Distributed Systems

Byzantine Agreement

He Sun
School of Informatics
University of Edinburgh

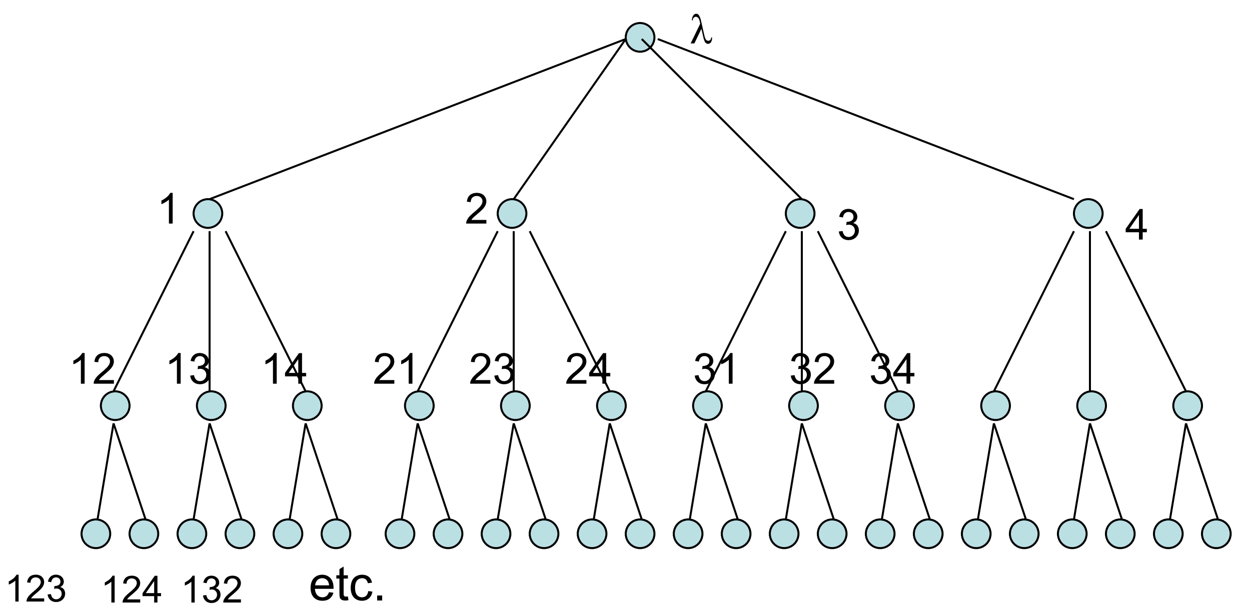
THE UNIVERSITY *of* EDINBURGH

Outline

- Finish EIG algorithm for Byzantine agreement.
- Number-of-processors lower bound for Byzantine agreement.
- Connectivity bounds.

Exponential Information Gathering (EIG)

- A strategy for consensus algorithms, which works for Byzantine agreement as well as stopping agreement.
- Based on EIG tree data structure.
- EIG tree $T_{n,f}$, for n processes, f failures:
  - $f + 2$ levels
  - Paths from root to leaf correspond to strings of $f + 1$ distinct process names.
- Example: $T_{4,2}$

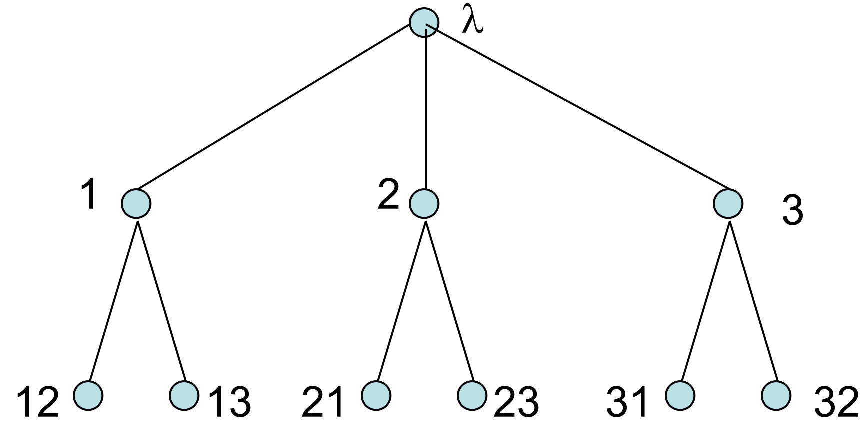


EIG Stopping Agreement Algorithm

- Each process i uses the same EIG tree, $T_{n,f}$.
- Decorates nodes of the tree with values in V, level by level.
- Initially: Decorate root with i's input value.
- Round $r \geq 1$:
  - Send all level $r-1$ decorations for nodes to everyone.
    - Including yourself---simulate locally.
  - Use received messages to decorate level r nodes---to determine label, append sender's id at the end.
  - If no message received, use $\perp$.
- The decoration for node $(i_1, i_2, i_3, \dots, i_k)$ in i's tree is the value v such that (i_k told i) that (i_{k-1} told i_k) that …that (i_1 told i_2) that **i_1**'s initial value was v.
- Decision rule for stopping case:
  - Trivial
  - Let W = set of all values decorating the local EIG tree.
  - If $|W| = 1$ decide that value, else default v_0.

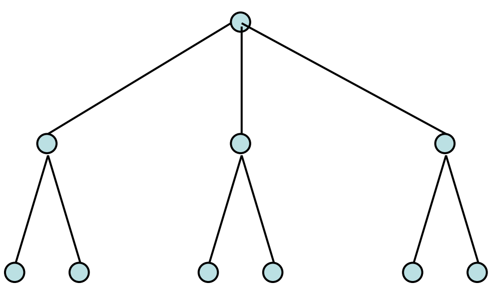
Example

- 3 processes, 1 failure
- Use $T_{3,1}$:



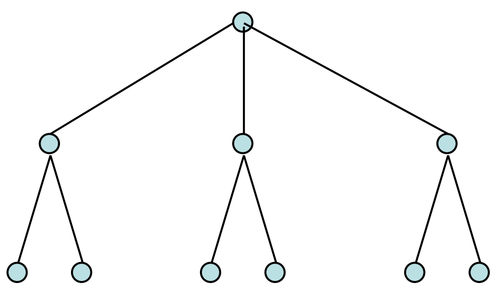
Initial values:

1



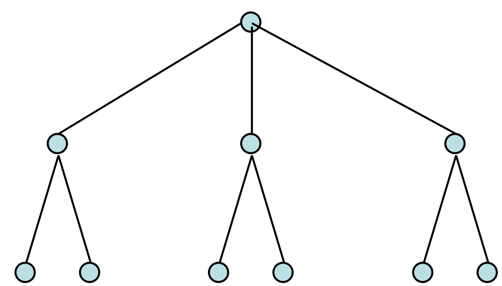
Process 1

0



Process 2

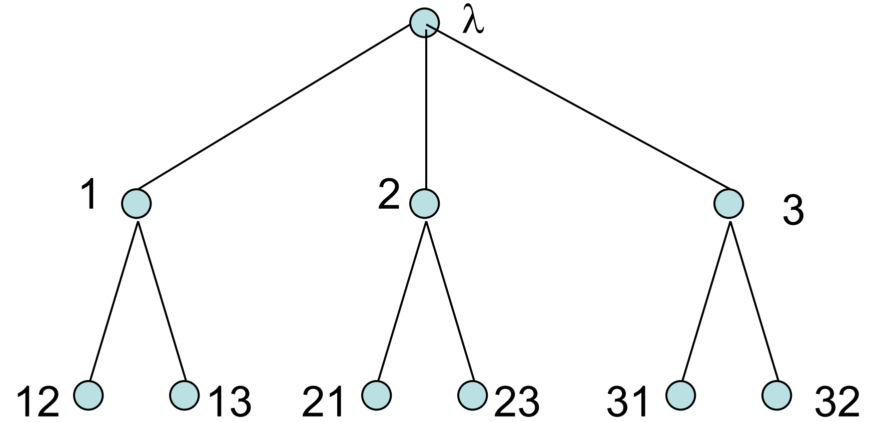
1

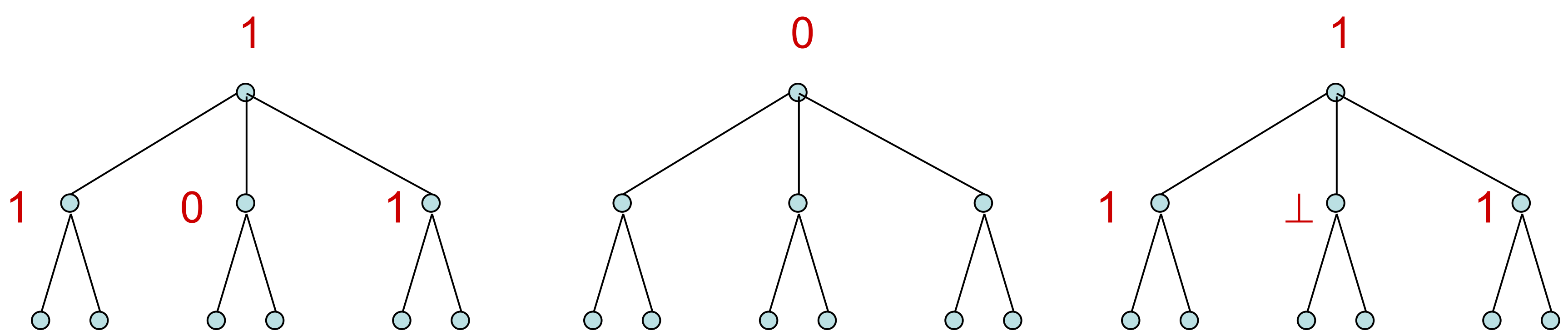


Process 3

Example

- Process 2 is faulty, fails after sending to process 1 at round 1.
- After round 1:

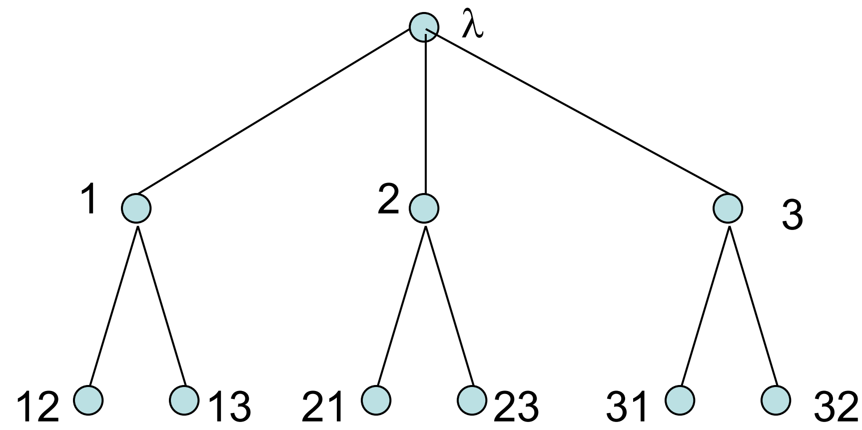


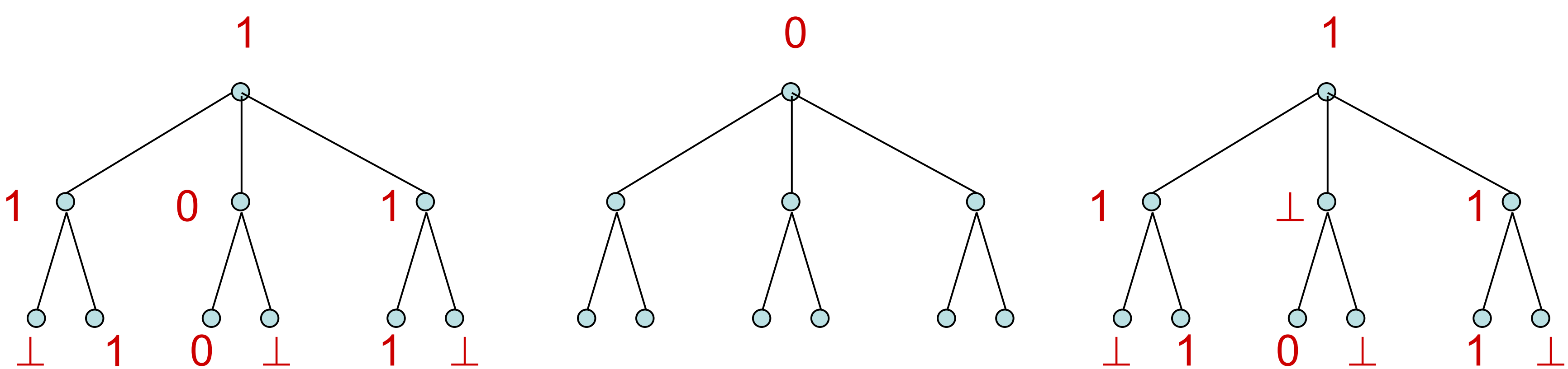


Process 1 Process 2 Process 3

Example

- After round 2:





Process 1 Process 2 Process 3

p3 discovers that p2's value is 0 after round 2, by hearing it from p1.

Byzantine Agreement

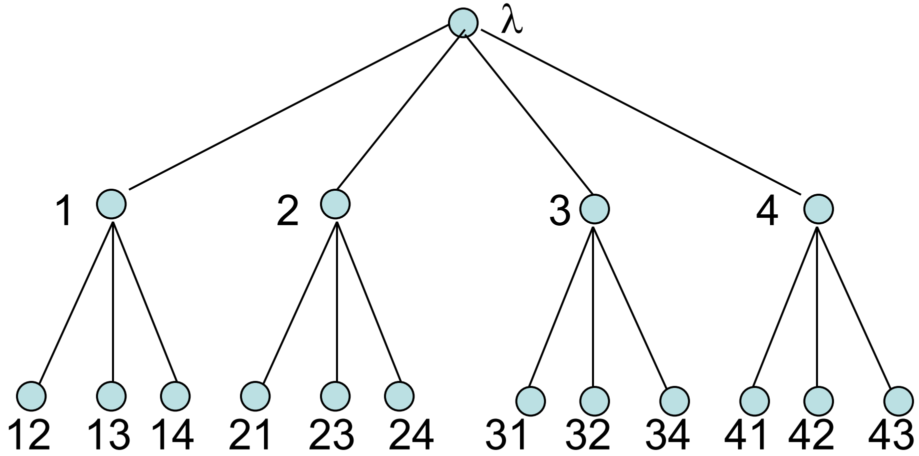
- Recall correctness conditions:
  - Agreement: No two nonfaulty processes decide on different values.
  - Validity: If all nonfaulty processes start with the same v, then v is the only allowable decision for nonfaulty processes.
  - Termination: All nonfaulty processes eventually decide.
- Present EIG algorithm for Byzantine agreement, using:
  - Exponential communication (in f)
  - $f + 1$ rounds
  - $n > 3f$

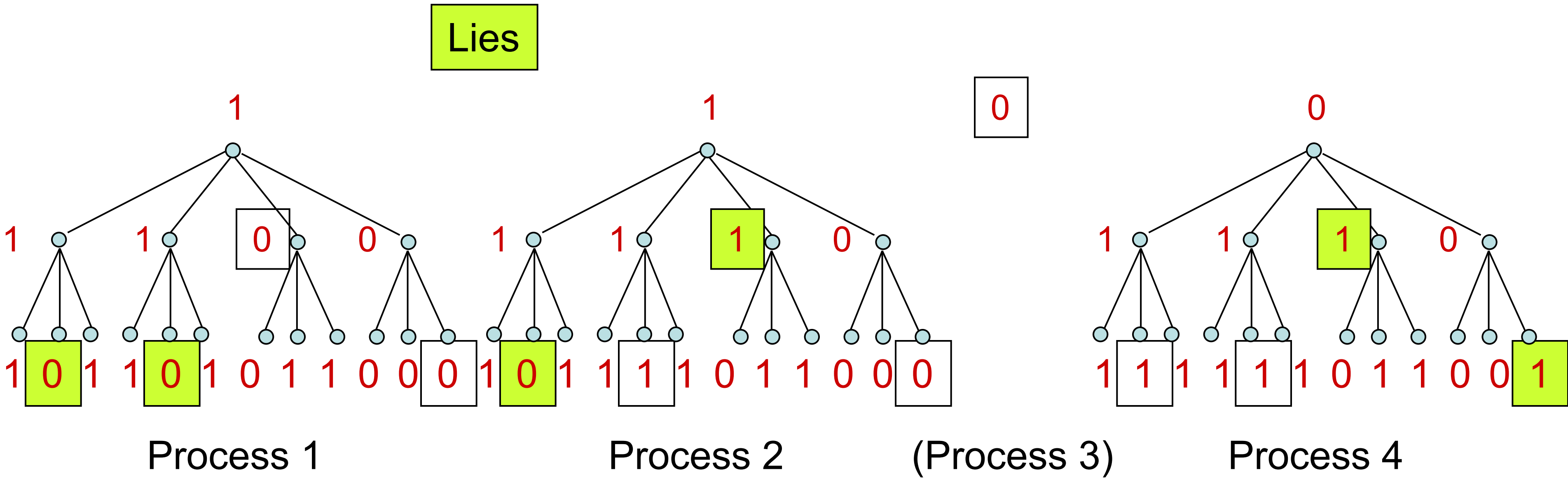
EIG Algorithm for Byzantine Agreement

- Use EIG tree.
- Relay messages for $f+1$ rounds.
- Decorate the EIG tree with values from V, replacing any garbage messages with default value v_0.
- Call the decorations val(x), where x is any node label.
- Decision rule:
  - Redecorate the tree, defining newval(x).
    - Proceed bottom-up.
    - Leaf: newval(x) = val(x)
    - Non-leaf: newval(x) =
      - newval of strict majority of children in the tree, if majority exists,
      - v_0 otherwise.
  - Final decision: newval(λ) (newval at root)

Example: $n = 4, f = 1$

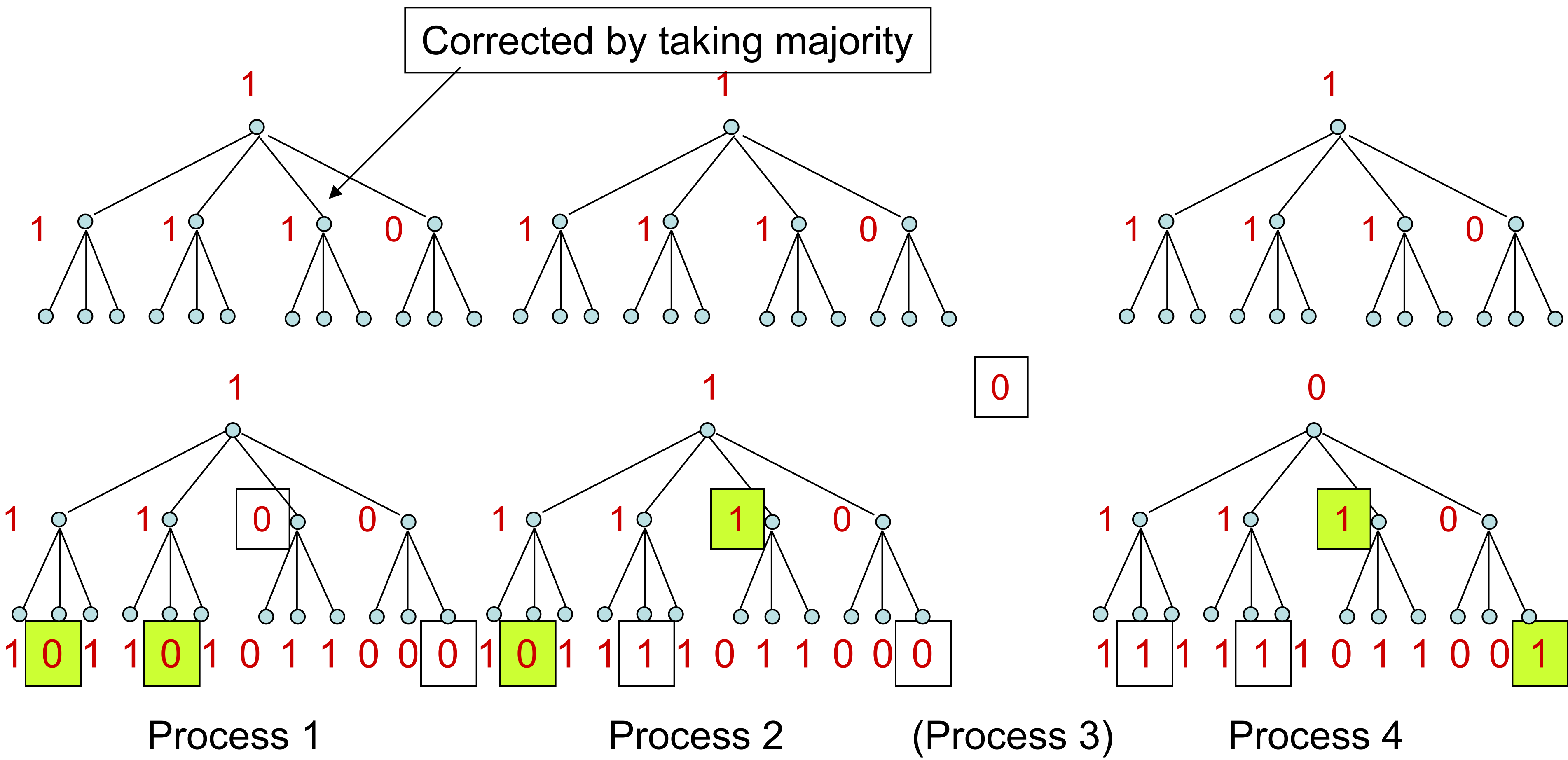
- $T_{4,1}$:
- Consider a possible execution in which p_3 is faulty.
- Initial values 1 1 0 0
- Round 1
- Round 2





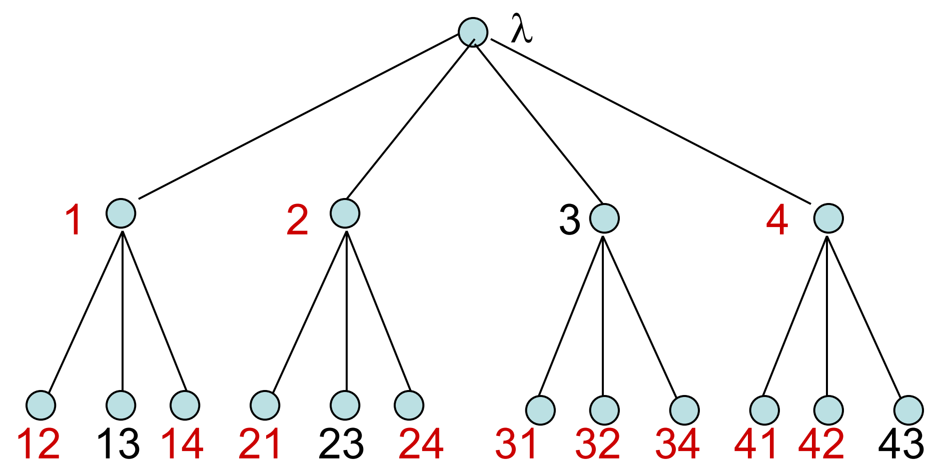
Example: $n = 4, f = 1$

- Now calculate newvals, bottom-up, choosing majority values, $v_0 = 0$ if no majority.



Correctness Proof

- Lemma 1: If i, j, k are nonfaulty, then val(x)$_i$ = val(x)$_j$ for every node label x ending with k.
- In example, such nodes are:



- Proof: k sends same message to i and j and they decorate accordingly.

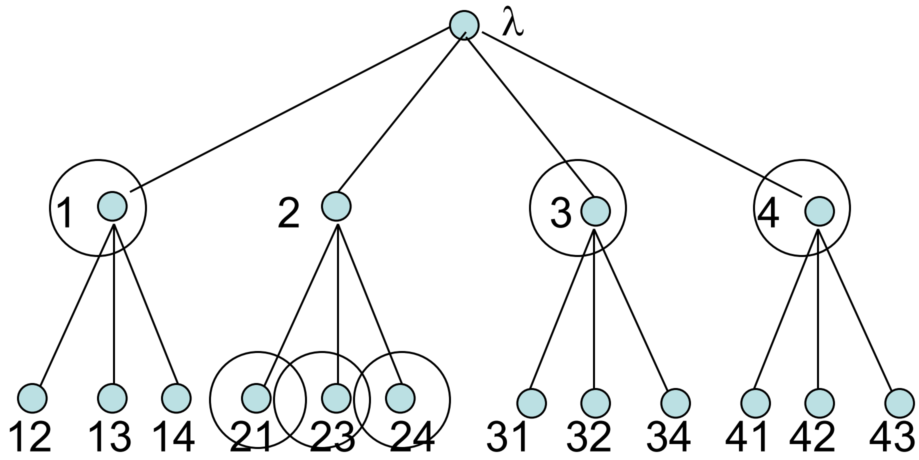
Correctness Proof (cont.)

- Lemma 2: If x ends with nonfaulty process index then $\exists v \in V$ such that $\mathrm{val}(x)_i = \mathrm{newval}(x)_i = v$ for every nonfaulty i.
- Proof: Induction on lengths of labels, bottom up.
  - Basis: Leaf.
    - Lemma 1 implies that all nonfaulty processes have same $\mathrm{val}(x)$.
    - newval = val for each leaf.
  - Inductive step: $|x| = r \leq f$ ($|x| = f + 1$ at leaves)
    - Lemma 1 implies that all nonfaulty processes have same $\mathrm{val}(x)$, say v.
    - We need $\mathrm{newval}(x) = v$ everywhere also.
    - Every nonfaulty process j broadcasts same v for x at round $r + 1$, so $\mathrm{val}(xj)_i = v$ for every nonfaulty j and i.
    - By inductive hypothesis, also $\mathrm{newval}(xj)_i = v$ for every nonfaulty j and i.
    - A majority of labels of x's children end with nonfaulty process indices:
      - Number of children of node x is $\geq n - f > 3f - f = 2f$.
      - At most f are faulty.
    - So, majority rule applied by i leads to $\mathrm{newval(x)}_i = v$, for all nonfaulty i.

Main Correctness Conditions

- *Validity:*
  - If all nonfaulty processes begin with v, then all nonfaulty processes broadcast v at round 1, so val$(j)_i = v$ for all nonfaulty i, j.
  - By Lemma 2, also newval$(j)_i = v$ for all nonfaulty i, j.
  - Majority rule implies newval$(\lambda)_i = v$ for all nonfaulty i.
  - So all nonfaulty i decide v.
- *Termination:*
  - Obvious.
- *Agreement:*

Agreement

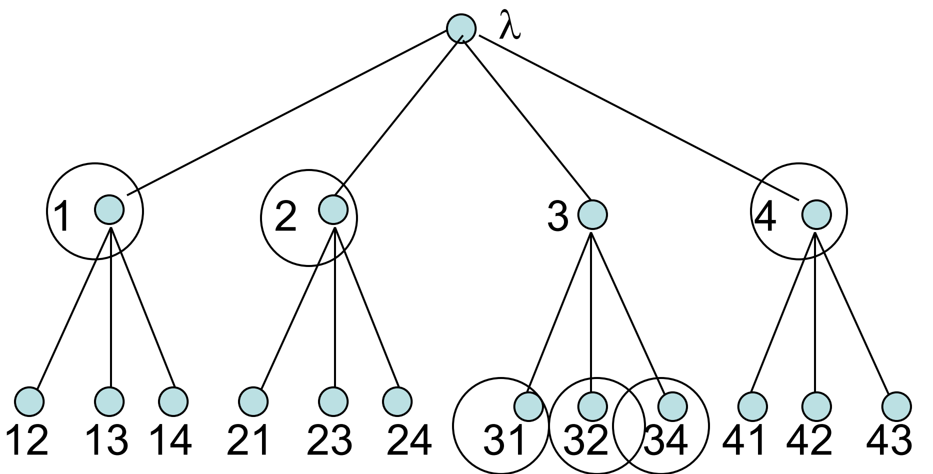


- Path covering: Subset of nodes containing at least one node on each path from root to leaf.

- Common node: One for which all nonfaulty processes have the same newval.
  - If label ends in nonfaulty process index, Lemma 2 implies it's common.
  - Might be others too.

Agreement

- Lemma 3: There exists a path covering all of whose nodes are common.
- Proof:
  - Let C = nodes with labels of the form xj, j nonfaulty.
  - By Lemma 2, all of these are common.
  - Claim these form a path covering:
    - There are at most f faulty processes.
    - Each path contains $f + 1$ labels ending with $f + 1$ distinct indices.
    - So at least one of these labels ends with a nonfaulty process index.



Agreement

- Lemma 4: If there's a common path covering of the subtree rooted at any node x, then x is common
- Proof:
  - By induction, from the leaves up.
  - "Common-ness" propagates upward.

- Lemmas 3 and 4 together imply that the root is common.
- So all nonfaulty processes get the same newval(λ).
- Yields Agreement.

Complexity Bounds

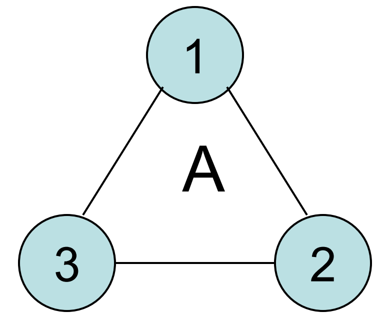
- As for EIG for stopping agreement:
  - Time: $f + 1$
  - Communication: $O(n^{f+1})$
- But now, also requires $n > 3f$ processors

#Processors for Byzantine Agreement

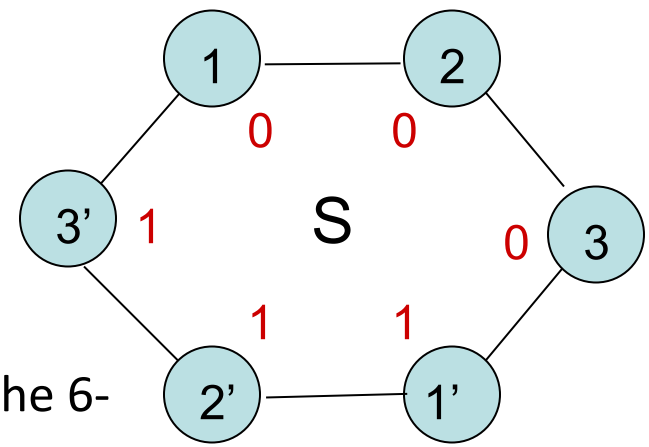
- $n > 3f$ is necessary!
  - Holds for any n-node (undirected) graph.
  - For graphs with low connectivity, may need even more processors.
  - Number of failures that can be tolerated for Byzantine agreement in an undirected graph G has been completely characterized, in terms of number of nodes and connectivity.

- Theorem 1: 3 processes cannot solve BA with 1 possible failure.

Proof (3 vs. 1 BA)

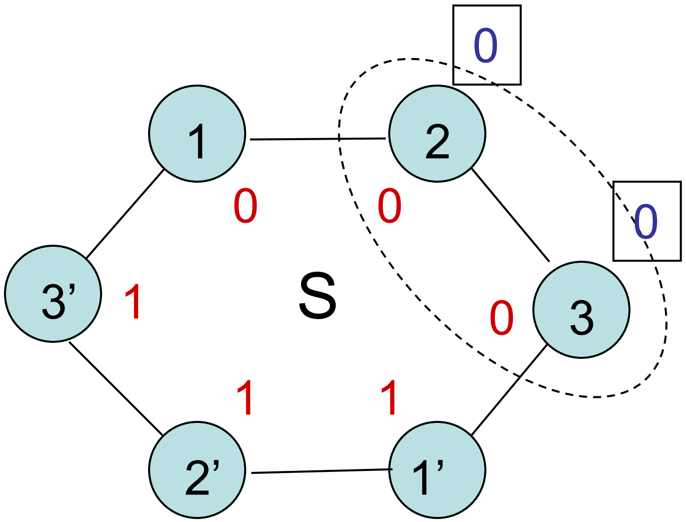


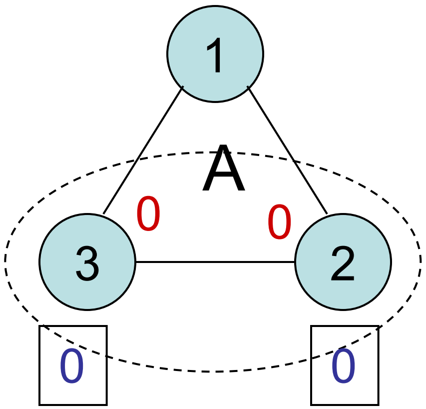
- By contradiction. Suppose algorithm A, consisting of procs 1, 2, 3, solves BA with 1 possible fault.
- Construct new system S from 2 copies of A, with initial values:
- What is S?
  - A synchronous system of some kind.
  - Not required to satisfy any particular correctness conditions.
  - Not necessarily a correct BA algorithm for the 6-node ring.
  - Just a synchronous system, which runs and does something.
  - We'll use it to get our contradiction.



Proof (3 vs. 1 BA)

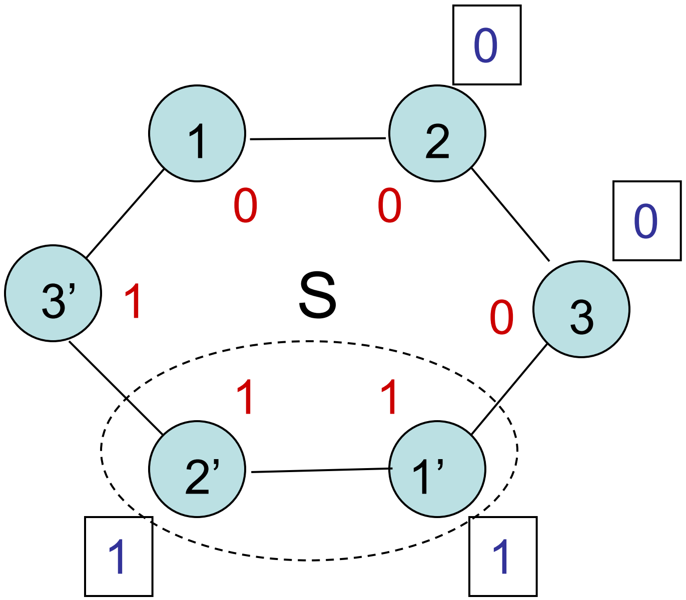
- Consider 2 and 3 in S:
- Looks to them like:
  - They're in A, with a faulty process 1.
  - 1 emulates 1'-2'-3'-1 from S.
- In A, 2 and 3 must decide 0
- So by indistinguishability, they decide 0 in S also.

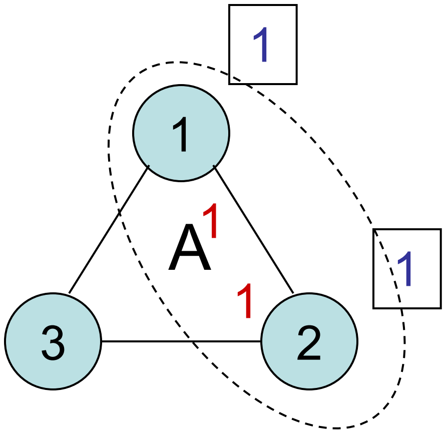




Proof (3 vs. 1 BA)

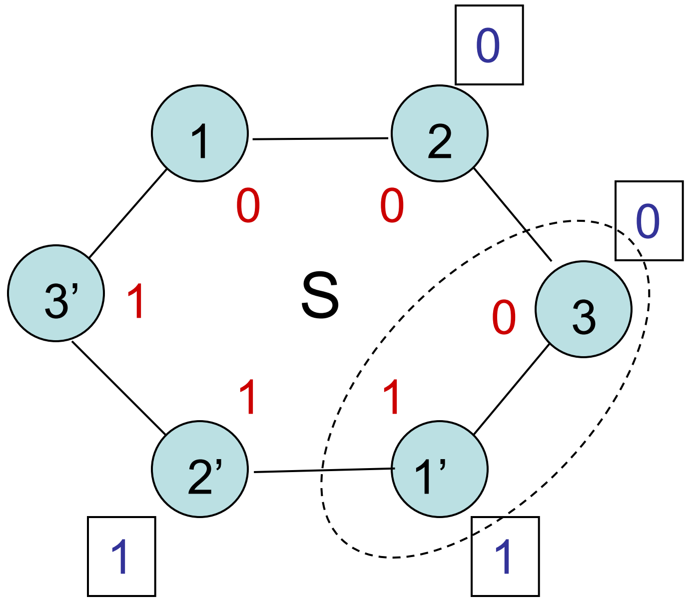
- Now consider 1’ and 2’ in S.
- Looks to them like:
  - They’re in A with a faulty process 3.
  - 3 emulates 3’-1-2-3 from S.
- They must decide 1 in A, so decide 1 in S also.



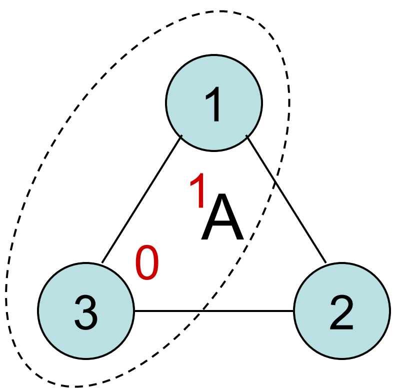


Proof (3 vs. 1 BA)

- Finally, consider 3 and 1’ in S:
- Looks to them like:
  - They’re in A, with a faulty process 2.
  - 2 emulates 2’-3’-1-2 from S.
- In A, 3 and 1 must agree
- So by indistinguishability, 3 and 1’ agree in S also.

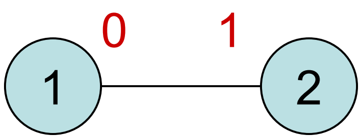


- But we already know that process 1’ decides 1 and process 3 decides 0, in S.
- Contradiction!



Impossibility for $n = 3f$

- Theorem 2: n processes can't solve BA, if $n \leq 3f$.
- Proof:
  - Similar construction, with f processes treated as a group.
  - Or, can use a reduction:
    - Show how to transform a solution for $n \leq 3f$ to a solution for 3 vs. 1.
    - Since 3 vs. 1 is impossible, we get a contradiction.

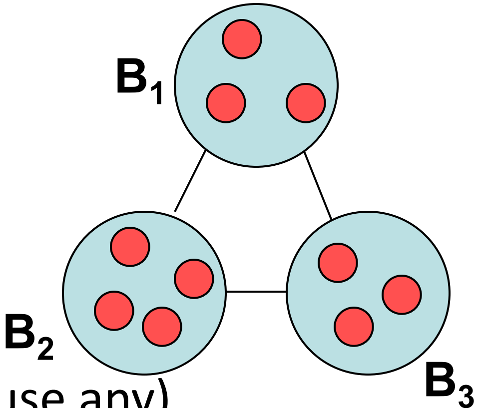


- Consider $n = 2$ as a special case:
  - $n = 2, f = 1$
  - Each could be faulty, requiring the other to decide on its own value.
  - Or both nonfaulty, which requires agreement, contradiction.

- So from now on, assume $3 \leq n \leq 3f$.
- Assume a Byzantine Agreement algorithm A for (n, f).
- Transform it to a BA algorithm B for $(3,1)$.

Transforming A to B

- Algorithm:
  - Partition A-processes into groups ℓ_1, ℓ_2, ℓ_3, where $1 \leq |\ell_1|, |\ell_2|, |\ell_3| \leq f$.
  - Each B_i process simulates the entire ℓ_i group.
  - B_i initializes all processes in ℓ_i with B_i's initial value.
  - At each round, B_i simulates sending messages:
  - If any simulated process decides, B_i decides the same (use any).



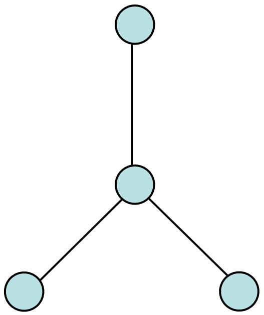
- Show B satisfies correctness conditions:
  - Consider any execution of B with at most 1 fault.
  - Simulates an execution of A with at most f faults.
  - Correctness conditions must hold in the simulated execution of A.
  - Show these all carry over to B's execution.

B's correctness

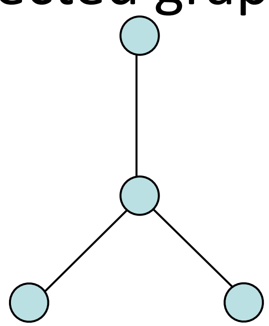
- *Termination:*
  - If B_i is nonfaulty in B, then it simulates only nonfaulty processes of A (at least one).
  - Those terminate, so B_i does also.
- *Agreement:*
  - If B_i, B_j are nonfaulty processes of B, they simulate only nonfaulty processes of A.
  - Agreement in A implies all these agree.
  - So B_i, B_j agree.
- *Validity:*
  - If all nonfaulty processes of B start with v, then so do all nonfaulty processes of A.
  - Then validity of A implies that all nonfaulty A processes decide v, so the same holds for B.

General Graphs and Connectivity Bounds

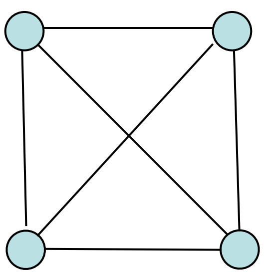
- $n > 3f$ isn't the whole story:
  - 4 processes, can't tolerate 1 fault:



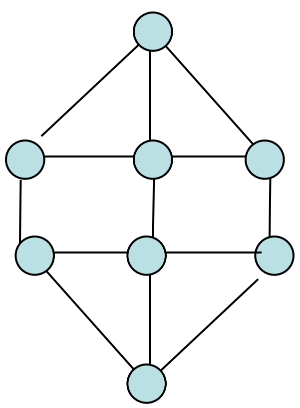
- Theorem 3: BA is solvable in an n-node graph G, tolerating f faults, if and only if both of the following hold:
  - $n > 3f$, and
  - $\mathrm{conn}(G) > 2f$.
- $\mathrm{conn}(G)$ = minimum number of nodes whose removal results in either a disconnected graph or a 1-node graph.
- Examples:



conn = 1



conn = 3



conn = 3

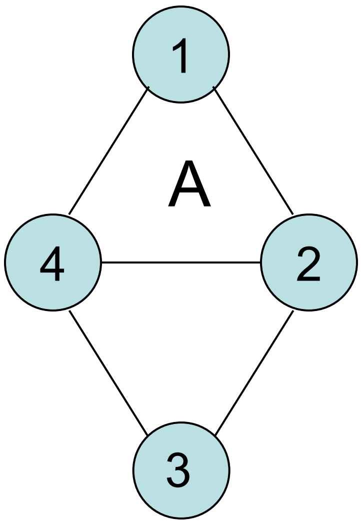
Proof: "If" Direction

- Theorem 3: BA is solvable in an n-node graph G, tolerating f faults, if and only if $n > 3f$ and $\mathrm{conn}(G) > 2f$.
- Proof ("if"):
  - Suppose both hold.
  - Key is to emulate reliable communication from any node i to any other node j.
  - Rely on Menger's Theorem, which says that a graph is c-connected (that is, has conn $\geq c$) if and only if each pair of nodes is connected by $\geq c$ node-disjoint paths.
  - Since $\mathrm{conn}(G) \geq 2f + 1$, we have $\geq 2f + 1$ node-disjoint paths between i and j.
  - To send message, send on all these paths (assumes graph is known).
  - Majority must be correct, so take majority message.

Proof: "Only If" Direction

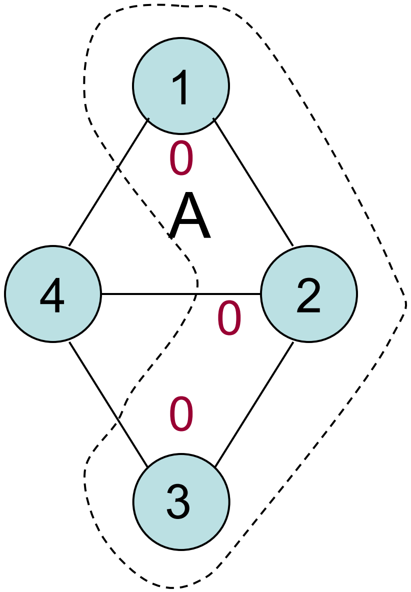
- Theorem 3: BA is solvable in an n-node graph G, tolerating f faults, if and only if $n > 3f$ and $\mathrm{conn}(G) > 2f$.

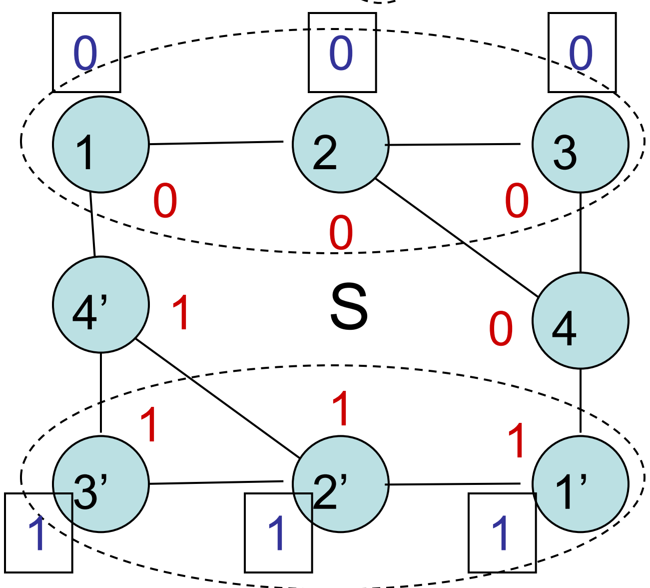
- Proof ("only if"):
  - We already showed $n > 3f$; remains to show $\mathrm{conn}(G) > 2f$.
  - Show key idea with simple case, $\mathrm{conn} = 2, f = 1$.
  - Canonical example:
    - Disconnect 1 and 3 by removing 2 and 4
  - Proof by contradiction.
  - Assume some algorithm A that solves BA in this canonical graph, tolerating 1 failure.



Proof (conn=2, 1 failure)

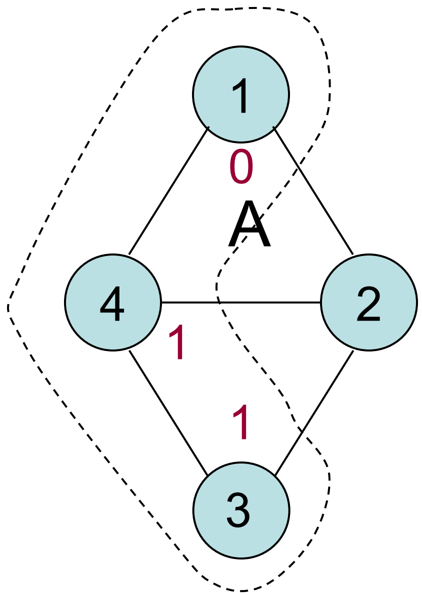
- Now construct S from two copies of A.
- Consider 1, 2, and 3 in S:
  - Looks to them like they're in A, with a faulty process 4.
  - In A, 1, 2, and 3 must decide 0
  - So they decide 0 in S also.
- Similarly, 1', 2', and 3' decide 1 in S.

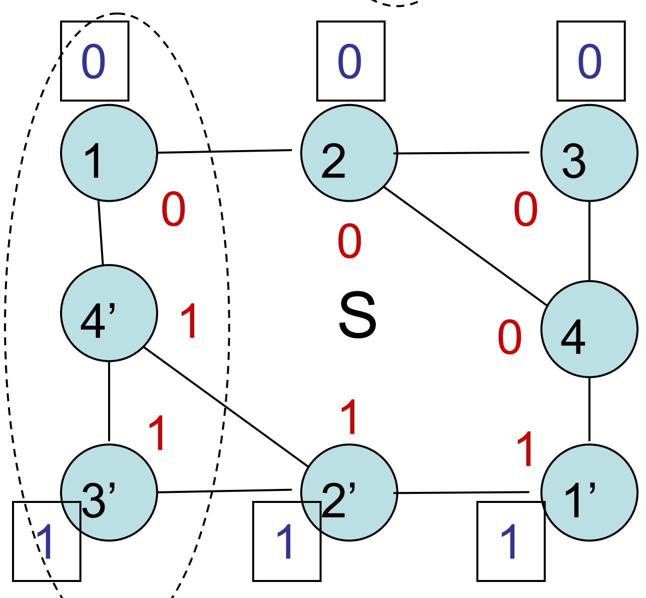




Proof (conn=2, 1 failure)

- Finally, consider 3', 4', and 1 in S:
  - Looks to them like they're in A, with a faulty process 2.
  - In A, they must agree, so they also agree in S.
  - But 3' decides 0 and 1 decides 1 in S, contradiction.
- Therefore, we can't solve BA in canonical graph, with 1 failure.
- As before, can generalize to $conn(G) \leq 2f$, or use a reduction.





Byzantine Processor Bounds

- The bounds $n > 3f$ and $\mathrm{conn} > 2f$ are fundamental for consensus-style problems with Byzantine failures.
- Same bounds hold, in synchronous settings with f Byzantine faulty processes, for:
  - Byzantine Firing Squad synchronization problem
  - Weak Byzantine Agreement
  - Approximate agreement
- Also, in timed (partially synchronous settings), for maintaining clock synchronization.
- Proofs used similar methods.