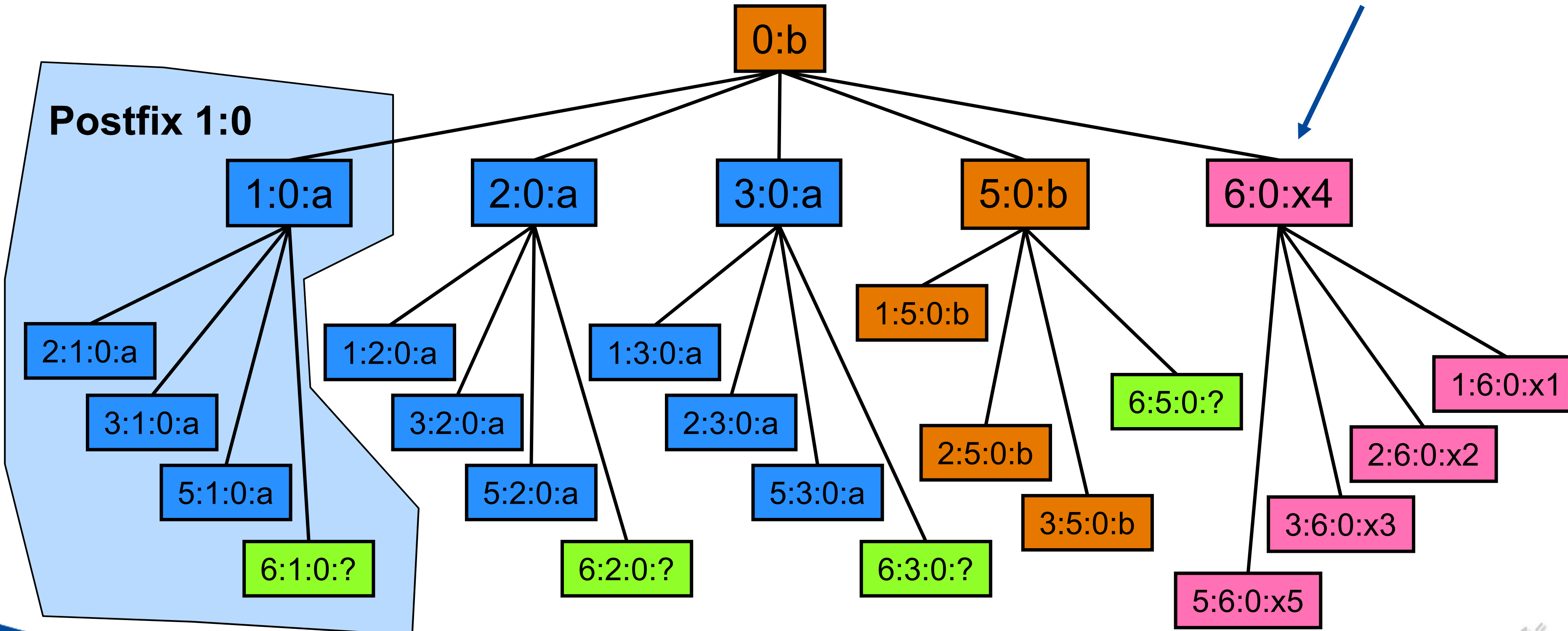
Message Tree for General 1 ($n = 7$, $m = 2$)

- Generals 0 (commander) and 6 faulty
- Commander sends *a* to generals 1 to 3 and *b* to generals 4 to 6
- In 2nd & 3rd round, general 6 sends arbitrary values to the others
- Depending on the values *x*1 to *x*5 of general 6, *all* correct generals agree on either value *a* or the default value



Message Tree for General 4 (n = 7, m = 2)

- Generals 0 (commander) and 6 faulty
- Commander sends *a* to generals 1 to 3 and *b* to generals 4 to 6
- In 2nd & 3rd round, general 6 sends arbitrary values to the others
- Depending on the values *x*1 to *x*5 of general 6, *all* correct generals agree on either *a* or the default value



Impossibility for Asynchronous Systems

> Precondition so far: synchronous system

> Fischer et al. proved 1985 that (deterministic) consensus is impossible to achieve in asynchronous systems

> This is the case even if there is only one faulty process

> Many other (theoretical) papers about this topic with differing assumptions

Exemplary Exam Questions

1. Describe the problem of the Byzantine generals and the algorithm to solve this problem!
2. How many rounds this algorithm needs?
3. What is the message complexity of the algorithm?
4. How do the tree majority function work?

Literature

1. **L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3):382--401, 1982.**
2. D. K. Pradhan: Fault-Tolerant Computer System Design, section 3.4 and chapter 8
3. N.A. Lynch: Distributed Algorithms, Chapter 6
4. M. Barborak, M. Malek, A. Dahbura: The Consensus Problem in Fault-Tolerant Computing, ACM Computing Survey Vol 25, Nr. 2, 1993
5. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. J. ACM 32, 2 (April 1985), 374-382.

Thank you for your kind attention!

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

`gero.muehl@uni-rostock.de`
`http://wwwava.informatik.uni-rostock.de`