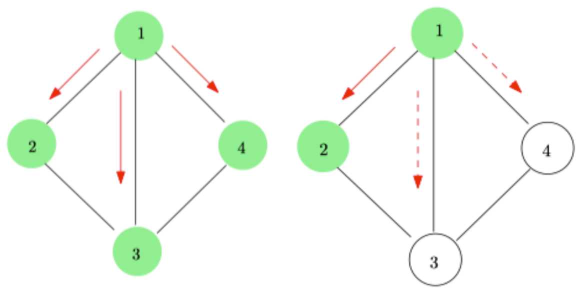
COMPSCI 711 Outline

Github: cwgavin

Basics
- Parallel computing & Distributed computing
- Graph theory (eccentricity, diameter, radius)
- BFS and DFS spanning trees
- Synchronous/asynchronous models
    - Async: transit time in [0, 1]
        - FIFO channel -> congestion
    - Sync: transit time = 1 (special case of Async)
    - Both are non-deterministic
- Confluent system always arrives to the same final decision (although non-deterministic in middle)

Echo
- broadcast: build spanning tree
- convergecast: confirm termination

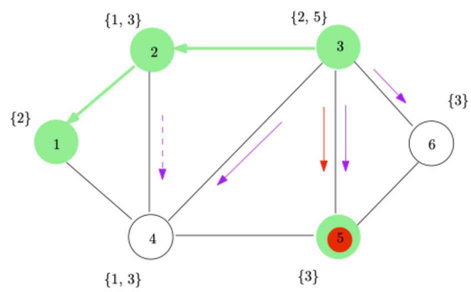


Echo (in **Sync** mode = SyncBFS)
- time $O(2D+1)$, message $O(2|E|)$
- minimum height spanning tree

Echo (in **Async** mode)
- time $O(|V|)$, message $O(2|E|)$
- rush to build a spanning tree (greedy), could end up more time on convergecast

Echo-Size

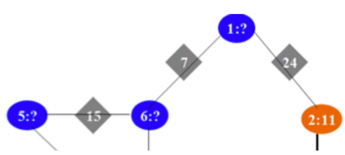
Distributed DFS
- Classical DFS
    - time $O(2|E|)$, message $O(2|E|)$
- Cidon DFS
    - time $O(2|V|-2)$, message $O(3|E|)$
    - tok + **vis** (visited) tokens
- Cidon DFS (in Sync mode)
- Cidon DFS (in Async mode)



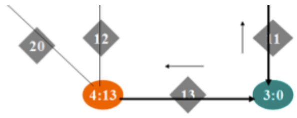
Distributed BFS
- in Sync mode = Echo
- in Async mode: hard to implement

Bellman-Ford
- find all shortest paths from a single source (shortest path spanning tree)
- time $O(|V||E|)$

Distributed Bellman-Ford (in **Sync** mode ≈ **Echo**++)
- time $O(|V|)$, message $O(|V||E|)$

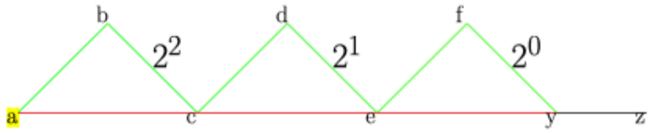


- Echo + single source distance + adjust phase
- termination: TTL = |V|

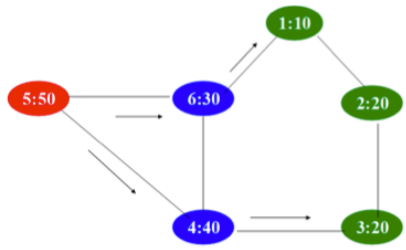


Distributed Bellman-Ford (in **Async** mode)

- time with FIFO: $O(|V|^{|V|})$, no FIFO: $O(|V|)$; message $O(|V|^{|V|})$



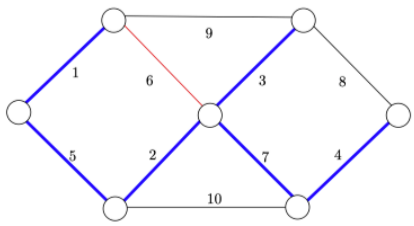
Maximal Independent Set

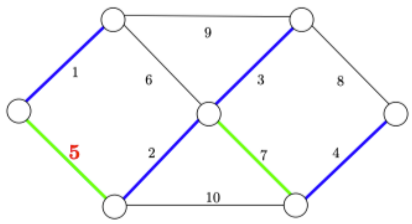


- No two neighbours in the same set, cannot be extended anymore.
- Luby's algorithm (Sync)
  - stop with probability 1, expected rounds O(log N)
  - range of random numbers = [1, …, N^4]
  - each round: generate random number, notify neighbours, winners and losers disconnect

Minimum Spanning Tree (MST)

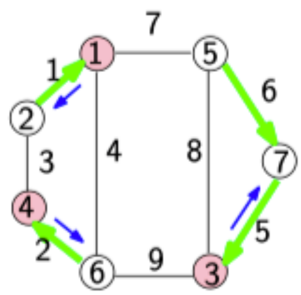
- If edges have different weights, there is a unique MST.
- Prim
  - keep growing a single tree by merging with one node at a time
  - consider only the current tree's MWOE
- Kruskal
  - merge two trees at a time
  - consider only the lowest cost MWOE
- Boruvka
  - multi-way merges at the same time
  - consider all MWOEs
  - one node on the common MWOE will be the root of tree





- Distributed MST (in **Sync** mode ≈ distributed **Boruvka**)
  - time O(N log N), message O((N+M) log N)
  - #levels O(log N)
  - at each level, merge at least 2 components
  - at level k ≥ 0, total size is at least 2^k (exponential)

  1. each node find its MWOE, send *connect* message, the node on common MWOE with smaller ID becomes root
  2. root sends *initiate* message (with root ID) to children to explore MWOEs
     [decrease counter = N for synchronization]
     [time O(N log N)]
  3. all nodes send *test* message (with root ID) to



all unexplored edges

4. all nodes send *accept* to nodes in different tree, *reject* to nodes in the same tree [this step not really needed]
5. children nodes send *report* message (with min MWOE of children and self) to root
   [decrease counter = N for synchronization]
   [time O(N log N)]
6. root decides the tree's MWOE, send *connect* to the node on tree's MWOE, and reverse parent/child along the way
   [decrease counter = N for synchronization]
   [time O(N log N)]
7. nodes on tree's MWOE send *connect* (actual connection)

- Distributed MST (in **Async** mode: solved by GHS)
  - difficulty: two trees may be at different levels

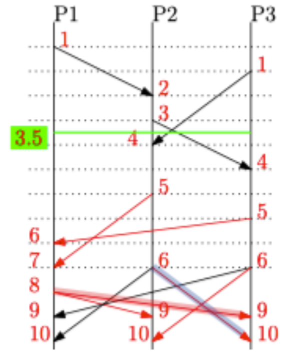
Logical clocks

- Independent of the physical time.
- $a \prec b$ event a happens before event b iff
  - a and b occur in the same node, and a happens before b, or
  - a is a send event and b is the corresponding receive event
  - transitive closure: ∃c s.t. $a \prec c \prec b$
- $\prec$ determines a partial order, which creates directed acyclic graphs
- Logical time is a mapping C from events to elements of a partially ordered set:
  $a \prec b \Rightarrow C(a) < C(b)$
- Ideal/exact match:
  $a \prec b \Leftrightarrow C(a) < C(b)$
- Property: Must be determined by nodes themselves, must be unique, good to be same for essentially same executions
- **Lamport** (total order -> not ideal)
  - before sending messages, clock += 1
  - after receiving messages, clock = max(clock, m.clock) + 1
- **Vector** (partial order, ideal)
  - $V(P_2) = (v1, v2, v3)$
  - $(v1, v2, v3) \leq (v1', v2', v3') \Leftrightarrow v1 \leq v1', v2 \leq v2', v3 \leq v3'$
  - before sending messages:
    - $V(P_2)$: $(v1, v2, v3) \Rightarrow (v1, v2 + 1, v3)$
  - after receiving messages:
    - $V(P_2)$: $(v1, v2, v3), (r1, r2, r3) \Rightarrow (\max(v1,r1), \max(v2,r2) + 1, \max(v3,r3))$

- can detect FIFO

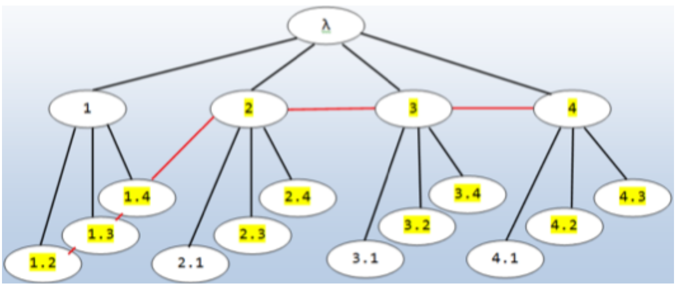
Distributed snapshot (count money example)

- for each channel, keep adding received money from time t until getting the first message sent after time t (marker message)
- assumption 1: infinite message flows
- assumption 2: FIFO flows (if no FIFO, fix by adding sequence numbers for each channel & including messages we should receive before the marker message)



Byzantine agreement (**Sync**)

- Impossible to deterministically solve (even a simple stopping failure) in Async mode (FLP, 1985)
- In Sync mode, deterministically solved iff
  - **N ≥ 3F + 1**
- Termination: all loyal processes eventually decide
- Agreement: all loyal processes decide on the same value
- Validity: if all loyal processes start with the same initial value v, the final decision must be v; otherwise could be any of initial values.
- EIG tree
  - L = F + 1



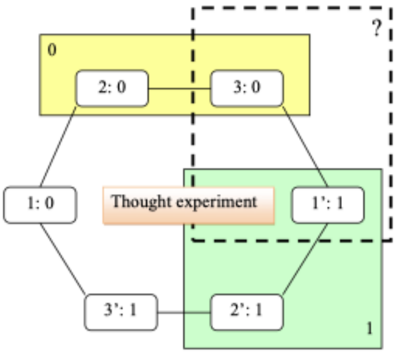
sibling group size:
N - L + 1

(highlighted tree nodes are messages relayed by loyal processes, red line is the common path covering)

- Triple modular redundancy (TMR)
  - all loyal modules give the same initial value - very different from Byzantine problem

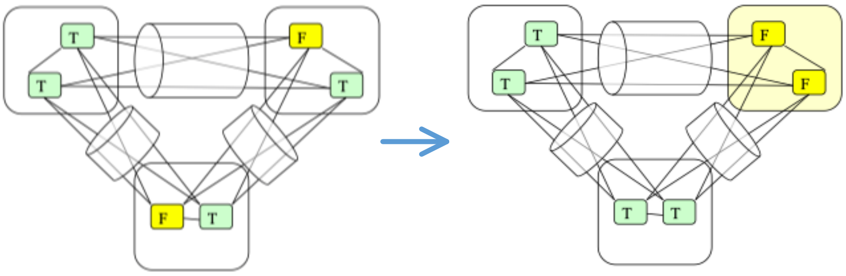
Byzantine proofs - #processes

- No deterministic algorithm for n=3, f=1



- No deterministic algorithm for 2 ≤ n ≤ 3f

- n = 2, f = 1: each node assumes the other is faulty, so must decide on its own value -> cannot agree
- 3 ≤ n ≤ 3f



essentially the same as n=3, f=1

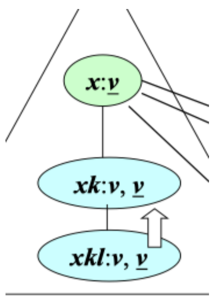
- Therefore: n must be greater than 3f

Byzantine proofs - #levels

- L ≥ F + 1
  - each root-to-leaves branch contains at least one node containing message relayed by loyal nodes (trustworthy)
- L ≤ F + 1
  - each sibling group has a strict majority of nodes containing message relayed by loyal nodes (trustworthy)
- Therefore: L = F + 1

Byzantine proofs - EIG Agreement

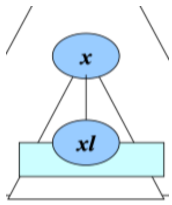
- A tree node is *common* if it has the same newval (final decision) across all loyal processes.
- A *path covering* is a set containing at least one tree node on each root-to-leaves path (covering all paths).
- A *common path covering* is a path covering where all tree nodes are common. (top-down, find the first tree node that ends with a loyal process)
- **Lemma 6.16**: for a tree node **xk**, where k is a loyal process, val(xk)_i = newval(xk)_i = val(xk)_j = newval(xk)_j **= v** for all loyal processes (more than common)
  - for leaves, since relayed by loyal process k, val(xk) would be the same across processes, and by definition of leaves, val = newval
  - for non-leaves, the majority of children are xkl, where l is loyal process.
    val(xk) = val(xkl) since both k and l are loyal.
    val(xkl) = newval(xkl) since xkl are leaves (height induction).
    newval(xkl) = newval(xk) since majority.



- For a tree node x, if there is a common path covering its substree, then x is common.
  - for leaves, the subtree is itself -> it's on

common path covering -> it's common

- for non-leaves,
  - if it's on common path covering, it's common
  - if it's above common path covering, it's common by induction



Byzantine proofs - EIG Termination

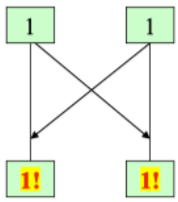
- stops after L = F + 1 messaging rounds

Byzantine proofs - EIG Validity

- If all loyal processes start with the same initial value v, according to Lemma 6.16, val(k)_i = newval(k)_i = val(k)_j = newval(k)_j **= v** (k is a first level tree node of loyal process), and they form a strict majority, so the root decision will be v.

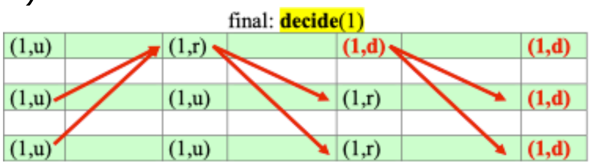
Other consensus algorithms

- Synchronizer (Async -> Sync)
  - GlobSync & LocSync
  - could be faster by avoiding greedy choices
  - avoid complex async algorithm
- No deterministic agreement is possible with unbounded communication failures.

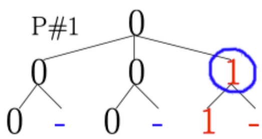


- Distributed Commit (**Sync**, stopping failure only, no recovering)
  - Termination:
    - 2PC: if no failures, all processes eventually decide
    - 3PC: all non-faulty processes eventually decide
  - Agreement:
    - all processes (including faulty ones) decide on the same value
  - Validity:
    - if any process start with 0, the decision must be 0 (0,d)
    - if all processes start with 1, and there are no failures, then the decision must be 1
  - Leader process & cohort processes
  - States:
    - 2PC: uncertain(1,u), decided
    - 3PC: uncertain(1,u), ready(1,r), decided
  - 2 Phase Commit
    - weak termination = blocking
    - faster, fewer messages
    - cohorts send to leader, leader decides and send back
  - 3 Phase Commit
    - strong termination = non-blocking (max N rounds)
    - cohorts send to leader, leader decide on

- cohorts send to leader, leader decide on 0 or make cohort ready on 1, and finally decide on 1.
  - if leader fails, elect a new leader, repeat the three phases
- in the first 3 rounds: missing message -> decide on 0
- in the following rounds: missing message -> ignore
- any (1,r) -> attempt decide on 1
- all (1,u) -> decide on 0 after leader fails (must assume failed process decided on 0)



Stopping failures model (**Sync**)
- Impossible to deterministically solve (even a simple stopping failure) in Async mode (FLP, 1985)
- Solution: randomization, failure detectors (timeout)
- Stopping failure only, no recovering
  - **N ≥ F + 1**
- Termination: all non-faulty processes eventually decide
- Agreement: all processes decide on the same value
- Validity: if all processes start with v, the final decision must be v; otherwise could be any of initial values.
- EIG Stop
  - F + 1 messaging rounds
  - check all non-null values in EIG tree
    - if single value, that would be decision
    - otherwise, decision is V0



- EIG Stop (optimized)
  - each process only sends two messages:
    1. initial value to all processes
    2. a different value to all processes (once it gets a different value)

Initial	EIGStop	EIGByz	3PC
0 0 0 0	0	0	0
0 0 0 1	V0	0	0
0 0 1 1	V0	V0	0
0 1 1 1	V0	1	0
1 1 1 1	1	1	1
0 0 0 X	0	0	0
0 0 1 X	V0	V0	0
0 1 1 X	V0	V0	0
1 1 1 X	1	1	0

EIGStop discards unsent messages

EIGByz regards unsent messages as V0

必须假定 failed process 决定为 0

(assuming X stops from start)
- Byzantine agreement with authentication -> cannot forge messages, becomes a stopping failure only problem

TurpinCoan (**Sync**, multiple choice)
- Two extra rounds + binary Byzantine
- initial choice $x \in V$
- proposal $y \in V \cup \perp = \perp$
- candidate $z \in V \cup \perp = \perp$
- vote $\hat{v} \in \{0,1\} = 0$
- $W \subseteq V \cup \perp$ multiset of received messages
    - $|W| = N$, missing messages are regarded as $\perp$

1. send x to all processes
   if $|W|_v \geq$ **N - F** $= 2F + 1$, then $y = v$;
   else $y = \perp$
    - all loyal processes either have same y or remain undefined
2. send y to all processes
   if $|W|_v \geq$ **N - F** $= 2F + 1$, then $z = v$, $v\text{^} = 1$; [vote for z]
   else if exist $\mathrm{argmax}_v|W|_v$, then $z = v$, $v\text{^} = 0$;
   else $z = \perp$, $v\text{^} = 0$
3. binary Byzantine on v^
   if Byz decision is 1, then final decision is z; [elected]
   else final decision is V0

BenOr (**Async**, stopping failure)
- randomization
- weaker termination: evertual termination with probability = 1
- initial choice $x \in \{0,1\}$
- proposal $y \in \{0,1,\perp\} = \perp$
- M is multiset of first **N - F** = 2F + 1 received messages
- each step s has two rounds ($s \geq 0$):
    1. send (I, s, x) to all processes
       if all $m \in M$ have the same value v, then $y = v$;
       else $y = \perp$
    2. send (II, s, y) to all processes
       if all $m \in M$ have the same value v, then $x = v$, decide v, and continue;
       else if at least N - 2F = **F + 1** $m \in M$ have the same value v, then $x = v$, but do not decide;
       else x = Random(0 or 1)

BenOr	$F < \sqrt{N}$
Lynch	$F < N / 3$
Aguilera, Toueg	$F < N /2$