Characterization of Distributed Systems

Marco Aiello

(based on http://www.cdk4.net)

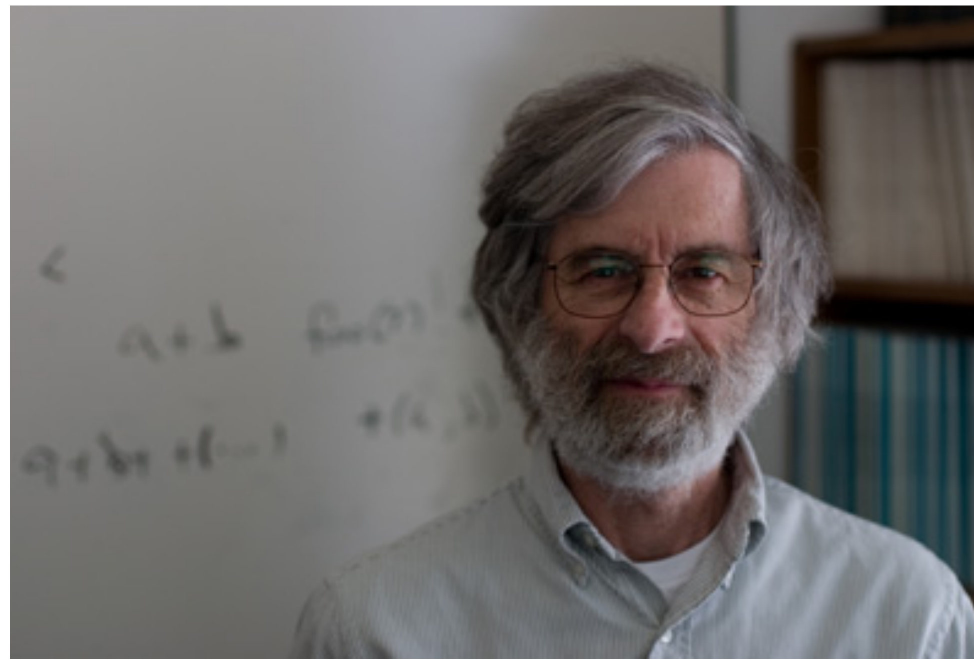
What is a distributed system?

A distributed system is a collection of autonomous hosts that are connected through a computer network. Each host executes computations and operates a distribution middleware, which enables the components to coordinate their activities via message-passing in such a way that users perceive the system as a single, integrated computing facility.

or better said

"A distributed system is one in which the failure of a machine you have never heard of can cause your own machine to become unusable"

Leslie Lamport



How to think in Distributed Systems Terms

- Verify correctness of protocols
- Verify efficiency of protocols
- Hosts are independent state machines
- Deal with concurrency, consistency, replication

Two generals problem

- Two Generals need to coordinate an attack against an enemy. If they attack individually, they will loose, if they attack together they will win.
- But the enemy lies in the middle and can intercept the coordination messages and avoid delivery
- Can the generals defeat the enemy?

Two generals problem

Proof sketch

- **Theorem:** there is no non-trivial protocol that guarantees that the generals will always attack simultaneously

- **Proof:** Ab absurdum, suppose there is one such protocol that does the job in the minimum number of steps n>0.
- Consider the last message sent, the n-th. The state of the sender cannot depend on its receipt, the state of the receiver cannot depend on its arrival, so they both do not need the n-th message. So we would have a protocol with n-1 messages. But that contradict the hypothesis

- **Fact:** A solution requires reliable message delivery.

Modelling the state of the computation

- State machine to model a system who's output depends on the current state and input (e.g., incoming messages)
- Finite State Machine (FSM):
  - Finite set of states $S \supseteq \{\text{initial state } s0\}$
  - Set of inputs messages I
  - Set of outputs messages O
  - Next state function: $f: S \times I \rightarrow S \times O$

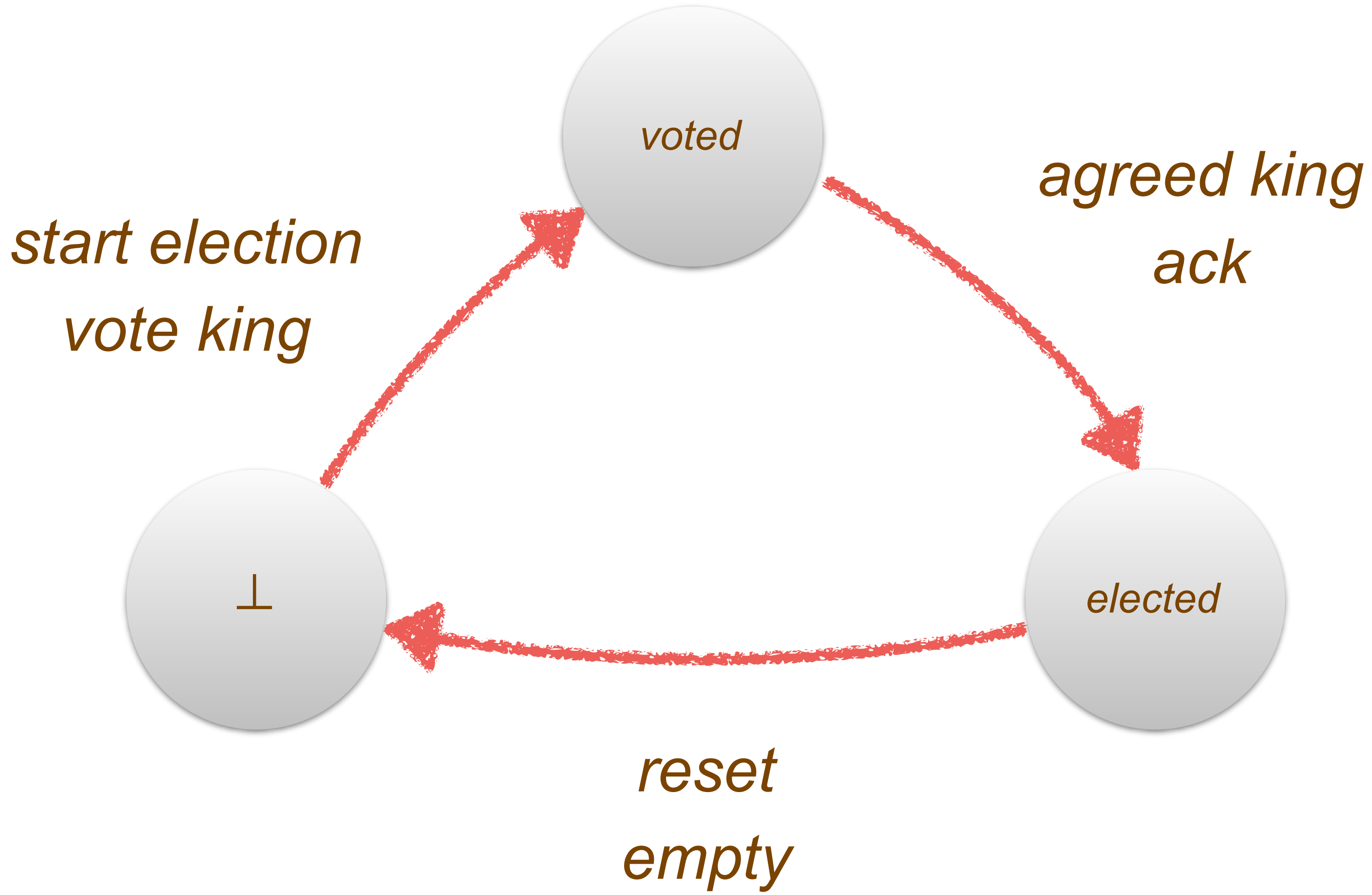
Finite State Machine

Example

- S={⊥,voted,elected}
- s0 = ⊥
- I = {start election, agreed king, reset}
- O = {vote king, ack, empty}
- Next state function:
    - (⊥,start election) -> (voted,vote king)
    - (voted,agreed king) -> (elected,ack)
    - (elected,reset) -> (⊥,empty)
    - any other combination - > (⊥,empty)

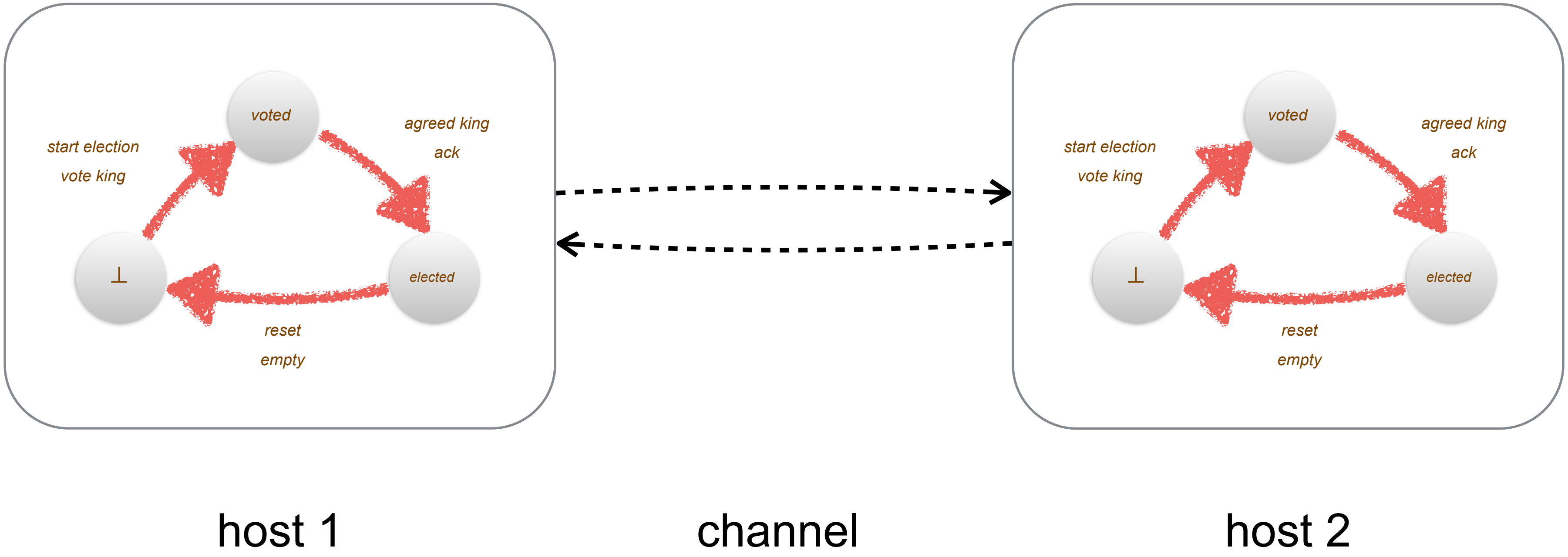
Finite State Machine

Example schema



State of a Distributed System

- A distributed system is modelled by the set of states of its components and of its channels



Some important (negative) results

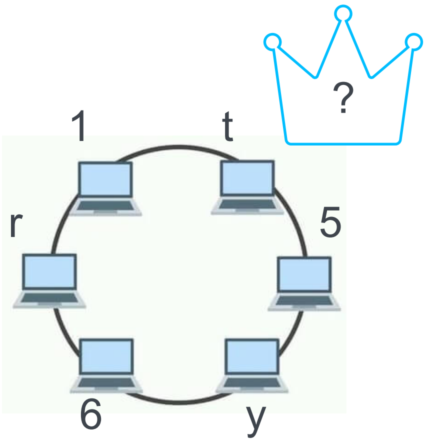
- No leader election in anonymous rings

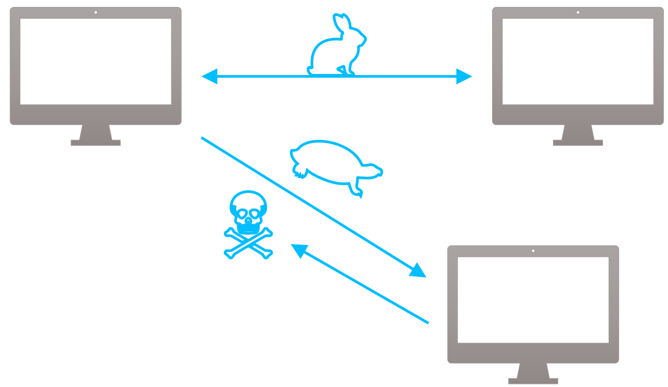
- No consensus in asynchronous systems

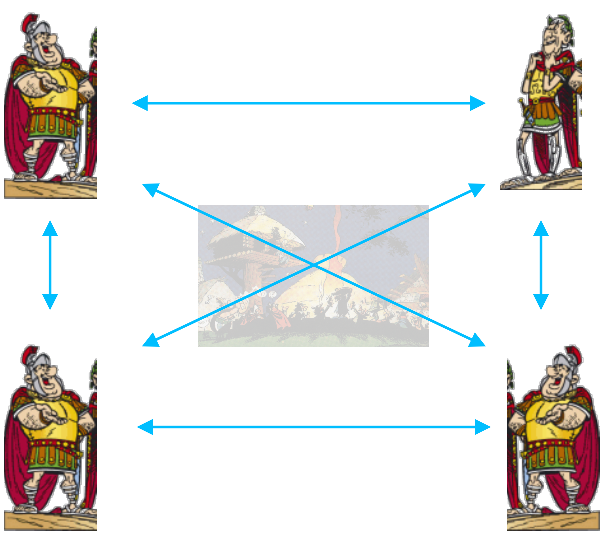
- Byzantine fault tolerance with at most $\lceil \frac{1}{3}n - 1 \rceil$

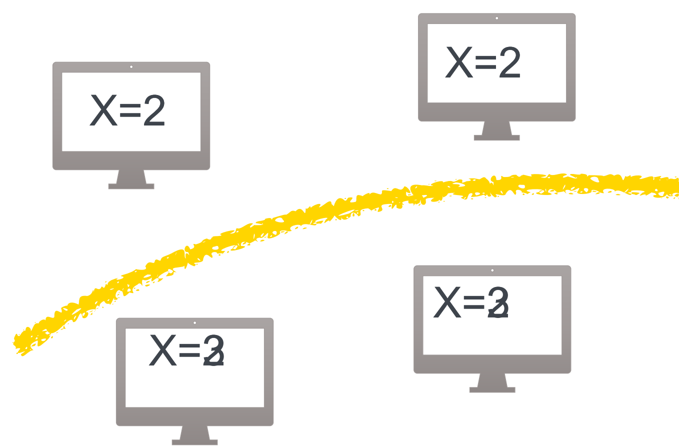
- CAP theorem (Consistency, Availability, Partitioning Tolerance)

In pictures



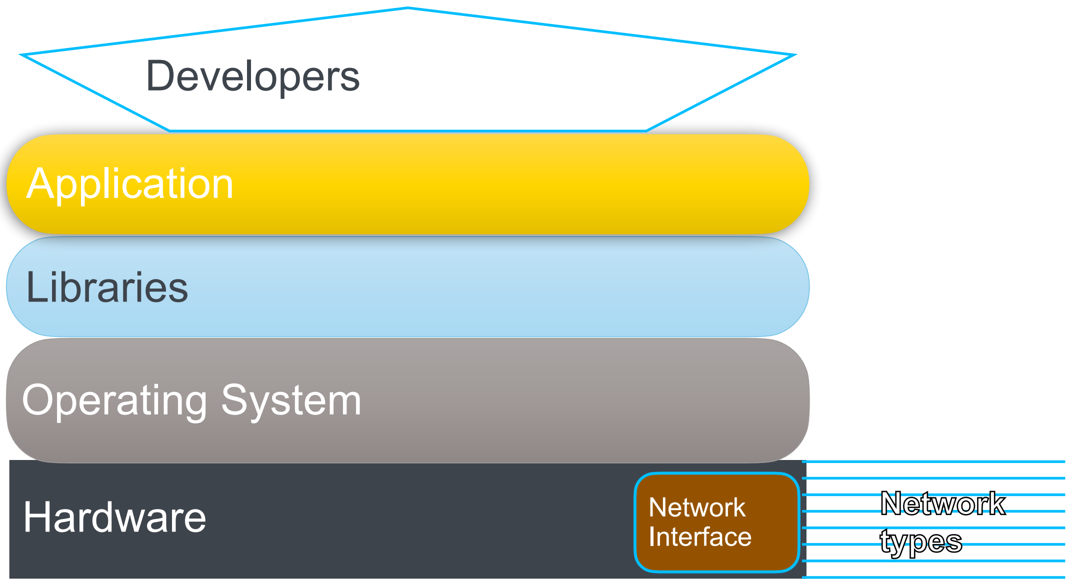






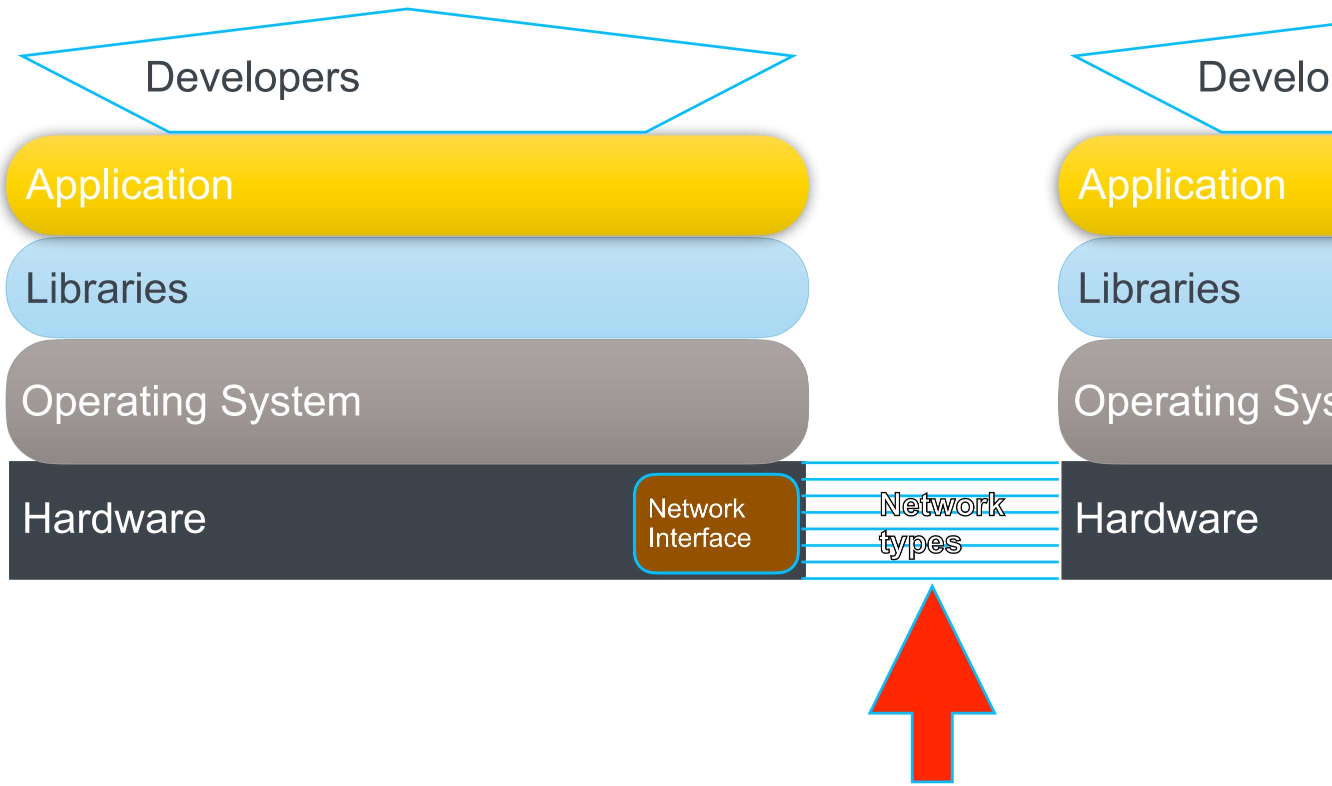
Distributed Systems Challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency



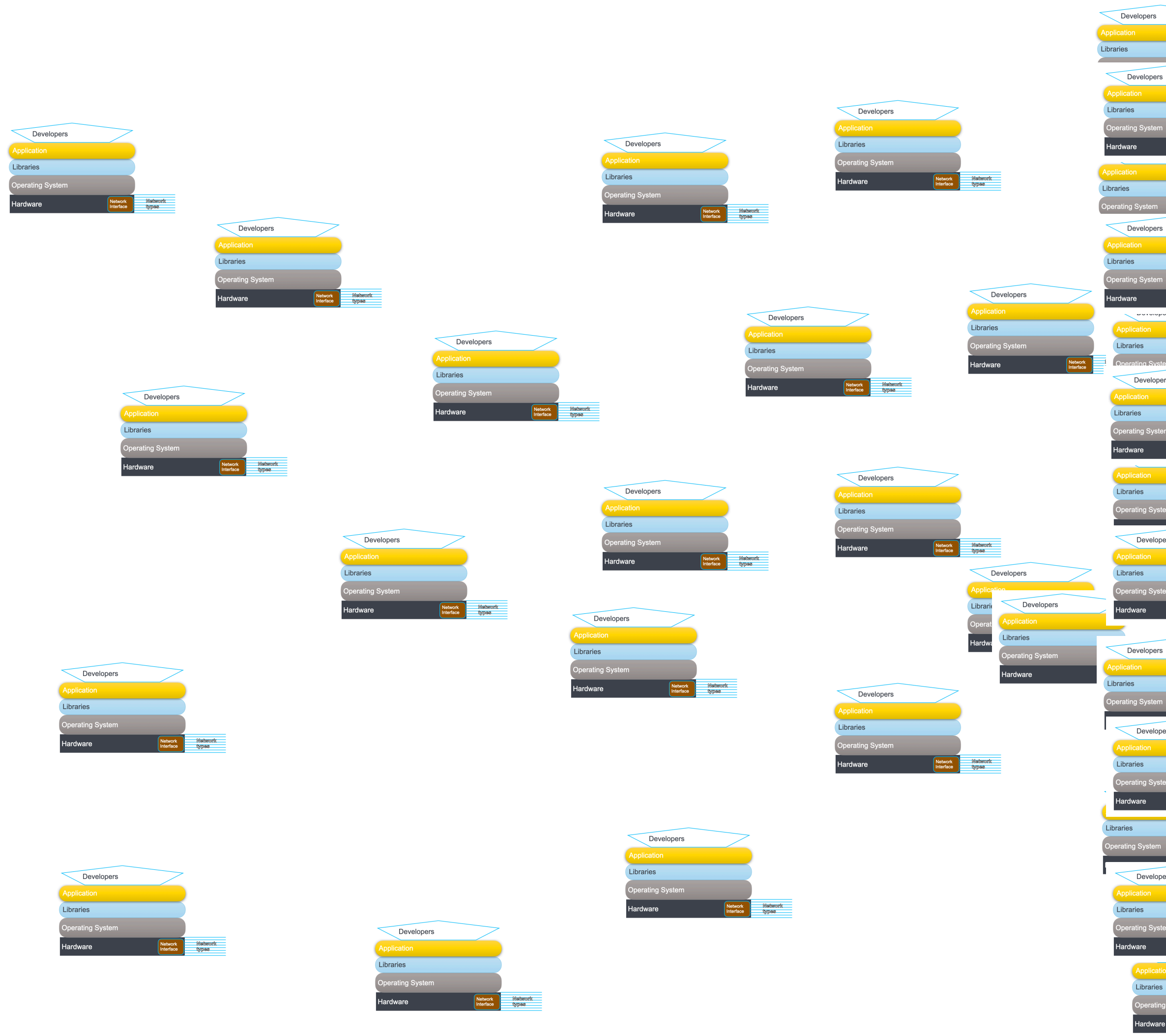
Distributed Systems Challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency



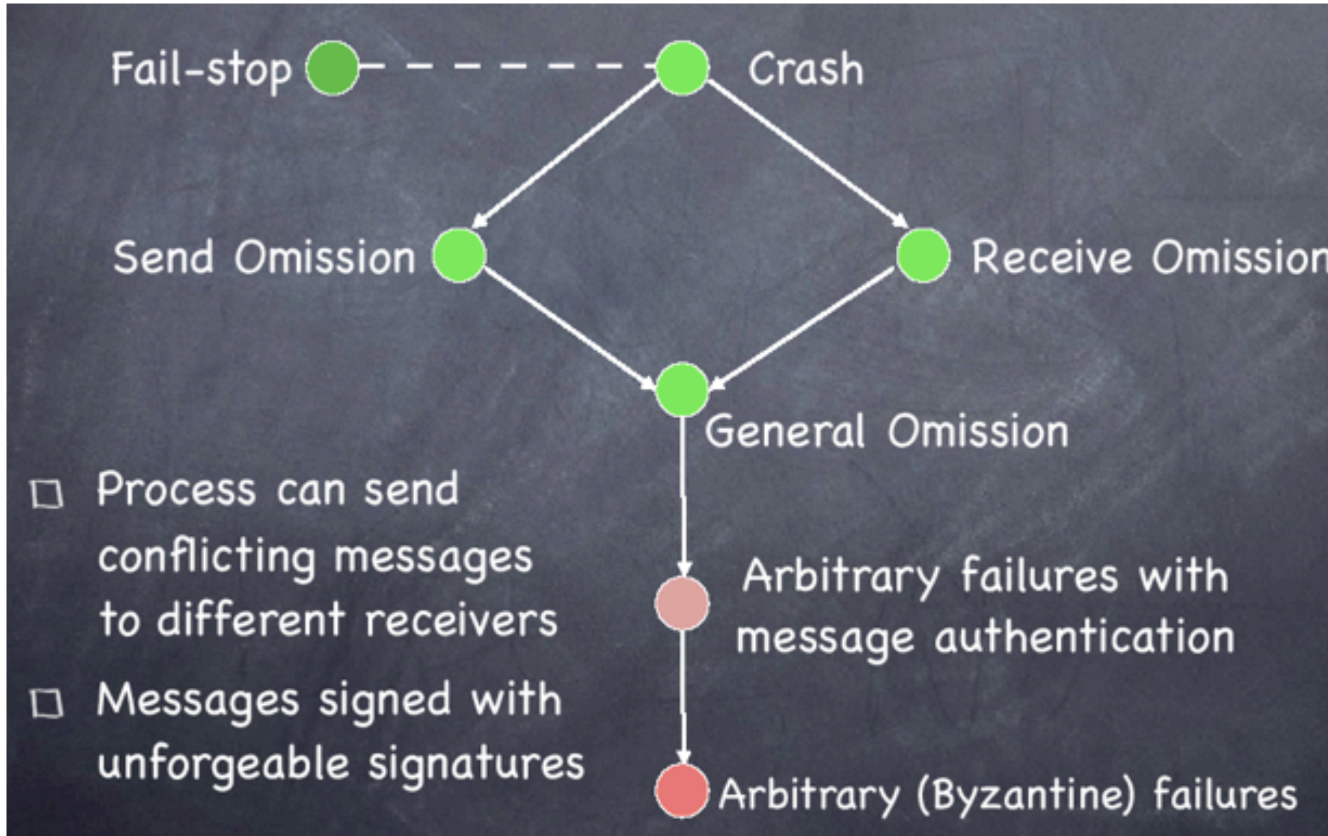
Distributed Systems Challenges

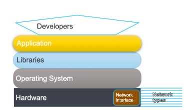
- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency

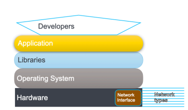


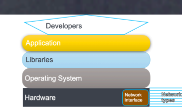
Distributed Systems Challenges

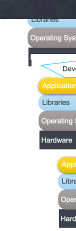
- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency





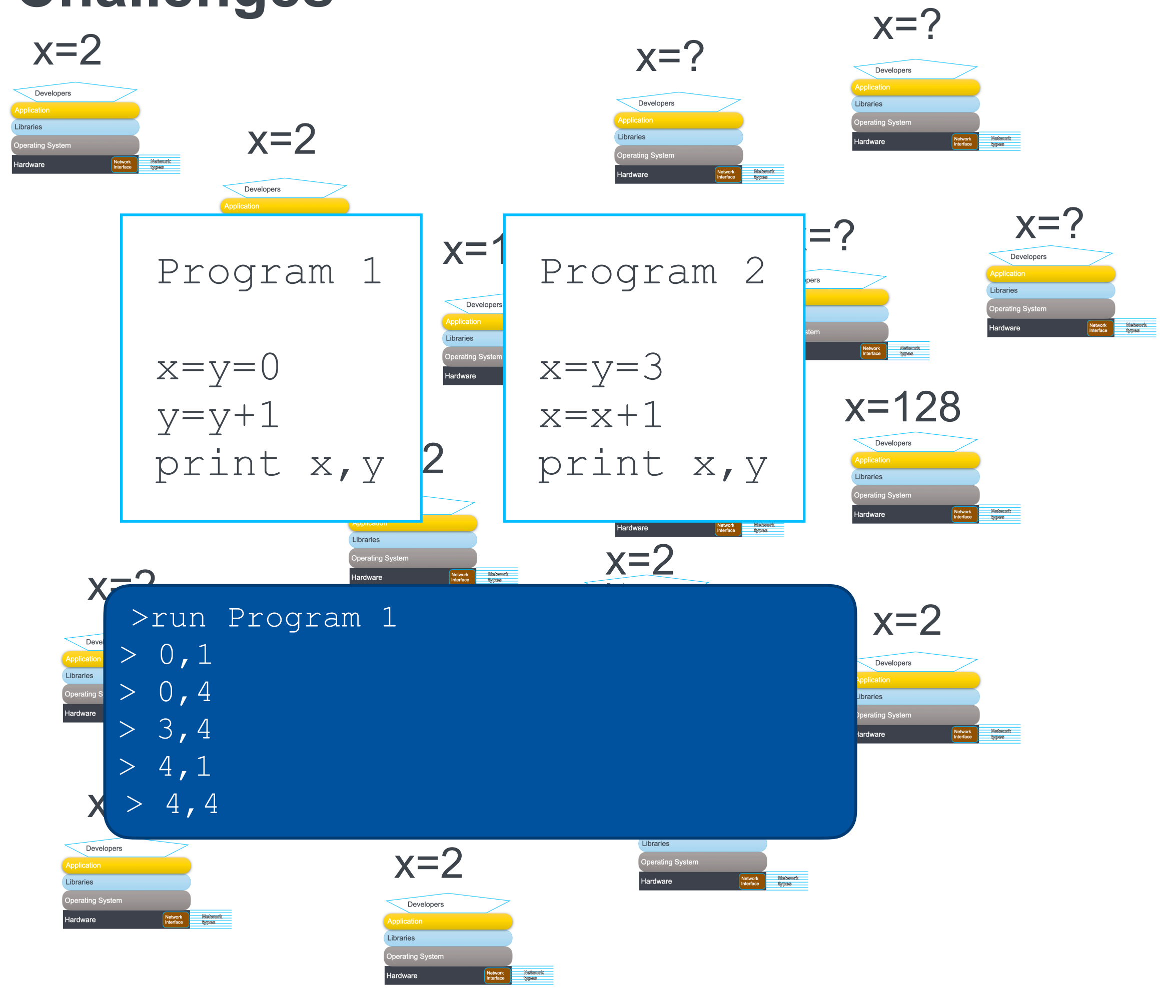


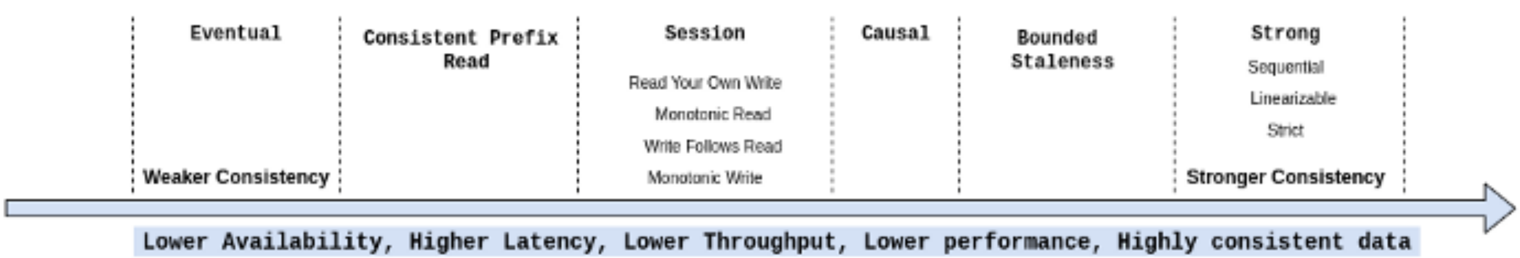




Distributed Systems Challenges

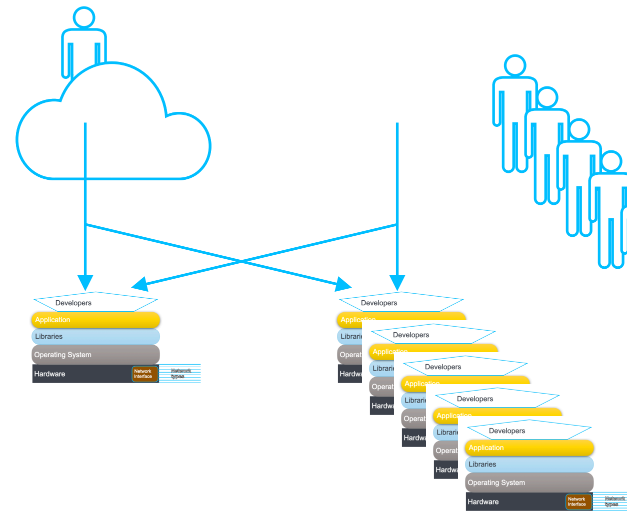
- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency





Distributed Systems Challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Failure handling
- Consistency
- Concurrency
- Transparency



- Access: *same operations*
- Location: *any position*
- Mobility: *dynamic*
- Replication: *any copy*
- Failure: *tolerance*
- Scaling: *any number of nodes*
- …