Distributed Systems

Rik Sarkar

University of Edinburgh
Spring 2020

Course Information

- Instructors:
  - Rik Sarkar (IF 3.45, rsarkar@inf.ed.ac.uk)
  - Hugh Leather (IF 1.18A, hleather@inf.ed.ac.uk)

- Web site: http://www.inf.ed.ac.uk/teaching/courses/ds

- Lectures:
  - Tuesdays 11:10-13:00
    - Hunter building Lecture Theatre O17
  - With short break in the middle.

Exams and Assignments

- Grading:
  - Coursework: 1 assignment, 25%
    - Design a distributed algorithm for a given problem
    - Implement the essential idea in a simulation
    - Program in Java
    - Given out around mid Feb. Due around mid March
  - Final Exam: 75%

- Coursework
  - Marked on:
    - Description and analysis of solution design
    - Implementation (Practical, in Java)
  - Requirements:
    - Java, general understanding of working in UNIX/LINUX environment
    - Basic understanding of algorithm design, data structures, probability

Course background & expectations

- Understanding of
  - Algorithm design
  - Data structures
  - Graph theory (DFS, BFS, MST, cycles, paths, shortest paths) and relevant algorithms
  - Asymptotic notations and complexity
    - Big O, Ω, Θ
  - Basic probability, expectation
  - (Optionally) Networks and communication

Reading & Books

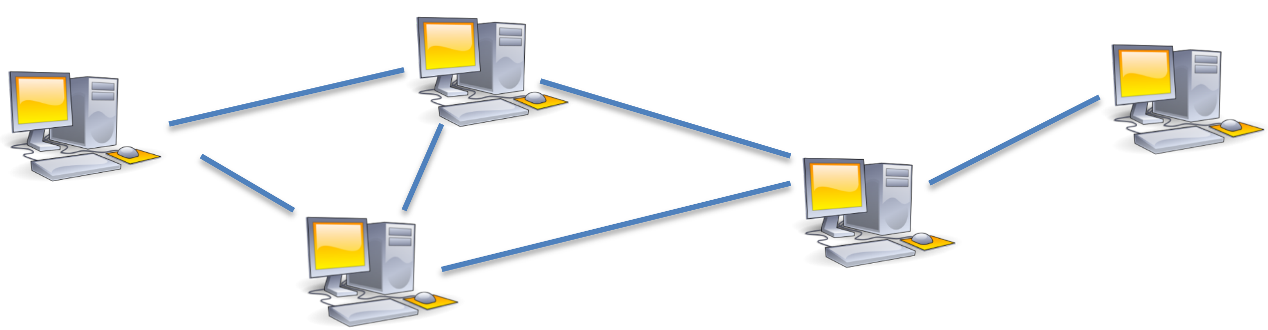
- **No *required* textbook**

- Suggested references:
  - [CDK] Coulouris, Dollimore, Kindberg; Distributed Systems: Concepts and Design
    - 4th Edition: http://www.cdk4.net/wo
    - 5th Edition: http://www.cdk4.net/wo
  - [VG] Vijay Garg; Elements of Distributed Computing
  - [NL] Nancy Lynch; Distributed Algorithms
- We will use these abbreviations to suggest suitable books for topics.
- Most things can found wikipedia/web

What is a distributed system?

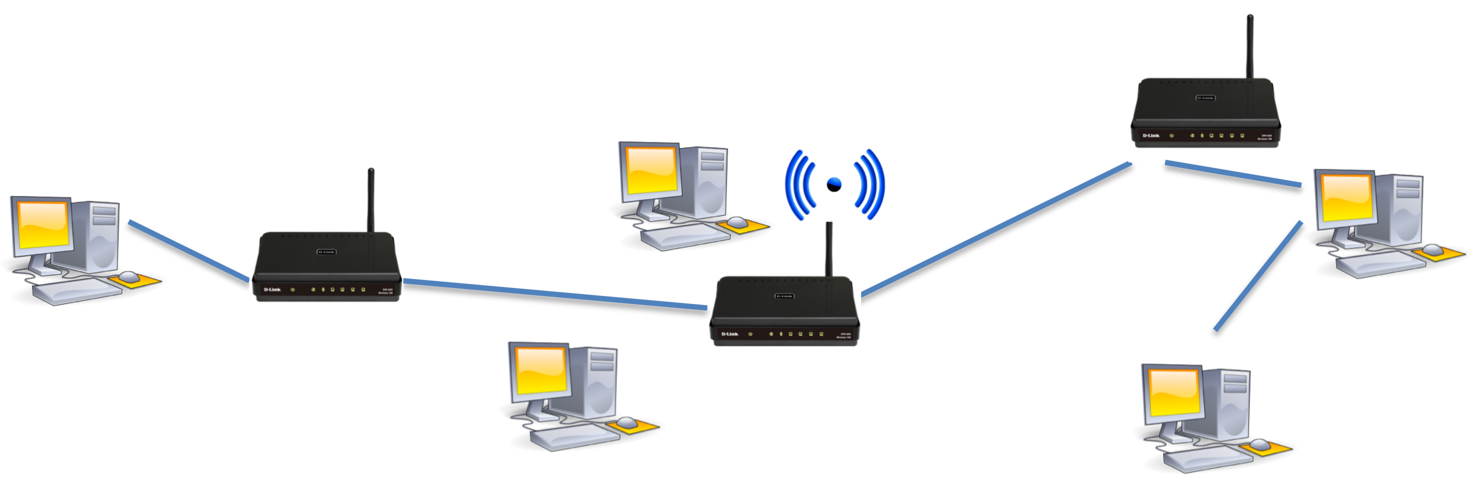
What is a distributed system?

- Multiple computers working together on one task
- Computers are connected by a network, and exchange information



What is a distributed system?

- Multiple computers working together on one task
- Computers are connected by a network, and exchange information



Networks Vs Distributed Systems

Networks Vs Distributed Systems

Computation
Using many computers
Sending messages to
Each-other

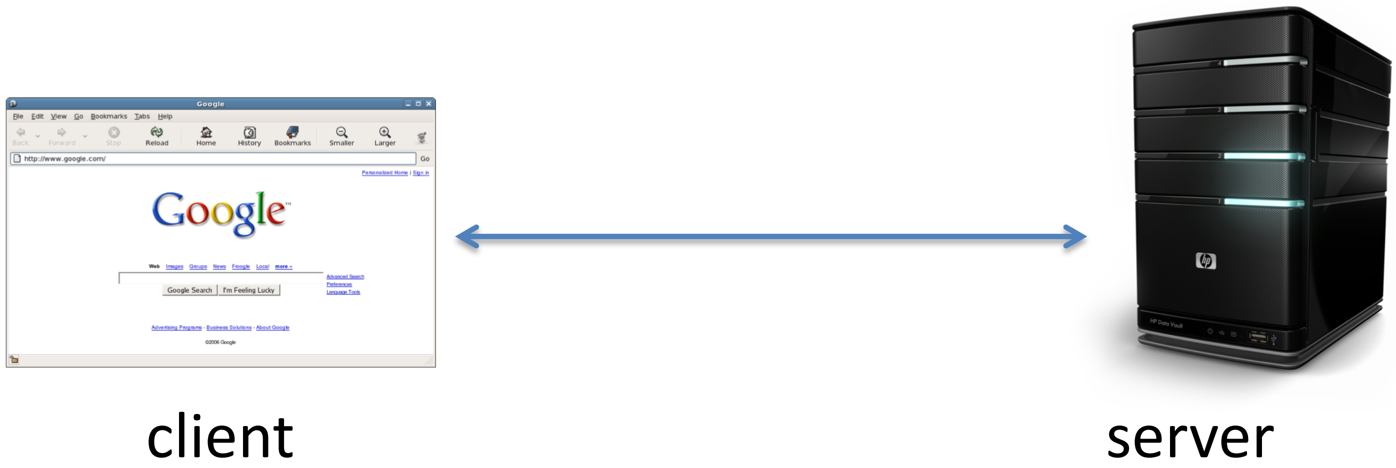
data transport
routing
medium access

Distributed Systems: how to write programs that use the network to make use of multiple computers

Networks: How to send messages from one computer to another

Distributed Systems: Examples

- **Web browsing:**



- In this case:
  - Client requests what is needed
  - Server computes and decides what is to be shown
  - Client shows information to user

Distributed Systems: Examples

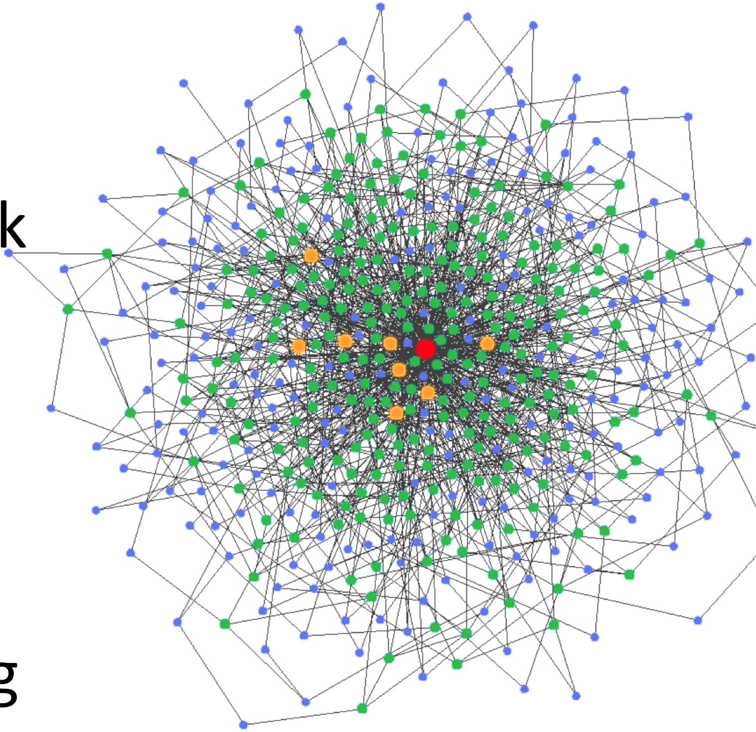
- **Multiplayer Games**
  - Different players are doing different things
  - Their actions must be *consistent*
    - Don’t allow one person to be at different locations in views of different people
    - Don’t let two people stand at the same spot
    - If X shoots Y, then everyone must know that Y is dead
  - Made difficult by the fact that players are on different computers
  - Sometimes network may be slow
  - Sometimes messages can be lost

Distributed Systems: Examples

- **Stock markets: Multiplayer games with High stakes!**
- Everyone wants information quickly and to buy/sell without delay
- Updates must be sent to many clients *fast*
- Transactions must be executed in right order
- Specialized networks worth millions are installed to reduce latency

Distributed Systems: Examples

- **Hadoop**
  - A big data processing framework
  - *Mapper* nodes partition data, *reducer* nodes process data by partitions
  - User decides partitioning, and processing of each partition
  - Hadoop handles tasks of moving data from node to node
  - Hadoop/mapreduce is a specific setup for distributed processing of data

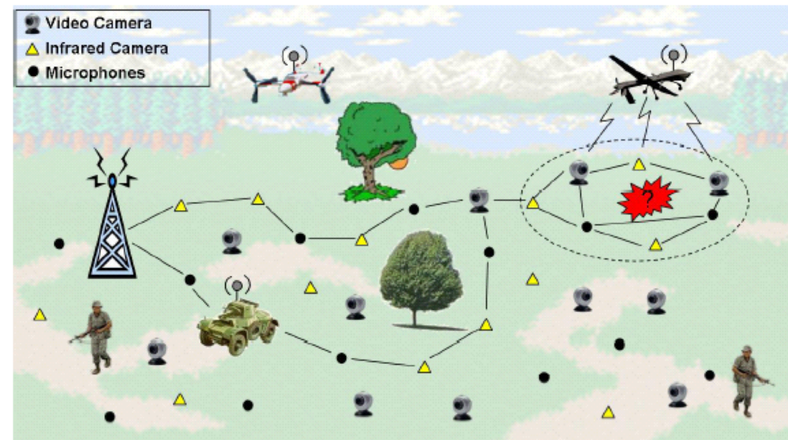


Distributed Systems: Examples

- **Main issue in networking: one node does not have complete (global) knowledge of the rest of the network**
  - Need *distributed* solutions – network protocols
  - Nodes work with local information

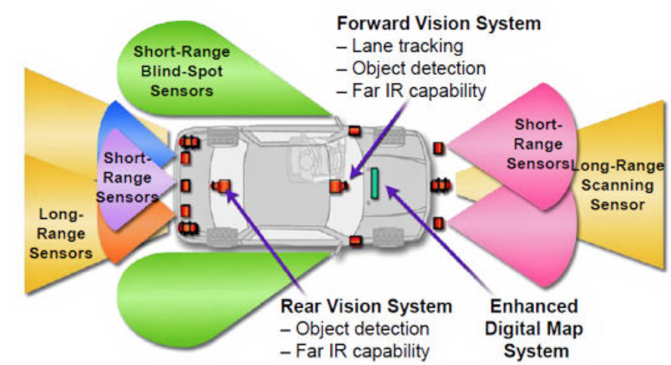
Distributed Systems: Examples

- **Mobile and Sensor Systems**
  - Mobile phones and smart sensors are computers
  - Opportunity to process data at sensors instead of servers
  - Distributed networked operation
  - In addition, nodes are low powered, battery operated
  - Nodes may move
- **Ubiquitous computing & Internet of things**
  - Embedded computers are everywhere in the environment
  - We can use them to process data available to them through sensors, actions of users, etc.
  - Networking and distributed computing everywhere in the environment



Distributed Systems: Examples

- **Autonomous vehicles**
  - Computer operated vehicles, will use sensors to map the environment and navigate
  - Sensors in the car, in the environment, other cars
  - Need to communicate and analyze data to make quick decisions
  - Many sensors and lots of data
  - Strict consistency rules – two cars cannot be at the same spot at the same time!
  - Need very fast information processing
  - Nodes are mobile



Challenges in Distributed Computing

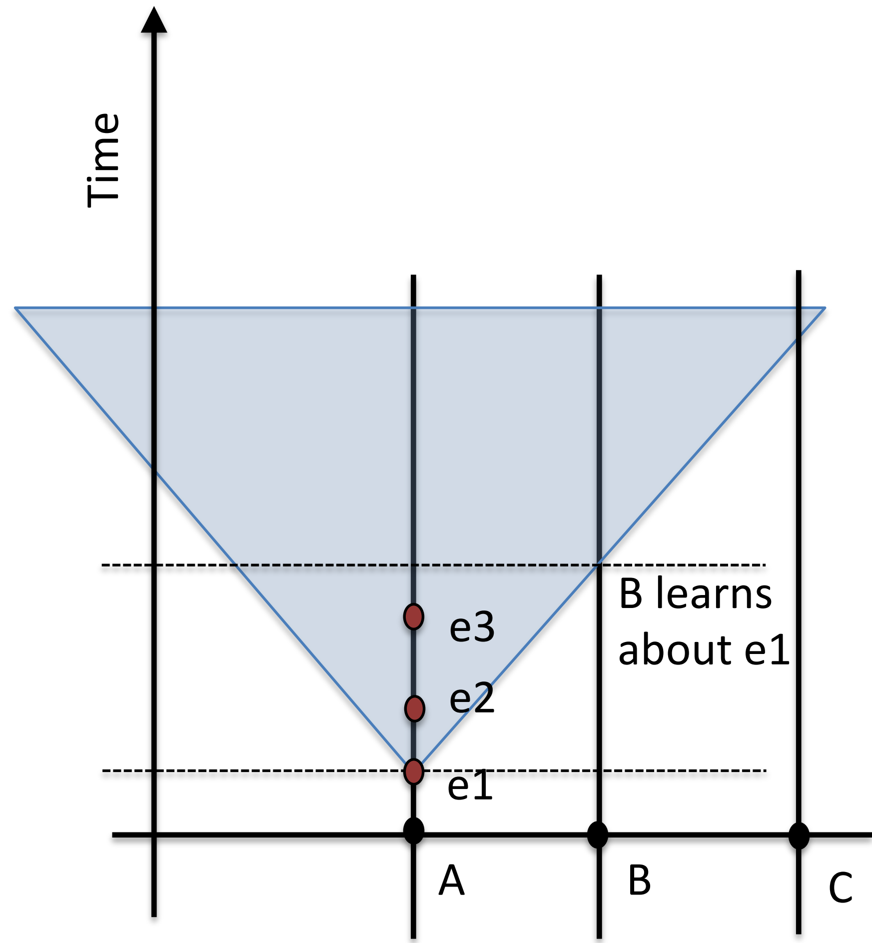
- Fundamental issue: Different nodes have different knowledge. One node does know the status of other nodes in the network

- If each node knew exactly the status at all other nodes in the network, computing would be easy.

- But this is impossible, theoretically and practically

Theoretical issue: Knowledge cannot be perfectly up to date

- Information transmission is bounded by speed of light (plus hardware and software limitations of the nodes & network)
- New things can happen while information is traveling from node A to node B
- B can never be perfectly up to date about the status of A



Practical challenges in distributed systems

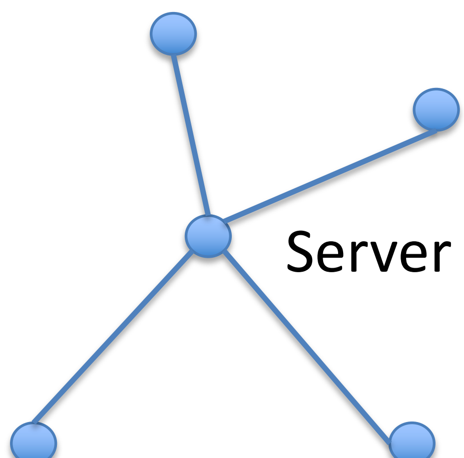
- Communication is costly: It is not practical to transmit everything from A to B all the time
- There are many nodes: Transmitting updates to all nodes and receiving updates from all nodes are even more impractical

- The critical question in distributed systems:

- What message/information to send to which nodes, and when?

Example 1

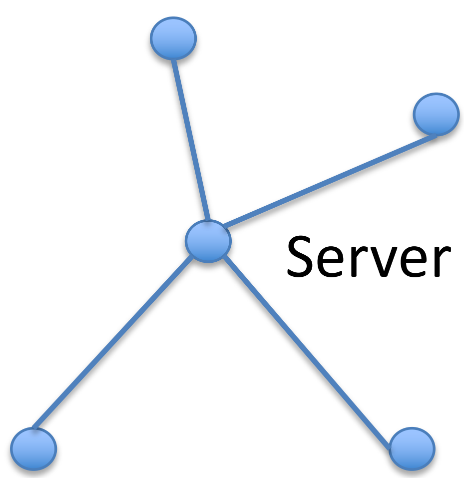
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of these numbers



How many messages does it take?

Example 1

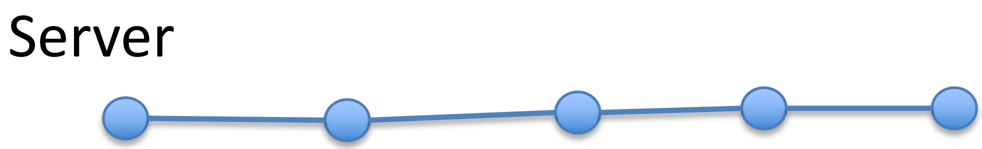
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of these numbers



4

Example 2

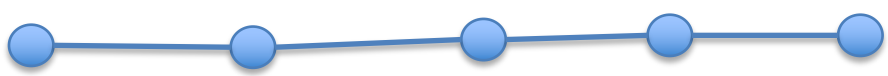
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of the numbers



How many messages does it take?

Example 2

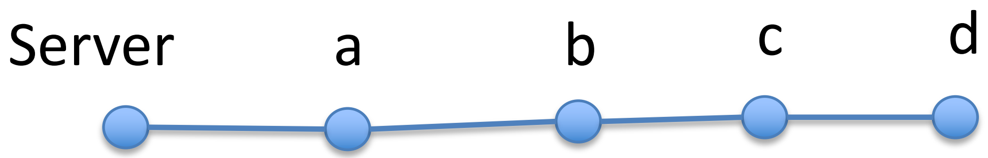
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of the numbers



- Complexity may depend on the Network

Example 2

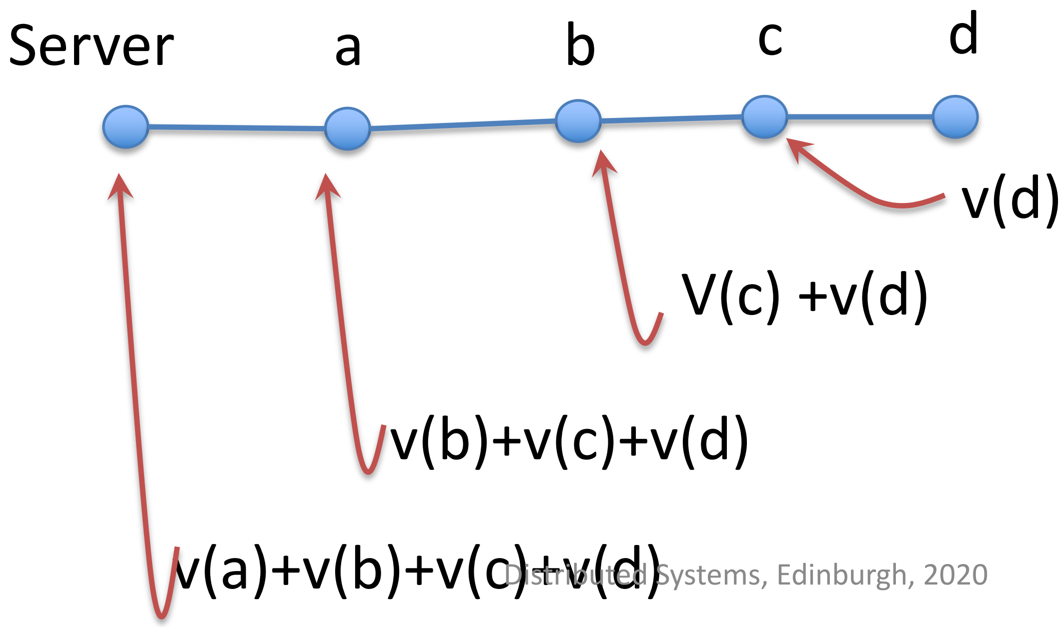
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of the numbers



Can you find a better, more efficient way?

Example 2

- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of the numbers



Cost: 4 messages

Practical challenges in distributed systems

- Time cannot be measured perfectly
  - Clocks always move slightly faster/slower; speeds change
  - Hard to compare before/after relations between events at different nodes
  - Makes it difficult to keep causal relations correct
  - E.g. In a multi-player game, two players fired their guns. Who shot first?

Practical challenges in distributed systems

- Failures
  - Some nodes may fail
  - Some communication links may fail, messages get lost

  - We need systems *resilient* to failures – it should continue to work even if some nodes/links fail, or at least recover from failures
  - E.g. In network routing, if some nodes fail, the routing protocols find new paths to the destination

Practical challenges in distributed systems

- Mobility
  - Some nodes may be mobile
  - Not easy to find and communicate with moving nodes
  - Communication properties, delays, message loss rates etc change with changing locations
  - Locations of nodes are important, determine their needs and preferences

Practical challenges in distributed systems

- Scalability with size (number of nodes)
  - Systems may need to grow in number of nodes when it has to handle more data or users
  - The design should easily adapt to this growth and not get stuck trying to handle large amounts of data or many nodes
  - E.g. In a multiplayer game with many players, if all actions of each player in every second is sent to all other players, this will generate $O(n^2)$ messages every second.
  - Options:
    - Make efficient systems that can handle O(n^2) messages per second (more and more difficult with growing n)
    - Or, make clever choices of which messages to send to which players, and keep it manageable

Practical challenges in distributed systems

- Transparency
  - User should not have to worry about details
    - How many nodes
    - How they are connected
    - Locations, addresses
    - mobility
    - Failures
    - concurrency
    - Network protocols

Practical challenges in distributed systems

- Security
  - Confidentiality – only authorized users can access
  - Integrity – should not get altered/corrupted or get into an undesirable state
  - Availability – should not get disrupted by enemies (e.g. by a denial of service attack)

  - Perfect security is impossible. Good practical security is usually possible, but takes some care and effort. Encryption helps.

Distributed computations: Examples

- Agreement
  - Get nodes to agree on the value of something
    - When should we go to the movie?
    - What should be the multiplayer strategy?
    - When should we sell the shares?
    - …

Distributed computations: Examples

- Leader election
  - Which node is the coordinator in hadoop?
  - Which node is the which returns the final result?

Distributed computations: Examples

- Deciding matters of time:
  - What happened first? A or B?
  - What sequence of events definitely happened and what cannot have happened?

Distributed computations: Examples

- Store and retrieve data
  - Peer to peer systems
  - Sensor networks

Distributed computations: Examples

- Aggregation: getting data from many nodes
  - What is the average temperature recorded by the mobile phones?
  - How many people are in the building?
  - What is the maximum speed of cars on the highway?

Summary: Distributed Systems

- Multiple computers operating by sending messages to each other over a network
- Integral to many emerging trends in computing
- Reasons for distributed systems:
  - Tasks get done faster
  - Can be made more resilient: If one computer fails, another takes over
  - Load balancing and resource sharing
  - Sometimes, systems are inherently distributed. E.g. people from different locations collaborating on tasks, playing games, etc.
  - Brings out many natural questions about how natural world, ecosystems, economies, emergent behaviors work
    - Eg. Birds flocking, fireflies blinking in sync, people walking without colliding, economic game theory and equilibria…

Summary: Distributed Systems

- Examples:
  - Web browsing
  - Multiplayer games
  - Digital (Stock) markets
  - Collaborative editing (Wikipedia, reddit, slashdot..)
  - Big data processing (hadoop etc)
  - Networks
  - Mobile and sensor systems
  - Ubiquitous computing
  - Autonomous vehicles

Ref: CDK

Challenges in Distributed system design

- Lack of global knowledge
- No perfect (shared) clock
- Communication is costly in large volumes
- Failures of nodes/links, loss of messages
- Scalability
- Transparency
- Security
- Mobility

The course

- Bring paper and pen
- Take notes
  - Identify what is important
- Previous exam papers are online