GST \Delta

\langle \text{amount, recipient, memo} \rangle

1. Coin Distribution | Proof-of-work
2. Double-spend problem

? \text{H}(b_{i..i})
\text{ nonce, new transactions}
$b' = \exists \nu, H(b), x^3$

Diagram:

- $b' \Rightarrow \nu \Rightarrow b''$
- $b \Leftarrow b''$
- $b \Leftarrow b$
State at R_i:

- QC: N-f matching signed A2 messages for highest view seen
- locked: A "locked" block for which R_i sent A2, or NULL if none locked

\( {} = \text{unsigned}, \quad <> = \text{signed} \)

- \( R_i \rightarrow L_v: \{A1, v, QC|NULL\} \)
- \( L_v \rightarrow R_i: \{M2, v, \text{block}, QC\} \)
- \( R_i \rightarrow L_v: \langle A2, v, \text{block} \rangle \)
- \( L_v \rightarrow R_i: \{M3, v, QC2\} \)
- \( R_i \rightarrow L_v: \langle A3, v, \text{block} \rangle \)
- \( L_v \rightarrow R_i: \{M4, v, QC3\} \)
A2 rules

if block does not extend QC block

drop

if m.QC.v > lock.v && lock <= block

if m.QC > QC update QC
set lock to NULL
\[ R_i \rightarrow L_v : \{ \text{EMO, v, QC | NULL} \} \]
\[ L_v \rightarrow R_i : \{ \text{EMI, v, QC | NULL} \} \]
\[ R_i \rightarrow L_v : <A_i, v, \text{block, QC} \} \]

AO = NEW-VIEW
A1 = PREPARE
A2 = PRE-COMMIT
M41 = DECIDE
- Commit/externalize when you see N-f signed A3s
- Don't send A3 unless you see N-f signed A2s
- Don't send A2 if its block is incompatible with your locked block
- Lock A2.block when sending A2
- Unlock if you see QC with QC.v > lock.v and !extends(QC.block, v.block)
- Leader chooses block extending highest QC it knows
Safety of 2-round conflict

(N-f) A3 msgs from
(N-2f) honest, N = 3f+1
w.v < b.v, f+1 honest is maj. 2f+1

QC (N-f) A2 msgs, N-2f honest
- Still commit/externalize when you see N-f signed A3s
- Don't send A3 unless you see N-f signed A2s
- Don't send A2 unless you see N-f signed A1s
- Don't send A1 if its block is incompatible with your locked block
- Still lock A2.block when sending A2
- But unlock with N-f incompatible higher *A1* messages

When lock?

1round: if you know N-2f honest
       won't lock conflicting block

2round: if N-2f honest nodes know
        that won't lock conflicting block
1. Can't commit conflicting blocks in same view

2. A committed block is locked and can't be unlocked.

\( w_1 \leftarrow w_2 \leftarrow w_3 \leftarrow w_4 \)