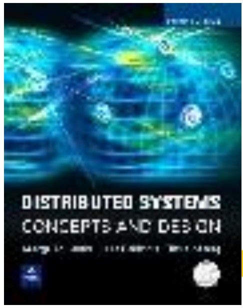
Prof Philippas Tsigas
Distributed Computing and Systems Research Group

DISTRIBUTED SYSTEMS II

FAULT-TOLERANT AGREEMENT

Teaching material based on Distributed Systems: Concepts and Design, Edition 3, Addison-Wesley 2001.



Distributed Systems Course

Coordination and Agreement

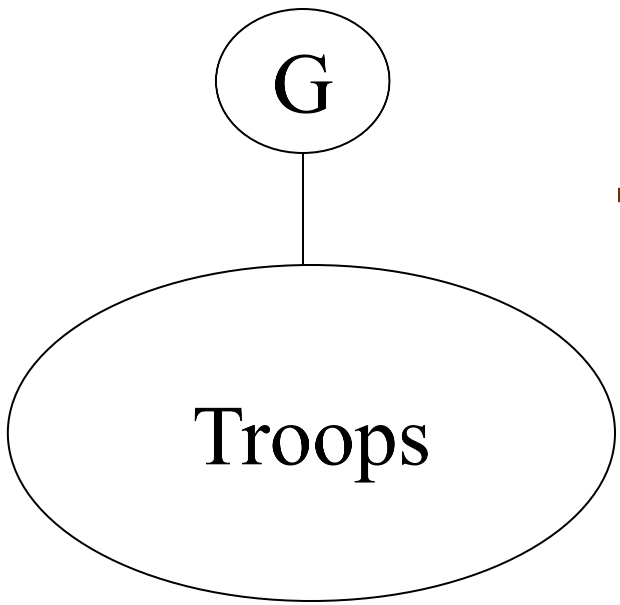
11.5 Consensus and Related problems

Viewing: These slides must be viewed in slide show mode.

Agreement

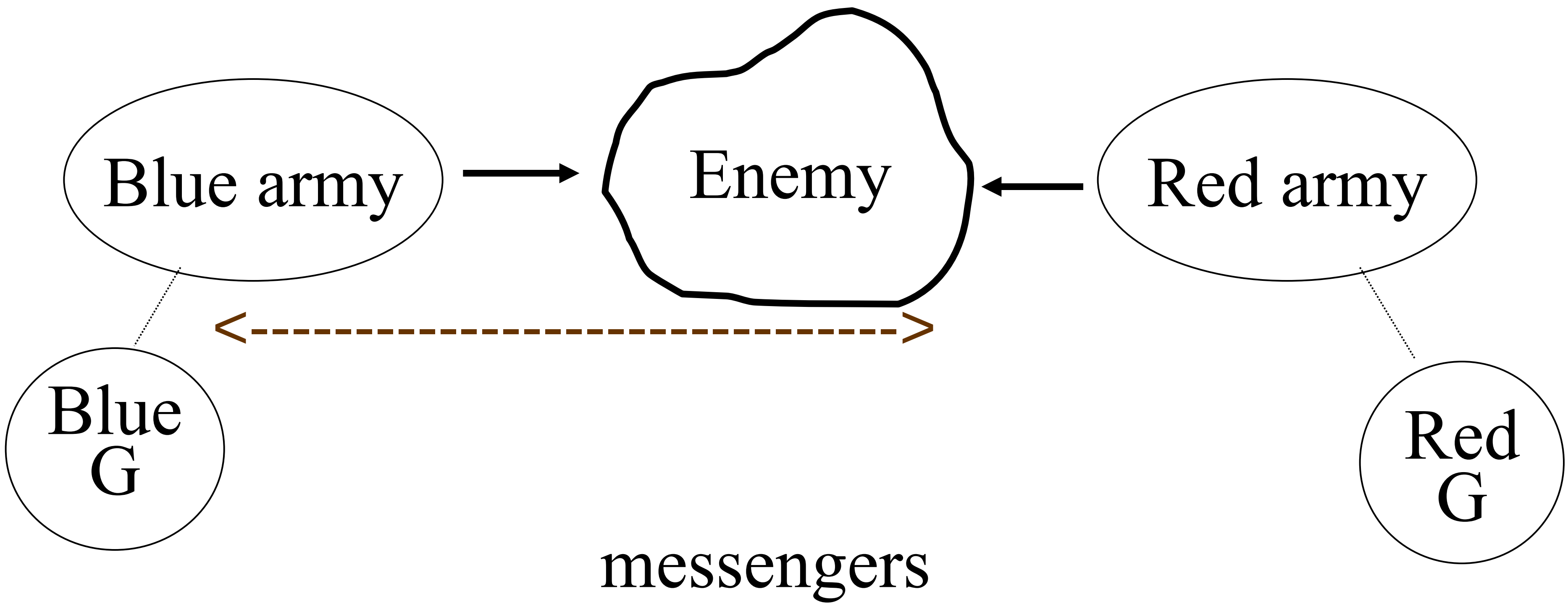
- All processes start with **an initial value from some set V**
- •Every process has to decide on a value in **V such that:**
  - **Agreement:** no two non-faulty processes decide on different values
  - **Validity:** if all processes start with the same value v, then no non-faulty process decides on a value different from v
  - **Termination:** all non-faulty processes decide within finite time

The one general problem (Trivial!)



- Battlefield

The two general problem:



Rules:

- Blue and red army must attack at same time
- Blue and red generals synchronize through messengers
- ***Messengers (messages) can be lost***

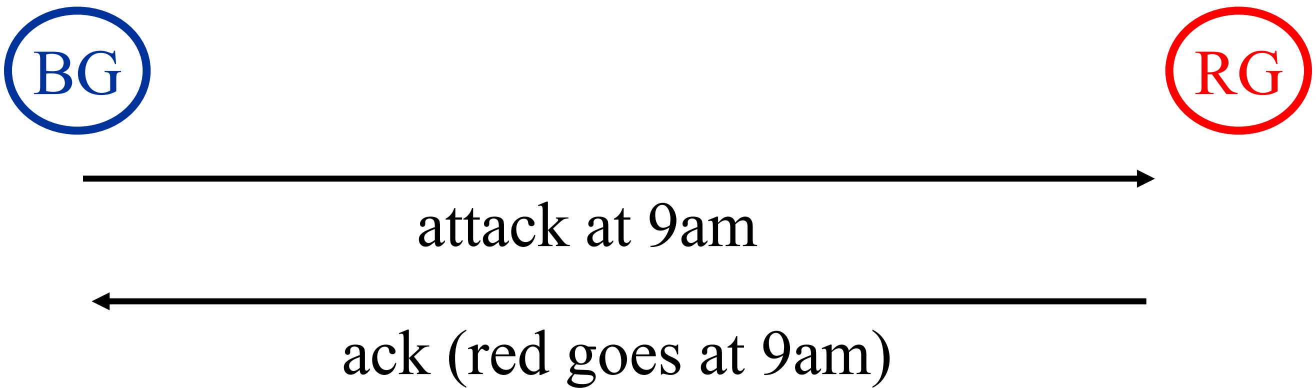
How Many Messages Do We Need?

assume blue starts...

Is this enough??

How Many Messages Do We Need?

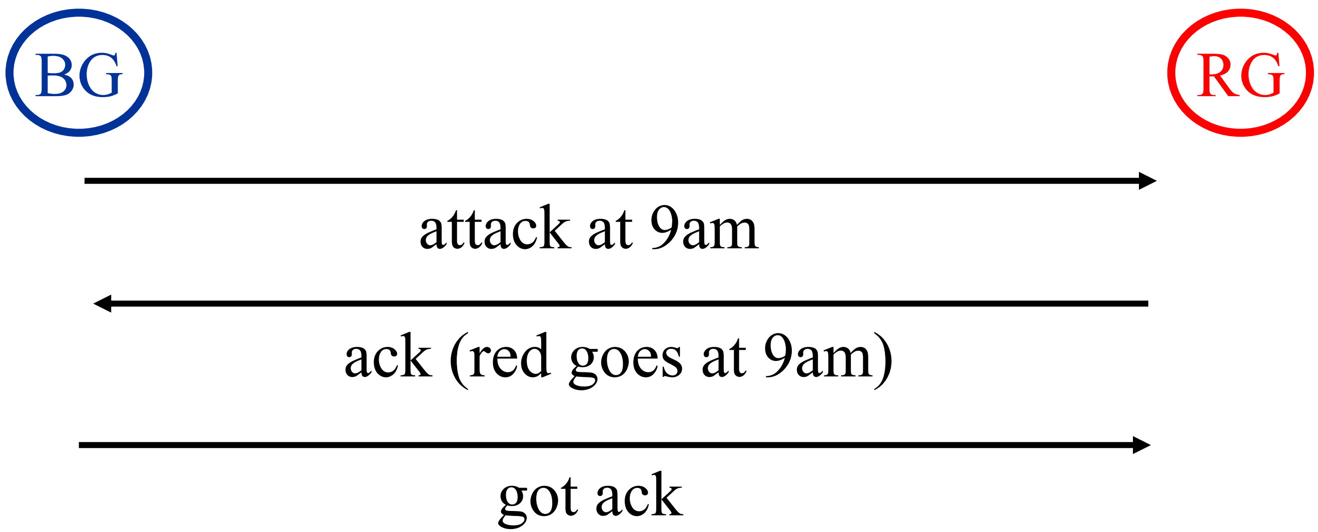
assume blue starts...



Is this enough??

How Many Messages Do We Need?

assume blue starts...



Is this enough??

Stated problem is Impossible!

- **Theorem:** There is no protocol that uses a finite number of messages that solves the two-generals problem (as stated here)

- **Proof:** Consier the shortest such protocol(execution)
  - Consider last message
  - Protocol must work if last message never arrives
  - So don’t send it
  - But, now you have a shorter protocol(execution)

Stated problem is Impossible!

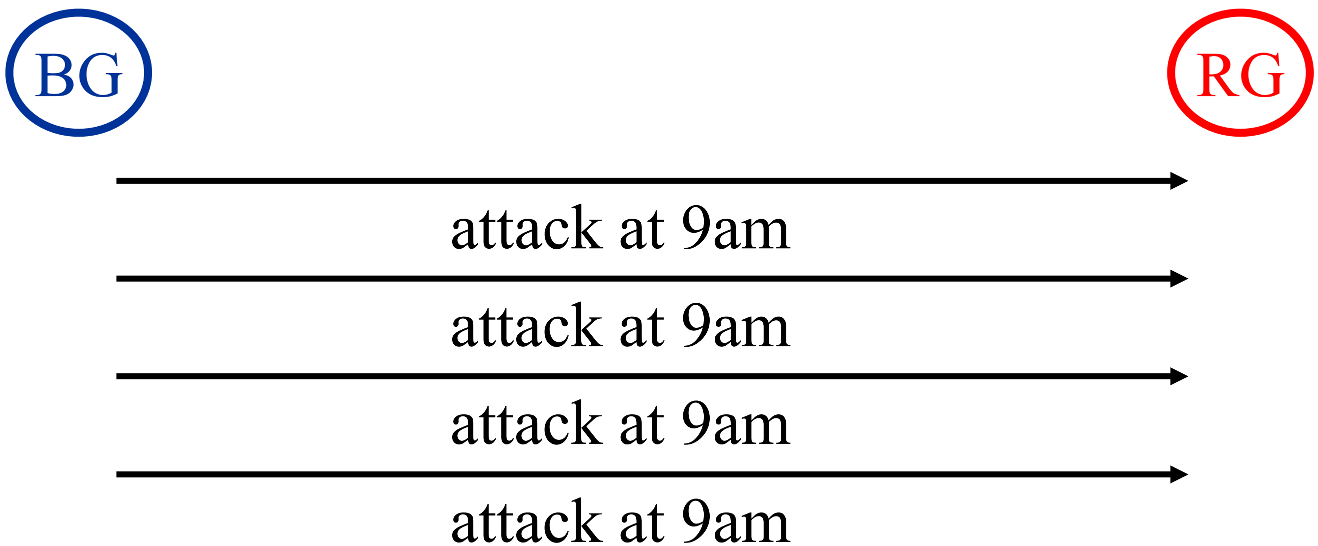
- **Theorem:** There is no protocol that uses a finite number of messages that solves the two-generals problem (as stated here)

Alternatives??

Probabilistic Approach?

- Send as many messages as possible, hope one gets through...

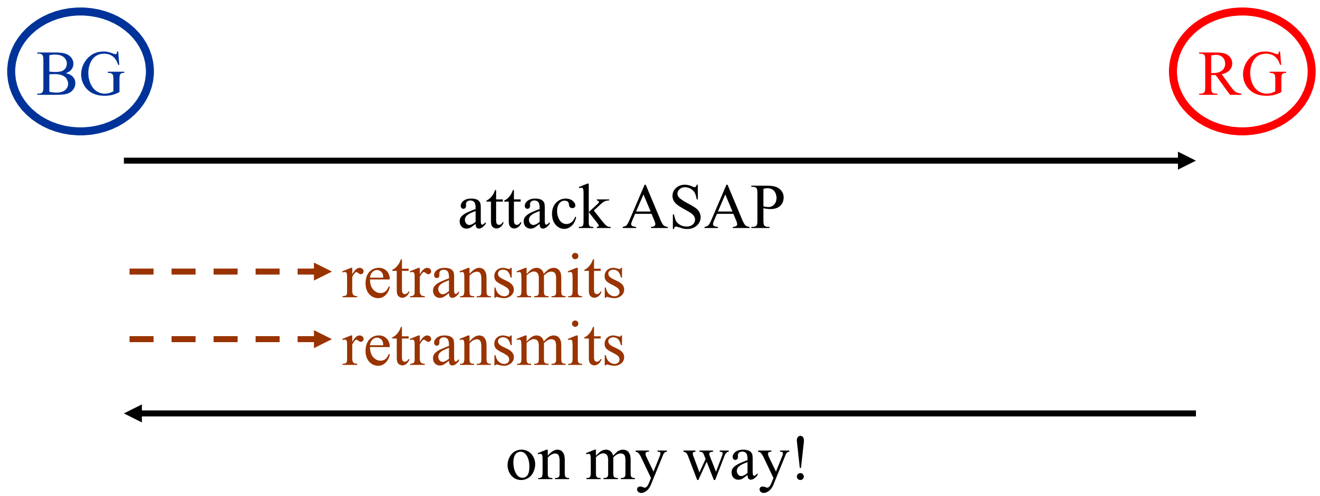
assume blue starts...



Eventual Commit

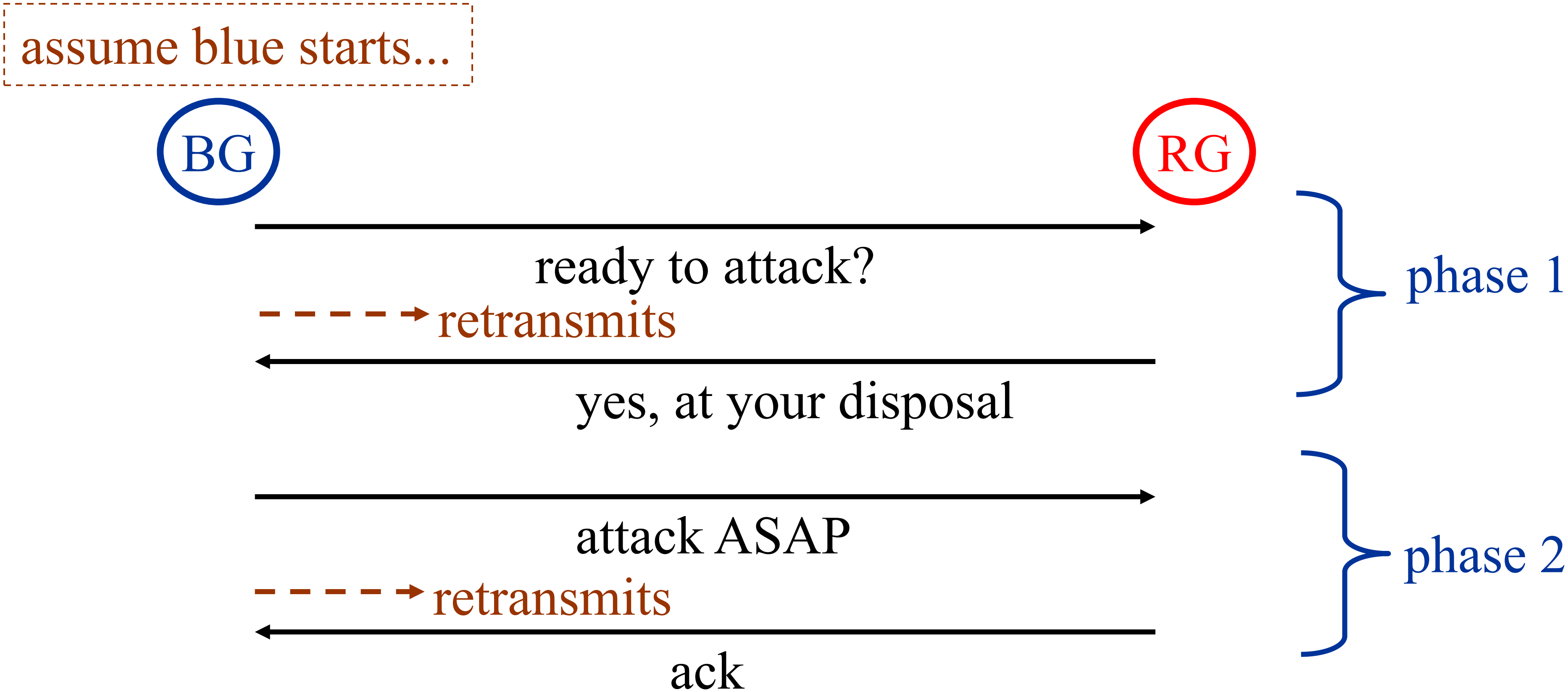
- Eventually both sides attack...

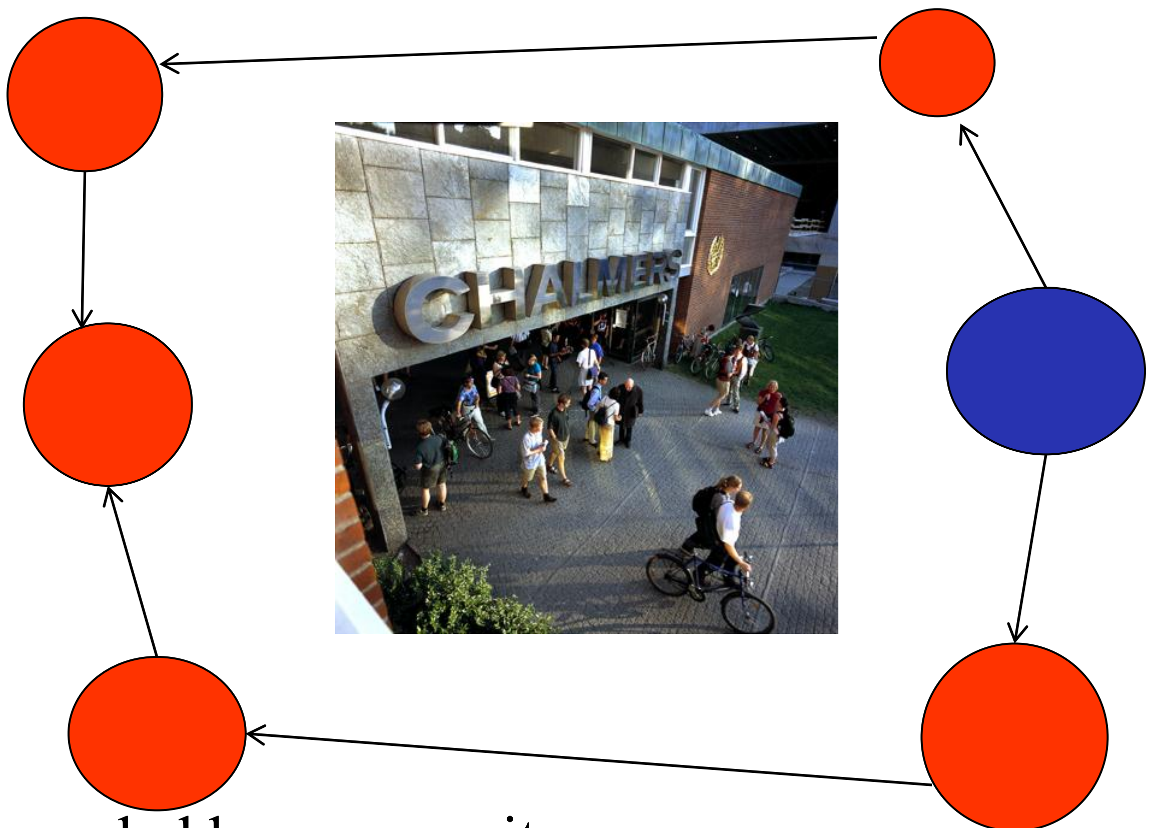
assume blue starts...



2-Phase Eventual Commit

- Eventually both sides attack...





- Chalmers surrounded by army units
- Armies have to attack **simultaneously in order to conquer Chalmers**
- Communication between generals by **means of messengers**
- Some generals of the armies are **traitors**

The Byzantine agreement problem

- **One process(the source or commander) starts with a binary value**
- •Each of the remaining processes (the **lieutenants) has to decide on a binary value such that:**
- •**Agreement:** all non-faulty processes agree on the same value
- •**Validity:** if the source is non-faulty, then all non-faulty processes agree on the initial value of the source
- •**Termination:** all processes decide within finite time
- •So if the source is faulty, the non-faulty processes can **agree on any value**
- •It is irrelevant on what value **a faulty process decides**

Byzantine Empire

Conditions for a solution for Byzantine faults

- Number of processes: **n**
- Maximum number of possibly failing processes: **f**
- **Necessary and sufficient condition** for a solution to Byzantine agreement:

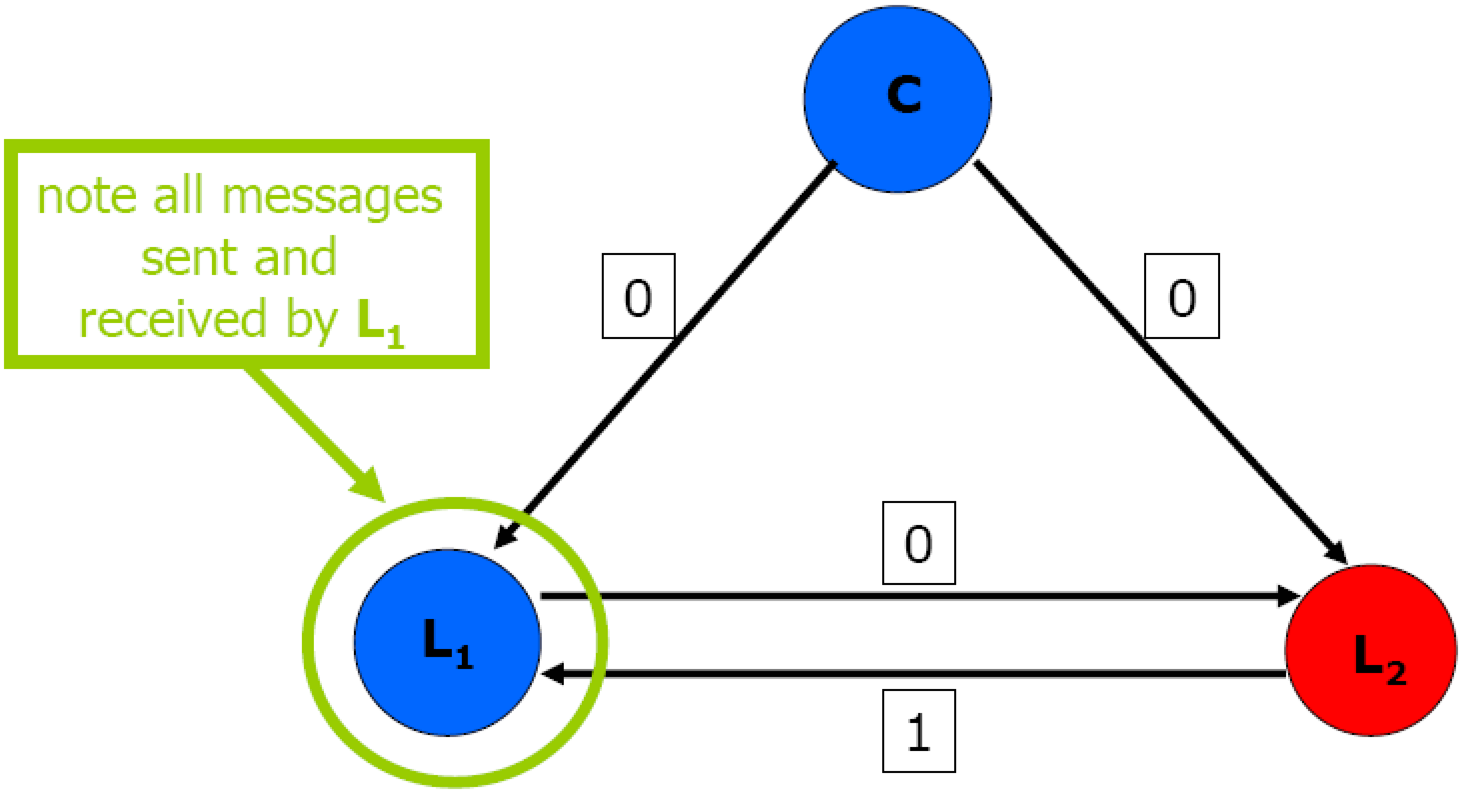
  $f<n/3$

- •**Minimal number of rounds** in a deterministic solution**:**

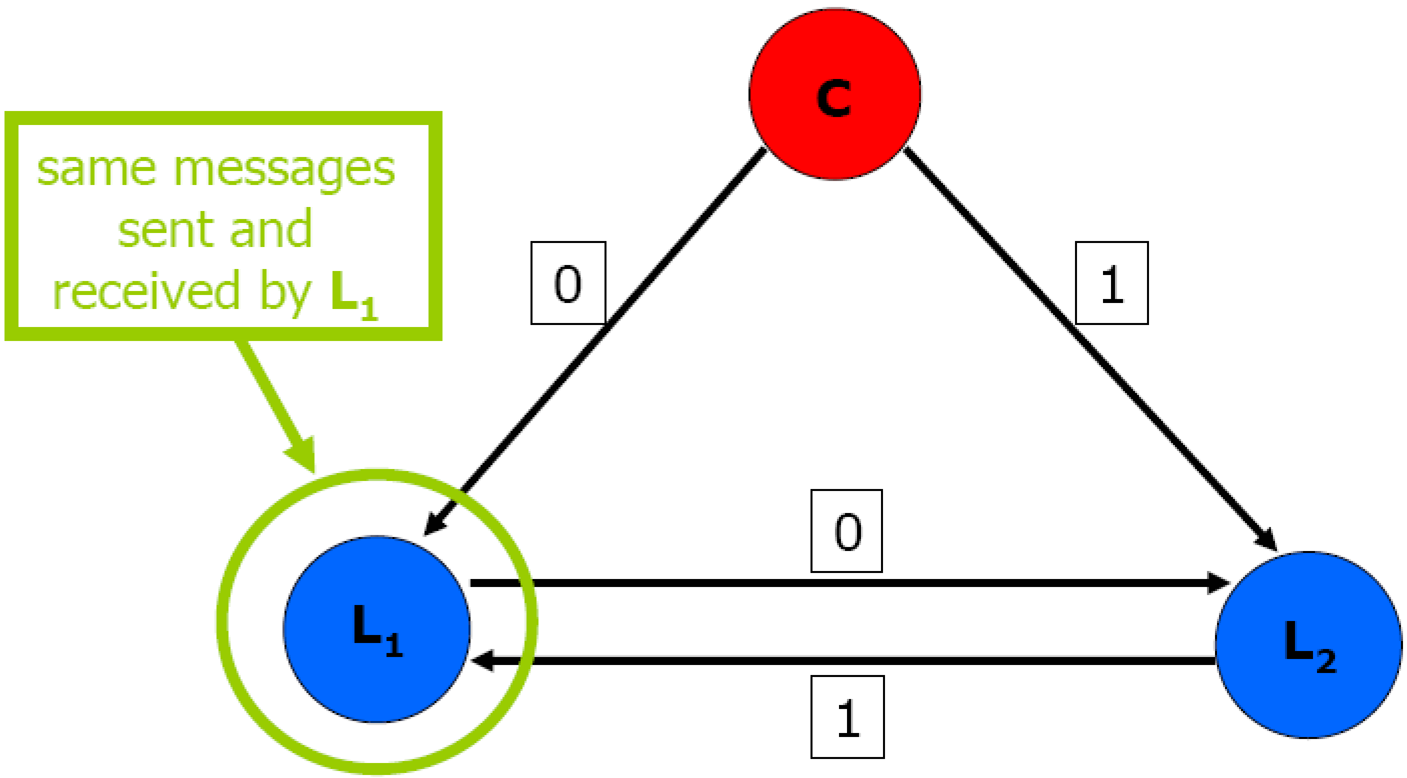
  $f+1$

- •There exist **randomized solutions with a lower expected number of rounds**

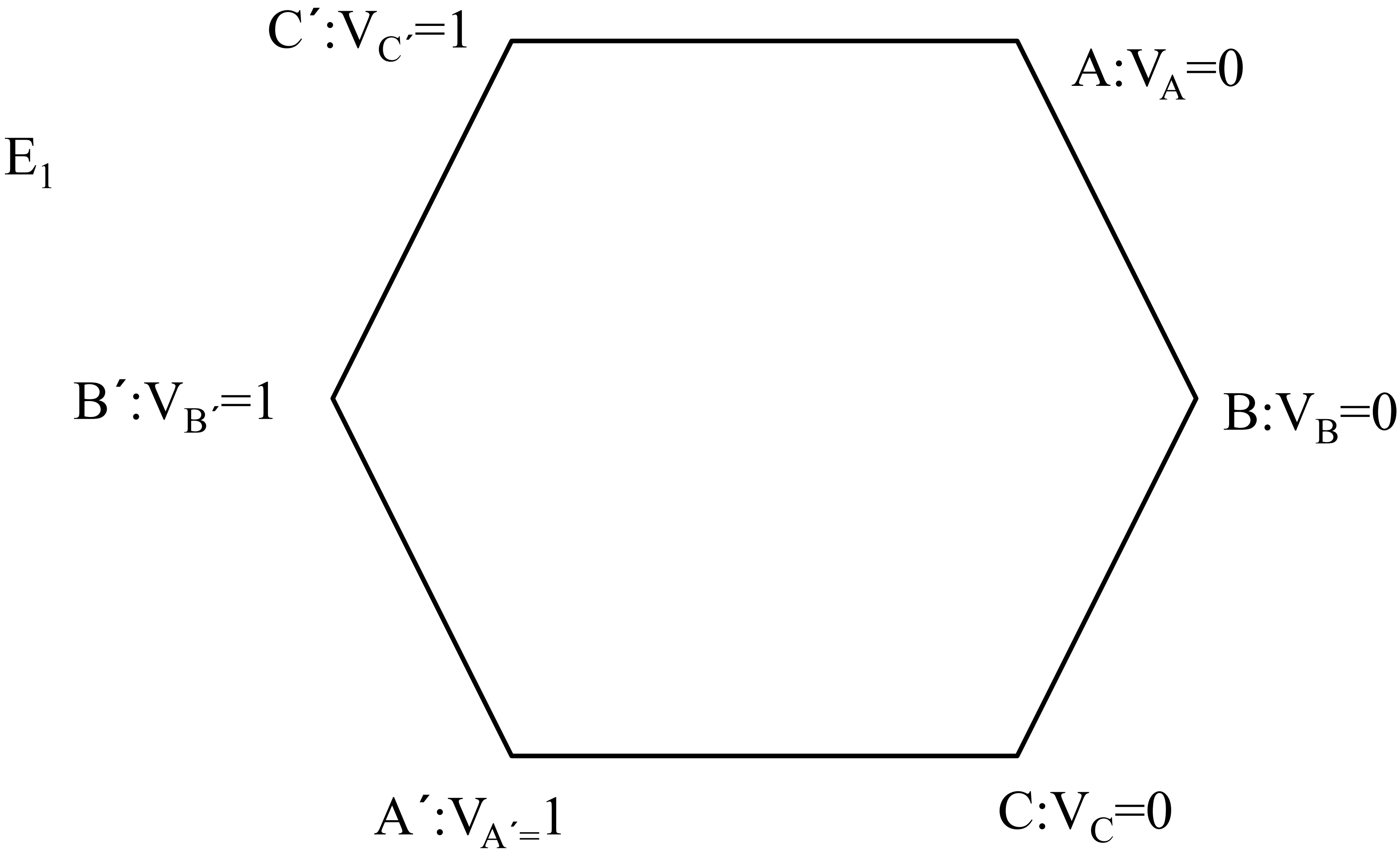
Senario 1



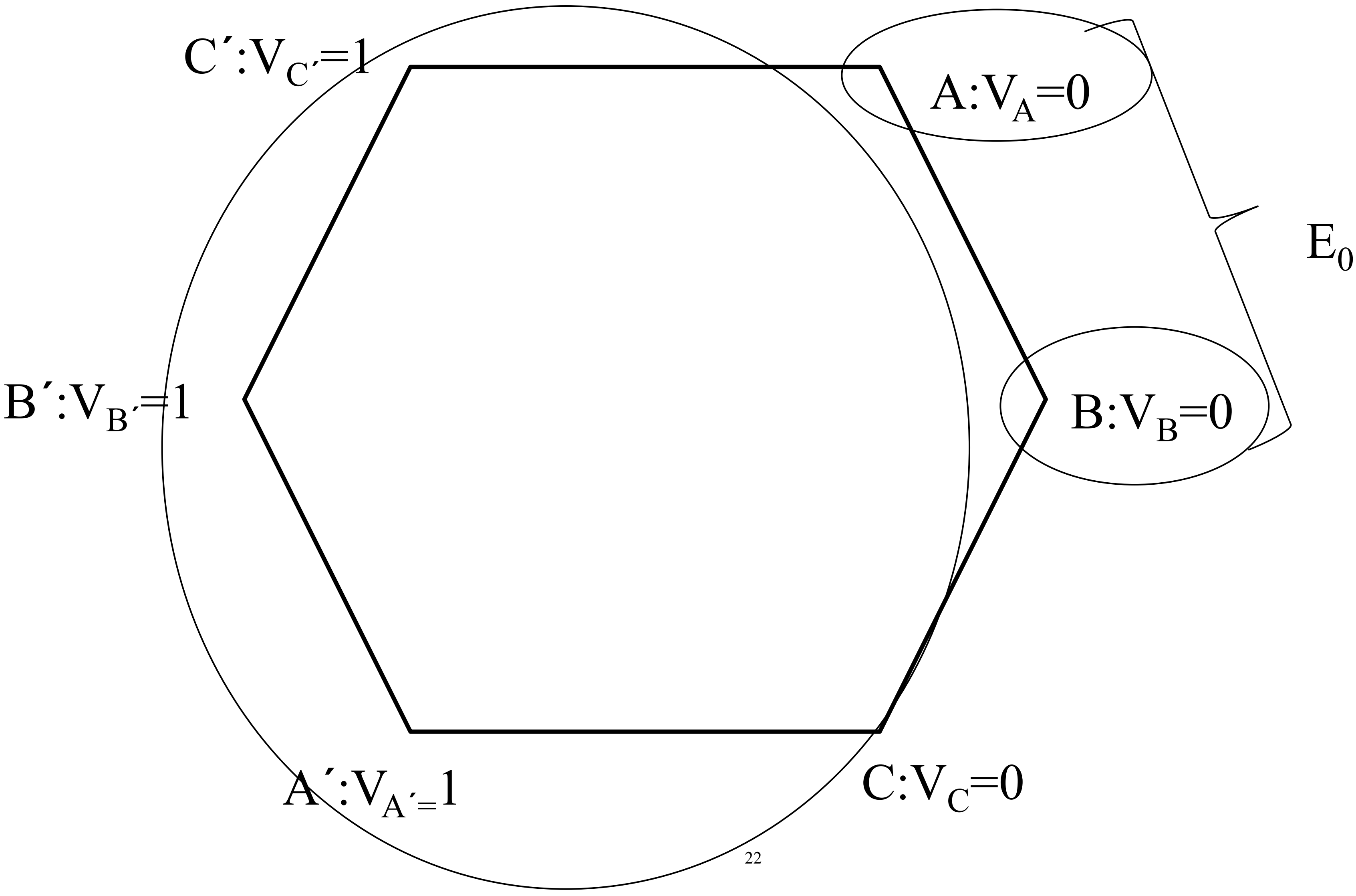
Senario 2



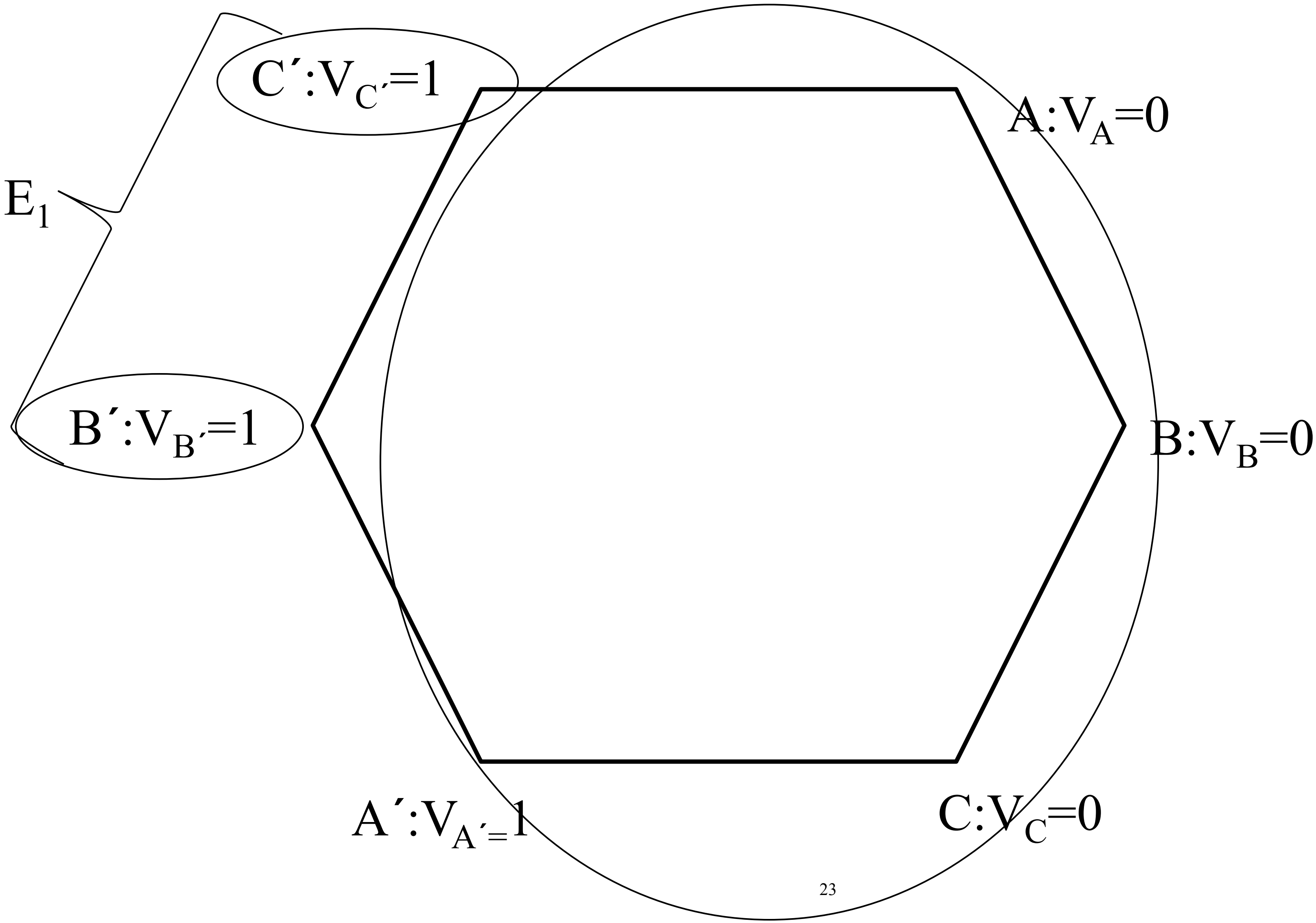
Impossibility of 1-resilient 3-processor Agreement



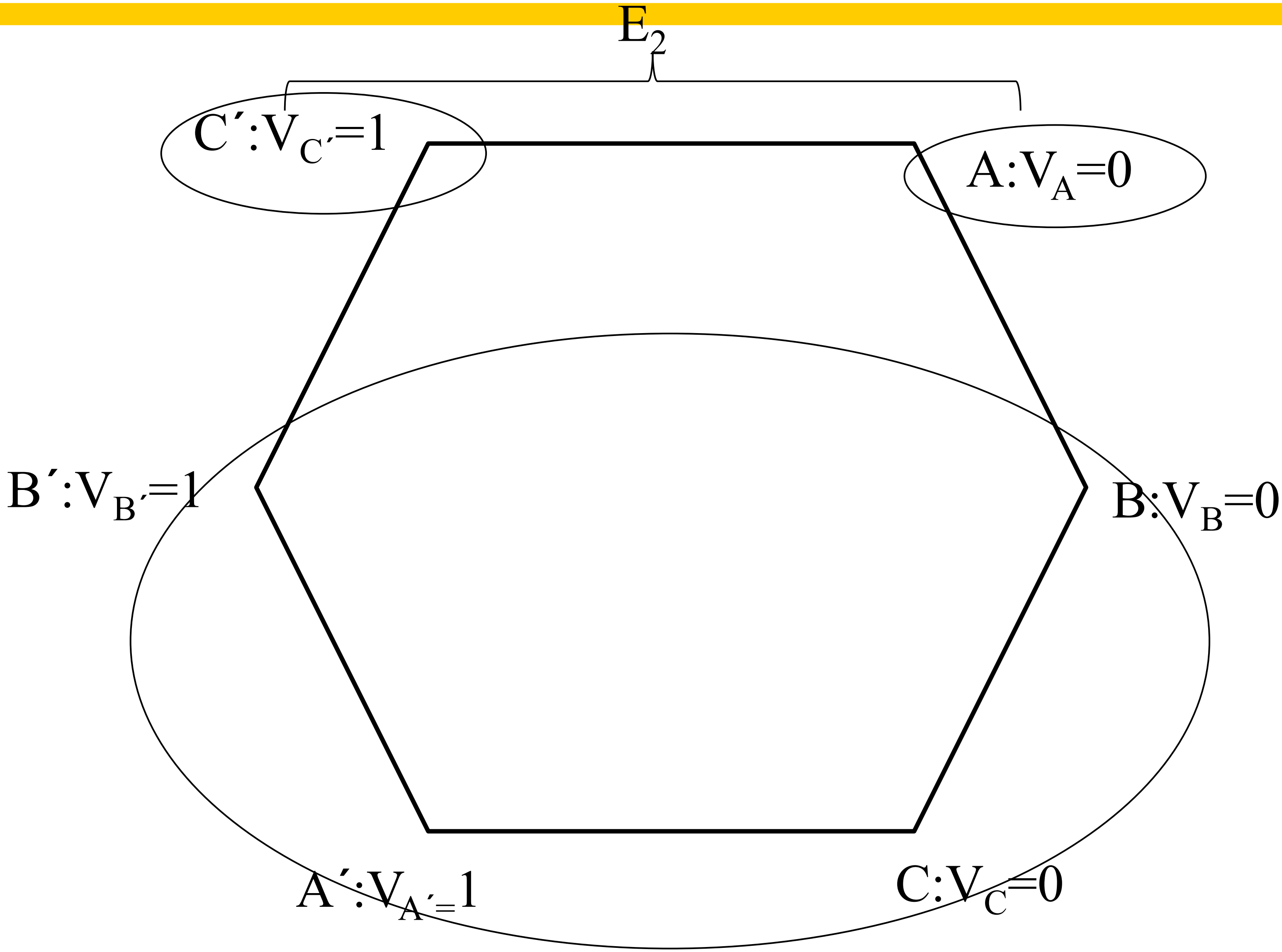
Impossibility of 1-resilient 3-processor Agreement



Impossibility of 1-resilient 3-processor Agreement



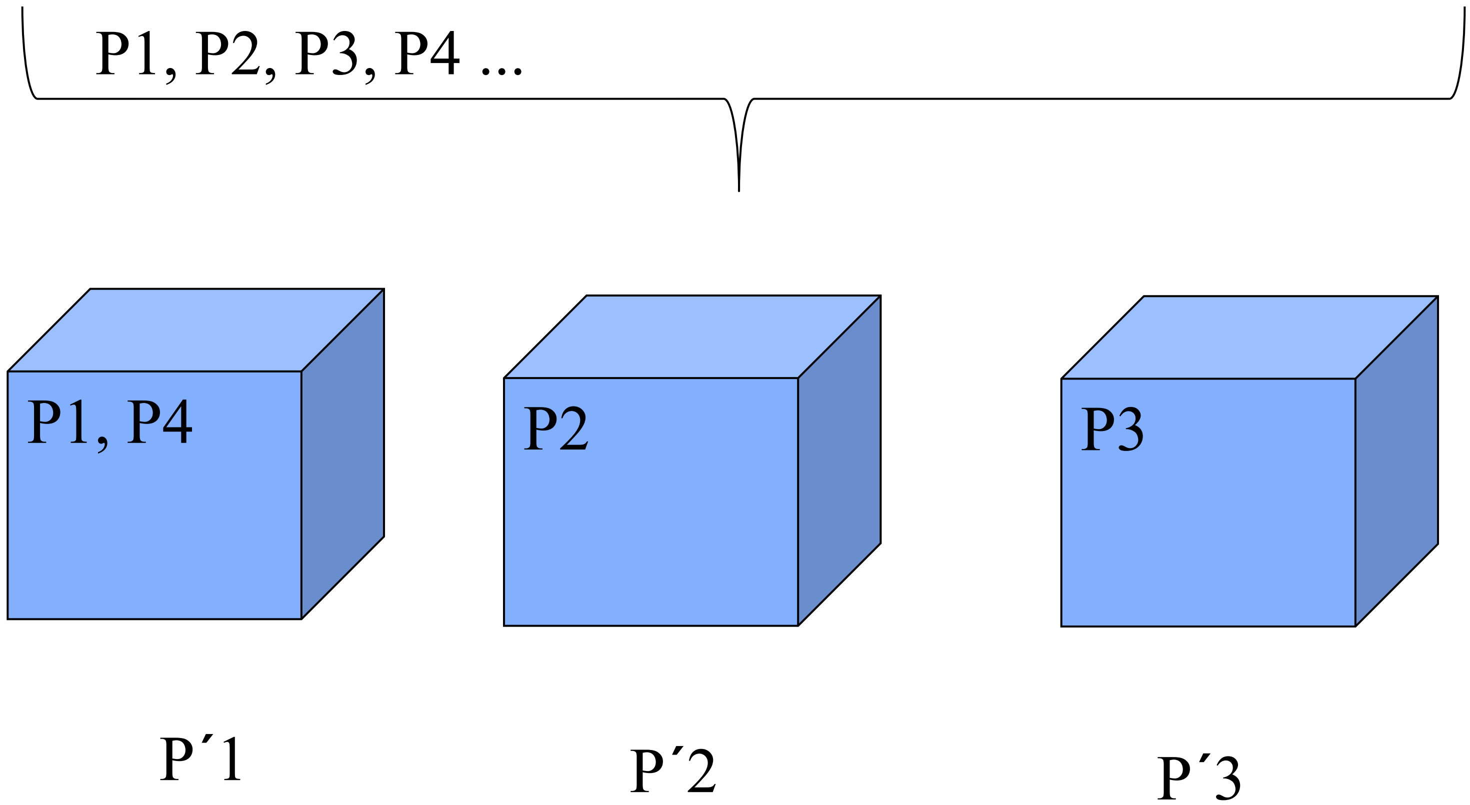
Impossibility of 1-resilient 3-processor Agreement



Proof

- In E_0 A and B decide 0
- In E_1 B´ and C´ decide 1
- In E_2 C´ has to decide 1 and A has to decide 0, contradiction!

t-resilient algorithm requiring n<=3t processors, t=>2



Consensus in a Synchronous System

- For a system with at most *f* processes crashing, the algorithm proceeds in *f+1* rounds (with timeout), using basic multicast.
- $Values^r_i$: the set of proposed values known to P_i at the beginning of round *r*.
- Initially $Values^0_i = \{\}$; $Values^1_i = \{v_i\}$

for round = 1 to f+1 do

multicast ($Values^r_i - Values^{r-1}_i$)

$Values^{r+1}_i \leftarrow Values^r_i$

for each V_j received

$Values^{r+1}_i = Values^{r+1}_i \cup V_j$

end

end

d_i = minimum($Values^{f+2}_i$)

Proof of Correctness

- Proof by contradiction.
- Assume that two processes differ in their final set of values.
- Assume that p_i possesses a value v that p_j does not possess.
  - → A third process, p_k, sent v to p_i, and crashed before sending v to p_j.
  - → Any process sending v in the previous round must have crashed; otherwise, both p_k and p_j should have received v.
  - → Proceeding in this way, we infer at least one crash in each of the preceding rounds.
  - → But we have assumed at most f crashes can occur and there are $f+1$ rounds → contradiction.

Byzantine agreem. with authentication

- Every message **carries a signature**
- The signature of a loyal general **cannot be forged**
- Alteration of the contents of a signed message can be detected
- Every (loyal) general can **verify the signature of any other (loyal) general**
- **Any number f of traitors can be allowed**
- Commander is process **0**
- Structure of message from (and signed by) the commander, and subsequently signed and sent by lieutenants **Li1, Li2,…:**
- **(v : s0 : si1: … : sik)**
- Every lieutenant maintains **a set of orders V**
- Some choice function on **V for deciding (e.g., majority, minimum)**

- •Algorithm in commander:

  send**(v: s0)to every lieutenant**

  – Algorithm in every **lieutenant Li**:

  upon receipt of (v : s0: si1: …. : sik) do

  if (v not in V) then

  V := V union {v}

  if (k < f) then

  for(j in {1,2,…,n-1} \{i,i1,…,ik}) do

  send(v: s0: si1: … : sik: i) to Lj

  If (Li will not receive any more messages) then decide(choice(V))