CS 371D
Distributed Computing

Lorenzo Alvisi
Sandhya Rao Chagaleti

A first course in Distributed Computing...

Two basic approaches
- cover many interesting systems, and distill from them fundamental principles
- focus on a deep understanding of the fundamental principles, and see them instantiated in a few systems

What is a distributed system?

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

Leslie Lamport

A few intriguing questions

- How do we talk about a distributed execution?
- Can we draw global conclusions from local information?
- Can we coordinate operations without relying on synchrony?
- For the problems we know how to solve, how do we characterize the “goodness” of our solution?
- Are there problems that simply cannot be solved?
- What are useful notions of consistency, and how do we maintain them?
- What if part of the system is down? Can we still do useful work? What if instead part of the system becomes “possessed” and starts behaving arbitrarily—all bets are off?
Saving the world before bedtime

Two Generals' Problem

Problem:
- Romans win if attack simultaneously
- Otherwise, Barbarians win
- Romans must coordinate attack

Two Generals' Problem

Claim: There is no non-trivial protocol that guarantees that the Romans will always attack simultaneously.

Two Generals' Problem

- only communication is by messenger
- messengers must sneak through the valley
- they don't always make it

Two Generals' Problem

- Save Western Civilization
  (i.e. design a protocol that ensures Romans always attack simultaneously)
**Claim** There is no non-trivial protocol that guarantees that the Romans will always attack simultaneously.

**Proof:** By contradiction, consider a protocol that solves the Two Generals problem using the least number of messages. Let that number be $n$. Consider the $n$-th message $m_{last}$.

- The state of sender of $m_{last}$ cannot depend on $m_{last}$ receipt.
- The state of receiver of $m_{last}$ cannot depend on $m_{last}$ receipt because in some executions $m_{last}$ could be lost!
- So both sender and receiver would come to the same conclusion even without sending $m_{last}$.
- We now have a new solution requiring only $n-1$ messages!

**Conclusion:** A solution requires reliable message delivery.