Block header:
$b_k.header = \langle H(b_{k-1}.header), k, iter \rangle$

Block:
$b_k = \langle \langle H(b_{k-1}.header), k, iter \rangle, txs \rangle$

Notarized Block:
$b_k = \langle \langle H(b_{k-1}.header), k, iter \rangle, txs, cert \rangle$

Blockchain:
$\mathcal{B} = \{\langle \langle H(b_{k-1}.header), k, iter \rangle, txs, cert \rangle\}_{k=0}^{|\mathcal{B}|}$

extendable(b): $b_{|\mathcal{B}|}.k = b.k - 1 \wedge b.hash = H(b_{|\mathcal{B}|}.header) \wedge b_{|\mathcal{B}|}.iter < b.iter$

validCertificate$(cert, header)$: $|\{j \mid \langle \_, \_, \langle h \rangle_j \rangle \in cert, h = header\}| \geq n - f$

Note: For simplicity when referring to $b_k.header.iter$ we use $b_k.iter$ instead for abbreviation.

Algorithm 1 Practical Simplex – replica i.

task `start`()
1: $iter \leftarrow 1$
2: **trigger** `newIteration`($iter$)

task `handleNotarization`($header, cert, txs, it$)
3: `notarize`($header, cert, txs$)
4: **if** $\neg isTimeout$
5: **broadcast** $\langle \textsc{Finalize}, it \rangle_i$
6: **send** $\langle \textsc{State}, header, cert \rangle_i$ **to** everyone except i
7: **trigger** `newIteration`($it + 1$)

upon `newIteration`(h)
8: $iter \leftarrow h$
9: `resetTimer`($timer$)
10: $isTimeout \leftarrow false$
11: **if** $i =$ `leader`(h)
12: $b \leftarrow \langle \langle H(b_{|\mathcal{B}|}.header), b_{|\mathcal{B}|}.k + 1, h \rangle, txs \rangle$
13: **broadcast** $\langle \textsc{Propose}, b \rangle_i$

upon expiring $timer$
14: `stopTimer`($timer$)
15: $isTimeout \leftarrow true$
16: **broadcast** $\langle \textsc{Timeout}, iter + 1 \rangle_i$

upon receiving $\langle \textsc{Propose}, b \rangle_j$
17: **pre:** $proposes[b.iter] = \perp \wedge j =$ `leader`($b.iter$) $\wedge\ b.iter = iter \wedge \neg isTimeout$
18: $proposes[iter] \leftarrow b$
19: **if** `extendable`(b)
20: **broadcast** $\langle \textsc{Vote}, h, \langle b.header \rangle_i \rangle$

upon receiving $\{\langle \textsc{Vote}, h, \langle header \rangle_j \rangle : j \in Q\} = V$ **from** a quorum Q
21: **pre:** $h = iter \wedge proposes[h] \neq \perp$
22: `handleNotarization`($header, \{\langle header \rangle_j \mid \langle \_, \_, \langle header \rangle_j, \_, \_ \rangle \in V\}, proposes[h].txs, iter$)

upon receiving $\{\langle \textsc{Finalize}, iter \rangle_j : j \in Q\}$ **from** a quorum Q
23: **if** $\exists b \in \mathcal{B}, b.iter = iter$
24: `finalize`($iter$)
25: **else**
26: **send** $\langle \textsc{Request}, |\mathcal{B}| \rangle_i$ **to** one $j \in Q$

upon receiving $\{\langle \textsc{Timeout}, nextIter \rangle_j : j \in Q\}$ **from** a quorum Q
27: **pre:** $nextIter = iter + 1$
28: **trigger** `newIteration`($nextIter$)

upon receiving $\langle \textsc{State}, header, cert \rangle_j$
29: **pre:** $h > |\mathcal{B}| \wedge$ `validCertificate`($cert, header$)
30: **send** $\langle \textsc{Request}, |\mathcal{B}| \rangle_i$ **to** j

upon receiving $\langle \textsc{Request}, h \rangle_j$
31: **pre:** $h < |\mathcal{B}|$
32: **send** $\langle \textsc{Reply}, \{\langle b_k.header, b_k.cert, b_k.txs \rangle\}_{k=h}^{|\mathcal{B}|} \rangle_i$ **to** j

upon receiving $\langle \textsc{Reply}, \mathcal{M} = \{\langle b_k.header, b_k.cert, b_k.txs \rangle\}_{k=|\mathcal{B}|+1}^{h} \rangle_j$
33: **pre:** $h > |\mathcal{B}| \wedge \forall \langle header, cert, \_ \rangle \in \mathcal{M}$, `validCertificate`($cert, header$)
34: **for** each $\langle header, cert, txs \rangle \in \mathcal{M}$
35: `handleNotarization`($header, cert, txs, header.iter$)

Algorithm 2 Probabilistic Practical Simplex – replica i.

task `start`()
1: $iter \leftarrow 1$
2: **trigger** `newIteration`($iter$)

task `handleNotarization`($header, cert, txs, it$)
3: `notarize`($header, cert, txs$)
4: **if** $\neg isTimeout$
5: $S_f, P_f \leftarrow$ `VRF_prove`($K_{p,i}, it \mid\mid$ "finalize", $o \times q$)
6: **send** $\langle$FINALIZE, $it, S_f, P_f\rangle_i$ **to** S_f
7: **trigger** `newIteration`($it + 1$)

upon `newIteration`(h)
8: $iter \leftarrow h$
9: `resetTimer`($timer$)
10: $isTimeout \leftarrow false$
11: **if** $i =$ `leader`(h)
12: $b \leftarrow \langle\langle H(b_{|\mathcal{B}|}.header), b_{|\mathcal{B}|}.k + 1, h\rangle, txs\rangle$
13: **broadcast** $\langle$PROPOSE, $b, b_{|\mathcal{B}|}.cert, b_{|\mathcal{B}|}.iter\rangle_i$

upon expiring $timer$
14: `stopTimer`($timer$)
15: $isTimeout \leftarrow true$
16: **broadcast** $\langle$TIMEOUT, $iter + 1\rangle_i$

upon receiving $\langle$PROPOSE, $b, cert, it\rangle_j$
17: **if** $proposes[b.iter] = \perp \wedge j =$ `leader`($b.iter$)
18: $proposes[b.iter] \leftarrow b$
19: **if** $b.iter > iter \wedge proposes[it] \wedge$ `validCertificate`($cert, proposes[it].header$)
20: `handleNotarization`($proposes[it].header$), $cert, proposes[it].txs), it$)
21: **if** $b.iter > iter \wedge \neg proposes[it]$
22: **send** $\langle$REQUEST, $|\mathcal{B}|\rangle_i$ **to** j
23: **pre:** $b.iter = iter \wedge$ `extendable`(b) $\wedge \neg isTimeout$
24: $S_v, P_v \leftarrow$ `VRF_prove`($K_{p,i}, iter \mid\mid$ "vote", $o \times q$)
25: **send** $\langle$VOTE, $iter, \langle b.header\rangle_i, S_v, P_v\rangle$ **to** S_v

upon receiving $\{\langle$VOTE, $h, \langle header\rangle_j, S, P\rangle : j \in Q\} = V$ **from** a probabilistic quorum Q
26: **pre:** $h = iter \wedge proposes[h] \neq \perp \wedge \forall\langle \_, \_, \_, S, P\rangle_j \in V : i \in S \wedge$
`VRF_verify`($K_{u,j}, iter \mid\mid$"vote", $o \times q, S, P$))
27: `handleNotarization`($header, \{\langle header\rangle_j \mid \langle \_, \_, \langle header\rangle_j, \_, \_\rangle \in V\}, proposes[h].txs, iter$)

upon receiving $\{\langle$FINALIZE, $iter, S, P\rangle_j : j \in Q\} = F$ **from** a probabilistic quorum Q
28: **pre:** $\forall\langle \_, \_, S, P\rangle_j \in F : i \in S \ \wedge$ `VRF_verify`($K_{u,j}, iter \mid\mid$ "finalize", $o \times q, S, P$))
29: **if** $\exists b \in \mathcal{B}, b.iter = iter$
30: `finalize`($iter$)

upon receiving $\langle$TIMEOUT, $nextIter\rangle_j$ **from** j for the first time
31: **if** $\exists b \in \mathcal{B}, b.iter = nextIter - 1$
32: **send** $\langle$REPLY, $\{\langle b.header, b.cert, b.txs\rangle\}\rangle_i$ **to** j

upon receiving $\{\langle$TIMEOUT, $nextIter\rangle_j : j \in Q\}$ **from** a quorum Q
33: **pre:** $nextIter = iter + 1$
34: **trigger** `newIteration`($nextIter$)

upon receiving $\langle$REQUEST, $h\rangle_j$
35: **pre:** $h < |\mathcal{B}|$
36: **send** $\langle$REPLY, $\{\langle b_k.header, b_k.cert, b_k.txs\rangle\}_{k=h}^{|\mathcal{B}|}\rangle_i$ **to** j

upon receiving $\langle$REPLY, $\mathcal{M} = \{\langle b_k.header, b_k.cert, b_k.txs\rangle\}_{k=|\mathcal{B}|+1}^{h}\rangle_j$
37: **pre:** $h > |\mathcal{B}| \wedge \forall\langle header, cert, \_\rangle \in \mathcal{M},$ `validCertificate`($cert, header$)
38: **for** each $\langle header, cert, txs\rangle \in \mathcal{M}$
39: `handleNotarization`($header, cert, txs, header.iter$)