CS 60002: Distributed Systems

T3: Common Knowledge

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

Knowledge Hierarchy in Distributed System

- **Common knowledge:** The "Publicly known" fact
  - Each can assume **others know it**
  - $k \geq 1$ is the common knowledge when the father mentions that "*at least one of you are muddy*"

Knowledge Hierarchy in Distributed System

- **Common knowledge:** The "Publicly known" fact
  - Each can assume **others know it**
  - k ≥ 1 is the common knowledge when the father mentions that "*at least one of you are muddy*"

- **Distributed knowledge:** The knowledge that is "distributed" among the members of the group, known by someone in the group
  - A node **cannot assume** that others know it
  - k ≥ 1 is the distributed knowledge when the father does not mention anything, and the knowledge is purely based on the observation

Knowledge Hierarchy in Distributed System

- **Common knowledge:** The "Publicly known" fact
  - Each can assume **others know it**
  - k ≥ 1 is the common knowledge when the father mentions that "*at least one of you are muddy*"

- **Distributed knowledge:** The knowledge that is "distributed" among the members of the group, known by someone in the group
  - A node **cannot assume** that others know it
  - k ≥ 1 is the distributed knowledge when the father does not mention anything, and the knowledge is purely based on the observation

- **Reference**: *Joseph Y. Halpern and Yoram Moses. 1990. **Knowledge and common knowledge in a distributed environment**. J. ACM 37, 3 (July 1990)*

Hierarchy of Knowledge

- π is a given fact, $K_i(\pi)$ -> Agent i knows the fact π

Hierarchy of Knowledge

- π is a given fact, $K_i(\pi)$ -> Agent i knows the fact π
- Knowledge should satisfy two properties
  - An agent's knowledge at a given time must depend on its **local history**

Hierarchy of Knowledge

- π is a given fact, $K_i(\pi)$ -> Agent i knows the fact π
- Knowledge should satisfy two properties
  - An agent's knowledge at a given time must depend on its **local history**

The information that it started out + the events that is has observed

Hierarchy of Knowledge

- π is a given fact, $K_i(\pi)$ -> Agent i knows the fact π
- Knowledge should satisfy two properties
  - An agent's knowledge at a given time must depend on its **local history**
  - Agent knows only true things
- Say, G is a group of agents, Agent i belongs to the group G

Hierarchy of Knowledge

- π is a given fact, $K_i(\pi)$ -> Agent i knows the fact π
- Knowledge should satisfy two properties
  - An agent's knowledge at a given time must depend on its **local history**
  - Agent knows only true things
- Say, G is a group of agents, Agent i belongs to the group G

What does it mean to say that G knows π ?

Hierarchy of Knowledge

- D(G,π): The group has distributed knowledge π
  - Knowledge π is distributed in G if the union of knowledge for G equals to π
  - Someone who knew everything that each member of G knows would know π

Hierarchy of Knowledge

- D(G,π): The group has distributed knowledge π
  - Knowledge π is distributed in G if the union of knowledge for G equals to π
  - Someone who knew everything that each member of G knows would know π

- S(G,π): Someone in G knows π
  - Some members of G know π

Hierarchy of Knowledge

- D(G,π): The group has distributed knowledge π
  - Knowledge π is distributed in G if the union of knowledge for G equals to π
  - Someone who knew everything that each member of G knows would know π
- S(G,π): Someone in G knows π
  - Some members of G know π
- E(G,π): Everyone in G knows π
  - All members of G know π

Hierarchy of Knowledge

- D(G,π): The group has distributed knowledge π
  - Knowledge π is distributed in G if the union of knowledge for G equals to π
  - Someone who knew everything that each member of G knows would know π

- S(G,π): Someone in G knows π
  - Some members of G know π

- E(G,π): Everyone in G knows π
  - All members of G know π

- E(E(G,π)): Everyone in G knows that everyone in G knows π

Hierarchy of Knowledge

- E(G, k, π): π is E^k-knowledge in G
  - E(E(E…E(G,π))) k times

Hierarchy of Knowledge

- E(G, k, π): π is E^k-knowledge in G
  - E(E(E…E(G,π))) k times

- C(G,π): π is the common knowledge in G
  - E(G, k, π) for all k ≥ 1
  - E(G, 1, π) AND E(G, 2, π) AND E(G, 3, π) AND ….. AND E(G, m, π) AND ...

Hierarchy of Knowledge

- $E(G, k, \pi)$: π is E^k-knowledge in G
  - $E(E(E\ldots E(G,\pi)))$ k times

- $C(G,\pi)$: π is the common knowledge in G
  - $E(G, k, \pi)$ for all $k \geq 1$
  - $E(G, 1, \pi)$ AND $E(G, 2, \pi)$ AND $E(G, 3, \pi)$ AND ….. AND $E(G, m, \pi)$ AND ...

- The above notion of knowledge may not be distinct for all scenarios
  - Consider than n processor share a common memory
  - Knowledge is based on the contents of the common memory
  - $C(G,\pi) = E(G, k, \pi) = E(G, \pi) = S(G, \pi) = D(G, \pi)$

Hierarchy of Knowledge – Muddy Childrens

N number of children playing, k gets muddy

- π: "At least one child is muddy"
- If C(G, π) is not available
  - E(G, k-1, π): True
  - **E(G, k, π): False**

Hierarchy of Knowledge – Muddy Childrens

N number of children playing, k gets muddy

- π: "At least one child is muddy", k = 2
- Everyone sees at least one muddy child
  - E(G, 1, π) is true (everyone knows π: "At least one child is muddy")
  - Only muddy child that C1 sees is C2, C1 does not know whether he is muddy -> C1 consider it's possible that C2 is the only muddy child -> **C1 does not know that C2 knows E(G, 1, π)**

Hierarchy of Knowledge – Muddy Childrens

N number of children playing, k gets muddy

- π: "At least one child is muddy", k = 2
- Everyone sees at least one muddy child
  - E(G, 1, π) is true
  - Only muddy child that C1 sees is C2, C1 does not know whether he is muddy -> C1 consider it's possible that C2 is the only muddy child -> **C1 does not know that C2 knows E(G, 1, π)** -> **E(G, 2, π) is not true**

Hierarchy of Knowledge – Muddy Childrens

N number of children playing, k gets muddy

- π: "At least one child is muddy", k = 2
- At Round 2, when C1 and C2 both say "No"
  - C1 gets the knowledge that C2 has seen at least one muddy child -> C1 knows that C2 knows π; same for C2

Hierarchy of Knowledge – Muddy Childrens

N number of children playing, k gets muddy

- π: "At least one child is muddy", k = 2
- At Round 2, when C1 and C2 both say "No"
  - C1 gets the knowledge that C2 has seen at least one muddy child -> C1 knows that C2 knows π; same for C2
  - E(G, 2, π) becomes true -> C1 determines he must be muddy, same for C2

Hierarchy of Knowledge – Muddy Childrens

- The role of the father's statement was to improve children's state of knowledge from E(G, k-1, π) to E(G, k, π)
    - Father's statement is the common knowledge
    - Before father's announcement: D(G, π) and S(G, π) was true

Hierarchy of Knowledge – Muddy Childrens

- The role of the father's statement was to improve children's state of knowledge from E(G, k-1, π) to E(G, k, π)
  - Father's statement is the common knowledge
  - Before father's announcement: D(G, π) and S(G, π) was true
  - Father provides C(G, π) -> helps to make E(G, 1, π) to be true
  - When E(G, 1, π) is true, the responses from the muddy children provides additional knowledge to make E(G, 2, π) to be true

Hierarchy of Knowledge – Muddy Childrens

- The role of the father's statement was to improve children's state of knowledge from E(G, k-1, π) to E(G, k, π)
    - Father's statement is the common knowledge
    - Before father's announcement: D(G, π) and S(G, π) was true
    - Father provides C(G, π) -> helps to make E(G, 1, π) to be true
    - When E(G, 1, π) is true, the responses from the muddy children provides additional knowledge to make E(G, 2, π) to be true

- When there are k muddy children, and E(G, k, m) holds true, then the muddy children will be able to prove their correctness

Hierarchy of Knowledge

- Message passing (communication) in a distributed system helps to improve the state of knowledge -> **climbing up the hierarchy**

- **Fact discovery**: The act of changing the state of knowledge of a fact from being distributed knowledge to the levels of explicit knowledge
  - From D to S to E to C
  - Example: Finding deadlock in a set of distributed locks

- **Fact publication**: The act of changing the state of knowledge of a fact that is not common knowledge to common knowledge
  - From S to C
  - Example: New protocol for communication

Common Knowledge

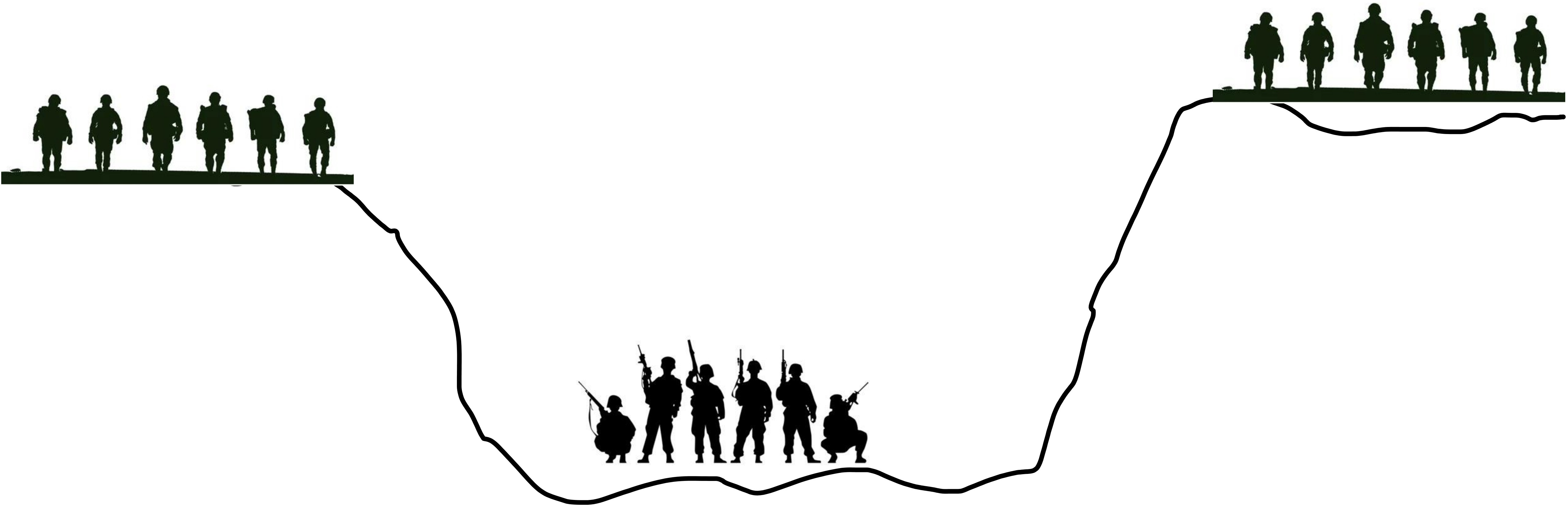
- How can we establish the common knowledge?

Common Knowledge

- **How can we establish the common knowledge?**
  - By being the part of the community
    - Example: The community of licensed drivers know what a driving sign means
  - By being co-present during the knowledge creation
    - Example: The children being same room of the father when he makes the announcement

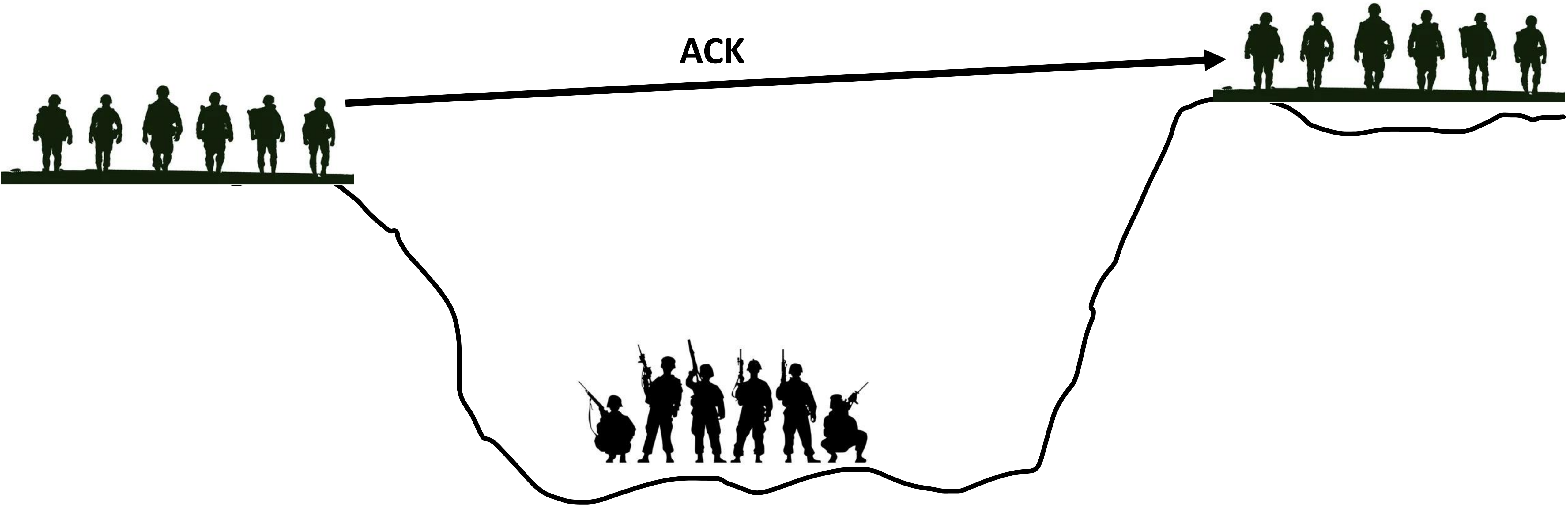
Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



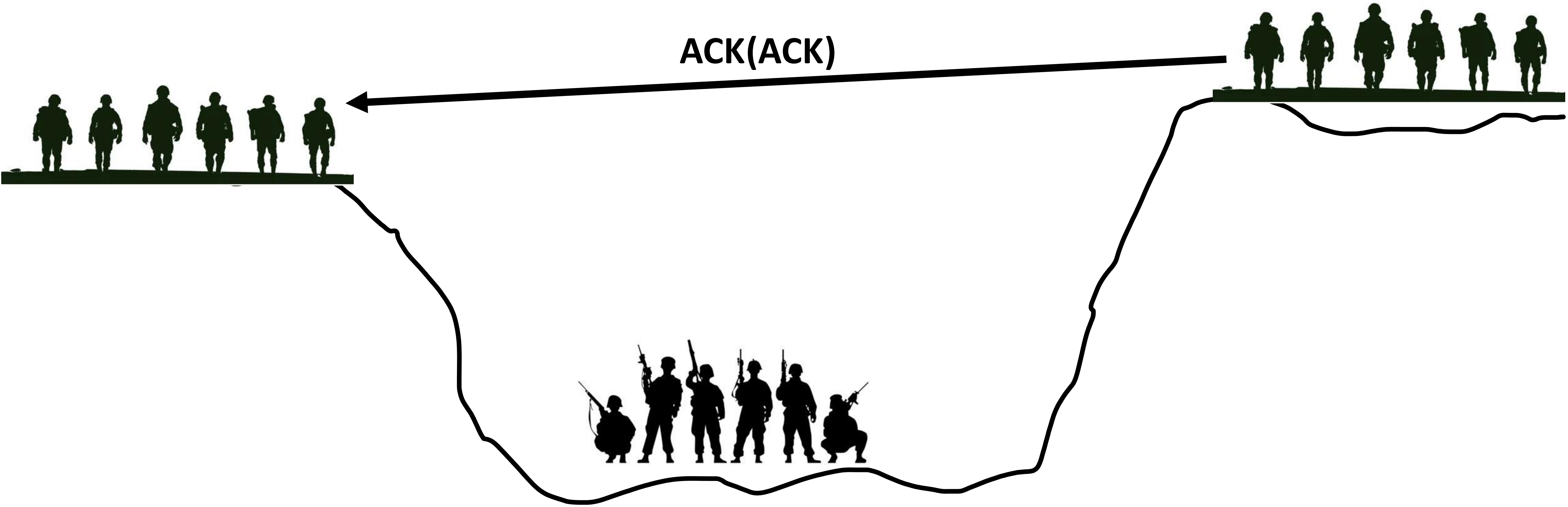
Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



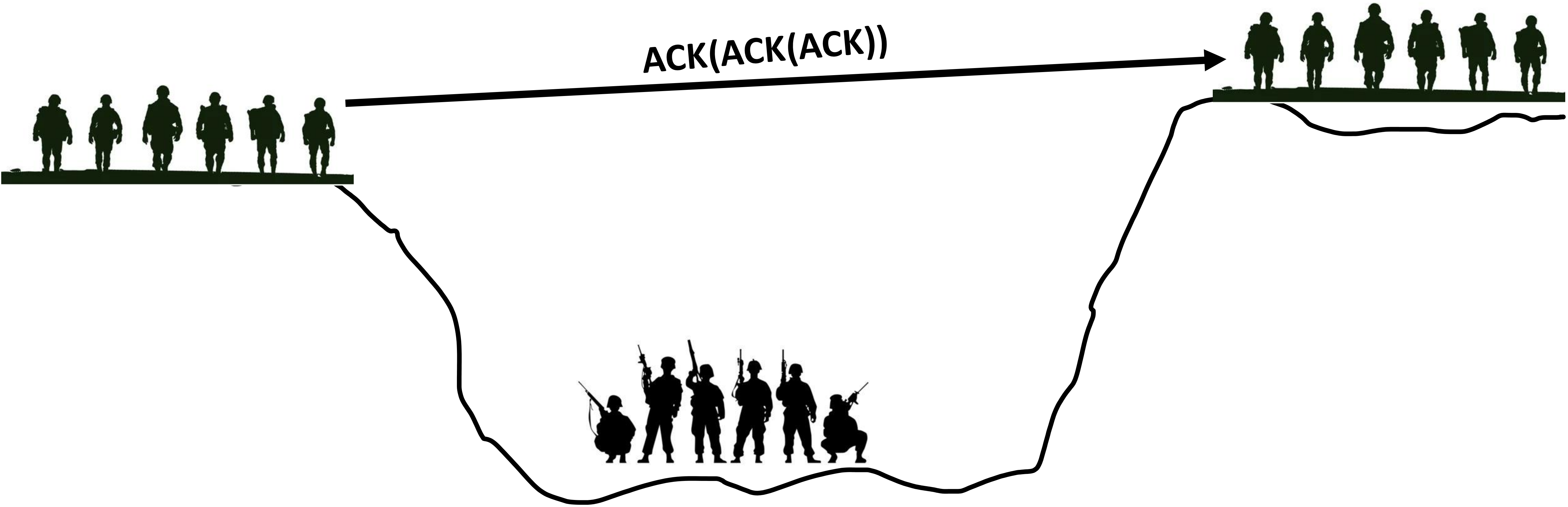
Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



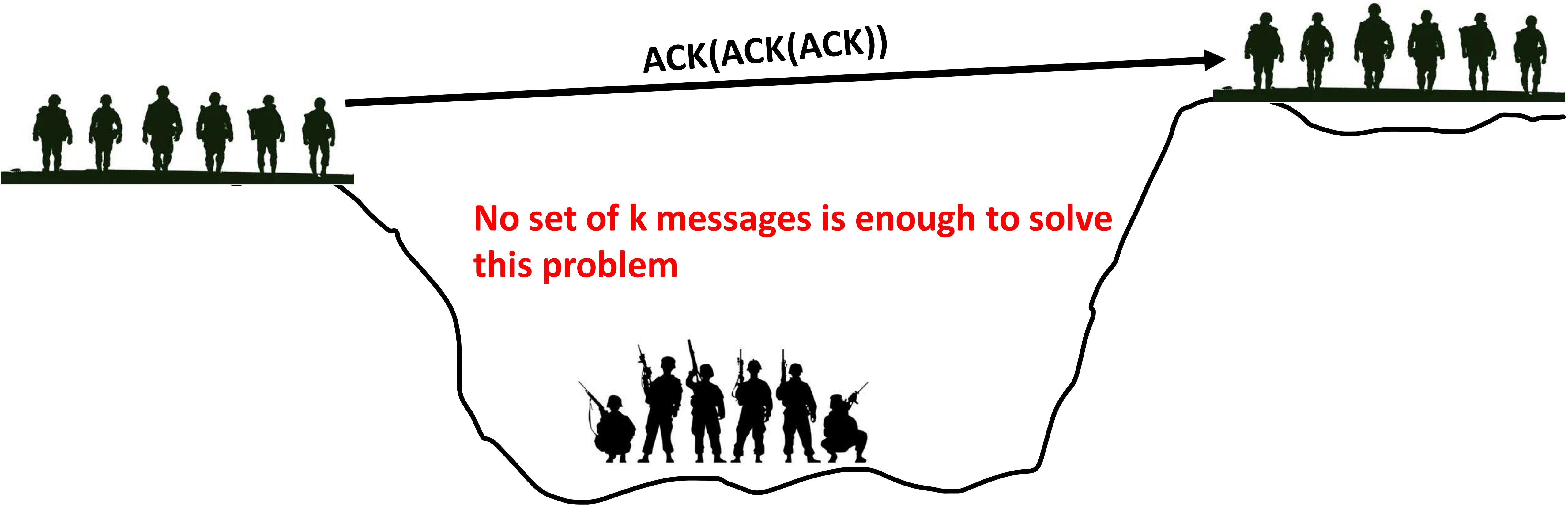
Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms

ACK(ACK(ACK))
Can go on infinitely ….

Coordinated Attack Problem

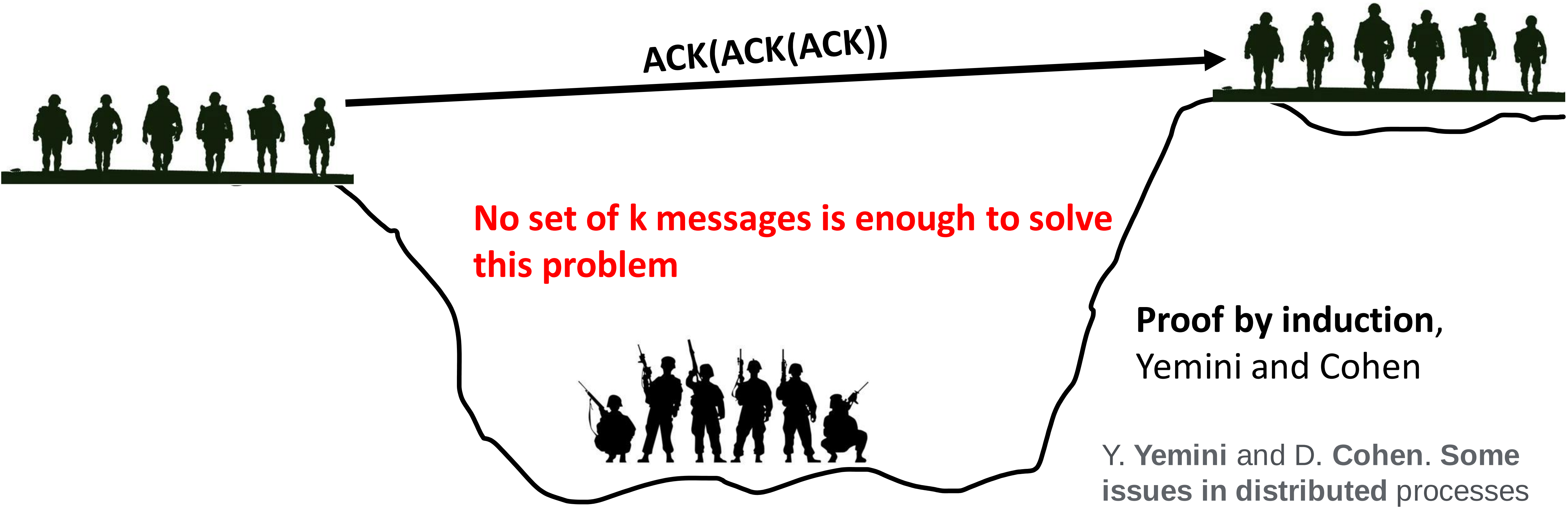
- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



No set of k messages is enough to solve this problem

Coordinated Attack Problem

- **Two generals want to attack in a coordinated way**
  - Does not attack until the other confirms



No set of k messages is enough to solve this problem

Proof by induction,
Yemini and Cohen

Y. **Yemini** and D. **Cohen**. **Some issues in distributed** processes **communication, ICDCS 1979**

Common Knowledge in the Coordinated Attack Problem

- When the generals attack, they are guaranteed to be attacking simultaneously
  - General A would know that General B is attacking
  - General B would know that General A knows that General B is attacking
  - General A would know that General B knows that General B knows that General A is attacking
  - …
  - **Both will have the common knowledge about the time of the attack**

Common Knowledge in the Coordinated Attack Problem

- Each message can add at most one level of knowledge and no more
  - When the message is first delivered to B, B knows about the A's desire to coordinate the attack
  - A does not know whether the message has been delivered -> does not know whether B knows A's desire
  - The message returns to A with B's ACK -> A knows that B knows A's desire, **but B does not know whether the message has been delivered**

Common Knowledge in the Coordinated Attack Problem

- Each message can add at most one level of knowledge and no more
  - When the message is first delivered to B, B knows about the A's desire to coordinate the attack
  - A does not know whether the message has been delivered -> does not know whether B knows A's desire
  - The message returns to A with B's ACK -> A knows that B knows A's desire, **but B does not know whether the message has been delivered**

- **Communication cannot be used to attain common knowledge in a system in which communication is not guaranteed**

We'll see many such impossibilities later

General Model of a Distributed System

- A distributed system is a finite collection $\{p_1, p_2, \ldots, p_n\}$ of two or more processors that are connected by a communication network
- Processors do not have a global view of the time, no *global clock* (*real time*)
- Processors are state machines that possibly clock
  - Clock is monotone non-decreasing function of the real time
- If a processor has a clock, then its clock reading is part of its state.
- A ***run*** of a distributed system is a description of an execution of the system, from time zero to the end of execution
  - We assume that the system executes forever
  - If it terminates after finite time, then we consider that the state freezes there.
- A ***point*** is a pair (r, t) consisting of a run r and time $t \geq 0$.

General Model of a Distributed System

- A run r is characterized as follows
    - For each point (r, t), associate every processor p_i's local history $h(p_i, r, t)$
- $h(p_i, r, t)$: The sequence of events that p_i has observed up to time t in run r
- Every processor may not start at the same time. A processor *wakes up* at some time $t_{init} \geq 0$
    - ***Initial state****:* The state of the processor when it wakes up
- The history $h(p_i, r, t)$ consist of p_i's initial state and the sequence of messages p_i has sent and received up to, but not including those sent of received at time t

General Model of a Distributed System

- Any distributed system can be characterized by the set of runs R
  - Relative behavior of clocks
  - The properties of communication in the system
  - …

- A system is **synchronous** if in all possible runs of the system the processors and the communication medium work in synchronous phases

- A **truly asynchronous** system is one in which the set of runs allows any message sent to be delayed an unbounded amount of time before being delivered.

View in a Distributed System

- Every distributed agent maintains a **view** during each point of execution of the protocol

- At every point, each processor is assigned a **view**; we say that two points are *indistinguishable* to the processor if it has the same view in both.
    - Every view is associated with a set of facts – both **directly known** and **inferred**

- A processor is said to know a fact at a given point exactly if the fact holds at all of the points that the processor cannot distinguish from the given one.
    - A processor knows all of the facts that (information theoretically) follow from its view at the current point.

Coordinated Attack Problem

- **Any correct protocol for the coordinated attack problem has the property that whenever the generals attack, it is common knowledge that they are attacking**
  - General A attacks at the run (r, t) when the local knowledge on General B's attack time as well as General B's local knowledge about General A's attack time is available.
  - General B attacks at the run (r', t') when the local knowledge on General A's attack time as well as General A's local knowledge about General B's attack time is available.
  - The protocol to be correct, A should know that B knows (r, t) and B should know that A knows (r', t') => t should be equal to t'

Coordinated Attack Problem

- **Any correct protocol for the coordinated attack problem has the property that whenever the generals attack, it is common knowledge that they are attacking**
  - General A attacks at the run (r, t) when the local knowledge on General B's attack time as well as General B's local knowledge about General A's attack time is available.
  - General B attacks at the run (r', t') when the local knowledge on General A's attack time as well as General A's local knowledge about General B's attack time is available.
  - The protocol to be correct, A should know that B knows (r, t) and B should know that A knows (r', t') => t should be equal to t'

- **Unfortunately, common knowledge is not always attainable !**

Characterizing Communication in a Distributed System

- Given a system R, *communication* in R is **not guaranteed** if the following two conditions hold --
  - For all runs r and time t, there exists a run r' extending (r, t), such that r and r' have the same initial configuration and the same clock readings, and no messages are received in r' at or after time t => *it is always possible that from some point on no messages will be received.*
  - If in run r, processor p_i does not receive any message in the interval (t', t), then there is a run r' extending (r, t') such that r and r' have the same initial configuration and the same clock readings, $h(p_i, r, t'') = h(p_i, r', t'')$ for all $t'' \leq t$, and no processor $p_j \neq p_i$ receives a message in r' in the interval [t', t). => *if processor p_i does not get any information to the contrary (by receiving some message), then p_i considers it possible that none of its messages were received*

Communication and Common Knowledge

- **In a system in which communication is not guaranteed, common knowledge is not attainable.**
  - We omit the formal proof, check the paper for a formal proof of this statement

Communication and Common Knowledge

- **In a system in which communication is not guaranteed, common knowledge is not attainable.**
  - We omit the formal proof, check the paper for a formal proof of this statement

- **Point of Caution**: The above statement does not say that no fact can become a common knowledge in a system where communication is not guaranteed
  - Assume that there is a global clock in which all processor have access
  - *5:00 pm* in the clock is a common knowledge even if communication is not guaranteed
  - Same for a system with a shared memory where all the processors have access => Something written in the shared memory is a common knowledge irrespective of whether the communication is guaranteed

Some More Conjectures on the *Coordinated Attack Problem*

- **Even with a guaranteed delivery, attack cannot be coordinated if the communication is over an asynchronous channel**
  - Messages can be delayed for an unbounded amount of time
  - No guarantee that the other party sees the message before the attack time!

Some More Conjectures on the *Coordinated Attack Problem*

- **Even with a guaranteed delivery, attack cannot be coordinated if the communication is over an asynchronous channel**
  - Messages can be delayed for an unbounded amount of time
  - No guarantee that the other party sees the message before the attack time!

- **Let the delay in delivery time be bounded at β, although this bound is not a common knowledge. Can the attack be coordinated at that case?**

Some More Conjectures on the *Coordinated Attack Problem*

- **Even with a guaranteed delivery, attack cannot be coordinated if the communication is over an asynchronous channel**
  - Messages can be delayed for an unbounded amount of time
  - No guarantee that the other party sees the message before the attack time!

- **Let the delay in delivery time be bounded at β, although this bound is not a common knowledge. Can the attack be coordinated at that case?**
  - A sends message M1 to B at t1
  - B receives M1 within (t1 + β); B sends M2 (ACK or M1) to A => B needs to wait for at most β time to be sure that A received M2
  - Information of message M1 becomes a common knowledge after (t1 + 2β)
  - **However, there is a twist.**

Some More Conjectures on the *Coordinated Attack Problem*

- **Even with a guaranteed delivery, attack cannot be coordinated if the communication is over an asynchronous channel**
  - Messages can be delayed for an unbounded amount of time
  - No guarantee that the other party sees the message before the attack time!

- **Let the delay in delivery time be bounded at β, although this bound is not a common knowledge. Can the attack be coordinated at that case?**
  - A might receive M2 at time (t1 + β + α) where α ≤ β and sends message M3 (ACK of M2) because A does not know whether B knows that A has received M2 (**delivery time is not a common knowledge).**
  - The information of M2 needs to be a common knowledge => needs more 2β time.
  - This is an infinite process => needs infinite time to coordinate.

How Can We Attain Common Knowledge in a Distributed System?

- If multiple agents in the system can **simultaneously** converge on a single option among many available options
  - Think of the *Muddy Children Problem* with *k=2*. Two muddy children, C1 and C2. After Round 2, C1 knows that C2 is muddy and have seen C1 muddy, C2 knows that C1 is muddy and have seen C2 muddy, all other children have seen both C1 and C2 muddy => everyone can converge simultaneously

- The history of all agents must simultaneously change to reflect the fact which then becomes the common knowledge.

Attaining Common Knowledge

- Common knowledge is required for simultaneous coordination in a distributed system
  - Ensures that the agents of a distributed system converges to the **correct decision** (**Safety**) within a **finite number of iterations (Liveness)**

Attaining Common Knowledge

- Common knowledge is required for simultaneous coordination in a distributed system
  - Ensures that the agents of a distributed system converges to the **correct decision** (**Safety**) within a **finite number of iterations (Liveness)**

- **ε-Common knowledge**: When every agent knows the fact within time unit ε
  - Achieved through **synchronous broadcast** -- all agents are guaranteed to receive the fact within ε time units

Attaining Common Knowledge

- Common knowledge is required for simultaneous coordination in a distributed system
  - Ensures that the agents of a distributed system converges to the **correct decision** (Safety) within a **finite number of iterations (**Liveness**)**

- **ε-Common knowledge**: When every agent knows the fact within time unit ε
  - Achieved through **synchronous broadcast** -- all agents are guaranteed to receive the fact within ε time units

- **Eventual Common knowledge:** Every agent knows that *every other agents will either know the fact or will know it in the future*
  - Used in **Asynchronous System**; example: Byzantine agreement – once a value is decided by a correct agent, it knows that others will eventually decide the same value

In Summary

- To show that a distributed system would yield the correct output => Show that the output is a common knowledge of the system (either ε-common or eventual common, depending on whether the system is synchronous or asynchronous)
  - **Be Careful:** Every agent needs to be sure that others know (or will eventually know) the fact
  - Put your assumptions carefully – **Network** (Reliable/Fair loss/Arbitrary?), **Timing** (Synchronous/Partially Synchronous/Asynchronous?), **Failures** (Crash-stop, Crash-recovery, Byzantine?)

- **Safety** and **Liveness** criteria of a distributed system
  - More details to come soon

Happy Learning