Technische Universität Berlin

Distributed Algorithms 2015/16
Consensus and Related Problems

Reinhardt Karnapke | Communication and Operating Systems Group

Overview

Introduction (last lecture)

Masking fault tolerance (this lecture)

- Consensus and related problems

Non-masking fault tolerance (next lecture)

- Self-Stabilization

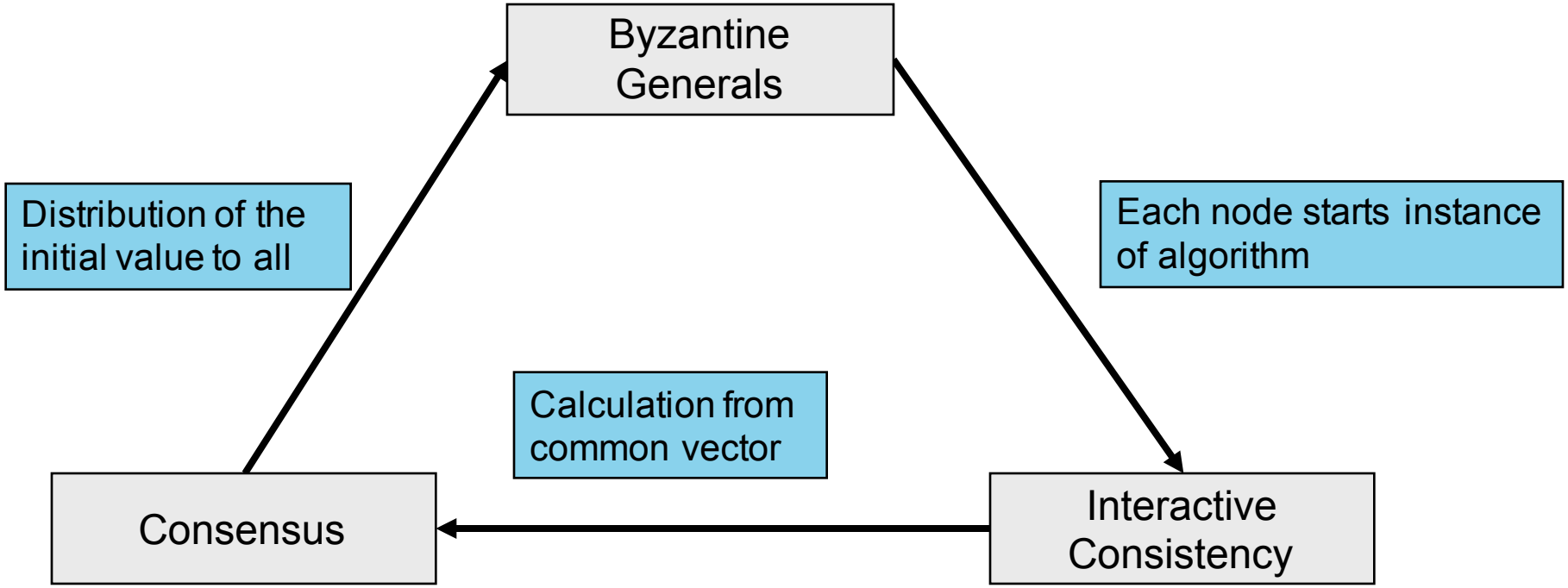
Consensus and Related Problems

- **Byzantine generals**
  - ***One*** value is proposed by **one** distinguished process
  - All correct processes agree on the same value (IC1)
  - If the proposing process is fault-free, the proposed **value** is the value agreed on (IC2)
- **Interactive consistency**
  - ***One*** value is given by ***each*** process
  - All correct processes agree on the same ***value vector***
  - The proposed values of fault-free processes are rendered correctly in the value vector
- **Consensus**
  - ***One*** value is given by ***each*** process
  - All correct processes agree on the same **value**
  - If all correct processes propose the same value, they all agree on that value

Agreement vs. Consensus vs. Consistency

The three problems can be transferred into each other!

See also Coulouris, pages 452-455



Byzantine Generals → Interactive Consistency

- Solution for Byzantine generals exists, we are searching for a solution for interactive consistency
  - Each node starts an instance of the algorithm for the Byzantine generals with its value
  - Each fault-free node takes the value that it calculates as solution for the byzantine generals with process P_i as leader as i-th component of its result vector
  - The result vectors of the nodes are a correct solution of the problem of interactive consistency

Interactive Consistency → Consensus

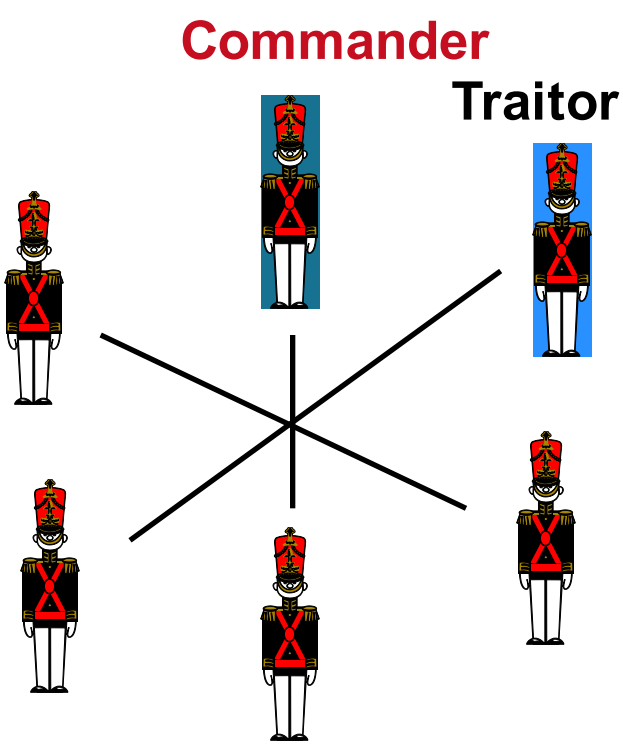
- Solution for interactive consistency exists, we are searching for consensus
  - The nodes determine a solution for the interactive consistency
  - Each node gives the value it would also propose for the consensus
  - Then, each node determines a value from its result vector by means of the majority function; if no majority exists, it takes the default-value
  - The solution built by the values of the nodes is a correct solution of the consensus problem

Consensus → Byzantine Generals

- Solution for consensus exists; we are searching for one for the byzantine generals
  - The initial value is distributed by the leader to all lieutenants and to itself
  - Then, a solution of the consensus problem is determined among all generals
  - Each lieutenant has a consensus value afterwards
  - The consensus values of all lieutenants form a correct solution of the problem of the byzantine generals

Byzantine Generals

- Lamport, Shostak, and Pease, 1982
- $n > 3$ generals, m of them are traitors (cause byzantine faults)
- One of the generals is the commander (commanding general) and proposes a value $v \in \{0, 1\}$
- The other generals (lieutenant generals) shall execute the order of the commander
- At least 1 lieutenant is loyal (fault free)
- Even the commander can be a traitor (faulty)
- Question: Attack together ($v = 1$) or wait ($v = 0$)?



Byzantine Generals – Assumptions

- Synchronous system model
- Each process is directly connected to every other process → complete meshing
- Messages
  - Do not get lost
  - Are not duplicated
  - Arrive as sent
  - Cannot be signed forgery-proof
  - Allow the determination of the sender‘s identity
    - → oral messages

Byzantine Generals – Preconditions

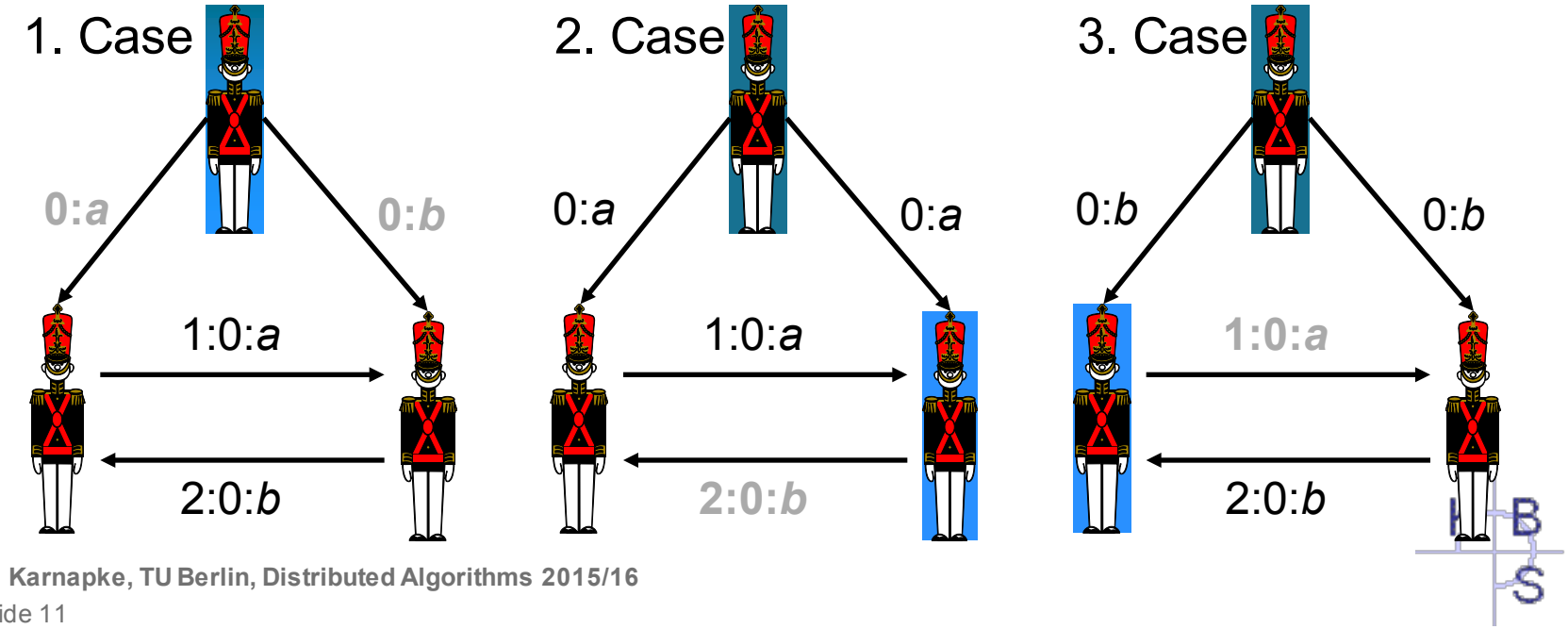
- For m traitors and n generals no algorithm exists that solves the problem of byzantine generals for $n \leq 3m$
- Simplest special case:
  There is no solution for $n = 3$ and $m = 1$
  - Intuitive argument:
    - How should a loyal general communicating with a loyal general and a traitor figure out who is who if they blame each other?
    - For the next slide, assume that the commander sends the command to the generals. Afterwards, they send the received message to each other.

BG – Intuitive Argument for $n = 3$ and $m = 1$

1. Case: Commander is faulty → Good generals receive {*a*, *b*}
2. Case: Right general is faulty → Left general receives {*a*, *b*}
3. Case: Left general is faulty → Right general receives {*a*, *b*}

Commander: 0
Lieutenants: 1, 2

It applies $a \neq b$!



BG – Intuitive Argument for $n = 3$ and $m = 1$

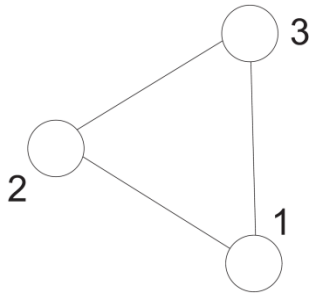
- The left general cannot distinguish case 1 from case 2. Thus, it has to choose the value *a* given by the commander to fulfill IC2
- The right general cannot distinguish case 1 from case 3. Thus, it has to choose value *b* given by the commander to fulfill IC2
- But that means that both generals choose different values in the 1. case. Thus, IC1 is violated

- Similar arguments 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.

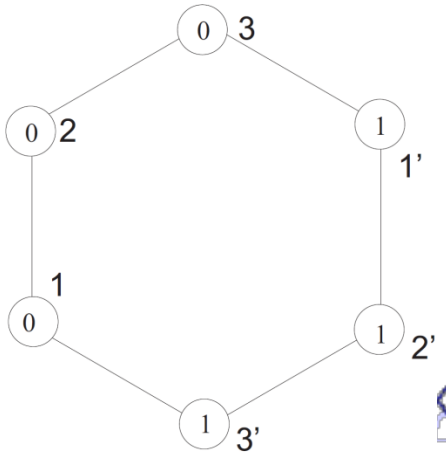
BG – Proof of Impossibility for n=3, m=1

Proof:

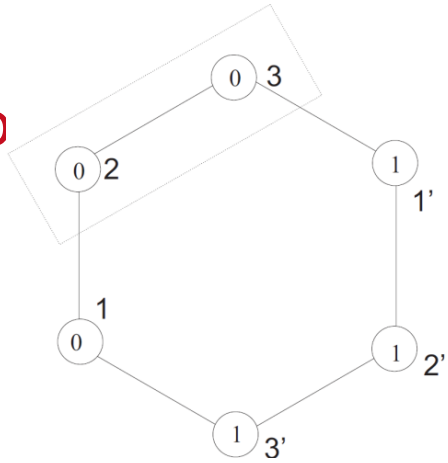
Indirect approach: Assuming, there is an algorithm solving the problem in a system S:

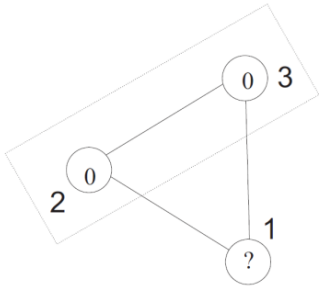


Construct S' based on two instances of S with inverse starting values for the second instance



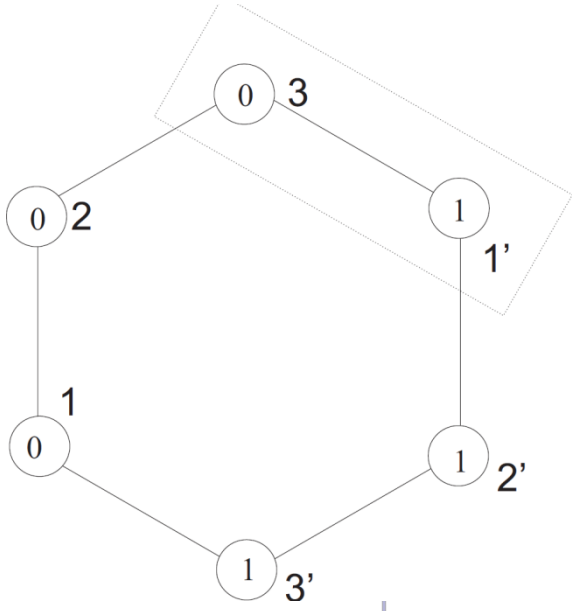
BG – Proof of Impossibility fo





- For two nodes, no difference is detectable between S' and S with a faulty node

- Successive application of this argument to pairs of generals leads to a contradiction



> For $n = 3$ and $m = 1$, no solution is possible!

- This result can be generalized:

  For $n \leq 3m$ no solution is possible

BG – Algorithm for $m = 1$

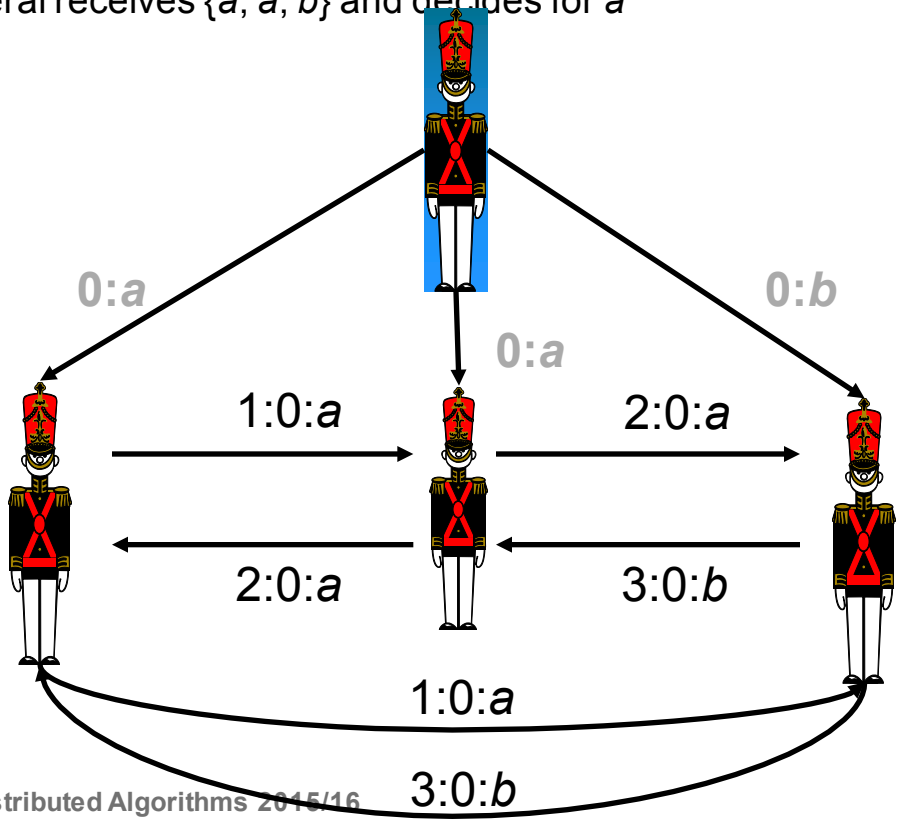
1. The commander reveals its value to the others (0 or 1)
2. Then, each general tells each other general the value it received from the commander
3. Each general makes a majority decision according to the received values

Attention: A faulty commander or general can send what it wants!

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

Most simple case with $n \geq 3m + 1$

1. Case: Commander is faulty
   - Each general receives {*a*, *a*, *b*} and decides for *a*

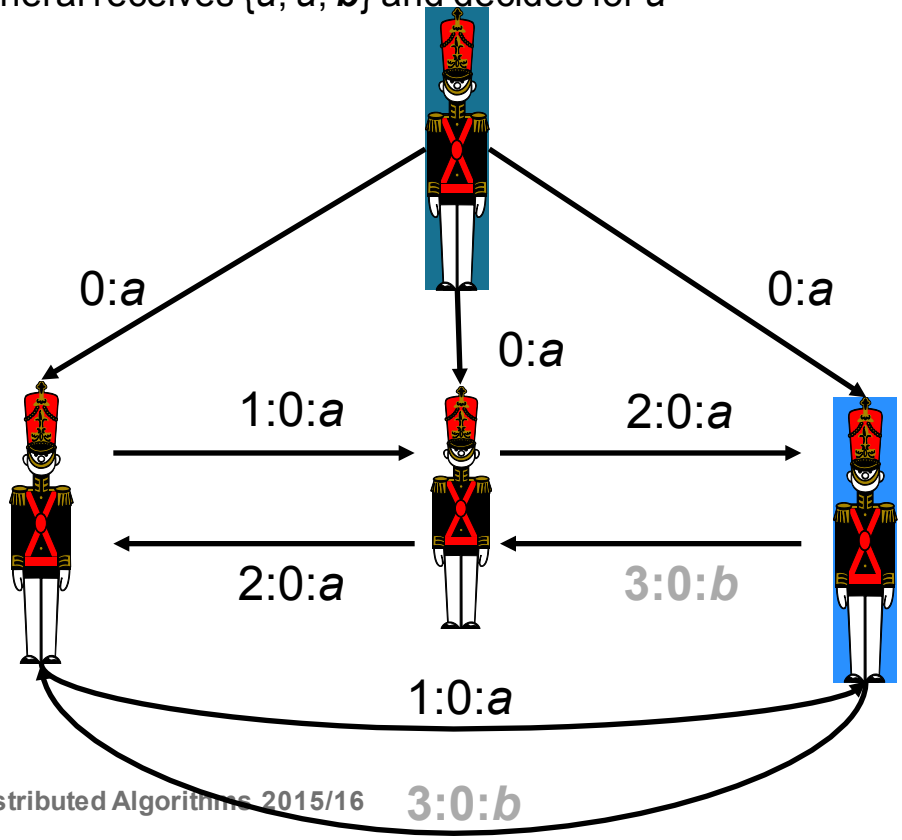


Commander: 0
Lieutenants: 1, 2, 3

It applies $a \neq b$!

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

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



It applies $a \neq b$!

Byzantine Generals

- Algorithm can be generalized for larger m through a recursive execution of the algorithm
  - The algorithm needs $m + 1$ rounds (that is minimal!)
  - ⇒ Unit time complexity $m + 1$
- With m faulty processes, agreement is possible if there are at least $2m + 1$ correct processes

⇒ It must apply: $n \geq 3m + 1$ (The barrier is hard!)

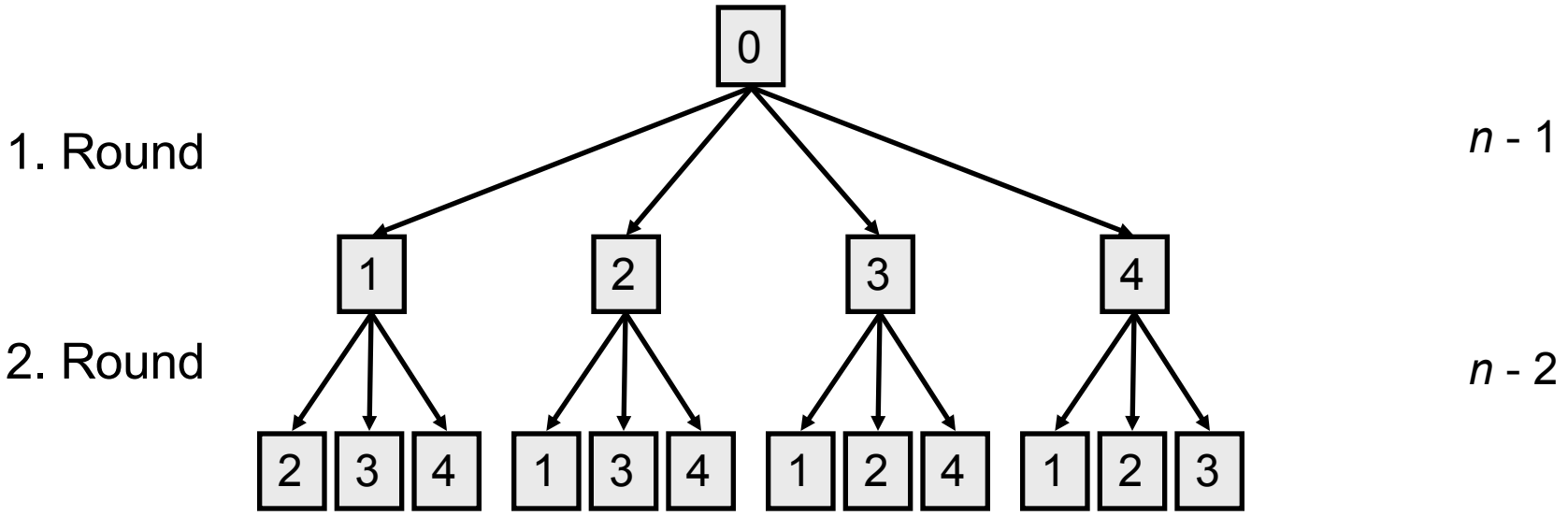
⇒ More than 2 / 3 of all processes must work correctly

Recursive algorithm *OM* for Oral Messages

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

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

{Message (m, G, t) is received by lieutenant L}:
    IF (m, G, t) is pending THEN
        M_L := M_L ∪ t;
        IF m != 0 THEN
            OM(m - 1, L, G \ {L}, t);
        FI
    FI

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

Commander:	General 0
Lieutenants:	Generals 1 – (n-1)

Recursion Tree for $n = 5$ and $m = 1$



Byzantine Generals – Message Complexity

An 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

Byzantine Generals – Message Complexity

1. round: 1 instance with $n - 1$ messages each
2. round: $n - 1$ instances with $n - 2$ messages each
3. 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

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

Derivation of the message complexity

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

Byzantine Generals – 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

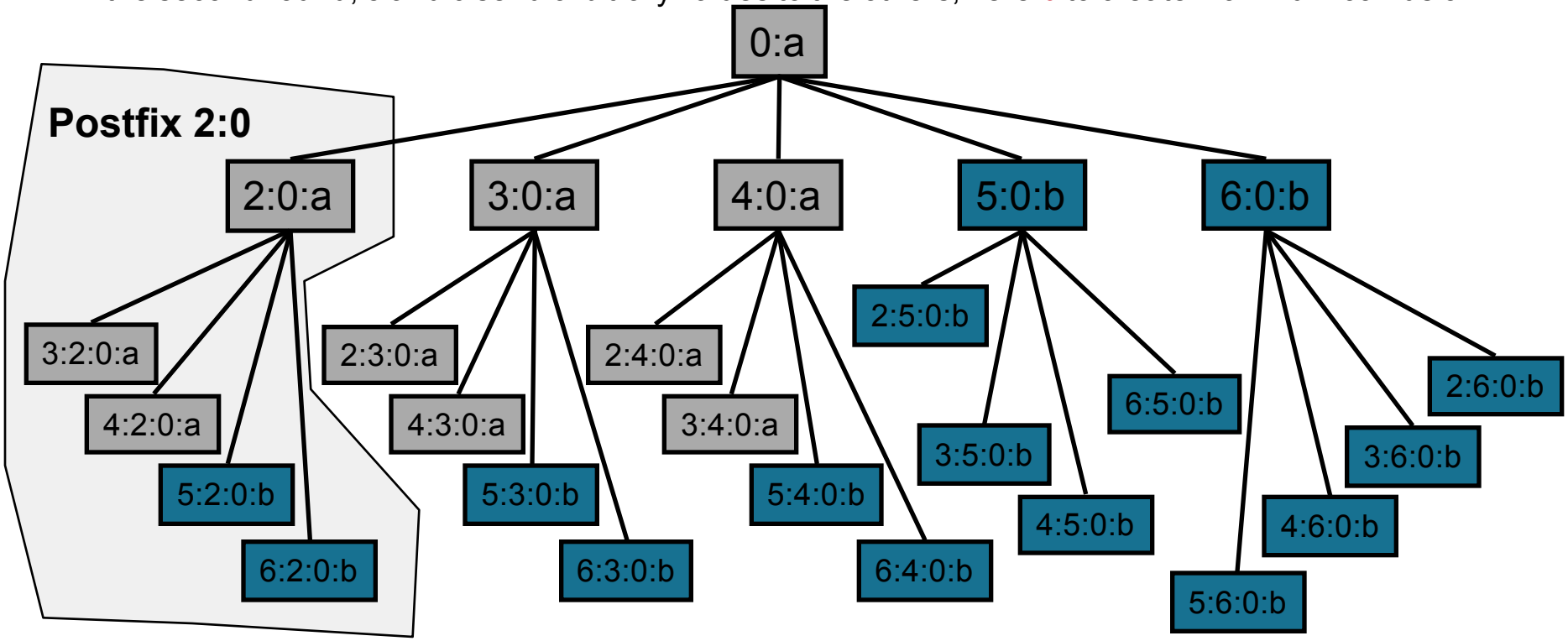
- Each general forms a message tree from the messages it has received
- Therefore, it arranges the received messages in accordance to the postfix of their message path

- Repeat majority formation, until a single value is derived, after following method
  - The 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

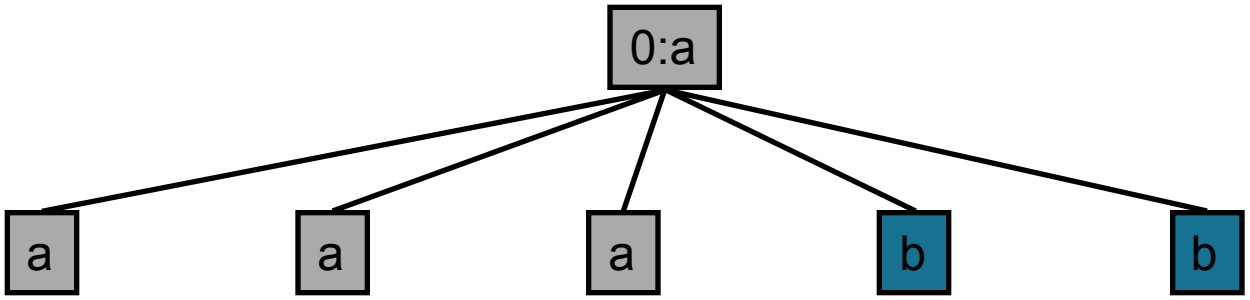
Message Tree for General 1 and $n = 7$, $m = 2$

General 5 and 6 faulty, commander sends ***a*** to the nodes 1 to 6

In the second round, 5 and 6 send arbitrary values to the others; here ***b*** to create maximum confusion



Message Tree for General 1 and *n* = 7, m = 2



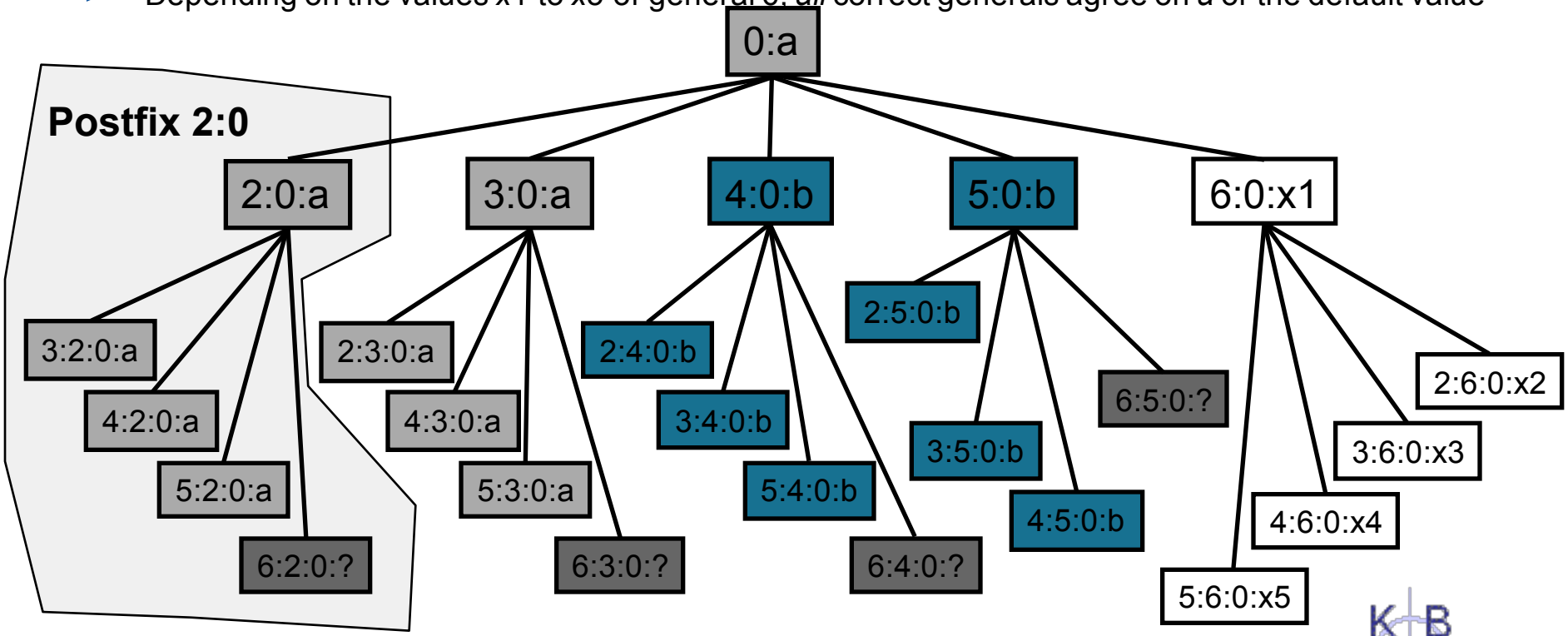
This figure shows the message tree of the previous slide after the first majority formation.

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

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

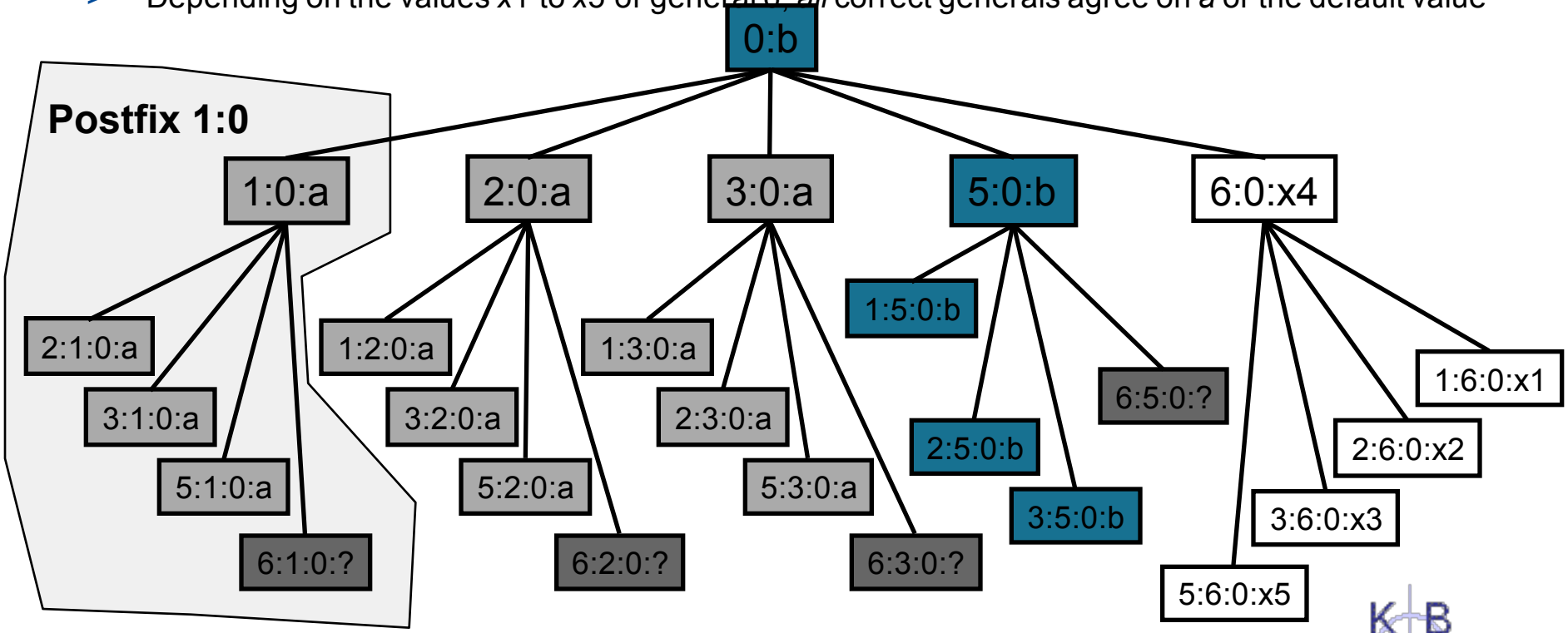
Message Tree for General 1 and $n = 7$, $m = 2$

- Generals 0 (commander) and 6 faulty
- Commander sends ***a*** to generals 1 to 3 and ***b*** to 4 to 6
- General 6 sends in **2. + 3.** round arbitrary values to the others (x1,x2,x3,x4,x5)
- Depending on the values *x*1 to *x*5 of general 6, *all* correct generals agree on *a* or the default value



Message Tree for General 4 and $n = 7$, $m = 2$

- Generals 0 (commander) and 6 faulty
- Commander sends ***a*** to generals 1 to 3 and ***b*** to 4 to 6
- General 6 sends in **2. + 3.** round arbitrary values to the others (x1,x2,x3,x4,x5)
- Depending on the values *x*1 to *x*5 of general 6, *all* correct generals agree on *a* or the default value



Byzantine Generals – Signed Messages

- If messages can be signed forgery-proof (signed messages), the agreement can be achieved with arbitrarily many faulty processes

- Precondition so far: Synchronous system
- "In an asynchronous system no agreement is possible."
- Fisher et al., 1985
- Many other (theoretical) papers with this topic with differing assumptions

Byzantine Generals – Conditions for Solvability

- In literature, a number of impossibility proofs for byzantine generals or similar problems was given

- Dolev et. Al. identified five system characteristics that are important
  - Execution at nodes: Synchronous vs. asynchronous
  - Communication: Synchronous vs. asynchronous
  - Order of messages: ordered (FIFO) vs. not ordered
  - Communication scheme: broadcast vs. point to point
  - Atomicity of send/receive: atomic vs. not atomic

Byzantine Generals – Conditions for Solvability

- Exactly four cases were identified for existence of byzantine agreement with $f > 1$ faulty nodes
  - Synchronous nodes and synchronous communication
  - Synchronous nodes and ordered messages
  - Broadcast and ordered messages
  - Synchronous communication, broadcast, atomic send/receive
- For $f = 1$ another case exists:
  - Asynchronous nodes, synchronous communication, point-to-point communication and atomic send/receive

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