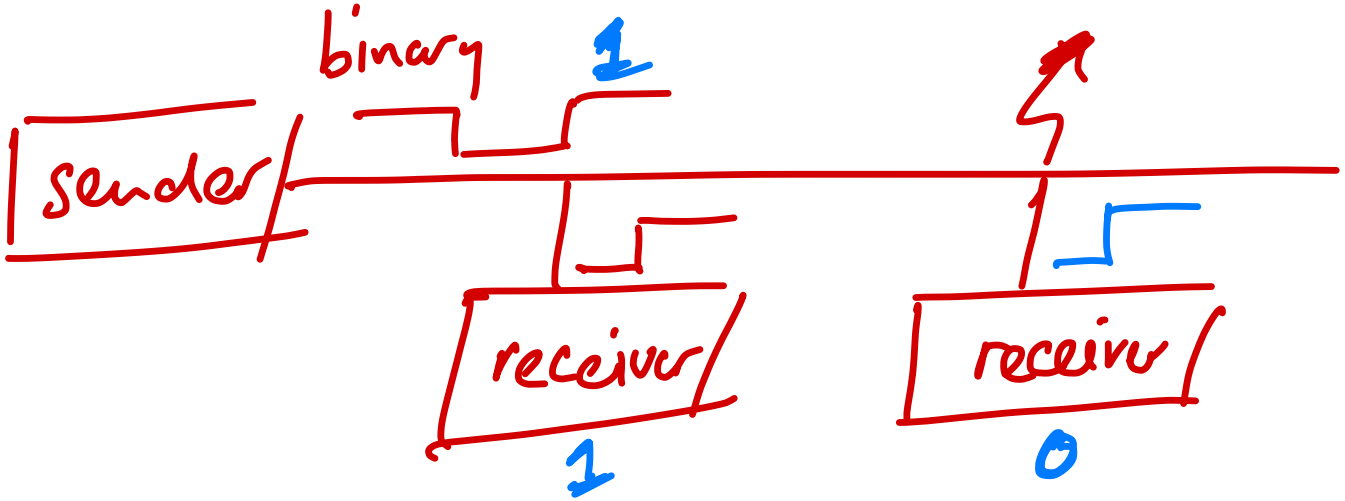
Advanced Network Security

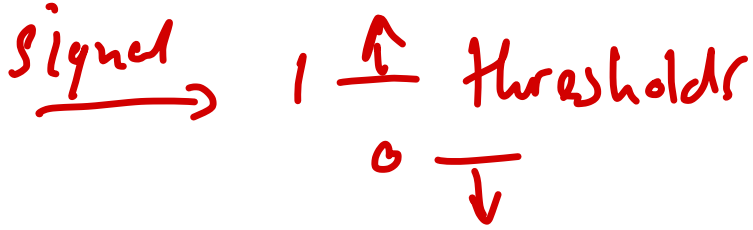
6. Agreement and consensus II: Byzantine failures

Jaap-Henk Hoepman

Digital Security (DS)
Radboud University Nijmegen, the Netherlands
@xotoxot // ✉ *jhh@cs.ru.nl* // 🖱 *www.cs.ru.nl/~jhh*

Byzantine failures are real





The consensus problem (again)

- ***All* processes have a *binary* input value (0 or 1)**
  - So it is different from a broadcast
- **Consistency condition**
  - All correct processes decide on the same value (*Agreement*)
  - If all processors have the same input value b, then all correct processors must decide b (*Validity*)
- **Termination condition**
  - Deterministic

- **Now tolerating $f < n/3$ byzantine failures**
  - Instead of arbitrary number of crash failures

Consensus for Byzantine failures

- Remember: Byzantine processors may lie…
- So: what goes wrong in the protocol for crash failures?

└ essential strategy : gossip
(for crash failures)

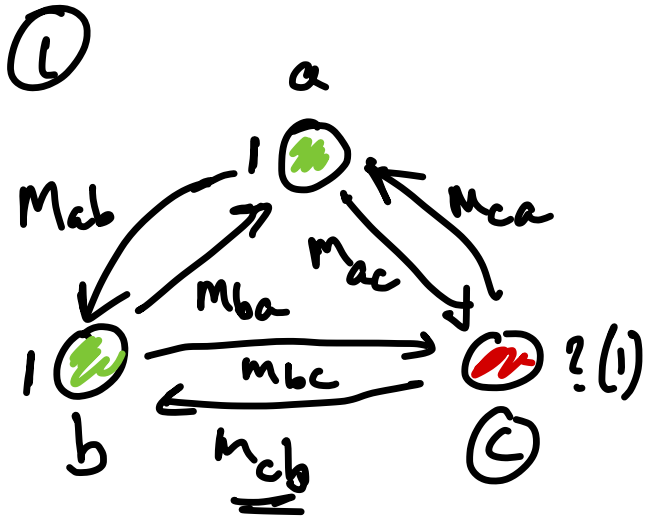
└ problem: the gossip may be a lie

Correctness proof of protocol for crash failures

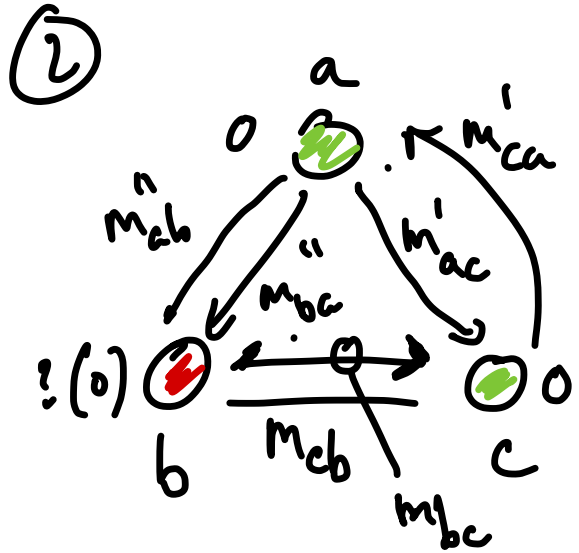
- **Lemma: suppose both processors p and q are correct (i.e don't fail). Then if $v \in V_p$ then $v \in V_q$**
- **Proof**
  - If $v \in V_p$ then $v = v_\sigma^p$ for some σ with $p \notin \sigma$
    - ★ *If $p \in \sigma$, i.e. $\sigma = \alpha; p; \beta$ then p sent $v = m_{\alpha;p}^p$ and hence $v = v_\alpha^p$ too, with $p \notin \alpha$*
  - If $|\sigma| < f + 1$ then p will send $m_{\sigma;p}^q = v_\sigma^p = v$ to q and then $v_{\sigma;p}^q = v$ and so $v \in V_q$
  - If $|\sigma| = f + 1$ then there is a non faulty processor z with $\sigma = \alpha; z; \beta$ such that $v_\alpha^z = v_\sigma^p$ Then at round $|\alpha| + 1$ processor z sent $v = v_\alpha^z$ to q as well (as message $v = m_{\alpha;z}^q$). Hence $v_{\alpha;z}^q = v$. Again $v \in V_q$

Byzantine failures: $f < n/3$ is necessary

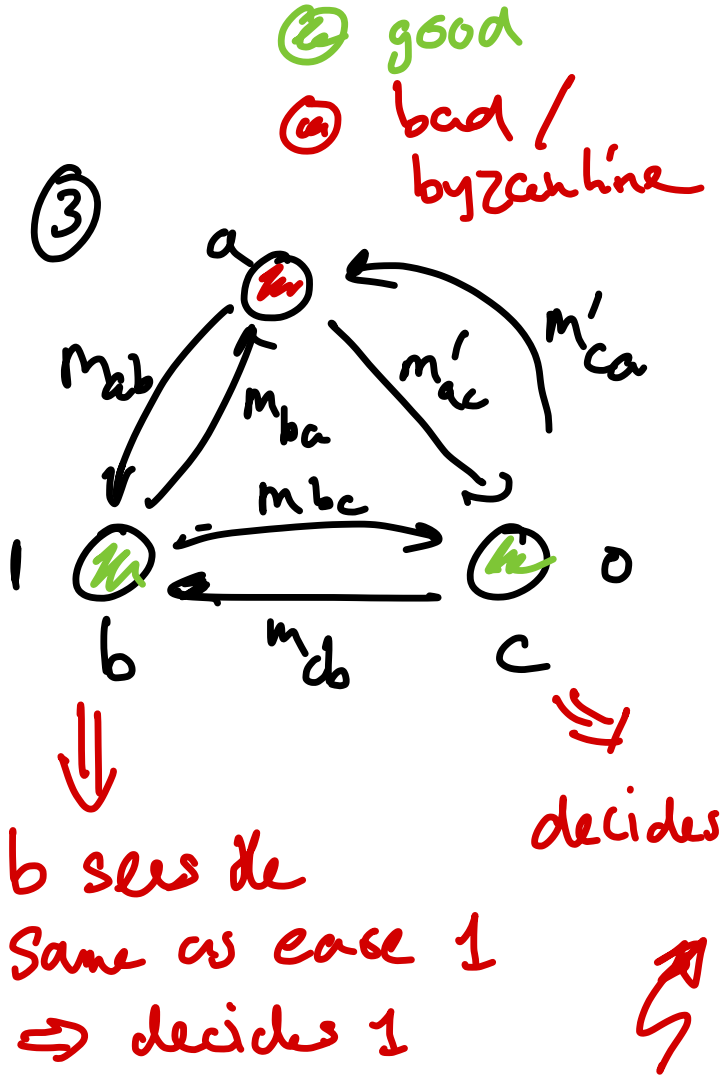
- **Suppose $n = 3$ and $f = 1$ (and two rounds)**



a & b must decide 1

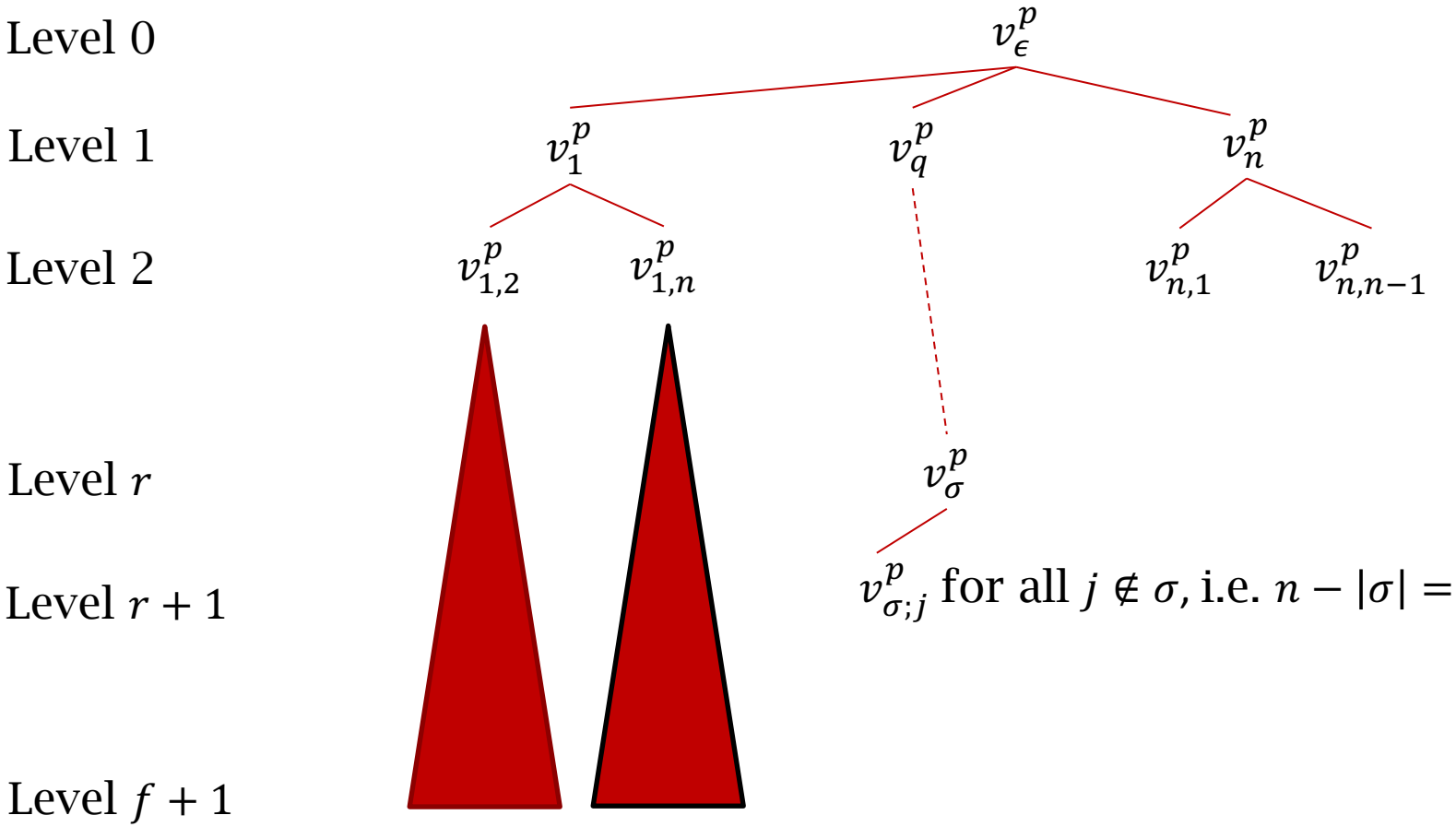


a & c must decide 0



A protocol tolerating $f < n/3$ byzantine failures

- **Again each processor p builds the following tree T_p**



$v_{q_1,q_2,..,q_k}^p$ means: q_k told p,
that q_{k-1} told q_k,
....
that q_1's value is v
Initially all $\perp$
$v_\epsilon^p = C[p].in$

Byzantine failures: decision more complex

- **Associate a decision value d_σ^p to each node in the tree**
  - After tree is filled with values top down, it is filled with decision values bottom up
  - d_ϵ^p is the value for $C[p].decision$ that p decides on

- **Define $Majority(S)$ be the value that occurs most in a set S, using some constant $\perp$ to break ties**

Lamport's OM protocol for building the tree

- **We write $OM^p_\sigma(i, v)$ to make clear processor p executes this to propagate v and to keep track of 'stack trace' σ**
  - i is recursion parameter (starts at f and ends at 0) (Lamport uses m in the paper)
  - $OM^p_\sigma(i, v)$ is executed by p for all σ s.t. $|\sigma| = f - i$ and $p \notin \sigma$
  - It sends $v = v^p_\sigma$ to all nodes q (as message $m^q_{\sigma;p}$, stored by q as $v^q_{\sigma;p}$), and instructs them to propagate the value through recursion
  - It essentially builds p's part of the subtrees rooted at σ for all processors; together with the other $OM^q_\sigma()$ the whole subtrees rooted at σ are built.
  - The protocol starts with $OM^p_\epsilon(f, C[p].in)$ for all p

Lamport's OM protocol

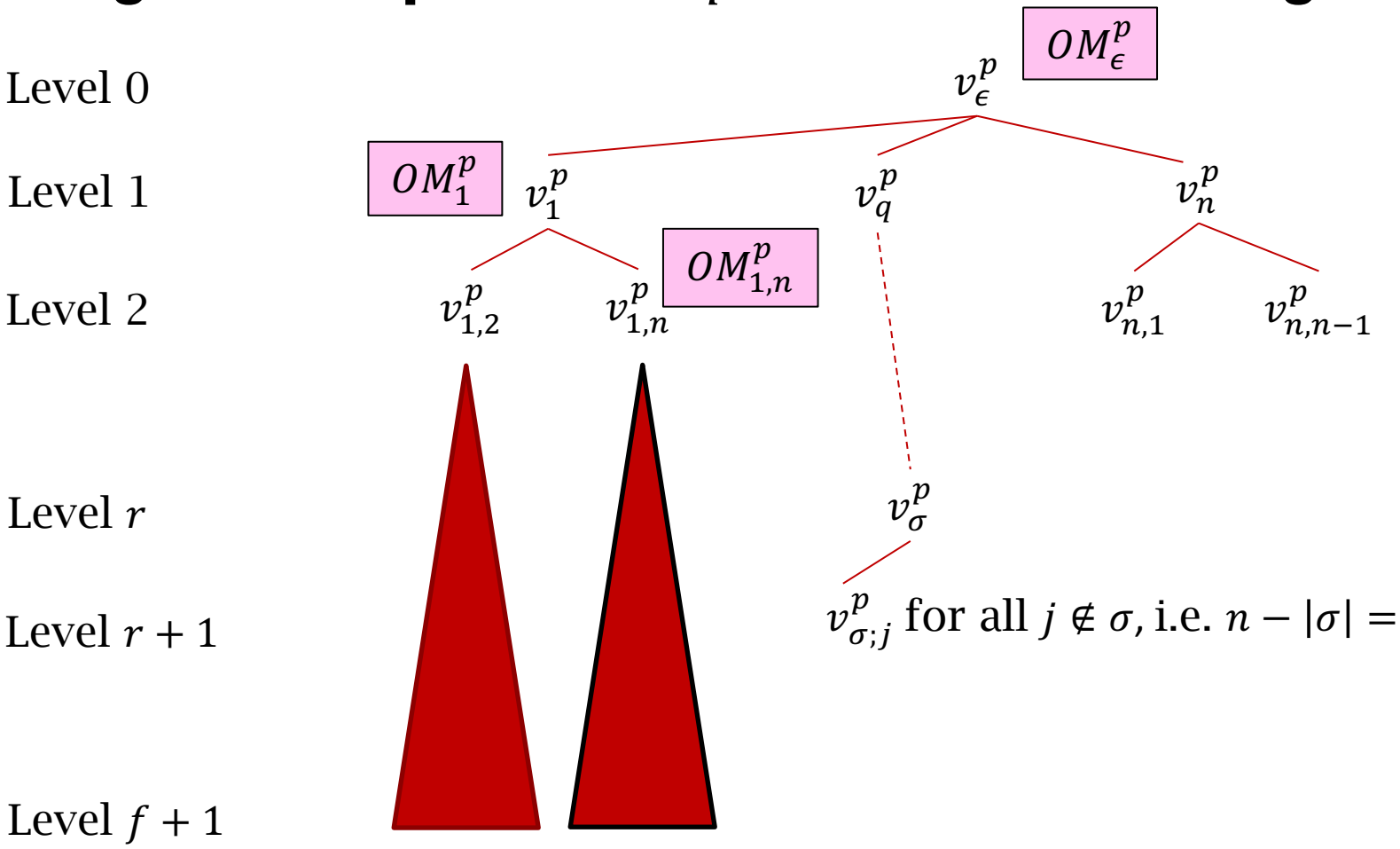
Here $|\sigma; p| = f + 1$

- $OM^p_\sigma(0, v)$:
  - Send v^p_σ as $m^q_{\sigma;p}$ to all q
  - All processors q that receive it set $v^q_{\sigma;p} = m^q_{\sigma;p}$; set $\perp$ if no value received ; and set $d^q_{\sigma;p} = v^q_{\sigma;p}$
  - Set $d^p_\sigma = Majority(\{d^p_{\sigma;q} | q \notin \sigma\})$
- $OM^p_\sigma(i, v)$ **for** $0 < i \leq f$
  - Send v as $m^q_{\sigma;p}$ to all q
  - All processors q that receive it set $v^q_{\sigma;p} = m^q_{\sigma;p} = v$; set $\perp$ if no value received
  - Trigger $OM^q_{\sigma;p}(i-1, v^q_{\sigma;p})$ for all $q \notin \sigma; p$
    - ★ *Or rather: when receiving* $m^p_{\sigma;q}$ *execute* $OM^p_{\sigma;q}(i-1, m^p_{\sigma;q})$ *if* $p \notin \sigma; q$

    m-1 rounds
  - Set $d^p_\sigma = Majority(\{d^p_{\sigma;q} | q \notin \sigma\})$
- **Start as** $OM^p_\epsilon(f, C[p].in)$ **for all** p **in round 0**
  - Storing $C[p].in$ as v^p_ϵ

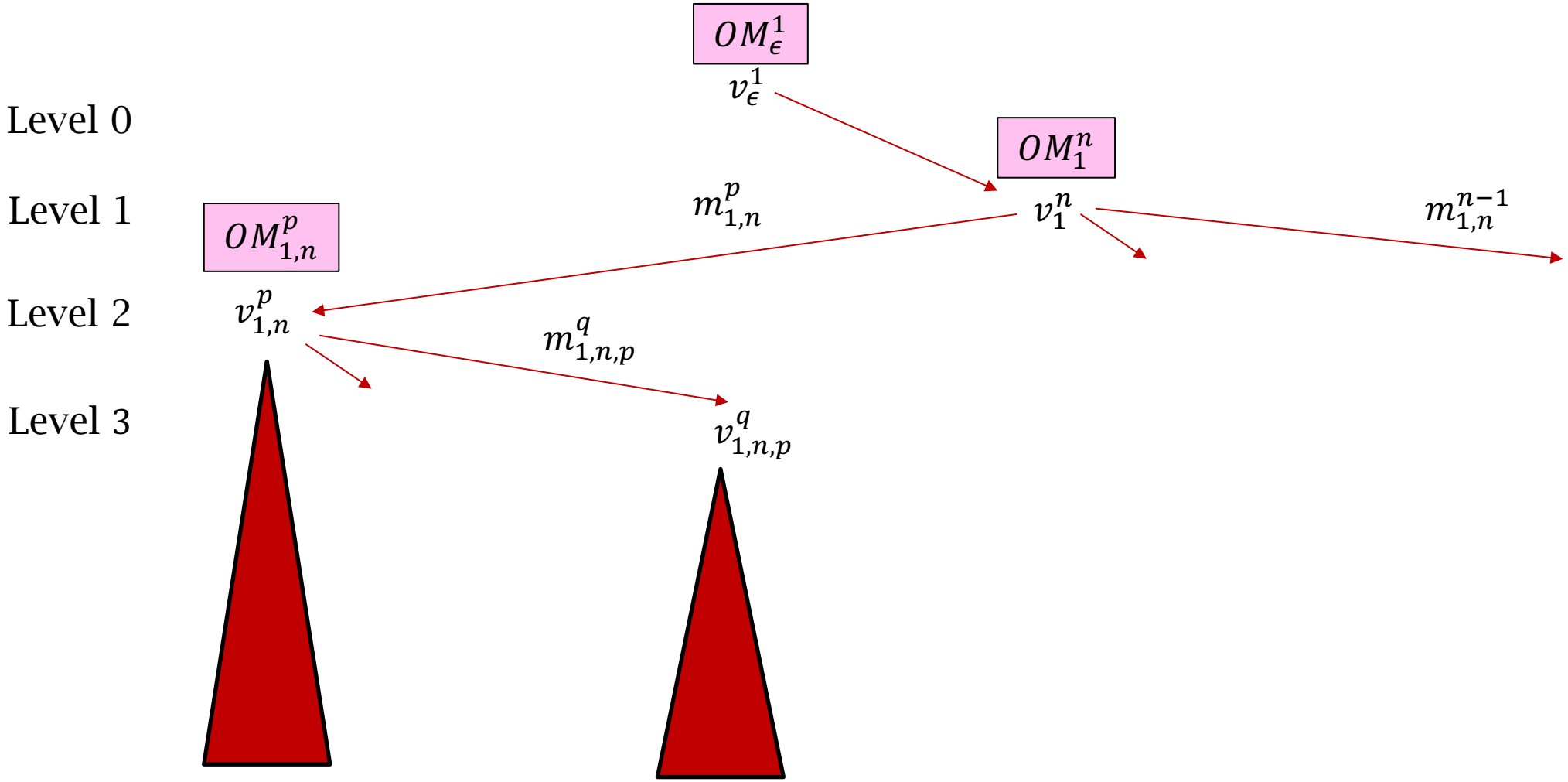
A protocol tolerating $f < n/3$ byzantine failures

- **Again each processor p builds the following tree T_p**



$v^p_{q_1,q_2,..,q_k}$ means: q_k told p,
that q_{k-1} told q_k,
....
that q_1's value is v
Initially all $\perp$
$v^p_\epsilon = C[p].in$

One step in detail



So *building* the tree is the same protocol as for crash failures.

$v^p_{q_1,q_2,..,q_k}$ means: q_k told p,
that q_{k-1} told q_k,
....
that q_1's value is v
Initially all $\perp$
$v^p_\epsilon = C[p].in$

- **Before round 1**
  - Initialise tree. Set all $v^p_\sigma = \perp$ and $v^p_\epsilon = C[p].in$
- **Round** $r, 1 \leq r \leq f + 1$
  - For all σ with $|\sigma| = r - 1 \wedge p \notin \sigma$, send v^p_σ to all processors q (including p)
    - ★ *Call this message* $m^q_{\sigma;p}$
  - Receive all $m^p_{\sigma;x}$ addressed to p and store in $v^p_{\sigma;x}$
    - ★ *By the protocol* $x \notin \sigma$ *so* p *receives* $n - (r-1)$ *such messages from each* x

Deciding on a value

- **Work from the leaves upwards**
  - $d_\sigma^p = v_\sigma^p \;\; for \; |\sigma| = f + 1$
  - $d_\sigma^p = Majority(\{d_{\sigma;q}^p | q \notin \sigma\})$ otherwise
  - Node p decides on d_ϵ^p

Correctness

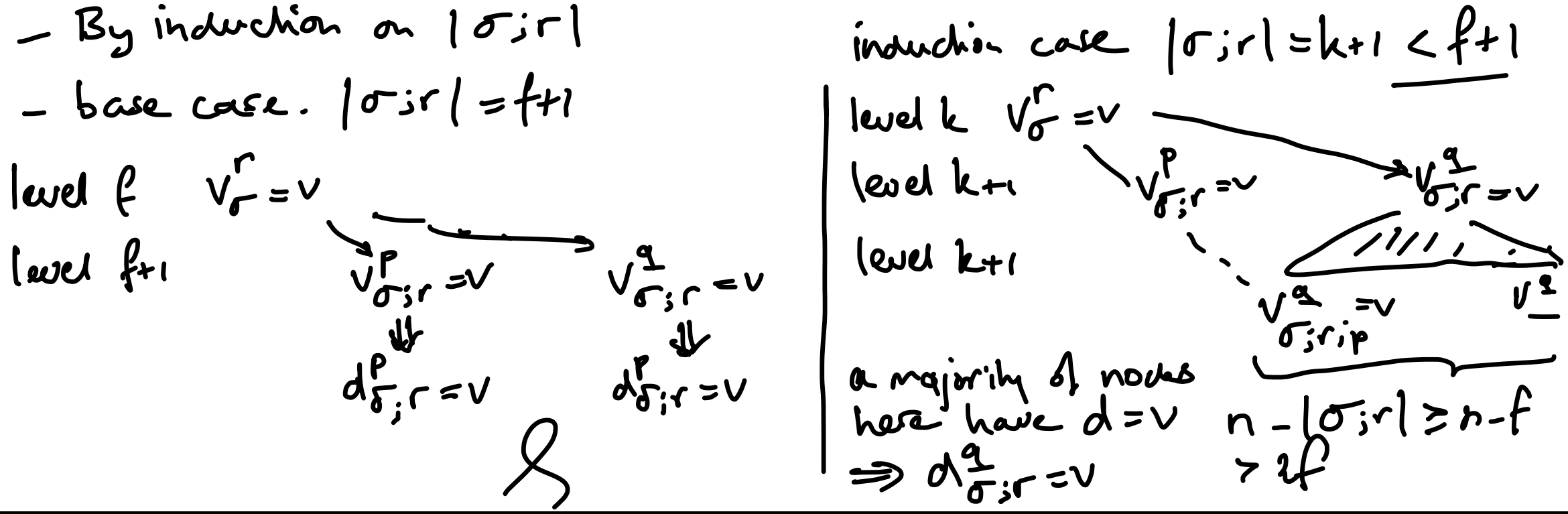
We reason over all trees

- **Lemma 1: If p, q, r are non faulty, then for all σ we have $v^p_{\sigma;r} = v^q_{\sigma;r}$**
  - Proof: r is correct, so it sends the same value to p & q

- **Set $d^p_\sigma = v^p_\sigma$ for all leaves, ie $|\sigma| = f + 1$**

Correctness

- **Lemma 1: If p, q, r are non faulty, then for all σ we have $v^p_{\sigma;r} = v^q_{\sigma;r}$**
- **Lemma 2: Let σ be arbitrary and let r be non faulty. Then there is a value v such that for all non faulty p we have $d^p_{\sigma;r} = v^p_{\sigma;r} = v$.**



Correctness

- **Lemma 1: If p, q, r are non faulty, then for all σ we have $v^p_{\sigma;r} = v^q_{\sigma;r}$**
- **Lemma 2: Let σ be arbitrary and let r be non faulty. Then there is a value v such that for all non faulty p we have $d^p_{\sigma;r} = v^p_{\sigma;r} = v$.**
  - By induction on the length of $\sigma; r$ starting with the leaves (length $f + 1$)
  - The base case follows from lemma 1 and the fact that for $|\sigma; r| = f + 1$ we have $d^p_{\sigma;r} = v^p_{\sigma;r}$.
  - Now suppose $0 \leq |\sigma; r| < f + 1$. By lemma 1 all non faulty processors have the same value $v^p_{\sigma;r} = v$. Then all non-faulty processors $p \notin \sigma; r$ send v as $m^q_{\sigma;r;p}$ to all other processors q. If non faulty, q sets $v^q_{\sigma;r;p} = v$.
  - By the induction hypothesis we have $d^q_{\sigma;r;p} = v^q_{\sigma;r;p} = v$ for all non faulty q.
  - The number of children of a node with label $\sigma; r$ is $n - |\sigma; r| \geq n - f > 2f$
  - Hence the majority of children is non-faulty, and so $d^q_{\sigma;r} = Majority(\{d^q_{\sigma;r;p} | p \notin \sigma\}) = v$ as required

Validity

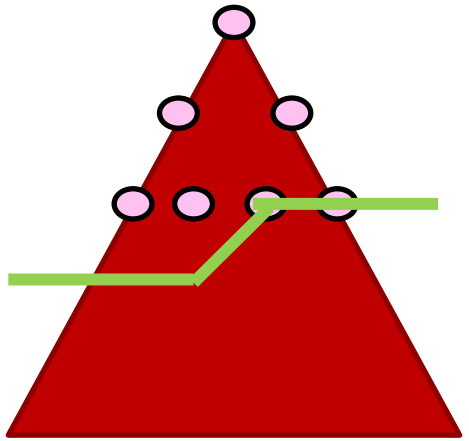
- **Theorem: If all non faulty processors have input v they decide on v**

  - if all non-faulty p have value v they send v to all non-fault q in the first round $\Rightarrow$ $v_q^p = v$ for all correct p & q.
  - Lemma 2 $d_q^p = v$ for all correct p & q

  $\Rightarrow d_p^p = \text{Maj}(\{d_q^p \mid \text{for all } q\}) = v$

Agreement

- **Definition 1. σ is *common* if $d_\sigma^p = d_\sigma^q$ for all pairs of non faulty p, q.**
- **Definition 2. A subset C of nodes in a tree T is a *path cover* of T if all paths from the leaves to the root visit at least one node in C.**
- **Definition 3. A path cover C is common if all nodes in C are common.**
  - Note: this does not require $d_\sigma^p = d_{\sigma'}^q$, for different σ, σ'.



Agreement

- **Lemma 3. There exists a common path covering of the tree constructed by the consensus algorithm**
  - All paths from the root to a leaf correspond to a label σ with length $f + 1$.
  - Then $\sigma = \sigma'; r; \sigma''$ for some non faulty r
  - By lemma 2 $d^p_{\sigma';r} = d^q_{\sigma';r}$ for all non faulty p, q and so $\sigma'; r$ is common and on the path

Agreement

- **Lemma 4. Let σ be a node. If there is a common path covering of the subtree rooted at σ, then σ is common itself.**
  - By induction on the length of σ
  - For $|\sigma| = f + 1$ the lemma trivially follows
  - Let $0 \leq |\sigma| < f + 1$ and assume there is a common path covering C of the subtree rooted at σ. If $\sigma \in C$ we are done. If not, then the trees rooted in all children have a common path covering and by the induction hypothesis then all children $\sigma; r$ of σ are common.
  - Hence $d^p_{\sigma;r} = d^q_{\sigma;r}$ for all pairs of non faulty p, q. Hence $d^p_\sigma = Majority(\{d^p_{\sigma;r} | r \notin \sigma\}) = Majority(\{d^q_r | r \notin \sigma\}) = d^q_\sigma$ and hence σ is common as well.
- **Theorem: All non faulty nodes decide on the same value**
  - Follows from lemma 3 and 4.

Using authentication

Signing messages

- **Every processor p has a private signing key. The corresponding signature verification key is known to all processors.**
- **Signatures of correct processors cannot be forged**
- **Let us write $[m]_p$ for a message m signed by p. Write $[m]_\sigma$ for $[\ldots[m]_p..]_r$ with $\sigma = p; \ldots; r$**
- **Processors reject any messages with incorrect signatures.**
  - Byzantine nodes cannot forge values pretending they heard another value *from a correct processor*
  - But they can send conflicting initial values in the first round!

Using authentication to tolerate Byzantine failures

- **Could the protocol for crash failures by used to toleratean arbitrary number if Byzantine failures?**
  - By always sending signed messages, so Byzantine processors cannot forge information?

Validity: if all processors have the same input value v, then v must be the decision

(0 is the default decision value

$\Rightarrow V_p = \{0,1\} \rightarrow$ decision = the default = $\emptyset$

(Re)onsider the weak broadcast protocol

- **One server p holds a bit**
  - Either 0 or 1
- **Consistency condition:**
  - (*Agreement)* All correct processes decide on the same value
  - (*Validity*) If p is not faulty, this should be p's input
- **Termination condition:**
  - deterministic
- **Assumptions**
  - Byzantine failures
  - Synchronous communication

(Binary) Broadcast (aka agreement)

$f < n$!

- **Sender p in round 1**
  - If $C[p].in = 1$ then send $[1]_p$ to all, otherwise stay silent
  - Decide on $C[p].in$
- **Other nodes q**
  - For each round $r \in \{1, \ldots, f+1\}$
    - ★ *If you receive a valid* $[1]_\sigma$ *message (note* $|\sigma| = r$*) with* $\sigma = p;\sigma'$ *then send* $[[1]_\sigma]_p = [1]_{\sigma;q}$ *to all, decide on* 1 *and terminate*
  - After round $f+1$ decide on 0 and terminate

Correctness

- **Agreement**
  - Suppose a correct node decides on 1 in round r. This means it received a valid $[1]_\sigma$ message.
  - If $r < f + 1$ then p sends a valid $\left[[1]_\sigma\right]_p = [1]_{\sigma;q}$ message to all correct q who therefore decide on 1 too.
  - If $r = f + 1$ then $|\sigma| = f + 1$ hence $\sigma = \sigma'; q; \sigma''$ for some correct q that sent a valid $[1]_{\sigma';q}$ message to all correct nodes that therefore decided on 1 in round $|\sigma'| + 1$.

Correctness

- **Validity**
  - Suppose p is correct.
  - Either its input is 1 so it sends $[1]_p$ to all, and all correct nodes decide 1 in round 1.
  - Or its input is 0 so it does not send anything. As a result no correct node receives a valid $[1]_\sigma$ message, so all correct nodes decide 0 in round $f + 1$

Using agreement to reach consensus

- **Use Byzantine agreement as a subprotocol**
  - Each node maintains a vector V of values, one for each node i, initially empty
  - Each node i uses the Byzantine agreement (i.e. broadcast algorithm) to broadcast its input value to all other nodes. This takes at most $f + 1$ rounds
  - All other nodes receive this value and store it in $V[i]$
  - All other nodes obtain (by the agreement property of the broadcast) the same vector of input values
  - All nodes decide on the majority of values in this vector (breaking ties in a deterministic way)
  - If $f < n/2$ then if all nodes have the same input value, all nodes decide on this value.

Strong validity condition yields $f < n/2$

- **Consider weak validity**
  - If all processes are correct and all have the same input value, then this is the decision value
- **Then the algorithm for crash failures strengthened with authentication becomes a consensus algorithm for Byzantine failures for arbitrary $f < n$**