Distributed Systems
COMP 212

Lectures 1 and 2

Othon Michail

UNIVERSITY OF LIVERPOOL

Course Information

- Lecturer:
  - Othon Michail
  - Office 2.14 Holt Building
- Module Website:
  - VITAL 201920-COMP212-DISTRIBUTED SYSTEMS will be used for all module material, assessment submission and feedback
- Structure
  - 30 Lectures + 10 lab practicals
- Assessment
  - **2 programming assignments** (10% each) and **final exam** (80%)

Recommended Reading

- Main Books:
  - Maarten van Steen, Andrew S. Tanenbaum. Distributed Systems, 3rd edition, 2017 (free digital copy: https://www.distributed-systems.net/index.php/books/ds3/)
  - Andrew S. Tanenbaum, Maarten van Steen. Distributed Systems: Principles and Paradigms, Pearson, 2nd edition, 2006
  - Nancy Lynch. Distributed Algorithms, Morgan Kaufmann Publishers, 1996
  - Hagit Attiya and Jennifer Welch, Distributed Computing Fundamentals, Simulations, and Advanced Topics, 2nd Edition, Wiley-Interscience, 2004

- Java references
  - Thinking in Java
    - https://www.mindviewllc.com/quicklinks/
  - The Java Tutorials (Oracle)
    - https://docs.oracle.com/javase/tutorial/

Other Relevant Books

- George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. Distributed Systems: Concepts and Design, Pearson, 5th Edition, 2011
- Marko Boger. Java in Distributed Systems: Concurrency, Distribution and Persistence, John Wiley & Sons, 1st Edition, 2001

Content of the Course

- Introduction to Distributed Systems

Part I: Distributed Algorithms

- Broadcast (2 lectures)
- Leader Election (3 lectures)
- BFS

Part II: Distributed Systems (and some more algorithms)

- Communication (4 lectures)
- Architectures and Processes
- Naming
- Synchronisation
- Fault tolerance (2 lectures including some algorithms)
- Transactions
- Concurrency and Mutual Exclusion (includes some algorithms)
- Consistency and Replication
- Distribution Protocols
- Security
- Virtualisation and Cloud Computing

Additionally

- Research Lecture

Aims

- Provide an understanding of the technical issues involved in the design of modern distributed systems
- Present some of the major current paradigms

Learning Outcomes

- Appreciation of the main principles underlying distributed systems:
  - processes, communication, naming, synchronisation, consistency, fault tolerance, and security

- Familiarity with some of the main paradigms in distributed systems
- Knowledge and understanding of the essential facts, concepts, principles and theories related to distributed computing
- In depth understanding of the appropriate theory, practices, languages and tools for the specification, design, implementation and evaluation of distributed systems

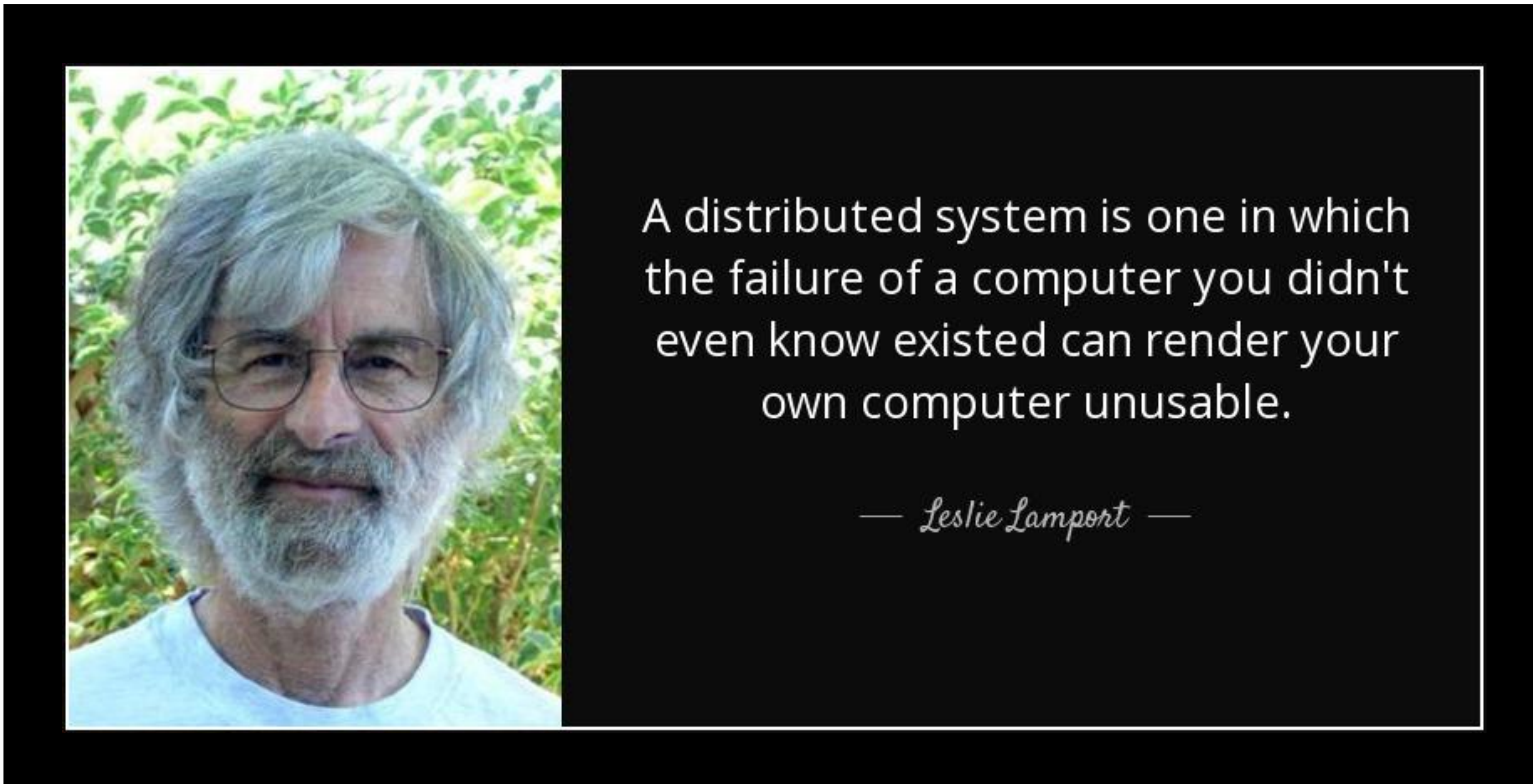
Introduction

What is a Distributed System?

- A distributed system is:

> *A collection of independent computers that appears to its users as a single coherent system*

What is a Distributed System?



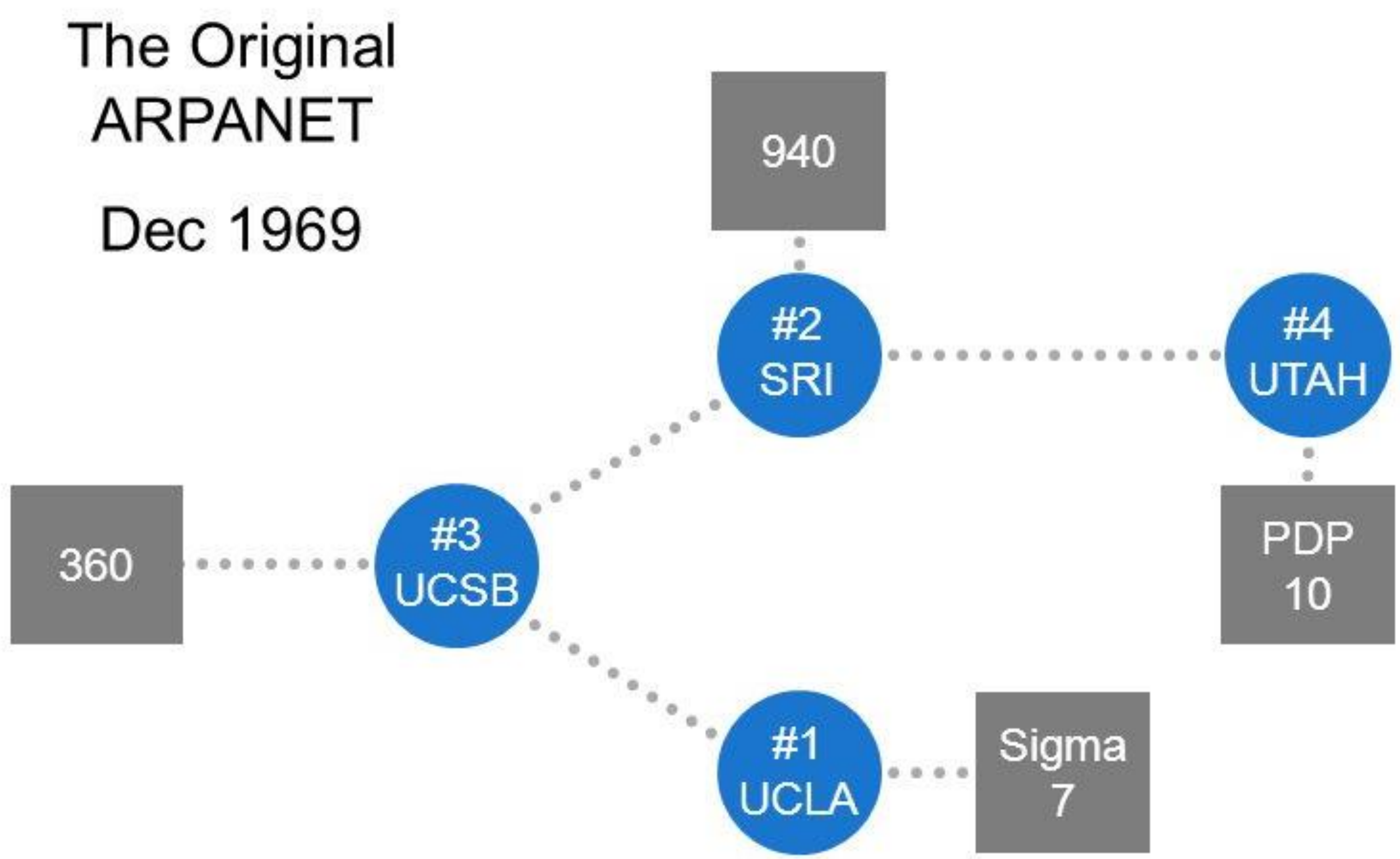
Examples of Distributed Systems

- Can you think of some examples?

Examples of Distributed Systems

- University computer network
- The Internet
- Real-time process control
  - Aircraft/Industrial control systems
- Distributed databases
- Distributed file systems
- Distributed information processing systems
  - Banks (Cash machines)
  - Ticket reservation systems
- Parallel computing (cluster computing, multiprocessor systems)
- Cloud computing

The Beginning of the Internet (ARPANET)



- 4 interconnected computers forming a simple packet switching network
- First to implement TCP/IP

The Internet

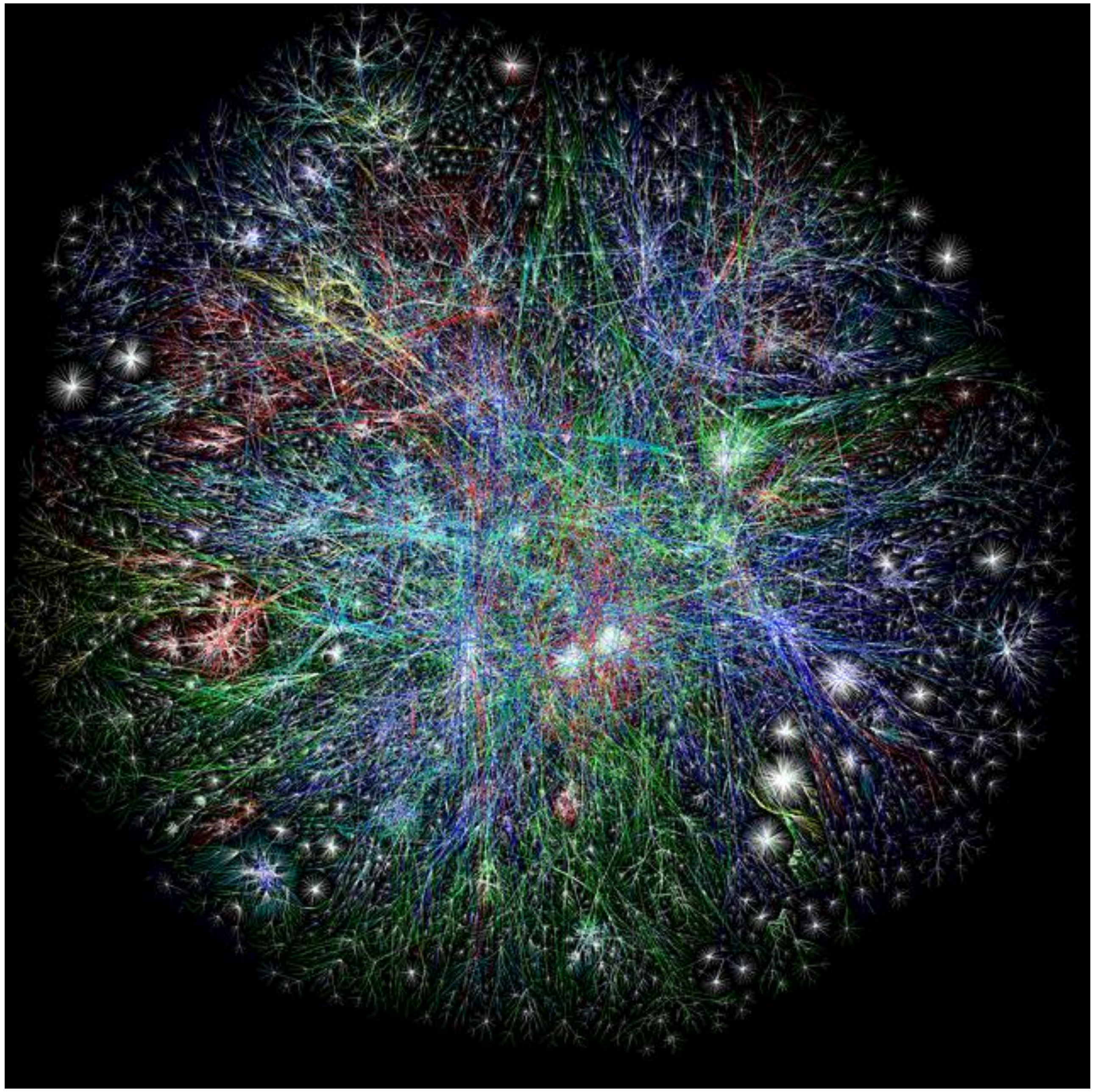
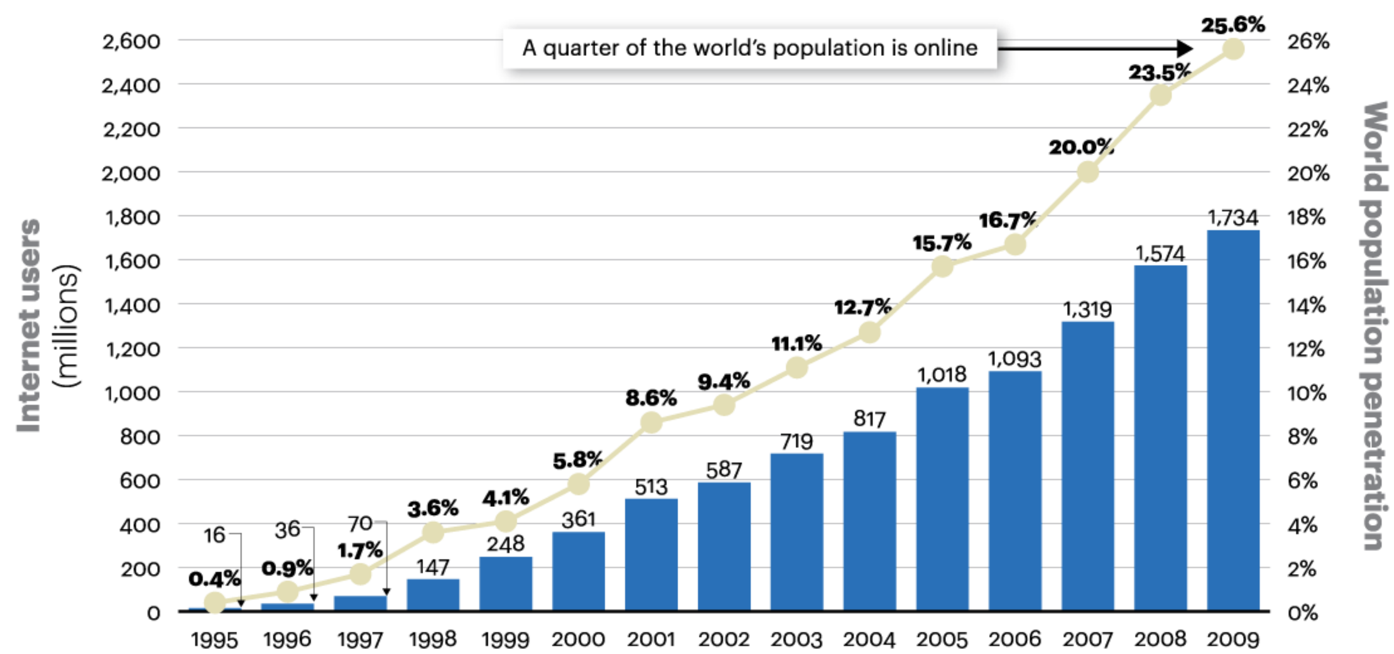
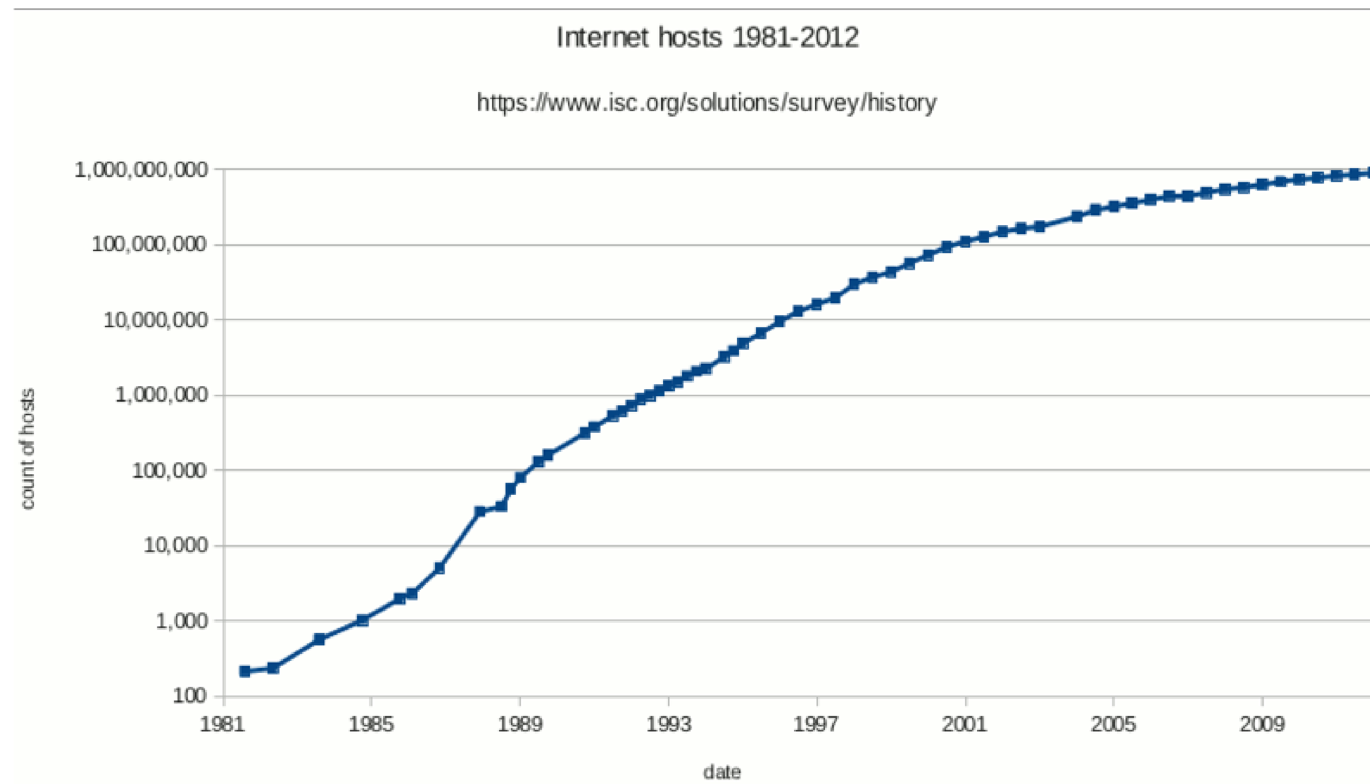


Figure 1
Global Internet users and penetration rate (1995-2009)

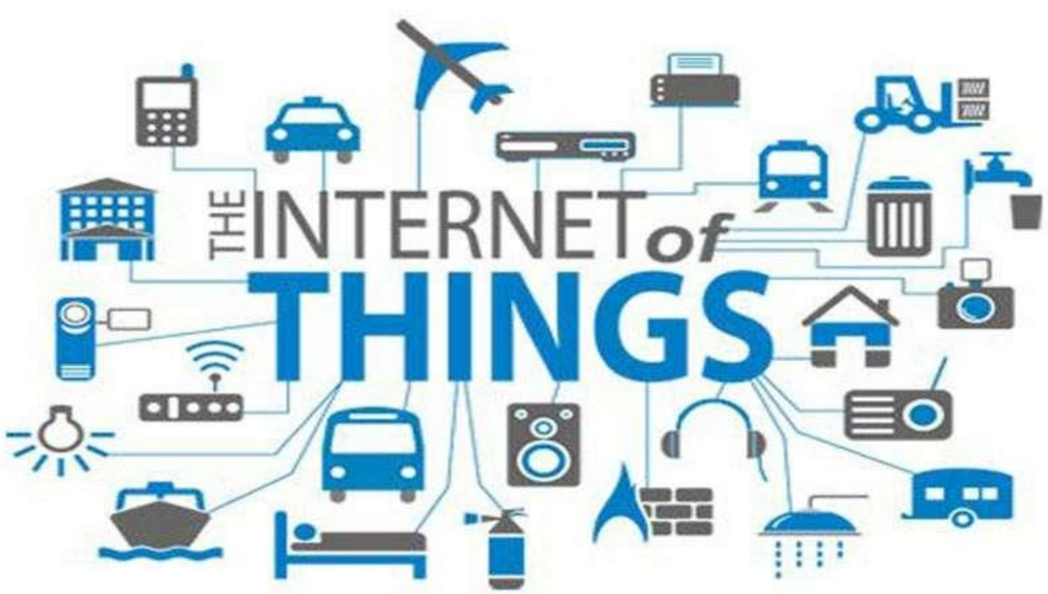


Sources: Nielsen, ITU; A.T. Kearney analysis

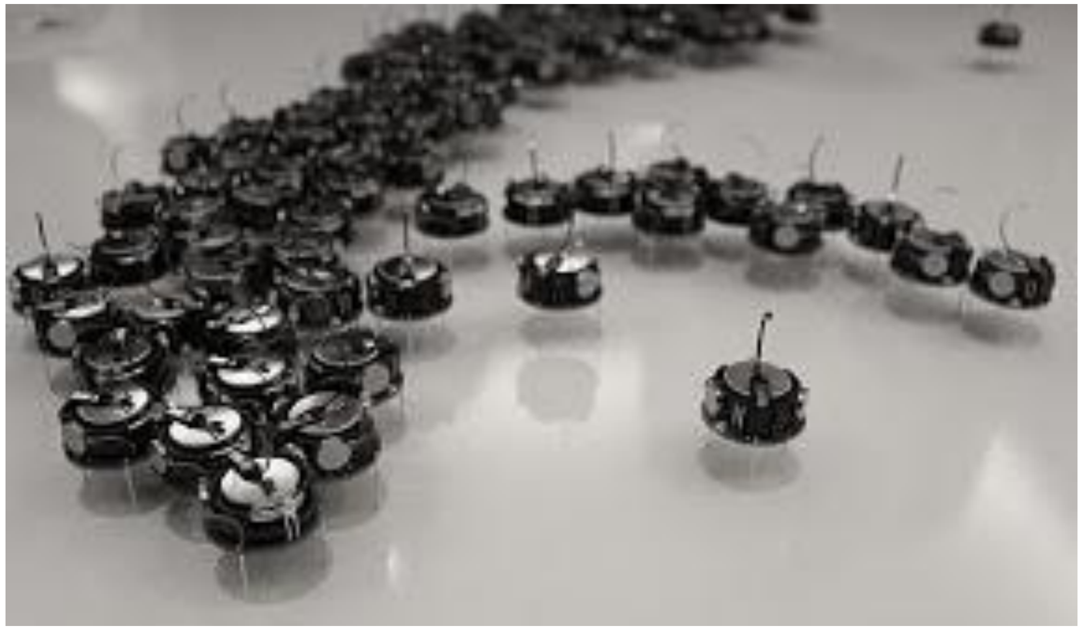


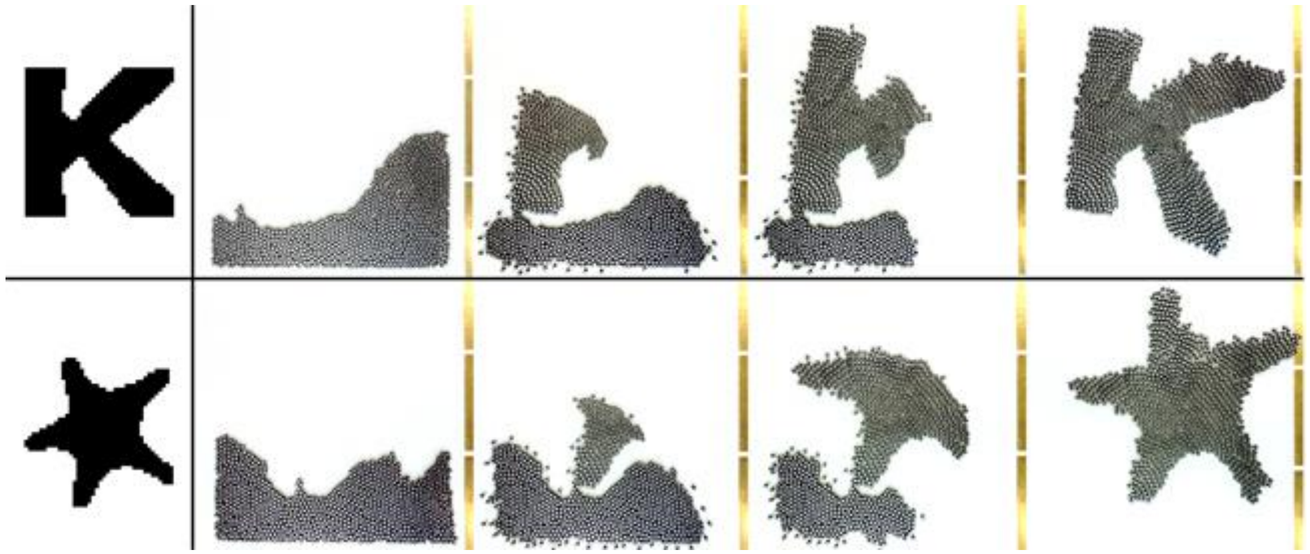
- Over 1.5 billion websites
- Over 4 billion users
- Check the rate of growth at: http://www.internetlivestats.com/

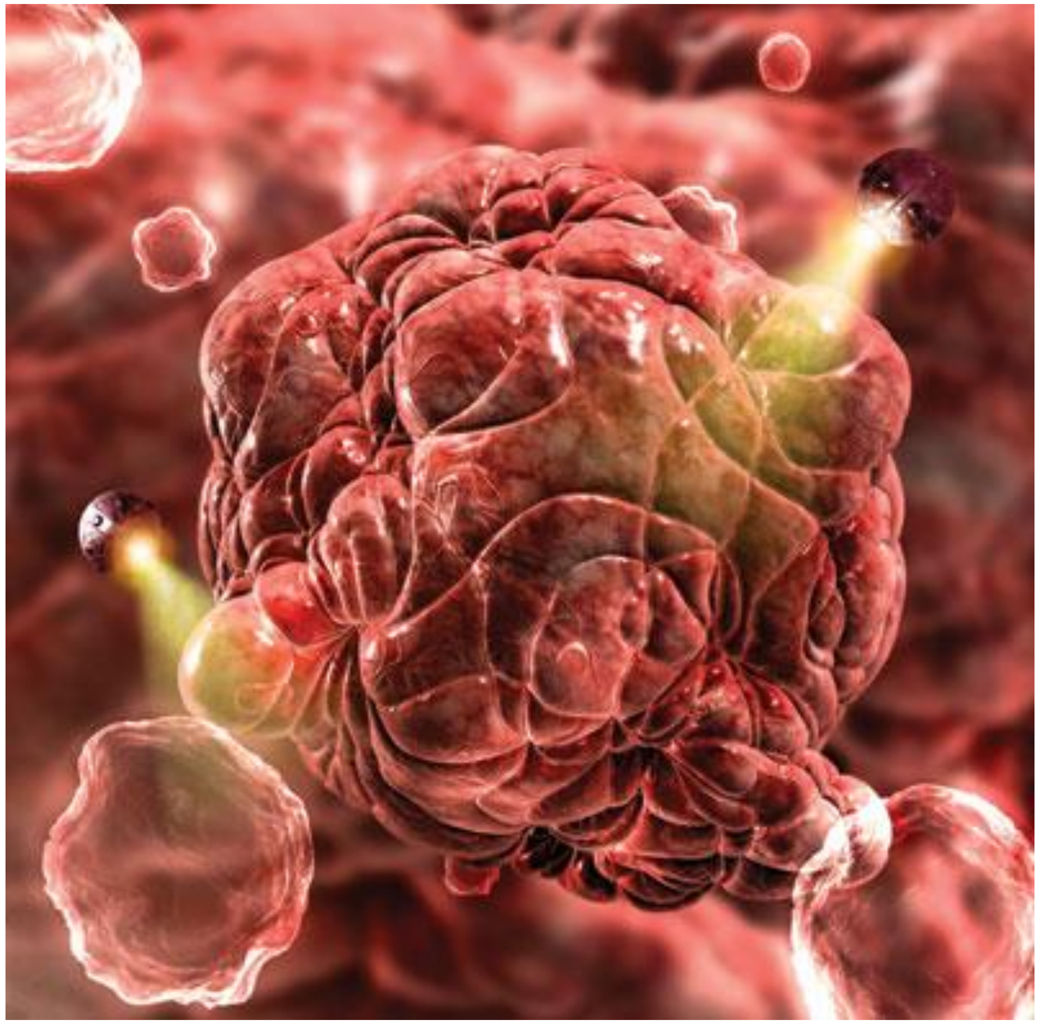
Internet-enabled Devices

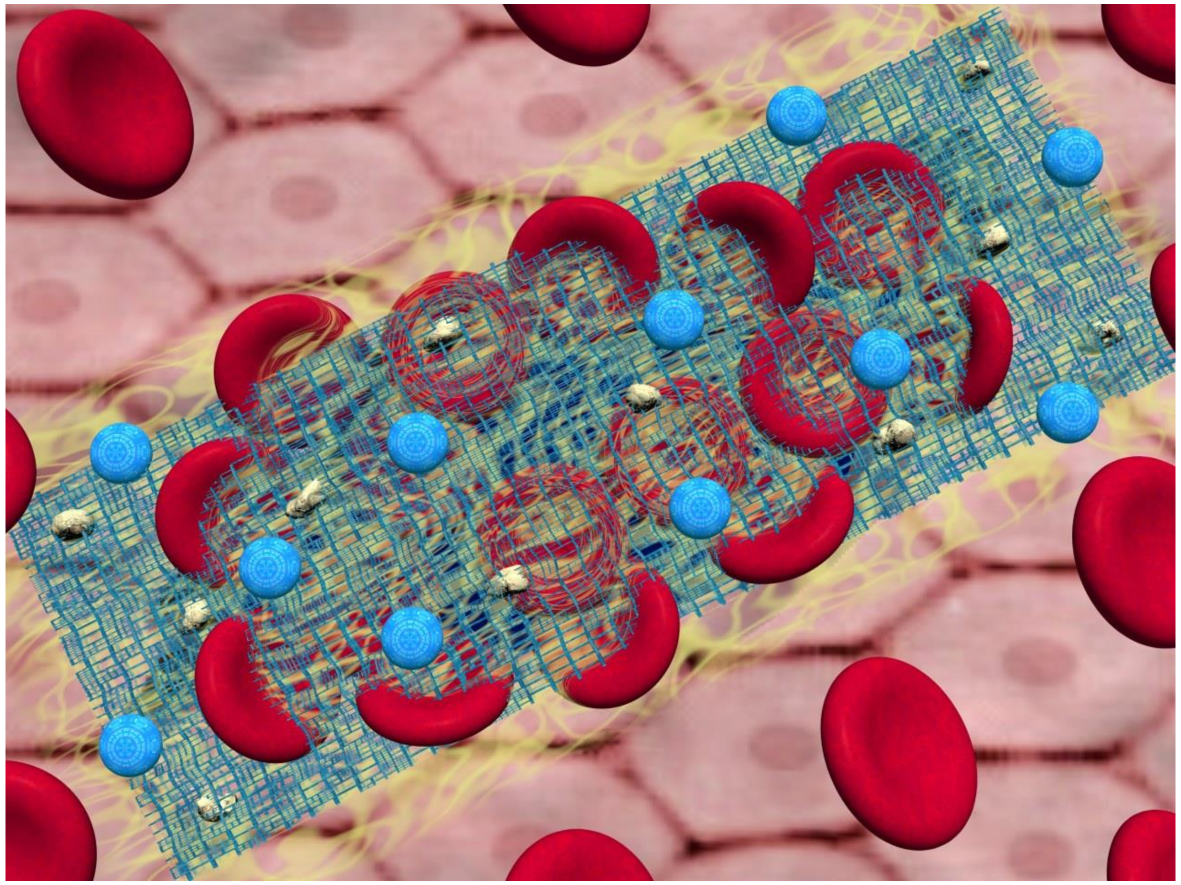


Nanorobotics and Self-Assembly









Why Distributed Systems?

- Pervade society
  - Billions of computers and mobile devices
  - Continuous advances in communication technology
  - Most crucial modern information systems are distributed
- Turing Award 2013 was awarded to Leslie Lamport for his work on Distributed Systems

Goals of Distributed Systems

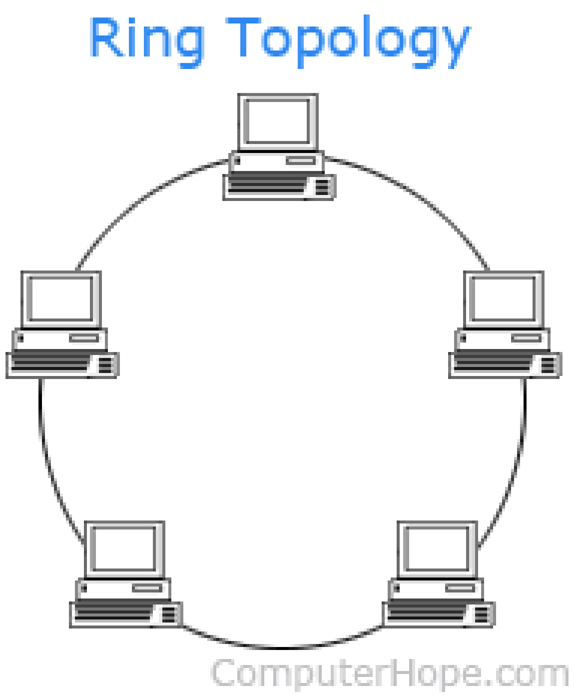
- Easily connect users/resources
- Transparency
- Openness
- Reliability
- Performance
- Scalability

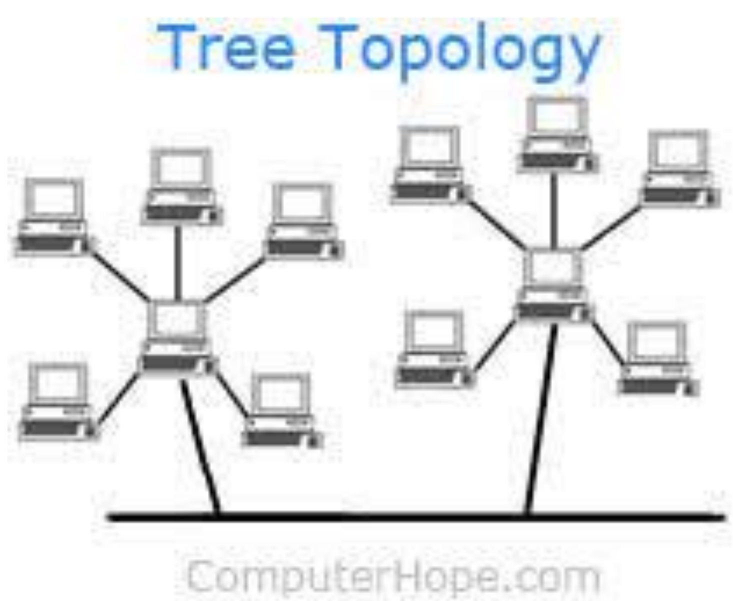
Beyond Systems

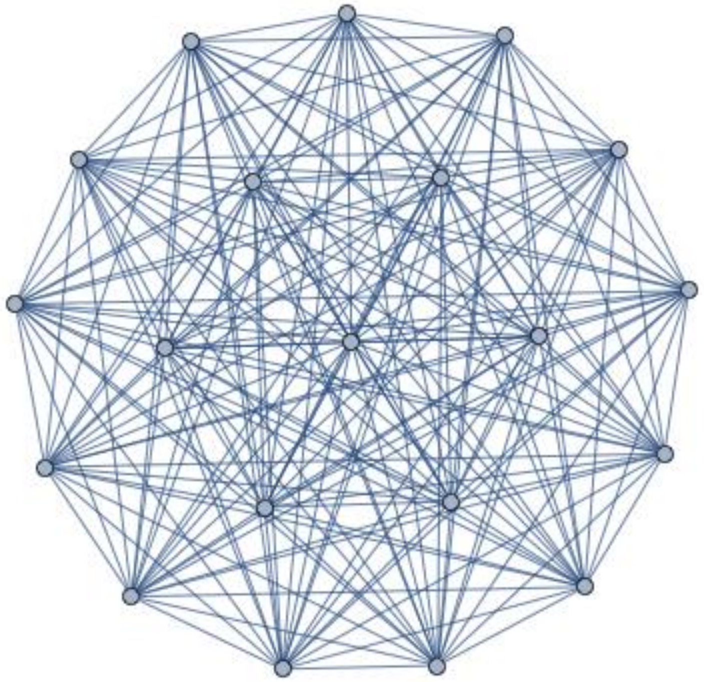
- Distributed systems also require a rigorous theory to support them
- Theory of Distributed Computing
  - The distributed counterpart of the theory of (centralised) computation and of sequential algorithms
- Why?
  - Theory can provide insights into existing systems
  - Allows to argue about *any* potential system
- For example,
  - Prove what is in principle feasible in a given setting or
  - Prove impossibility results for a given setting

A Formal Model

- n processors
- Network: Directed graph $G=(V, E)$
  - $n = |V|$
- Examples: ring, tree, complete network
- Distributed algorithms may assume a specific topology or not







Processes

- Executed in processors
- A process u :
  - $states_u$ (set of states)
    - initial states $start_u$ and halting states $halt_u$ subsets of $states_u$
  - $msgs_u : states_u \times out\text{-}nbrs_u \rightarrow M \cup \{\text{null}\}$ (message-generation function)
  - $trans_u : states_u \times \{M \cup \{\text{null}\}\}^{|in\text{-}nbrs_u|} \rightarrow states_u$ (state-transition function)

Synchronous Networks

- All processes initially in initial state(s)
- All processes repeat synchronously (in lock-step):

Step 1

1. Apply message-generation function
2. Messages produced for out-neighbours
3. Messages transmitted through corresponding channels

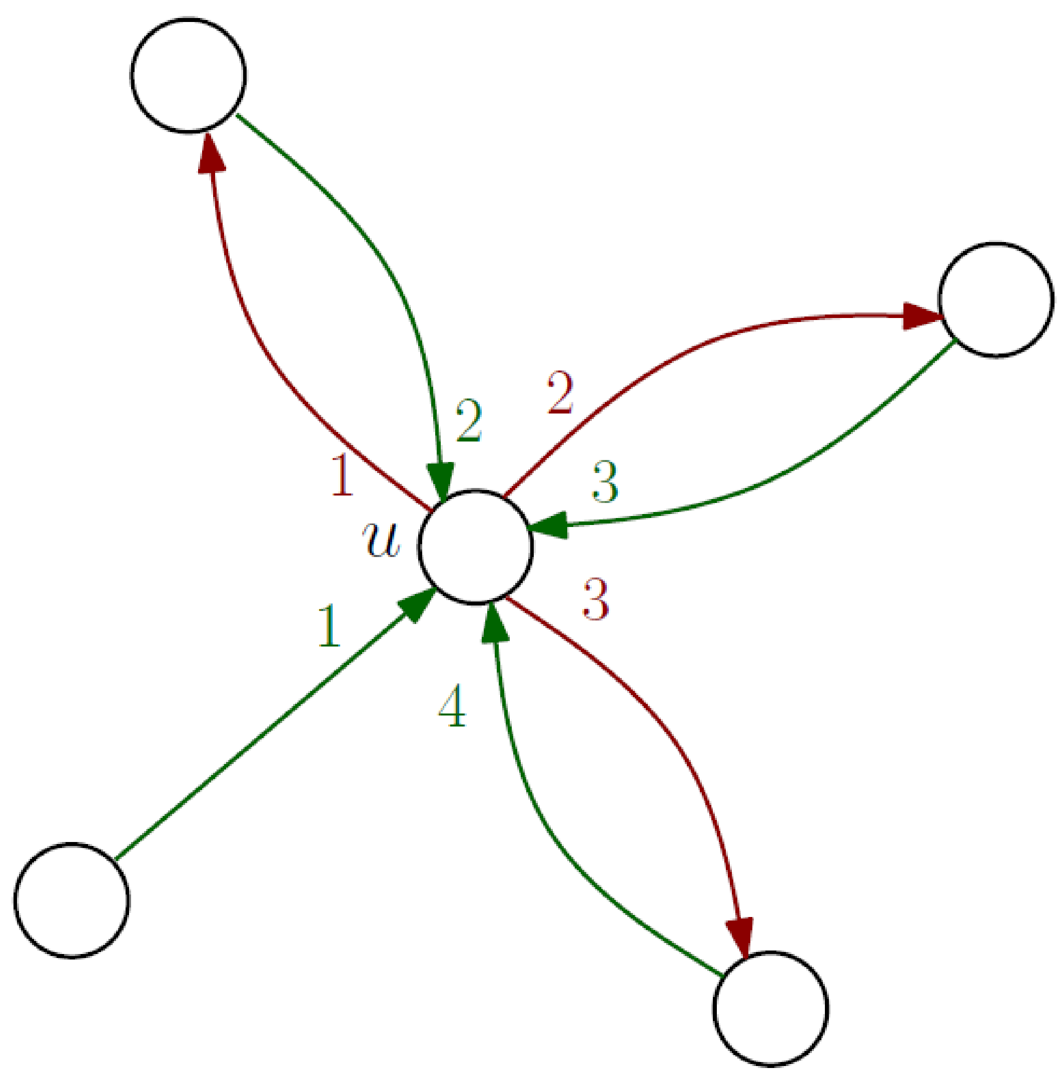
Step 2

1. Apply state-transition function (taking into account any received messages)
2. Remove all messages from the channels

Step 1 + Step 2 = 1 Round

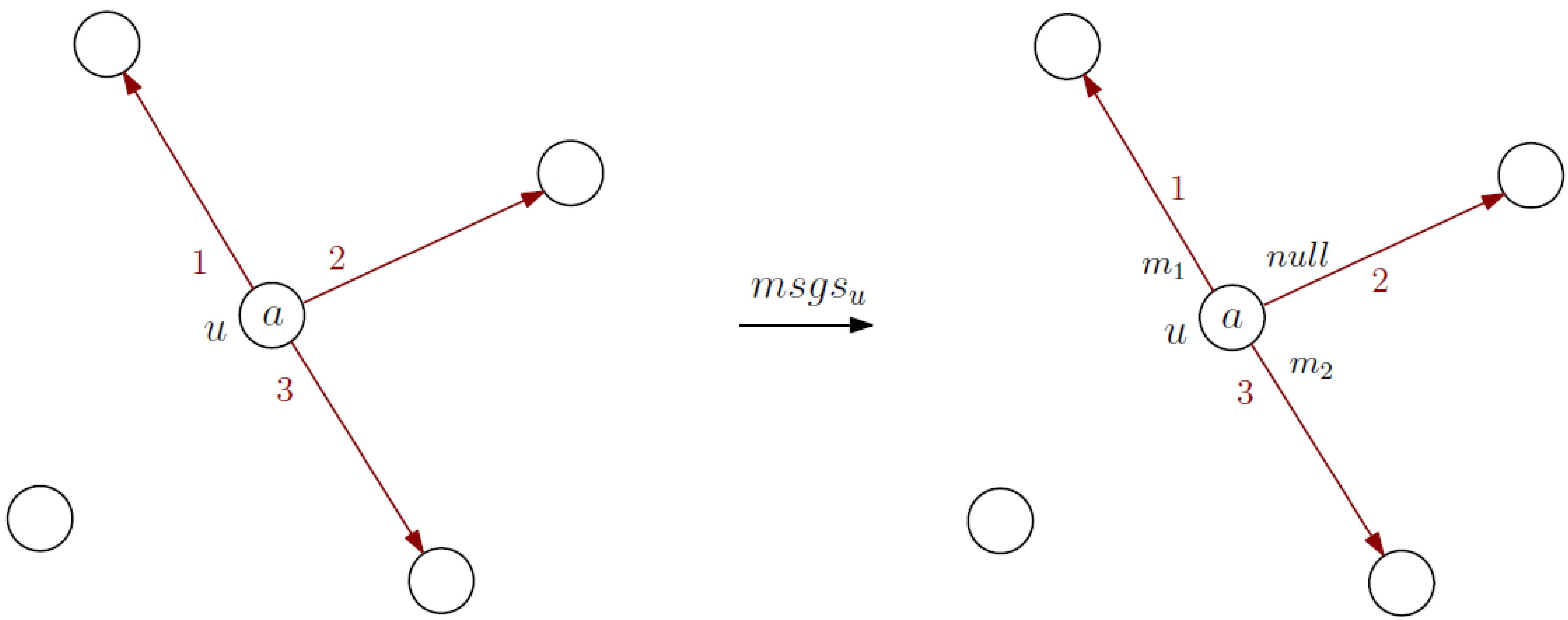
An Illustration

- $states_u = \{a, b, c\}$, $M = \{m_1, m_2\}$ (also null)
- $msgs_u(a, 1) = m_1$, $msgs_u(a, 2) = null$, $msgs_u(a, 3) = m_2$
- $trans_u(a, (m_1, m_2, \text{null}, m_2)) = b$



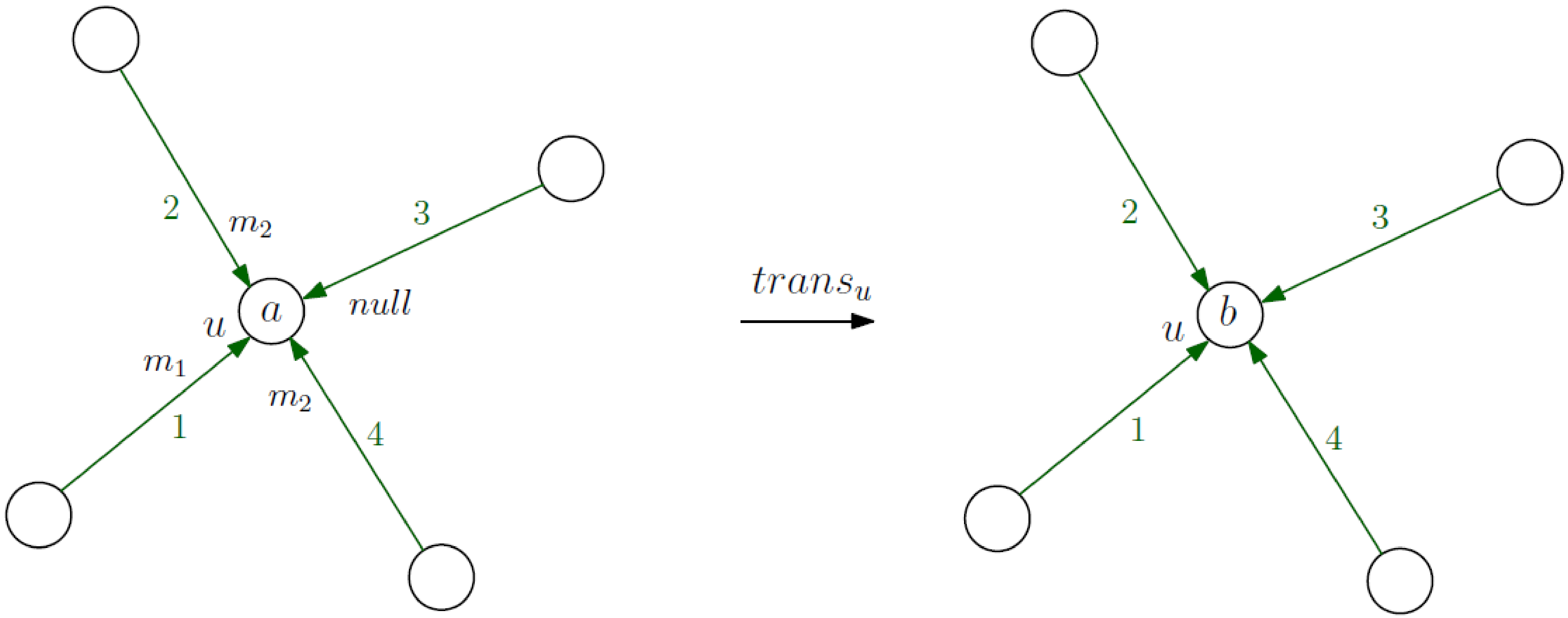
An Illustration

- $states_u$ = {a, b, c }, M = {m_1, m_2} (also null)
- $msgs_u(a, 1) = m_1$, $msgs_u(a, 2) = null$, $msgs_u(a, 3) = m_2$



An Illustration

- *states*$_u$ = {*a*, *b*, *c* }, *M* = {m_1, m_2} (also null)
- *trans*$_u$ (*a*, (m_1, m_2, null, m_2)) = *b*

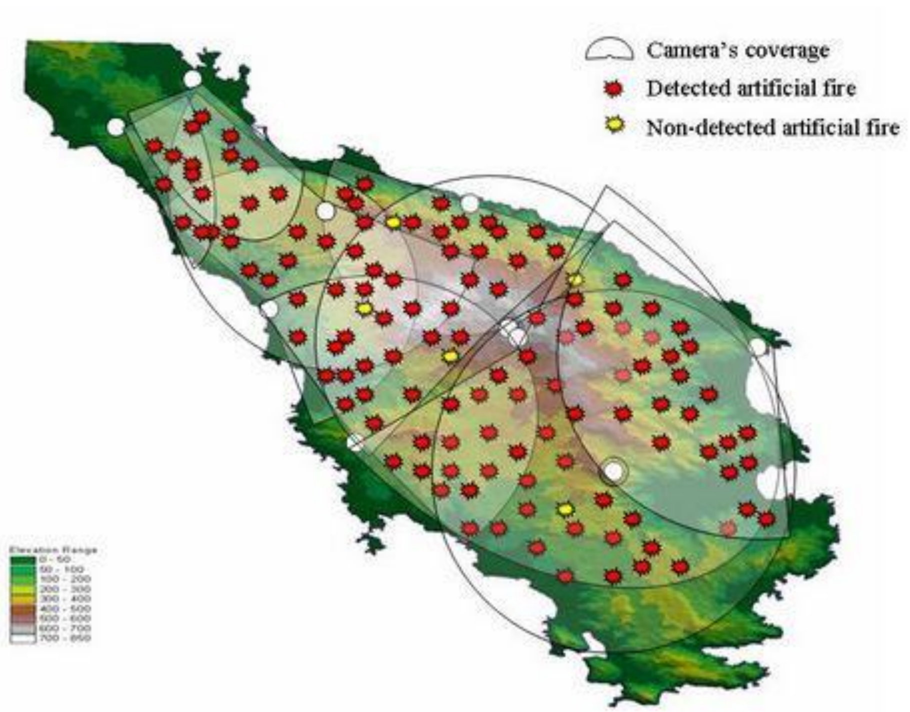


A Simple Example: Flooding

- Initially: A leader-node has a token (piece of info)
- Goal: All nodes must learn the token
- Distributed Algorithm: Ideas???

Example application:

- Wireless Sensor Network
- A node detects fire
- All nodes must be informed about the location to act accordingly



A Simple Example: Flooding

- Initially: A leader-node has a token (piece of info)
- Goal: All nodes must learn the token
- Distributed Algorithm: Ideas???

- Formally, we must define: M, $states_u$, $msgs_u$ and $trans_u$
- Algorithmic Idea (informal):
  - *All nodes awake initially*
  - *If in the beginning of this round you are awake and you have the token,*
    - *forward it to all your neighbours and*
    - sleep
  - *If you are asleep, do nothing*