The Two Generals Problem

J. Gray,
Notes on Database Operating Systems,
In: Operating Systems: an Advanced Course,
Bayer et. al. eds., Lecture notes in Computer Science vol. 60, Springer-Verlag, 1978

Based on Proof by E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber, 1975
The Two Generals Problem
The N-Generals Problem
(aka: the Coordinated Attack Problem)

• Every process starts with an input in \{0,1\}
  – 1 is attack (commit)
  – 0 is don’t attack (abort)

• Goal is for all processes to eventually decide 0 or 1
  – Once a process makes a decision, it’s decision cannot be changed
Correctness Conditions

A solution (algorithm) must satisfy the following:

• **Agreement**: No two processes decide on different values

• **Validity**:
  - 1. If all processes start with 0, then 0 is the only possible decision value
  - 2. If all processes start with 1, and all messages are delivered, then 1 is the only possible decision value

• **Termination**: All processes eventually decide
System Model

• N process, connected network
  – Each process knows the entire network graph
• There are no process failures
• Messaging is not reliable (messages may be lost)
Algorithm Execution

• Initially:
  – Each process in arbitrary state
  – All channels are empty

• Processes repeatedly perform following, in lock-step:
  1. Receive and process messages from incoming channels (if any), and update state.
  2. Send messages on outgoing channels (optionally)
     * Messages may be lost

• These two steps are called a **round**
Impossibility Result

**Theorem:** There is no algorithm that solves the two generals problem.

There is no algorithm that guarantees agreement, validity, and termination.
Proof

• Proof by contradiction

• Assume such an algorithm exists.
  – Agreement: \( p_1 \) and \( p_2 \) decide on same value.
  – Validity: If \( p_1 \) and \( p_2 \) start with 0, they decide 0.
    If they both start with 1, and no messages are lost, they both decide 1.
  – Termination: Both \( p_1 \) and \( p_2 \) eventually decide.

• Suppose the algorithm takes \( r \) rounds to decide.
  – w.l.o.g. \( p_1 \) and \( p_2 \) both send a message in each round.
Execution A

• By the termination requirement, both processes decide.
• By the validity requirement, both decide 1.

No messages are lost.
Execution B looks exactly like Execution A to $p_1$
Therefore, $p_1$ decides 1
By termination and agreement, $p_2$ must also decide 1
Execution C

- Execution C looks exactly like Execution B to \( p_2 \)
- Therefore, \( p_2 \) decides 1
- By termination and agreement, \( p_1 \) must also decide 1

Last two messages are lost.

Round \( r \)

\( p_1 \) eventually decides 1

\( p_2 \) eventually decides 1
Execution X

- Following this reasoning, we reach execution where no messages are delivered and both processes must decide 1
Execution Y

- Execution Y looks exactly like execution X to $p_1$
- Therefore, $p_1$ decides 1
- By termination and agreement, $p_2$ must also decide 1
**Execution Z**

- Execution Z looks exactly like execution Y to $p_2$
- Therefore, $p_2$ decides 1
- By termination and agreement, $p_1$ must also decide 1
- This violates validity – both start with 0 and must decide 0
No Coordinated Attack?

- There is no algorithm that solves the coordinated attack problem in the present of message loss
  - Holds for $N > 2$ as well

- But we do need to solve this problem in real-world systems:
  - Must strengthen the model and/or relax the requirements

- Can make assumptions on probability of message loss:
  - There will be some chance of violating validity or agreement

- Can let processes use randomization
  - There will be some chance of violating validity or agreement