Distributed Systems

Course Review

Rik Sarkar

University of Edinburgh
Fall 2014

Today: Review of course

- Slight updates to slides
  - Including references etc.
  - Always use the up to date online version in studying

Distributed Computing is everywhere

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

Ref: CDK

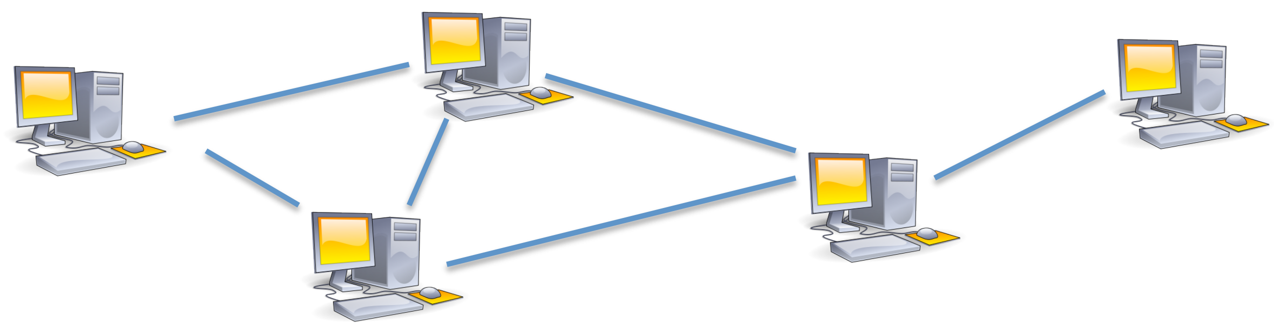
Reading & Books

- **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
  - [Wiki] : Wikipedia

Distributed system

- Computing in a graph
  - Nodes: computers
  - Edges: Connections



The main challenge:

- Knowledge is distributed
- No one node knows everything
- Different nodes have different views (data) of the system
- Yet, nodes are expected to achieve a common goal

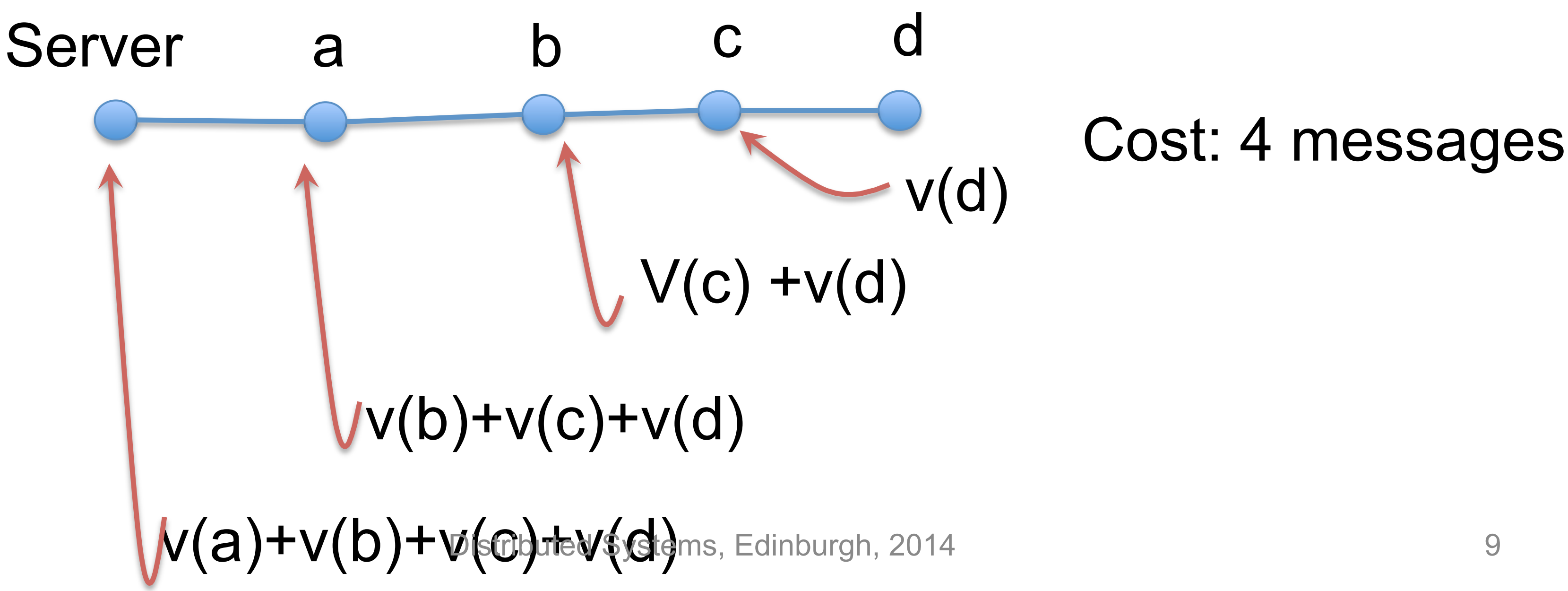
Other challenges

- Communication is expensive
  - We have to be efficient: Send the right messages
  - Communication is usually measured in asymptotic notation
    - O, Ω, θ
- Time is relative
  - Makes hard to compare events
- There may be failures: nodes, links etc
- Mobility
- Security
- Scalability: There can be many nodes: all problems become more challenging

Simple Algorithms

Example

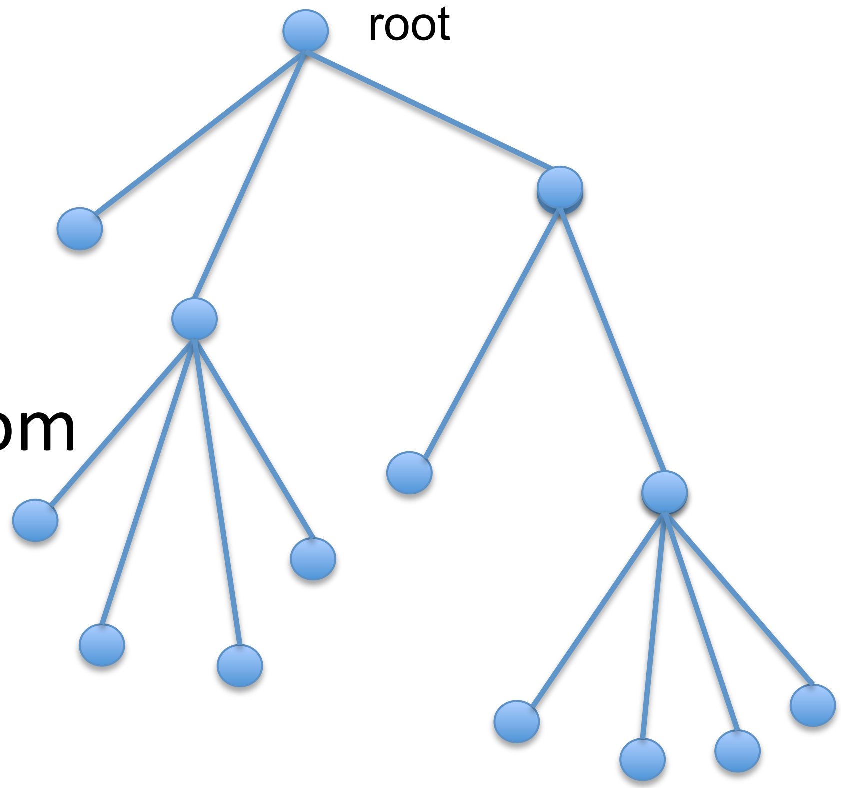
- A simple distributed computation:
  - Each node has stored a numeric value
  - Compute the total of the numbers



Convergecast

Ref: NL

- Suppose root wants to know sum of values at all nodes
- It sends “compute” message to all children
- The values move upward
- Each node adds values from all children and its own value
- Sends it to its parent



Communication with all nodes

- Flooding
- Constructing a (BFS or spanning) tree using flooding

Minimum spanning trees

- Trees of smallest total edge costs
- Useful in communication
- Prim's & Kruskal's algorithms
  - [Ref: Wiki]
- GHS algorithm
  - [Ref: NL]

- Maximum independent set and maximal independent sets

Time

Ref: CDK

- Time & ordering of events are important
- Clocks are not perfect
  - Drift and skew

- Simple algorithms to unify time
  - Christian's algorithm, berkeley, NTP etc..

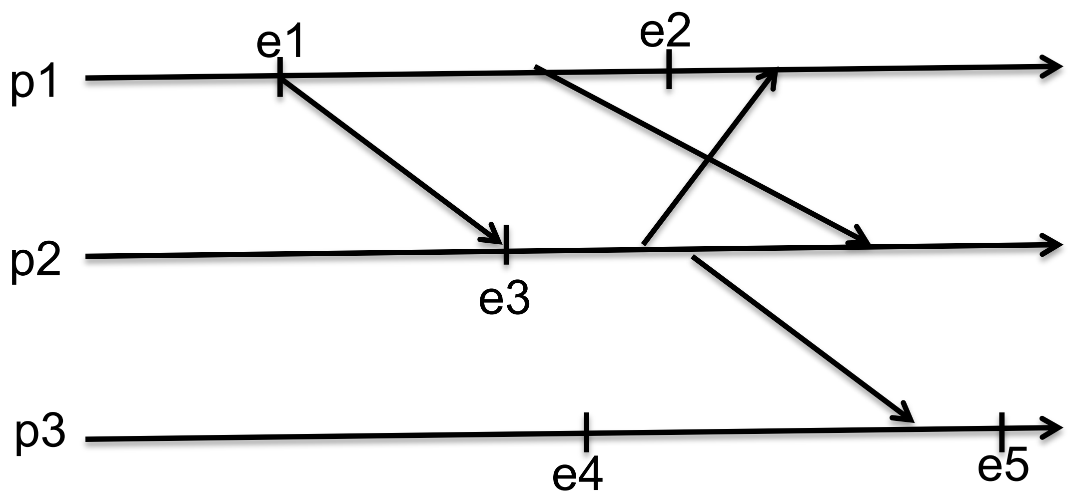
- Not in exam: GPS, special relativity

Logical time

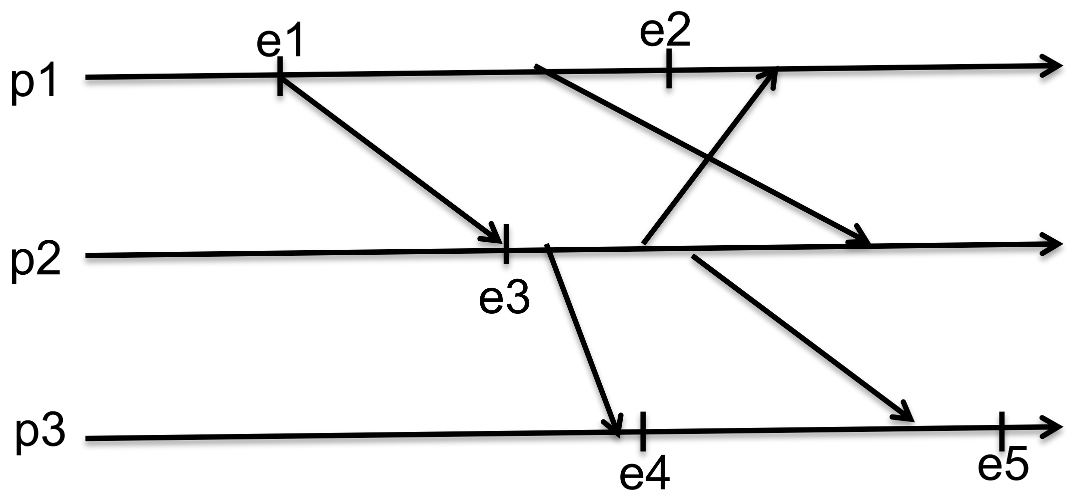
- For ordering of events without using clocks
  - Ref: CDK & VG

Happened before

- $a \longrightarrow b$: a happened before b
  - If a and b are successive events in same process then $a \longrightarrow b$
  - Send before receive
    - If a : "send" event of message m
    - And b : "receive" event of message m
    - Then $a \longrightarrow b$
  - Transitive: $a \longrightarrow b$ and $b \longrightarrow c \Longrightarrow a \longrightarrow c$


e1
e2
p1
p2
e3
p3
e4
e5

- Events without a happened before relation are “concurrent”
- e1⟶e2, e3⟶e4,e1⟶e5, e5||e2



Happened before & causal order

- Happened before == could have caused/ influenced
- Preserves causal relations
- Implies a partial order
  - Implies time ordering between certain pairs of events
  - Does not imply anything about ordering between concurrent events

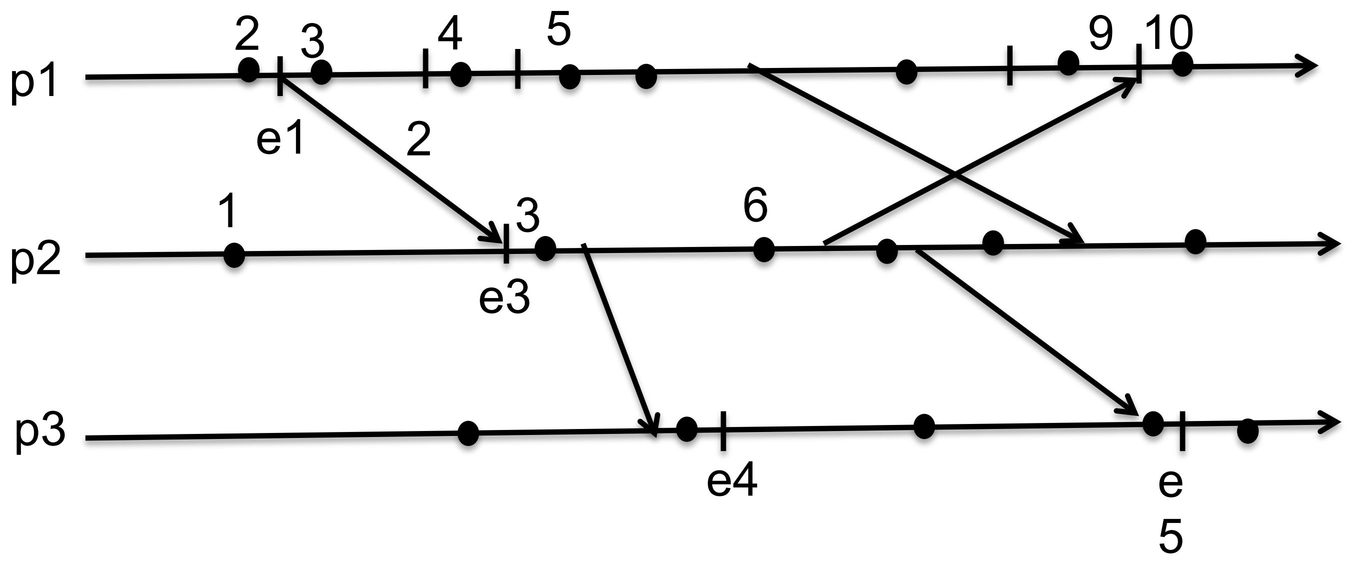
Logical clocks

- Idea: Use a counter at each process
- Increment after each event
- It counts the states of the process
- Each event has an associated time: The count of the state when the event happened

Lamport clocks

- Keep a logical clock (counter)
- Send it with every message
- On receiving a message, set own clock to max({own counter, message counter}) + 1
- For any event e, write c(e) for the logical time
- Property:
  - If $a \longrightarrow b$, then $c(a) < c(b)$
  - If $a \,||\, b$, then no guarantees

Lamport clocks: example



Concurrency and lamport clocks

- If e1⟶e2
  - Then no lamport clock C exists with C(e1)== C(e2)

Concurrency and lamport clocks

- If e1⟶e2
  - Then no lamport clock C exists with C(e1)== C(e2)

- If e1||e2, then there exists a lamport clock C such that C(e1)== C(e2)

The purpose of Lamport clocks

- If $a \longrightarrow b$, then $c(a) < c(b)$
- If we order all events by their lamport clock times
  - We get a partial order, since some events have same time
  - The partial order satisfies "causal relations"

Modifications

- Basic lamport clocks can have same time for 2 events in different processes
  - We can break these ties by process id
  - Then any 2 events are ordered: total order
- Vector clocks
  - Lamport clock ordering do not imply causal relation
  - Vector clocks can be used to get perfect knowledge of causality

Distributed snapshots

Ref: CDK

- Consistent cuts
  - Snapshot algorithms record consistent states
- Single snapshots are good for detecting stable predicates
- Non-stable predicates
  - Possibly, definitely etc
  - Require checking all consistent cuts

Mutual Exclusion

Ref: CDK, VG

- Properties: Safety, Liveness, Fairness
- Central server
- Token ring
- Lamport
- Ricart & Agrawala
- Maekawa's quorum system with grids

Communication and models

- Medium access & broadcast
- Routing & point to point communication
- Transport: ordering and congestion control
- Each layer of a network solves a different distributed problem

- Synchronous and asynchronous communication
  - Communication in rounds
  - Easy to implement when message transmission time is bounded

Failure detectors

Ref: CDK

- With bounded message delays
- With probabilities

Leader election

Ref: NL & CDK

- Find the highest id node
- Convergecast
- Ring search: chang and roberts
- Ring search: Exponentially growing: Hirshberg Sinclair
- Bully algorithm

Multicast

- Usually used in local/small networks with broadcast
- When used in slightly larger networks
- Can we ensure that messages are delivered reliably to all nodes in group?
- We use basic multicast as a building block for reliable multicast
- Possible guarantees: FIFO, causality, total order

Termination and OS

- Termination detection
  - Weight throwing
  - Dijkstra Scholten
    - Ref: Wiki, VG
- OS
  - Networked OS
  - Distributed OS
  - Virtualization
    - Ref: CDK

Peer to Peer

Ref: CDK, Wiki

- The challenges and benefits
- Examples: Internet, napster, gnutella, chord, skype, bittirrent, SETI@home
- DHT

Localization & Location based routing

- Ref: Slides only
  - You can find more material on internet, wiki, other course slides

- Not in exam: MDS, lower and upper bounds on complexity of greedy and face routing, cross link detection protocol, Rumor routing

Coloring and MIS

- Assignment of non-interfering communication channels
- Finding largest sets of non-interfering nodes
- Randomized algorithm can be much more efficient
- Ref: given in slides

Security

- Main defense is Encryption
- Public key encryption, RSA

Course Matter

- Assignment
- Course
- Material