Advanced Network Security
5. 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

Consensus for Byzantine failures

- Remember: Byzantine processors may lie...
- What goes wrong in the protocol for crash failures?
  - If \(|e| = f + 1\) then there is a non faulty processor \(z\) with \(\sigma = a; \gamma; \beta\).
    Then at round \(|e| + 1\) processor \(z\) sent \(v = v^{\sigma}_{z}\) to \(q\) as well (as the first processor in \(\beta\)). By construction \(v^{\sigma}_{z} = v^{\gamma}_{z}\) (as the processors in \(\beta\) forward this value to each other to finally deliver it to \(p\)). Again \(v \in I_0\)

Byzantine failures: \(f < n/3\) is necessary

- Suppose \(n = 3\) and \(f = 1\)
  - \(a\) and \(b\) must decide 1
  - \(a\) and \(c\) must decide 0
  - \(b\) decides 1, as it sees the same messages, yet \(c\) decides 0 as it sees the same messages

A protocol tolerating \(f < n/3\) byzantine failures

- Again each processor \(p\) builds the following tree \(T_p\)
  - Level 0
  - Level 1
  - Level 2
  - Level \(r\)
  - Level \(r + 1\)
  - Level \(f + 1\)
Byzantine failures: decision more complex

- Associate a decision value $d^p_v$ to each node in the tree
  - After tree is filled with values top down, it is filled with decision values bottom up
  - $d^p_v$ 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 $OM^\text{OM}_p(m,v)$ to make clear processor $p$ executes this to propagate $v$ and to keep track of ‘stack trace’ $\sigma$
  - $OM^\text{OM}_p(m,v)$ is executed by $p$ for all $\sigma$ s.t. $|\sigma| = f - m$ and $p \notin \sigma$
  - It sends $v = v^p_\sigma$ to all nodes $q$ (as message $m^p_\sigma$), stored by $q$ as $v^q_\sigma$, and instructs them to propagate the value through recursion
  - It essentially builds $p$’s part of the subtrees rooted at $\sigma$ for all processors; together with the other $OM^\text{OM}_p$ the whole subtrees rooted at $\sigma$ are built.
  - The protocol starts with $OM^\text{OM}_p(f,C[p].in)$ for all $p$

Lamport’s OM protocol for building the tree

- Start as $OM^\text{OM}_p(f,C[p].in)$ for all $p$ in round 0
  - Storing $C[p].in$ as $v^0_\sigma$

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

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

  - Level 0 builds $OM^\text{OM}_p(m,v)$
  - Level 1 builds $OM^\text{OM}_p(m,v)$
  - Level 2 builds $OM^\text{OM}_p(m,v)$
  - Level $r$ builds $OM^\text{OM}_p(m,v)$

**Example:**
- $v^0_{q1} = v^0_{q2}$ means: $q_1$ told $p$, that $q_2$, told $q_1$, that $q_1$’s value is $v$
- Initially all $i$’s value is $\perp$
- $v^r_{q3} = v^r_{q4}$ for all $j \notin \sigma$, i.e. $n - |\sigma| = n - r$ children
**One step in detail**

- **Level 0**:
  - \( v^0 \)
  - \( m^0_{1,0} \)
  - \( m^0_{1,n} \)

- **Level 1**:  
  - \( v^1 \)
  - \( m^1_{1,0} \)
  - \( m^1_{1,n} \)

- **Level 2**:  
  - \( v^2 \)
  - \( m^2_{1,0} \)
  - \( m^2_{1,n} \)

- **Level 3**:  
  - \( v^3 \)
  - \( m^3_{1,0} \)
  - \( m^3_{1,n} \)

---

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

- **Before round 1**
  - Initialise tree. Set all \( v^r \) and \( v^r = C[p] \) in
  - Initially all \( v^0 = C[p] \) in

- **Round** \( r, 1 \leq r \leq f + 1 \)
  - For all \( \sigma \) with \( |\sigma| = r - 1 \land p \not\in \sigma \), send \( v^r \) to all processors \( q \) (including \( p \))
    - Call this message \( m^r_{q,p} \)
    - Receive all \( m^r_{x,y} \) addressed to \( p \) and store in \( v^r_{p,x} \)
    - By the protocol \( x \not\in \sigma \) so \( p \) receives \( n - (r - 1) \) such messages from each \( x \)

---

**Deciding on a value**

- **Work from the leaves upwards**
  - \( d^f_0 = v^f_0 \) for \( |\sigma| = f + 1 \)
  - \( d^f_x = \text{Majority}(d^f_{x,q} | q \not\in \sigma) \) otherwise
  - Node \( p \) decides on \( d^f_p \)

---

**Correctness**

- **Lemma 1**: If \( p, q, r \) are non faulty, then for all \( \sigma \) we have \( v^f_{q,r} = v^f_{p,r} \)
  - If \( r \) is non faulty, it sends the same value to \( p \) and \( q \). If \( p \) and \( q \) are non faulty the record this value correctly.

- **Set** \( d^f_x = v^f_x \) for all leaves, ie \( |\sigma| = f + 1 \)
Correctness

- **Lemma 1**: If \( p, q, r \) are non faulty, then for all \( v \) we have \( d^p_v = d^q_v = d^r_v \).
- **Lemma 2**: Let \( \sigma \) be arbitrary and let \( r \) be non faulty. Then there is a value \( v \) such that for all non faulty \( p \) we have \( d^p_v = v \).
  - By induction on the length of \( p, r \) starting with the leaves (length \( f + 1 \)).
  - The base case follows from lemma 1 and the fact that for \( p, r = f + 1 \) we have \( d^p_v = v \).
  - Now suppose \( 0 \leq |p, r| < f + 1 \). By lemma 1 all non faulty processors have the same value \( v \). Then all non-faulty processors \( p \not\equiv r \) send \( v \) as \( m_{p,r} \) to all other processors \( q \). If non faulty, \( q \) sets \( v_{p,r} = v \).
  - By the induction hypothesis we have \( v_{s,r} = v \) for all non faulty \( s \).
  - The number of children of a node with label \( s, r \) is \( n - |s, r| \geq n - f > 2f \).
  - Hence the majority of children is non-faulty, and so \( d^p_v = \text{Majority}([d^s_v | p \not\equiv s]) = v \).

Lemma 2

- **Lemma 2**: Let \( \sigma \) be arbitrary and \( r \) be non faulty.
  - **Base case** \( |\sigma, r| = f + 1 \)
  - **Induction** \( |\sigma, r| = k + 1 < f + 1 \)

Validity

- **Theorem**: If all non faulty processors have input \( v \) they decide on \( v \)
  - If all non faulty processors have input \( v \), they send \( v \) to all other nodes in the first round. As a result \( v^p_v = v \) for all correct \( p \) and \( q \).
  - By lemma 2 \( d^p_v = v \) for all correct \( p \) and \( q \) and hence \( d^p_v = \text{Majority}([d^q_v | \text{for all } q]) = v \).

Agreement

- **Definition 1**: \( \sigma \) is common if \( d^p_v = d^q_v \) for all pairs of non faulty \( p, q \).
- **Definition 2**: A subset \( \tilde{C} \) 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 \( \tilde{C} \).
- **Definition 3**: A path cover \( \tilde{C} \) is common if all nodes in \( \tilde{C} \) are common. (Note: this does not require \( d^p_\sigma = d^q_\sigma \), for different \( \sigma, \sigma' \).
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 \( \sigma \) with length \( f + 1 \).
- Then \( \sigma = \sigma'; \tau; \sigma'' \) for some non faulty \( \tau \)
- By lemma 2 \( d^\sigma_{p,q} = d^\sigma'_{p,q} \) for all non faulty \( p,q \) and so \( \sigma'; \tau \) is common and on the path.

Lemma 4. Let \( \sigma \) be a node. If there is a common path covering of the subtree rooted at \( \sigma \), then \( \sigma \) is common itself.
- By induction on the length of \( \sigma \)
- 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 \( \sigma \). 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 \( \sigma'; \tau \) of \( \sigma \) are common.
- Hence \( d^\sigma_{p,q} = d^{\sigma'}_{p,q} \) for all pairs of non faulty \( p,q \). Hence \( d^\sigma = \text{Majority}(d^{\sigma'}_{p,q}) \) and hence \( \sigma \) is common as well.

Theorem: All non faulty nodes decide on the same value
- Follows from lemma 3 and 4.

Signing messages

Every processor \( p \) has a private signing key. The corresponding signature verification key is known to all processors.
- Let us write \( [m]_p \) for a message \( m \) signed by \( p \). Write \( [m]_{p,q} \) for \( [\ldots[m]_{p,q-1}]_{p} \), with \( \sigma = p; \ldots ; \tau \)
- 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!
- Now consider the weak broadcast protocol

Using authentication
(Binary) Broadcast (aka agreement)

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

Correctness

- **Agreement**
  - Suppose a correct node decides on 1 in round $r$. This means it received a valid $[1]_p$ message. If $r < f + 1$ then $p$ sends a valid $[[1]_p]_{\sigma} = [1]_{\sigma'}$ message to all correct $q$ who therefore decide on 1 too. If $r = f + 1$ then $|p| = f + 1$ hence $\sigma = \sigma'; q; \sigma''$ for some correct $q$ that sent a valid $[1]_{\sigma''}$ message to all correct nodes that therefore decided on 1 in round $|p'| + 1$.
- **Validity**
  - Suppose $p$ is correct. Either it sends $[1]_p$ to all, and all correct nodes decide 1 in round 1. Or it does not send anything. As a result no correct node receives a valid $[1]_q$ message, so all correct nodes decide 0 in round $f + 1$.

Reaching consensus

- Each node uses the broadcast algorithm to send its input value to all other nodes
- 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.