Synchronous Byzantine Agreement Revisited:

Work in Progress

Achour MOSTEFAOUI

in collaboration with **Darek KOWALSKI**

Achour.Mostefaoui@univ-nantes.fr

LINA, Université de Nantes, France

Synchronous Distributed System Model

- A set Π of n processes $\{p_1, \ldots, p_n\}$
- A bound on message transfer delays (message-passing)
- A bound on the time for a process to execute a computation step
- Some processes may exhibit a Byzantine behavior
- A process is *correct* in a run if it does not turn Byzantine Otherwise it is *faulty*
- At most t processes are faulty

The Byzantine Agreement Problem (1)

Each process p_i proposes a value v taken from a set V ($\mid V \mid \geq 2$) of proposable values.

Goal : Make processes decide a same value

- Termination: Every correct process eventually decides some value.
- Agreement: No two correct processes decide on different values.
- Validity: If all correct processes propose the same initial value v, then v is the only possible decision value.

The Byzantine Agreement Problem (2)

- Resilience: There exists a solution to the BA problem only if $n > 3t$ (Pease, Shostack and Lamport, 1980).
- Time complexity: Any correct BA algorithm requires at least $t + 1$ rounds in the worst case if t processes may be faulty (Fischer and Lynch, 1982).
- Bit complexity: No upper bound is known. The only known bit complexity result is $\Omega(n^2)$ (Dolev and Reischuck, 1985).

... Some history

Protocol	n	rounds	comm.
[PSL] 80	$3t+1$	$t+1$	$\exp(n)$ *vs* $\Omega(n^2)$
[DFFLS,TPS] 82	$3t+1$	$2t+c$	$poly(n)$
[c] 85	$4t+1$	$t+t/d$	$O(n^d)$
[DRS,BD] 86	$\Omega(t^2)$	$t+1$	$poly(n)$
[BDDS] 87	$3t+1$	$t+t/d$	$O(n^d)$
[MW] 88	$6t+1$	$t+1$	$poly(n)$
[BGP1] 89	$4t+1$	$t+1$	$poly(n)$
[BG2] 91	$(3+\epsilon)t$	$t+1$	$poly(n).O(2^{1/\epsilon})$
[GM] 93	$3t+1$	$t+1$	$O(n^9)$

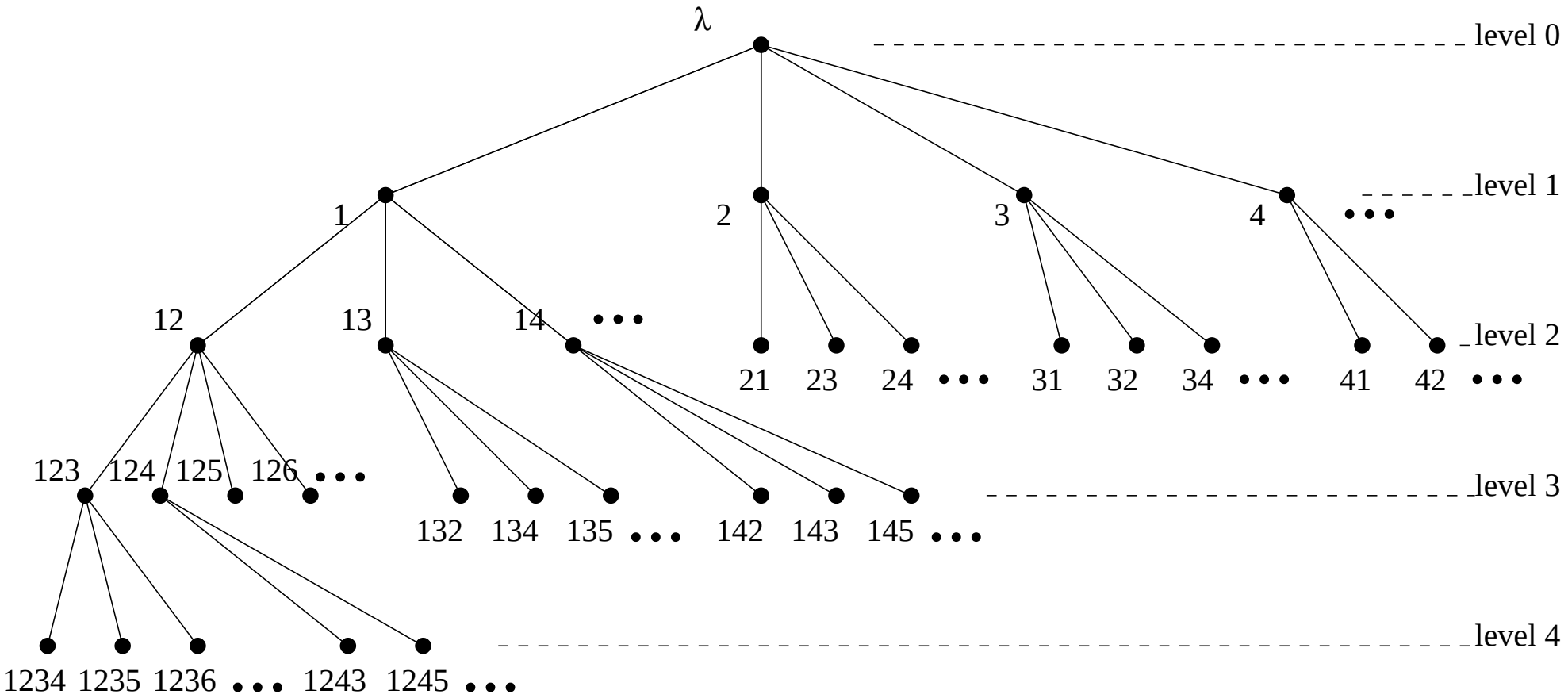
... and nothing since!

The Basic Solution: EIG_BYZ

- The first proposed solution (Pease, Shostack and Lamport in 1980).
- Meets lower bounds for time complexity and resilience.
- Very simple as reformulated by Bar-Noy et al. in 1987 using a data structure: EIG tree.
  - ⋆ Exponential Information Gathering tree structure
- Many proposed solutions rely on the same structure.
- Some parts of the tree are cut to reduce the communication bit complexity due to the inherent redundancy of the tree structure.
  - ⋆ Very complex solutions

Exponential Information Gathering Tree

- Each process manages an EIG tree data structure



EIG Protocol

- Each process manages an EIG tree with $t + 2$ levels
- Each node x of the tree:
  - ⋆ is labelled with a sequence of process ids
  - ⋆ is associated with two values $val(x)$ and $newval(x)$
- $val(x)$ is set during the successive rounds
  $val(ijk\ell)$ is a fourth hand value received from p_ℓ
- $newval(x)$ is set during the decision phase
  - ⋆ Leaf nodes: $newval(x)$ is set to $val(x)$
  - ⋆ Internal nodes: $newval(x)$ is set to a strict majority of the $newval$ of its child nodes or to a default value otherwise
- The decision value is the new value of the root λ

Elements of the proof

- $val(xk)_i = val(xk)_j$ for i, j and k correct processes
- $newval(xk)_i = val(xk)_i$ for i and k correct processes

Definition 1: A subset C of the nodes of a rooted tree is a *path covering* if each path of the tree contains at least one node in C.

Definition 2: A node is x common if for all correct processes i and j: $newval(x)_i = newval(x)_j$

- After $t + 1$ rounds:
  - ⋆ there is a common path covering of the EIG tree.
  - ⋆ if there is a common path covering rooted at x, then x is common.

The Proposed Protocol

- Each process manages an EIG tree with $t + 2$ levels
- Each node x of the tree:
  - ⋆ is labelled with a sequence of process ids
  - ⋆ is associated with three values $val(x)$, $cval(x)$and $newval(x)$
- $val(x)$ and $cval(x)$ are set during the successive rounds
- $newval(x)$ is set during the decision phase
- Processes no more exchange first, second, third ... hand values.
  - ⋆ They exchange information on suspicions.
  - ⋆ For this purpose they use a confirmation mechanism to detect cheating processes

The Proposed Protocol: Basics

- Confirmation mechanism (Srikanth and Toueg, 1987 for asynchronous systems)
  - ⋆ A sent value is echoed during the next round
  - ⋆ If it is not echoed enough times $(n - t)$ the sender is Byzantine
- The echoes are not themselves echoed
- Each process maintains a list of uncovered Byzantine processes
- Each message has two fileds: main data and echo

The Proposed Protocol

- Round 1: Processes exchange proposed values (there is no suspicion information available). These values are stored in $val(x)$ for each node x.
- Round 2: Processes exchange second hand values (there is still no suspicion information available). These values are stored in $cval(x)$ for each node x.
- During round 2, each process starts detecting cheating processes.
- Round 3: Processes exchange suspicion values and echo the echos of round 2.
- Starting from round 4: Processes exchange suspicion values and echo the suspicion information received during the previous round.

The Proposed Protocol: EIG Tree

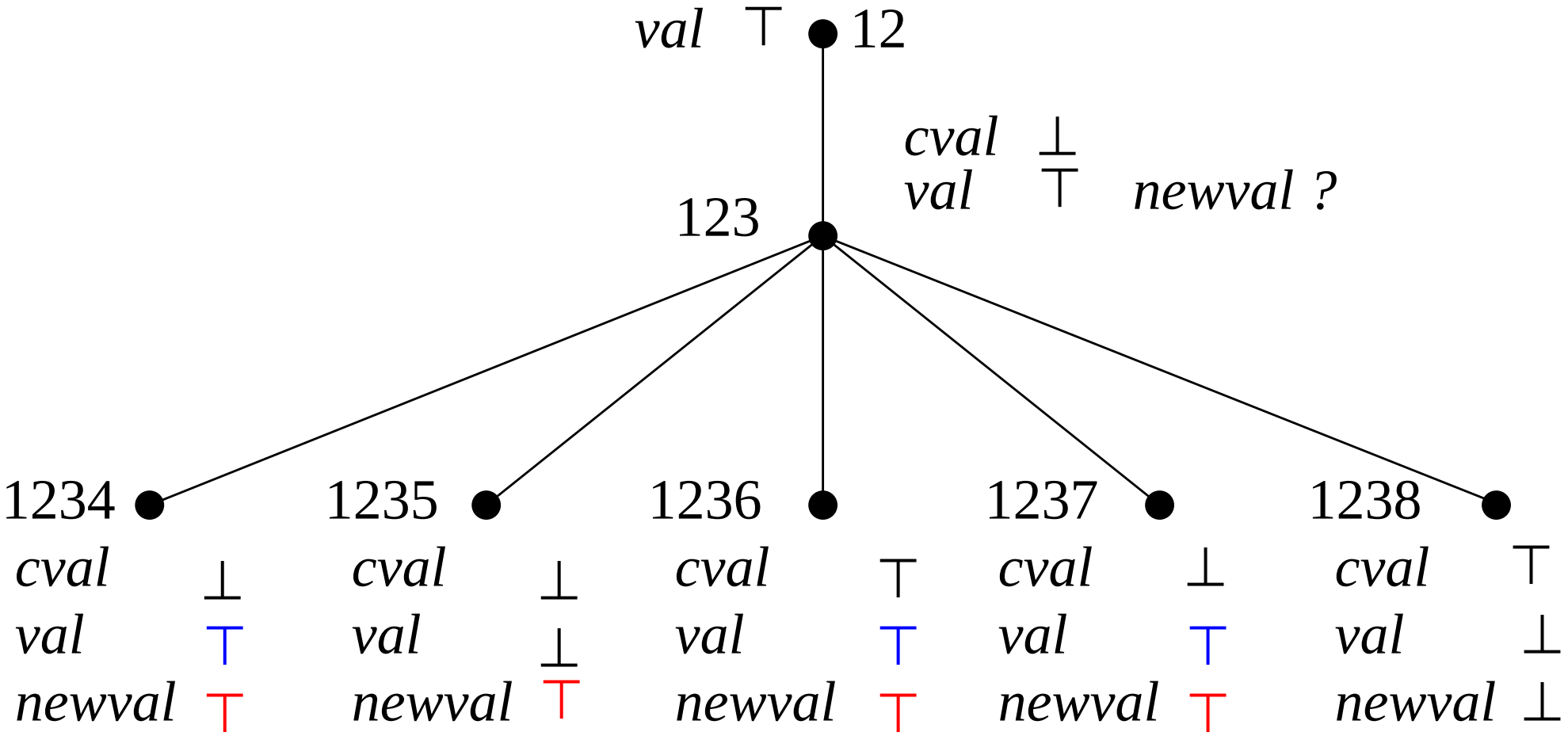
- When all the rounds are over, the processes set $val(x)$ and $cval(x)$ variables for the rounds greater than 1.
- The only used values are $\perp$ (suspect) and $\top$ (do not suspect)
  - ⋆ For $x = yk\ell$: $val(x) \leftarrow \top$ if p_ℓ never reported by round t that it suspects p_k; otherwise, $val(x) \leftarrow \perp$.
  - ⋆ For $x = yjk\ell$: $cval(x) \leftarrow \top$ if p_ℓ never reported by round $t+1$ that p_k reported that it suspects p_j (second-hand information); otherwise, $cval(x) \leftarrow \perp$.

The Proposed Protocol: Decision Making

- Leaf nodes: $newval(x) \leftarrow val(x)$ that is ($\perp$ or $\top$)
- Internal nodes ($x = yj$):
  - ⋆ Let $T = \{z \mid z$ child of x $(z = xk) \wedge newval(z) = \top\}$. T is the set of child nodes the label of which ends with the id k of a process that does not suspect p_j
  - ⋆ $newval(x) \leftarrow v$ if a strict majority of the $cval$ of the children of x that belong to T are set to a same value v, $\perp$ otherwise.
- The root λ: $newval(\lambda) \leftarrow v$ if a strict majority of the new values of its children are set to a same non-$\perp$ value v; otherwise it is set to a default value v_0.
  The decision value is the new value of the root.

Illustration

- Example of node 123 of the EIG tree of p_i: $newval(123)$ will be set to $\top$



Elements of the proof

- No correct process suspects another correct process
- $val(xk)_i = val(xk)_j$ and $cval(xk)_i = cval(xk)_j$ for i, j and k correct processes
- $newval(xk)_i = val(xk)_i$ for i and k correct processes
- After $t + 1$ rounds:
  - ⋆ there is a common path covering of the EIG tree.
  - ⋆ if there is a common path covering rooted at x, then either x is common or one of its ancestors is common.

Complexity Analysis

- The size of the main exchanged information per round is $O(n^2)$ (who suspects who).
- The size of the echoed information per round is $O(n^3)$.
- Optimization: during the whole execution of the protocol, a correct process can suspect at most t other processes and will echo the suspicions of all the other processes. Consequently, the total bit complexity of the protocol is $n^2 t \log(n)$.

Protocol	n	rounds	comm.
[PSL] 80	$3t+1$	$t+1$	$\exp(n)$
[GM] 93	$3t+1$	$t+1$	$O(n^9)$
This result	$3t+1$	$t+1$	$O(n^3 \log(n))$