Distributed OS

Hermann Härtig

Modelling

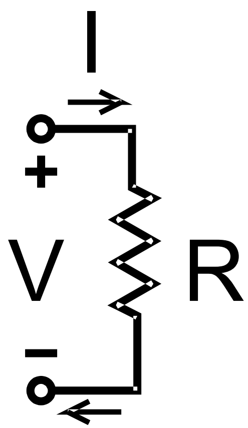
Aspects of Distributed Systems

TECHNISCHE UNIVERSITÄT DRESDEN

Models

- abstract from details
- concentrate on functionality, properties, ... that are considered important for a specific system/application
- use model to analyse, prove, predict, ... system properties
- models in engineering disciplines very common, not (yet) so in CS
- we'll see many models in lecture: “Real-Time Systems”
- Objective of lecture:
  understand the need for careful understanding of models
- 1rst lecture: Amdahl's Law, Today: 3 areas

Model examples



$$I = V / R$$

UML

Models for 3 areas

- Limits of Reliability of systems made of unreliable components
- Consensus
- Open source and security → separate slides

Fault Tolerance

- Techniques how to build reliable systems from less reliable components

- Fault(Error, Failure, ...):
  synonymously used for “something goes wrong”
  (more precise definitions and types of faults in SE)

Properties

Reliability:

- $R(t)$: probability for a system to survive time t

Availability:

- A: fraction of time a system works

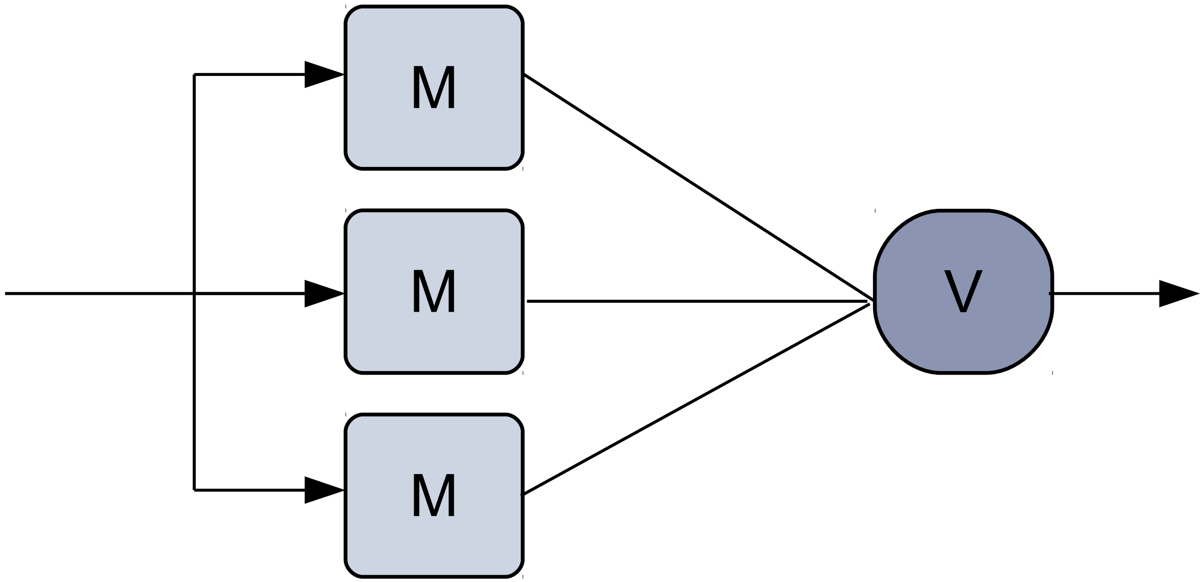
Fault Tolerance: key ingredients

- Fault detection and confinement
- Recovery
- Repair

- Redundancy
  - Information
  - time
  - structural
  - functional

Examples: RAID, Triple Modular Redundancy

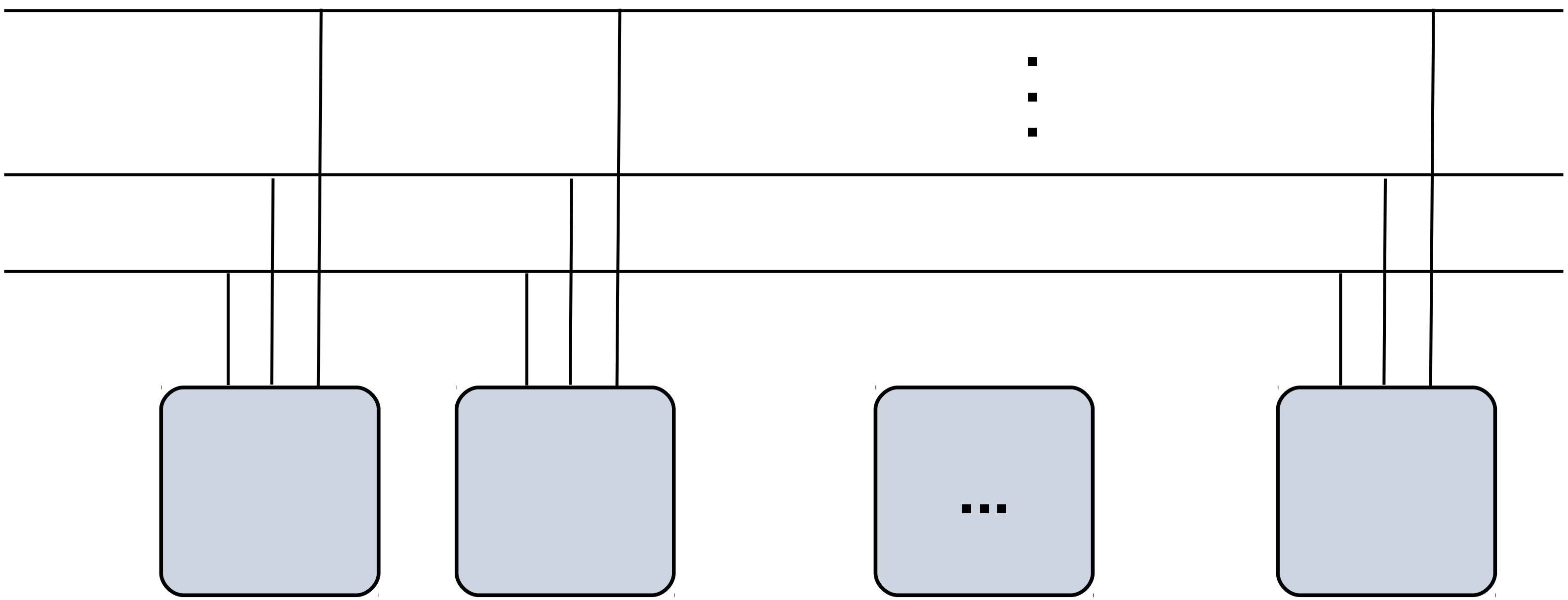
John v. Neumann
Voter: *single point of failure*



Can we do better
→ distributed solutions?

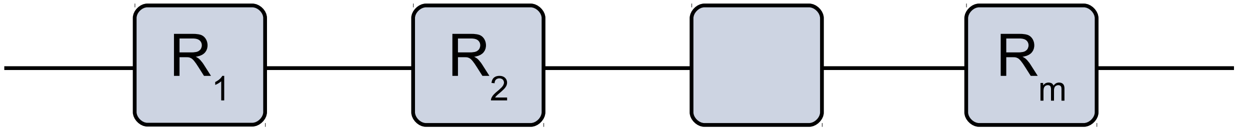
Limits(mathematical) of Reliability, Variant 1

Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)



Reliability Models

Serial Systems

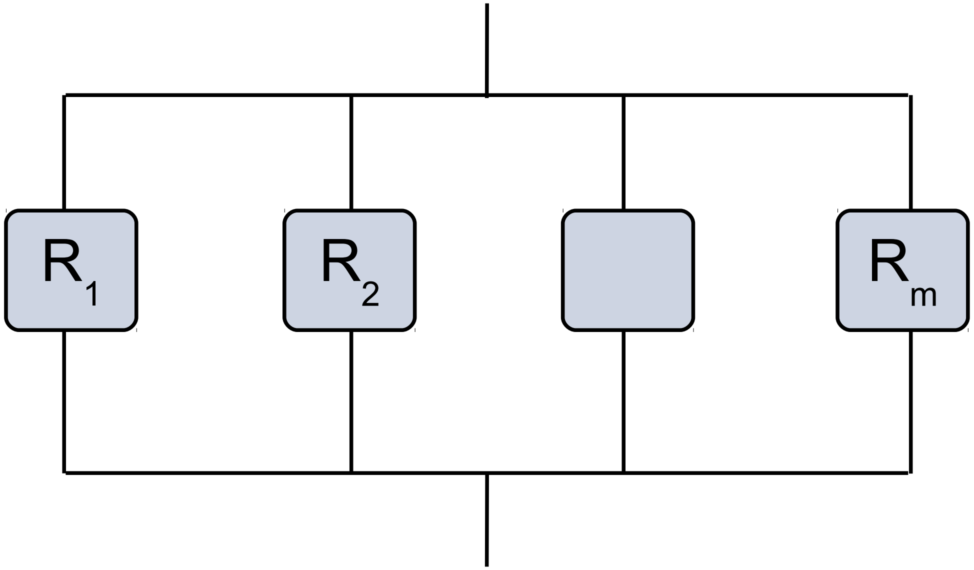


$$R_{whole} = \prod_{j=1}^{n} R_j$$

- Each component must <u>work</u> for the whole system to <u>work.</u>

Reliability Models

Parallel Systems

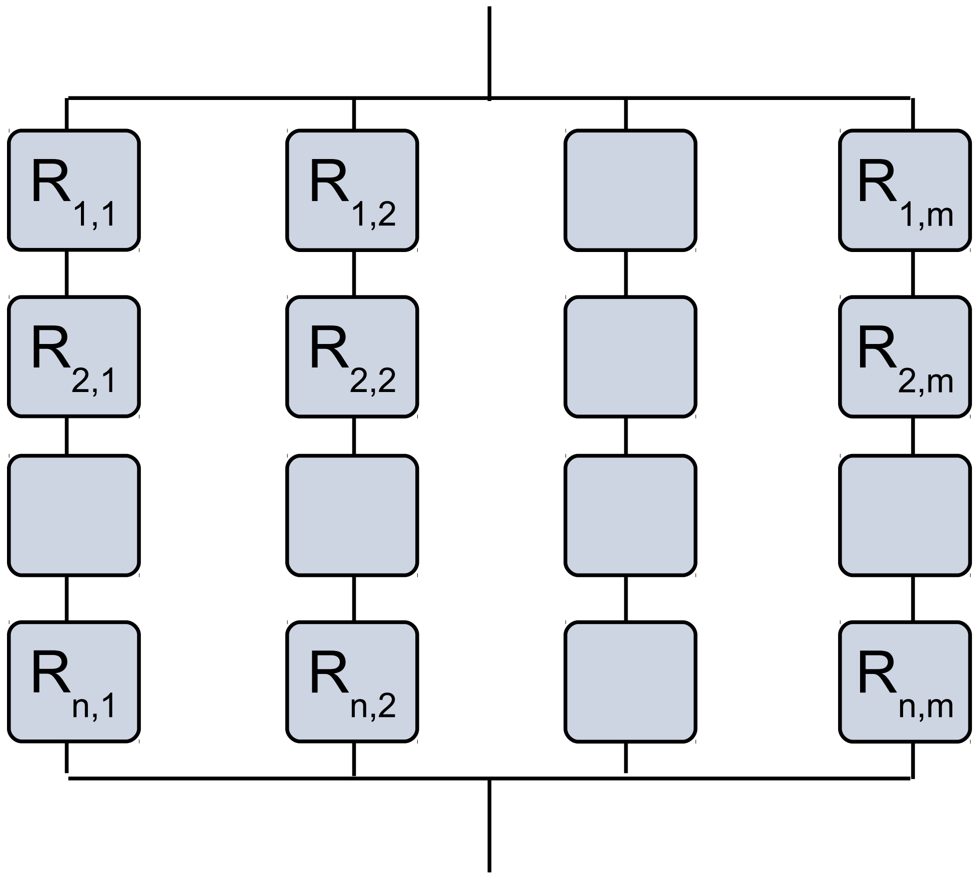


$$R_{whole} = 1 - \prod_{i=1}^{m} (1 - R_i)$$

- One component must <u>work</u> for the whole system to <u>work</u>.
- Each component must <u>fail</u> for the whole system to <u>fail</u>.

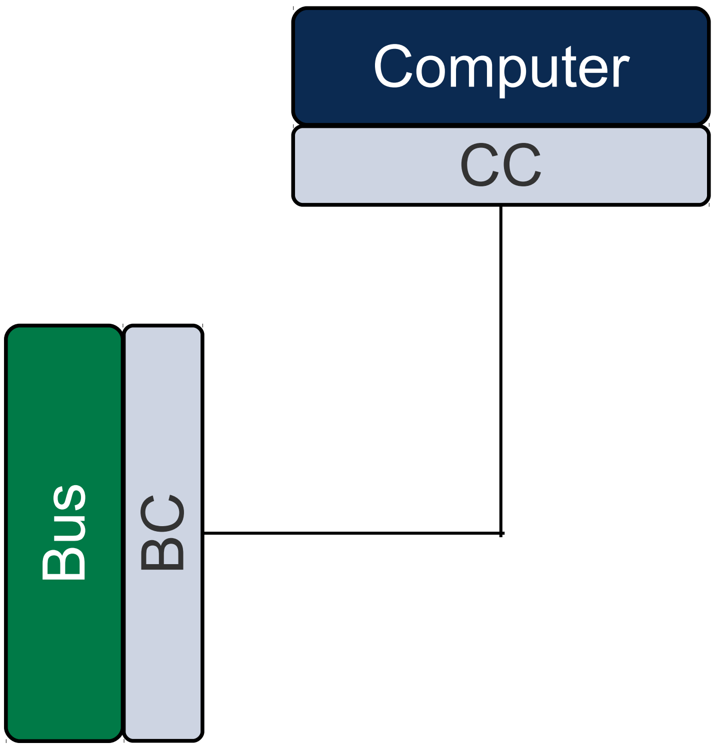
Reliability Models

Serial-Parallel Systems



$$R_{whole} = 1 - \prod_{j=1}^{m} \left(1 - \prod_{i=1}^{n} R_{i,j} \right)$$

Our Example



Fault Model

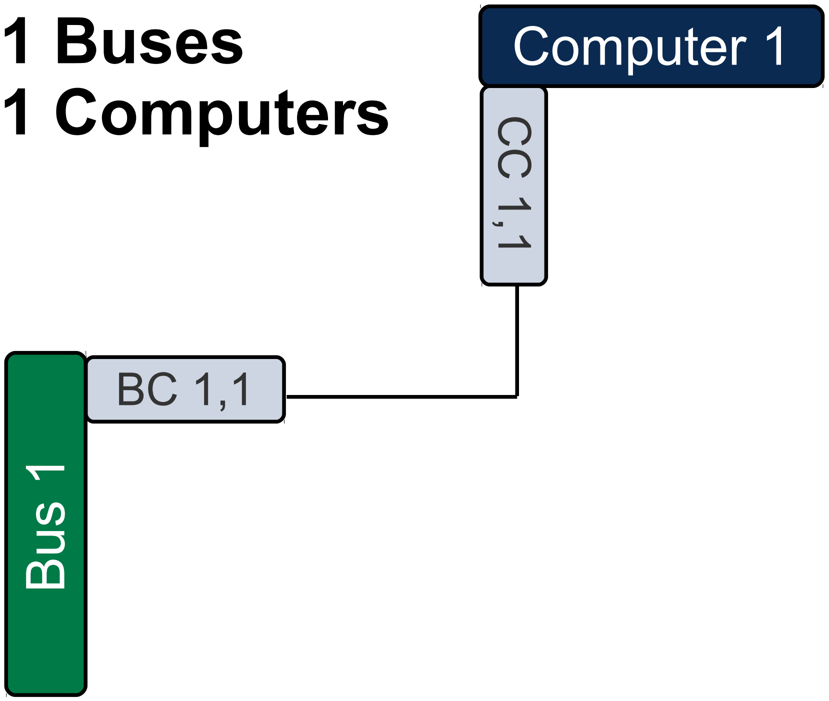
„Computer-Bus-Connector" can fail such that Computer and/or Bus also fail

therefore we model: conceptual separation of connector into

- CC: Computer-Connector, whose fault also breaks the Computer
- BC: Bus-Connector, ...

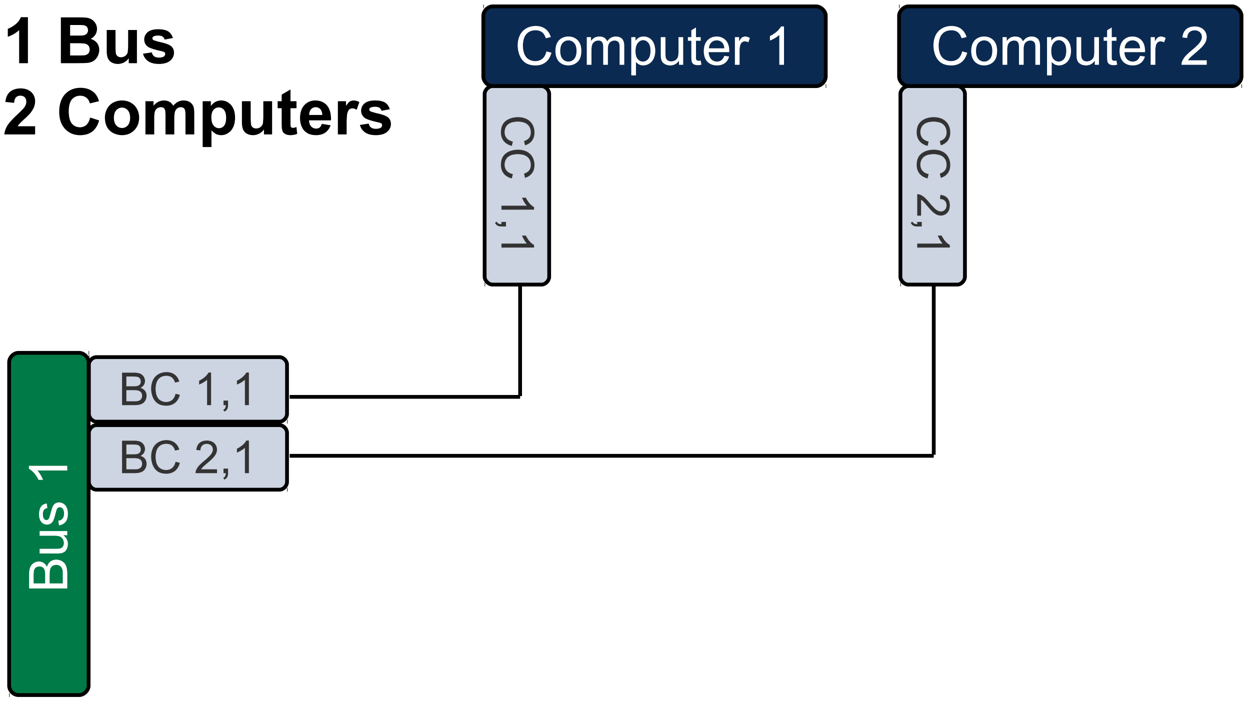
Our Example

1 Buses
1 Computers



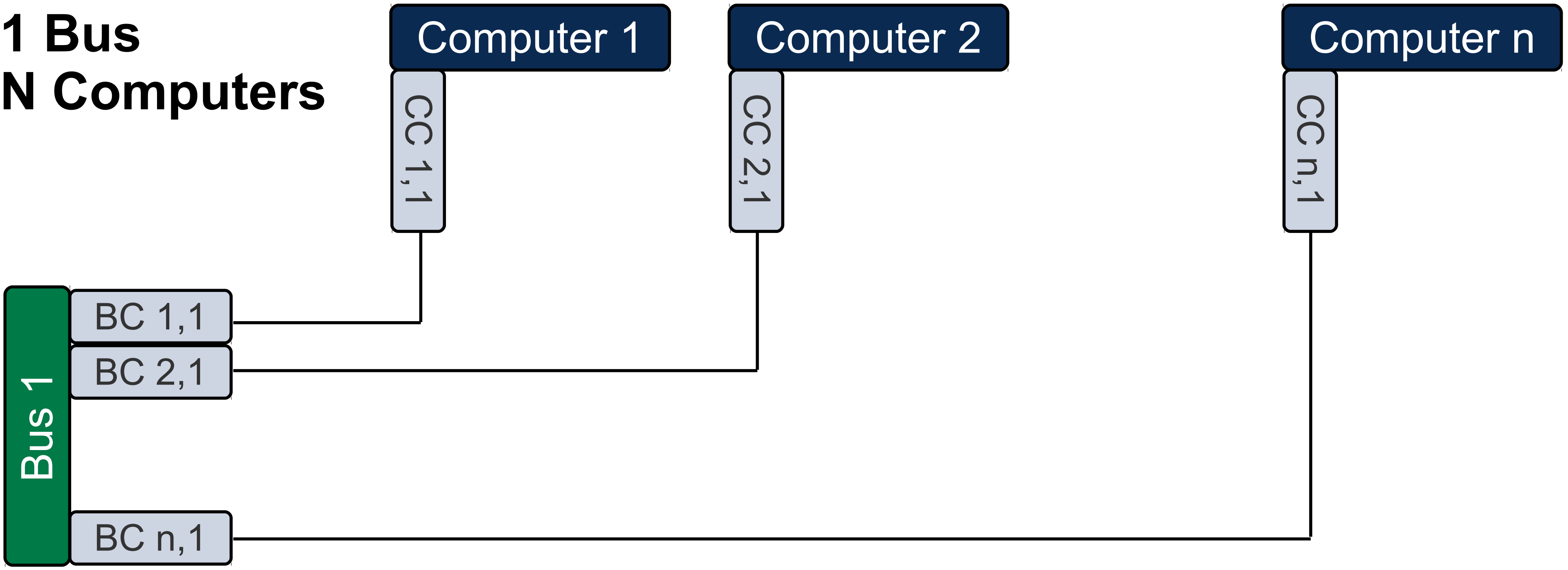
Our Example

1 Bus
2 Computers



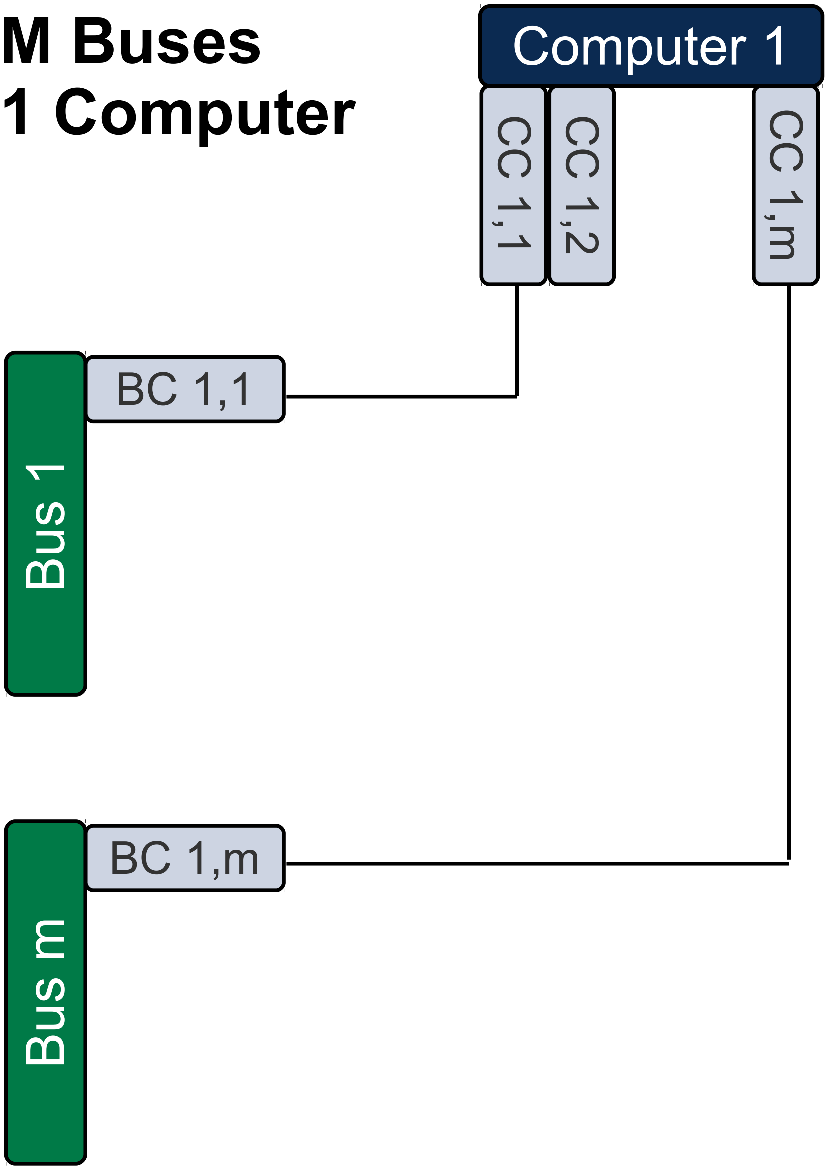
Our Example

1 Bus
N Computers



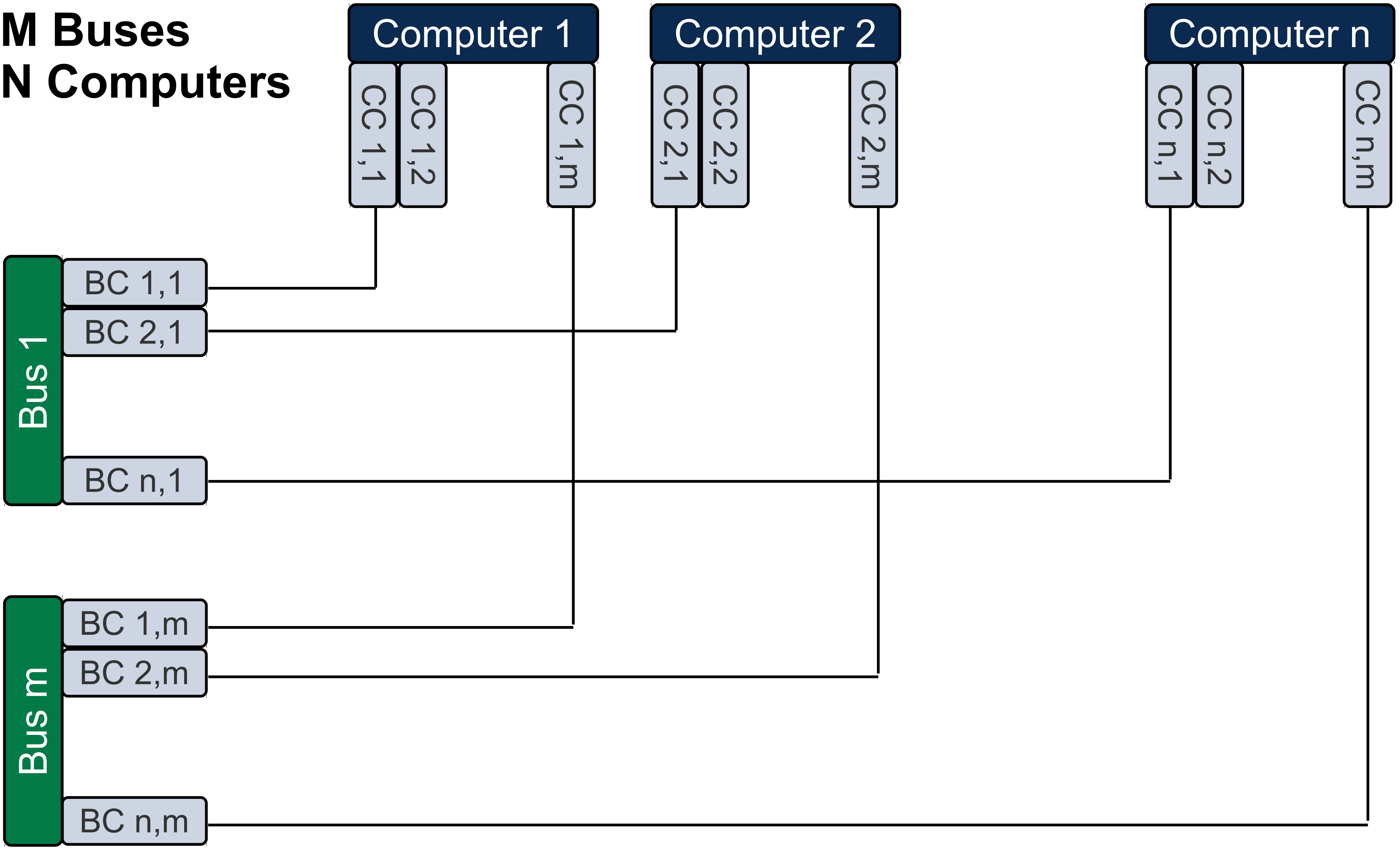
Our Example

M Buses
1 Computer

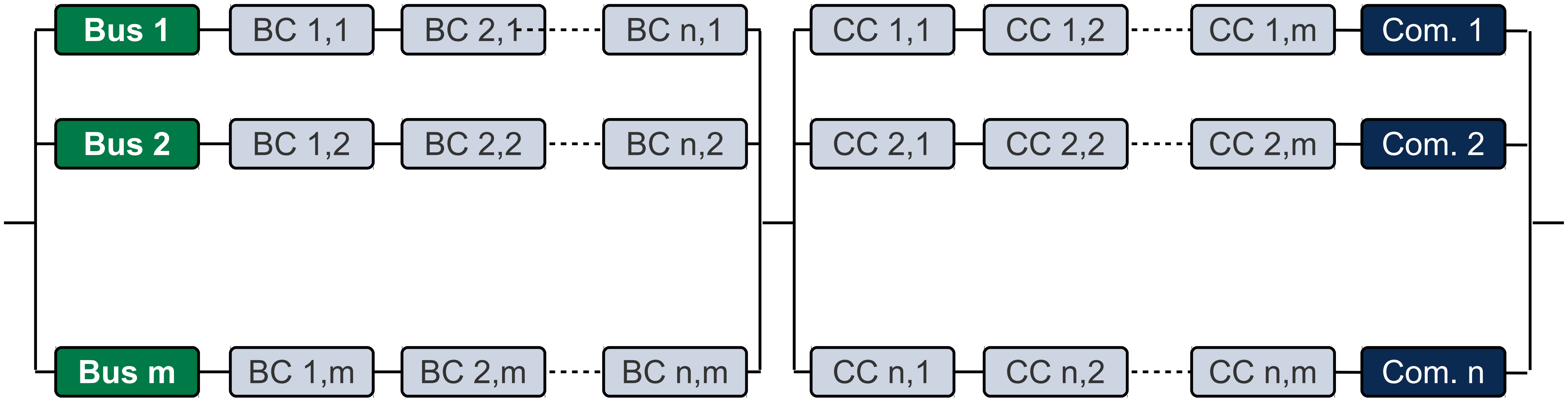


Our Example

M Buses
N Computers



Model for m,n



$$R_{whole}(n,m) = \left(1-\left(1-R_{Bus}\cdot R_{BC}^{n}\right)^{m}\right)\cdot\left(1-\left(1-R_{Computer}\cdot R_{CC}^{m}\right)^{n}\right)$$

$$then:\ R_{CC}, R_{BC} < 1:\ \lim_{n,m\to\infty} R(n,m) = ??$$

Limits(mathematical) of Reliability, Variant 2

- System built of Synapses (John von Neumann, 1956)
- <u>Computation and Fault Model</u>:
  - Synapses deliver „0“ or „1”
  - Synapses deliver with R > 0,5:
    - with probability R correct result
    - with (1-R) wrong result
- Then we can build systems that deliver correct result for any (arbitrary high) probability R

Report here: cum grano salis!!

Two Army Problem (Coordinated Attack)

- p,q processes
  - communicate using messages
  - messages can get lost
  - no upper time for message delivery known
  - do not crash, do not cheat
- p,q to agree on action (e.g. attack, retreat, ...)
- how many messages needed ?
- first mentioned: Jim Gray 1978

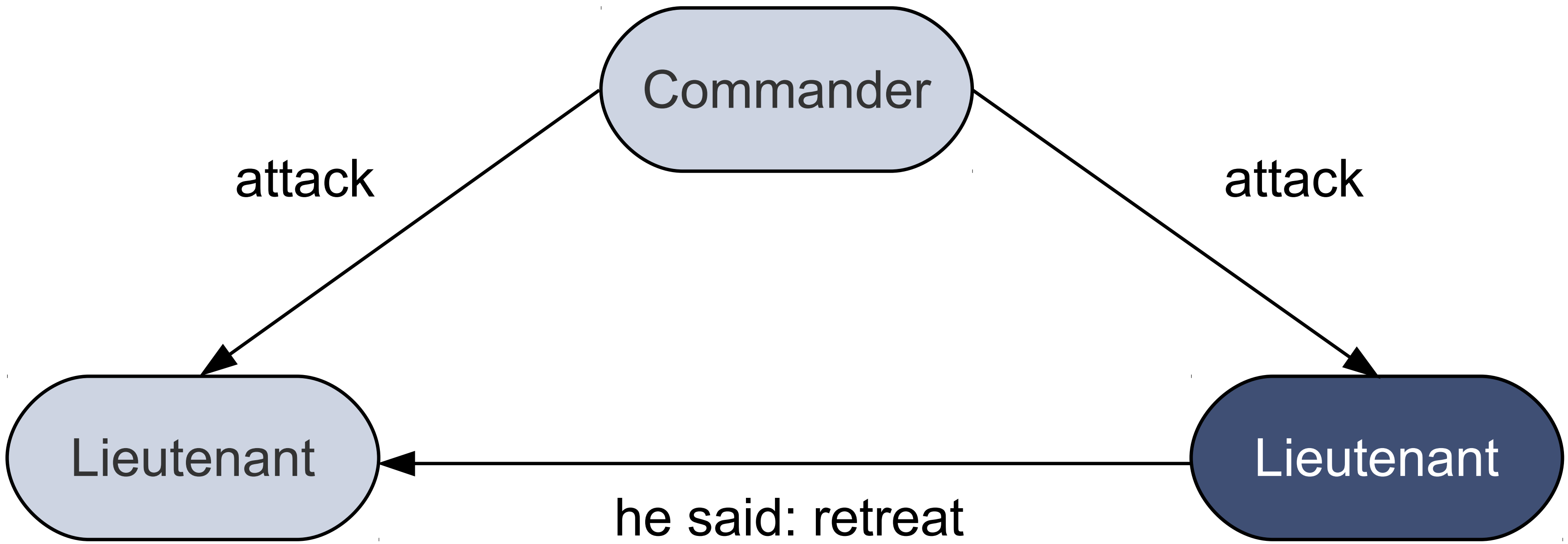
Two Army Problem (Coordinated Attack)

- Result: there is no protocol with finite messages
- Prove:
  - by contradiction
  - assume there are finites protocols $(m_{p \dashrightarrow q}, m_{q \dashrightarrow p})^*$
  - choose the shortest protocol MP,
  - last message MX: $\underline{m}_{p \dashrightarrow q}$ or $\underline{m}_{q \dashrightarrow p}$
  - MX can get lost
  - => must not be relied upon => can be omitted
  - => MP not the shortest protocol.
  - => no finite protocol

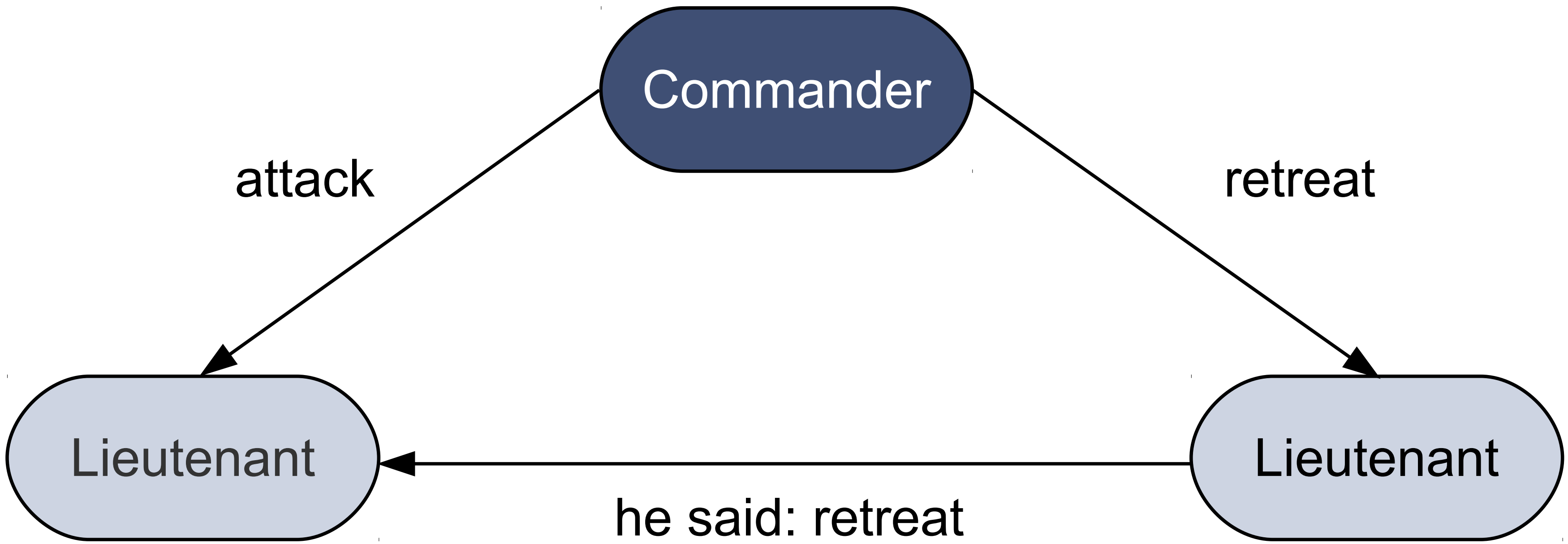
Byzantine Agreement

- n processes, f traitors, n-f loyals
  - communicate by reliable and timely messages
  - (synchronous messages)
  - traitors lye, also cheat on forwarding messages
  - try to confuse loyals
- Goal:
  - loyals try to agree on action (attack, retreat)
  - more specific:
    - one process is commander
    - if commander is loyal and gives an order, loyals follow the order otherwise loyals agree on arbitrary action

3 Processes: 1 traitor, 2 loyals

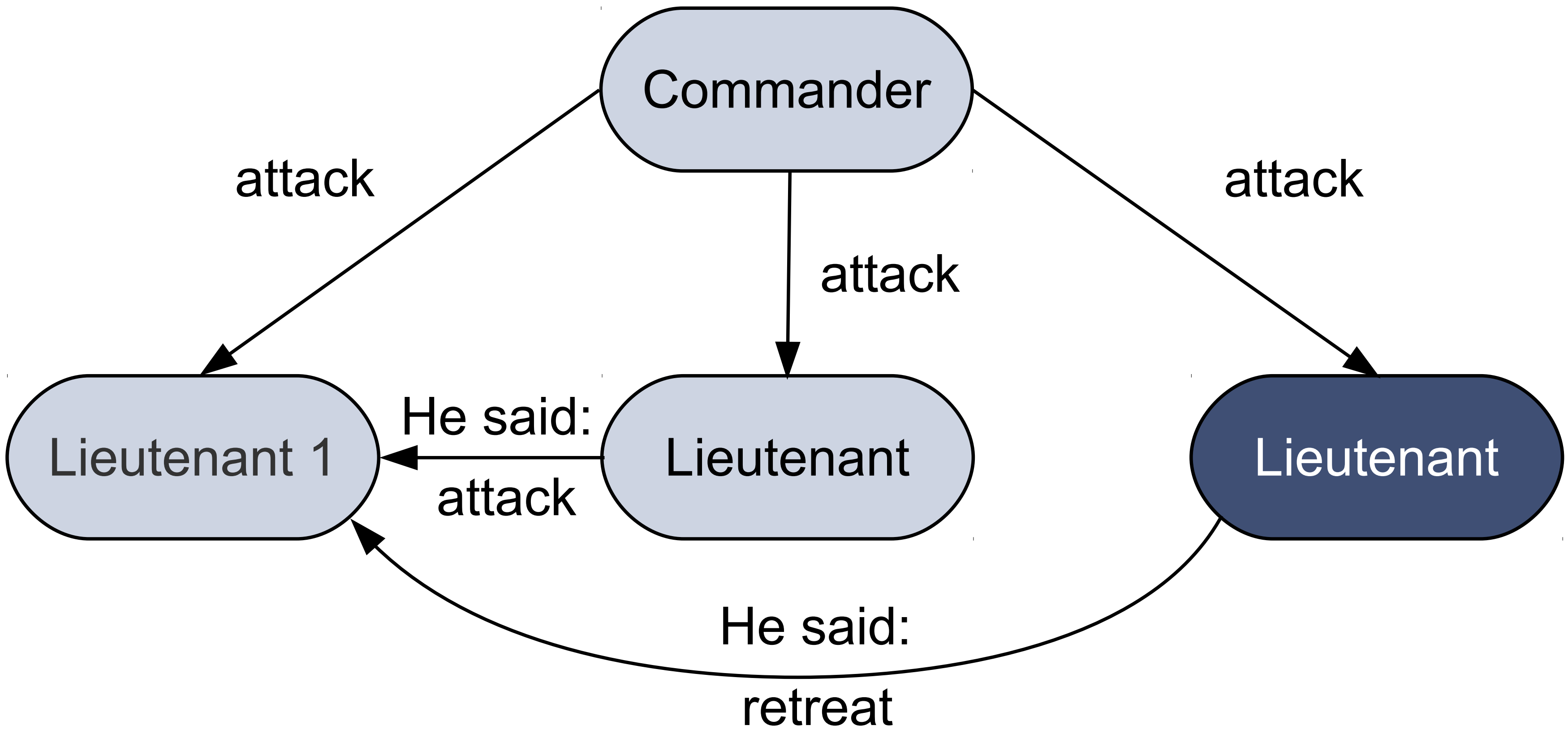


3 Processes: 1 traitor, 2 loyals



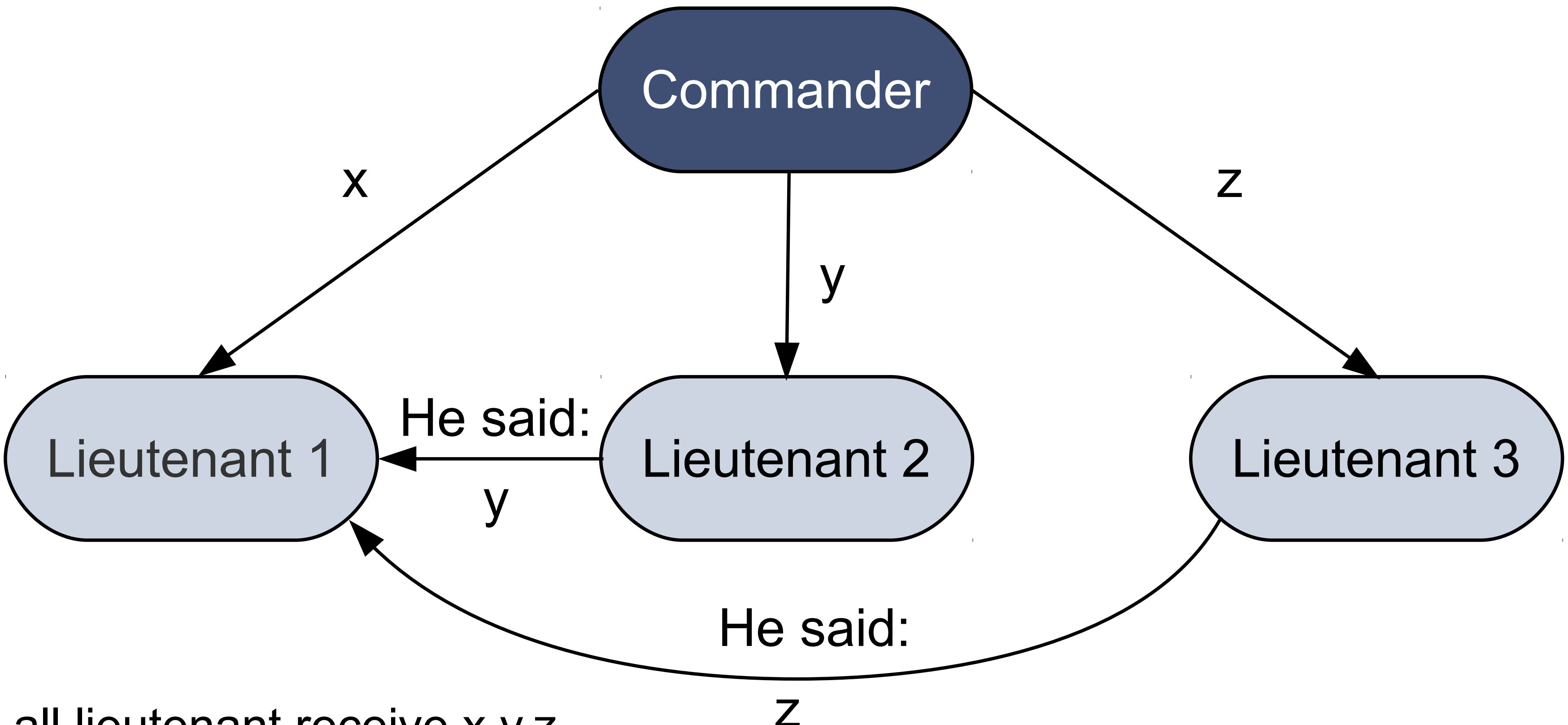
- 3 processes not sufficient to tolerate 1 traitor

4 Processes



Commander
attack
attack
attack
Lieutenant 1
He said:
attack
Lieutenant
Lieutenant
He said:
retreat

4 Processes



- all lieutenant receive x,y,z
- can decide
- General result:3 f + 1 processes needed to tolerate f traitors

To take away

- modeling is very powerful
- extreme care needed to do it correctly