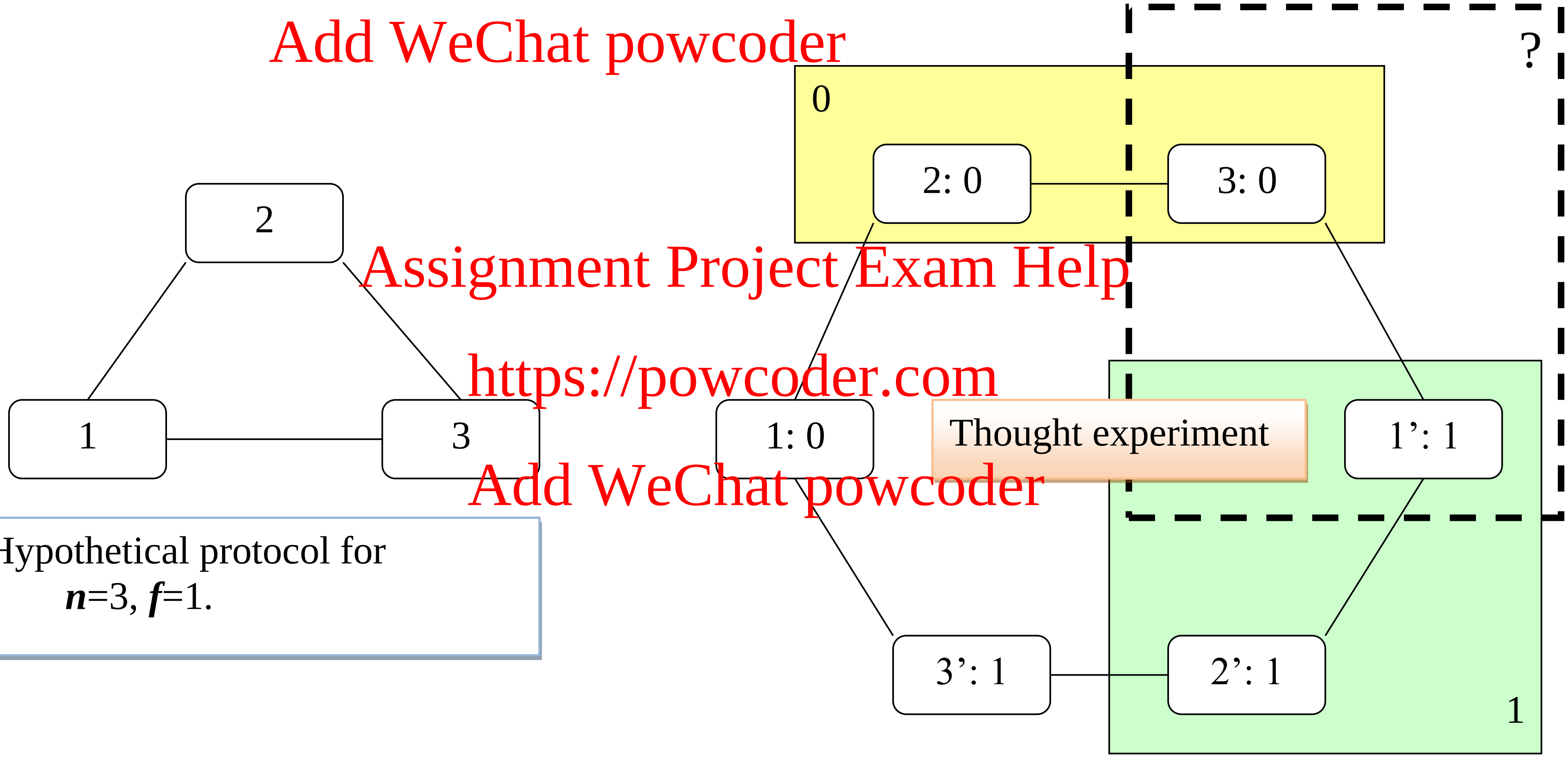
Byzantine Agreement – A Few Theoretical Results.

Number of Processes for Byz – Lemma 6.26



Hypothetical protocol for
n=3, ***f***=1.

Conclusion: no algorithm for ***n***=3, ***f***=1.

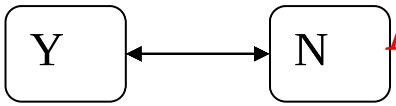
THEOREM 6.27

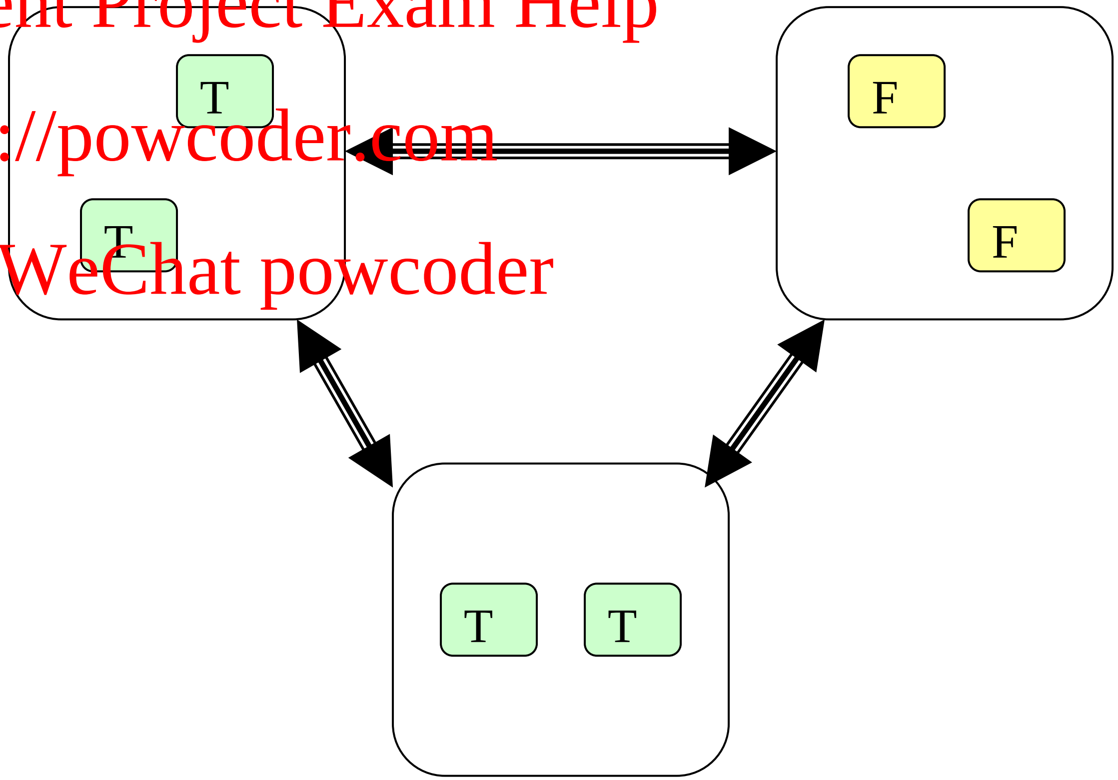
No solution for $2 \le n \le 3f$

for $3 \le n \le 3f$

- 3 “subnets” with at most f processes in each
- we assume that there is an algorithm that can solve the Byz agreement for such an n, and we construct an algorithm that can solve the problem for 3 processes,
  - contradiction (with lemma 6.26)

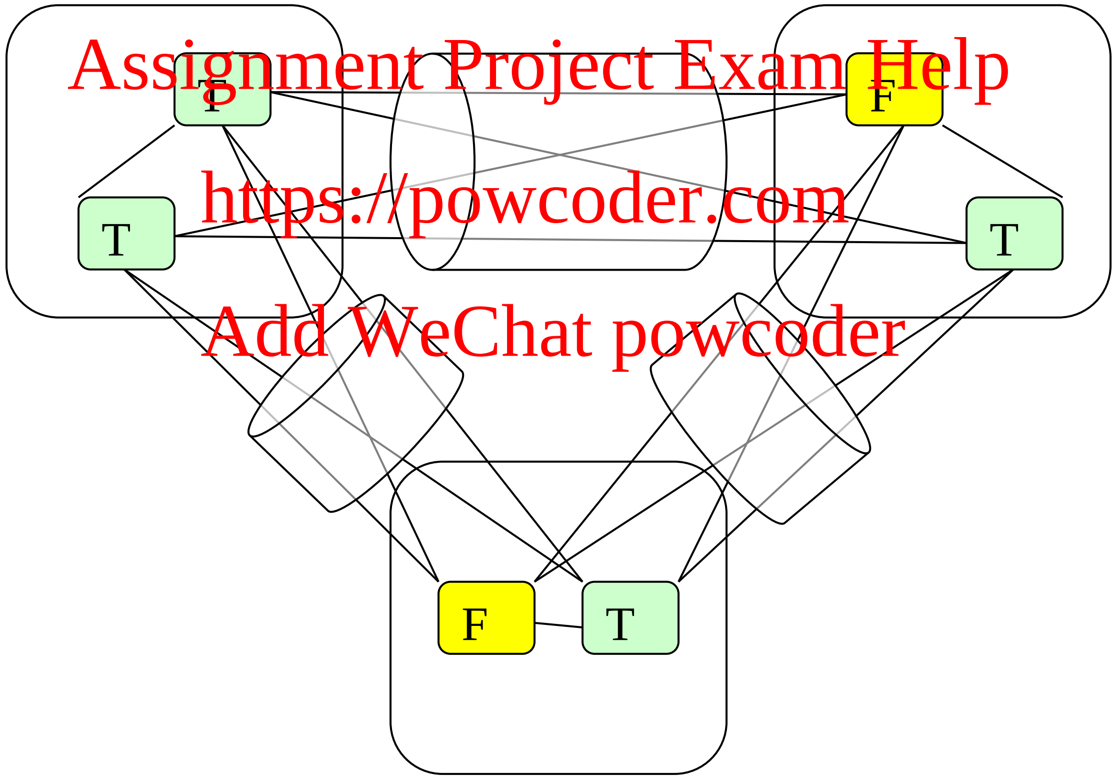
$n = 2$



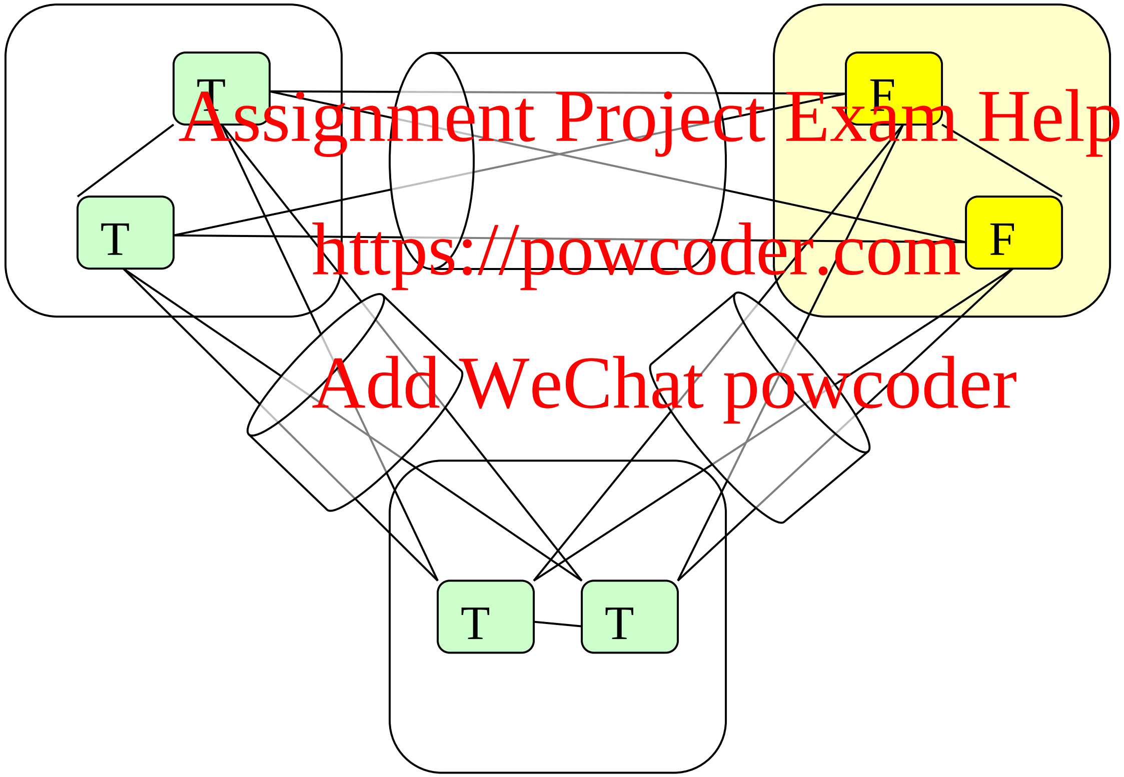


Proof by contradiction

- We assume that the ***n*** "small" processes can solve the Byz problem, if at most ***f*** are faulty – regardless how these ***f*** faulty nodes are distributed
- These "small" processes are totally unaware that they are now clustered into 3 "large" nodes, connected by 3 "large" channels



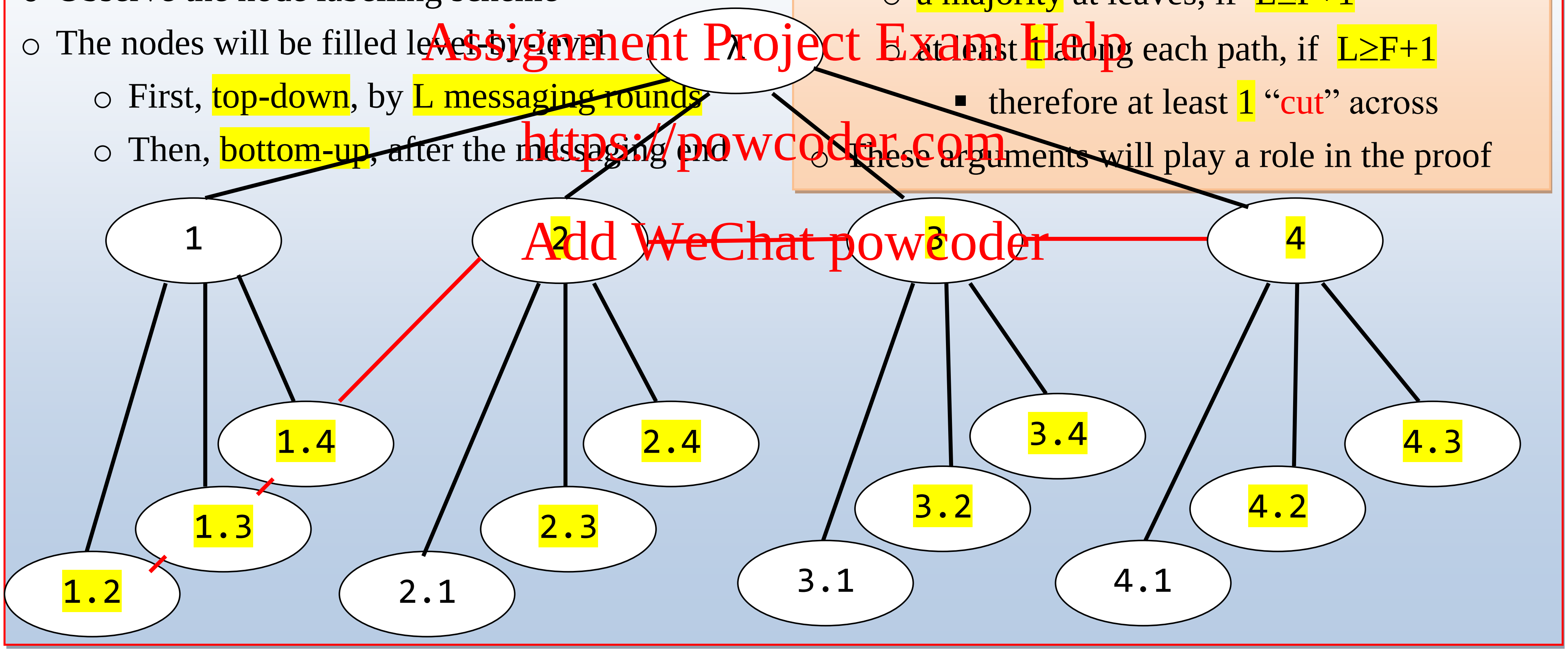
- Replacing an arbitrary one "large" node by a "large" Byzantine node is tantamount to replacing its content by the same number of "small" faulty nodes (w/ their channels)
- Not doing this will be easily detected, by the others, as wrongly formatted messages



AN EIG TREE WITH N=4 (# OF PROCESSES) AND L=2 (LEVELS)

- Level #1 : 1 group with N=4 siblings
- Level #2 : 4 groups with N-1=3 siblings each
  - Level #L : each group has N-L+1 siblings
- For Byz agreement, L=F+1, *here* F=1, L=2
- Observe the node labelling scheme
- The nodes will be filled level-by-level
  - First, top-down, by L messaging rounds
  - Then, bottom-up, after the messaging end

- Consider that process #1 is "faulty"
  - but #2, #3, #4 are "non-faulty"
- Observe the distribution of labels ending in one of 2,3,4
  - a majority at leaves, if L≤F+1
  - at least 1 along each path, if L≥F+1
    - therefore at least 1 "cut" across
- These arguments will play a role in the proof



Tree nodes with labels ending in the number of a non-faulty process play a critical role!

- Examples (prev. slide): 2, 1.2, 2.3, 2.4, ...

The number of levels, L, is a well-chosen critical number!

- If $L \geq F+1$, then each root-to-leaves branch contains at least one such tree node

  Except λ, there are F+1 tree nodes (in each branch), but only F faulty processes (and labels cannot contain duplicates)

- If $L \leq F+1$, then each sibling group (including at the last level) has a strict majority of such tree nodes

  The smallest sibling group (at the leaves level), has N-L+1 tree nodes, thus

  $N-L+1 = N-(F+1)+1 = N-F \geq 3F+1-F = 2F+1$ tree nodes at least,

  out of which at most F can end in numbers of faulty processes

- The algorithm chooses $L = F+1$!

COMMON NODES (THOSE ON WHICH A COMMON DECISION IS TAKEN)

DEFINITION:

A tree node with label x is ***common*** if it has the same **newval**() across all ***non-faulty*** processes, i.e.,

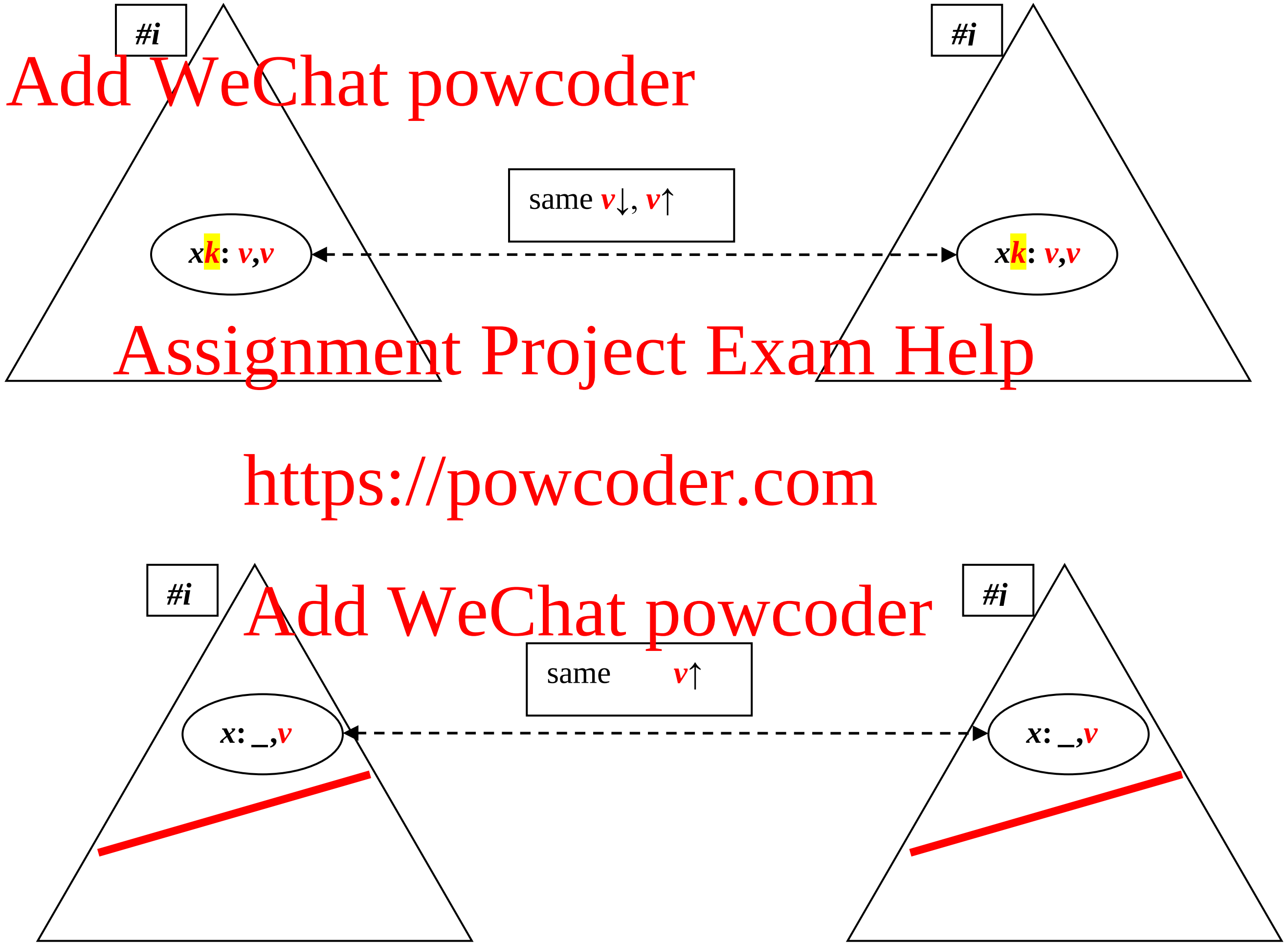
newval$(x)_i$ = **newval**$(x)_j$ for all i, j that are ***non-faulty*** processes

Note: **val**$(x)_i$ and **val**$(x)_j$ may or may not be equal

- A set of tree nodes which contains at least **one tree node on each (root-to-leaves) path** is called a ***path covering***.
- The red "**cut**" across the previous sample EIG tree is a ***path covering***.

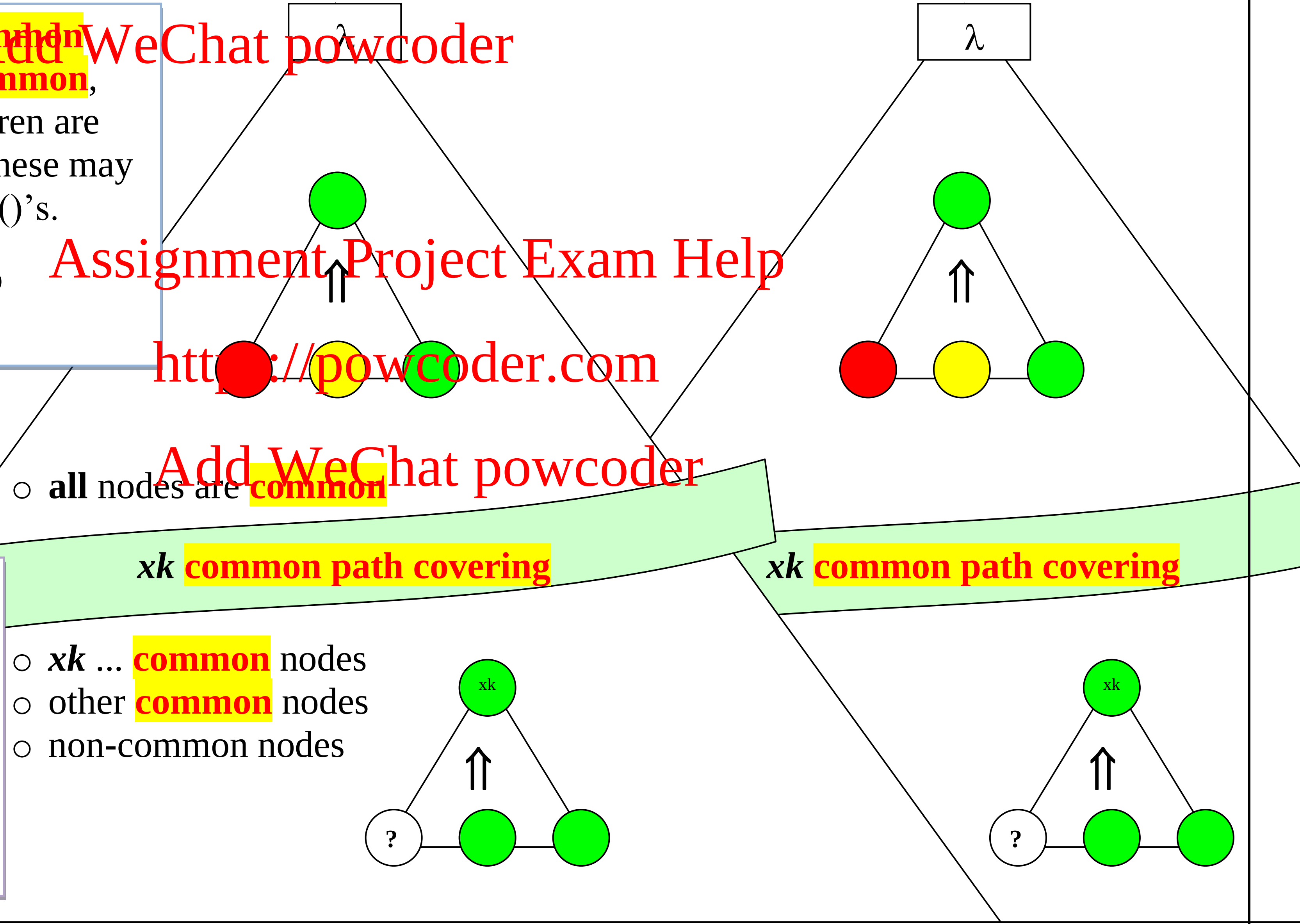
- A ***common path covering*** is a ***path covering*** where all tree nodes are ***common***.
- As we will see, the red "**cut**" across the previous sample EIG tree is also a ***common path covering***.

THE ESSENCE OF THE PROOF – BIRD'S EYE VIEW



THE ESSENCE OF THE PROOF – MORE DETAILS

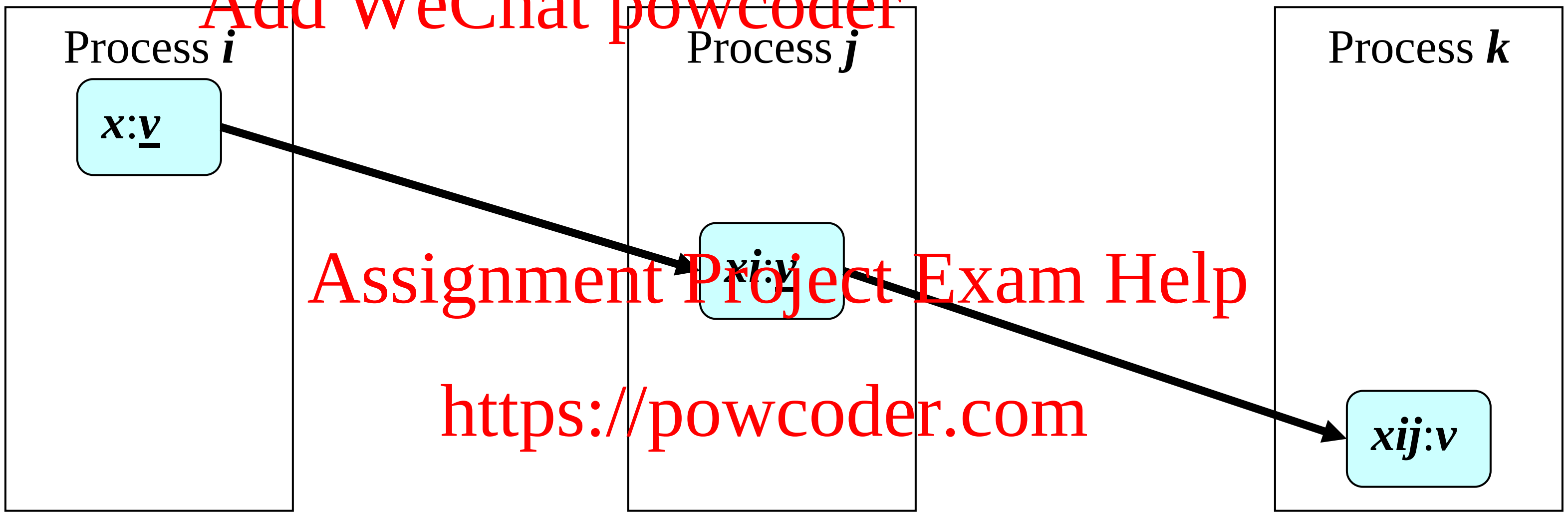
- **All** nodes above a **common path covering** are **common**, because **all** their children are **common** – although these may have different **newval**()'s.
- Thus the ***root*** λ is also ***common***.



- **all** nodes are **common**
- *xk* … **common** nodes
- other **common** nodes
- non-common nodes

- All ***xk*** nodes are **common** because they have a **strict majority** of ***xkl xkl'*** … **common** children sharing the **same** **val**() and **newval**()

TOP-DOWN MESSAGING IN THE EIG PROTOCOL (RECALL)

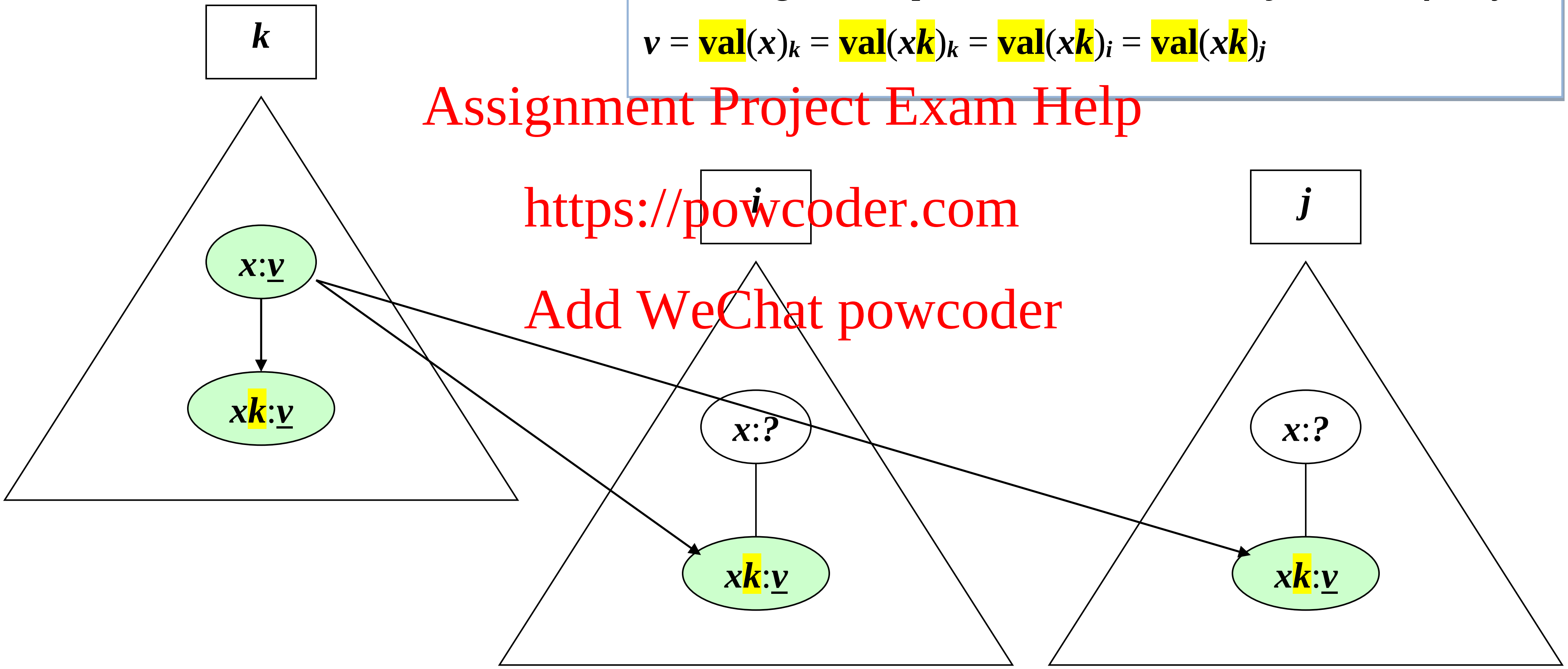


assume that ***x*** does not contain ***i, j***

LEMMA 6.15

Assuming that all processes here, i.e., ***k***, ***i***, ***j***, are ***non-faulty***

v = **val**(x)$_k$ = **val**(x***k***)$_k$ = **val**(x***k***)$_i$ = **val**(x***k***)$_j$



LEMMA 6.16

All nodes with labels of the form xk, where k is number of a ***non-faulty*** process, have the same **val**() and **newval**() across all ***non-faulty*** processes, i.e.,

$\mathbf{newval}(x\boldsymbol{k})_{\boldsymbol{i}} = \mathbf{val}(x\boldsymbol{k})_{\boldsymbol{i}} = \mathbf{val}(x\boldsymbol{k})_{\boldsymbol{j}} = \mathbf{newval}(x\boldsymbol{k})_{\boldsymbol{j}}$ for all ***i***, ***j*** that are ***non-faulty*** processes

As a corollary, all such nodes are ***common***!

In fact, they are “more than common”, as their **val**() attributes are also equal, to the same value

The proof follows on next slides

As we will further see:

- The condition of lemma 6.16 is ***not*** necessary, i.e., there could also be other ***common*** nodes with labels of the form ***xk’***, where ***k’*** is a faulty process.
- All first level nodes are **common**! This result ensures a common decision at the root λ.

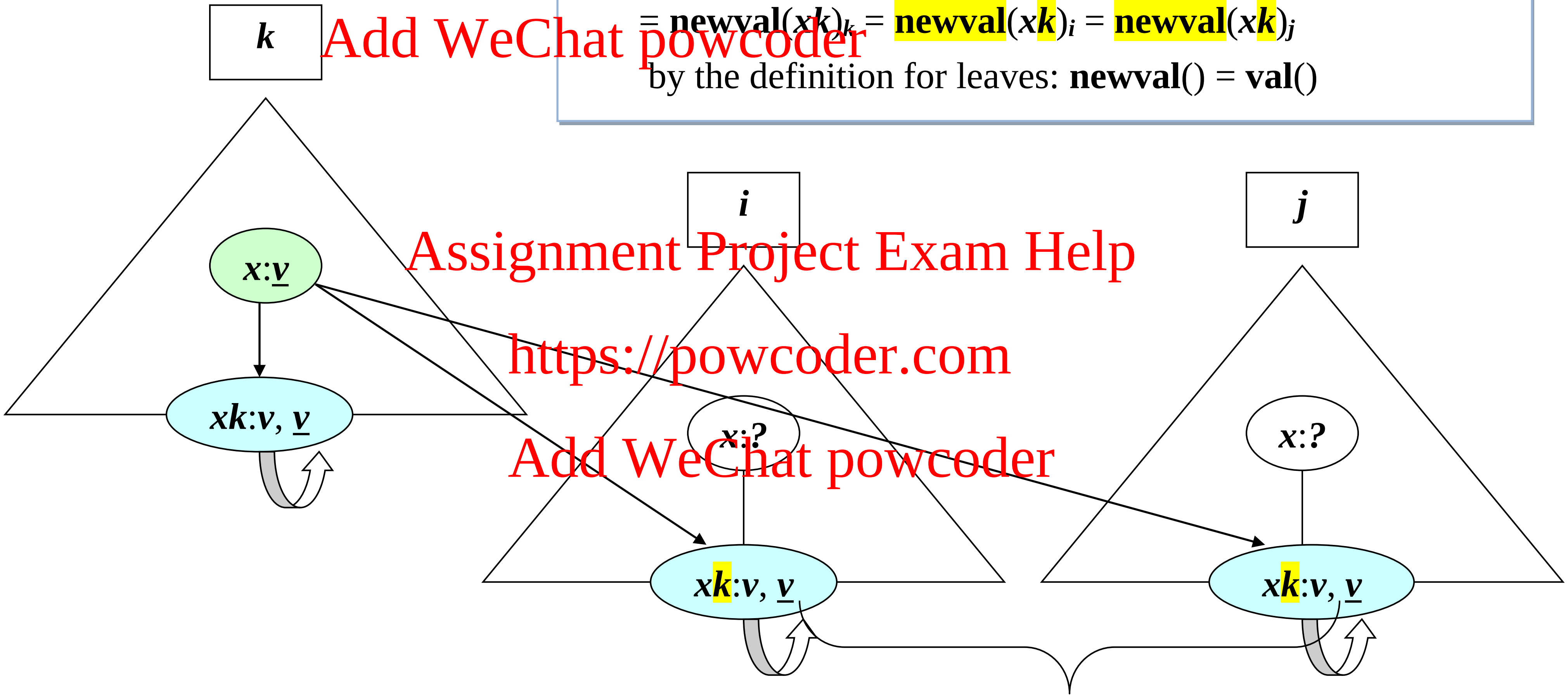
LEMMA 6.16 FOR LEAVES

- Assuming that all processes here, i.e., ***k***, ***i***, ***j***, are ***non-faulty***

  $\boldsymbol{v} = \mathbf{val}(\boldsymbol{x})_{\boldsymbol{k}} = \mathbf{val}(\boldsymbol{xk})_{\boldsymbol{k}} = \mathbf{val}(\boldsymbol{xk})_{\boldsymbol{i}} = \mathbf{val}(\boldsymbol{xk})_{\boldsymbol{j}} =$

  $= \mathbf{newval}(\boldsymbol{xk})_{\boldsymbol{k}} = \mathbf{newval}(\boldsymbol{xk})_{\boldsymbol{i}} = \mathbf{newval}(\boldsymbol{xk})_{\boldsymbol{j}}$

  by the definition for leaves: **newval**() = **val**()



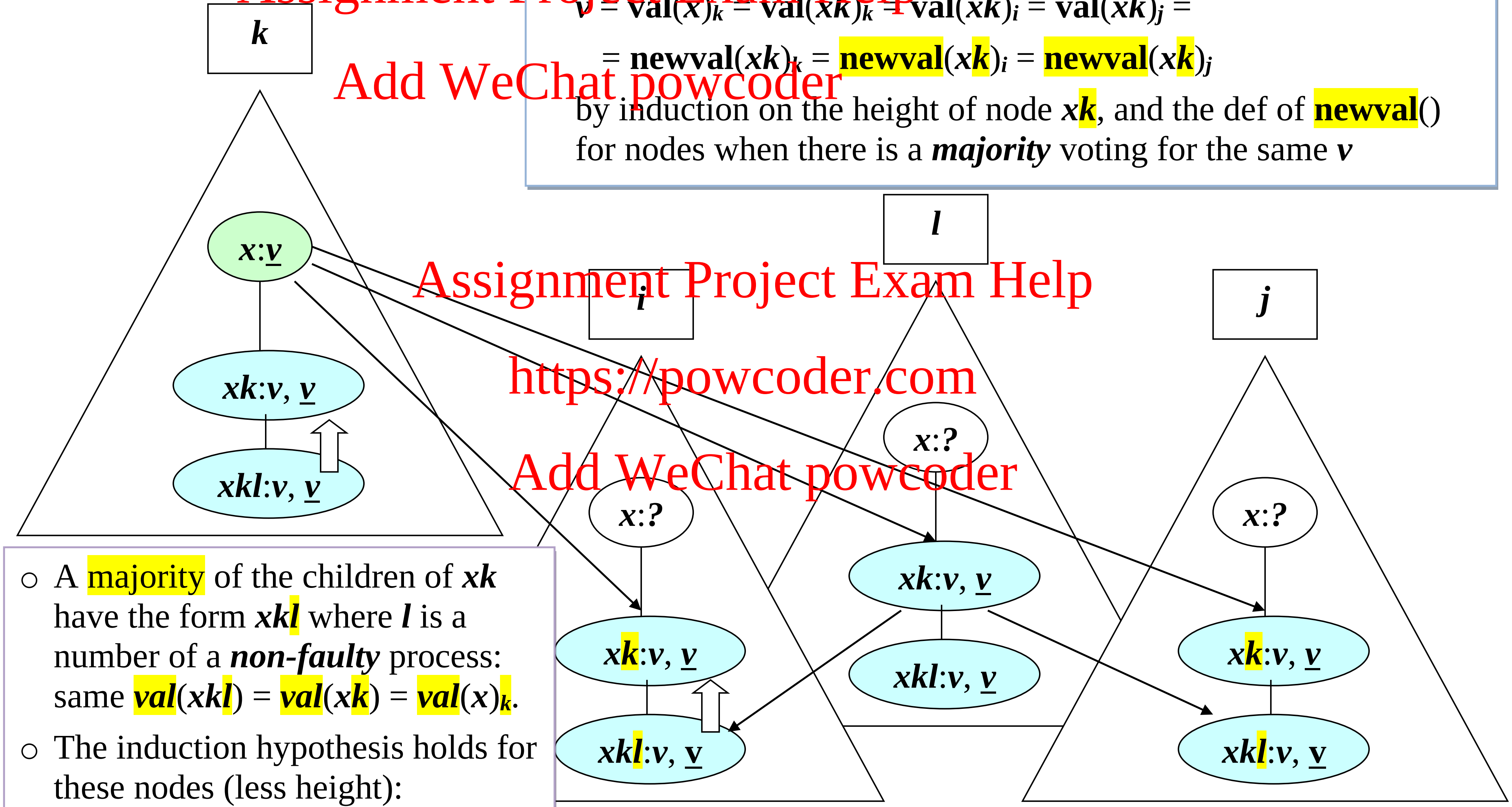
LEMMA 6.16 FOR NON-LEAVES

- Assuming all processes here, i.e., ***k***, ***l***, ***i***, ***j***, are ***non-faulty***

  v = **val**(*x*)$_k$ = **val**(*xk*)$_k$ = **val**(*xk*)$_i$ = **val**(*xk*)$_j$ =

  = **newval**(*xk*)$_k$ = **newval**(*xk*)$_i$ = **newval**(*xk*)$_j$

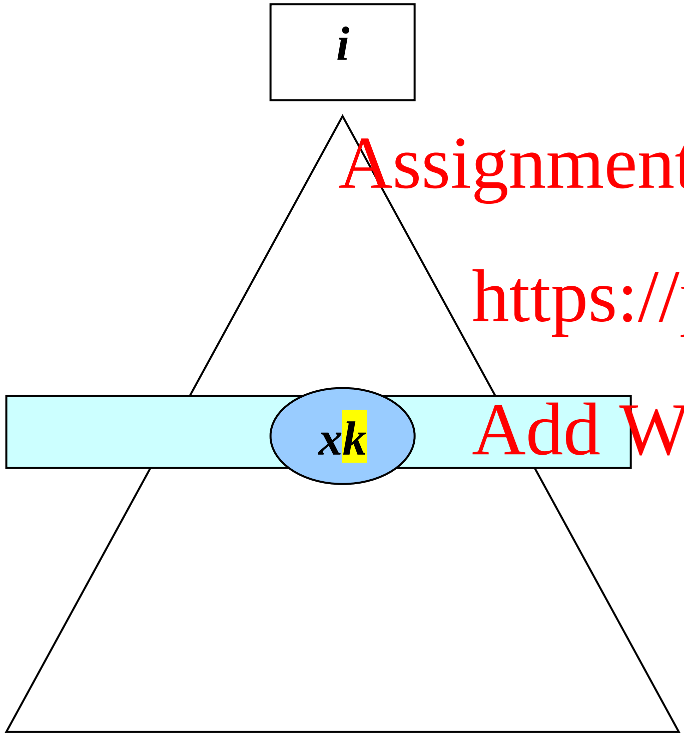
  by induction on the height of node ***xk***, and the def of **newval**() for nodes when there is a ***majority*** voting for the same ***v***



- A majority of the children of ***xk*** have the form ***xkl*** where ***l*** is a number of a ***non-faulty*** process: same ***val***(***xkl***) = ***val***(***xk***) = ***val***(***x***)$_k$.
- The induction hypothesis holds for these nodes (less height):

  same ***newval***(***xkl***)

LEMMA 6.18

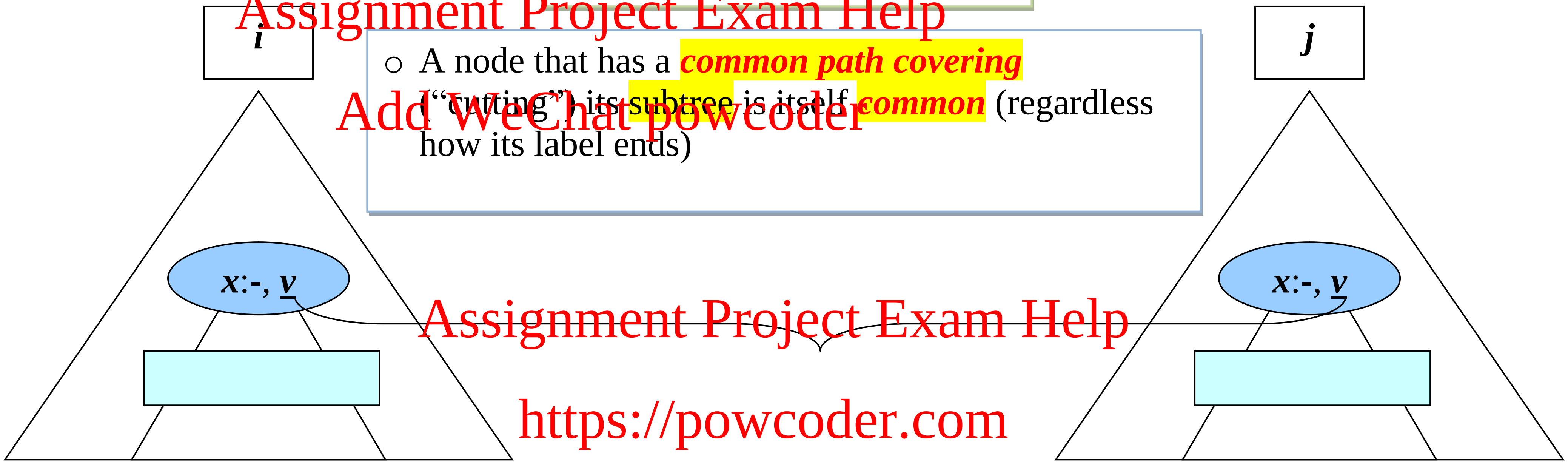


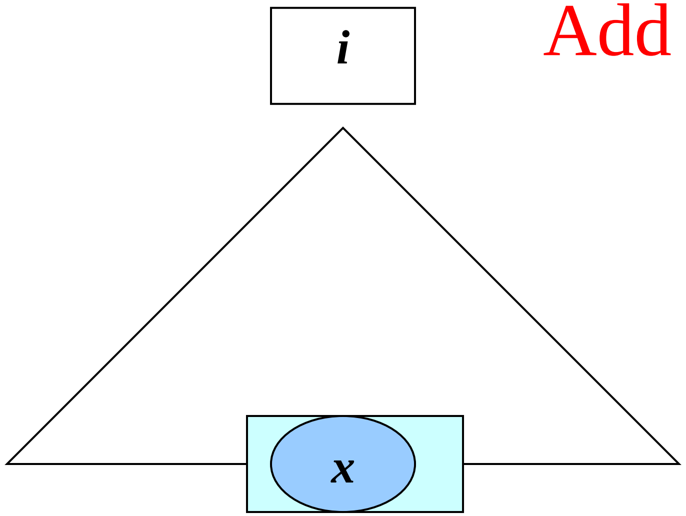
There is a ***common path covering*** the whole EIG tree!

- How to build such a common path covering?
- Top down on each branch, until we find a tree node which ends with the label of a non-faulty process
  - Labels that end with ***k***, where ***k*** is a ***non-faulty*** process (there is such a label on each branch)
- As we have seen, all such tree nodes have common **newval**()
- Thus, this is ***common path covering*** of the EIG tree!

LEMMA 6.19 – FOR LEAVES

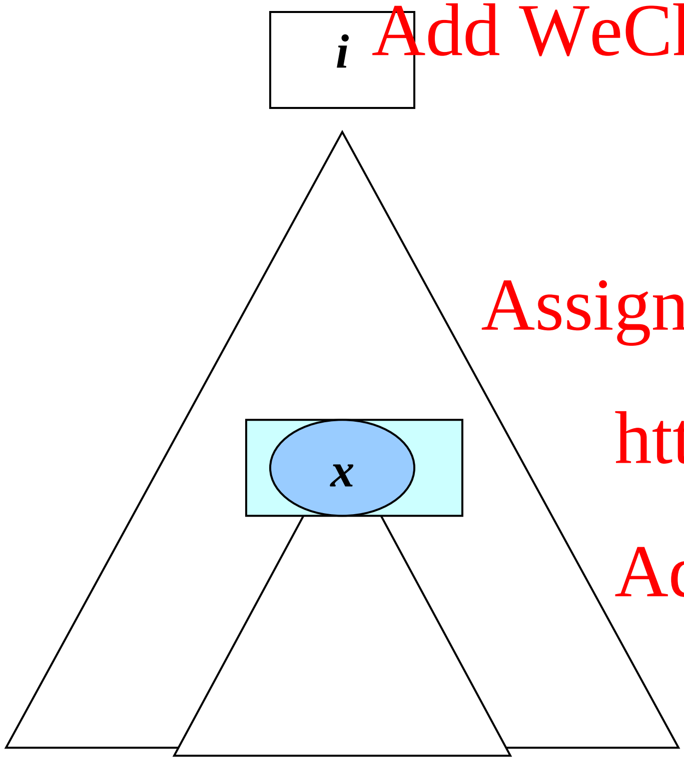
- A node that has a ***common path covering*** ("cutting") its subtree is itself ***common*** (regardless how its label ends)

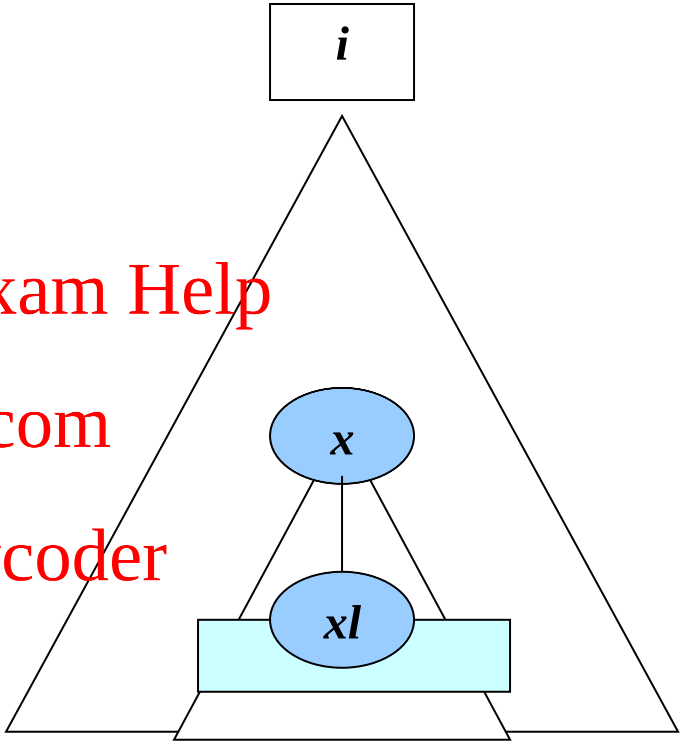




- Base case for leaves
- The subtree of a leaf ***x*** is the leaf itself.
- Thus any leaf ***x*** which is part of the ***common path covering***… is ***common***, qed.

LEMMA 6.19 – FOR NON-LEAVES

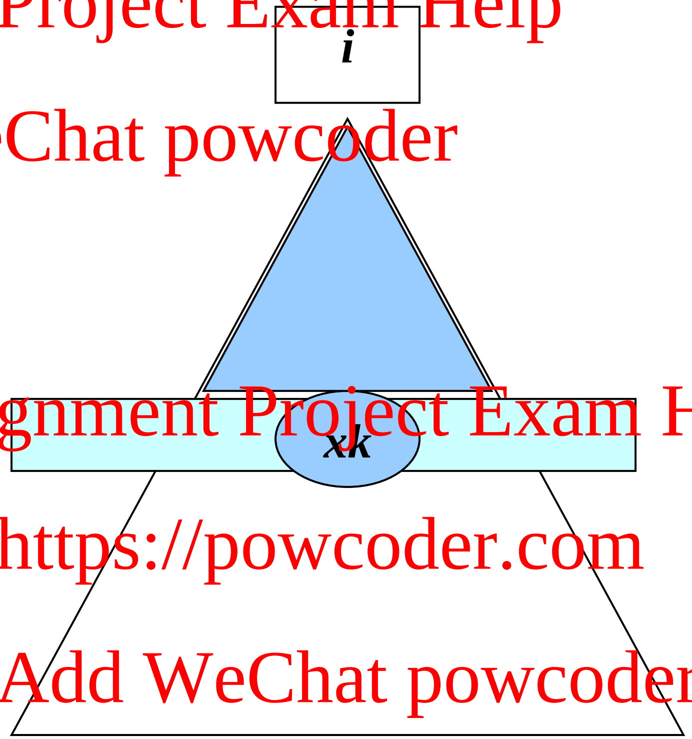




- If ***x*** is itself part of the ***common path covering***, then, well, it is ***common***, qed.

- **Otherwise**, all its children - such as ***xl*** (no matter if ***l*** is or not faulty) will be ***common*** by induction (height-1)
- And then ***x*** will be ***common*** by the definition of **newval**()

LEMMA 6.20



- All nodes above a ***common path covering*** the tree are also ***common***, including the root λ.
- Thus, they have the same **newval**() across all non-faulty processes
- **Agreement**!

- We almost proved that the EIG algorithm solves the Byz agreement problem.
- We have now the **agreement**
- What is still missing?

- **Termination** is straightforward: the protocol stops after F+1 messaging rounds

- What else?

LEMMA 6.17

Validation!

Assume that all non-faulty processes start with the same initial value $\boldsymbol{v}$

Proof - using **LEMMA 6.16**

$\mathbf{newval}(x\boldsymbol{k})_i = \mathbf{val}(x\boldsymbol{k})_i = \mathbf{val}(x\boldsymbol{k})_j = \mathbf{newval}(x\boldsymbol{k})_j$, for all non-faulty $\boldsymbol{k}, \boldsymbol{i}, \boldsymbol{j}$

In particular

$\mathbf{newval}(\boldsymbol{k})_i = \mathbf{val}(\boldsymbol{k})_i = \mathbf{val}(\boldsymbol{k})_j = \mathbf{newval}(\boldsymbol{k})_j = \boldsymbol{v}$, for all non-faulty $\boldsymbol{k}, \boldsymbol{i}, \boldsymbol{j}$

Thus, all first level nodes corresponding to non-faulty processes share the same **newval**() = $\boldsymbol{v}$

And they form a strict majority, so the root decision will be $\boldsymbol{v}$ – **validity** !

THEOREM 6.21

The EIG algorithm solves the Byz agreement problem!