Temporal Logic of Composable Distributed Components Appendix

Contents

1 Logic

1.1 Sequent Logic

$$\mathrm{I}\ \frac{}{\mathcal{A} \vdash \mathcal{A}} \qquad \textsc{Thin}\ \frac{\Gamma \vdash \mathcal{A}'}{\Gamma, \mathcal{A} \vdash \mathcal{A}'} \qquad \textsc{Contraction}\ \frac{\Gamma, \mathcal{A}, \mathcal{A} \vdash \mathcal{A}'}{\Gamma, \mathcal{A} \vdash \mathcal{A}'} \qquad \textsc{Exchange}\ \frac{\Gamma, \mathcal{A}, \mathcal{A}' \vdash \mathcal{A}''}{\Gamma, \mathcal{A}', \mathcal{A} \vdash \mathcal{A}''}$$

$$\textsc{Cut}\ \frac{\Gamma \vdash \mathcal{A} \qquad \Gamma, \mathcal{A} \vdash \mathcal{A}'}{\Gamma \vdash \mathcal{A}'} \qquad \neg l\ \frac{\Gamma \vdash \mathcal{A}}{\Gamma, \neg\mathcal{A} \vdash \mathcal{A}'} \qquad \neg r\ \frac{\Gamma, \mathcal{A} \vdash \bot}{\Gamma \vdash \neg\mathcal{A}}$$

$$\wedge l\ \frac{\Gamma, \mathcal{A}, \mathcal{A}' \vdash \mathcal{A}''}{\Gamma, \mathcal{A} \wedge \mathcal{A}' \vdash \mathcal{A}''} \qquad \wedge r\ \frac{\Gamma \vdash \mathcal{A} \qquad \Gamma \vdash \mathcal{A}'}{\Gamma \vdash \mathcal{A} \wedge \mathcal{A}'} \qquad \vee l\ \frac{\Gamma, \mathcal{A} \vdash \mathcal{A}'' \qquad \Gamma, \mathcal{A}' \vdash \mathcal{A}''}{\Gamma, \mathcal{A} \vee \mathcal{A}' \vdash \mathcal{A}''}$$

$$\vee rl\ \frac{\Gamma \vdash \mathcal{A}}{\Gamma \vdash \mathcal{A} \vee \mathcal{A}'} \qquad \vee rr\ \frac{\Gamma \vdash \mathcal{A}'}{\Gamma \vdash \mathcal{A} \vee \mathcal{A}'} \qquad \to l\ \frac{\Gamma \vdash \mathcal{A} \qquad \Gamma, \mathcal{A}' \vdash \mathcal{A}''}{\Gamma, \mathcal{A} \to \mathcal{A}' \vdash \mathcal{A}''}$$

$$\to r\ \frac{\Gamma, \mathcal{A} \vdash \mathcal{A}'}{\Gamma \vdash \mathcal{A} \to \mathcal{A}'} \qquad \forall l\ \frac{\Gamma, \mathcal{A}(x') \vdash \mathcal{A}'}{\Gamma, \forall x.\ \mathcal{A}(x) \vdash \mathcal{A}'} \qquad \forall r\ \frac{\Gamma \vdash \mathcal{A}(x') \qquad x' \text{ fresh}}{\Gamma \vdash \forall x.\ \mathcal{A}(x)}$$

$$\exists l\ \frac{\Gamma, \mathcal{A}(x) \vdash \mathcal{A}' \qquad x' \text{ fresh}}{\Gamma, \exists x.\ \mathcal{A}(x) \vdash \mathcal{A}'} \qquad \exists r\ \frac{\Gamma \vdash \mathcal{A}(x')}{\Gamma \vdash \exists x.\ \mathcal{A}(x)}$$

Figure 1: Basic rules

1.2 Basic Rules

NODE
$\vdash_c \Box\ \mathsf{n} \in \mathbb{N}$

IR
$\vdash_c \top \downarrow e \ \Rightarrow\ (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), e)$

II
$\vdash_c i \uparrow e \ \Rightarrow\ (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{indication}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), (i, e))$

PE
$\vdash_c \top \wr \mathsf{per} \ \Rightarrow\ (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}))$

OR
$\vdash_c n \bullet (i, e) \in \mathsf{ors} \wedge \mathsf{self} \ \Rightarrow\ \hat{\Diamond}(n \bullet i \downarrow e)$

OI
$\vdash_c n \bullet e \in \mathsf{ois} \wedge \mathsf{self} \ \Rightarrow\ \hat{\Diamond}(n \bullet \top \uparrow e)$

OR′
$\vdash_c n \bullet i \downarrow e \ \Rightarrow\ \hat{\Diamond}(n \bullet (i, e) \in \mathsf{ors} \wedge \mathsf{self})$

OI′
$\vdash_c n \bullet \top \uparrow e \ \Rightarrow\ \hat{\Diamond}(n \bullet e \in \mathsf{ois} \wedge \mathsf{self})$

INIT
$\vdash_c \circledS\ (\mathsf{s} = \lambda n.\ \mathsf{init}_c(n))$

POSTPRE
$\vdash_c \circledS\ (\mathsf{s}' = s \ \Leftrightarrow\ \bigcirc\, \mathsf{s} = s)$

SEQ
$\vdash_c \mathsf{n} \neq n \ \Rightarrow\ \mathsf{s}'(n) = \mathsf{s}(n)$

ASELF
$\vdash_c \circledS\ \Box \mathsf{self}$

SINV
$\vdash_c (\circledS\ \mathcal{I}) \ \leftrightarrow\ \mathsf{restrict}(\mathsf{self}, \mathcal{I})$

APER
$\vdash_c n \in \mathsf{Correct} \ \leftrightarrow\ \Box\Diamond(n \bullet \top \wr \mathsf{per})$

Figure 2: Program Logic

RSeq
$\vdash_c \mathsf{r} = r \;\Rightarrow\; \hat{\boxminus}(\mathsf{r} \leq r)$

GST
$\vdash_c n' \in \mathsf{Correct} \wedge r > r_{GST} \wedge$
$(n \bullet d \downarrow \mathsf{send_l}(n', m) \wedge \mathsf{r} = r) \Rightarrow$
$\Diamond(n' \bullet d \uparrow \mathsf{deliver_l}(n, m) \wedge \mathsf{r} = r)$

FDup
$\vdash_c \Box\Diamond(n' \bullet d \uparrow \mathsf{deliver_l}(n, m)) \rightarrow \Box\Diamond(n \bullet d \downarrow \mathsf{send_l}(n', m))$

NForge
$\vdash_c (n' \bullet d \uparrow \mathsf{deliver_l}(n, m)) \Rightarrow \Diamond\!\!\!-(n \bullet d \downarrow \mathsf{send_l}(n', m))$

UniOR
$\vdash_c (\mathsf{occ}(\mathsf{ors}, e) \leq 1 \wedge$
$\hat{\boxminus}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow (i, e) \notin \mathsf{ors}) \wedge$
$\hat{\Box}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow (i, e) \notin \mathsf{ors})) \Rightarrow$
$(n \bullet i \downarrow e) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet i \downarrow e) \wedge \hat{\Box}\neg(n \bullet i \downarrow e)$

UniOI
$\vdash_c (\mathsf{occ}(\mathsf{ois}, e) \leq 1 \wedge$
$\hat{\boxminus}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow e \notin \mathsf{ois}) \wedge$
$\hat{\Box}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow e \notin \mathsf{ois})) \Rightarrow$
$(n \bullet \top \uparrow e) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow e) \wedge \hat{\Box}\neg(n \bullet \top \uparrow e)$

ExeOrderOR
$\vdash_c [n \bullet (i, e) \in \mathsf{ors} \wedge \mathsf{self} \wedge$
$\hat{\Diamond}(n \bullet (i, e') \in \mathsf{ors} \wedge \mathsf{self})] \Rightarrow$
$\hat{\Diamond}[n \bullet i \downarrow e \Rightarrow$
$\hat{\Diamond}(n \bullet i \downarrow e)]$

ExeOrderOI
$\vdash_c [n \bullet e \in \mathsf{ois} \wedge \mathsf{self} \wedge$
$\hat{\Diamond}(n \bullet e' \in \mathsf{ois} \wedge \mathsf{self})] \Rightarrow$
$\hat{\Diamond}[n \bullet \top \uparrow e \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow e)]$

ExeFEOI
$\vdash_c \Box\neg(n \bullet e \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$\Diamond\Box\neg(n \bullet i \uparrow e)$

ExeFEOR
$\vdash_c \Box\neg(n \bullet (i, e) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$\Diamond\Box\neg(n \bullet i \downarrow e)$

Figure 3: Program Logic

1.3 Derived Rules

FLoss
$\vdash_c n' \in \mathsf{Correct} \rightarrow$
$\quad \Box\Diamond(n \bullet d \downarrow \mathsf{send_l}(n', m)) \rightarrow \Box\Diamond(n' \bullet d \uparrow \mathsf{deliver_l}(n, m))$

IRSe
$\vdash_c \circledS\, [\top \downarrow e \;\Rightarrow\; (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), e)]$

PeSe
$\vdash_c \circledS\, [\top \wr \mathsf{per} \;\Rightarrow\; (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}))]$

IISe
$\vdash_c \circledS\, [i \uparrow e \;\Rightarrow\; (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{indication}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), (i, e))]$

ORSe
$\vdash_c \circledS\, [n \bullet (i, e) \in \mathsf{ors} \;\Rightarrow\; \hat{\Diamond}(n \bullet i \downarrow e)]$

OISe
$\vdash_c \circledS\, [n \bullet e \in \mathsf{ois} \;\Rightarrow\; \hat{\Diamond}(n \bullet \top \uparrow e)]$

ORSe′
$\vdash_c \circledS\, [n \bullet i \downarrow e \;\Rightarrow\; \hat{\Diamond}(n \bullet (i, e) \in \mathsf{ors})]$

OISe′
$\vdash_c \circledS\, [n \bullet \top \uparrow e \;\Rightarrow\; \hat{\Diamond}(n \bullet e \in \mathsf{ois})]$

Figure 4: Program Logic, Derived rules. S is a predicate on state.

IROI
$$\frac{\forall s.\ S(s) \wedge \mathsf{request}_c(n,s,e) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c n \bullet \top \downarrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet \top \uparrow e')}$$

IIOI
$$\frac{\forall s.\ S(s) \wedge \mathsf{indication}_c(n,s,(i,e)) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c n \bullet i \uparrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet \top \uparrow e')}$$

PEOI
$$\frac{\forall s.\ S(s) \wedge \mathsf{periodic}_c(n,s) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c n \bullet i \wr \mathsf{per} \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet \top \uparrow e')}$$

IROR
$$\frac{\forall s.\ S(s) \wedge \mathsf{request}_c(n,s,e) = (\_, \_, \mathsf{ors}) \rightarrow (i,e') \in \mathsf{ors}}{\vdash_c n \bullet \top \downarrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet i \downarrow e')}$$

IIOR
$$\frac{\forall s.\ S(s) \wedge \mathsf{indication}_c(n,s,(i,e)) = (\_, \_, \mathsf{ors}) \rightarrow (i,e') \in \mathsf{ors}}{\vdash_c n \bullet i \uparrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet i \downarrow e')}$$

PEOR
$$\frac{\forall s.\ S(s) \wedge \mathsf{periodic}_c(n,s) = (\_, \_, \mathsf{ors}) \rightarrow (i,e') \in \mathsf{ors}}{\vdash_c n \bullet i \wr \mathsf{per} \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond(n \bullet i \downarrow e')}$$

Figure 5: Program Logic, Derived rules. S is a predicate on state.

UniOISe
$\vdash_c \textcircled{S}\ (\mathsf{occ}(\mathsf{ois}, e) \leq 1 \wedge$
$\quad \hat{\boxminus}(\mathsf{n} = n \rightarrow e \notin \mathsf{ois}) \wedge$
$\quad \hat{\square}(\mathsf{n} = n \rightarrow e \notin \mathsf{ois})) \Rightarrow$
$(n \bullet \top \uparrow e) \Rightarrow \hat{\boxminus} \neg (n \bullet \top \uparrow e) \wedge \hat{\square} \neg (n \bullet \top \uparrow e)$

UniORSe
$\vdash_c \textcircled{S}\ (\mathsf{occ}(\mathsf{ors}, e) \leq 1 \wedge$
$\quad \hat{\boxminus}(\mathsf{n} = n \rightarrow (i, e) \notin \mathsf{ors}) \wedge$
$\quad \hat{\square}(\mathsf{n} = n \rightarrow (i, e) \notin \mathsf{ors})) \Rightarrow$
$(n \bullet i \downarrow e) \Rightarrow \hat{\boxminus} \neg (n \bullet i \downarrow e) \wedge \hat{\square} \neg (n \bullet i \downarrow e)$

IROISe
$$\frac{\forall s.\ S(s) \wedge \mathsf{request}_c(n, s, e) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c \textcircled{S}\ n \bullet \top \downarrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet \top \uparrow e')}$$

IIOISe
$$\frac{\forall s.\ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c \textcircled{S}\ n \bullet i \uparrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet \top \uparrow e')}$$

PeOISe
$$\frac{\forall s.\ S(s) \wedge \mathsf{periodic}_c(n, s) = (\_, \mathsf{ois}, \_) \rightarrow e' \in \mathsf{ois}}{\vdash_c \textcircled{S}\ n \bullet i \wr \mathsf{per} \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet \top \uparrow e')}$$

IRORSe
$$\frac{\forall s.\ S(s) \wedge \mathsf{request}_c(n, s, e) = (\_, \_, \mathsf{ors}) \rightarrow (i, e') \in \mathsf{ors}}{\vdash_c \textcircled{S}\ n \bullet \top \downarrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet i \downarrow e')}$$

IIORSe
$$\frac{\forall s.\ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (\_, \_, \mathsf{ors}) \rightarrow (i, e') \in \mathsf{ors}}{\vdash_c \textcircled{S}\ n \bullet i \uparrow e \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet i \downarrow e')}$$

PeORSe
$$\frac{\forall s.\ S(s) \wedge \mathsf{periodic}_c(n, s) = (\_, \_, \mathsf{ors}) \rightarrow (i, e') \in \mathsf{ors}}{\vdash_c \textcircled{S}\ n \bullet i \wr \mathsf{per} \wedge S(\mathsf{s}(n)) \ \Rightarrow\ \Diamond (n \bullet i \downarrow e')}$$

APerSe
$\vdash_c \textcircled{S}\ n \in \mathsf{Correct} \rightarrow \square \Diamond (n \bullet \top \wr \mathsf{per})$

Quorum
$|\mathsf{Correct}| > t_1 \ \vdash_c$
$N \subseteq \mathbb{N} \ \wedge \ |N| > t_2 \ \wedge \ t_1 + t_2 \geq |\mathbb{N}| \Rightarrow \exists n.\ n \in N \wedge n \in \mathsf{Correct}$

Figure 6: Program Logic, Derived rules

SEqSe

$$\vdash_c \textcircled{S}\ [\mathsf{n} \neq n \ \Rightarrow\ \mathsf{s}'(n) = \mathsf{s}(n)]$$

CSelf

$$n \bullet \mathsf{self} \ \Leftrightarrow$$
$$(\exists e.\ n \bullet \top \downarrow e) \ \vee\ (\exists i, e.\ n \bullet \top \uparrow (i, e)) \ \vee\ (n \bullet \top \wr \mathsf{per})$$

InvSe

$$\frac{\Gamma \vdash_c \textcircled{S}\ \forall e.\ \top \downarrow e \Rightarrow \mathcal{A} \quad \Gamma \vdash_c \textcircled{S}\ \forall i, e.\ i \uparrow e \Rightarrow \mathcal{A} \quad \Gamma \vdash_c \textcircled{S}\ \top \wr \mathsf{per} \Rightarrow \mathcal{A}}{\Gamma \vdash_c \textcircled{S}\ \Box\mathcal{A}}$$

InvSe′

$$\frac{\Gamma \vdash_c \textcircled{S}\ \forall e.\ \top \downarrow e \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A} \quad \Gamma \vdash_c \textcircled{S}\ \forall i, e.\ i \uparrow e \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A} \quad \Gamma \vdash_c \textcircled{S}\ \top \wr \mathsf{per} \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A}}{\Gamma \vdash_c \textcircled{S}\ \Box\mathcal{A}}$$

InvUSe

$$\frac{\begin{array}{c}\Gamma \vdash_c \textcircled{S}\ \forall e.\ \top \downarrow e \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), e) \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A} \\ \Gamma \vdash_c \textcircled{S}\ \forall e, i.\ i \uparrow e \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{indication}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), (i, e)) \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A} \\ \Gamma \vdash_c \textcircled{S}\ \top \wr \mathsf{per} \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n})) \wedge \hat{\boxminus}\mathcal{A} \Rightarrow \mathcal{A}\end{array}}{\Gamma \vdash_c \textcircled{S}\ \Box\mathcal{A}}$$

InvMSe

$$\frac{\Gamma \vdash_c \textcircled{S}\ \forall e.\ \top \downarrow e \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}' \quad \Gamma \vdash_c \textcircled{S}\ \forall e, i.\ i \uparrow e \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}' \quad \Gamma \vdash_c \textcircled{S}\ \top \wr \mathsf{per} \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}'}{\Gamma \vdash_c \textcircled{S}\ \Box(\mathcal{A} \wedge \mathcal{A}')}$$

InvMSe′

$$\frac{\begin{array}{c}\Gamma \vdash_c \textcircled{S}\ \forall e.\ \top \downarrow e \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), e) \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}' \\ \Gamma \vdash_c \textcircled{S}\ \forall e, i.\ i \uparrow e \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{indication}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), (i, e)) \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}' \\ \Gamma \vdash_c \textcircled{S}\ \top \wr \mathsf{per} \wedge (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n})) \wedge \hat{\boxminus}\mathcal{A} \wedge \hat{\boxminus}\mathcal{A}' \Rightarrow \mathcal{A} \wedge \mathcal{A}'\end{array}}{\Gamma \vdash_c \textcircled{S}\ \Box(\mathcal{A} \wedge \mathcal{A}')}$$

Figure 7: Program Logic, Derived rules. S is a predicate on state.

InvLSe

$$\frac{\begin{array}{c}\forall e.\ \top \downarrow e \wedge \mathsf{request}_c(\mathsf{n},\mathsf{s}(\mathsf{n}),e) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \forall e,i.\ i \uparrow e \wedge \mathsf{indication}_c(\mathsf{n},\mathsf{s}(\mathsf{n}),(i,e)) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \top \wr \mathsf{per} \wedge \mathsf{periodic}_c(\mathsf{n},\mathsf{s}(\mathsf{n})) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \mathcal{A} \text{ non-temporal}\end{array}}{\vdash_c \textcircled{S}\ \Box\mathcal{A}}$$

InvL

$$\frac{\begin{array}{c}\forall e.\ \top \downarrow e \wedge \mathsf{request}_c(\mathsf{n},\mathsf{s}(\mathsf{n}),e) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \forall e,i.\ i \uparrow e \wedge \mathsf{indication}_c(\mathsf{n},\mathsf{s}(\mathsf{n}),(i,e)) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \top \wr \mathsf{per} \wedge \mathsf{periodic}_c(\mathsf{n},\mathsf{s}(\mathsf{n})) = (\mathsf{s}'(\mathsf{n}),\mathsf{ors},\mathsf{ois}) \to \mathcal{A} \\ \mathcal{A} \text{ non-temporal}\end{array}}{\vdash_c \mathsf{self} \Rightarrow \mathcal{A}}$$

InvUSSe

$$\frac{\begin{array}{c}\Gamma \vdash_c \textcircled{S}\ \forall e.\ n \bullet \top \downarrow e \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{request}_c(n,\mathsf{s}(n),e) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n)) \\ \\ \Gamma \vdash_c \textcircled{S}\ \forall e,i.\ n \bullet i \uparrow e \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{indication}_c(n,\mathsf{s}(n),(i,e)) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n)) \\ \\ \Gamma \vdash_c \textcircled{S}\ n \bullet \top \wr \mathsf{per} \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{periodic}_c(n,\mathsf{s}(n)) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n))\end{array}}{\Gamma \vdash_c \textcircled{S}\,[S(\mathsf{s}(n)) \Rightarrow \Box S(\mathsf{s}(n))]}$$

InvUSSe′

$$\frac{\begin{array}{c}S(\mathsf{init}_c(n)) \\ \\ \Gamma \vdash_c \textcircled{S}\ \forall e.\ n \bullet \top \downarrow e \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{request}_c(n,\mathsf{s}(n),e) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n)) \\ \\ \Gamma \vdash_c \textcircled{S}\ \forall e,i.\ n \bullet i \uparrow e \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{indication}_c(n,\mathsf{s}(n),(i,e)) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n)) \\ \\ \Gamma \vdash_c \textcircled{S}\ n \bullet \top \wr \mathsf{per} \wedge \\ (\mathsf{s}'(n),\mathsf{ors},\mathsf{ois}) = \mathsf{periodic}_c(n,\mathsf{s}(n)) \wedge \\ S(\mathsf{s}(n)) \Rightarrow S(\mathsf{s}'(n))\end{array}}{\Gamma \vdash_c \textcircled{S}\ \Box S(\mathsf{s}(n))}$$

Figure 8: Program Logic, Derived rules

InvSSe

$$\frac{\begin{array}{c}\forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c \textcircled{S}\,[S(\mathsf{s}(n)) \Rightarrow \Box S(\mathsf{s}(n))]}$$

InvS

$$\frac{\begin{array}{c}\forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c \mathsf{self} \wedge S(\mathsf{s}(n)) \Rightarrow (\mathsf{self} \Rightarrow S(\mathsf{s}(n)))}$$

InvSSe′

$$\frac{\begin{array}{c}S(\mathsf{init}_c(n)) \\ \forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c \textcircled{S}\,\Box S(\mathsf{s}(n))}$$

InvS′

$$\frac{\begin{array}{c}S(\mathsf{init}_c(n)) \\ \forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c (\mathsf{self} \Rightarrow S(\mathsf{s}(n)))}$$

Figure 9: Program Logic, Derived rules

$$\textsc{InvSSe}''\quad \frac{\begin{array}{c}\forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c \circledS\ S(\mathsf{s}'(n)) \Rightarrow \hat{\Box} S(\mathsf{s}(n))}$$

$$\textsc{InvS}''\quad \frac{\begin{array}{c}\forall s, e, s'.\ \ S(s) \wedge \mathsf{request}_c(n, s, e) = (s', \_, \_) \rightarrow S(s') \\ \forall s, i, e, s'.\ \ S(s) \wedge \mathsf{indication}_c(n, s, (i, e)) = (s', \_, \_) \rightarrow S(s') \\ \forall s, s'.\ \ S(s) \wedge \mathsf{periodic}_c(n, s) = (s', \_, \_) \rightarrow S(s')\end{array}}{\vdash_c (\mathsf{self} \wedge S(\mathsf{s}'(n))) \Rightarrow \hat{\Box}(\mathsf{self} \rightarrow S(\mathsf{s}(n)))}$$

$$\textsc{InvSASe}\quad \frac{\begin{array}{c}\neg S(\mathsf{init}_c(n)) \\ \\ \forall e.\ n \bullet \top \downarrow e \wedge \\ \mathsf{request}_c(n, \mathsf{s}(n), e) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ \forall e, i.\ n \bullet i \uparrow e \wedge \\ \mathsf{indication}_c(n, \mathsf{s}(n), (i, e)) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ n \bullet \top \wr \mathsf{per} \wedge \\ \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n})) = (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ \mathcal{A} \text{ non-temporal}\end{array}}{\vdash_c \circledS\ [S(\mathsf{s}(n)) \Rightarrow \hat{\Diamond}(n \bullet \mathcal{A})]}$$

$$\textsc{InvSA}\quad \frac{\begin{array}{c}\neg S(\mathsf{init}_c(n)) \\ \\ \forall e.\ n \bullet \top \downarrow e \wedge \\ \mathsf{request}_c(n, \mathsf{s}(n), e) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ \forall e, i.\ n \bullet i \uparrow e \wedge \\ \mathsf{indication}_c(n, \mathsf{s}(n), (i, e)) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ n \bullet \top \wr \mathsf{per} \wedge \\ \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n})) = (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) \wedge \\ \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A} \\ \\ \mathcal{A} \text{ non-temporal}\end{array}}{\vdash_c [\mathsf{self} \wedge S(\mathsf{s}(n))] \Rightarrow \hat{\Diamond}(n \bullet \mathcal{A})}$$

Figure 10: Program Logic, Derived rules

INVSAG

$$\neg S(\mathsf{init}_c(n))$$

$$\forall e.\ n \bullet \top \downarrow e \wedge \mathsf{request}_c(n, \mathsf{s}(n), e) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A}$$

$$\forall e, i.\ n \bullet i \uparrow e \wedge \mathsf{indication}_c(n, \mathsf{s}(n), (i, e)) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \wedge \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A}$$

$$n \bullet \top \wr \mathsf{per} \wedge \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(\mathsf{n})) = (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) \wedge \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n)) \rightarrow \mathcal{A}$$

$$\frac{\mathcal{A} \text{ non-temporal}}{\vdash_c [\mathsf{self} \wedge \neg S(\mathsf{s}(n)) \wedge \Diamond S(\mathsf{s}(n))] \Rightarrow \hat{\Diamond}(n \bullet \mathcal{A})}$$

Figure 11: Program Logic, Derived rules

InvMSIASe

$$\forall x.\ \neg S(\mathsf{init}_c(n))$$

$$\Gamma \vdash_c \circledS\ \forall e.\ [\top \downarrow e \wedge \hat{\boxminus}\mathcal{A} \wedge (\forall x.\ S(\mathsf{s(n)}) \to \hat{\diamondminus}\mathcal{A}')] \Rightarrow \mathcal{A} \wedge (\forall x.\ S(\mathsf{s}'\mathsf{(n)}) \to \diamondminus\mathcal{A}')$$

$$\Gamma \vdash_c \circledS\ \forall i, e.\ [i \uparrow e \wedge \hat{\boxminus}\mathcal{A} \wedge (\forall x.\ S(\mathsf{s(n)}) \to \hat{\diamondminus}\mathcal{A}')] \Rightarrow \mathcal{A} \wedge (\forall x.\ S(\mathsf{s}'\mathsf{(n)}) \to \diamondminus\mathcal{A}')$$

$$\frac{\Gamma \vdash_c \circledS\ [\top \wr \mathsf{per} \wedge \hat{\boxminus}\mathcal{A} \wedge (\forall x.\ S(\mathsf{s(n)}) \to \hat{\diamondminus}\mathcal{A}')] \Rightarrow \mathcal{A} \wedge (\forall x.\ S(\mathsf{s}'\mathsf{(n)}) \to \diamondminus\mathcal{A}')}{\Gamma \vdash_c \circledS\ \Box\mathcal{A} \wedge (\forall x.\ S(\mathsf{s(n)}) \Rightarrow \hat{\diamondminus}\mathcal{A}')}$$

ASASe

$$\frac{\Gamma \vdash_c \circledS\ \mathcal{A} \Rightarrow S(\mathsf{s}'(n)) \qquad \Gamma \vdash_c \circledS\ S(\mathsf{s}(n)) \Rightarrow \Box\mathcal{A}'}{\Gamma \vdash_c \circledS\ \mathcal{A} \Rightarrow \hat{\Box}\mathcal{A}'}$$

ASA

$$\frac{\Gamma \vdash_c \mathsf{self} \wedge \mathcal{A} \Rightarrow S(\mathsf{s}'(n)) \qquad \Gamma \vdash_c \mathsf{self} \wedge S(\mathsf{s}(n)) \Rightarrow (\mathsf{self} \Rightarrow \mathcal{A}') \qquad \mathcal{A} \text{ non-temporal}}{\Gamma \vdash_c \mathsf{self} \wedge \mathcal{A} \Rightarrow \hat{\Box}(\mathsf{self} \to \mathcal{A}')}$$

APerSA

$$\frac{S(\mathsf{s}(n)) \wedge \mathsf{periodic}_c(n, \mathsf{s}(n)) = (\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) \to \mathcal{A} \qquad \mathcal{A} \text{ non-temporal}}{\Gamma \vdash_c n \in \mathsf{Correct} \to (\mathsf{self} \Rightarrow S(\mathsf{s}(n))) \ \Rightarrow\ \Box\Diamond\mathcal{A}}$$

Figure 12: Program Logic, Derived rules

2 Reasoning about Quorums

We have successfully applied the programming model, the lowering transformation and TLC to program and verify the stacks of distributed components shown in Figure 2.(b) and (c): stubborn links, perfect links, best-effort broadcast, uniform reliable broadcast and epoch consensus. The specification and implementation of these components are available in section 4 and the detailed proofs are available in subsection 5.3. In this section, we present the uniform reliable broadcast component and present how its proof of uniformity uses the quorum inference rule. Uniform reliable broadcast guarantees that if a message is delivered to a node (even a faulty one), then it is delivered to every correct node as well. Uniform reliable broadcast guarantees that the set of messages delivered to correct nodes is the same and is a superset of the messages delivered to faulty nodes; hence the name uniform.

Uniform reliable broadcast accepts requests $\mathsf{broadcast}_{\mathsf{urb}}(m)$ to broadcast the message m and issues indications $\mathsf{deliver}_{\mathsf{urb}}(n, m)$ to deliver a message m broadcast by a node n. Figure 13.(a) presents the specification of the uniform reliable broadcast. In particular, the uniform agreement property states that if a message is delivered to a node, a correct node does not miss it. More precisely, if a message m is delivered to some node (whether correct or faulty), then m is eventually (in the past or future) delivered to every correct node. In this section, we focus on this property. Uniform reliable broadcast also provides the following properties. Validity: a correct node eventually receives his own broadcast messages. No-duplication: Messages are not redundantly delivered. No-forge: delivered messages are previously broadcast.

Figure 13.(b) presents the uniform reliable broadcast component URBC. It assumes that a majority of nodes is correct i.e. more than $|\mathbb{N}|/2$ nodes are correct (where $\mathbb{N}$ is the set of nodes). The high-level idea of the protocol is that each node, before delivering a message, makes sure that a quorum (majority) of nodes have acknowledged the receipt of the message. As a majority of nodes are correct, there is at least one correct node in the acknowledging nodes. Even if the sender fails, that correct node rebroadcasts the message that leads to its delivery to every correct node.

The component uses a best-effort broadcast subcomponent bebc. It stores the number of messages sent by the current node *count*, the set of delivered messages *delivered*, the set of messages that are received but are pending for acknowledgement *pending*, and a mapping from each message to the set of nodes that have acknowledged the receipt of the message *ack*. Each message is uniquely identified by the identifier of the sender node and the number of the message in that node. Messages are transmitted and stored with their identifiers. The state *count* is initialized to zero, and the other sets are all initialized to $\varnothing$.

Upon a broadcast request, the counter is incremented, the message is added to the pending set and is broadcast using bebc together with the current node identifier and the new value of the counter. Upon a delivery indication by bebc, it is recorded in the acknowledgement map that the sender has acknowledged the receipt of the message and if the message is not already in the pending set, it is added to the pending set and broadcast by bebc. Thus, every node broadcasts a message by bebc only once when it receives it for the first time. In the periodic function, messages in the pending set are iterated, and if an acknowledgement for a message is received from a quorum of nodes, and it is not delivered before, then it is delivered.

We present a proof sketch for the uniform agreement property. Let Γ be the assumption that a quorum of nodes is correct and the lowered specification of the best-effort broadcast.

$$\Gamma = |\mathsf{Correct}| > |\mathbb{N}|/2;\ \mathcal{A}_{\mathsf{bebc}} \tag{1}$$

The premise is that a message is delivered to a node. Thus the message is previously issued. The component issues delivery of a message only in the periodic function when the acknowledgement set

Assumption:
$|\mathsf{Correct}| > |\mathbb{N}|/2$

URB_1 (Validity)
$n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \rightsquigarrow$
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$
If a correct node n broadcasts a message m, then n itself eventually delivers m.

URB_2 (No-duplication)
$[n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m) \Rightarrow$
$\quad \hat{\boxminus} \neg (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))] \rightarrow$
$[(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m)) \Rightarrow$
$\quad \hat{\boxminus} \neg (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))]$
If a message is broadcast at most once, it will be delivered at most once.

URB_3 (No-forge)
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$
If a node delivers a message m with sender n', then m was previously broadcast by node n'.

URB_4 (Uniform Agreement)
$n \in \mathsf{Correct} \rightarrow$
$(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m)) \Rightarrow$
$\Diamond (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m))$
If a message m is delivered by some node (whether correct or faulty), then m is eventually delivered by every correct node.

(a) Specification

```
URBC: Component Req_urb Ind_urb (Req_beb, Ind_beb) :=
    let bebc := 0 in
    ⟨State :=
        ⟨count: Nat,
         delivered: Set[⟨M, ℕ, Nat⟩],
         pending: Set[⟨M, ℕ, Nat⟩],
         ack: Map[⟨M, ℕ, Nat⟩, Set[ℕ]] ⟩
     init := λn. ⟨0, ∅, ∅, ∅⟩
     request := λ n, s, ir.
        let ⟨c, d, p, a⟩ := s in
        match ir with
        | broadcast_urb(m) ⇒
            let c' := c + 1 in
            let p' := p ∪ {⟨m, n, c'⟩} in
            let or := (bebc, broadcast_beb⟨m, n, c'⟩) in
            ⟨⟨ c', d, p', a⟩, [or], [] ⟩
        end
     indication := λ n', s, ii.
        let ⟨c, d, p, a⟩ := s in
        match ii with
        | (bebc, deliver_beb(n'', ⟨m, n, c'⟩)) ⇒
            let a' := a[⟨m, n, c'⟩ ↦ a(⟨m, n, c'⟩) ∪ {n''}] in
            if ⟨m, n, c'⟩ ∉ p
                let p' := p ∪ {⟨m, n, c'⟩} in
                let or := (bebc, broadcast_beb⟨m, n, c'⟩) in
                ⟨⟨c, d, p', a'⟩, [or], []⟩
            else
                ⟨⟨c, d, p, a'⟩, [], [] ⟩
        end
     periodic := λ n, s.
        let ⟨c, d, p, a⟩ := s in
        let ⟨d', ois⟩ := foldl (
            (λ ⟨d', ois⟩, ⟨m, n', c'⟩.
                if |a(⟨m, n', c'⟩)| > |ℕ|/2 ∧ ⟨m, n', c'⟩ ∉ d
                    ⟨d' ∪ {⟨m, n', c'⟩}, ois :: deliver_urb(n', m)⟩
                else
                    ⟨d', ois⟩),
            ⟨∅, []⟩,
            p ) in
        ⟨⟨c, d ∪ d', p, a⟩, [], ois ⟩ ⟩
```

(b) Implementation

Figure 13: Uniform Reliable Broadcast Component

of the message is a quorum. The other handler functions do not issue delivery events. Thus, by rule InvLSe, we have

$$\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n' \bullet \mathsf{deliver}_{\mathsf{urb}}(n'', m) \in \mathsf{ois}) \Rightarrow \exists c.\ |ack(\mathsf{s}(n'))(\langle m, n'', c\rangle)| > |\mathbb{N}|/2 \tag{2}$$

that states that when a delivery is issued for a message, the size of its acknowledgement set is more than half of the number of nodes. As a quorum of nodes are correct, there should be at least one correct node in the acknowledging set. We sketch the proof of this fact using the Quorum. We can separately show that every element of the acknowledgement set is a node.

$$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \Box\ ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle) \subseteq \mathbb{N} \tag{3}$$

and obviously

$$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \Box\ (\mathbb{N}/2 + \mathbb{N}/2 \geq \mathbb{N}) \tag{4}$$

By rule Quorum on Equation 1, Equation 3 and Equation 4, we have

$$\Gamma \vdash_{\mathsf{URBC}} \circledS\ |ack(\mathsf{s}(n'))(\langle m, n'', c\rangle)| > |\mathbb{N}|/2 \Rightarrow \exists n.\ n \in \mathsf{Correct} \wedge n \in ack(\mathsf{s}(n'))(\langle m, n'', c\rangle) \tag{5}$$

that states that if the size of the acknowledgement set for a message is more than half of the number of nodes, then a correct node is in the set. We note that reasoning about quorums is done in a simple single step. Thus, from Equation 2 and Equation 5, we have

$$\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n' \bullet \mathsf{deliver}_{\mathsf{urb}}(n'', m) \in \mathsf{ois}) \Rightarrow \exists c, n.\ n \in \mathsf{Correct} \wedge n \in ack(\mathsf{s}(n'))(\langle m, n'', c\rangle) \tag{6}$$

We now present a summary of the rest of the proof that is available in subsection 5.3. We use the following properties of bebc. Validity: If a correct node broadcasts a message, then the message is eventually delivered to every correct node. No-forge: Delivered messages are previously broadcast.

We know that a correct node is in the acknowledgement set. The component adds a node to the acknowledgement set only if a message is delivered from that node via bebc. By the no-forge property of bebc, the message should have been broadcast. Thus, a correct node has broadcast the message. By the validity property of bebc, the message is delivered to every correct node. When the component receives a message via bebc, if the message is not already in the pending set, it is rebroadcast. If it is already in the pending set, it can be shown that it is already broadcast. Thus, in any case, every correct node eventually (in the past or future) broadcasts the message. Thus, by the validity property of bebc, the message is delivered to every correct node from every correct node via bebc. When the component receives a message via bebc, it adds the sender to the acknowledgement set. The elements of the acknowledgement set always stay the same or increase. Thus, eventually forever, every correct node will have every correct node in its acknowledgement set for the message. As a majority of nodes is correct, the size of this set is more than half of the number of the nodes. As the periodic function is infinitely often called, it will be eventually called when the size of the acknowledgement set for the message is more than half of the number of the nodes. When the periodic function iterates the pending set, if the message is not in the delivered set, as its acknowledgement set is already large enough, it is delivered. On the other hand, if it is in the delivered set, it can be shown that it is already delivered. Thus, the message is eventually (in the past or future) delivered at every correct node.

3 Lowering Soundness

The following theorem states the soundness of the lowering transformation for compositional reasoning. If a top-level invariant $\mathcal{I}_i^{[]}$ is valid for the stack $\mathcal{S}_i$ and $\mathcal{S}_i$ is a substack of the stack $\mathcal{S}$, then the lowered invariant $\mathsf{lower}(i, \mathcal{I}_i^{[]})$ is valid for $\mathcal{S}$. We use the validity judgement $\models_{\mathcal{S}} \mathcal{A}$ that states that $\mathcal{A}$ is valid in every trace of $\mathcal{S}$. (Validity is defined more precisely in section 7, and the detailed proofs are available in subsection 5.2.)

Theorem 3. *For all $\mathcal{S}$, c, and $\overline{\mathcal{S}_i}$, such that $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i})$, if $\models_{\mathcal{S}_i} \mathcal{I}_i^{[]}$ then $\models_{\mathcal{S}} \mathsf{lower}(i, \mathcal{I}_i^{[]})$.*

We now state the *compositional proof technique* and its soundness. The specifications of substacks can be lowered and used to derive the specification of the stack. Judgements of TLC are of the form $\Gamma \vdash_c \mathcal{A}$ where Γ is the assumed assertions and $\mathcal{A}$ is the deduced assertion. Consider valid top-level invariants $\overline{\mathcal{I}_i^{[]}}$ for stacks $\overline{\mathcal{S}_i}$ and a stack $\mathcal{S}$ built by the component c on top of $\overline{\mathcal{S}_i}$. The following theorem states that assuming the lowered invariants $\overline{\mathsf{lower}(i, \mathcal{I}_i^{[]})}$, any assertion that TLC deduces for c is valid for $\mathcal{S}$.

Corollary 1 (Composition Soundness)**.** *For all $\mathcal{S}$, c, and $\overline{\mathcal{S}_i}$ such that $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i})$, if $\overline{\models_{\mathcal{S}_i} \mathcal{I}_i^{[]}}$ and $\overline{\mathsf{lower}(i, \mathcal{I}_i^{[]})} \vdash_c \mathcal{A}$ then $\models_{\mathcal{S}} \mathcal{A}$.*

Let us present an overview of why the above theorem holds. We first define a few helper definitions and lemmas.

The transitions of the operational semantics generate labels ℓ that are tuples of (n, d, o, e, σ, σ', ors, ois). A label represents the execution of an event and its components are the node identifier n, the location d, the orientation o, the user event e, the pre-state σ, the post-state σ', issued output requests ors and issued output indications ois of the executed event. A trace τ is a sequence of labels. The set of traces of a stack $\mathcal{S}$ is denoted as $T(\mathcal{S})$. To determine satisfiability of temporal assertions, we define a model $m = (\tau, i, I)$ as a tuple of a trace τ, a position i in the trace and an interpretation I (to evaluate rigid free variables). The set of models of a stack $\mathcal{S}$ is denoted as $M(\mathcal{S})$. (We elaborate these definitions in section 6 and section 7). A trace is pushed to a branch i by appending i to the location of its events. Pushing is naturally lifted to a model by pushing its trace.

Definition 7 (Pushing a Trace and a Model)**.**

$$\begin{array}{rcl} \mathsf{push}(i, \tau) & \triangleq & \mathsf{map}\,(\lambda(d, e, n, s, s', ors, ois). \\ & & \qquad (i :: d, e, n, s, s', ors, ois), \tau) \\ \mathsf{push}(i, (\tau, j, I)) & \triangleq & (\mathsf{push}(i, \tau), j, I) \end{array}$$

The following lemma intuitively states that a substack generates a subset of the traces that it generates at the top level (modulo the location of events). More precisely, for every stack $\mathcal{S}$ with a substack $\mathcal{S}_i$, for any trace τ of $\mathcal{S}$, there exists a trace τ' of $\mathcal{S}_i$, such that subtrace of τ for $\mathcal{S}_i$ is equal to τ' pushed to branch i. We use $\tau|_{d \sqsupseteq [i]}$ to denote the projection of τ over events whose location is an extension of $[i]$ i.e. the events that are from the stack at location $[i]$. This projection is simply lifted to models.

Lemma 1. *For all c, $\overline{\mathcal{S}_i}$ and τ, if $\tau \in T(\mathsf{stack}(c, \overline{\mathcal{S}_i}))$, there exists $\tau' \in T(\mathcal{S}_i)$ such that $\tau|_{d \sqsupseteq [i]} = \mathsf{push}(i, \tau')$.*

The above lemma is proved using a simulation from the semantics of $\mathsf{stack}(c, \overline{\mathcal{S}_i})$ to the semantics of $\mathcal{S}_i$. The following corollary for models is immediate from the definition of push on models (Definition 7).

Corollary 2. *For all c, $\overline{\mathcal{S}_i}$ and m, if $m \in M(\mathsf{stack}(c, \overline{\mathcal{S}_i}))$, there exists $m' \in M(\mathcal{S}_i)$ such that $m|_{d \supseteq [i]} = \mathsf{push}(i, m')$.*

The following lemma states that if a model m satisfies a top-level assertion $\mathcal{A}^{[]}$, then the model resulted from pushing m to branch i satisfies the assertion resulted from pushing $\mathcal{A}^{[]}$ to branch i.

Lemma 2. *For all m, $\mathcal{A}^{[]}$ and i, if $m \vDash \mathcal{A}^{[]}$ then $\mathsf{push}(i, m) \vDash \mathsf{push}(i, \mathcal{A}^{[]})$.*

The above lemma is proved by induction on the structure of $\mathcal{A}^{[]}$. We now define extensions of a model that preserve the subtrace of branch i. A model is extended when its trace is extended. A model $(\tau'', 0, I)$ is an extension of the model $(\tau, 0, I)$ if τ'' is an interleaving of τ with another trace τ'; to preserve the events under branch i in τ, there should not be any event under branch i in τ'.

Definition 8 (Extending a Model)**.**

$$\mathsf{extend}((\tau, 0, I), i) \triangleq \{(\tau'', 0, I) \mid \exists \tau'.\ \tau'' \in \mathsf{interleave}(\tau, \tau') \wedge \forall j.\ d(\tau'(j)) \not\supseteq [i]\}$$

Based on the above definition, the following lemma states that if a model m satisfies an invariant $\mathcal{I}^{[i]}$, then any extension of m (that preserves the events of branch i) satisfies the restriction of $\mathcal{I}^{[i]}$ to branch i.

Lemma 3. *For all m, i, $\mathcal{I}^{[i]}$, m', if $m \vDash \mathcal{I}^{[i]}$ and $m' \in \mathsf{extend}(m, i)$ then $m' \vDash \mathsf{restrict}(\mathsf{d} \supseteq [i], \mathcal{I}^{[i]})$.*

A generalization of this lemma is proved by structural induction on the invariant $\mathcal{I}^{[i]}$. To prove this lemma for the not operator, the lemma is generalized to bi-implication. As we mentioned before, the next operator was excluded from the sublanguage of invariants. If we had mapped the next operator to the eventually operator, the forward implication would but the backward implication would not hold.

Now, use apply the above lemmas, to present a proof sketch for Theorem 3. From $\vDash_{\mathcal{S}_i} \mathcal{I}_i^{[]}$, we have that for every model m' of $\mathcal{S}_i$, (1) $m' \vDash \mathcal{I}_i^{[]}$. By Lemma 2 on [1], both the model and the invariant can be pushed i.e. (2) $\mathsf{push}(i, m') \vDash \mathsf{push}(i, \mathcal{I}_i^{[]})$. By Corollary 2 on [2], for every model m in $M(\mathcal{S})$, there exists a model m' in $M(\mathcal{S}_i)$ such that (3) the projection $m|_{d \supseteq [i]}$ is equal to $\mathsf{push}(i, m')$. Thus, from [2] and [3], we have (4) $m|_{d \supseteq [i]} \vDash \mathsf{push}(i, \mathcal{I}_i^{[]})$. The whole model m is an extension of the projection that is (5) $m \in \mathsf{extend}(i, m|_{d \supseteq [i]})$. Thus, by Lemma 3 on [4] and [5], we have that m models the restriction of $\mathsf{push}(i, \mathcal{I}_i^{[]})$ i.e. $m \vDash \mathsf{restrict}(\mathsf{d} \supseteq [i], \mathsf{push}(i, \mathcal{I}_i^{[]}))$ that by definition of lower (Definition 1) is $m \vDash \mathsf{lower}(i, \mathcal{I}_i^{[]})$. Therefore, as m is an arbitrary model of $\mathcal{S}$, we have $\vDash_{\mathcal{S}} \mathsf{lower}(i, \mathcal{I}_i^{[]})$.

4 Components

4.1 Stubborn Links

Figure 14 shows the specification of stubborn links. It accepts requests $\mathsf{send}_{\mathsf{sl}}(n, m)$ to send the message m to the node n. It issues indications $\mathsf{deliver}_{\mathsf{sl}}(n, m)$ to deliver the message m sent by the node n. A stubborn link stubbornly retransmits messages. The stubborn delivery property states that once a message is sent, it is delivered infinitely often. More precisely, if a correct node n sends a message m to a correct node n', then n' delivers m infinitely often. A stubborn link never forges a message. More precisely, if a node n delivers a message m with sender n', then m was in fact previously sent to n by n'.

Figure 15 shows the component SLC that implements the stubborn link. It uses the underlying basic link to retransmit messages. Its state stores the set of sent messages as the set sent of pairs of destination and message with the initial state $\varnothing$. Upon receiving a request $\mathsf{send}_{\mathsf{sl}}(n', m)$, the pair $\langle n', m \rangle$ is added to the sent set and a $\mathsf{send}_{\mathsf{l}}(n', m)$ request is issued to the lower-level basic link. Upon receiving an indication $\mathsf{deliver}_{\mathsf{l}}(n, m)$ from the basic link, a $\mathsf{deliver}_{\mathsf{sl}}(n, m)$ indication is issued. The basic link transmits the message with losses. Therefore, the messages in the sent set are periodically resent by the basic link.

Stubborn Link Interface

$\mathsf{Req_{sl}} \coloneqq \mathsf{send_{sl}}(n, m)$	Requests to send message m to node n
$\mathsf{Ind_{sl}} \coloneqq \mathsf{deliver_{sl}}(n, m)$	Delivers message m sent by node n

SL_1 (Stubborn delivery):

$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow \Box\Diamond(n' \bullet \top \uparrow \mathsf{deliver_{sl}}(n, m))$

If a correct node n sends a message m to a correct node n', then n' delivers m infinitely often.

SL_2 (No-forge):

$(n \bullet \top \uparrow \mathsf{deliver_{sl}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{send_{sl}}(n, m))$

If a node n delivers a message m with sender n', then m was previously sent to n by n'.

Figure 14: Stubborn Links Specification

```
SLC: Component Req_sl Ind_sl (Req_l, Ind_l) ≔
    let lc ≔ 0 in
    ⟨State ≔ ⟨sent: Set[⟨ℕ, M⟩]⟩,
     init ≔ λn. ∅,
     request ≔ λ n, s, ir.
          match ir with
          | send_sl(n', m) ⇒
               ⟨⟨s ∪ {⟨n', m⟩}⟩,
                [(lc, send_l(n', m))],
                []⟩
          end
     indication ≔ λ n', s, ii.
          match ii with
          | (lc, deliver_l(n, m) ⇒
               ⟨s, [], [deliver_sl(n, m)]⟩
          end
     periodic ≔ λ n, s.
          let ors ≔ map (λ⟨n, m⟩. (lc, send_l(n, m))) s in
          ⟨s, ors, []⟩⟩
```

Figure 15: Stubborn Link Component

4.2 Perfect Links

The specification of perfect links is presented in Figure 16. Perfect links accept requests $\mathsf{send}_{\mathsf{pl}}(n, m)$ to send the message m to the node n and issue indications $\mathsf{deliver}_{\mathsf{pl}}(n, m)$ to deliver a message m sent by a node n. The reliable delivery property states that perfect links can reliably transmit messages between correct nodes. More precisely, it states that if a correct node n sends a message m to a correct node n', then n' will eventually deliver m. The no-duplication property states that perfect links do not redundantly deliver messages. More precisely, it states that if a message is sent at most once, it will be delivered at most once. The no-forge property states that perfect links do not forge messages. More precisely, it states that if a node n delivers a message m with sender n', then m was previously sent to n by node n'.

Figure 17 presents the component PLC that implements the perfect link. It uses a stubborn link as the lower-level component. As the stubborn link delivers messages infinitely often, the component keeps track of delivered messages and ignores redelivered messages. It stores the number of messages sent by the current node counter initialized to zero and the set of received messages received initialized to empty. The counter is used to uniquely assign a number to each message sent by the same node. The set of received messages are the pairs of node identifier and the number of the message in that node. Upon a request to send a message, the counter is incremented and the message is sent using the stubborn link subcomponent with the new counter value. Upon an indication of delivery of a message from the stubborn link subcomponent, if the message is already received, it is ignored. Otherwise, the pair of the sending node and the message number are added to the received set and the component delivers the message.

Perfect Link Interface

$\mathsf{Req}_{\mathsf{pl}} \coloneqq \mathsf{send}_{\mathsf{pl}}(n, m)$	Requests to send message m to node n
$\mathsf{Ind}_{\mathsf{pl}} \coloneqq \mathsf{deliver}_{\mathsf{pl}}(n, m)$	Delivers message m sent by node n

PL_1 (Reliable delivery):

$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$

$(n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

If a correct node n sends a message m to a correct node n', then n' will eventually deliver m.

PL_2 (No-duplication):

$[n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m) \Rightarrow$

$\quad \hat{\boxminus} \neg (n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))] \rightarrow$

$[n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$

$\quad \hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))]^*$

If a message is sent at most once, it will be delivered at most once.

PL_3 (No-forge):

$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))$

If a node n delivers a message m with sender n', then m was previously sent to n by node n'.

*It is notable that $(p \Rightarrow \hat{\boxminus} \neg p) \rightarrow (p \Rightarrow \hat{\Box} \neg p)$

Figure 16: Perfect Links Specification

```
PLC: Component Req_pl Ind_pl (Req_sl, Ind_sl) :=
    let slc := 0 in
    ⟨State := ⟨counter: Nat,
                received: Set[⟨ℕ, Nat⟩]⟩,
     init := λn. ⟨0, ∅⟩,

     request := λ n, s, ir.
         let ⟨c, r⟩ := s in
         match ir with
         | send_pl(n′, m) ⇒
             let c′ := c + 1 in
             let or := (slc, send_sl(n′, ⟨c′, m⟩)) in
             ⟨⟨c′, r⟩, [or], []⟩,
         end

     indication := λ n′, s, ii.
         let ⟨c, r⟩ =: s in
         match ii with
         | (slc, deliver_sl(n, ⟨c′, m⟩) ⇒
             if (⟨n, c′⟩ ∈ r)
                 ⟨s, [], []⟩
             else
                 let r′ := r ∪ {⟨n, c′⟩} in
                 let oi := deliver_pl(n, m) in
                 ⟨⟨c, r′⟩, [], [oi]⟩
         end

     periodic := λ n, s. ⟨s, [], []⟩ ⟩
```

Figure 17: Perfect Link Component

4.3 Best-Effort Broadcast

The specification of best-effort broadcast is presented in Figure 18. Best-effort broadcast accepts requests $\mathsf{broadcast}_{\mathsf{beb}}(m)$ to broadcast the message m and issues indications $\mathsf{deliver}_{\mathsf{beb}}(n, m)$ to deliver a message m broadcast by a node n. The validity property states that the best-effort broadcast delivers the same set of messages from every correct node to every correct node. More precisely, it states that if a correct node broadcasts a message m, then every correct node eventually delivers m. The no-duplication property states that the best-effort broadcast does not redundantly deliver messages. More precisely, it states that if a message is broadcast at most once, it will be delivered at most once. The no-forge property states that the best-effort broadcast does not forge messages. More precisely, it states that if a correct node delivers a message m with sender n', then m was previously broadcast by node n'.

Figure 19 presents the component BEBC that implements the best-effort broadcast. It uses a perfect link as the lower-level component. It does not store any state. Upon a request to broadcast a message, it sends the message to every node using the perfect-link subcomponent. Upon an indication of delivery of a message from the perfect link subcomponent, the component delivers the message.

Best-Effort Broadcast

$\mathsf{Req}_{\mathsf{beb}} \coloneqq \mathsf{broadcast}_{\mathsf{beb}}(m)$	Broadcast a message m to all nodes.
$\mathsf{Ind}_{\mathsf{beb}} \coloneqq \mathsf{deliver}_{\mathsf{beb}}(n, m)$	Delivers a message m broadcast by node n.

BEB_1 (Validity)
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \rightsquigarrow$
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$
If a correct node broadcasts a message m, then every correct node eventually delivers m.

BEB_2 (No-duplication)
$[n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow$
$\hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))] \rightarrow$
$[n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$
If a message is broadcast at most once, it will be delivered at most once.

BEB_3 (No-forge)
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
If a correct node delivers a message m with sender n', then m was previously broadcast by node n'.

Figure 18: Best-Effort Broadcast Specification

```
BEBC: Component Req_beb Ind_beb (Req_pl, Ind_pl) ≔
    let plc ≔ 0 in
    ⟨State ≔ Unit
     init  ≔ λn. ⊥
     request  ≔ λ n, s, ir.
          match ir with
          | broadcast_beb(m) ⇒
               let ors ≔ map (λn. (plc, send_pl(n, m))) ℕ in
               ⟨s, ors, []⟩
          end
     indication ≔ λ n′, s, ii.
          match ii with
          | (plc, deliver_pl(n, m)) ⇒
               ⟨s, [], [deliver_beb(n, m)]⟩
          end
     periodic ≔ λ n, s.
          ⟨s, [], []⟩⟩
```

Figure 19: Best-Effort Broadcast Component

4.4 Uniform Reliable Broadcast

Uniform reliable broadcast guarantees that if a node (even a faulty one) delivers a message, then every correct node also delivers it. It precludes the situation where a faulty node delivers a message and failes and a correct node never delivers the message. Uniform reliable broadcast guarantees that the set of messages delivered by correct nodes is always a superset of the messages delivered by faulty nodes; hence, it is called uniform.

Figure 20 presents the specification of uniform reliable broadcast. It accepts requests $\mathsf{broadcast}_{\mathsf{urb}}(m)$ to broadcast the message m and issues indications $\mathsf{deliver}_{\mathsf{urb}}(n, m)$ to deliver a message m broadcast by a node n. The validity property states that a correct node receives his own messages. More precisely, it states that if a correct node n broadcasts a message m, then n itself eventually delivers m. The no-duplication property states that messages are not redundantly delivered. More precisely, it states that if a message is broadcast at most once, it will be delivered at most once. The no-forge property states that messages are not forged. More precisely, it states that if a node delivers a message m with sender n', then m was previously broadcast by node n'. The uniform agreement states that if a node delivers a message, a correct node does not miss it. More precisely, it states that if a message m is delivered by some node (whether correct or faulty), then m is eventually delivered by every correct node.

Uniform Reliable Broadcast
$\mathsf{Req}_{\mathsf{urb}} \coloneqq \mathsf{broadcast}_{\mathsf{urb}}(m)$
Broadcasts a message m to all nodes.
$\mathsf{Ind}_{\mathsf{urb}} \coloneqq \mathsf{deliver}_{\mathsf{urb}}(n, m)$
Delivers a message m broadcast by node n.
Assumption:
$|\mathsf{Correct}| > |\mathbb{N}|/2$

URB_1 (Validity)
$n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \rightsquigarrow$
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$
If a correct node n broadcasts a message m, then n itself eventually delivers m.

URB_2 (No-duplication)
$[n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))] \rightarrow$
$[(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m)) \Rightarrow$
$\hat{\boxminus} \neg (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))]$
If a message is broadcast at most once, it will be delivered at most once.

URB_3 (No-forge)
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$
If a node delivers a message m with sender n', then m was previously broadcast by node n'.

URB_4 (Uniform Agreement)
$n \in \mathsf{Correct} \rightarrow$
$(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m)) \Rightarrow$
$\Diamond (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n'', m))$
If a message m is delivered by some node (whether correct or faulty), then m is eventually delivered by every correct node.

Figure 20: Uniform Reliable Broadcast Specification

Figure 21 presents the uniform reliable broadcast component URBC that implements the uniform reliable broadcast. It uses a best-effort broadcast beb and assumes that a majority of processes are correct. It stores the number of messages sent by the current node *count*, the set of delivered messages *delivered*, the set of messages that are received but are pending for acknowledgement *pending*, and a mapping from each message to the set of nodes that have acknowledged the receipt of the message *ack*. Each message is uniquely identified by the identifier of the sender node and the number of the message in that node. The state *count* is initialized to zero, and the other sets are all initialized to $\varnothing$.

The high-level idea is that each node, before delivering a message, makes sure that a majority of nodes have acknowledged the receipt of the message. As a majority of nodes are correct, there is at least one correct node in the acknowledging nodes. Not only the sender but every receiver broadcasts the message. Thus, the correct acknowledging node broadcasts the message as well and its broadcast will be properly delivered by every correct node. The other correct nodes will in turn broadcast the message. Thus, every correct node eventually receives an acknowledgement from every correct node. There is a majority of correct nodes. Thus, each correct node will eventually receive acknowledgement from a majority of nodes and deliver the message.

Upon a broadcast request, the counter is incremented, the message is added to the pending set and is broadcast using beb together with the sender identifier and the new value of the counter. Upon a delivery indication by beb, it is recorded in the acknowledgement map that the sender has acknowledged the receipt of the message and if the message is not already in the pending set, it is added to the pending set and broadcast by beb. Every node broadcasts each message by beb only once when it receives it for the first time. In the periodic function, messages in the pending set are iterated, and if an acknowledgement for a message is received from a majority of processes, and it is not delivered before, then it is delivered.

```
URBC: Component Req_urb Ind_urb (Req_beb, Ind_beb) ≔
    let bebc ≔ 0 in
    ⟨State ≔
        ⟨count: Nat,
         delivered: Set[⟨M, ℕ, Nat⟩],
         pending: Set[⟨M, ℕ, Nat⟩],
         ack: Map[⟨M, ℕ, Nat⟩, Set[ℕ]] ⟩
     init  ≔ λn. ⟨0, ∅, ∅, ∅⟩
     request  ≔ λ n, s, ir.
        let ⟨c, d, p, a⟩ ≔ s in
        match ir with
        | broadcast_urb(m) ⇒
            let c′ ≔ c + 1 in
            let p′ ≔ p ∪ {⟨m, n, c′⟩} in
            let or ≔ (bebc, broadcast_beb⟨m, n, c′⟩) in
            ⟨⟨ c′, d, p′, a⟩, [or], [] ⟩
        end
     indication ≔ λ n′, s, ii.
        let ⟨c, d, p, a⟩ ≔ s in
        match ii with
        | (bebc, deliver_beb(n″, ⟨m, n, c′⟩)) ⇒
            let a′ ≔ a[⟨m, n, c′⟩ ↦ a(⟨m, n, c′⟩) ∪ {n″}] in
             if ⟨m, n, c′⟩ ∉ p
                let p′ ≔ p ∪ {⟨m, n, c′⟩} in
                let or ≔ (bebc, broadcast_beb⟨m, n, c′⟩) in
                ⟨⟨c, d, p′, a′⟩, [or], []⟩
            else
                ⟨⟨c, d, p, a′⟩, [], [] ⟩
        end
     periodic ≔ λ n, s.
        let ⟨c, d, p, a⟩ ≔ s in
        let ⟨d′, ois⟩  ≔ foldl (
            (λ ⟨d′, ois⟩, ⟨m, n′, c′⟩.
                if |a(⟨m, n′, c′⟩)| > |ℕ|/2 ∧ ⟨m, n′, c′⟩ ∉ d
                    ⟨d′ ∪ {⟨m, n′, c′⟩}, ois :: deliver_urb(n′, m)⟩
                else
                    ⟨d′,  ois⟩),
            ⟨∅, []⟩,
            p ) in
        ⟨⟨c, d ∪ d′, p, a⟩, [], ois ⟩ ⟩
```

Figure 21: Uniform Reliable Broadcast Component

4.5 Eventually Perfect Failure Detector

Eventually Perfect Failure Detector

$\mathsf{Req}_{\mathsf{epfd}} = \mathsf{Unit}$	None.
$\mathsf{Ind}_{\mathsf{epfd}} \coloneqq \mathsf{suspect}(n)$	The indication that the node n is suspected to have crashed.
$\mid\ \mathsf{restore}(n)$	The indication that the node n is not suspected anymore.

EPFD_1 (Strong Completeness):
$\forall n, n'.\ n \in \mathsf{Correct} \wedge \neg n' \in \mathsf{Correct} \rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{suspect}(n')) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{restore}(n'))]$
Every incorrect node is eventually permanently suspected by every correct node.

EPFD_2 (Eventual Strong Accuracy):
$\forall n, n'.\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\Box\neg(n \bullet \top \uparrow \mathsf{suspect}(n')) \vee$
$\Diamond[(n \bullet \top \uparrow \mathsf{restore}(n')) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{suspect}(n'))]$
Eventually no correct node is suspected by any correct node.

Figure 22: Eventual Perfect Failure Detector Specification

```
EPFDC: Component Req_epfd Ind_epfd (Req_l, Ind_l) ≔
    ⟨State ≔
        ⟨alive: Nat → Set[ℕ],
         failed: Set[ℕ],
         r: Nat⟩

     init ≔ ⟨ℕ, [], 0⟩

     request ≔ λ n, s, ir.
         ⟨s, [], []⟩

     indication ≔ λ n′, s, ii.
         let ⟨a, f, r⟩ ≔ s in
         match ii with
         | (0, deliver_l(n, HB)) ⇒
               ⟨⟨a[r ↦ a(r) ∪ {n}], f, r⟩, [], []⟩
         end

     periodic ≔ λ n, s.
         let ⟨a, f, r⟩ ≔ s in
         let ⟨f′, ors, ois⟩ ≔ foldl(
             ℕ
             (λ⟨f′, ors, ois⟩, n.
                 if (n ∉ a(r) ∧ n ∉ f)
                     ⟨f′ ∪ {n},
                      ors ∪ {send_l(n, HB)}
                      ois ∪ {suspect(n)}⟩
                 else if (n ∈ a(r) ∧ n ∈ f)
                     ⟨f′ ∖ {n},
                      ors ∪ {send_l(n, HB)}
                      ois ∪ {restore(n)}⟩
                 else
                     ⟨f′,
                      ors ∪ {send_l(n, HB)}
                      ois⟩)
             ⟨f, [], []⟩ ) in
         ⟨⟨[], f′, r + 1⟩, ors, ois⟩⟩
```

Figure 23: Eventual Perfect Failure Detector Component

4.6 Eventual Leader Elector

Eventual Leader Elector
$\mathsf{Req}_{\mathsf{eld}} = \mathsf{Unit}$ None.
$\mathsf{Ind}_{\mathsf{eld}} \coloneqq \mathsf{trust}(n)$ It indicates that the node n is the leader.

ELE_1(Eventual Leadership)
Eventually every correct process trusts the same correct process.
$\exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{trust}(l) \wedge \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{trust}(l')))]$

Figure 24: Eventual Leader Elector Specification

```
ELEC: Component Req_eld Ind_eld (Req_epfd, Ind_epfd) ≔
    let epfd ≔ 0 in
    ⟨State ≔
        ⟨suspected: Set[ℕ],
         leader: ℕ⟩,

     init ≔ λn. ⟨∅, ⊥⟩,

     request  ≔ λ n, s, ir.
        ⟨s, [], []⟩

     indication ≔ λ n, s, ii.
        let ⟨p, l⟩ ≔ s in
        match ii with
        | (epfd, suspect(n')) ⇒
            let p' ≔ p ∪ {n'} in
            ⟨⟨p', l⟩, [], []⟩
        | (epfd, restore(n')) ⇒
            let p' ≔ p ∖ {n'} in
            ⟨⟨p', l⟩, [], []⟩
        end,

     periodic ≔ λ n, s.
        let ⟨p, l⟩ ≔ s in
        if l ≠ maxRank(ℕ ∖ p) in
            let l' = maxRank(ℕ ∖ p) in
            ⟨⟨p, l'⟩, [], []⟩
        else
            ⟨s, [], []⟩⟩
```

Figure 25: Eventual Leader Elector Component

4.7 Epoch Consensus

Epoch Consensus
$\mathsf{Req}_{\mathsf{ec}} \coloneqq \mathsf{propose}_{\mathsf{ec}}(v) \mid \mathsf{epoch}_{\mathsf{ec}}(n, ts)$
$\mathsf{propose}_{\mathsf{ec}}(v)$ proposes value v.
$\mathsf{epoch}_{\mathsf{ec}}(n, ts)$ starts a new epoch with the leader n and timestamp ts.
$\mathsf{Ind}_{\mathsf{ec}} \coloneqq \mathsf{decide}_{\mathsf{ec}}(v)$
$\mathsf{decide}_{\mathsf{ec}}(v)$ outputs the decided value v.

EC_1 (Validity)
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\exists n'.\ \Diamond\!\!\!-(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$
If a node decides the value v, then v was proposed by a node.

EC_2 (Uniform agreement)
$n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \wedge$
$\Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \Rightarrow$
$v = v'$
No two nodes decide differently.

EC_3 (Integrity)
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$
Every node decides at most once.

EC_4 (Termination)
$|\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \Rightarrow$
$(n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge \Box\neg(n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n', ts')) \Rightarrow$
$\forall n'.\ n' \in \mathsf{Correct} \rightarrow \Diamond\!\!\!-\Diamond \exists v'.\ (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$
If a correct node proposes and an epoch is started with that node as the leader, then every correct node eventually decides a value.

Figure 26: Epoch Consensus Specification

```
ECC: Component Req_ec Ind_ec (Req_beb, Ind_beb, Req_pl, Ind_pl) ≔
    let plc ≔ 0 in
    let bebc ≔ 1 in
    ⟨State ≔
        ⟨ets: Nat,
         valts: Nat, val: Nat,
         rts: Nat,
         prop: Nat,
         wval: Map[Nat, Nat]
         states: Map[ℕ, ⟨Nat, Nat⟩],
         accepted: Set[ℕ],
         decided: Bool ⟩

     init ≔ λn. ⟨−n, 0, ⊥, 0, ⊥, λn. ⊥, ∅, ∅, false⟩

     request ≔ λ n, s, ir.
        let ⟨ets, vts, val, rts, p, wv, st, ac, d⟩ ≔ s in
        match ir with
        | propose_ec(v) ⇒
            let p′ ≔ v in
            ⟨⟨ets, vts, val, rts, p′, wv, st, ac, d⟩, [], []⟩
        | epoch_ec(n_l, ts_l) ∧ n_l = n ⇒
            let ets′ ≔ ts_l in
            let st′ ≔ ∅ in
            let or ≔ (bebc, broadcast_beb(PREPARE (ets′))) in
            ⟨⟨ets′, vts, val, rts, p, wv, st′, ac, d⟩, [or], []⟩
        | _ ⇒
            ⟨s, [], []⟩
        end
```

Figure 27: Epoch Consensus (part 1)

```
indication := λ n′, s, ii.
    let ⟨ets, vts, val, rts, p, wv, st, ac, d⟩ := s in
    match ii with
    | (bebc, deliver_beb(n, m)) ⇒
        match m with
        | PREPARE(ets′) ∧ ets′ > rts ⇒
            let rts′ := ets′ in
            let or := (plc, send_pl(n_l, STATE(ets′, vts, val)) in
            ⟨⟨ets, vts, val, rts′, p, wv, st, ac, d⟩, [or], []⟩

        | ACCEPT(ets′, v) ∧ ets′ ≥ rts ⇒
            let ⟨vts′, val′⟩ := ⟨ets′, v⟩ in
            let or := (plc, send_pl(n, ACCEPTED(ets′))) in
            ⟨⟨ets, vts′, val′, rts, p, wv, st, ac, d⟩, [or], []⟩

        | DECIDED(ets′, v) ∧ ¬d ⇒
            let d′ := true in
            let oi := decide_ec(v)
            ⟨⟨ets, vts, val, rts, p, wv, st, ac, d′⟩, [], [oi]⟩

        | _ ⇒ ⟨s, [], []⟩
        end

    | (plc, deliver_pl(n, m)) ⇒
        match m with
        | STATE(ets′, ts, v) ∧ ets′ = ets ⇒
            let st′ := st[n ↦ ⟨ts, v⟩] in
            if (|dom(st′)| > |ℕ|/2 ∧ wv(ets) = ⊥ ∧ p ≠ ⊥)
                let st″ = ∅
                let ⟨vts′, val′⟩ := highest(st′) in
                let wv′ := wv[ets ↦ if (val′ ≠ ⊥) val′ else p] in
                let or := (bebc, broadcast_beb(ACCEPT(ets, wv′(ets)))) in
                ⟨⟨ets, vts′, val′, rts, p, wv′, st″, ac, d⟩, [or], []⟩
            else
                ⟨⟨ets, vts, val, rts, p, wv, st′, ac, d⟩, [], []⟩

        | ACCEPTED(ets′) ∧ ets′ = ets ⇒
            let ac′ := ac ∪ {n} in
            if (|ac′| > ℕ/2)
                let or := (bebc, broadcast_beb(DECIDED(ets, wv(ets)))) in
                ⟨⟨ets, vts, val, rts, p, wv, st, ac′, d⟩, [or], []⟩
            else
                ⟨⟨ets, vts, val, rts, p, wv, st, ac′, d⟩, [], []⟩

        | _ ⇒ ⟨s, [], []⟩
        end
    end

periodic := λ n, s.
    ⟨s, [], []⟩ )
```

Figure 28: Epoch Consensus (part 2)

4.8 Epoch Change

Epoch Change
$\mathsf{Ind}_{\mathsf{ech}} \coloneqq \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)$
The event $\mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)$ starts the epoch identified by timestamp ts with the leader n_l.

ECH_1 (Monotonicity)
$n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Box}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \rightarrow ts' > ts)$
If a correct process starts an epoch (ts, n_l) and later starts an epoch (ts', n_l'), then $ts' > ts$.

ECH_2 (Consistency)
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$(n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l') \Rightarrow n_l = n_l')$
If a correct process starts an epoch (ts, n_l) and another correct process starts an epoch (ts, n_l'), then $n_l = n_l'$.

ECH_3 (Eventual leadership)
$\exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))]]$
There is a timestamp ts and a correct process n_l such that eventually every correct process starts an epoch with ts and n_l and does not start another epoch afterwards.

Figure 29: Epoch Change Specification

```
ECHC: Component Ind_ech (Req_eld, Ind_eld, Req_beb, Ind_beb, Req_pl, Ind_pl) ≔
    let plc ≔ 0 in
    let bebc ≔ 1 in
    let eldc ≔ 2 in
    ⟨State ≔
        ⟨trusted: ℕ,
        lastts: Int,
        ts: Int,
        nk: Set[Bool] ⟩

     init ≔
        ⟨n_l0, 0, rank(n), ∅ ⟩

     request ≔ λ n, s, ir. ⟨s, [], []⟩

     indication ≔ λ n, s, ii.
        let ⟨tr, lts, t, nk⟩ ≔ s in
        match ii with
        | (eldc, trust_eld(n_l)) ⇒
            let tr′ ≔ n_l in
            let nk[tr′] ≔ false in
            if (tr′ ≔ n)
                let t′ ≔ t + N in
                let or ≔ (bebc, broadcast_beb(NewEpoch(t′))) in
                ⟨⟨tr′, lts, t′, nk⟩, [or], []⟩
            else
                ⟨⟨tr′, lts, t, nk⟩, [], []⟩
        | (bebc, deliver_beb(n_l, NewEpoch(newt))) ⇒
            if (tr = n_l ∧ nk[tr] = false)
                let nk[tr] ≔ true in
                let or ≔ (plc, send_pl(n_l, State(lts))) in
                ⟨⟨tr, lts′, t, nk⟩, [], [oi]⟩
            else if (tr = n_l ∧ newt > lts)
                let lts′ ≔ newt in
                let oi ≔ (ech, startEpoch(lts′, tr)) in
                ⟨⟨tr, lts, t, nk⟩, [or], []⟩
            else if (tr ≠ n_l)
                let or ≔ (plc, send_pl(n_l, Nack(lts))) in
                ⟨⟨tr, lts, t, nk⟩, [or], []⟩
            else
                ⟨s, [], []⟩
```

Figure 30: Epoch Change Component (part 1)

```
    | (plc, deliver_pl(n', STATE(lts'))) ∧ tr = n ⇒
        if (lts' > t)
            let t' := t + ((lts' − t)/N + 1) ∗ N in
            let or := (bebc, broadcast_beb(NEWEPOCH(t'))) in
            ⟨⟨tr, lts, t', nk⟩, [or], []⟩
        else
            let or := (bebc, broadcast_beb(NEWEPOCH(t))) in
            ⟨⟨tr, lts, t, nk⟩, [or], []⟩

    | (plc, deliver_pl(n', NACK(lts'))) ∧ tr = n ⇒
        let or := (bebc, broadcast_beb(NEWEPOCH(t))) in
        ⟨⟨tr, lts, t, nk⟩, [or], []⟩

    | _ ⇒
        ⟨s, [], []⟩
    end

periodic := λ n, s. ⟨s, [], []⟩ )
```

Figure 31: Epoch Change Component (part 2)

4.9 Uniform Consensus

Uniform Consensus
$\mathsf{Req}_{\mathsf{uc}} \coloneqq \mathsf{propose}_{\mathsf{uc}}(v)$
Propose value v for consensus.
$\mathsf{Ind}_{\mathsf{uc}} \coloneqq \mathsf{decide}_{\mathsf{uc}}(v)$
Outputs a decided value v of consensus.

UC_1 (Termination)
$|\mathsf{Correct}| > |\mathbb{N}|/2 \land n \in \mathsf{Correct} \rightarrow$
$[\forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ v \neq \bot \land \diamondsuit(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))] \Rightarrow$
$\exists v.\ \diamondsuit\diamondsuit(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v))$
Every correct node eventually decides some value.

UC_2 (Validity)
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \leftsquigarrow$
$\exists n'.\ (n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))$
If a node decides v, then v was proposed by some node.

UC_3 (Integrity)
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v'))$
No node decides twice.

UC_4 (Uniform agreement)
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \land \diamondsuit(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$v = v'$
No two nodes decide differently.

```
UCC: Component Req_uc Ind_uc (Req_ec, Ind_ec, Req_ech, Ind_ech) :=
    let echc := 0 in
    let ecc := 1 in
    ⟨State := ⟨prop : Nat,
               leader : ℕ,
               ts : Nat
               started : false⟩

     init  := λn.⟨⊥, ⊥, 0, false⟩

     request  := λ n, s, ir.
         let ⟨p, l, ts, str⟩ := s in
         match ir with
         | propose_uc(v) ⇒
             let p' = v in
             let or := (ecc, propose_ec(v)) in
             ⟨⟨p', l, ts, str⟩, [or], []⟩
         end

     indication := λ n', s, ii.
         let ⟨p, l, ts, str⟩ := s in
         match ii with
         | (echc, startEpoch_ech(ets, n_l)) ⇒
             let l' := n_l in
             let ts' := ets in
             let str' := false
             if (n' = n_l)
                 if (p ≠ ⊥)
                     let str' := true
                     let or := (ecc, epoch_ec(n_l, ets)) in
                     ⟨⟨p, l', st', str'⟩, [or], []⟩
                 else
                     ⟨⟨p, l', st', str'⟩, [], []⟩
             else
                 let or := (ecc, epoch_ec(n_l, ets)) in
                 ⟨⟨p, l', st', str'⟩, [or], []⟩

         | (ecc, decide_ec(v)) ⇒
             let oi := decide_uc(v) in
             ⟨s, [], [oi]⟩
         end

     periodic := λ n, s.
         let ⟨p, l, ts, str⟩ := s in
         if (l = n ∧ str = false ∧ p ≠ ⊥)
             let str' = true
             let or := (ecc, epoch_ec(n_l, ts)) in
              ⟨⟨p, l, ts, str'⟩, [or], []⟩
         else
              ⟨s, [], []⟩ ⟩
```

Figure 32: Uniform Consensus Component

5 Proofs

5.1 Soundness

Definition 9 (Transition)**.**
$\xrightarrow{\tau \cdot \tau'} \;=\; \xrightarrow{\tau}{}^{*}_{t} \xrightarrow{\tau'}_{p}$

Definition 10 (Initial World)**.**
$w_0(\mathcal{S}) = (\lambda d, n.\ \mathit{let}\ (c, \_) = \mathcal{S}(d)\ \mathit{in}\ \mathit{init}_c(n), \varnothing, \varnothing, r_0)$

The trace semantics $T(\mathcal{S})$ of $\mathcal{S}$ is the set of traces τ of infinite transitions $\xrightarrow{\tau}{}^{*}$ starting from the initial state $w_0(\mathcal{S})$ (with any r_{GST}). We consider infinite traces to reason about liveness properties.

Definition 11 (Traces of a stack)**.**
$T(\mathcal{S}) = \{\tau \mid w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*} \}$

Definition 12 (Models of a stack)**.**
$M(\mathcal{S}) = \{(\tau, 0, I_0) \mid \tau \in T(\mathcal{S})\}$

Definition 13. *Let*
$\mathit{mself}(d, o) \triangleq$
$\quad (d = [] \wedge o = \downarrow) \vee$
$\quad (d = [] \wedge o = \lightning) \vee$
$\quad (d = [i] \wedge o = \uparrow)$
$\mathit{mself}(\ell) \triangleq$
$\quad \mathit{mself}(\mathit{d}(\ell), \mathit{o}(\ell))$

Definition 14. *Let*
$\mathit{mcorrect}(\tau, n) =$
$\quad \forall j \geq 0.\ \exists k \geq j.$
$\quad \tau_k = (n, \_, \_, \_, \_, \_, \_, \_, \_)$

Theorem 2. (Soundness)
Let $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}'})$
If $\Gamma \vdash_c \mathcal{A}$, then $\Gamma \vDash_{\mathcal{S}} \mathcal{A}$.

Proof.

Assumption:

(1) $\Gamma \vdash_c \mathcal{A}$

Conclusion:

$\Gamma \vDash_{\mathcal{S}} \mathcal{A}$

From Definition 5, the conclusion is

$\forall m \in M(\mathcal{S}), m \vDash \Gamma \to m \vDash \mathcal{A}$

Assumption:

(2) $m \in M(\mathcal{S})$

(3) $m \vDash \Gamma$

Conclusion:

(4) $m \vDash \mathcal{A}$

From Definition 12 and Definition 11 on [2], we have

(5) $m = (\tau, 0, I_0)$

(6) $w_0(\mathcal{S}) \xrightarrow{\tau}{}^*$

Induction on the derivation of [1]:

Case SEQ:

(7) $\mathcal{A} \;=\; \mathsf{n} \neq n \;\Rightarrow\; \mathsf{s}'(n) = \mathsf{s}(n)$

From Definition 6 on [4], [5] and [7], we need to show that for all $j \geq 0$:

Let $\tau(j) = (n_j, r_j, d_j, o_j, e_j, \sigma_j, \sigma'_j, ors_j, ois_j)$,

$d'_j = \begin{cases} d & \text{if } o = \downarrow \vee o = \lightning \\ \mathsf{tail}(d_j) & \text{else} \end{cases}$

$n_j \neq n \to \sigma'(d'_j)(n) = \sigma(d'_j)(n)$

Immediate from Lemma 4.

Case IR:

(8) $\mathcal{A} \;=\; \forall e.\ \top \downarrow e \;\Rightarrow\; (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}(\mathsf{n}, \mathsf{s}(\mathsf{n}), e)$

From Definition 6 on [4], [5] and [8], we need to show that for all $j \geq 0$:

Let $\tau(j) = (n_j, r_j, d_j, o_j, e_j, \sigma_j, \sigma'_j, ors_j, ois_j)$,

$\forall e.\ d_j = [\,] \wedge o_j = \downarrow \wedge e_j = e \to$

$(\sigma'(d_j)(n_j), ois_j, ors_j) = \mathsf{request}(n_j, \sigma(d_j)(n_j), e)$

Induction on the derivation of [6]:

Cases except REQ: By induction hypothesis.

Case REQ: Immediate from the assumption of REQ.

$\sigma(d) = s$

$\mathsf{request}(c, n, s(n), e) = (s'_n, (i, e_1), e_2)$

$s' = s[n \mapsto s'_n]$

$\sigma' = \sigma[d \mapsto s']$

Case II:

Similar to the case IR.

Case PE:

Similar to the case IR.

Case OI:

(9) $\mathcal{A} = \forall n, e.\ \ n \bullet e \in \mathsf{ois} \wedge \mathsf{self} \ \Rightarrow\ \hat{\Diamond}(n \bullet \top \uparrow e)$

From Definition 6 on [4], [5] and [9], we need to show that:

$\forall j \geq 0. \forall n, d, ois.$

$\tau(j) = (n, \_, d, o, \_, \_, \_, \_, ois) \wedge e \in ois \wedge \mathsf{mself}(d, o) \rightarrow$

$\exists k > j.\ \ \tau_k = (n, \_, [\,], e, \uparrow, \_, \_, \_, \_)$

Case analysis on the disjuncts of Definition 13:

Case 1:

(10) $d = [\,] \wedge o = \downarrow$

We assume that

(11) $\tau(j) = (n, \_, [\,], \downarrow, \_, \_, \_, \_, ois)$

(12) $e \in ois$

We show that

$\exists k > j.\ \ \tau_k = (n, \_, [\,], \uparrow, e, \_, \_, \_, \_)$

Induction on the derivation of [6]:

Case Req: From the assumption of Req:

$\mathsf{request}(n, s(n), e') = (\_, \_, e)$

$\tau_2 = (n, \_, [\,], \uparrow, e, \_, \_, \_, \_) \ \cdot\ \tau_2'$

$\tau = (n, \_, [\,], \downarrow, e', \_, \_, \_, e) \ \cdot\ \_ \ \cdot\ \tau_2$

Thus

$\tau = (n, \_, [\,], \downarrow, e', \_, \_, \_, e) \ \cdot\ \_ \ \cdot\ (n, \_, [\,], \uparrow, e, \_, \_, \_, \_) \ \cdot\ \_$

After the event $(n, \_, [\,], \downarrow, e', \_, \_, \_, e)$,

the event $(n, \_, [\,], \uparrow, e, \_, \_, \_, \_)$ comes.

Also, the induction hypothesis is used for the

rest of events in τ such as the events in τ_2'.

Cases except Req:

By induction hypothesis.

Case 2:

(13) $d = [\,] \wedge o = \wr$

Similar to case Req.

Case 3:

(14) $d = [i] \wedge o = \uparrow$

Similar to Case 1.

Induction on the derivation of [6].

Rule Ind is main case.

Case OI′:

(15) $\mathcal{A} \ \ = \ \ \forall n, e.\ \ n \bullet \top \uparrow e \ \Rightarrow\ \hat{\boxminus}(n \bullet e \in \mathsf{ois} \wedge \mathsf{self})$

From Definition 6 on [4], [5] and [15], we need to show that:

$\forall j \geq 0.$ Let $\tau(j) = (n_j, \_, d_j, o_j, e_j, \_, \_, \_, \_)$,

$\forall n, e.\ \ n_j = n \wedge d_j = [\,] \wedge o_j = \uparrow \wedge e_j = e \rightarrow$

$\exists k < j.$ Let $\tau_k = (n_k, \_, d_k, o_k, \_, \_, \_, ois_k)$,

$n_k = n \wedge e \in ois_k \wedge \mathsf{mself}(d_k, o_k)$

Induction on the derivation of [6]:

Cases except Req, Ind and Per: By induction hypothesis.

Case Req: From the assumption of Req:

$\mathsf{request}(c, n, s(n), e') = (\_, \_, e)$

$\tau_2 = (n, \_[], \uparrow, e, \_, \_, \_, \_) \cdot \tau_2'$
$\tau = (n, \_, [], \downarrow, e', \_, \_, \_, e) \cdot \tau_1 \cdot \tau_2$
Before the event $(n, \_, [], \uparrow, e, \_, \_, \_, \_)$,
there is the event $(n, \_, [], \downarrow, e', \_, \_, \_, e)$.
$d = [] \wedge o = \downarrow$ satisfies the first disjunct of mself.
Also, the induction hypothesis is used for the
rest of events in τ such as the events in τ_2'.

Case IND:
Similar to Case REQ.
The second disjunct of mself is satisfied.

Case PER:
Similar to Case REQ.
The first disjunct of mself is satisfied.

Case OR:
Similar to the Case OI.

Case OR′:
Similar to the Case OI′.

Case INIT:
(16) $\mathcal{A} = \textcircled{S}\ (\mathsf{s} = \lambda n.\ \mathsf{init}_c(n))$
By [4], [5], and rule SELFM, we have
$\tau_1' = \epsilon$
$\tau_2' = \tau_{0..}|_{\mathsf{mself}}$
$\tau' = \tau_1' \cdot \tau_2'$
$i' = 0$
We need to show that
$(\tau_{0..}|_{\mathsf{mself}}, 0, I_0) \vDash (\mathsf{s} = (\lambda n.\ \mathsf{init}_c(n)))$
By [6]
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^*$
By Definition 10
$w_0(\mathcal{S}) = (\lambda d, n.\ \mathsf{let}\ (c, \_) = \mathcal{S}(d)\ \mathsf{in}\ \mathsf{init}_c(n), \varnothing, \varnothing, r_0)$
Thus, the state is equal to the user-defined init at the beginning.
$\sigma(\tau_0')([])(n) = \mathsf{init}_c(n)$
By Definition 9, $\rightarrow_t$ appears at the beginning of $\rightarrow$
$\tau = \tau' \cdot \tau'' \cdot \tau'''$
$w_0(\mathcal{S}) \xrightarrow{\tau'}{}_t^* \xrightarrow{\tau''}{}_p W' \xrightarrow{\tau'''}$
There are two $\rightarrow_t$ transitions:
The rule FAIL: It preserves the state σ.
The rule REQUEST: It is a self transition.
Thus, the init state is preserved to the first self step,
that is either by the rule REQUEST or the rule PERIODIC.

Case ASELF:
(17) $\mathcal{A} = \textcircled{S}\ \Box\mathsf{self}$
From Definition 6 on [4], [5] and [17], we need to show that:

$\forall j \geq 0$. Let $(\tau|_{\mathsf{mself}})_j = (\_, \_, d_j, o_j, \_, \_, \_, \_, \_, \_)$,
$\mathsf{mself}(d_j, o_j)$

That is

$\forall j \geq 0$. Let $\tau(j) = (\_, \_, d_k, o_k, \_, \_, \_, \_, \_, \_)$,
$\mathsf{mself}(d_j, o_j) \rightarrow \mathsf{mself}(d_j, o_j)$

That is trivial.

Case SInv:

(18) $\mathcal{A} \;=\; \textcircled{S}\, \mathcal{I} \;\leftrightarrow\; \mathsf{restrict}(\mathsf{self}, \mathcal{I})$

Let $\mathcal{I} = \Box\mathcal{A}$
where $\circ$ and $\textcircled{S}$ are not used in $\mathcal{A}$.

Forward Direction:

We have

(19) $m \vDash \textcircled{S}\, \Box\mathcal{A}$

We show that

(20) $m \vDash \mathsf{restrict}(\mathsf{self}, \Box\mathcal{A})$

Thus, from [5], we need to show that

(21) $(\tau, 0, I_0) \vDash \mathsf{restrict}(\mathsf{self}, \Box\mathcal{A})$

Thus, from Definition 3 on [21], we need to show that

(22) $(\tau, 0, I_0) \vDash \Box(\mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A}))$

Thus, from Definition 6, we need to show that:

$(\tau, k, I_0) \vDash \mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A})$ forall $k \geq 0$

From Definition 6 on [19] and [5], we have

(23) $\tau' = \tau|_{\mathsf{mself}}$

(24) $(\tau', 0, I_0) \vDash \Box\mathcal{A}$

Let

(25) j be the location of the first mself event in τ

that is

(26) $\tau'_0 = \tau(j)$

We consider two cases:

Case:

(27) $k < j$

From [25] and [27],

(28) $\neg\mathsf{mself}(\tau_k)$

Thus

$(\tau, k, I_0) \vDash \mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A})$

Case:

(29) $k \geq j$

From Lemma 14 on [23], [26], [24],

(30) $(\tau, j, I_0) \vDash \mathsf{restrict}(\mathsf{self}, \Box\mathcal{A})$

From Definition 3 on [30],

(31) $(\tau, j, I_0) \vDash \Box(\mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A}))$

From Definition 6 on [31], we have

$(\tau, k, I_0) \vDash \mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A})$ forall $k \geq j$

Backward Direction:

We have

(32) $m \vDash \mathsf{restrict}(\mathsf{self}, \Box\mathcal{A})$

We show that

$m \vDash \textcircled{S}\ \Box\mathcal{A}$

Let

(33) $\tau' = \tau|_{\mathsf{mself}}$

From [5] and Definition 6, we need to show that

$(\tau', k, I_0) \vDash \mathcal{A}$ forall $k \geq 0$

From [32] and [5], we have

(34) $(\tau, 0, I_0) \vDash \mathsf{restrict}(\mathsf{self}, \Box\mathcal{A})$

Thus, from Definition 3 on [34], we have

(35) $(\tau, 0, I_0) \vDash \Box(\mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A}))$

Thus, from Definition 6, we have

(36) $(\tau, j, I_0) \vDash \mathsf{self} \rightarrow \mathsf{restrict}(\mathsf{self}, \mathcal{A})$ forall $j \geq 0$

From [33], forall $k \geq 0$, there exists j such that

(37) $\tau'_k = \tau(j)$

(38) $\mathsf{mself}(\tau(j))$

From [36] and [38],

(39) $(\tau, j, I_0) \vDash \mathsf{restrict}(\mathsf{self}, \mathcal{A})$

From Lemma 14 on [33], [37], [39],

$(\tau', k, I_0) \vDash \mathcal{A}$

Case PostPre:

(40) $\mathcal{A} = \textcircled{S}\ (\mathsf{s}' = s \Leftrightarrow \bigcirc\, \mathsf{s} = s)$

From Definition 6 on [4], [5] and [40], we need to show that:

$\forall j \geq 0.\ \ \sigma'((\tau|_{\mathsf{mself}})_j)([]) = \sigma((\tau|_{\mathsf{mself}})_{j+1})([])$

That is straightforward from Lemma 5 and Lemma 6.

Case RSeq:

Immediate from the periodic transition $\rightarrow_p$ that increments the round r.

Case ROrd:

Immediate from the periodic transition $\rightarrow_p$ that increments the round r for the transition $\rightarrow_{per}$ for the periodic events per.

Case GST:

Every message that is sent in round r stays in ms until the periodic transition $\rightarrow_p$ of round r (Lemma 7).

After the GST round r_{GST}, the periodic transition $\rightarrow_p$ does not drop messages.

The periodic transition $\rightarrow_p$ of round r includes the transition $\rightarrow_{msg}$ that delivers the messages.

Case FDup:

(41) $\mathcal{A} =$

$\Box\Diamond(n' \bullet d \uparrow \mathsf{deliver}_{\mathsf{I}}(n, m)) \rightarrow$

$\Box\Diamond(n \bullet d \downarrow \mathsf{send_l}(n', m))$

We prove the contra-positive that is

(42) $\mathcal{A}$ =

$\Diamond\Box\neg(n \bullet d \downarrow \mathsf{send_l}(n', m)) \rightarrow$
$\Diamond\Box\neg(n' \bullet d \uparrow \mathsf{deliver_l}(n, m))$

From Definition 6 on [4], [5] and [42], we need to show that:

$(\exists j.\ \forall i \geq j.$
$\neg(\mathsf{n}(\tau(i)) = n \wedge \mathsf{d}(\tau(i)) = d \wedge$
$\mathsf{o}(\tau(i)) = \downarrow \wedge \mathsf{e}(\tau(i)) = \mathsf{send_l}(n', m)))$
$\rightarrow$
$(\exists j.\ \forall i \geq j.$
$\neg(\mathsf{n}(\tau(i)) = n' \wedge \mathsf{d}(\tau(i)) = d \wedge$
$\mathsf{o}(\tau(i)) = \uparrow \wedge \mathsf{e}(\tau(i)) = \mathsf{deliver_l}(n, m)))$

Immediate by Lemma 24.

Case NForge:

(43) $\mathcal{A}$ =

$(n' \bullet d \uparrow \mathsf{deliver_l}(n, m)) \Rightarrow$
$\Diamond\!\!\!-(n \bullet d \downarrow \mathsf{send_l}(n', m))$

From Definition 6 on [4], [5] and [43], we need to show that:

$\forall i.$
$(\mathsf{n}(\tau(i)) = n' \wedge \mathsf{d}(\tau(i)) = d \wedge$
$\mathsf{o}(\tau(i)) = \uparrow \wedge \mathsf{e}(\tau(i)) = \mathsf{deliver_l}(n, m))) \rightarrow$
$\rightarrow$
$\exists j \leq i.$
$(\mathsf{n}(\tau(j)) = n \wedge \mathsf{d}(\tau(j)) = d \wedge$
$\mathsf{o}(\tau(j)) = \downarrow \wedge \mathsf{e}(\tau(j)) = \mathsf{send_l}(n', m)$

Immediate by Lemma 27.

Case UniOI:

(44) $\mathcal{A}$ =

$(\mathsf{occ}(\mathsf{ois}, e) \leq 1 \wedge$
$\hat{\Box}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow e \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow e \notin \mathsf{ois})) \Rightarrow$
$(n \bullet \top \uparrow e) \Rightarrow$
$\hat{\Box}\neg(n \bullet \top \uparrow e) \wedge \hat{\boxminus}\neg(n \bullet \top \uparrow e)$

From Definition 6 on [4], [5] and [44], we need to show that:

$\forall \mathcal{S}, \tau, i, j, k, oi, n.$
$\mathsf{occ}(\mathsf{ois}(\tau(i)), e) \leq 1 \wedge$
$\forall k < i.\ (\mathsf{n}(\tau_k) = n \wedge \mathsf{mself}(\tau_k)) \rightarrow e \notin \mathsf{ois}(\tau_k) \wedge$
$\forall k > i.\ (\mathsf{n}(\tau_k) = n \wedge \mathsf{mself}(\tau_k)) \rightarrow e \notin \mathsf{ois}(\tau_k) \wedge$
$\mathsf{n}(\tau(j)) = n \wedge$
$\mathsf{d}(\tau(j)) = [] \wedge$
$\mathsf{o}(\tau(j)) = \uparrow \wedge$
$\mathsf{e}(\tau(j)) = e \ \rightarrow$
$k < j \ \rightarrow \neg[\mathsf{n}(\tau_k) = n \wedge \mathsf{d}(\tau_k) = [] \wedge \mathsf{o}(\tau_k) = \uparrow \wedge \mathsf{e}(\tau_k) = e]$
$k > j \ \rightarrow \neg[\mathsf{n}(\tau_k) = n \wedge \mathsf{d}(\tau_k) = [] \wedge \mathsf{o}(\tau_k) = \uparrow \wedge \mathsf{e}(\tau_k) = e]$

That is immediate from Lemma 32.

Case UniOR:
Similar to rule UniOI

Case APer:
(45) $\mathcal{A}$ =
$n \in \mathsf{Correct} \rightarrow \Box\Diamond(n \bullet \top \wr \mathsf{per})$
From Definition 6 on [4], [5] and [45], we need to show that:
$\forall n.\ n \in \mathbb{N} \wedge$
$\forall i.\ \neg(\mathsf{n}(\tau(i)) = n \wedge \mathsf{d}(\tau(i)) = [\,] \wedge \mathsf{e}(\tau(i)) = \mathsf{fail}) \rightarrow$
$\forall j.\ \exists k \geq j.$
$\mathsf{n}(\tau_k) = n \wedge \mathsf{d}(\tau_k) = [\,] \wedge \mathsf{o}(\tau_k) = \wr \wedge\ \mathsf{e}(\tau_k) = \mathsf{per}$

This is proved using Lemma 13 on [6].

Case Node:
(46) $\mathcal{A}$ =
$\Box\ \mathsf{n} \in \mathbb{N}$
From Definition 6 on [4], [5] and [46], we need to show that:
$\forall i.\ \mathsf{n}(\tau(i)) \in \mathbb{N}$
This is proved using Lemma 33.

Corollary 2.
For all Γ, $\mathcal{A}$, $\mathcal{S}$, c and $\overline{\mathcal{S}'}$ such that $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}'})$, if $\vDash_{\mathcal{S}} \Gamma$ and $\Gamma \vdash_c \mathcal{A}$, then $\vDash_{\mathcal{S}} \mathcal{A}$.

Proof.
Assumption:
(1) $\vDash_{\mathcal{S}} \Gamma$
(2) $\Gamma \vdash_c \mathcal{A}$
Conclusion:
$\vDash_{\mathcal{S}} \mathcal{A}$

From Definition 4 on [1]
(3) $\forall m \in M(\mathcal{S}).\ m \vDash \mathcal{A}$
From Theorem 2 on [2], we have
(4) $\Gamma \vDash_{\mathcal{S}} \mathcal{A}$
From Definition 5 on [4], we have
(5) $\forall m \in M(\mathcal{S}).\ m \vDash \Gamma \rightarrow m \vDash \mathcal{A}$
From [3] and [5], we have
(6) $\forall m \in M(\mathcal{S}).\ m \vDash \mathcal{A}$
From Definition 4 on [6], we have
$\vDash_{\mathcal{S}} \mathcal{A}$

Lemma 4.
Only steps in a node change the state for that node.

$\forall W_1, W_2, \tau, n, n', d, o, \sigma, \sigma'.$
$W_1 \xrightarrow{\tau}{}^* W_2 \wedge$
Let $(n, \_, d, o, \_, \sigma, \sigma', \_, \_) \in \tau$
$d' = \begin{cases} d & \text{if } o = \downarrow \vee o = \wr \\ \mathsf{tail}(d) & \text{else} \end{cases}$
$n' \neq n \rightarrow \sigma'(d')(n') = \sigma(d')(n')$

Proof.
Immediate from induction on steps.

Lemma 5.
The post-state of every event is the same as the pre-state of its next event.
$\forall W_1, W_2, \tau, j.$
$W_1 \xrightarrow{\tau}{}^* W_2$
$\rightarrow$
$\sigma'(\tau(j)) = \sigma(\tau_(j + 1)$

Proof.
Immediate from induction on steps.

Lemma 6.
Only the self events change the top-level state.
$\forall W_1, W_2, \tau, d, o, \sigma, \sigma'.$
$W_1 \xrightarrow{\tau}{}^* W_2 \wedge$
Let $(\_, \_, d, o, \_, \sigma, \sigma', \_, \_) \in \tau$
$\neg \mathsf{mself}(d, o) \rightarrow$
$\sigma'([]) = \sigma([])$

Proof.
Immediate from induction on steps.

Lemma 7.
The transitions $\xrightarrow{\tau}_t$, $\xrightarrow{\tau}_{req}$, $\xrightarrow{\tau}_{ind}$, *and* $\xrightarrow{\tau}_{per}$ *preserve messages.*
$\forall \sigma, \sigma', ms, ms', f, f', r, r', \tau.$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_t (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{req} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{ind} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{per} (\sigma', ms')$
$\rightarrow$
$ms \subseteq ms'$

Proof.
Immediate from induction on steps.

Lemma 8.
If a node is failed, it remains failed.
$\forall \sigma, \sigma', ms, ms', f, f', r, r', \tau, n.$
$[(\sigma, ms, f, r) \xrightarrow{\tau}{}^{*} (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^{*}_{t} (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{p} (\sigma', ms', f', r')] \wedge$
$n \in f$
$\rightarrow$
$n \in f'$

Proof.
Immediate from Definition 9 and induction on steps.

Lemma 9.
If a node is failed, it does not take any step.
$\forall \sigma, \sigma', ms, ms', f, f', r, r', \tau, n.$
$[(\sigma, ms, f, r) \xrightarrow{\tau}{}^{*} (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^{*}_{t} (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{p} (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{req} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{ind} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{per} (\sigma', ms')] \wedge$
$n \notin \mathbb{N} \vee n \in f$
$\rightarrow$
$\nexists j.\ \ \mathit{n}(\tau(j)) = n \wedge$

Proof.
Immediate from Definition 9, induction on steps and Lemma 8.

Lemma 10.
If a node is not failed, it takes a periodic step on a p step.
$\forall \sigma, \sigma', ms, ms', f, f', r, r', \tau, n.$
$(\sigma, ms, f, r) \xrightarrow{\tau}_{p} (\sigma', ms', f', r')$
$n \in \mathbb{N} \setminus f$
$\rightarrow$
$\exists i.\ \mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = [] \wedge \mathit{o}(\tau(i)) = \{\} \wedge\ \mathit{e}(\tau(i)) = \mathit{per}$

Proof.
Immediate from the definition of the rules rule Periodic and rule Per.

Lemma 11.
If a node takes steps, it is not failed.
$\forall \sigma, ms, f, r, \tau, n.$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^{*} \wedge$
$\mathit{mcorrect}(\tau, n)$
$\rightarrow$
$n \in \mathbb{N} \setminus f$

Proof.
Contra-positive is proved using induction on the steps and using Lemma 9 and Lemma 8.

Lemma 12.
If a node is in the failed set, it has failed before.
$\forall \sigma, ms, f, r, \tau, n.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*} (\sigma, ms, f, r) \wedge$
$n \in f$
$\rightarrow$
$\exists i.\ \mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = [] \wedge \mathit{e}(\tau(i)) = \mathit{fail}$

Proof.
By induction on the steps.

Lemma 13.
For all $\mathcal{S}$ and τ,
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*} \wedge n \in \mathbb{N}$
$\forall i.\ \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = [] \wedge \mathit{e}(\tau(i)) = \mathit{fail}) \rightarrow$
$\forall j.\ \exists k \geq j.$
$\quad \mathit{n}(\tau_k) = n \wedge \mathit{d}(\tau_k) = [] \wedge \mathit{o}(\tau_k) = \wr \wedge \ \mathit{e}(\tau_k) = \mathit{per}$

Proof.
We assume
(1) $w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*}$
(2) $n \in \mathbb{N}$
We prove the contra-positive that is
(3) $\exists j.\ \nexists k \geq j.$
$\quad \mathsf{n}(\tau_k) = n \wedge \mathsf{d}(\tau_k) = [] \wedge \mathsf{o}(\tau_k) = \wr \wedge \ \mathsf{e}(\tau_k) = \mathsf{per} \rightarrow$
$\quad \exists i.\ \mathsf{n}(\tau(i)) = n \wedge \mathsf{d}(\tau(i)) = [] \wedge \mathsf{e}(\tau(i)) = \mathsf{fail}$
We assume
(4) $\exists j.\ \nexists k \geq j.$
$\quad \mathsf{n}(\tau_k) = n \wedge \mathsf{d}(\tau_k) = [] \wedge \mathsf{o}(\tau_k) = \wr \wedge \ \mathsf{e}(\tau_k) = \mathsf{per} \rightarrow$
By Definition 9 on [1]
(5) $\tau = \overline{\tau_i' \cdot \tau_i}$

(6) $W_1 = w_0(\mathcal{S})$

(7) $W_i \xrightarrow{\tau_i'}{}^*_t\ W_i' \xrightarrow{\tau_i}_p\ W_{i+1}$

From [4], [5],

(8) $\exists i.\ \forall \ell \in \tau_i.$
$\neg(\mathsf{n}(\ell) = n \wedge \mathsf{d}(\ell) = [] \wedge \mathsf{o}(\ell) = \wr \wedge\ \mathsf{e}(\ell) = \mathsf{per})$

By the contra-positive of Lemma 10 on [7], [2] and [8]

(9) $n \in f(W_i')$

By Lemma 12 on [7], [5], and [9]

$\exists i.\ \mathsf{n}(\tau(i)) = n \wedge \mathsf{d}(\tau(i)) = [] \wedge \mathsf{e}(\tau(i)) = \mathsf{fail}$

Lemma 14.
$\forall \tau, \tau', i, j.$
$\tau' = \tau|_{\mathit{mself}} \wedge$
$\tau'(i) = \tau(j)\ \rightarrow$
$(\tau', i, I) \vDash \mathcal{A}\ \leftrightarrow\ (\tau, j, I) \vDash \mathit{restrict}(\mathit{self}, \mathcal{A})$
where $\circ$ *and* Ⓢ *are not used in* $\mathcal{A}$.

Proof.
Similar to Lemma 36.

Lemma 15.
A message is not added to the message set unless it is sent.
$\forall \sigma, \sigma', ms, ms', f, f', r, r', \tau, n, n', d, m.$
$(n, n', d, m) \notin ms \wedge$
$[(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_t (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{req} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{ind} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{per} (\sigma', ms')] \wedge$
$(\forall i,\ \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin ms'$

Proof.
Immediate from induction on steps.

Lemma 16.
$\forall W_1, W_2, n, n', d, m.$
$W_1 \xrightarrow{\tau}_p\ W_2 \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin \mathit{ms}(W_2)$

Proof.

By the definition of the transition $\rightarrow_p$ and Lemma 15.

Lemma 17.
$\forall W_1, W_2, \tau, \tau_1, \tau_2, n, n', d, m.$
$W_1 \xrightarrow{\tau_1}{}^*_t \xrightarrow{\tau_2}_p W_2 \wedge$
$\tau = \tau_1 \cdot \tau_2 \wedge$
$(n, n', d, m) \notin \mathit{ms}(W_1) \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin \mathit{ms}(W_2)$

Proof.
By Lemma 15 and Lemma 16.

Lemma 18.
$\forall \sigma, \sigma', ms, ms', f, r, r', \tau, n, n', d, m.$
$(n, n', d, m) \notin ms \wedge$
$[(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_t (\sigma', ms', f', r') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{req} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{ind} (\sigma', ms') \vee$
$(\sigma, ms, f, r) \xrightarrow{\tau}{}^*_{per} (\sigma', ms')] \wedge$
$\rightarrow$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
Immediate from induction on steps.

Lemma 19.
$\forall W_1, W_2, n, n', d, m.$
$W_1 \xrightarrow{\tau}_p W_2 \wedge$
$(n, n', d, m) \notin \mathit{ms}(W_1) \wedge$
$\rightarrow$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By the definition of the transition $\rightarrow_p$ and Lemma 18.

Lemma 20.
$\forall W_1, W_2, \tau, \tau_1, \tau_2, n, n', d, m.$
$W_1 \xrightarrow{\tau_1}{}^*_t \xrightarrow{\tau_2}_p W_2 \wedge$
$\tau = \tau_1 \cdot \tau_2 \wedge$

$(n, n', d, m) \notin \mathit{ms}(W_1)$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By Lemma 18 and then Lemma 15 and Lemma 19.

Lemma 21.
$\forall W_1, W_2, \tau, \tau_1, \tau_2, n, n', d, m.$
$W_1 \xrightarrow{\tau_1}{}^*_t \xrightarrow{\tau_2}_p W_2 \wedge$
$\tau = \tau_1 \cdot \tau_2 \wedge$
$(n, n', d, m) \notin \mathit{ms}(W_1) \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin \mathit{ms}(W_2) \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By Lemma 17 and Lemma 20.

Lemma 22.
$\forall W_1, \tau, n, n', d, m.$
$W_1 \xrightarrow{\tau}{}^* \wedge$
$(n, n', d, m) \notin \mathit{ms}(W_1) \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By Definition 9, induction on the steps and Lemma 21.

Lemma 23.
$\forall W_1, W_2, \tau, \tau_1, \tau_2, n, n', d, m.$
$W_1 \xrightarrow{\tau_1}_p W_2 \xrightarrow{\tau_2}{}^* \wedge$
$\tau = \tau_1 \cdot \tau_2 \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.

By Lemma 16 and Lemma 22.

Lemma 24.
$\forall W, \tau, n, n', d, m.$
$W \xrightarrow{\tau}{}^{*} \quad \wedge$
$(\exists j.\ \forall i \geq j.$
$\quad \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(\exists j.\ \forall i \geq j.$
$\quad \neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By Definition 9 and Lemma 23.

Lemma 25.
$\forall \mathcal{S}, W, \tau, n, n', d, m, i.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*} W \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin \mathit{ms}(W)$

Proof.
By Definition 9, induction on the steps and Lemma 21.

Lemma 26.
$\forall \mathcal{S}, W, W', \tau_1, \tau_2, \tau, n, n', d, m, i.$
$w_0(\mathcal{S}) \xrightarrow{\tau_1}{}^{*} W \xrightarrow{\tau_2}{}_t^{*} W' \wedge$
$\tau = \tau_1 \cdot \tau_2 \wedge$
$(\forall i. \neg(\mathit{n}(\tau(i)) = n \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \downarrow \wedge \mathit{e}(\tau(i)) = \mathit{send}_l(n', m)))$
$\rightarrow$
$(n, n', d, m) \notin \mathit{ms}(W')$

Proof.
By Lemma 25 and Lemma 15.

Lemma 27.
$\forall \mathcal{S}, \tau, n, n', d, m, i.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^{*} \quad \wedge$
$(\forall j \leq i.$
$\quad \neg(\mathit{n}(\tau(j)) = n \wedge \mathit{d}(\tau(j)) = d \wedge \mathit{o}(\tau(j)) = \downarrow \wedge \mathit{e}(\tau(j)) = \mathit{send}_l(n', m))$
$\rightarrow$
$\neg(\mathit{n}(\tau(i)) = n' \wedge \mathit{d}(\tau(i)) = d \wedge \mathit{o}(\tau(i)) = \uparrow \wedge \mathit{e}(\tau(i)) = \mathit{deliver}_l(n, m)))$

Proof.
By Definition 9, we consider two cases:
Case 1: The index i is in a $\rightarrow_t$ transition.

(1) $w_0(\mathcal{S}) \xrightarrow{\tau_1}{}^* W' \xrightarrow{\tau_2}{}_t^* W''$
(2) $\tau = \tau_1 \cdot \tau_2$
(3) $|\tau_1| \leq i < |\tau_1| + |\tau_2|$

Immediate from Lemma 18.
Case 2: The index i is in a $\rightarrow_p$ transition.

(4) $w_0(\mathcal{S}) \xrightarrow{\tau_1}{}^* W' \xrightarrow{\tau_2}{}_t^* W'' \xrightarrow{\tau_3}{}_p W'''$
(5) $\tau = \tau_1 \cdot \tau_2$
(6) $|\tau_1| + |\tau_2| \leq i < |\tau_1| + |\tau_2| + |\tau_3|$

Immediate from Lemma 26 and Lemma 19.

Definition 15.
We instrument the transition system with event ID (ei), parent ID (pi) and child index (ci).
The event ID ei uniquely identifies events in a trace.
The parent ID pi is the ID of the event that issued the event.
The child index ci for an event is the index of that event in the list of events that its parent event issued. For example, if a parent event issues the list of request events $[e_1, e_2, e_3]$*, then the child index of* e_2 *is* 1*. Similarly, if a parent event issues the list of indication events* $[e_3, e_4, e_5]$*, then the child index of* e_4 *is* 1*.*
To create unique IDs, a counter ei is added to the state of the transition system and the counter is weaved through the rules.
As an example, the rule Req *is updated as follows:*
The state of the transition system includes the counter ei:
(s, ms, f, ei)
The events include the event ID ei, the parent ID pi and the child index ci:
$(ei, pi, ci, \_, \_, \_, \_, \_, \_, \_, \_)$
We use the functions ei*,* pi *and* ci *to extract the corresponding components of an event.*
As the counter in the pre-state is ei, the event id of the main event is $ei + 1$ *and the updated counter is passed in the pre-state to the next transition for the issued request event.*
The parent Id of the issued indication and request events is $ei + 1$*.*
Let us recall that in the interest of simplicity, the rules show only one issued request event and one issued indication event.
The child index of the issued indication and request events is 0*.*

Req

$$\frac{\begin{array}{c} n \in \mathbb{N} \setminus f \qquad \mathcal{S}(d) = (c, \_) \qquad \sigma(d) = s \\ \mathit{request}_c(n, s(n), e) = (s'_n, [(i, e_1)], [e_2]) \\ s' = s[n \mapsto s'_n] \qquad \sigma' = \sigma[d \mapsto s'] \\ (\sigma', ms, f, r, ei + 1) \xrightarrow{\tau_1}_{req} (\sigma_1, ms_1, ei') \qquad \tau_1 = (\_, ei + 1, 0, n, r, i :: d, \downarrow, e_1, \_, \_, \_, \_) \cdot \tau'_1 \\ (\sigma_1, ms_1, f, r, ei') \xrightarrow{\tau_2}_{ind} (\sigma_2, ms_2, ei'') \qquad \tau_2 = (\_, ei + 1, 0, n, r, d, \uparrow, e_2, \_, \_, \_, \_) \cdot \tau'_2 \\ \tau = (ei + 1, \_, \_, n, d, \downarrow, e, \sigma, \sigma', (i, e_1), e_2) \cdot \tau_1 \cdot \tau_2 \end{array}}{(\sigma, ms, f, r, ei) \xrightarrow{\tau}_{req} (\sigma_2, ms_2, ei'')}$$

It is easy to show that there is a bi-simulation between the original and the instrumented transition system.

Lemma 28.
Based on Definition 15, the event ID uniquely identifies events.
$\forall \mathcal{S}, \tau, i, j$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^* \wedge$
$\mathit{ei}(\tau(i)) = \mathit{ei}(\tau(j)) \rightarrow$
$i = j$

Proof.
By the invariants:
In every transition, the event ID in the post-state is greater than the event ID in the pre-state.
The ID of every event in a trace is greater than the ID passed in the pre-state of the transitions.
Straightforward induction on the steps.

Lemma 29.
Based on Definition 15, the parent ID, child index and the event orientation uniquely identify events.
$\forall \mathcal{S}, \tau, i, j.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^* \wedge$
$\mathit{pi}(\tau(i)) = \mathit{pi}(\tau(j)) \wedge$
$\mathit{ci}(\tau(i)) = \mathit{ci}(\tau(j)) \wedge$
$o(\tau(i)) = o(\tau(j)) \rightarrow$
$i = j$

Proof.
By the invariants:
In every transition, the event ID in the post-state is greater than the event ID in the pre-state.
The ID of every event in a trace is greater than the ID passed in the pre-state of the transitions.
The event ID in the pre-state is greater than or equal to the child ID of the first event in the trace.
The event ID of every event is greater than its parent ID.
In the trace of every transition, the parent ID of the first event is less than the parent ID of later events.
Straightforward induction on the steps.

Lemma 30.
Based on Definition 15, the parent ID and the child index of an event point back to the origin of that event.
$\forall \mathcal{S}, \tau, i.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^* \wedge$

$d(\tau(i)) = [] \wedge$
$o(\tau(i)) = \uparrow \rightarrow$
$ois(\tau(pi(\tau(i))))(ci(\tau(i))) = e(\tau(i)) \wedge$
$n(\tau(pi(\tau(i)))) = n(\tau(i)) \wedge$
$mself(\tau(pi(\tau(i))))$

Proof.
Straightforward induction on the steps.

Lemma 31.
If an event is issued only once, every executed event that matches it has the same parent ID and child index.
$\forall \mathcal{S}, \tau, i, j, e, n, d. \exists k.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^* \wedge$
$occ(ois(\tau(i)), e) \leq 1 \wedge$
$\forall k \neq i.\ (n(\tau_k) = n \wedge mself(\tau_k)) \rightarrow e \notin ois(\tau_k) \wedge$
$n(\tau(j)) = n \wedge$
$d(\tau(j)) = [] \wedge$
$o(\tau(j)) = \uparrow \wedge$
$e(\tau(j)) = e \ \rightarrow$
$pi(\tau(j)) = i \wedge$
$ci(\tau(j)) = k$

Proof.
Immediate from Lemma 30.

Lemma 32.
$\forall \mathcal{S}, \tau, i, j, k, e, n, d.$
$w_0(\mathcal{S}) \xrightarrow{\tau}{}^* \wedge$
$occ(ois(\tau(i)), e) \leq 1 \wedge$
$\forall k \neq i.\ (n(\tau_k) = n \wedge mself(\tau_k)) \rightarrow e \notin ois(\tau_k) \wedge$
$n(\tau(j)) = n \wedge$
$d(\tau(j)) = [] \wedge$
$o(\tau(j)) = \uparrow \wedge$
$e(\tau(j)) = e \wedge$
$k \neq j \ \rightarrow$
$\neg[n(\tau_k) = n \ \wedge \ d(\tau_k) = [] \ \wedge \ o(\tau_k) = \uparrow \ \wedge \ e(\tau_k) = e]$

Proof.
Immediate from Lemma 31.

Lemma 33.
$\forall W, W', \tau, i.$

$W \xrightarrow{\tau}{}^{*} W'$
$\rightarrow$
$n(\tau(i)) \in \mathbb{N}$

Proof.
By induction on the steps.

Lemma 34.
The derived rules are sound.

Proof.
Case rule FLoss:
Immediate from GST.

Case rule IRSe:
By the rule IR, definition of self and rule SInv.
$\mathsf{self} \wedge a \Rightarrow b$
$\mathsf{self} \Rightarrow a \to b$
$\Box(\mathsf{self} \to a \to b)$
Ⓢ $\Box(a \to b)$
Ⓢ $(a \Rightarrow b)$

Case rule PeSe:
Similar to rule IRSe.
By the rule Pe, definition of self and rule SInv.

Case rule IISe:
Similar to rule IRSe.
By the rule II, definition of self and rule SInv.

Case rule ORSe:
By the rule OR, and rule SInv.

Case rule OISe:
By the rule OR, and rule SInv.

Case rule ORSe′:
By the rule OR′, definition of self and rule SInv.

Case rule OISe′:
By the rule OI′, definition of self and rule SInv.

Case rule IROI:
By the rule IR and rule OI.

Case rule IIOI:
By the rule II and rule OI.

Case rule PeOI:
By the rule Pe and rule OI.

Case rule IROR:
By the rule IR and rule OR.

Case rule IIOR:
By the rule II and rule OR.

Case rule PeOR:
By the rule Pe and rule OR.

Case rule IROISe to rule PeORSe:
By the rule IROI and rule PeOR and rule SInv.

Case rule APerSe:
By rule APer, rule InvS and rule ASelf.

Case rule UniORSe:
By the rule UniOR and rule SInv.

Case rule UniOISe:
By the rule UniOI and rule SInv.

Case rule CSelf:
By the definition of self.

Case rule SEqSe:
By rule SEq and rule SInv.

Case rule InvSe:
First proving $\Box(\mathsf{self} \to \mathcal{A})$, by the definition of self,
distributing $\vee$ over $\to$, using $\mathcal{A} \wedge \mathcal{A}' \to \mathcal{A} \vee \mathcal{A}'$, Lemma 84, and
rule $\wedge r$.
Then using rule ASelf and Lemma 79.

Case rule InvSe′:
Similar to the proof of rule InvSe′. Instead of $\Box(\mathsf{self} \to \mathcal{A})$,
we first prove $\Box(\mathsf{self} \wedge \hat{\boxminus}\mathcal{A} \to \mathcal{A})$.
At the end, we use Lemma 110.

Case rule InvUSe:
Using rule InvSe′ and rule IRSe, rule IISe, and rule PeSe.

Case rule InvMSe:
By rule InvSe′ by instantiating $\mathcal{A}$ to $\mathcal{A} \wedge \mathcal{A}'$ and Lemma 93.

Case rule InvMSe′:
By rule InvUSe by instantiating $\mathcal{A}$ to $\mathcal{A} \wedge \mathcal{A}'$ and Lemma 93.

Case rule InvLSe:
By rule InvUSe and rule Gen.

Case rule InvL:

By rule InvLSe and rule SInv.

Case rule InvUSSe:
By Lemma 82 with p instantiated to $S(\mathsf{s}(n))$.
Then using rule PostPre to convert $\bigcirc \mathsf{s}(n)$ to $\mathsf{s}'(n)$.
Then case analysis on whether $\mathsf{n} = n$.
Then rule SEqSe for $\mathsf{n} \neq n$ and rule InvUSe for $\mathsf{n} = n$.

Case rule InvUSSe′:
By rule InvUSSe′, Lemma 83 and rule Init.

Case rule InvSSe:
By rule InvUSSe and rule Gen.

Case rule InvS:
By rule InvSSe and rule SInv.

Case rule InvSSe′:
By rule InvSSe, Lemma 83 and rule Init.

Case rule InvS′:
By rule InvSSe′, and rule SInv.

Case rule InvSSe″:
By rule InvSSe, Lemma 81, rule PostPre.

Case rule InvS″:
By rule InvSSe″, and rule SInv.

Case rule InvSASe:
We have
(1) $\neg S(\mathsf{init}_c(n))$
By rule InvLSe with
$\mathcal{A}$ instantiated to $n \bullet \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n))) \rightarrow \mathcal{A}$,
we have
(2) $\vdash_c \circledS\, [(n \bullet \neg S(\mathsf{s}(n)) \wedge S(\mathsf{s}'(n))) \Rightarrow \mathcal{A}]$
We need to show that
$\vdash_c \circledS\, (S(\mathsf{s}(n)) \Rightarrow \hat{\Diamond}\mathcal{A})$
We show the contra-positive that is
$\vdash_c \circledS\, (\hat{\Box}\neg\mathcal{A} \Rightarrow \neg S(\mathsf{s}(n)))$
By rule Init and [1]
(3) $\vdash_c \circledS\, \neg S(\mathsf{s}(n))$
The contra-positive of [2] is
(4) $\vdash_c \circledS\, [(\neg\mathcal{A} \Rightarrow \mathsf{n} \neq n \vee S(\mathsf{s}(n)) \vee \neg S(\mathsf{s}'(n)))]$
that is
(5) $\vdash_c \circledS\, [(\neg\mathcal{A} \Rightarrow (\mathsf{n} \neq n \vee \neg S(\mathsf{s}(n)) \rightarrow \neg S(\mathsf{s}'(n))))]$
By rule SEqSe and [5]
(6) $\vdash_c \circledS\, [(\neg\mathcal{A} \Rightarrow$

$(\neg S(\mathsf{s}(n)) \to \neg S(\mathsf{s}'(n))) \vee \neg S(\mathsf{s}(n)) \to \neg S(\mathsf{s}'(n))))]$

that is

(7) $\vdash_c \textcircled{S}\ [(\neg\mathcal{A} \Rightarrow \neg S(\mathsf{s}(n)) \to \neg S(\mathsf{s}'(n))))]$

By rule PostPre and [7]

(8) $\vdash_c \textcircled{S}\ [(\neg\mathcal{A} \Rightarrow (\neg S(\mathsf{s}(n)) \to \bigcirc \neg S(\mathsf{s}(n))))]$

By Lemma 103, we have

(9) $\vdash_c \textcircled{S}\ [(\neg S(\mathsf{s}(n)) \to$
$(\hat{\boxminus}((\neg S(\mathsf{s}(n)) \Rightarrow \bigcirc(\neg S(\mathsf{s}(n))) \Rightarrow \neg S(\mathsf{s}(n))]$

By [9] and [3]

(10) $\vdash_c \textcircled{S}\ [(\hat{\boxminus}((\neg S(\mathsf{s}(n)) \Rightarrow \bigcirc(\neg S(\mathsf{s}(n))) \Rightarrow \neg S(\mathsf{s}(n))]$

By [8] and [10]

$\vdash_c \textcircled{S}\ [(\hat{\boxminus}\neg\mathcal{A} \Rightarrow \neg S(\mathsf{s}(n))]$

Case rule InvSA:

By rule InvSASe, and rule SInv.

Case rule InvMSIASe:

By rule InvSe on the assumptions, we have

$\vdash_c \textcircled{S}\ \hat{\boxminus}\mathcal{A} \wedge (\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}')] \Rightarrow$
$\mathcal{A} \wedge (\forall x.\ S(\mathsf{s}'(\mathsf{n})) \to \Diamond\mathcal{A}')$

By Lemma 110, we have

$\vdash_c \Box\mathcal{A}$

By rule PostPre, Lemma 114 and Axiom 18, we have

$\vdash_c \textcircled{S}\ (\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}') \Rightarrow$
$\bigcirc(\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}')$

By Lemma 82

$\vdash_c \textcircled{S}\ (\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}') \Rightarrow \Box(\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}')$

From the assumption $\forall x.\neg S(\mathsf{init}_c(n))$ and rule Init

$\vdash_c \textcircled{S}\ (\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}')$

By Lemma 97

$\vdash_c \textcircled{S}\ \Box(\forall x.\ S(\mathsf{s}(\mathsf{n})) \to \hat{\Diamond}\mathcal{A}')$

Case rule ASASe:

Assumption:

(11) $\vdash_c \textcircled{S}\ \mathcal{A} \Rightarrow S(\mathsf{s}'(n))$

(12) $\vdash_c \textcircled{S}\ S(\mathsf{s}(n)) \Rightarrow \Box\mathcal{A}'$

We prove

$\vdash_c \textcircled{S}\ \mathcal{A} \Rightarrow \hat{\Box}\mathcal{A}'$

By rule PostPre on [11],

(13) $\vdash_c \textcircled{S}\ \mathcal{A} \Rightarrow \bigcirc S(\mathsf{s}(n))$

By Lemma 81 ($\bigcirc$M) on [12]

(14) $\vdash_c \textcircled{S}\ \bigcirc S(\mathsf{s}(n)) \Rightarrow \hat{\Box}\mathcal{A}'$

By Lemma 80 ($\Rightarrow$T) on [13] and [14],

$\vdash_c \textcircled{S}\ \mathcal{A} \Rightarrow \hat{\Box}\mathcal{A}'$

Case rule ASA:

By rule ASASe, and rule SInv.

Case rule APerSA:
By rule APer, and rule Pe.

Case rule Quorum:
Let
(15) $\Gamma = |\mathsf{Correct}| > t_1;\ N \subseteq \mathbb{N};\ |N| > t_2;\ t_1 + t_2 \geq |\mathbb{N}|$
We show that
$\Gamma \vdash_c \exists n.\ n \in N \wedge n \in \mathsf{Correct}$

We have
(16) $\Gamma \vdash_c \mathsf{Correct} \subseteq \mathbb{N}$
From [15]
(17) $\Gamma \vdash_c |\mathsf{Correct}| > t_1$
(18) $\Gamma \vdash_c N \subseteq \mathbb{N}$
(19) $\Gamma \vdash_c |N| > t_2$
(20) $\Gamma \vdash_c |t_1 + t_2| \geq |\mathbb{N}|$
From set theory
(21) $\Gamma \vdash_c \forall S_1, S_2, S.\ \ S_1 \subseteq S \wedge S_1 \subseteq S \rightarrow S_1 \cup S_2 \subseteq S$
By [21] on [16] and [18]
(22) $\Gamma \vdash_c \mathsf{Correct} \cup N \subseteq \mathbb{N}$
From set theory
(23) $\Gamma \vdash_c \forall S_1, S_2.\ \ S_1 \subseteq S_2 \rightarrow |S_1| \leq |S_2|$
By [23] on [22]
(24) $\Gamma \vdash_c |\mathsf{Correct} \cup N| \leq |\mathbb{N}|$
From [17], [19], [24], and [20]
(25) $|\mathsf{Correct}| + |N| - |\mathsf{Correct} \cup N| > 0$
From set theory
(26) $\Gamma \vdash_c \forall S_1, S_2.\ \ |S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2|$
that is
(27) $\Gamma \vdash_c \forall S_1, S_2.\ \ |S_1 \cap S_2| = |S_1| + |S_2| - |S_1 \cup S_2|$
By [27] on [25]
(28) $\Gamma \vdash_c |\mathsf{Correct} \cap N| > 0$
From set theory
(29) $\Gamma \vdash_c \forall S.\ \ |S| > 0 \leftrightarrow S \neq \varnothing$
By [29] on [28]
(30) $\Gamma \vdash_c \mathsf{Correct} \cap N \neq \varnothing$
From set theory
(31) $\Gamma \vdash_c \forall S.\ \ S = \varnothing \leftrightarrow \nexists s.\ s \in S$
By contrapositive of [31] on [30]
(32) $\Gamma \vdash_c \exists n.\ n \in \mathsf{Correct} \cap N$
From set theory
(33) $\Gamma \vdash_c \forall S_1, S_2. \forall s.\ \ s \in S_1 \cap S_2 \leftrightarrow s \in S_1 \wedge s \in S_2$
By contrapositive of [33] on [32]
(34) $\Gamma \vdash_c \exists n.\ n \in \mathsf{Correct} \wedge n \in N$

Thus, we showed that
$|\mathsf{Correct}| > t_1;\ N \subseteq \mathbb{N};\ |N| > t_2;\ t_1 + t_2 \geq |\mathbb{N}| \vdash_c$

$\exists n.\ n \in N \wedge n \in \mathsf{Correct}$

By rule $\rightarrow r$, we have

$|\mathsf{Correct}| > t_1 \ \vdash_c$

$N \subseteq \mathbb{N} \wedge |N| > t_2 \wedge t_1 + t_2 \geq |\mathbb{N}| \rightarrow \exists n.\ n \in N \wedge n \in \mathsf{Correct}$

As the assertion is non-temporal, we have

$|\mathsf{Correct}| > t_1 \ \vdash_c$

$N \subseteq \mathbb{N} \wedge |N| > t_2 \wedge t_1 + t_2 \geq |\mathbb{N}| \Rightarrow \exists n.\ n \in N \wedge n \in \mathsf{Correct}$

5.2 Composition

Lemma 1.
For all $\tau \in T(\mathsf{stack}(c, \overline{\mathcal{S}}))$ there exists $\tau' \in T(\mathcal{S}_i)$ such that $\tau|_{d \supseteq [i]} = \mathsf{push}(i, \tau')$

Proof. Induction on the transition steps of the network semantic (Figure 14) for τ :
The trace τ' on $\mathcal{S}_i$ with state (σ', ms', f, r) is inductively build from the trace τ on $\mathsf{stack}(c, \overline{\mathcal{S}})$ with the state (σ, ms, f, r).
$\sigma' = \lambda d.\ \sigma(i :: d)$ and
$ms' = \{(n, n', d, m) \mid (n, n', i :: d, m) \in ms\}$
The induction hypothesis is strengthened with the fact
$\forall d.\ \sigma(i :: d) = \sigma'(d)$
that is the state of the substack i of $\mathcal{S}$ is the same as the state of $\mathcal{S}_i$.
On every step j of τ on (σ, ms, f, r):
(1) If the step is not in substack i that is $d(\tau(j)) \not\supseteq [i]$, no step ($\epsilon$ step) is taken for τ'.
(2) If the step is in substack i that is $\tau(j) = (n, r, i :: d, o, e, s, s', ors, ois)$, a corresponding step $(n, r, d, o, e, s|_{d \supseteq [i]}, s'|_{d \supseteq [i]}, ors, ois)$ can be applied to (σ', ms', f', r'). Note that all the elements except d and the states of the two events are the same. The state $s|_{d \supseteq [i]}$ is a projection over s for locations that are an extension of $[i]$.

Corollary 2.
For all c, $\overline{\mathcal{S}_i}$ and $m \in M(\mathsf{stack}(c, \overline{\mathcal{S}_i}))$, there exists $m' \in M(\mathcal{S}_i)$ such that $m|_{d \supseteq [i]} = \mathsf{push}(i, m')$

Proof. Immediate from Definition 12 and Corollary 2.

Lemma 2. For all m, $\mathcal{A}^{[]}$ and i,
$m \vDash \mathcal{A}^{[]} \rightarrow$
$\mathsf{push}(i, m) \vDash \mathsf{push}(i, \mathcal{A}^{[]})$

Proof. Immediate from induction on the structure of $\mathcal{A}^{[]}$.

Definition 16 (Expanding a Model)**.**

$$\begin{aligned} &\mathit{ext}((\tau, j, I), i) \triangleq \{(\tau'', j'', I) \mid \\ &\quad \exists \tau'.\ \forall j.\ d(\tau'(j)) \not\supseteq [i] \wedge \tau'' \in \mathit{interleave}(\tau, \tau') \wedge \tau''(j'') = \tau(j)\} \end{aligned}$$

We assume that events of a trace are unique. Uniqueness of events can be simply provided by the semantics with separate counters in nodes.

Lemma 3.
For all m, i, m' and $\mathcal{I}^{[i]}$,
$m \vDash \mathcal{I}^{[i]} \wedge$
$m' \in \mathsf{extend}(i, m) \rightarrow$
$m' \vDash \mathsf{restrict}(\mathsf{d} = [i], \mathcal{I}^{[i]})$.

Proof.
Let
(1) $\mathcal{I}^{[i]} = \Box\mathcal{A}^{[i]}$
We assume that
(2) $m \vDash \Box\mathcal{A}^{[i]}$
(3) $m' \in \mathsf{extend}(i, m)$
We show that
$m' \vDash \mathsf{restrict}(\mathsf{d} = [i], \Box\mathcal{A}^{[i]})$
From Definition 8 (the definition of extend), on [3], there exists τ, τ' and I such that
(4) $m = (\tau, 0, I)$
(5) $m' = (\tau'', 0, I)$
(6) $\forall j.\ d(\tau'(j)) \not\equiv [i]$
(7) $\tau'' \in \mathsf{interleave}(\tau, \tau')$
From [5], we need to show that
$(\tau'', 0, I) \vDash \mathsf{restrict}(\mathsf{d} = [i], \Box\mathcal{A}^{[i]})$
From Definition 3 (layering of assertions), we need to show that
$(\tau'', 0, I) \vDash \Box(\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$
From Definition 6 (the definition of the models relation), we need to show that
$(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$ forall $k'' \geq 0$

By [7], let j'' be the index where the element at index 0 of τ appears i.e.
(8) $\tau''(j'') = \tau(0)$
We consider two cases:

Case:
(9) $k'' < j''$
From [7], [9], [8], and [6],
(10) $d(\tau''(k'')) \neq [i]$
Therefore, the following assertion is trivially holds
$(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$

Case:
(11) $k'' \geq j''$
From Definition 16 (the definition of ext), on [6], [7] and [8],
(12) $(\tau'', j'', I) \in \mathsf{ext}(i, (\tau, 0, I))$
From [12] and [4],
(13) $(\tau'', j'', I) \in \mathsf{ext}(i, m)$
By Lemma 35 on [2], [13]
(14) $(\tau'', j'', I) \vDash \mathsf{restrict}(\mathsf{d} = [i], \Box\mathcal{A}^{[i]})$
From Definition 3 (layering of assertions) on [14]
(15) $(\tau'', j'', I) \vDash \Box(\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$
From Definition 6 (the definition of the models relation) on [15]
(16) $(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$
forall $k'' \geq j''$
From [16] and [11]

$(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$

Lemma 35.
For all m, i, m' *and* $\mathcal{A}^{[i]}$,
$m \vDash \mathcal{A}^{[i]} \ \wedge$
$m' \in \mathit{ext}(i, m) \ \rightarrow$
$m' \vDash \mathit{restrict}(\mathit{d} = [i], \mathcal{A}^{[i]})$

Proof.
Immediate from Lemma 36.

Lemma 36. *For all* m, i, $\mathcal{A}^{[i]}$, *and* m',
if $m' \in \mathit{ext}(i, m)$, *then*
$m \vDash \mathcal{A}^{[i]} \ \leftrightarrow \ m' \vDash \mathit{restrict}(\mathit{d} = [i], \mathcal{A}^{[i]})$

Proof.
We assume that
(1) $m' \in \mathsf{ext}(i, m)$
From Definition 16 (the definition of ext), on [1], we have
(2) $m = (\tau, j, I)$
(3) $m' = (\tau'', j'', I)$
(4) $\forall j.\ d(\tau'(j)) \not\supseteq [i]$
(5) $\tau'' \in \mathsf{interleave}(\tau, \tau')$
(6) $\tau''(j'') = \tau(j)$
We show that
$m \vDash \mathcal{A}^{[i]}$
if and only if
$m' \vDash \mathsf{restrict}(\mathsf{d} \supseteq [i], \mathcal{A}^{[i]})$

Induction on the structure of $\mathcal{A}^{[i]}$

Case $\mathsf{n} = t_1 \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = t_2 \ \wedge \ \mathsf{e} = t_3 \quad d' \supseteq [i]$:
The forward direction:
We assume that
(7) $m \vDash \mathsf{n} = t_1 \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = t_2 \ \wedge \ \mathsf{e} = t_3 \quad d' \supseteq [i]$
We show that
$m' \vDash \mathsf{restrict}(\mathsf{d} \supseteq [i], \mathsf{n} = t_1 \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = t_2 \ \wedge \ \mathsf{e} = t_3)$
From Definition 6 (the definition of the models relation),
on [2] and [7]
(8) $m \vDash t_1 : v_1$
(9) $m \vDash t_2 : v_2$
(10) $m \vDash t_3 : v_3$
(11) $n(\tau(j)) = v_1$
(12) $d(\tau(j)) = d' \quad d' \supseteq [i]$
(13) $o(\tau(j)) = v_2$

(14) $e(\tau(j)) = v_3$

From [2], [3] and [6] on [11], [12], [13] and [14]

(15) $m' \vDash t_1 : v_1$

(16) $m' \vDash t_2 : v_2$

(17) $m' \vDash t_3 : v_3$

(18) $n(\tau''(j'')) = v_1$

(19) $d(\tau''(j'')) = d' \quad d' \supseteq [i]$

(20) $o(\tau''(j'')) = v_2$

(21) $e(\tau''(j'')) = v_3$

From Definition 6 (the definition of the models relation), on [15] to [21]

(22) $m' \vDash (\mathsf{n} = n \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = o \ \wedge \ \mathsf{e} = e) \quad d' \supseteq [i]$

From Definition 3, (layering of assertions)

(23) $\mathsf{restrict}(\mathsf{d} \supseteq [i], \mathsf{n} = n \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = o \ \wedge \ \mathsf{e} = e) =$
$\mathsf{n} = n \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = o \ \wedge \ \mathsf{e} = e$

From [22], [23]

$m' \vDash \mathsf{restrict}(\mathsf{d} = [i], \mathsf{n} = n \ \wedge \ \mathsf{d} = d' \ \wedge \ \mathsf{o} = o \ \wedge \ \mathsf{e} = e)$

The backward direction is similar.

Case $\mathcal{A}_1^{[i]} \wedge \mathcal{A}_2^{[i]}$:

Immediate from the induction hypothesis.

Case $\neg\mathcal{A}^{[i]}$:

Forward direction Immediate from the backward direction of the induction hypothesis and vice versa.

Case $\Box\mathcal{A}^{[i]}$:

The forward direction:

We assume that

(24) $m \vDash \Box\mathcal{A}^{[i]}$

We show that

$m' \vDash \mathsf{restrict}(\mathsf{d} = [i], \Box\mathcal{A}^{[i]})$

From Definition 3 (layering of assertions), we show that

$m' \vDash \Box(\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$

From Definition 6 (the definition of the models relation), on [3], we show that

$(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$

forall $k'' \geq j''$

From Definition 6 (the definition of the models relation), on [2] and [24]

(25) $(\tau, k, I) \vDash \mathcal{A}^{[i]} \quad$ forall $k \geq j$

For all k'', such that

(26) $k'' \geq j''$

We consider two cases:

Case:

(27) $\mathsf{d}(\tau''(k'')) \neq [i]$

The assertion is trivially true.

$(\tau'', k'', I) \vDash (\mathsf{d} = [i] \rightarrow \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]}))$

Case:

(28) $\mathsf{d}(\tau''(k'')) = [i]$

We need to show that

$(\tau'', k'', I) \vDash \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]})$

From [5], [4], and [6] on [28] and [26], there exists k, such that

(29) $k \geq j$

(30) $\tau''(k'') = \tau_k$

From [25] and [29]

(31) $(\tau, k, I) \vDash \mathcal{A}^{[i]}$

From [4], [5] and [30],

(32) $(\tau'', k'', I) \in \mathsf{ext}(i, (\tau, k, I))$

By the induction hypothesis on [31] and [32],

$(\tau'', k'', I) \vDash \mathsf{restrict}(\mathsf{d} = [i], \mathcal{A}^{[i]})$

The backward direction is similar.

Other cases are similar.

Theorem 1 (Composition).
For all $\mathcal{S}$, c, and $\overline{\mathcal{S}_i}$,
$\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i}) \wedge$
$\vDash_{\mathcal{S}_i} \mathcal{I}_i^{[]}$
$\rightarrow$
$\vDash_{\mathcal{S}} \mathsf{lower}(i, \mathcal{I}_i^{[]})$

Proof.
We assume that

(1) $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i})$

(2) $\vDash_{\mathcal{S}_i} \mathcal{I}_i^{[]}$

We show that

$\mathcal{S} \vDash \mathsf{lower}(i, \mathcal{I}_i^{[]})$

From Definition 4 on [2]

(3) $\forall m' \in M(\mathcal{S}_i).\ m' \vDash \mathcal{I}_i^{[]}$

By Lemma 2 on [3]

(4) $\forall m' \in M(\mathcal{S}_i).\ \mathsf{push}(i, m') \vDash \mathsf{push}(i, \mathcal{I}_i^{[]})$

By Corollary 2

(5) $\forall m \in M(\mathcal{S}).\ \exists m' \in M(\mathcal{S}_i).$
$m|_{d \supseteq [i]} = \mathsf{push}(i, m')$

From [5] and [4]

(6) $\forall m \in M(\mathcal{S}).\ m|_{d \supseteq [i]} \vDash \mathsf{push}(i, \mathcal{I}_i^{[]})$

We have

(7) $\forall m \in M(\mathcal{S}).\ m \in \mathsf{extend}(i, m|_{d \supseteq [i]})$

By Lemma 3 on [6] and [7], we have

(8) $\forall m \in M(\mathcal{S}).\ m \vDash \mathsf{restrict}(\mathsf{d} = [i], \mathsf{push}(i, \mathcal{I}_i^{[]}))$

From Definition 1 on [8], we have

(9) $\forall m \in M(\mathcal{S}).\ m \vDash \mathsf{lower}(i, \mathcal{I}_i^{[]})$

From Definition 4 on [8], we have

(10) $\vDash_{\mathcal{S}} \mathsf{lower}(i, \mathcal{I}_i^{[]})$

Corollary 1 (Composition Soundness).
For all $\mathcal{S}$, c, and $\overline{\mathcal{S}_i}$,
$\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i}) \wedge$
$\overline{\vDash_{\mathcal{S}_i} \mathcal{I}_i^{[]}} \wedge$
$\overline{\mathsf{lower}(i, \mathcal{I}_i^{[]})} \vdash_c \mathcal{A}$
$\rightarrow$
$\vDash_{\mathcal{S}} \mathcal{A}$.

Proof.
We assume that

(1) $\mathcal{S} = \mathsf{stack}(c, \overline{\mathcal{S}_i})$

(2) $\overline{\vDash_{\mathcal{S}_i} \mathcal{I}_i^{[]}}$

(3) $\overline{\mathsf{lower}(i, \mathcal{I}_i^{[]})} \vdash_c \mathcal{A}$

We show that

$\vDash_{\mathcal{S}} \mathcal{A}$

From Theorem 1 on [2], we have

(4) $\overline{\vDash_{\mathcal{S}} \mathsf{lower}(i, \mathcal{I}_i^{[]})}$

From Corollary 2 on [4] and [3], we have

$\vDash_{\mathcal{S}} \mathcal{A}$

5.3 Component Verification

5.3.1 Stubborn Links

Theorem 4. *(SL_1: Stubborn delivery)*
If a correct node n sends a message m to a correct node n', then n' delivers m infinitely often.

$$\vdash_{\mathsf{SLC}} n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow (n \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow \Box\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, m))$$

Proof.

The proof idea: upon a stubborn link send request for a message m, m is added to the sent set. The periodic section of SLC, reissues a send request for every message in the sent set. The periodic function executes infinitely often. Thus, send requests are infinitely often issued for every messages in the send set. By the fair-loss property of the basic link, if a message is infinitely sent, then it will be infinitely often delivered. If a basic deliver indication is issued, then a stubborn link deliver indication is issued and then eventually executed.

We assume

(1) $\Gamma = n \in \mathsf{Correct}; n' \in \mathsf{Correct}$

We prove that

$$\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow \Box\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, m))$$

By rule IR,

(2) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow (\mathsf{self} \wedge \langle n', m\rangle \in \mathsf{s}'(n))$

By rule InvS″ with $S = \lambda s.\ \langle n', m\rangle \in s$,

(3) $\Gamma \vdash_{\mathsf{SLC}} (\mathsf{self} \wedge \langle n', m\rangle \in \mathsf{s}'(n)) \Rightarrow \hat{\Box}(\mathsf{self} \Rightarrow \langle n', m\rangle \in \mathsf{s}(n))$

From [2] and [3],

(4) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow \hat{\Box}(\mathsf{self} \Rightarrow \langle n', m\rangle \in \mathsf{s}(n))$

By rule APerSA with:

$S = \lambda s.\ \langle n', m\rangle \in s$ and

$\mathcal{A} = n \bullet (0, \mathsf{send}_{\mathsf{l}}(n', m)) \in \mathsf{ors} \wedge \mathsf{self}$ on [1], we have

(5) $\Gamma \vdash_{\mathsf{SLC}} n \in \mathsf{Correct} \rightarrow (\mathsf{self} \Rightarrow \langle n', m\rangle \in \mathsf{s}(n)) \Rightarrow \Box\Diamond(n \bullet (0, \mathsf{send}_{\mathsf{l}}(n', m)) \in \mathsf{ors} \wedge \mathsf{self})$

From [4] and [5], we have

(6) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow$

$$\Box\Diamond(n \bullet (0, \mathsf{send_l}(n', m)) \in \mathsf{ors} \wedge \mathsf{self})$$

From rule OR,

(7) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet (0, \mathsf{send_l}(n', m)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$\Diamond(n \bullet 0 \downarrow \mathsf{send_l}(n', m))$

From [6] and [7],

(8) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow$
$\Box\Diamond\Diamond(n \bullet 0 \downarrow \mathsf{send_l}(n', m))$

From Lemma 87 on [8],

(9) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow$
$\Box\Diamond(n \bullet 0 \downarrow \mathsf{send_l}(n', m))$

By rule FLoss on [1] and [9],

(10) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow$
$\Box\Diamond(n' \bullet 0 \downarrow \mathsf{deliver_l}(n, m))$

From rule IIOI,

(11) $\Gamma \vdash_{\mathsf{SLC}} (n' \bullet 0 \downarrow \mathsf{deliver_l}(n, m)) \Rightarrow$
$\Diamond(n' \bullet \top \downarrow \mathsf{deliver_{sl}}(n, m))$

From [10] and [11],

(12) $\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow$
$\Box\Diamond\Diamond(n' \bullet \top \downarrow \mathsf{deliver_{sl}}(n, m))$

From Lemma 87 and [12]

$\Gamma \vdash_{\mathsf{SLC}} (n \bullet \top \downarrow \mathsf{send_{sl}}(n', m)) \Rightarrow$
$\Box\Diamond(n' \bullet \top \downarrow \mathsf{deliver_{sl}}(n, m))$

Theorem 5. *(SL_2: No-forge)*
If a node n delivers a message m with sender n', then m was previously sent to n by n'.

$\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$

Proof.

The proof idea: If a stubborn link delivery event is executed, it is issued by a previous basic link delivery event. By the no-forge property of basic links, we know that any basic link delivery event is preceded by a basic link send event. A basic link send event is only issued by a previous stubborn link send event.

By rule OI′,
(1) $\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \Rightarrow$
$\Diamond\!\!-(n \bullet \mathsf{deliver}_{\mathsf{sl}}(n', m) \in \mathsf{ois} \wedge \mathsf{self})$
By rule InvL,
(2) $\vdash_{\mathsf{SLC}} (n \bullet \mathsf{deliver}_{\mathsf{sl}}(n', m) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{l}}(n', m))$
By using [1] and [2],
$\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \Rightarrow$
$\Diamond\!\!-(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{l}}(n', m))$
That is,
(3) $\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \leftsquigarrow$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{l}}(n', m))$
By rule NForge,
(4) $\vdash_{\mathsf{SLC}} (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{l}}(n', m)) \leftsquigarrow$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{l}}(n, m))$
By Lemma 88 on [3], [4],
(5) $\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \leftsquigarrow$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{l}}(n, m))$
By rule OR′,
(6) $\vdash_{\mathsf{SLC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{l}}(n, m)) \Rightarrow$
$\Diamond\!\!-(n' \bullet (0, \mathsf{send}_{\mathsf{l}}(n, m)) \in \mathsf{ors} \wedge \mathsf{self})$
By rule InvL,
(7) $\vdash_{\mathsf{SLC}} (n' \bullet (0, \mathsf{send}_{\mathsf{l}}(n, m)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$
From [6] and [7],
$\vdash_{\mathsf{SLC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{l}}(n, m)) \Rightarrow$
$\Diamond\!\!-(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$
That is,
(8) $\vdash_{\mathsf{SLC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{l}}(n, m)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$
From [5] and [8],
$\vdash_{\mathsf{SLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \leftsquigarrow$

$(n' \bullet \mathsf{T} \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$

5.3.2 Perfect Links

Definition 17.
Lowering the properties of the stubborn link.
$\Gamma = \mathsf{SL}'_1, \mathsf{SL}'_2$

$\mathsf{SL}'_1 = \mathsf{lower}(0, \mathsf{SL}_1) =$
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n', m)) \Rightarrow$
$\Box\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, m))$

$\mathsf{SL}'_2 = \mathsf{lower}(0, \mathsf{SL}_2) =$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', m)) \rightsquigarrow$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n, m))$

Theorem 6. *(PL_1: Reliable delivery)*
If a correct node n sends a message m to a correct node n', then n' will eventually deliver m.

$\Gamma \vdash_{\mathsf{PLC}}$
$\quad n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad (n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow$
$\quad (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$
where
Γ is defined in Definition 17.

Proof.

The proof idea: As a $\mathsf{send}_{\mathsf{pl}}$ event is executed at the sender, a $\mathsf{send}_{\mathsf{sl}}$ event is issued and eventually executes. Thus, by the stubborn link, a $\mathsf{deliver}_{\mathsf{sl}}$ event eventually executes at the receiver. At the $\mathsf{deliver}_{\mathsf{sl}}$ event, either the sender and counter pair is already in the current state or not. (1) If it is not, a $\mathsf{deliver}_{\mathsf{pl}}$ event is issued that eventually executes. This is the desired conclusion. (2) If the counter is already in the current state, a $\mathsf{deliver}_{\mathsf{sl}}$ event with the same sender and counter (with potentially a different message) has executed in the past that has issued a $\mathsf{deliver}_{\mathsf{pl}}$ event that will eventually execute. There cannot be two $\mathsf{deliver}_{\mathsf{sl}}$ events from the same sender with the same counter. Thus, the two $\mathsf{deliver}_{\mathsf{sl}}$ events are the same and carry the same message. Thus, the $\mathsf{deliver}_{\mathsf{pl}}$ event that will be executed is with the sent message.

We assume
$\quad \mathcal{A}_1 = n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct}$
$\quad \Gamma' = \Gamma; A_1$
We prove that
$\quad \Gamma' \vdash_{\mathsf{PLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

By rule IR, the definition of $\mathsf{request}$ and rule OR, there exists c such that
$\quad (1)\ \Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \Rightarrow$
$\qquad\qquad \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \in \mathsf{ors} \wedge \Diamond(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle))$

From SL_1' and Axiom 1,
$\quad (2)\ \Gamma' \vdash_{\mathsf{PLC}} n \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle) \Rightarrow$
$\qquad\qquad \Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle))$

By rule Lemma 89 on [1] and [2],
$\quad (3)\ \Gamma' \vdash_{\mathsf{PLC}} (n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m)) \Rightarrow$

$$\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m\rangle))$$

By rule Lemma 99 on [3] and Lemma 38,

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \Rightarrow$
$\quad \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge$
$\quad \Diamond[\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m)) \vee$
$\quad\quad \exists m.\ \Diamond\!\!\!-[\mathsf{self} \wedge (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))]]$

that is

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \Rightarrow$
$\quad \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge$
$\quad (\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m)) \vee$
$\quad\quad \exists m.\ \hat{\Diamond}\!\!\!-[\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))] \vee$
$\quad\quad \exists m.\ [\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))] \vee$
$\quad\quad \exists m.\ \hat{\Diamond}[\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))])$

By Lemma 37

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \Rightarrow$
$\quad \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge$
$\quad (\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m)) \vee$
$\quad\quad \exists m.\ [\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))])$

Thus,

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \Rightarrow$
$\quad \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

Lemma 37.

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \ \Rightarrow$
$\quad \hat{\Box}\neg[\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m'\rangle)) \in \mathsf{ors}]$
$\quad \hat{\boxminus}\neg[\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m'\rangle)) \in \mathsf{ors}]$

Proof.

By the rule InvL

(1) $\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \Rightarrow$
$\quad\quad \mathsf{counter}(\mathsf{s}'(n)) = c$

By the rule InvS

(2) $\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \mathsf{counter}(\mathsf{s}(n)) \geq c \Rightarrow$
$\quad\quad (\mathsf{self} \Rightarrow \mathsf{counter}(\mathsf{s}(n)) \geq c)$

By the rule ASA on [1] and [2],

(3) $\Gamma' \vdash_{\mathsf{PLC}} n \bullet \mathsf{self} \wedge (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \Rightarrow$
$\quad\quad \hat{\Box}(\mathsf{self} \rightarrow \mathsf{counter}(\mathsf{s}(n)) \geq c)$

By the rule InvL

(4) $\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \mathsf{counter}(\mathsf{s}(n)) \geq c \Rightarrow$
$\quad\quad \neg(n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m'\rangle)) \in \mathsf{ors})$

From [3] and [4]

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \Rightarrow$
$\quad \hat{\Box}(\mathsf{self} \rightarrow \neg(n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m'\rangle)) \in \mathsf{ors}))$

that is

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \Rightarrow$

$\hat{\Box}\neg(\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m' \rangle))) \in \mathsf{ors})$

The proof of the second conjunct is similar.

Lemma 38.

$\Gamma' \vdash_{\mathsf{PLC}}$
$(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle)) \Rightarrow$
$\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m)) \vee$
$\exists m.\ \Diamond\!\!\!-[(n \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))]$

Proof.

Immediate from considering two cases $(n, m) \notin \mathsf{received}(\mathsf{s}(n'))$ and $(n, m) \in \mathsf{received}(\mathsf{s}(n'))$ and Lemma 39 and Lemma 40.

Lemma 39.

$\Gamma' \vdash_{\mathsf{PLC}} (n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle)) \wedge \langle n, c \rangle \notin \mathsf{received}(\mathsf{s}(n')) \Rightarrow$
$\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

Proof.

Immediate from rule II and rule OI.

Lemma 40.

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \mathsf{n} = n' \wedge \langle n, c \rangle \in \mathsf{received}(\mathsf{s}(n')) \Rightarrow$
$\exists m.\ \Diamond\!\!\!-[n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))]$

Proof.

By Lemma 41

(1) $\Gamma' \vdash_{\mathsf{PLC}} \langle n, c \rangle \in \mathsf{received}(\mathsf{s}(n')) \wedge \mathsf{self} \Rightarrow$
$\exists m.\ \Diamond\!\!\!-[n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle) \wedge \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}]$

By SL_2'

(2) $\Gamma' \vdash_{\mathsf{PLC}} n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle) \Rightarrow \Diamond\!\!\!-(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle))$

By rule OR$'$,

(3) $\Gamma' \vdash_{\mathsf{PLC}} (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \Rightarrow \Diamond\!\!\!-(n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self})$

By the Lemma 89 on [2], and [3],

(4) $\Gamma' \vdash_{\mathsf{PLC}} n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m \rangle) \Rightarrow \Diamond\!\!\!-(n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self})$

By rule OI,

(5) $\Gamma' \vdash_{\mathsf{PLC}} (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}) \Rightarrow \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

By [1], [4], [5],

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \langle n, m \rangle \in \mathsf{received}(\mathsf{s}(n')) \Rightarrow \exists m.\ \Diamond\!\!\!-[\Diamond\!\!\!-(n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))]$

that is

$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge (n, m) \in \mathsf{received}(\mathsf{s}(n')) \Rightarrow \exists m.\ \Diamond\!\!\!-[n \bullet (0, \mathsf{send}_{\mathsf{sl}}(n', \langle c, m\rangle)) \in \mathsf{ors} \wedge \mathsf{self} \wedge \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))]$

Lemma 41.
$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \langle n, c\rangle \in \mathsf{received}(\mathsf{s}(n')) \Rightarrow$
$\exists m.\ \Diamond\!\!\!-[n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m\rangle) \wedge \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}]$

Proof.
We use rule InvSA with
n instantiated to n',
S instantiated to $\lambda s.\ \langle n, c\rangle \in \mathsf{received}(s)$ and
$\mathcal{A}$ instantiated to $\exists m.\ n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m\rangle) \wedge \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}$
From the definition of PLC
$\mathsf{init}_{\mathsf{PLC}} = \lambda n.\ \langle 0, \varnothing\rangle$
Thus
$(n, m) \notin \mathsf{received}(\mathsf{init}_{\mathsf{PLC}}(n))$
Thus
(1) $\neg S(\mathsf{init}_{\mathsf{PLC}}(n))$

From the definition of request, we have
(2) $\mathsf{o} = \downarrow \wedge\ \mathsf{d} = \top \wedge$
$\mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), \mathsf{e}) = (\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) \wedge$
$\langle n, c\rangle \notin \mathsf{received}(\mathsf{s}(\mathsf{n})) \wedge \langle n, c\rangle \in \mathsf{received}(\mathsf{s}'(\mathsf{n})) \rightarrow$
$\exists m.\ n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m\rangle) \wedge \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}$
and similarly for indication and periodic.

By rule InvSA on [1] and [2],
$\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge (n, m) \in \mathsf{received}(\mathsf{s}(n')) \Rightarrow$
$\exists m.\ \Diamond\!\!\!-[n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n, \langle c, m\rangle) \wedge \mathsf{deliver}_{\mathsf{pl}}(n, m) \in \mathsf{ois}]$

Theorem 7. *(PL_2: No-duplication)*
If a message is sent at most once, it will be delivered at most once.

$$\Gamma \vdash_{\mathsf{PLC}} [n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n,m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n,m))] \rightarrow$$
$$[n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m))]$$

where
Γ is defined in Definition 17.

Proof.

Proof idea: If there is a perfect-link delivery event, there is a stubborn link delivery event before it. That stubborn link delivery event is the event that issues one perfect-link delivery event and there is no event before or after it that issues a perfect-link delivery event (Lemma 42). Therefore, no perfect-link delivery event other than the current one can be executed.

We assume
$\mathcal{A}_1 = n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n,m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n,m))$
$\Gamma' = \Gamma; \mathcal{A}_1$
and prove
$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m))$

By rule OI′,
(1) $\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \Rightarrow$
$\diamondminus(\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n',m) \in \mathsf{ois} \wedge \mathsf{self})$
By rule InvL,
(2) $\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n',m) \in \mathsf{ois} \Rightarrow$
$\mathsf{self} \wedge \mathsf{n} = n \wedge \mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{pl}}(n',m)) = 1]$
By Lemma 99 on [1] and [2],
(3) $\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \Rightarrow$
$\diamondminus(\mathsf{self} \wedge \mathsf{n} = n \wedge \mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{pl}}(n',m)) = 1])$

From Lemma 42,
(4) $\Gamma' \vdash_{\mathsf{PLC}} \textcircled{S}\ [\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n',m) \in \mathsf{ois} \Rightarrow$
$\hat{\boxminus}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \notin \mathsf{ois})]$
By rule SInv on [4],
(5) $\Gamma' \vdash_{\mathsf{PLC}} \mathsf{self} \wedge \mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n',m) \in \mathsf{ois} \Rightarrow$
$\hat{\boxminus}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \notin \mathsf{ois})$

From [3] and [5],
(6) $\Gamma' \vdash_{\mathsf{PLC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \Rightarrow$
$\diamondminus(\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{pl}}(n',m)) = 1] \wedge$
$\hat{\boxminus}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \notin \mathsf{ois}) \wedge$

$$\hat{\boxminus}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}))$$

From rule UniOI,

(7) $\Gamma' \vdash_{\mathsf{PLC}} (\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{pl}}(n', m)) \leq 1 \wedge$
$\hat{\boxdot}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{self} \wedge \mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})) \Rightarrow$
$n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow \hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$

From [6] and [7],

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$
$\Diamond [n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow \hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))]$

which leads to

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$
$n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow \hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$

which leads to

$\Gamma' \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$

Lemma 42.
At every node, a perfect-link delivery event for a message is issued at most once.
$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\hat{\boxdot}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})]$

Proof.

Proof idea: A perfect-link delivery event is only issued in the execution of a stubborn link delivery event. In the execution of this event, the sender and counter pair is recorded and kept in the received set. In the future events, the received set and the past execution of the stubborn link delivery event prevents issuance of any perfect-link delivery event (Lemma 43).

By rule InvLSe, there exists c such that

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois}) \Rightarrow$
$(n', c) \in \mathsf{received}(\mathsf{s}'(n)) \wedge n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m \rangle)]$

Thus, by Lemma 90,

(1) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois}) \Rightarrow$
$(n', c) \in \mathsf{received}(\mathsf{s}'(n)) \wedge \Diamond (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m \rangle))]$

By Lemma 43,

(2) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \Diamond (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m \rangle)) \Rightarrow$
$\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}]$

By rule InvSSe,

(3) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n', c) \in \mathsf{received}(\mathsf{s}(n)) \Rightarrow$

$$\Box(n', c) \in \mathsf{received}(\mathsf{s}(n))]$$

By Lemma 107,

(4) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)) \Rightarrow$
$\Box(\diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)))]$

From [3] and [4]

(5) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)) \Rightarrow$
$\Box((n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)))]$

By Lemma 100 on [5] and [2],

(6) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)) \Rightarrow$
$\Box(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})]$

By rule ASASE on [1] and [6],

(7) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois} \Rightarrow$
$\hat{\Box}(\Box(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}))]$

that is

(8) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois} \Rightarrow$
$\hat{\Box}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})]$

By Lemma 108 on [8],

(9) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\mathsf{n} = n \wedge \mathsf{deliver}_{\mathsf{pl}}(n', m) \in \mathsf{ois} \Rightarrow$
$\hat{\Box}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})] \wedge$
$\hat{\boxminus}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois})]$

Lemma 43.
$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)) \Rightarrow$
$\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \notin \mathsf{ois}]$

Proof.

Proof idea: A perfect-link delivery event is only issued in the execution of a stubborn link delivery event. A stubborn link delivery event is executed in the past. Any other stubborn link delivery event with the same sender and message has the same counter (Lemma 44). On any stubborn link delivery event, if the pair of the sender and the counter are already in the delivered set, the indication function of the protocol does not issue a perfect-link delivery event. Thus, no perfect-link delivery event with the same sender and message is issued.

We use rule INVSE.
The request and periodic obligations are trivial by rule IR and rule PE.
For the indication obligation, we have to prove that

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\forall i, e.\ i \uparrow e \Rightarrow$
$(n', c) \in \mathsf{received}(\mathsf{s}(n)) \ \wedge\ \diamondminus(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle)) \rightarrow$

$$\mathsf{n} = n \rightarrow \mathsf{deliver_{pl}}(n', m) \notin \mathsf{ois}$$

That is

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\forall i, e.\ n \bullet i \uparrow e \ \wedge\ (n', c) \in \mathsf{received}(\mathsf{s}(n)) \ \wedge\ \Diamond\!\!-(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m\rangle)) \Rightarrow \mathsf{deliver_{pl}}(n', m) \notin \mathsf{ois}$$

By rule IISE,

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [n \bullet i \uparrow e \Rightarrow \exists n', c, m.\ i = 0 \ \wedge\ e = \mathsf{deliver_{sl}}(n', \langle c, m\rangle)$$

Thus, we show that

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\forall n'', c', m'. n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n'', (c', m')) \wedge (n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \Diamond\!\!-(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m\rangle)) \Rightarrow \mathsf{deliver_{pl}}(n', m) \notin \mathsf{ois}$$

We consider two cases whether $(n'' = n' \ \wedge\ m' = m)$:

If $\neg(n'' = n' \ \wedge\ m' = m)$, the result is immediate from the rule IISE.

Thus, we show that

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c', m\rangle) \wedge (n', c) \in \mathsf{received}(\mathsf{s}(n)) \wedge \Diamond\!\!-(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m\rangle)) \Rightarrow \mathsf{deliver_{pl}}(n', m) \notin \mathsf{ois}$$

By Lemma 44, we need to show that

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m\rangle) \ \wedge\ (n', c) \in \mathsf{received}(\mathsf{s}(n)) \Rightarrow \mathsf{deliver_{pl}}(n', m) \notin \mathsf{ois}$$

That is immediate from rule IISE.

Lemma 44.
Every two stubborn link delivery events with the same sender and message have the same counter.

$$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m\rangle) \ \wedge\ \Diamond\!\!-(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c', m\rangle))) \Rightarrow c = c']$$

Proof.

Proof idea: The execution of any stubborn link delivery event is preceded by a stubborn link send event. The send event is preceded by its issuance. A stubborn link send event is only issued by a perfect-link send event. Therefore, from the fact that there are two stubborn link delivery events, we have that in the past there has been two perfect-link send events. However, the assumption is that there is at most one perfect-link send event. Therefore, the two events are the same and they send the same counter. Thus, the counters of the two stubborn link delivery events are the same.

By SL_2',

(1) $\Gamma' \vdash_{\mathsf{PLC}} n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle) \Rightarrow$
$\Diamond(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n, \langle c, m\rangle))$

By rule OR′

(2) $\Gamma' \vdash_{\mathsf{PLC}} n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{sl}}(n, \langle c, m\rangle) \Rightarrow$
$\Diamond(n' \bullet (0, \mathsf{send}_{\mathsf{sl}}(n, \langle c, m\rangle)) \in \mathsf{ors} \wedge \mathsf{self})$

By the rule InvL,

(3) $\Gamma' \vdash_{\mathsf{PLC}} (n' \bullet (0, \mathsf{send}_{\mathsf{sl}}(n, \langle c, m\rangle)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c$

By Lemma 89 and Lemma 99 on [1], [2] and [3],

(4) $\Gamma' \vdash_{\mathsf{PLC}} n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle) \Rightarrow$
$\Diamond(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c$

By rule SInv on [4],

(5) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c, m\rangle) \Rightarrow$
$\Diamond(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c]$

Similarly, we can derive

(6) $\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{sl}}(n', \langle c', m\rangle) \Rightarrow$
$\Diamond(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c']$

Thus, from [5] and [6], we need to show that

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c) \wedge$
$\Diamond\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c')) \Rightarrow$
$c = c']$

That is

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [(\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c) \wedge$
$\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c')) \Rightarrow$
$c = c']$

By Lemma 105, there are two cases that are similar.

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c \wedge$
$\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c')) \Rightarrow$
$c = c']$

By assumption $\mathcal{A}_1$, we need to prove

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\Diamond(\hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c \wedge$
$\Diamond((n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c')) \Rightarrow$
$c = c']$

By Lemma 106, we need to prove

$\Gamma' \vdash_{\mathsf{PLC}} \circledS\, [\Diamond(\mathsf{counter}(\mathsf{s}'(n')) = c \wedge$
$(n' \bullet \top \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \wedge \mathsf{counter}(\mathsf{s}'(n')) = c')] \Rightarrow$
$c = c']$

that is trivial.

Theorem 8. *(PL_3: No-forge)*
If a node n delivers a message m with sender n', then m was previously sent to n by node n'.

$\Gamma \vdash_{\mathsf{PLC}} (n \bullet \top \uparrow \mathsf{deliver_{pl}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{send_{pl}}(n, m))$
where
Γ is defined in Definition 17.

Proof.
By rule OI′,

(1) $\Gamma \vdash_{\mathsf{PLC}} (n \bullet \top \uparrow \mathsf{deliver_{pl}}(n', m)) \Rightarrow$
$\Diamond\!\!\!-(n \bullet \mathsf{deliver_{pl}}(n', m) \in \mathsf{ois} \wedge \mathsf{self})$

By rule InvL,

(2) $\Gamma \vdash_{\mathsf{PLC}} (n \bullet \mathsf{deliver_{pl}}(n', m) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$\exists c.\ (n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m \rangle))$

By Lemma 99 on [1] and [2],

$\Gamma \vdash_{\mathsf{PLC}} (n \bullet \top \uparrow \mathsf{deliver_{pl}}(n', m)) \Rightarrow$
$\exists c.\ \Diamond\!\!\!-(n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m \rangle))$

that is

(3) $\Gamma \vdash_{\mathsf{PLC}} n \bullet \top \uparrow \mathsf{deliver_{pl}}(n', m) \leftsquigarrow$
$\exists c.\ (n \bullet 0 \uparrow \mathsf{deliver_{sl}}(n', \langle c, m \rangle))$

By rule OR′,

(4) $\Gamma \vdash_{\mathsf{PLC}} \forall c.\ (n' \bullet 0 \downarrow \mathsf{send_{sl}}(n, \langle c, m \rangle)) \Rightarrow$
$\Diamond\!\!\!-(n' \bullet (0, \mathsf{send_{sl}}(n, \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self})$

By rule InvL

(5) $\Gamma \vdash_{\mathsf{PLC}} \forall c.\ n' \bullet (0, \mathsf{send_{sl}}(n, \langle c, m \rangle)) \in \mathsf{ors} \wedge \mathsf{self} \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{send_{pl}}(n, \langle c, m \rangle))$

By Lemma 99 on [4] and [5]

(6) $\Gamma \vdash_{\mathsf{PLC}} \forall c.\ (n' \bullet 0 \downarrow \mathsf{send_{sl}}(n, \langle c, m \rangle)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{send_{pl}}(n, m))$

From Lemma 88 on [3], SL_2' and [6],

$\Gamma \vdash_{\mathsf{PLC}} (n \bullet \top \uparrow \mathsf{deliver_{pl}}(n', m)) \leftsquigarrow$
$(n' \bullet \top \downarrow \mathsf{send_{pl}}(n, m))$

5.3.3 Best-Effort Broadcast

Definition 18.
$\Gamma = \mathsf{PL}'_1; \mathsf{PL}'_2; \mathsf{PL}'_3$

$\mathsf{PL}'_1 = \mathsf{lower}(0, \mathsf{PL}_1) =$
$\forall n, n', m.$
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow$
$(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

$\mathsf{PL}'_2 = \mathsf{lower}(0, \mathsf{PL}_2) =$
$[n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m) \Rightarrow \hat{\boxminus} \neg (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))] \rightarrow$
$[n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow \hat{\boxminus} \neg (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))]]$

$\mathsf{PL}'_3 = \mathsf{lower}(0, \mathsf{PL}_3) =$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \rightsquigarrow$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))$

Theorem 9. *(BEB_1: Validity)*
If a correct node broadcasts a message m, then every correct node eventually delivers m.

$\Gamma \vdash_{\mathsf{BEBC}} \forall n, n', m.$
$\quad n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad [(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \rightsquigarrow$
$\quad (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$
where
Γ is defined in Definition 18.

Proof.

The proof idea: The assumption is that the nodes n and n' are both correct. Upon the execution of a $\mathsf{broadcast}_{\mathsf{beb}}$ of a message m from n to n', the component sends m to every node using the perfect link subcomponent. By the reliable delivery property of the perfect link, as both the sender n and receiver n' are correct, m is eventually delivered to n'. Upon the delivery of m by the perfect link, the component issues the delivery of m that eventually executes.

We assume
(1) $\Gamma' = \Gamma; n \in \mathsf{Correct}; n' \in \mathsf{Correct}$
We prove
$\Gamma' \vdash (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \Rightarrow \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

By rule IROR, and the definition of request,
(2) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \Rightarrow \Diamond(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m))$

From PL_1 and [1],
$\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m)) \rightsquigarrow (n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$
That is,
(3) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m)) \Rightarrow \Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

By rule IIOI,
(4) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m)) \Rightarrow \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, m))$

By Lemma 89 on [2], [3] and [4],
$\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow \Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, m))$
That is,
$\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \rightsquigarrow (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, m))$

Theorem 10. *(BEB_2: No-duplication)*
If a message is broadcast at most once, it will be delivered at most once.

$$\Gamma \vdash_{\mathsf{BEBC}} [(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))] \rightarrow [(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$$

where
Γ is defined in Definition 18.

Proof.

The proof idea: We assume that a message is broadcast at most once. By the definition of BEBC, a message is sent by the perfect link subcomponent plc once, only when a broadcast request is processed. Thus, BEBC sends the message to every node at most once by plc. Thus, by the no-duplication property of plc, the message is delivered to every node at most once by plc. By the definition of BEBC, a delivery is issued only when a delivery indication is received from plc. Thus, BEBC issues delivery at every node at most once.

We assume
(1) $\Gamma' = \Gamma$; $[n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))]$
We want to prove
$\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

From [1],
(2) $\Gamma' \vdash_{\mathsf{BEBC}} n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
By Lemma 108 on [2],
(3) $\Gamma' \vdash_{\mathsf{BEBC}} n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \wedge \hat{\Box}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

By rule InvL,
(4) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet (0, \mathsf{send}_p(n, m)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
The contra-positive of [4] is
(5) $\Gamma' \vdash_{\mathsf{BEBC}} \neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \Rightarrow \neg(n' \bullet (0, \mathsf{send}_p(n, m)) \in \mathsf{ors} \wedge \mathsf{self})$

From [3], and [5]
(6) $\Gamma' \vdash_{\mathsf{BEBC}} n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet (0, \mathsf{send}_p(n, m)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge$

$$\hat{\Box}\neg(n' \bullet (0, \mathsf{send}_p(n,m)) \in \mathsf{ors} \wedge \mathsf{self})$$

that is

(7) $\Gamma' \vdash_{\mathsf{BEBC}} n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow$
$\hat{\boxminus}(n' \bullet \mathsf{self} \rightarrow (0, \mathsf{send}_p(n,m)) \notin \mathsf{ors}) \wedge$
$\hat{\Box}(n' \bullet \mathsf{self} \rightarrow (0, \mathsf{send}_p(n,m)) \notin \mathsf{ors})$

By rule OR′

(8) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow \Diamond(n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n,m)) \in \mathsf{ors} \wedge \mathsf{self})$

By rule InvL with $\mathcal{A} =$
$(n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n,m)) \in \mathsf{ors}) \rightarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \wedge \mathsf{occ}(\mathsf{ors}, (0, \mathsf{send}_p(n,m))) = 1$

(9) $\Gamma' \vdash_{\mathsf{BEBC}} \mathsf{self} \wedge (n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n,m)) \in \mathsf{ors}) \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \wedge \mathsf{occ}(\mathsf{ors}, (0, \mathsf{send}_p(n,m))) = 1$

From [8] and [9],

(10) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \wedge \mathsf{occ}(\mathsf{ors}, (0, \mathsf{send}_p(n,m))) = 1]$

From [10] and [7],

(11) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow \Diamond[$
$\mathsf{occ}(\mathsf{ors}, (0, \mathsf{send}_p(n,m))) = 1 \wedge$
$\hat{\boxminus}(n' \bullet \mathsf{self} \rightarrow (0, \mathsf{send}_p(n,m)) \notin \mathsf{ors}) \wedge$
$\hat{\Box}(n' \bullet \mathsf{self} \rightarrow (0, \mathsf{send}_p(n,m)) \notin \mathsf{ors})]$

By rule UniOR on [11]

(12) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow \Diamond[$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow$
$\hat{\boxminus}\neg(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m))]$

By Lemma 109 on [12]

(13) $\Gamma' \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m)) \Rightarrow$
$\hat{\boxminus}\neg(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n,m))$

By PL'_2 on [13],

(14) $\Gamma' \vdash_{\mathsf{BEB}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m))$

By Lemma 108 on [14],

(15) $\Gamma' \vdash_{\mathsf{BEB}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \wedge$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m))$

By rule InvL,

(16) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet (\mathsf{deliver}_{\mathsf{beb}}(n',m)) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m))$

The contra-positive of [16] is

(17) $\Gamma' \vdash_{\mathsf{BEBC}} \neg(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \Rightarrow \neg(n \bullet (\mathsf{deliver}_{\mathsf{beb}}(n',m)) \in \mathsf{ois} \wedge \mathsf{self})$

From [15], and [17],

(18) $\Gamma' \vdash_{\mathsf{BEB}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet (\mathsf{deliver}_{\mathsf{beb}}(n',m)) \in \mathsf{ois} \wedge \mathsf{self}) \wedge$
$\hat{\Box}\neg(n \bullet (\mathsf{deliver}_{\mathsf{beb}}(n',m)) \in \mathsf{ois} \wedge \mathsf{self})$

that is

(19) $\Gamma' \vdash_{\mathsf{BEB}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{pl}}(n',m)) \Rightarrow$

$\hat{\boxminus}(n \bullet \mathsf{self} \rightarrow (\mathsf{deliver}_{\mathsf{beb}}(n', m) \notin \mathsf{ois}) \wedge$
$\hat{\Box}(n \bullet \mathsf{self} \rightarrow (\mathsf{deliver}_{\mathsf{beb}}(n', m) \notin \mathsf{ois})$

By rule OR$'$

(20) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \Diamond(n \bullet (0, \mathsf{deliver}_{\mathsf{beb}}(n', m)) \in \mathsf{ois} \wedge \mathsf{self})$

By rule InvL with $\mathcal{A} =$

$(n \bullet (0, \mathsf{deliver}_{\mathsf{beb}}(n', m)) \in \mathsf{ois}) \rightarrow (n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \wedge \mathsf{occ}(\mathsf{ois}, (0, \mathsf{deliver}_{\mathsf{beb}}(n', m))) = 1$

(21) $\Gamma' \vdash_{\mathsf{BEBC}} \mathsf{self} \wedge (n \bullet (0, \mathsf{deliver}_{\mathsf{beb}}(n', m)) \in \mathsf{ois}) \Rightarrow$
$(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \wedge \mathsf{occ}(\mathsf{ois}, (0, \mathsf{deliver}_{\mathsf{beb}}(n', m))) = 1$

From [20] and [21],

(22) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow$
$\Diamond[(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \wedge \mathsf{occ}(\mathsf{ois}, (0, \mathsf{deliver}_{\mathsf{beb}}(n', m))) = 1]$

From [22] and [19],

(23) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \Diamond[$
$\mathsf{occ}(\mathsf{ois}, (0, \mathsf{deliver}_{\mathsf{beb}}(n', m))) = 1 \wedge$
$\hat{\boxminus}(n \bullet \mathsf{self} \rightarrow (\mathsf{deliver}_{\mathsf{beb}}(n', m) \notin \mathsf{ois}) \wedge$
$\hat{\Box}(n \bullet \mathsf{self} \rightarrow (\mathsf{deliver}_{\mathsf{beb}}(n', m) \notin \mathsf{ois})]$

By rule UniOI on [23],

(24) $\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \Diamond[$
$(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$

By Lemma 109 on [24]

$\Gamma' \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

Theorem 11. *(BEB_3: No-forge)*
If a node delivers a message m with sender n', then m was previously broadcast by node n'.

$\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
where
Γ is defined in Definition 18.

Proof.

The proof idea: If a best-effort broadcast delivery event is executed, it is previously issued. The component issues a best-effort broadcast delivery event only when a delivery event from the perfect link is processed. By the no-forge property of the perfect link, every delivery indication is preceded by a corresponding send event. The component issues a perfect link send request only when a broadcast request is processed.

By rule OI′,
(1) $\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \Diamond(n \bullet \mathsf{deliver}_{\mathsf{beb}}(n', m) \in \mathsf{ois} \wedge \mathsf{self})$
By rule InvL
(2) $\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \mathsf{deliver}_{\mathsf{beb}}(n', m) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$
From [1] and [2]
$\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \Rightarrow \Diamond(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$
that is,
(3) $\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))$

From PL_3
(4) $\Gamma \vdash_{\mathsf{BEBC}} (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \leftsquigarrow (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))$

By rule OR′,
(5) $\Gamma \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \Rightarrow \Diamond(n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n, m)) \in \mathsf{ors} \wedge \mathsf{self})$
By rule InvL
(6) $\Gamma \vdash_{\mathsf{BEBC}} (n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n, m)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
From [5] and [6]
$\Gamma \vdash_{\mathsf{BEBC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \Rightarrow \Diamond(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$
that is
(7) $\Gamma \vdash_{\mathsf{BEBC}} n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

From Lemma 88 on [3], [4] and [7],
$\Gamma \vdash_{\mathsf{BEBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

5.3.4 Uniform Reliable Broadcast

Definition 19.

$\Gamma =$
$\quad |\mathsf{Correct}| > \mathbb{N}/2;$
$\quad \mathsf{BEB}'_1; \mathsf{BEB}'_2; \mathsf{BEB}'_3$

$\mathsf{BEB}'_1 = \mathsf{lower}(0, \mathsf{BEB}_1) =$
$\quad n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad (n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \rightsquigarrow$
$\quad (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

$\mathsf{BEB}'_2 = \mathsf{lower}(0, \mathsf{BEB}_2) =$
$\quad [n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(n, m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(n, m))] \rightarrow$
$\quad [n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$

$\mathsf{BEB}'_3 = \mathsf{lower}(0, \mathsf{BEB}_3) =$
$\quad (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow$
$\quad (n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

Theorem 12. *(URB_1: Validity)*
If a correct node n broadcasts a message m, then n itself eventually delivers m.

$\Gamma \vdash_{\mathsf{URBC}} \forall n.\ n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \rightsquigarrow (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$
where
Γ is defined in Definition 19.

Proof.

The proof idea: Upon a broadcast request for a message m by a node n, the request function broadcasts m using the best-effort broadcast subcomponent bebc. By the validity property of bebc, as n is correct, bebc will eventually deliver m to every correct node. Upon delivery of a message from bebc, the indication function checks whether the message is in the pending set. If it does not find the message in the pending set, it rebroadcasts the message using bebc. Otherwise, the node has already broadcast the message. Therefore, every correct node broadcasts m using bebc. Therefore, by the validity of bebc, bebc eventually delivers m from all correct nodes to the node n. Upon delivery of a message from bebc, the indication function adds the sender to the set of nodes that have acknowledged the message. Thus, at node n, the identifiers of all the correct nodes will eventually be in the acknowledged set for m. This acknowledgement set never shrinks. On the next execution of the periodic function either m is in the delivered set or not. If it is not in the delivered set, since acknowledgement from all the correct nodes has been received and the correct nodes are a majority, the delivery condition is satisfied and the delivery event for m is issued.
Otherwise, if m is not in the delivered set, the delivery of m is previously issued. This delivery cannot be before the original request, based on the following two invariants. First, the counter of a node monotonically increases. Second, the counter of every message in the pending set is less than or equal to the counter of the node. Based on the two invariants, if the delivery of m happens before its broadcast request, the counter of n should have decreased from the former to the latter.

We assume
(1) $\Gamma' = \Gamma; n \in \mathsf{Correct}$
We prove that
$\Gamma \vdash_{\mathsf{URBC}} (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \rightsquigarrow (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$

By rule IRSE and ORSE,
(2) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$
$\exists c.\ \mathit{count}(\mathsf{s}(n)) = c \wedge \Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n, c+1 \rangle))$
By Lemma 45 on [2] and [1]
(3) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$
$\exists c.\ \mathit{count}(\mathsf{s}(n)) = c \wedge \Diamond[$
$\Diamond\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m)) \wedge \Box c + 1 \leq \mathit{count}(\mathsf{s}(n)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))]$
By Lemma 112 on [3] and simplification
$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \ (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$

$$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n,m)) \vee$$
$$[\exists c.\ count(\mathsf{s}(n)) = c \ \wedge \Diamond\!\!\!-(\Box c + 1 \leq count(\mathsf{s}(n)))]$$

that is

$$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ \ (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$$
$$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n,m)) \vee$$
$$[\exists c.\ count(\mathsf{s}(n)) = c \ \wedge c + 1 \leq count(\mathsf{s}(n))]$$

that is

$$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ \ (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$$
$$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n,m))$$

that by rule SInv is

$$\Gamma \vdash_{\mathsf{URBC}} (n \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \rightsquigarrow (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n,m))$$

Lemma 45.

$$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$$
$$n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle) \Rightarrow$$
$$\Diamond\!\!\!-(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s,m)) \wedge \Box c \leq count(\mathsf{s}(n')) \vee$$
$$\Diamond(n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s,m))$$

Proof.

Immediate from Lemma 46 and Lemma 48.

Lemma 46.

$$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$$
$$n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle) \Rightarrow$$
$$\Diamond\Box|ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2$$

Proof.

We assume that

(1) $\Gamma' = \Gamma;\ n \in \mathsf{Correct};\ n' \in \mathsf{Correct}$

We show that

$$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle) \Rightarrow$$
$$\Diamond\Box|ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2$$

By Definition 19 (BEB'_1) on [1] and rule InvS,

(2) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ \forall n'.\ n' \in \mathsf{Correct} \rightarrow$
$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle))$$

By Lemma 47

(3) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n', n''.$
$$(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond\!\!\!-(n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \vee$$
$$\Diamond(n' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle))$$

By Definition 19 (BEB'_1) and rule InvS,

(4) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n'.\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$

$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle))$$

From [3] and [4]

$$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n', n''.\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond\!\!\!-\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle)) \vee$$
$$\Diamond\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle))$$

that by Lemma 112, Lemma 86 and Lemma 87 is

(5) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n', n''.\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond\!\!\!-(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle)) \vee$$
$$\Diamond(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n_s, c\rangle))$$

From [2] and [5]

(6) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ \forall n'.\ n' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow \Diamond[$$
$$\forall n''.\ n'' \in \mathsf{Correct} \rightarrow$$
$$\Diamond\!\!\!-(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle)) \vee$$
$$\Diamond(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle))]$$

that is

(7) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ \forall n'.\ n' \in \mathsf{Correct} \wedge n'' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow \Diamond[$$
$$\Diamond\!\!\!-(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle)) \vee$$
$$\Diamond(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle))]$$

that by Lemma 112, Lemma 86 and Lemma 87 is

(8) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ \forall n'.\ n' \in \mathsf{Correct} \wedge n'' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond\!\!\!-(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle)) \vee$$
$$\Diamond(n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle))$$

By rule IISE,

(9) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n'.$$
$$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle)) \Rightarrow$$
$$n' \in ack(\mathsf{s}'(n))(\langle m, n_s, c\rangle)$$

By rule INVSSE″,

(10) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n'.$$
$$n' \in ack(\mathsf{s}'(n))(\langle m, n_s, c\rangle) \Rightarrow$$
$$\hat{\Box} n' \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle)$$

From [9] and [10]

(11) $$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, n'.$$
$$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', \langle m, n_s, c\rangle)) \Rightarrow$$
$$\hat{\Box} n' \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle)$$

From [8] and [11]

$$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n', n''.\ n' \in \mathsf{Correct} \wedge n'' \in \mathsf{Correct} \rightarrow$$
$$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$$
$$\Diamond\!\!\!-\hat{\Box} n' \in ack(\mathsf{s}(n''))(\langle m, n_s, c\rangle) \vee$$
$$\Diamond \hat{\Box} n' \in ack(\mathsf{s}(n''))(\langle m, n_s, c\rangle)$$

that is

(12) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n', n''.\ n' \in \mathsf{Correct} \wedge n'' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\Box n' \in ack(\mathsf{s}(n''))(\langle m, n_s, c\rangle)$

From [12] and [1] (instantiating n'' with n' and renaming n' to n''),

(13) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n''.\ n'' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\Box n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

that is

(14) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ (n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\Box\forall n''.\ n'' \in \mathsf{Correct} \rightarrow n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

that is

(15) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ (n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\Box\mathsf{Correct} \subseteq ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

By Definition 19 (a majority of correct nodes) on [1]

$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \ (n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\Box|ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2$

Lemma 47.

$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \Rightarrow$
$\Diamond\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \ \vee$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

Proof.

By rule InvSA with
$S = \lambda s.\ \langle m, n', c\rangle \in pending(s)$
$\mathcal{A} = (0, \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \in \mathsf{ors}$
(1) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ \langle m, n', c\rangle \in pending(\mathsf{s}(n)) \Rightarrow$
$\hat{\Diamond}\!\!-(n \bullet (0, \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \in \mathsf{ors})$

By rule ORSe
(2) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \bullet (0, \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \in \mathsf{ors}\ \Rightarrow$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

By Lemma 80 and Lemma 112 on [1] and [2]
(3) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ \langle m, n', c\rangle \in pending(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \ \vee$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

By rule IIORSe,
(4) $\Gamma \vdash_{\mathsf{URBC}} (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle)) \wedge \langle m, n', c\rangle \notin pending(\mathsf{s}(n)) \Rightarrow$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

From [3] and [4]
(5) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ [\langle m, n', c\rangle \in pending(\mathsf{s}(n))] \vee$
$[(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle)) \wedge \langle m, n', c\rangle \notin pending(\mathsf{s}(n))] \Rightarrow$
$\Diamond\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \ \vee$

$$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$$

that (after adding a conjunct) is

(6) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ [(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle)) \wedge \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n))] \vee$
$[(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle)) \wedge \langle m, n', c\rangle \notin \mathit{pending}(\mathsf{s}(n))] \Rightarrow$
$\Diamond\!\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \vee$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

that (after folding distribution of or over and) is

(7) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ [(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle))] \wedge$
$[\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \vee \langle m, n', c\rangle \notin \mathit{pending}(\mathsf{s}(n))] \Rightarrow$
$\Diamond\!\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \vee$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

that is

(8) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n, \langle m, n', c\rangle)) \Rightarrow$
$\Diamond\!\!\!-(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \vee$
$\Diamond(n \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

Lemma 48.

$\Gamma \vdash_{\mathsf{URBC}} \circledS\ n \in \mathsf{Correct} \rightarrow$
$\Diamond\Box|\mathit{ack}(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \wedge \Box c \leq \mathit{count}(\mathsf{s}(n)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))$

Proof.

We assume that

(1) $\Gamma' = \Gamma;\ n \in \mathsf{Correct}$

We show that

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Diamond\Box|\mathit{ack}(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))$

We consider two cases: $\langle m, n_s, c\rangle \in \mathit{delivered}(\mathsf{s}(n))$ and $\langle m, n_s, c\rangle \notin \mathit{delivered}(\mathsf{s}(n))$.

By Lemma 49 on

(2) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \wedge \Box c \leq \mathit{count}(\mathsf{s}(n))) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))$

By rule PeOISe

(3) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ n \bullet \top \wr\, \mathsf{per} \wedge$
$\langle m, n_s, c\rangle \in \mathit{pending}(\mathsf{s}(n)) \wedge |\mathit{ack}(\mathsf{s}(n))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2 \wedge \langle m, n_s, c\rangle \notin \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$

$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$

By rule InvSSe′

(4) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \mathit{ack}(\mathsf{s}(n))(\langle m, n_s, c\rangle \neq \varnothing \Rightarrow \langle m, n_s, c\rangle \in \mathit{pending}(\mathsf{s}(n))$

From [3] and [4]

(5) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ n \bullet \top \wr\, \mathsf{per} \wedge$
$|\mathit{ack}(\mathsf{s}(n))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2\ \wedge \langle m, n_s, c\rangle \notin \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$

$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$

By rule APerSe on [1]

(6) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \Box\Diamond(n \bullet \top \wr\, \mathsf{per})$

From [6]

(7) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Box|ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow$
$\Diamond[n \bullet \top \wr\, \mathsf{per} \wedge |ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2]$

By Lemma 87 on [7]

(8) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Diamond\Box|ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow$
$\Diamond[n \bullet \top \wr\, \mathsf{per} \wedge |ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2]$

From [8]

(9) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Diamond\Box|ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow \Diamond$
$\langle m, n', c\rangle \in delivered(\mathsf{s}(n)) \vee$
$[n \bullet \top \wr\, \mathsf{per} \wedge |ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \wedge \langle m, n', c\rangle \notin delivered(\mathsf{s}(n))]$

From [9], [2] and [5]

(10) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Diamond\Box|ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow \Diamond$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \wedge \Box c \leq count(\mathsf{s}(n))) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$

By Lemma 112 and Lemma 87 on [10]

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\ \Diamond\Box|ack(\mathsf{s}(n))(\langle m, n', c\rangle)| > |\mathbb{N}|/2 \Rightarrow$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \wedge \Box c \leq count(\mathsf{s}(n)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n, m))$

Lemma 49.

$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in delivered(\mathsf{s}(n)) \Rightarrow$
$[\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \wedge \Box c \leq count(\mathsf{s}(n)))]$

Proof.

By rule InvSASe with
n instantiated to n,
S instantiated to $\lambda s.\ \langle m, n_s, c\rangle \in delivered(s)$ and
$\mathcal{A}$ instantiated to $\mathsf{deliver}_{\mathsf{urb}}(\langle n, m\rangle)) \in \mathsf{ois} \wedge \langle m, n_s, c\rangle \in pending(\mathsf{s}(n))$

(1) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in delivered(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!\!-[n \bullet \mathsf{deliver}_{\mathsf{urb}}(\langle n_s, m\rangle)) \in \mathsf{ois} \wedge \langle m, n_s, c\rangle \in pending(\mathsf{s}(n))]$

By rule OISe,

(2) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\ (n \bullet \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \in \mathsf{ois}) \Rightarrow \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))$

From [1] and [2],

(3) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in delivered(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!\!-[\langle m, n_s, c\rangle \in pending(\mathsf{s}(n)) \wedge \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))]$

By rule InvSSe′

(4) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \ \langle m, n_s, c\rangle \in pending(\mathsf{s}(n)) \Rightarrow c \leq count(\mathsf{s}(n))$

From [3] and [4],

(5) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!\!-[c \leq \mathit{count}(\mathsf{s}(n)) \wedge \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))]$

By rule InvSSe

(6) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \ c \leq \mathit{count}(\mathsf{s}(n)) \Rightarrow (c \leq \mathit{count}(\mathsf{s}(n)))$

From [5] and [6],

(7) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$
$\Diamond\!\!\!-[(\Box c \leq \mathit{count}(\mathsf{s}(n))) \wedge \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))]$

By Lemma 115 and Lemma 112 on [7],

$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$
$[\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \wedge \Box c \leq \mathit{count}(\mathsf{s}(n))) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \wedge \Box c \leq \mathit{count}(\mathsf{s}(n)))]$

that is

$\Gamma \vdash_{\mathsf{URBC}} \circledS\ \langle m, n_s, c\rangle \in \mathit{delivered}(\mathsf{s}(n)) \Rightarrow$
$[\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \wedge \Box c \leq \mathit{count}(\mathsf{s}(n)))]$

Theorem 13. *(URB_2: No-duplication)*
If a message is broadcast at most once, it will be delivered at most once.

$$\Gamma \vdash_{\mathsf{URBC}} [n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))] \rightarrow [(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))]$$

where
Γ is defined in Definition 19.

Proof.

The proof idea: The uniform reliable broadcast for a message m by a node n is executed at most once. When a uniform reliable broadcast is executed, a best-effort broadcast request is issued. As m is broadcast from n at most once, it is broadcast with a unique counter c. By the definition of the component, a best-effort broadcast is issued only when a uniform reliable broadcast or a best-effort delivery is executed. By the no-forge property of best-effort broadcast, a best-effort delivery is executed only if the execution of a corresponding best-effort broadcast precedes it. Thus by a mutual induction, every best-effort delivery for m is executed with the initial unique counter c. A message is added to the pending set only when a best-effort delivery is executed. Thus, the message m from n is added to the pending set only with the initial unique counter c. The periodic function issues a uniform reliable delivery for a message only when it is in the pending set and not delivered before. When it issues a uniform reliable delivery for a message, it is added to the delivered set. Thus, if a uniform reliable delivery for m from n is issued at a node, it will not be issued again. Thus, a uniform reliable delivery for m from n is executed at a node at most once.

We assume

(1) $\Gamma' = \Gamma$;

$$n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m) \Rightarrow \hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$$

and prove

$$\Gamma' \vdash_{\mathsf{URBC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow \hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))$$

By rule OI′

(2) $\Gamma \vdash_{\mathsf{URBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \Rightarrow \Diamond(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois} \wedge \mathsf{self})$

From Lemma 57 and Lemma 108,

(3) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\,[n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois} \Rightarrow$
$\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leq 1 \wedge$
$\hat{\Box}\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \wedge$
$\hat{\boxminus}\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$

From rule UniOISe,

(4) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\,(\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leq 1 \wedge$
$\hat{\Box}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \notin \mathsf{ois}) \wedge$
$\hat{\boxminus}(\mathsf{n} = n \rightarrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \notin \mathsf{ois})) \Rightarrow$
$n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m))$

From [3] and [4]

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow \Diamond[$
$n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)]$

That is,

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)$

By rule SInv

$\Gamma' \vdash_{\mathsf{URBC}} n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m) \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)$

Lemma 50.

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\,[\forall n, n', n'', m, c.$
$\Box(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

Proof.

Using Lemma 111 with

$\mathcal{A} = \circledS\,[\forall n', n'', m, c.$
$(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]$

$\mathcal{A}' = \circledS\,[\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

and then Lemma 51 and Lemma 52.

Lemma 51.

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\,[\hat{\boxminus}[\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

$$
\begin{array}{l}
\rightarrow \\
[\forall n', n'', m, c. \\
\ (n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow \\
\ \Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]
\end{array}
$$

Proof.

Let

(1) $\Gamma'' = \Gamma'$;

$\circledS$ $[\hat{\boxminus}[\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$

We prove that

$\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [\forall n', n'', m, c.$
$(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

From [1]

(2) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [\hat{\boxminus}[\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$

By rule OISE′,

(3) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\hat{\Diamond}(n'' \bullet (0, \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \in \mathsf{ors})]$

By rule INVLSE,

(4) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [(n'' \bullet \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle) \in \mathsf{ois}) \Rightarrow$
$[(n'' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n'')) = c - 1 \wedge n'' = n'] \vee$
$\exists n'''.\ (n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n''', \langle m, n', c\rangle))]$

From [4],

(5) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [\hat{\Diamond}(n'' \bullet \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle) \in \mathsf{ois}) \Rightarrow$
$\hat{\Diamond}[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1] \vee$
$\hat{\Diamond}\exists n'''.\ (n'' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n''', \langle m, n', c\rangle))]$

From [5] and [2]

(6) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [\hat{\Diamond}(n'' \bullet \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle) \in \mathsf{ois}) \Rightarrow$
$\hat{\Diamond}[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1] \vee$
$\hat{\Diamond}\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

that is

(7) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [\hat{\Diamond}(n'' \bullet \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle) \in \mathsf{ois}) \Rightarrow$
$\hat{\Diamond}[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]$

From [3] and [7]

(8) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\ [(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\hat{\Diamond}[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]$

Thus,

$\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]$

Lemma 52.

$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\hat{\boxminus}[\forall n', n'', m, c.$
$(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$
$\rightarrow$
$[\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$

Proof.

Let

(1) $\Gamma'' = \Gamma'$;
$\circledS\, [\hat{\boxminus}[\forall n', n'', m, c.$
$(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$

We prove that

$\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

From [1]

(2) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [\forall n', n'', m, c.$
$\hat{\boxminus}(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

From Γ, BEB_3 and rule SInv, we have

(3) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \Rightarrow$
$\hat{\Diamond}(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle))$

From [3] and [2]

(4) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\hat{\Diamond}\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

that is

(5) $\Gamma'' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

Lemma 53.
$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', m, c.$
$\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

Proof.
By rule InvSASe
(1) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow \hat{\Diamond}[$
$((n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1 \wedge n' = n) \vee$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle))]]$
By Lemma 50,
(2) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n', c\rangle)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$
From [1] and [2]
(3) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow \hat{\Diamond}[$
$((n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1) \vee$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$
thus
(4) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

Lemma 54.
$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, c, c'.\ c \neq c' \rightarrow$
$\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\boxminus \langle m, n', c'\rangle \notin \mathit{pending}(\mathsf{s}(n)) \ \wedge$
$\Box\langle m, n', c'\rangle \notin \mathit{pending}(\mathsf{s}(n))]$

Proof.
We show the contra-positive of the first conjunct.
$\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [c \neq c' \rightarrow$
$\Diamond\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\neg\langle m, n', c\rangle \notin \mathit{pending}(\mathsf{s}(n))]$

By Lemma 53,
(1) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\forall n, n', n'', m, c.$
$\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$
From [1],
(2) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\Diamond\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\Diamond\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c' - 1]]$
that is
(3) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, [\Diamond\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\Diamond[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c' - 1]]$

From the definition of Γ',

(4) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \Rightarrow$
$\hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))]$

From [4] and Lemma 108

(5) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c' - 1 \Rightarrow$
$\hat{\boxminus}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge$
$\hat{\Box}\neg(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))]$

thus

(6) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[c \neq c' \rightarrow$
$(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c' - 1 \Rightarrow$
$\boxminus\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1] \wedge$
$\Box\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

From [3] and [6]

(7) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[c \neq c' \rightarrow$
$\diamondminus\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\diamondminus[\boxminus\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1] \wedge$
$\Box\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]]$

thus

(8) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[c \neq c' \rightarrow$
$\diamondminus\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\boxminus\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1]]$

From the contra-positive of [1]

(9) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\boxminus\neg[(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m)) \wedge \mathit{count}(\mathsf{s}(n')) = c - 1] \Rightarrow$
$\neg\langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n))]$

From [8] and [9]

(10) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[c \neq c' \rightarrow$
$\diamondminus\langle m, n', c'\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$\neg\langle m, n', c\rangle \notin \mathit{pending}(\mathsf{s}(n))]$

Proof of the second conjunct is similar.

Lemma 55.
$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\forall n, n', m, c.$
$\Box\mathsf{occ}(\mathit{pending}(\mathsf{s}(n), \langle m, n', c\rangle) \leq 1]$

Proof.
By rule InvSSe′,

(1) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\ \forall n, m, c.$
$\langle m, n, c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow$
$c \leq \mathit{count}(\mathsf{s}(n))$

From the definition of request and [1]

(2) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\forall n', m, c.$

$(\mathsf{s}'(\mathsf{n}), \mathsf{ors}, \mathsf{ois}) = \mathsf{request}_c(\mathsf{n}, \mathsf{s}(\mathsf{n}), e) \wedge \mathsf{occ}(\mathit{pending}(\mathsf{s}(\mathsf{n}), \langle m, n', c\rangle) \leq 1 \Rightarrow$
$\mathsf{occ}(\mathit{pending}(\mathsf{s}'(\mathsf{n}), \langle m, n', c\rangle) \leq 1]$

By rule InvUSSe′
$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [\forall n, n', m, c.$
$\Box\mathsf{occ}(\mathit{pending}(\mathsf{s}(n), \langle m, n', c\rangle) \leq 1]$
The request case is from [2]. The indication and periodic cases are trivial.

Lemma 56.
$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [\forall n, n', m$
$\Box\, |\{\langle m, n'\rangle \mid \exists c.\ \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n))\}| \leq 1]$

Proof.
Immediate from Lemma 54 and Lemma 55.

Lemma 57.
The uniform-reliable broadcast delivery event is issued at most once at every node.
$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois} \Rightarrow$
$\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leq 1 \wedge$
$\hat{\boxminus}\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$

Proof.

By rule InvLSe
(1) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$(\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(n))]$

From the definition of periodic and Lemma 56
(2) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [(\mathsf{s}'(n), \mathsf{ors}, \mathsf{ois}) = \mathsf{periodic}_c(\mathsf{n}, \mathsf{s}(n)) \Rightarrow$
$\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leq 1]$

From [1] and [2]
$\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\mathsf{occ}(\mathsf{ois}, \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leq 1]$

We show the contra-positive of
(3) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois} \Rightarrow$
$\hat{\boxminus}\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$
that is
(4) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{\mathsf{S}} [\hat{\Diamond}(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$

By rule InvLSe

(5) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\exists c.\ \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \wedge$
$\langle m, n', c\rangle \notin \mathit{delivered}(\mathsf{s}(n)) \wedge$
$\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}'(n))]$

From the contra-positive of [5] is

(6) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\forall c.\ \langle m, n', c\rangle \notin \mathit{pending}(\mathsf{s}(n)) \vee$
$\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))] \Rightarrow$
$\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$

By rule InvSSe″

(7) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}'(n)) \Rightarrow$
$\hat{\Box}\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))]$

From [5] and [7]

(8) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})] \Rightarrow$
$\exists c.\ \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \wedge$
$\hat{\Box}\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))]$

From [8]

(9) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\hat{\Diamond}(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\hat{\Diamond}[\exists c.\ \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \wedge$
$\hat{\Box}\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))]]$

From [9] and Lemma 54

(10) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\hat{\Diamond}(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\hat{\Diamond}[\exists c.\ \forall c'.\ c \neq c' \rightarrow \Box\langle m, n', c'\rangle \notin \mathit{pending}(\mathsf{s}(n)) \wedge$
$\hat{\Box}\langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))]]$

thus

(11) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\hat{\Diamond}(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\exists c.\ \forall c'.\ c \neq c' \rightarrow$
$\langle m, n', c'\rangle \notin \mathit{pending}(\mathsf{s}(n)) \wedge \langle m, n', c\rangle \in \mathit{delivered}(\mathsf{s}(n))]$

From [11] and [6]

(12) $\Gamma' \vdash_{\mathsf{URBC}} \textcircled{S}\,[\hat{\Diamond}(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow$
$\neg(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})]$

Theorem 14. *(URB$_3$: No-forge)*
If a node delivers a message m with sender n', then m was previously broadcast by node n'.

$\Gamma \vdash_{\mathsf{URBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \leftsquigarrow (n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$
where
Γ is defined in Definition 19.

Proof.

The proof idea: A uniform-reliable delivery event is issued only in the periodic function. The periodic function only issues messages from the pending set. As shown in the proof of the no-forge property, any message that is put in the pending set is previously broadcast by the uniform-reliable broadcast.

By rule OISE$'$,
(1) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \Rightarrow \diamondminus(n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois})$

By rule InvLSe
(2) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n \bullet \mathsf{deliver}_{\mathsf{urb}}(n', m) \in \mathsf{ois}) \Rightarrow \exists c.\ \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n))$
By Lemma 53
(3) $\Gamma' \vdash_{\mathsf{URBC}} \circledS\, \langle m, n', c\rangle \in \mathit{pending}(\mathsf{s}(n)) \Rightarrow \diamondminus(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$
From [1], [2] and [3]
(4) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \Rightarrow \diamondminus(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$
From [4], and rule SInv,
$\Gamma \vdash_{\mathsf{URBC}} (n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n', m)) \Rightarrow \diamondminus(n' \bullet \top \downarrow \mathsf{broadcast}_{\mathsf{urb}}(m))$

Theorem 15. *(URB_4: Uniform Agreement)*
If a message m is delivered by some node (whether correct or faulty),
then m is eventually (in the past or future) delivered by every correct node.

$$\Gamma \vdash_{\mathsf{URBC}} \forall n, n', n_s.\ n \in \mathsf{Correct} \rightarrow (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee \Diamond\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))$$

where
Γ is defined in Definition 19.

Proof.

The proof idea: A node has delivered the message. The protocol delivers a message only in the periodic function when acknowledgement from a majority of nodes is received for the message. As a majority of nodes are correct, there is at least one correct node in the acknowledging set. A node is added to the acknowledging set only if a message is received (via best-effort broadcast) from it. By the no-forge property of the best-effort broadcast, the node should have broadcast the message. Thus, a correct node has broadcast the message. By the validity property of the best-effort broadcast, every correct node delivers the message. When a message is delivered (via best-effort broadcast), if it is not already in the pending set, it is rebroadcast. If it is already in the pending set, it is already broadcast. Thus, every correct node eventually (in the past or future) broadcasts the message. Thus, by the validity property of the best-effort broadcast, the message is delivered to every correct node from every correct node (via best-effort broadcast). When a message is delivered (via the best-effort broadcast), the sender is added to the acknowledgement set and always remains in it. Thus, eventually forever, every correct node will have every correct node in its acknowledgement set for the message. As at least a majority of nodes are correct, the size of this set is more than half of the number of the nodes. As the periodic function is infinitely often called, it will be eventually called when the size of the acknowledgement set for the message is more than half of the number of the nodes. The protocol maintains the invariant that every message that is in an acknowledgement set is in the pending set. When the periodic function iterates the pending set, if the message is not in the delivered set, as its acknowledgement set is already large enough, it is delivered. On the other hand, if it is in the delivered set, it is already delivered. Thus, the message is eventually (in the past or future) delivered at every correct node.

We assume that
(1) $\Gamma' = \Gamma; n \in \mathsf{Correct}$
We show that
$$\Gamma \vdash_{\mathsf{URBC}} (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow \Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee \Diamond\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))$$

By rule OISE′,
(2) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow \Diamond\!\!-(n' \bullet \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \in \mathsf{ois})$
By rule INVLSE
(3) $\Gamma \vdash_{\mathsf{URBC}} \circledS\ (n' \bullet \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \in \mathsf{ois}) \Rightarrow \exists c.\ |ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)| > |\mathbb{N}|/2$

By Lemma 58

(4) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, \Box ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle) \subseteq \mathbb{N}$

It is obvious that

(5) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, \Box|\mathbb{N}|/2 + |\mathbb{N}|/2 \geq |\mathbb{N}|$

By rule Quorum on Definition 19 (correct majority), [4] and [5] on [3]

(6) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n' \bullet \mathsf{deliver}_{\mathsf{urb}}(n_s, m) \in \mathsf{ois}) \Rightarrow$
$\exists c, n''.\ n'' \in \mathsf{Correct} \wedge n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

By Lemma 99 on [2] and [6]

(7) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow$
$\Diamond\!\!\!-\exists c, n''.\ n'' \in \mathsf{Correct} \wedge n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

By rule InvSASe with

$S = n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)$

$\mathcal{A} = n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle)$

(8) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\!\!\!-(n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle))$

By Definition 19 (BEB'_3) and rule SInv,

(9) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n' \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'', \langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\!\!\!-(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle))$

By Lemma 88 on [8] and [9],

(10) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n'' \in ack(\mathsf{s}(n'))(\langle m, n_s, c\rangle)) \Rightarrow$
$\Diamond\!\!\!-(n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle))$

By Lemma 88 on [7] and [10],

(11) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow$
$\Diamond\!\!\!-\exists c, n''.\ n'' \in \mathsf{Correct} \wedge (n'' \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n_s, c\rangle))$

By Lemma 45 on [11] and [1], and then Lemma 112

(12) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))$

By rule SInv on [12]

$\Gamma \vdash_{\mathsf{URBC}} (n' \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \Rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m)) \vee$
$\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{deliver}_{\mathsf{urb}}(n_s, m))$

Lemma 58.
$\Gamma \vdash_{\mathsf{URBC}} \circledS\, \Box ack(\mathsf{s}(n))(\langle m, n_s, c\rangle) \subseteq \mathbb{N}$

Proof.
We show that

$\Gamma \vdash_{\mathsf{URBC}} \circledS\, \forall n.\ n \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle) \Rightarrow n \in \mathbb{N}$

By rule InvSASe

(1) $\Gamma \vdash_{\mathsf{URBC}} \circledS\, n \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle) \Rightarrow \Diamond\!\!\!-(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_s, \langle m, n', c\rangle))$

By Definition 19 (BEB'_3)

(2) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_s, \langle m, n', c\rangle)) \Rightarrow \Diamond(n_s \bullet 0 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle m, n', c\rangle)))$
By Lemma 88 on [1], [2] and rule NODE,
(3) $\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle) \Rightarrow \Diamond n \in \mathbb{N}$
that is (n is rigid.)
$\Gamma \vdash_{\mathsf{URBC}} \textcircled{S}\ n \in ack(\mathsf{s}(n))(\langle m, n_s, c\rangle) \Rightarrow n \in \mathbb{N}$

5.3.5 Eventually Perfect Failure Detector

Theorem 16. *($EPFD_1$: Strong Completeness)*
Every incorrect node is eventually permanently suspected by every correct node.
$\forall n, n'.\ n \in \mathsf{Correct} \wedge \neg n' \in \mathsf{Correct} \rightarrow$
$\quad \Diamond[(n \bullet \top \uparrow \mathsf{suspect}(n')) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{restore}(n'))]$.

Proof.

By the rule APER, the periodic handler of an incorrect node n' is executed only finitely often. So the periodic handler of n' is eventually never executed. By the rule INVL, only the periodic handler sends heartbeat messages. Therefore, n' will eventually never send heartbeat messages. By the rule NFORGE, a message is received only if it is sent. Thus, no node will eventually receive heartbeat messages from n'. By the rule INVL, the node n' is added to the *active* set of the node n only when n receives a heartbeat message from n'. Therefore, eventually after a round r, the node n' will be absent in $active(r)$ of the node n. Therefore, in round $r+1$, the periodic handler of n issues a suspect event for n', if n' is not in the $failed$ set. If it is in the $failed$ set, it is already suspected and not restored since. From round $r+1$, the node n' will be never restored. This is because by the rule INVL, the restore indication for n' is issued only when it is found in the *active* set.

Theorem 17. *($EPFD_2$: Eventual Strong Accuracy)*
Eventually no correct node is suspected by any correct node.
$\forall n, n'.\ n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad \Box\neg(n \bullet \top \uparrow \mathsf{suspect}(n')) \vee$
$\quad \Diamond[(n \bullet \top \uparrow \mathsf{restore}(n')) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{suspect}(n'))]$.

Proof.

In the periodic handler, every correct node n' sends a heartbeat message to every node in every round r. By the rule GST, after the round global stabilization time (GST), every correct node n delivers every message set to it in the same round. On the receipt of the heartbeat message, the indication handler adds the sender n' to the *active* set of n. By the rule RSEQ, the periodic event per of round $r+1$ is executed after the indication events $\downarrow$ of round r. By the rule INVS, the node n' remains in the *active* set of n. When the periodic event per of round $r+1$ executes, the node n finds n' in *active* set. The node n restores n' if it is suspected before and does not suspect n'. By the rule INVL, suspecting nodes is only issued in the periodic handler. In the periodic handler, the node n' is always found in the *active* set and hence is never suspected again.

5.3.6 Eventual Leader Elector

Theorem 18. *(ELE_1 (Eventual Leadership)*
Eventually every correct process trusts the same correct process.
$\exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{trust}(l) \wedge \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{trust}(l')))]$.

Proof.

Immediate from the two properties EPFD_1 and EPFD_2. Eventually every node will have every incorrect node in the suspected set and every correct node not in the suspected set: the suspected set will be the set of incorrect nodes. The periodic event after the latest such event applies the *maxRank* funciton to the set of correct nodes and deterministically chooses the leader.

5.3.7 Epoch Consensus

Definition 20.

Γ =
$|\mathsf{Correct}| > \mathbb{N}/2;$
$\mathsf{PL}'_1, \mathsf{PL}'_2, \mathsf{PL}'_3, \mathsf{BEB}'_1; \mathsf{BEB}'_2; \mathsf{BEB}'_3$

PL'_1 = $\mathsf{lower}(0, \mathsf{PL}_1)$ =
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow$
$n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

PL'_2 = $\mathsf{lower}(0, \mathsf{PL}_2)$ =
$[n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m) \Rightarrow$
$\hat{\boxminus} \neg (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))] \rightarrow$
$[n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))]$

PL'_3 = $\mathsf{lower}(0, \mathsf{PL}_3)$ =
$(n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \leftsquigarrow$
$(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))$

BEB'_1 = $\mathsf{lower}(1, \mathsf{BEB}_1)$ =
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \rightsquigarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

BEB'_2 = $\mathsf{lower}(1, \mathsf{BEB}_2)$ =
$[n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m) \Rightarrow$
$\hat{\boxminus} \neg (n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))] \rightarrow$
$[n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m) \Rightarrow$
$\hat{\boxminus} \neg (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$

BEB'_3 = $\mathsf{lower}(1, \mathsf{BEB}_3)$ =
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow$
$(n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

Cons =
$(n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n_l)) \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n'_l)) \Rightarrow$
$n_l = n'_l$

Theorem 19. *(EC_1: Validity)*
If a node decides the value v, then v was proposed by the current leader n_l or is passed to a node during initialization.
$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\Diamond(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$
where
Γ is defined in Definition 20.

Proof.
We assume
(1) $\Gamma' = \Gamma; v \neq \bot$
We show
$\Gamma' \vdash_{\mathsf{ECC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$
By Lemma 59,
(2) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow \Diamond(wval(\mathsf{s}(n_l)) = v \wedge v \neq \bot)$
By InvSASe with,
n instantiated to n_l,
S instantiated to $\lambda s.\ wval(s) \neq \bot$ and
$\mathcal{A}$ instantiated to $wval(\mathsf{s}'(n_l)) = prop(\mathsf{s}(n_l)) \vee \exists n'.\ (\_, wval(\mathsf{s}'(n_l))) = states(\mathsf{s}'(n_l))(n')$
(3) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ wval(\mathsf{s}(n_l)) \neq \bot \Rightarrow \hat{\Diamond}(wval(\mathsf{s}'(n_l)) = prop(\mathsf{s}(n_l)) \vee \exists n'.\ (\_, wval(\mathsf{s}'(n_l))) = states(\mathsf{s}'(n_l))(n'))$
From [2] and [3] and Lemma 99 and Lemma 86
(4) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow \hat{\Diamond}(v = prop(\mathsf{s}(n_l)) \vee \exists n'.\ (\_, v) = states(\mathsf{s}'(n_l))(n')) \wedge v \neq \bot)$
By PostPre on [5]
(5) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow \hat{\Diamond}(v = prop(\mathsf{s}(n_l)) \vee \bigcirc \exists n'.\ (\_, v) = states(\mathsf{s}(n_l))(n')) \wedge v \neq \bot)$

By InvSASe with,
n instantiated to n_l,
S instantiated to $\lambda s.\ prop(s) = v \wedge prop(s) \neq \bot$ and
$\mathcal{A}$ instantiated to $\mathsf{propose}_{\mathsf{ec}}(v)$
(6) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ prop(\mathsf{s}(n_l)) = v \wedge prop(\mathsf{s}(n_l)) \neq \bot \Rightarrow \hat{\Diamond}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

From [5], [6] and Lemma 60,
(7) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow \hat{\Diamond}(\Diamond(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \vee \bigcirc [\exists n'.\ \Diamond(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)))]$

By Lemma 86 and Lemma 121 on [7]
(8) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$

$$\diamondsuit(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$$

Lemma 59.

$\Gamma \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\quad \diamondsuit[|states(\mathsf{s}(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}(n_l)) = v \wedge v \neq \bot \wedge$
$\quad highest(states(\mathsf{s}(n_l))) \neq \bot \rightarrow wval(\mathsf{s}(n_l)) = highest(states(\mathsf{s}(n_l)))]$

By OISE′,

(9) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\quad \hat{\diamondsuit}(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \in \mathsf{ois})$

By INVLSE,

(10) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \in \mathsf{ois}) \Rightarrow$
$\quad (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(\langle ts, v \rangle)))$

From Γ'' and BEB'_3, we have,

(11) $\Gamma' \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(\langle ts, v \rangle))) \Rightarrow$
$\quad \diamondsuit(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(\langle ts, v \rangle)))$

By ORSE′,

(12) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(\langle ts, v \rangle))) \Rightarrow$
$\quad \hat{\diamondsuit}(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(\langle ts, v \rangle))) \in \mathsf{ors})$

From [10], [11], and [12] and SINV

(13) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \in \mathsf{ois}) \Rightarrow$
$\quad \hat{\diamondsuit}(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(\langle ts, v \rangle))) \in \mathsf{ors})$

By INVLSE

(14) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(\langle ts, v \rangle))) \in \mathsf{ors}) \Rightarrow$
$\quad (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}(ts))) \wedge$
$\quad (wval(\mathsf{s}(n_l)) = v)$

From Γ', PL'_3 and SINV, we have,

(15) $\Gamma' \vdash_{\mathsf{ECC}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}(ts))) \Rightarrow$
$\quad \diamondsuit(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts)))$

By ORSE′,

(16) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts))) \Rightarrow$
$\quad \hat{\diamondsuit}(n_l \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts))) \in \mathsf{ors})$

By INVLSE

(17) $\Gamma' \vdash_{\mathsf{ECC}} (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts)) \in \mathsf{ors}) \Rightarrow$
$\quad \exists v'.\ (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(\langle ts, v' \rangle)))$

From Γ', BEB'_3 and SINV, we have,

(18) $\Gamma' \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(\langle ts, v' \rangle))) \Rightarrow$
$\quad \diamondsuit(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v' \rangle)))$

By ORSE′,

(19) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v' \rangle))) \Rightarrow$
$\quad \hat{\diamondsuit}(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v' \rangle))) \in \mathsf{ors})$

By INVLSE

(20) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v' \rangle))) \in \mathsf{ors}) \Rightarrow$
$\quad |states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \wedge$
$\quad highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))$

By Lemma 88, Lemma 99 and Lemma 86 on [14] and [13] to [20]

(21) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\quad \hat{\diamondsuit}[(wval(\mathsf{s}'(n_l)) = v) \wedge$

$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))]]$$

By InvSSe″

(22) $\Gamma' \vdash_{\mathsf{ECC}} wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \Rightarrow$
$$\hat{\Box} wval(\mathsf{s}(n_l)) = v'$$

From [21] and [22]

(23) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\hat{\Diamond}[(wval(\mathsf{s}(n_l)) = v) \wedge$$
$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))] \wedge$$
$$\hat{\Diamond}\hat{\Box} wval(\mathsf{s}(n_l)) = v']$$

that is

$\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\hat{\Diamond}[(wval(\mathsf{s}(n_l)) = v) \wedge$$
$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))] \wedge$$
$$wval(\mathsf{s}(n_l)) = v']$$

that is

$\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\hat{\Diamond}[v = v' \wedge$$
$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v' \wedge v' \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))]]$$

that is

$\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\hat{\Diamond}[$$
$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v \wedge v \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))]]$$

that by Lemma 86 is

(24) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\hat{\Diamond}[|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l)) = v \wedge v \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l)) = highest(states(\mathsf{s}'(n_l)))]$$

By PostPre and Lemma 121

(25) $\Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$$\Diamond[|states(\mathsf{s}(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}(n_l)) = v \wedge v \neq \bot \wedge$$
$$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}(n_l)) = highest(states(\mathsf{s}(n_l)))]$$

Lemma 60.

$\Gamma \vdash_{\mathsf{ECC}} \circledS\ [(val(\mathsf{s}(n)) = v) \vee$
$$(states(\mathsf{s}(n_l))(n') = (\_, v))] \Rightarrow$$
$$\Diamond(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$$

Proof.

By Lemma 110 on Lemma 61 and Lemma 62.

Lemma 61.
$\Gamma \vdash_{\mathsf{ECC}} \circledS$
$\quad \hat{\boxminus}[(states(\mathsf{s}(n_l))(n') = (\_, v)) \rightarrow$
$\qquad \hat{\diamondminus}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))]$
$\quad \Rightarrow$
$\quad (val(\mathsf{s}(n)) = v) \Rightarrow$
$\qquad \hat{\diamondminus}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

Proof.
By InvSASe with,
$\quad n$ instantiated to n,
$\quad S$ instantiated to $\lambda s.\ val(s) = v$ and
$\quad \mathcal{A}$ instantiated to $\mathsf{propose}_{\mathsf{ec}}(v)$
$\quad (1)\ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ \ (val(\mathsf{s}(n)) = v) \Rightarrow$
$\qquad\qquad \hat{\diamondminus}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(\langle ts, v\rangle)))$
From Γ' and BEB'_3, we have,
$\quad (2)\ \ \Gamma' \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(\langle ts, v\rangle))) \Rightarrow$
$\qquad\qquad \hat{\diamondminus}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v\rangle)))$
By Lemma 99 and Lemma 86 on [1] and [2] and SInv
$\quad (3)\ \ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ (val(\mathsf{s}(n)) = v) \Rightarrow$
$\qquad\qquad \hat{\diamondminus}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v\rangle)))$
By ORSe′,
$\quad (4)\ \ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v\rangle))) \Rightarrow$
$\qquad\qquad \hat{\diamondminus}(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v\rangle))) \in \mathsf{ors})$
By InvLSe,
$\quad (5)\ \ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(\langle ts, v\rangle))) \in \mathsf{ors}) \Rightarrow$
$\qquad\qquad (v = prop(\mathsf{s}(n_l)) \wedge v \neq \bot) \vee$
$\qquad\qquad \exists n'.\ (\_, v) = states(\mathsf{s}(n_l))(n')$
By InvSASe with,
$\quad n$ instantiated to n_l,
$\quad S$ instantiated to $\lambda s.\ prop(s) = v \wedge prop(s) \neq \bot$ and
$\quad \mathcal{A}$ instantiated to $\mathsf{propose}_{\mathsf{ec}}(v)$
$\quad (6)\ \ \Gamma \vdash_{\mathsf{ECC}} \circledS\ prop(\mathsf{s}(n_l)) = v \wedge prop(\mathsf{s}(n_l)) \neq \bot \Rightarrow$
$\qquad\qquad \hat{\diamondminus}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$
By Lemma 88, Lemma 99 and Lemma 86 on [3] to [6]
$\quad (7)\ \ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ (val(\mathsf{s}(n)) = v) \Rightarrow \hat{\diamondminus}$
$\qquad\qquad [(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \vee$
$\qquad\qquad \exists n'.\ (\_, v) = states(\mathsf{s}(n_l))(n')]$
After adding a premise,
$\quad (8)\ \ \Gamma' \vdash_{\mathsf{ECC}} \circledS\ \ \hat{\boxminus}[(states(\mathsf{s}(n_l))(n') = (\_, v)) \rightarrow$
$\qquad\qquad\qquad \hat{\diamondminus}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))] \Rightarrow$
$\qquad\qquad (val(\mathsf{s}(n)) = v) \Rightarrow$
$\qquad\qquad\qquad \hat{\diamondminus}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \vee$
$\qquad\qquad\qquad \hat{\diamondminus}\exists n'.\ (\_, v) = states(\mathsf{s}(n_l))(n')$

After applying the premise in the conclusion,
that is replacing
$(states(\mathsf{s}(n_l))(n') = (\_, v))$
with
$\Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))$
we have
(9) $\Gamma' \vdash_{\mathsf{ECC}} \circledS \;\hat{\boxminus}[(states(\mathsf{s}(n_l))(n') = (\_, v)) \rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))] \Rightarrow (val(\mathsf{s}(n)) = v) \Rightarrow \hat{\Diamond}(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v)) \vee \hat{\Diamond}\Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))$
that is
(10) $\Gamma' \vdash_{\mathsf{ECC}} \circledS \;\hat{\boxminus}[(states(\mathsf{s}(n_l))(n') = (\_, v)) \rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))] \Rightarrow (val(\mathsf{s}(n)) = v) \Rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))$

Lemma 62.
$\Gamma \vdash_{\mathsf{ECC}} \circledS$
$\hat{\boxminus}[(val(\mathsf{s}(n)) = v) \rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))]$
$\Rightarrow$
$(states(\mathsf{s}(n_l))(n') = (\_, v)) \Rightarrow \Diamond(n_l \bullet \top \downarrow \mathsf{propose_{ec}}(v))$

Proof.
By InvSASe with,
n instantiated to n_l,
S instantiated to $\lambda s.\ states(s)(n') = (vts, v)$ and
$\mathcal{A}$ instantiated to $(n_l \bullet 0 \uparrow \mathsf{deliver_{pl}}(n, \textsc{State}\langle vts, v\rangle)$
(1) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ states(\mathsf{s}(n_l))(n') = (vts, v) \Rightarrow \hat{\Diamond}(n_l \bullet 0 \uparrow \mathsf{deliver_{pl}}(n', \textsc{State}\langle vts, v\rangle))$
From Γ' and PL_3 we have,
(2) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ (n_l \bullet 0 \uparrow \mathsf{deliver_{pl}}(n', \textsc{State}\langle vts, v\rangle)) \Rightarrow \Diamond(n' \bullet 0 \downarrow \mathsf{send_{pl}}(n_l, \textsc{State}\langle vts, v\rangle))$
By Lemma 99 and Lemma 86 on [1] and [2] and SInv
(3) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ states(\mathsf{s}(n_l))(n') = (vts, v) \Rightarrow \Diamond(n' \bullet 0 \downarrow \mathsf{send_{pl}}(n_l, \textsc{State}\langle vts, v\rangle))$
By ORSe' on [3],
(4) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ (n' \bullet 0 \downarrow \mathsf{send_{pl}}(n_l, \textsc{State}\langle vts, v\rangle)) \Rightarrow \hat{\Diamond}(n' \bullet (0, \mathsf{send_{pl}}(n_l, \textsc{State}\langle vts, v\rangle)) \in \mathsf{ors})$
By InvLSe,
(5) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ (n' \bullet (0, \mathsf{send_{pl}}(n_l, \textsc{State}\langle vts, v\rangle)) \in \mathsf{ors}) \Rightarrow val(\mathsf{s}(n')) = v$
By Lemma 88, Lemma 99 and Lemma 86 on [3] to [5]
(6) $\Gamma \vdash_{\mathsf{ECC}} \circledS\ states(\mathsf{s}(n_l))(n') = (vts, v) \Rightarrow$

$$\hat{\Diamond} val(\mathsf{s}(n')) = v$$

After adding a premise,

(7) $\Gamma' \vdash_{\mathsf{ECC}} \circledS \;\; \hat{\Box}[(val(\mathsf{s}(n)) = v) \rightarrow \hat{\Diamond}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))] \Rightarrow states(\mathsf{s}(n_l))(n') = (vts, v) \Rightarrow \hat{\Diamond} val(\mathsf{s}(n')) = v$

After applying the premise in the conclusion,

(8) $\Gamma' \vdash_{\mathsf{ECC}} \circledS \;\; \hat{\Box}[(val(\mathsf{s}(n)) = v) \rightarrow \hat{\Diamond}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))] \Rightarrow states(\mathsf{s}(n_l))(n') = (vts, v) \Rightarrow \hat{\Diamond}(n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

Theorem 20. *(EC_2: Uniform Agreement)*
No two nodes decide differently.
$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \Rightarrow$
$v = v'$
where
Γ is defined in Definition 20.

Proof.
We prove the equivalent formula
(1) $\Gamma \vdash_{\mathsf{ECC}}$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge$
$(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \Rightarrow$
$v = v'$
By OI′,
(2) $\Gamma' \vdash_{\mathsf{ECC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Diamond}(n \bullet \mathsf{decide}_{\mathsf{ec}}(v) \in \mathsf{ois} \wedge \mathsf{self})$
By InvL,
(3) $\Gamma' \vdash_{\mathsf{ECC}} (\mathsf{self} \wedge n \bullet \mathsf{decide}_{\mathsf{ec}}(v) \in \mathsf{ois}) \Rightarrow$
$\exists ts, n_l.\ (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v)))$
From [2] and [3], we have
(4) $\Gamma \vdash_{\mathsf{ECC}} [\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge$
$(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \Rightarrow$
$\exists ts_1, n_{l1}, ts_2, n_{l2}.$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_{l1}, \textsc{Decided}(ts_1, v))) \wedge$
$\Diamond(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_{l2}, \textsc{Decided}(ts_2, v')))$
Without loss of generality, we assume that
(5) $\Gamma \vdash_{\mathsf{ECC}} ts_1 \leq ts_2$
From Lemma 63 on [5], we have
(6) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_{l1}, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_{l2}, \textsc{Decided}(ts_2, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_{l2}, \textsc{Decided}(ts_2, v'))) \rightarrow v' = v]$
From [4] and [6], we have
(7) $\Gamma \vdash_{\mathsf{ECC}} [\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge$
$(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \Rightarrow$
$v = v'$

Lemma 63.
$\Gamma \vdash_{\mathsf{ECC}}\ ts_2 \geq ts_1 \rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_2, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_2, v'))) \rightarrow v' = v]$
where
Γ is defined in Definition 20.

Proof.

Immediate from Lemma 64 and Lemma 68.

Lemma 64.
$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \leftsquigarrow$
$\exists n'.\ (n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v)))$
where
Γ is defined in Definition 20.

Proof.
By BEB'_3,
(1) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v)))$
By OR',
(2) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$\Diamond(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self})$
By InvL,
(3) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$\exists n'.\ (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}(ts)))$
By PL'_3,
(4) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}(ts))) \Rightarrow$
$\Diamond(n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts)))$
By OR',
(5) $\Gamma \vdash_{\mathsf{ECC}} (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts))) \Rightarrow$
$\Diamond(n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts))) \in \mathsf{ors} \wedge \mathsf{self})$
By InvL,
(6) $\Gamma \vdash_{\mathsf{ECC}} (n' \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$\exists v', n'_l.\ (n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'_l, \textsc{Accept}(ts, v')))$
From [1]-[6]
(7) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \Rightarrow$
$\exists v', n', n'_l.\ \Diamond(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n'_l, \textsc{Accept}(ts, v')))$
From [7], Lemma 65 and Lemma 66
(8) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \leftsquigarrow$
$\exists n'.\ (n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v)))$

Lemma 65.
$\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$(n'_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$n_l = n'_l$
where
Γ is defined in Definition 20.

Proof.
By a chain use of OR', InvL, BEB'_3, and PL'_3,

(1) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$\exists n.\ \Diamond(n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n))$

(2) $\Gamma \vdash_{\mathsf{ECC}}$
$(n'_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$\exists n.\ \Diamond(n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n))$

From Definition 20,

(3) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n_l)) \Rightarrow$
$ts \neq \bot$

From [1], [2], [3]

(4) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$ts \neq \bot$

(5) $\Gamma \vdash_{\mathsf{ECC}} (n'_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$ts \neq \bot$

By IR,

(6) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$ts = ets(\mathsf{s}(n_l))$

(7) $\Gamma \vdash_{\mathsf{ECC}}$
$(n'_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$ts = ets(\mathsf{s}(n'_l))$

By InvSA,

(8) $\Gamma \vdash_{\mathsf{ECC}}$
$ets(\mathsf{s}(n_l)) = ts \wedge ts \neq \bot \wedge \mathsf{self} \Rightarrow$
$\Diamond(n_l \bullet \top \downarrow \mathsf{epoch}(n_l, ts))$

From Definition 20,

(9) $\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n_l)) \Rightarrow$
$(n' \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(ts, n'_l)) \Rightarrow$
$n_l = n'_l$

From [4]-[9]

$\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$(n'_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$n_l = n'_l$

Lemma 66.
$\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$v = v'$
where
Γ is defined in Definition 20.

Proof.
By InvL,

(1) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v))) \in \mathsf{ors}) \wedge \mathsf{self} \Rightarrow$
$v = wval(\mathsf{s}'(n_l))(ts) \wedge v \neq \bot$

By InvS,

(2) $\Gamma \vdash_{\mathsf{ECC}}$
$wval(\mathsf{s}(n))(ts) = v \wedge v \neq \bot \wedge \mathsf{self} \Rightarrow$
$\Box(\mathsf{self} \Rightarrow wval(\mathsf{s}(n))(ts) = v)$

By ASA on [1] and [2],

(3) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v))) \in \mathsf{ors}) \wedge \mathsf{self} \Rightarrow$
$\hat{\Box}(\mathsf{self} \rightarrow wval(\mathsf{s}(n_l))(ts) = v)$

By InvL,

(4) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v'))) \in \mathsf{ors}) \wedge \mathsf{self} \Rightarrow$
$v' = wval(\mathsf{s}(n_l))(ts)$

From [3] and [4], we have

(5) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v))) \in \mathsf{ors}) \wedge \mathsf{self} \Rightarrow$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v'))) \in \mathsf{ors}) \wedge \mathsf{self} \Rightarrow$
$v = v'$

By OR′,

(6) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$\Diamond(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self})$

By OR′,

(7) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$\Diamond(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self})$

From [5], [6], and [7], we have

(8) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}(ts, v))) \Rightarrow$
$(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \Rightarrow$
$v = v'$

Lemma 67.

$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \Rightarrow$
$\Box[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v] \wedge$
$\boxminus[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v]$

where
Γ is defined in Definition 20.

Proof.

By Lemma 64,

$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \leftsquigarrow$
$\exists n'.\ (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts, v))) \in \mathsf{ors} \wedge \mathsf{self})$

Thus, we show the equivalent formula

$\Gamma \vdash_{\mathsf{ECC}}$

$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v))) \Rightarrow$
$(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \Rightarrow$
$v' = v$

The rest of the proof is similar to the proof of Lemma 65 and Lemma 66.

Lemma 68.

$\Gamma \vdash_{\mathsf{ECC}} ts_2 \geq ts_1 \rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts_2, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts_2, v'))) \rightarrow v' = v]$

where

Γ is defined in Definition 20.

Proof.

Proof by induction on ts_2 starting from ts_1.

(1) Base Case: $ts_2 = ts_1$

Immediate from Lemma 67.

(2) Inductive Case:

We assume that

(3) $\Gamma \vdash_{\mathsf{ECC}} \forall ts.\ ts_1 \leq ts < ts_2 \rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v]$

We show that

$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts_2, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts_2, v'))) \rightarrow v' = v]$

By BEB'_3,

(4) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts_2, v'))) \Rightarrow$
$\diamondminus(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v')))$

By OR′,

(5) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \Rightarrow$
$\diamondminus(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self})$

From [4] and [5], we need to show that

$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \Rightarrow$
$\Box[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v] \wedge$
$\boxminus[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v]$

From [3], we need to show that

$\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts_1, v))) \wedge$
$\forall ts.\ ts_1 \leq ts < ts_2 \rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v]$
$\Rightarrow$

$\Box[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v] \wedge$
$\boxminus[(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \rightarrow v' = v]$

We can show that

(6) $\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \Rightarrow$
$\exists N.\ |N| > |\mathbb{N}|/2 \ \wedge\ \forall n'.\ n' \in N \rightarrow \diamondminus(valts(\mathsf{s}'(n')) = ts \wedge val(\mathsf{s}'(n')) = v \wedge ts > rts(\mathsf{s}(n'))$

By InvLSe, we have

(7) $\Gamma \vdash_{\mathsf{ECC}}$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \wedge wval(\mathsf{s}'(n_l))(ts_2) = v' \wedge v' \neq \bot \wedge$
$highest(states(\mathsf{s}'(n_l))) \neq \bot \rightarrow wval(\mathsf{s}'(n_l))(ts_2) = highest(states(\mathsf{s}'(n_l)))$

We have (from the two intersecting sets)

(8) $\Gamma \vdash_{\mathsf{ECC}}$
$\forall N.\ |N| > |\mathbb{N}|/2 \wedge$
$\forall n'.\ n' \in N \rightarrow \diamondminus(valts(\mathsf{s}'(n')) = ts_1 \wedge val(\mathsf{s}'(n')) = v \wedge ts > rts(\mathsf{s}(n'))) \wedge$
$|states(\mathsf{s}'(n_l))| > |\mathbb{N}|/2$
$\Rightarrow$
$\exists n'.\ n' \in dom(states(\mathsf{s}'(n_l))) \wedge \diamondminus(valts(\mathsf{s}'(n')) = ts_1 \wedge val(\mathsf{s}'(n')) = v \wedge ts_1 > rts(\mathsf{s}(n')))$

By InvSA, then OI′ and InvL

(9) $\Gamma \vdash_{\mathsf{ECC}}$
$\mathsf{self} \wedge states(\mathsf{s}'(n_l))(n') = \langle ts_{n'}, v_{n'} \rangle \ \wedge\ ets(\mathsf{s}(n_l)) = ts_2 \Rightarrow$
$\diamondminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Prepare}(ts_2))) \wedge$
$\quad valts(\mathsf{s}(n')) = ts_{n'} \ \wedge\ val(\mathsf{s}(n')) = v_{n'} \ \wedge\ rts(\mathsf{s}'(n')) = ts_2]$

By InvS″

(10) $\Gamma \vdash_{\mathsf{ECC}}$
$(\mathsf{self} \wedge rts(\mathsf{s}'(n')) \geq ts_2) \Rightarrow \hat{\Box}(\mathsf{self} \rightarrow rts(\mathsf{s}(n')) \geq ts_2)$

From [10], we have

(11) $\Gamma \vdash_{\mathsf{ECC}}$
$ts_1 \leq ts_2 \rightarrow$
$\diamondminus(valts(\mathsf{s}'(n')) = ts_1 \wedge val(\mathsf{s}'(n')) = v \wedge ts_1 > rts(\mathsf{s}(n'))) \wedge$
$\diamondminus(valts(\mathsf{s}(n')) = ts_{n'} \ \wedge\ val(\mathsf{s}(n')) = v_{n'} \ \wedge\ rts(\mathsf{s}'(n')) = ts_2) \Rightarrow$
$\diamondminus[(valts(\mathsf{s}(n')) = ts_{n'} \ \wedge\ val(\mathsf{s}(n')) = v_{n'} \ \wedge\ rts(\mathsf{s}'(n')) = ts_2)$
$\quad \diamondminus(valts(\mathsf{s}'(n')) = ts_1 \wedge val(\mathsf{s}'(n')) = v \wedge ts_1 > rts(\mathsf{s}(n'))$

Thus, we have

(12) $\Gamma \vdash_{\mathsf{ECC}}$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}(ts, v))) \wedge$
$\forall ts.\ ts_1 \leq ts < ts_2 \rightarrow$
$\Box[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v] \wedge$
$\boxminus[(n' \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Accept}(ts, v'))) \rightarrow v' = v]$
$\Rightarrow$
$(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}(ts_2, v'))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$\exists ts, n'.\ states(\mathsf{s}'(n_l))(n') = \langle ts, v \rangle \wedge ts_1 \leq ts < ts_2 \wedge$
$highest(states(\mathsf{s}'(n_l))) = v' = v$

Theorem 21. *(EC_3: Integrity)*
Every node decides at most once.
$\Gamma \vdash_{\mathsf{ECC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\quad \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$
where
Γ is defined in Definition 20.

Proof.
By InvL,
(1) $\Gamma' \vdash_{\mathsf{ECC}} n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \Rightarrow$
$\quad deciced(\mathsf{s}(n)) = \mathsf{false} \wedge decided(\mathsf{s}'(n)) = \mathsf{true}$
By InvSSe,
(2) $\Gamma' \vdash_{\mathsf{ECC}} decided(\mathsf{s}(n_l)) = \mathsf{true} \Rightarrow$
$\quad \Box(decided(\mathsf{s}(n_l)) = \mathsf{true})$
By [1] and [2],
(3) $\Gamma' \vdash_{\mathsf{ECC}} n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \Rightarrow$
$\quad \Box decided(\mathsf{s}'(n)) = \mathsf{true}$
By PostPre on [3],
(4) $\Gamma' \vdash_{\mathsf{ECC}} n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \Rightarrow$
$\quad \Box\circ\, decided(\mathsf{s}(n)) = \mathsf{true}$
That is,
(5) $\Gamma' \vdash_{\mathsf{ECC}} n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \Rightarrow$
$\quad \hat{\Box} decided(\mathsf{s}(n)) = \mathsf{true}$
The contra-positive of [1] and changing v to v',
(6) $\Gamma' \vdash_{\mathsf{ECC}} \neg(deciced(\mathsf{s}(n)) = \mathsf{false} \wedge decided(\mathsf{s}'(n)) = \mathsf{true}) \Rightarrow$
$\quad \neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$
That is,
(7) $\Gamma' \vdash_{\mathsf{ECC}} (deciced(\mathsf{s}(n)) = true) \Rightarrow$
$\quad \neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$
From [5] and [7]
(8) $\Gamma' \vdash_{\mathsf{ECC}} n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v) \Rightarrow$
$\quad \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

Theorem 22. *(EC_4: Termination)*
If a correct node proposes and an epoch is started with that node as the leader, then every correct node eventually decides a value.

$\Gamma \vdash_{\mathsf{ECC}} n_l \in \mathsf{Correct} \rightarrow$
$\quad |\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow \Diamond(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \Rightarrow$
$\quad\quad (n \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \Rightarrow$
$\quad\quad \forall n'.\ n' \in \mathsf{Correct} \rightarrow \Diamond\Diamond\exists v'.\ (n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

where
Γ is defined in Definition 20.

Proof.
By IROR,
(1) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge n = n_l \Rightarrow$
$\quad\quad \Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts)))$
By BEB'_1 on [1],
(2) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\quad\quad \Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Prepare}(ts)))$
By IIOR,
(3) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Prepare}(ts)) \wedge ts > rts(\mathsf{s}(n))) \Rightarrow$
$\quad\quad \Diamond(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}\langle ts, vts, val\rangle))$
By PL'_1 on [3] and using the Lemma 89,
(4) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Prepare}(ts)) \wedge ts > rts(\mathsf{s}(n))) \Rightarrow$
$\quad\quad \Diamond(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}\langle ts, vts, val\rangle))$
By II,
(5) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}\langle ts, vts, val\rangle) \wedge ets(\mathsf{s}(n_l)) = ts) \Rightarrow$
$\quad\quad \mathsf{self} \wedge \langle vts, val\rangle \in states(\mathsf{s}'(n_l))(n)$
By InvS,
(6) $\Gamma \vdash_{\mathsf{ECC}} \mathsf{self} \wedge \langle vts, val\rangle \in states(\mathsf{s}'(n_l))(n) \Rightarrow$
$\quad\quad (\mathsf{self} \Rightarrow \langle vts, val\rangle \in states(\mathsf{s}'(n_l))(n))$
From [2] and [6] to [4],
(7) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\quad\quad \Diamond[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle) \wedge) \wedge$
$\quad\quad\quad \mathsf{self} \Rightarrow \langle vts, val\rangle \in states(\mathsf{s}'(n_l))(n)]$
Thus, there exists n' such that
(8) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\quad\quad \Diamond[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$\quad\quad\quad \mathsf{self} \Rightarrow \forall n.\ n \in \mathsf{Correct} \rightarrow \langle vts, val\rangle \in states(\mathsf{s}'(n_l))(n)]$
That is,
(9) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\quad\quad \Diamond[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$\quad\quad\quad \mathsf{self} \Rightarrow (|states(\mathsf{s}'(n_l))| \geq |\mathsf{Correct}|)]$
By Definition 20 (a majority of correct nodes) on [9],
(10) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\quad\quad \Diamond[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$\quad\quad\quad \mathsf{self} \Rightarrow (|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2)]$
Thus, as the deliver event is a self event

(11) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\Diamond[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2]$

By Lemma 69 on [11]

(12) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\Diamond[\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))]$

By Lemma 112 on [12]

(13) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Prepare}(ts))) \Rightarrow$
$\Diamond\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))$

From [1] to [3] and [13],

(14) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge n = n_l \Rightarrow$
$\Diamond[\Diamond\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))]$

By Lemma 112 on [15],

(15) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge n = n_l \Rightarrow$
$\Diamond\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))$

By BEB'_1 on [15],

(16) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!-\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))$

By Lemma 112 and Lemma 87 on [16],

(17) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!-(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))$

By IIOR,

(18) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \wedge\ t > rts(\mathsf{s}(n))) \Rightarrow$
$\Diamond(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Accepted}\langle t\rangle))$

From [17], [18] and PL'_1,

(19) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!-\Diamond(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle)) \vee$
$\Diamond\Diamond(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle))$

By Lemma 112, Lemma 87 on [19] and [1],

(20) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!-(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle)) \vee$
$\Diamond(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle))$

By II,

(21) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle)) \wedge ets(\mathsf{s}(n_l)) = t \Rightarrow$
$\mathsf{self} \wedge (n \in accepted(\mathsf{s}'(n_l)))$

By InvS,

(22) $\Gamma \vdash_{\mathsf{ECC}} \mathsf{self} \wedge (n \in accepted(\mathsf{s}'(n_l))) \Rightarrow$
$\mathsf{self} \Rightarrow (n \in accepted(\mathsf{s}'(n_l)))$

From [20] to [22], and Lemma 112,

(23) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!-\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Accepted}\langle t\rangle)) \wedge$
$\mathsf{self} \Rightarrow (n \in accepted(\mathsf{s}'(n_l)))]$

There exists n' such that

(24) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}\langle t\rangle)) \wedge$
$\mathsf{self} \Rightarrow \ \forall n.\ n \in \mathsf{Correct} \rightarrow n \in \mathit{accepted}(\mathsf{s}'(n_l))]$

That is,

(25) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}\langle t\rangle)) \wedge$
$\mathsf{self} \Rightarrow \ \forall n.\ \mathsf{Correct} \subseteq \mathit{accepted}(\mathsf{s}'(n_l))]$

By Definition 20 (a majority of correct nodes) on [25],

(26) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}\langle t\rangle)) \wedge$
$\mathsf{self} \Rightarrow |\mathit{accepted}(\mathsf{s}'(n_l))| > |\mathbb{N}|/2$

Thus, as the deliver event is a self event

(27) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}\langle t\rangle)) \wedge$
$|\mathit{accepted}(\mathsf{s}'(n_l))| > |\mathbb{N}|/2$

By InvL,

(28) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', \textsc{Accepted}\langle t\rangle)) \wedge$
$|\mathit{accepted}(\mathsf{s}'(n_l))| > |\mathbb{N}|/2 \ \wedge \ \mathit{ets}(\mathsf{s}(n_l)) = t \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}\langle v\rangle))]$

From [27], [28] and Lemma 112

(29) $\Gamma \vdash_{\mathsf{ECC}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}\langle v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}\langle v\rangle))$

By BEB'_1,

(30) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Decided}\langle v\rangle)) \Rightarrow$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}\langle v\rangle))$

From [29], [30], Lemma 87, and Lemma 112

(31) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-(n_l \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}\langle v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}\langle v\rangle))$

By IIOI

(32) $\Gamma \vdash_{\mathsf{ECC}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{Decided}\langle v\rangle)) \wedge \mathit{decided}(\mathsf{s}(n)) = \mathsf{false} \Rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))$

From [31] and [32],

(33) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \vee$
$\Diamond\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))$

By Lemma 112 and Lemma 87 on [33],

(34) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle) \Rightarrow$
$\Diamond\!\!\!-\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))$

From [15] and [34],

(35) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge n = n_l \Rightarrow$
$\Diamond\!\!\!-[\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))] \vee$
$\Diamond[\Diamond\!\!\!-(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))]$

By Lemma 112, Lemma 87 and Lemma 86 on [15],

(36) $\Gamma \vdash_{\mathsf{ECC}} \forall n.\ n \in \mathsf{Correct} \rightarrow (n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge n = n_l \Rightarrow$
$\Diamond\!\!\!\!-(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \vee$
$\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v))$

That is,

(37) $\Gamma \vdash_{\mathsf{ECC}} n_l \in \mathsf{Correct} \rightarrow$
$|\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow$
$(n_l \bullet \top \downarrow \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \Rightarrow$
$\forall n'.\ n' \in \mathsf{Correct} \rightarrow \Diamond\!\!\!\!-\Diamond \exists v'.\ (n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

Lemma 69.
$\Gamma \vdash_{\mathsf{ECC}}$
$[(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2)] \Rightarrow$
$\Diamond\!\!\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle)) \vee$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle t, v\rangle))$

Proof.
There are two cases, the first case is $wval(\mathsf{s}(n_l)) = \perp$.

By InvL

(1) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = \perp] \Rightarrow$
$\mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)) \in \mathsf{ors}$

By OR

(2) $\Gamma \vdash_{\mathsf{ECC}} \mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle v\rangle) \in \mathsf{ors}) \Rightarrow$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

From [1] and [2],

(3) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = \perp] \Rightarrow$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

That is,

(4) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = \perp] \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

The second case is $wval(\mathsf{s}(n_l))(ts) = v \wedge v \neq \perp$, by InvSA with,
S instantiated to $wval(\mathsf{s}(n_l))(ts) \neq \perp$
A instantiated to $(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

(5) $\Gamma \vdash_{\mathsf{ECC}} \mathsf{self} \wedge wval(\mathsf{s}(n_l))(ts) \wedge v \neq \perp \Rightarrow$
$\hat{\Diamond}\!\!\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

That is,

(6) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = v \wedge v \neq \perp] \Rightarrow$
$\hat{\Diamond}\!\!\!\!-(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)$

From [4] and [6],

(7) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver_{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = \bot] \vee$
$[(n_l \bullet 0 \downarrow \mathsf{deliver_{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \wedge wval(\mathsf{s}(n_l)) = v \wedge v \neq \bot] \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)) \vee$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle))$

That is,

(8) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver_{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2)] \wedge$
$[wval(\mathsf{s}(n_l)) = \bot \vee wval(\mathsf{s}(n_l)) \neq \bot] \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)) \vee$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle))$

That is,

(9) $\Gamma \vdash_{\mathsf{ECC}} [(n_l \bullet 0 \downarrow \mathsf{deliver_{pl}}(n, \textsc{State}\langle ts, vts, val\rangle)) \wedge$
$(|states(\mathsf{s}'(n_l))| \geq |\mathbb{N}|/2) \Rightarrow$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle)) \vee$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast_{beb}}(\textsc{Accept}\langle ts, wval(\mathsf{s}'(n_l))(ts)\rangle))$

5.3.8 Epoch Change

Definition 21.

$\Gamma =$
$\quad \mathsf{PL}'_1, \mathsf{PL}'_2, \mathsf{PL}'_3, \mathsf{BEB}'_1; \mathsf{BEB}'_2; \mathsf{BEB}'_3; \mathsf{ELE}'_1$

$\mathsf{PL}'_1 = \mathsf{lower}(0, \mathsf{PL}_1) =$
$\quad n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad (n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n', m) \rightsquigarrow$
$\quad n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, m))$

$\mathsf{PL}'_2 = \mathsf{lower}(0, \mathsf{PL}_2) =$
$\quad [n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))] \rightarrow$
$\quad [n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m))]$

$\mathsf{PL}'_3 = \mathsf{lower}(0, \mathsf{PL}_3) =$
$\quad (n \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n', m)) \leftsquigarrow$
$\quad (n' \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n, m))$

$\mathsf{BEB}'_1 = \mathsf{lower}(1, \mathsf{BEB}_1) =$
$\quad n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$\quad (n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m)) \rightsquigarrow$
$\quad (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))$

$\mathsf{BEB}'_2 = \mathsf{lower}(1, \mathsf{BEB}_2) =$
$\quad [n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(n, m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(n, m))] \rightarrow$
$\quad [n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m) \Rightarrow$
$\quad\quad \hat{\boxminus} \neg (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m))]$

$\mathsf{BEB}'_3 = \mathsf{lower}(1, \mathsf{BEB}_3) =$
$\quad (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n', m)) \leftsquigarrow$
$\quad (n' \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(m))$

The property of the eventual leader detector subcomponent:
Eventually every correct process trusts the same correct process.
$\mathsf{ELE}'_1 = \mathsf{lower}(2, \mathsf{ELE}_1) =$
$\quad \exists l.\ l \in \mathsf{Correct} \wedge$
$\quad [n \in \mathsf{Correct} \rightarrow$
$\quad\quad \Diamond (n \bullet \top \uparrow \mathsf{trust}(l) \wedge \hat{\Box} \neg (n \bullet \top \uparrow \mathsf{trust}(l')))]$

Theorem 23. *(ECH_1: Monotonicity)*
If a correct process starts an epoch (ts, n_l) and later starts an epoch (ts', n_l'), then $ts' > ts$.

$\vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$\quad (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\quad \hat{\Box}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \rightarrow ts' > ts)$

Proof.

By OI,
(1) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l))$
By OI′,
(2) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self})$
By InvL,
(3) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self} \Rightarrow$
$lastts(\mathsf{s}'(n)) = ts \wedge ts > lastts(\mathsf{s}(n))$
That is,
(4) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self} \Rightarrow$
$lastts(\mathsf{s}'(n)) = ts$
By InvL,
(5) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \wedge lastts(\mathsf{s}(n)) = ts \wedge \mathsf{self} \Rightarrow$
$ts' > ts$
That is,
(6) $\Gamma \vdash_{\mathsf{ECH}} lastts(\mathsf{s}(n)) = ts \wedge \mathsf{self} \Rightarrow$
$(\mathsf{self} \wedge n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \Rightarrow ts' > ts)$
That is,
(7) $\Gamma \vdash_{\mathsf{ECH}} lastts(\mathsf{s}(n)) = ts \wedge \mathsf{self} \Rightarrow$
$(\mathsf{self} \Rightarrow (n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \Rightarrow ts' > ts))$
From [4] and [7] and ASA,
(8) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self} \Rightarrow$
$\hat{\Box}(\mathsf{self} \rightarrow (n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \Rightarrow ts' > ts))$
That is,
(9) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$\bigcirc \Box(\mathsf{self} \rightarrow (n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \Rightarrow ts' > ts))$
That is,
(10) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$\bigcirc (\mathsf{self} \wedge n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \in \mathsf{ois} \Rightarrow \ ts' > ts)$
From [10] and ExeOrderOR,
(11) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \Rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \Rightarrow ts' > ts)$
That is,
(12) $\Gamma \vdash_{\mathsf{ECH}} n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \Rightarrow$
$\hat{\Box}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l') \rightarrow ts' > ts)$

Theorem 24. *(ECH_2: Consistency)*
If a correct process starts an epoch (ts, n_l) and another correct process starts an epoch (ts, n'_l), then $n_l = n'_l$.

$\vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$(n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n'_l) \Rightarrow n_l = n'_l)$

Proof.

By OI′,
(1) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Diamond}(n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self})$
By InvL,
(2) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \langle \textsc{NewEpoch}, ts \rangle))$
By BEB_3,
(3) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \langle \textsc{NewEpoch}, ts \rangle))$
$\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle \textsc{NewEpoch}, ts \rangle))$
From [1] to [3],
(4) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Diamond}\Diamond(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle \textsc{NewEpoch}, ts \rangle))$
That is,
(5) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle \textsc{NewEpoch}, ts \rangle))$
By OR′,
(6) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\langle \textsc{NewEpoch}, ts \rangle)) \Rightarrow$
$\hat{\Diamond}(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}, ts)) \in \mathsf{ors} \wedge ts(\mathsf{s}'(n_l)) = ts \wedge \mathsf{self})$
From [5] and [6],
(7) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Diamond}\hat{\Diamond}(n_l \bullet \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}, ts) \in \mathsf{ors} \wedge ts(\mathsf{s}'(n_l)) = ts \wedge \mathsf{self})$
By InvUSSe′ and $\mathcal{S}(\mathsf{s}(n_l)) : ts(\mathsf{s}(n_l)) \mod \mathbb{N} = rank(n_l)$,
(8) $\Gamma \vdash_{\mathsf{ECH}} \Box\, [ts(\mathsf{s}(n_l)) \mod \mathbb{N} = rank(n_l)]$
By PostPre on [8],
(9) $\Gamma \vdash_{\mathsf{ECH}} \Box\, [ts(\mathsf{s}'(n_l)) \mod \mathbb{N} = rank(n_l)]$
From [8] and [9],
(10) $\Gamma \vdash_{\mathsf{ECH}} \Box\, [ts(\mathsf{s}(n_l)) \mod \mathbb{N} = ts(\mathsf{s}'(n_l)) \mod \mathbb{N} = rank(n_l)]$
From [7] and [10]
(11) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Diamond}\hat{\Diamond}\Box[ts(\mathsf{s}'(n_l)) = ts \wedge ts \mod \mathbb{N} = rank(n_l)]$
That is,
(12) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\Box[ts(\mathsf{s}'(n_l)) = ts \wedge ts \mod \mathbb{N} = rank(n_l)]$
Similarly, we can say,
(13) $\Gamma \vdash_{\mathsf{ECH}} (n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n'_l)) \Rightarrow$

$$\Box[ts(\mathsf{s}'(n_l')) = ts \wedge ts \mod \mathbb{N} = rank(n_l')]$$

From [12] and [13]

(14) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$(n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l')) \Rightarrow$
$\Box[ts \mod \mathbb{N} = rank(n_l) \wedge ts \mod \mathbb{N} = rank(n_l')]$

From [14] and the uniqueness of rank function we can easily say,

(15) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$(n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l')) \Rightarrow \Box(n_l = n_l')p$

That is,

(16) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$((n' \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l')) \Rightarrow n_l = n_l')$

Theorem 25. *(ECH_3: Eventual leadership)*
There is a timestamp ts and a correct process n_l such that eventually every correct process starts an epoch with ts and n_l and does not start another epoch afterwards.

$$\vdash_{\mathsf{ECH}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))]]$$

Proof.

From ELE_1,

(1) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge \hat{\Box}\neg(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l'))]]$

By II,

(2) $\Gamma \vdash_{\mathsf{ECH}} n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l) \Rightarrow trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self}$

By InvL,

(3) $\Gamma \vdash_{\mathsf{ECH}} trusted(\mathsf{s}(n)) \neq trusted(\mathsf{s}'(n)) \Rightarrow n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l')$

The contrapositive of [3],

(4) $\Gamma \vdash_{\mathsf{ECH}} \neg\ (n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l')) \Rightarrow \neg\ (trusted(\mathsf{s}(n)) \neq trusted(\mathsf{s}'(n)))$

That is,

(5) $\Gamma \vdash_{\mathsf{ECH}} \neg\ (n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l')) \Rightarrow (trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)))$

From [1], [2] and [5],

(6) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[((n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self}) \wedge \hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)))]]$

By PostPre on [6],

(7) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge \bigcirc(trusted(\mathsf{s}(n)) = n_l \wedge \mathsf{self}) \wedge \hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)))]]$

From [7],

(8) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge \hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]$

By instantiating n to n_l,

(9) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge [n \in \mathsf{Correct} \rightarrow \Diamond[(n_l \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n_l)) = n_l \wedge \hat{\Box}(trusted(\mathsf{s}(n_l)) = trusted(\mathsf{s}'(n_l)) = n_l \wedge \mathsf{self})]]$

From [8], [9] and Lemma 127,

(10) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[\Diamond \hat{\Diamond}[(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n_l)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n_l)) = trusted(\mathsf{s}'(n_l)) = n_l \wedge \mathsf{self})] \wedge$
$(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]$

By IR,

(11) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \Rightarrow$
$n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t))) \in \mathsf{ors} \wedge \mathsf{self}$

By OR and BEB_1,

(12) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t))) \in \mathsf{ors} \wedge \mathsf{self} \Rightarrow$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t)))$

From [11] and [12],

(13) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(n_l \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \Rightarrow$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t)))$

From [10] and [13],

(14) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[\Diamond \hat{\Diamond} \Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]$

That is (node n can deliver NewEpoch message before or after receiving trust event),

(15) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[[\hat{\Diamond}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \vee$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t)))] \wedge$
$[(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]$

That is,

(16) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[[\hat{\Diamond}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})] \vee$
$[\hat{\Diamond}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$(n \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n_l)) \wedge trusted(\mathsf{s}'(n)) = n_l \wedge$
$\hat{\Box}(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]]$

That is,

(17) $\Gamma \vdash_{\mathsf{ECH}} \exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\hat{\Diamond} \Box(trusted(\mathsf{s}(n)) = n_l \wedge \mathsf{self})] \vee$
$\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box(trusted(\mathsf{s}'(n)) = n_l \wedge \mathsf{self})]]]$

By Lemma 70 and Lemma 71 on [17],

(18) $\Gamma \vdash_{\mathsf{ECH}} \exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[\hat{\Diamond}[\Diamond(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)] \vee$
$\Diamond[\Diamond(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)]]]$

That is,

(19) $\Gamma \vdash_{\mathsf{ECH}} \exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}(ts, l) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)]]$

By Lemma 72 on [19],

(20) $\Gamma \vdash_{\mathsf{ECH}} \exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond[\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}(ts, l) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{startEpoch}(ts', l')]]]$

That is,

(21) $\Gamma \vdash_{\mathsf{ECH}} \exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\Diamond((n \bullet \top \uparrow \mathsf{startEpoch}(ts, l) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{startEpoch}(ts', l')))]$

Lemma 70.
$\Gamma \vdash_{\mathsf{ECH}}$
$\exists l.\ l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Diamond[trusted(\mathsf{s}(n)) \neq n_l \wedge trusted(\mathsf{s}'(n)) = n_l \wedge \hat{\Box}\ trusted(\mathsf{s}(n)) = n_l] \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \wedge \hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)]$

Proof.

By InvL
(1) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \Rightarrow$
$\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts))) \in \mathsf{ois}$

By OR
(2) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts))) \in \mathsf{ois} \Rightarrow$
$\hat{\Diamond}(n \bullet 1 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts)))$

From [1] and [2] and $\hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l$ from the premise,
(3) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}(n \bullet 1 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts)))$

By PL_1 and correctness of n_l and n,
(4) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts))) \Rightarrow$
$\Diamond(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts)))$

By InvL,
(5) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts))) \Rightarrow$
$\mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lts')) \in \mathsf{ors}))$

By OR,

(6) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lst'))) \in \mathsf{ors}) \Rightarrow$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lst')))$

By BEB_1 and correctness of n_l and n,

(7) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lst'))) \Rightarrow$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lst')))$

From [3] to [7],

(8) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l$

By Lemma 123 on [8],

(9) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l] \vee$
$\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge \Box\ trusted(\mathsf{s}(n)) = n_l]$

By Lemma 124 on [9],

(10) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge \Box\ trusted(\mathsf{s}(n)) = n_l]$

By Lemma 71 and [9],

(11) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\Diamond[\Diamond(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)]$

That is,

(12) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$trusted(\mathsf{s}(n)) \neq n_l \wedge \hat{\Diamond}\Box\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n_l))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)]$

Lemma 71.

$\Gamma \vdash_{\mathsf{ECH}}$
$\exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge \Box\ trusted(\mathsf{s}(n)) = n_l] \Rightarrow$
$\Diamond(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l) \wedge \hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l)$

Proof.

The First case is that $nk(\mathsf{s}(n))(n_l) = \mathsf{false}$

By InvL,

(1) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge nk(\mathsf{s}'(n))(n_l) = \mathsf{true}$

By InvS″,

(2) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge nk[\mathsf{s}'(n_l)] = \mathsf{true} \Rightarrow \hat{\Box}(\mathsf{self} \rightarrow nk(\mathsf{s}(n))(n_l) = \mathsf{true})$

By OR,

(3) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ois} \Rightarrow$
$\hat{\Diamond}(n \bullet 1 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts)))$

By PL_1 and correctness of n_l and n,

(4) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \Rightarrow$
$\Diamond(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}(lts)))$

From [1] to [4],

(5) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\, trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\hat{\Diamond}(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

By InvL,

(6) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \Rightarrow$
$\mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lts')))) \in \mathsf{ors}) \wedge lst' > lst$

By OR,

(7) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lts')))) \in \mathsf{ors}) \Rightarrow$
$\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lts')))$

By BEB_1 and correctness of n_l and n,

(8) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 1 \downarrow \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(lts'))) \Rightarrow$
$\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts')))$

From [6] to [8],

(9) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \Rightarrow$
$\hat{\Diamond}(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge lst' > lst$

From [5] and [9],

(10) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\, trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\Diamond(\Diamond(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge lst' > lst) \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

That is,

(11) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\, trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\hat{\Diamond}[n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts')) \wedge$
$lst' > lst \wedge nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \wedge trusted(\mathsf{s}(n)) = n_l] \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

There are two possibilities for the value of $lastts(\mathsf{s}(n))$ when a node receieves $\mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))$ event,

(12) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\, trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\hat{\Diamond}[n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts')) \wedge$
$lst' > lst \wedge nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \wedge trusted(\mathsf{s}(n)) = n_l \wedge$
$(lastts(\mathsf{s}(n)) = lst \vee lastts(\mathsf{s}(n)) \neq lst)] \wedge$
$\Box\, trusted(\mathsf{s}(n)) = n_l lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

By Lemma 115,

(13) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$

$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$[\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge$
$lst' > lst \wedge lastts(\mathsf{s}(n)) = lst \wedge\ trusted(\mathsf{s}(n)) = n_l] \vee$
$\hat{\Diamond}[(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge$
$lastts(\mathsf{s}(n)) = lts'' \wedge lts'' \neq lst \wedge\ trusted(\mathsf{s}(n)) = n_l]] \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

By InvL

(14) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge$
$lts' > lastts(\mathsf{s}(n)) \wedge\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\mathsf{self} \wedge n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l) \in \mathsf{ois}$

By OI

(15) $\Gamma \vdash_{\mathsf{ECH}} \mathsf{self} \wedge n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l) \in \mathsf{ois} \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l))$

From [14] and [15],

(16) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(lts'))) \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge$
$lts' > lastts(\mathsf{s}(n)) \wedge\ trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l))$

By [13], [16] and removing unuseful parameters,

(17) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$[\hat{\Diamond}[\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \vee$
$\hat{\Diamond}[lastts(\mathsf{s}(n)) = lts'' \wedge lts'' \neq lst \wedge\ \hat{\Box} trusted(\mathsf{s}(n)) = n_l] \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]$

That is,

(18) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$[\hat{\Diamond}[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}] \vee$
$[\hat{\Diamond}[lastts(\mathsf{s}(n)) = lts'' \wedge lts'' \neq lst \wedge\ \hat{\Box} trusted(\mathsf{s}(n)) = n_l] \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]]$

By InvSAG,

(19) $\Gamma \vdash_{\mathsf{ECH}} lastts(\mathsf{s}(n)) \neq lts''\ \wedge\ \Diamond(lastts(\mathsf{s}(n)) = lts'') \Rightarrow$
$\Diamond(n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(lts'', n_l') \in \mathsf{ois}\ \wedge\ \mathsf{self})$

From [15], [19] and adding $\Box trusted(\mathsf{s}(n)) = n_l$,

(20) $\Gamma \vdash_{\mathsf{ECH}} lastts(\mathsf{s}(n)) \neq lts''\ \wedge\ \Diamond(lastts(\mathsf{s}(n)) = lts'') \wedge \Box trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts'', n_l))$

From [18] and [20],

(21) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$[\hat{\Diamond}[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}] \vee$
$[\hat{\Diamond}(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts'', n_l))] \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge lastts(\mathsf{s}(n)) = lts \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]]$

That is,

(22) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow$
$\hat{\Diamond}[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \wedge$

$$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]$$

The other case is that $nk(\mathsf{s}(n))(n_l) = \mathsf{true}$,
By InvSA,

(23) $\Gamma \vdash_{\mathsf{ECH}} nk(\mathsf{s}(n))(n_l) = \mathsf{true} \Rightarrow$
$\hat{\Leftrightarrow}(\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \ \wedge \ trusted(\mathsf{s}(n)) = n_l)$

From [23] and and adding premise,

(24) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \Rightarrow$
$\hat{\Leftrightarrow}(\mathsf{self} \wedge n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \ \wedge \ trusted(\mathsf{s}(n)) = n_l)$

By doing the same reasoning we can say,

(25) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{true} \Rightarrow$
$\hat{\Leftrightarrow}[\hat{\Diamond}[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]$

Therefore, we can say,

(26) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t))) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l] \Rightarrow$
$\hat{\Leftrightarrow}[\hat{\Diamond}[(n \bullet \top \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(lts', n_l)) \wedge$
$\Box\ trusted(\mathsf{s}(n)) = n_l \wedge \hat{\Box} nk(\mathsf{s}(n))(n_l) = \mathsf{true}]$

Lemma 72.

$\Gamma \vdash_{\mathsf{ECH}}$
$\exists n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge \hat{\Box}\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}(ts', l) \wedge \Box\neg(n \bullet \top \uparrow \mathsf{startEpoch}(ts'', l')]$

Proof.

By Lemma 73,

(1) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\Leftrightarrow\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\boxminus}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts)))]$

By Lemma 126 and [1],

(2) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \Rightarrow$
$\Diamond\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts)))$

By InvL,

(3) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t'))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts'))) \vee$
$(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Nack}(lts'))) \vee$
$(n_l \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n))$

By InvL,

(4) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet 2 \uparrow \mathsf{trust}_{\mathsf{eld}}(n)) \Rightarrow$
$trusted(\mathsf{s}(n_l)) \neq trusted(\mathsf{s}'(n_l))$

From [3] and [4],

(5) $\Gamma \vdash_{\mathsf{ECH}} (n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t'))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts')))\vee$
$(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Nack}(lts')))\vee$
$trusted(\mathsf{s}(n_l)) \neq trusted(\mathsf{s}'(n_l))$

The contra-positive of [5],

(6) $\Gamma \vdash_{\mathsf{ECH}} \neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts')))\wedge$
$\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Nack}(lts')))\wedge$
$trusted(\mathsf{s}(n_l)) = trusted(\mathsf{s}'(n_l)) \Rightarrow$
$\neg(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t'))) \in \mathsf{ors} \wedge \mathsf{self})$

That is,

(7) $\Gamma \vdash_{\mathsf{ECH}} \hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts')))\wedge$
$\Box\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Nack}(lts')))\wedge$
$\Box(trusted(\mathsf{s}(n_l)) = trusted(\mathsf{s}'(n_l))) \Rightarrow$
$\Box\neg(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t'))) \in \mathsf{ors} \wedge \mathsf{self})$

By InvL,

(8) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts'))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$trusted(\mathsf{s}(n)) \neq n_l$

The contra-positive of [8],

(9) $\Gamma \vdash_{\mathsf{ECH}} trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\neg(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts'))) \in \mathsf{ors} \wedge \mathsf{self})$

That is,

(10) $\Gamma \vdash_{\mathsf{ECH}} \Box trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\Box\neg(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts'))) \in \mathsf{ors} \wedge \mathsf{self})$

By ExeFEOR and PL'_1

(11) $\Gamma \vdash_{\mathsf{ECH}} \Box trusted(\mathsf{s}(n)) = n_l \Rightarrow$
$\Diamond\Box\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{Nack}(lts')))$

From [2] and adding a premise,

(12) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \wedge \Box(trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\Box(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l) \wedge$
$\Box(trusted(\mathsf{s}(n)) = n_l)$

From [12] and [11],

(13) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \wedge \Box(trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\Box(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l) \wedge$
$\Diamond\Box\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts')))$

That is,

(14) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \wedge \Box(trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond[\Box\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\Box(trusted(\mathsf{s}(n)) = trusted(\mathsf{s}'(n)) = n_l) \wedge$
$\Box\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n_l, \textsc{Nack}(lts')))]$

[7] and [14],

(15) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \wedge \Box(trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond[\Box\neg(n_l \bullet (1, \mathsf{broadcast}_{\mathsf{beb}}(\textsc{NewEpoch}(t'))) \in \mathsf{ors} \wedge \mathsf{self}]$

By ExeFEOR and BEB'_1

(16) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$(nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self}) \wedge \Box(trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond[\Diamond\Box\neg(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t'))]$

By InvL,

(17) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(t, n')\mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t')))$

The contra-positive of [17],

(18) $\Gamma \vdash_{\mathsf{ECH}} \neg(n \bullet 1 \uparrow \mathsf{deliver}_{\mathsf{beb}}(n_l, \textsc{NewEpoch}(t'))) \Rightarrow$
$\neg(n \bullet \mathsf{startEpoch}_{\mathsf{ech}}(t, n')\mathsf{ois} \wedge \mathsf{self})$

From [16] and [18],

(19) $\Gamma \vdash_{\mathsf{ECH}}$ $(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))[n_l] = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\Diamond\Box\neg(n \bullet \top \uparrow \mathsf{startEpoch}(ts'', l'))$

By Lemma 122,

(20) $\Gamma \vdash_{\mathsf{ECH}}$ $(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\hat{\Box}\ nk(\mathsf{s}(n))[n_l] = \mathsf{true} \wedge \Box\ trusted(\mathsf{s}(n)) = n_l) \Rightarrow$
$\Diamond[(n \bullet \top \uparrow \mathsf{startEpoch}(ts, n_l) \wedge$
$\Box\neg(n \bullet \top \uparrow \mathsf{startEpoch}(ts'', l'))]$

Lemma 73.

$\Gamma \vdash_{\mathsf{ECH}}$
$n \in \mathsf{Correct} \rightarrow$
$nk(\mathsf{s}(n))[n_l] = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\Diamond}\Diamond[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\boxminus}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts)))]$

By InvSASe with $S : nk(\mathsf{s}(n))(n_l) = \mathsf{true}$

(1) $\Gamma \vdash_{\mathsf{ECH}} \circledS\ nk(\mathsf{s}(n))[n_l] = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\Diamond}(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(\textsc{State}(lts))) \in \mathsf{ors} \ \wedge \mathsf{self} \wedge$
$lastts(\mathsf{s}(n)) = lts \wedge nk(\mathsf{s}(n))(n_l) = \mathsf{false} \wedge$
$nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \wedge trusted(\mathsf{s}(n)) = n_l)$

By InvSSe″

(2) $\Gamma \vdash_{\mathsf{ECH}} \circledS\ nk(\mathsf{s}'(n))(n_l) = \mathsf{true} \Rightarrow \hat{\Box}(nk(\mathsf{s}(n))(n_l) = \mathsf{true})$

By InvUSSe

(3) $\Gamma \vdash_{\mathsf{ECH}} \circledS\ nk(\mathsf{s}(n))(n_l) = \mathsf{true} \Rightarrow \Box(nk(\mathsf{s}(n))(n_l) = \mathsf{true})$

By Lemma 125,

(4) $\Gamma \vdash_{\mathsf{ECH}} \circledS\ nk(\mathsf{s}(n))(n_l) = \mathsf{false} \Rightarrow \boxminus(nk(\mathsf{s}(n))(n_l) = \mathsf{false})$

From [1], [2] [4] and PostPre,

(5) $\Gamma \vdash_{\mathsf{ECH}} \circledS\ nk(\mathsf{s}(n))[n_l] = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\Diamond}[n \bullet (0, \mathsf{send}_{\mathsf{pl}}(\textsc{State}(lts))) \in \mathsf{ors} \ \wedge \mathsf{self} \wedge$

$\hat{\boxminus}(nk(\mathsf{s}'(n))(n_l) = \mathsf{false}) \wedge$
$\hat{\Box}(nk(\mathsf{s}(n))(n_l) = \mathsf{true})]$

By InvL,

(6) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$nk(\mathsf{s}'(n))(n_l) = \mathsf{true}$

The contra-positive of [6],

(7) $\Gamma \vdash_{\mathsf{ECH}} nk(\mathsf{s}'(n))(n_l) = \mathsf{false} \Rightarrow$
$\neg(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge \mathsf{self})$

That is,

(8) $\Gamma \vdash_{\mathsf{ECH}} \hat{\boxminus} nk(\mathsf{s}'(n))(n_l) = \mathsf{false} \Rightarrow$
$\hat{\boxminus}(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \notin \mathsf{ors} \wedge \mathsf{self})$

By InvL,

(9) $\Gamma \vdash_{\mathsf{ECH}} (n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$nk(\mathsf{s}(n))(n_l) = \mathsf{false}$

The contra-positive of [9],

(10) $\Gamma \vdash_{\mathsf{ECH}} nk(\mathsf{s}(n))(n_l) = \mathsf{true} \Rightarrow$
$\neg(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors}) \wedge \mathsf{self}$

That is,

(11) $\Gamma \vdash_{\mathsf{ECH}} \hat{\Box}(nk(\mathsf{s}(n))(n_l) = \mathsf{true}) \Rightarrow$
$\hat{\Box}(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \notin \mathsf{ors}) \wedge \mathsf{self}$

From [5], [8], [11] and SInv,

(12) $\Gamma \vdash_{\mathsf{ECH}} nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\diamondminus}[(n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \in \mathsf{ors}) \wedge \mathsf{self} \wedge$
$\hat{\Box}((n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \notin \mathsf{ors}) \wedge \mathsf{self}) \wedge$
$\hat{\boxminus}((n \bullet (0, \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \notin \mathsf{ors}) \wedge \mathsf{self})]$

By UniOR and PL'_1,

(13) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\diamondminus}\hat{\diamondsuit}[(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \wedge$
$\hat{\Box}\neg(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts))) \wedge$
$\hat{\boxminus}\neg(n \bullet 0 \downarrow \mathsf{send}_{\mathsf{pl}}(n_l, \textsc{State}(lts)))]$

By PL'_2,

(14) $\Gamma \vdash_{\mathsf{ECH}} n \in \mathsf{Correct} \rightarrow$
$nk(\mathsf{s}(n))(n_l) = \mathsf{true} \wedge \mathsf{self} \Rightarrow$
$\hat{\diamondminus}\hat{\diamondsuit}[(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts))) \wedge$
$\hat{\boxminus}\neg(n_l \bullet 0 \uparrow \mathsf{deliver}_{\mathsf{pl}}(n, \textsc{State}(lts)))]$

5.3.9 Uniform Consensus

Definition 22.
$\Gamma = \mathsf{ECH}'_1, \mathsf{ECH}'_2, \mathsf{ECH}'_3, \mathsf{EC}'_1, \mathsf{EC}'_2, \mathsf{EC}'_3, \mathsf{EC}'_4$

$\mathsf{ECH}'_1 = \mathsf{lower}(0, \mathsf{ECH}_1) =$
$n \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$\hat{\Box}(n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l) \rightarrow ts' > ts)$

$\mathsf{ECH}'_2 = \mathsf{lower}(0, \mathsf{ECH}_2) =$
$n \in \mathsf{Correct} \wedge n' \in \mathsf{Correct} \rightarrow$
$(n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$(n' \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n'_l) \Rightarrow n_l = n'_l)$

$\mathsf{ECH}'_3 = \mathsf{lower}(0, \mathsf{ECH}_3) =$
$\exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[n \in \mathsf{Correct} \rightarrow$
$\quad \Diamond[(n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\quad\quad \hat{\Box}\neg(n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))]]$

$\mathsf{EC}'_1 = \mathsf{lower}(1, \mathsf{EC}_1) =$
$(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\exists n'.\ \Diamond\!\!\!-(n' \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

$\mathsf{EC}'_2 = \mathsf{lower}(1, \mathsf{EC}_2) =$
$n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)\ \wedge$
$\Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \Rightarrow$
$v = v'$

$\mathsf{EC}'_3 = \mathsf{lower}(1, \mathsf{EC}_3) =$
$(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

$\mathsf{EC}'_4 = \mathsf{lower}(1, \mathsf{EC}_4) =$
$|\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow$
$(n \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \Rightarrow$
$(n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge \Box\neg(n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n', ts')) \Rightarrow$
$\forall n'.\ n' \in \mathsf{Correct} \rightarrow \Diamond\!\!\!-\Diamond \exists v'.\ (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

Theorem 26 (UC_1: Termination)**.**
Every correct node eventually decides some value.
$\Gamma \vdash_{\mathsf{UC}} |\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow$
$[\forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ v \neq \bot \wedge \Diamond\!\!\!\!-(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))] \Rightarrow$
$\exists v.\ \Diamond\!\!\!\!-\Diamond(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v))$
where Γ is defined in Definition 22.

Proof.
By IRSe,
(1) $\Gamma' \vdash_{\mathsf{UCC}} \circledS\ (n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v) \wedge v \neq \bot) \Rightarrow$
$prop(\mathsf{s}'(n_l)) \neq \bot$
By InvSSe,
(2) $\Gamma' \vdash_{\mathsf{UCC}} \circledS\ prop(\mathsf{s}(n_l)) \neq \bot \Rightarrow$
$\Box(prop(\mathsf{s}(n_l)) \neq \bot)$
From [1], [2] and PostPre,
(3) $\Gamma' \vdash_{\mathsf{UCC}} \circledS\ (n_l \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v) \wedge v \neq \bot) \Rightarrow$
$\hat{\Box}(prop(\mathsf{s}(n_l)) \neq \bot)$
By using the premise and [3],
(4) $\Gamma' \vdash_{\mathsf{UCC}} \circledS\ |\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow$
$[\forall n'.n' \in \mathsf{Correct} \rightarrow$
$\exists v.\ v \neq \bot \wedge \Diamond\!\!\!\!-(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))] \Rightarrow$
$\hat{\Box}(prop(\mathsf{s}(n')) \neq \bot)$
By ECH'_3 and instantiating n to n_l,
(5) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))]$
From [4] and [5],
(6) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))] \wedge$
$\Diamond\!\!\!\!-\hat{\Box}(prop(\mathsf{s}(n_l)) \neq \bot)$
That is,
(7) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge$
$[(prop(\mathsf{s}(n_l)) \neq \bot) \vee \Diamond(prop(\mathsf{s}(n_l)) \neq \bot)] \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))]$
That is,
(8) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))] \vee$
$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n'_l))]]$
By InvSA,
(9) $\Gamma' \vdash_{\mathsf{UCC}} prop(\mathsf{s}(n_l)) \neq \bot \wedge \mathsf{self} \Rightarrow$
$\hat{\Diamond\!\!\!\!-}(n_l \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self})$
By using Lemma 74 and Lemma 74 on [8] and [9],
(10) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$

$$[n_l \in \mathsf{Correct} \rightarrow \Diamond[\hat{\Diamond}(n_l \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge (n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge \Box\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n'_l, ts')) \in \mathsf{ors} \wedge \mathsf{self})]]$$

That is,

(11) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge [n_l \in \mathsf{Correct} \rightarrow \Diamond\hat{\Diamond}[(n_l \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge \Diamond[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self}) \wedge \Box\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n'_l, ts')) \in \mathsf{ors} \wedge \mathsf{self})]]]$

By [11] and ExeOrderOR and ExeFEOR,

(12) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge [n_l \in \mathsf{Correct} \rightarrow \Diamond\hat{\Diamond}[(n_l \bullet \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \wedge \hat{\Diamond}[(n_l \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \wedge \Diamond\Box\neg(n_l \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n'_l, ts'))]]]$

by IR,

(13) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow (n \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self})$

By OR

(14) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow \hat{\Diamond}(n \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

From [13] and [14],

(15) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow \hat{\Diamond}(n \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

From the premise and [15] we know that,

(16) $\Gamma' \vdash_{\mathsf{UCC}} \forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ \Diamond\!\!\!-(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow \Diamond\!\!\!-\hat{\Diamond}(n \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

From [12] and [16],

(17) $\Gamma' \vdash_{\mathsf{UCC}} n \in \mathsf{Correct} \rightarrow \forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ \Diamond\!\!\!-(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow \Diamond\!\!\!-\hat{\Diamond}(n_l \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \wedge \Diamond[(n_l \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \wedge \Box\neg(n_l \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n'_l, ts'))]$

From EC'_4,

(18) $\Gamma' \vdash_{\mathsf{UCC}} |\mathsf{Correct}| > |\mathbb{N}|/2 \wedge n \in \mathsf{Correct} \rightarrow (n \bullet 1 \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \Rightarrow (n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n, ts)) \wedge \Box\neg(n \bullet 1 \downarrow \mathsf{epoch}_{\mathsf{ec}}(n', ts')) \Rightarrow \forall n'.\ n' \in \mathsf{Correct} \rightarrow \Diamond\!\!\!-\Diamond\exists v'.\ (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

From [17] and [18],

(19) $\Gamma' \vdash_{\mathsf{UCC}} n \in \mathsf{Correct} \rightarrow \forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ \Diamond\!\!\!-(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow \Diamond\!\!\!-\Diamond[\Diamond\!\!\!-\Diamond\exists v'.\ (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$

That is,

(20) $\Gamma' \vdash_{\mathsf{UCC}} n \in \mathsf{Correct} \rightarrow \forall n'.n' \in \mathsf{Correct} \rightarrow \exists v.\ \Diamond\!\!\!-(n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v)) \Rightarrow \Diamond\!\!\!-\Diamond[\exists v'.\ (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$

Lemma 74.

$\Gamma \vdash_{\mathsf{UC}}$

$\exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$

$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) \neq \bot) \wedge$

$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$

$\hat{\Diamond}\hat{\Diamond}[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \wedge$

$\hat{\Box}\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self})]$

By II,

(1) $\Gamma' \vdash_{\mathsf{UCC}} (n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge prop(\mathsf{s}(n_l)) \neq \bot \Rightarrow$
$[n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self}]$

By usinng premise and [1],

(2) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\Diamond[[n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self}] \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))]$

By InvL,

(3) $\Gamma' \vdash_{\mathsf{UCC}} started(\mathsf{s}(n_l)) = \mathsf{true} \wedge started(\mathsf{s}'(n_l)) = \mathsf{false} \Rightarrow$
$(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))$

The contra-positive of [3],

(4) $\Gamma' \vdash_{\mathsf{UCC}} \neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l')) \Rightarrow$
$started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true}$

From [2] and [4],

(5) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\Diamond[[n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self}] \wedge$
$\hat{\Box}(started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true})]$

By using induction and PostPre on [5] (considering the conjunction in $started(\mathsf{s}'(n_l)) = \mathsf{true}$ and $\hat{\Box}(started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true})$ for the base case of the induction),

(6) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\Diamond[[n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self}] \wedge$
$\hat{\Box}(started(\mathsf{s}'(n_l)) = \mathsf{true})]$

By InvL,

(7) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge n = n_l \wedge \mathsf{self}) \Rightarrow$
$started(\mathsf{s}'(n)) \neq started(\mathsf{s}(n))$

That is,

(8) $\Gamma' \vdash_{\mathsf{UCC}} (n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$started(\mathsf{s}'(n_l)) \neq started(\mathsf{s}(n_l))$

The contra-pasitive of [8],

(9) $\Gamma' \vdash_{\mathsf{UCC}} started(\mathsf{s}'(n_l)) = started(\mathsf{s}(n_l)) \Rightarrow$
$\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self})$

From [6] and [9]

(10) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$\Diamond[[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n')) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\Diamond[[n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self}] \wedge$
$\hat{\Box}\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self})]$

Lemma 75.

$\Gamma \vdash_{\mathsf{UC}}$
$\exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\hat{\Diamond}\hat{\Diamond}[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \wedge$
$\Box\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self})]$

By InvL,

(1) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge prop(\mathsf{s}(n)) = \bot \wedge n = n_l \wedge \mathsf{self}) \Rightarrow$
$started(\mathsf{s}'(n)) = \mathsf{false}$

That is,

(2) $\Gamma' \vdash_{\mathsf{UCC}} (n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge prop(\mathsf{s}(n_l)) = \bot \wedge \mathsf{self}) \Rightarrow$
$started(\mathsf{s}'(n_l)) = \mathsf{false}$

By using premise and [2],

(3) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$started(\mathsf{s}'(n_l)) = \mathsf{false} \ \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))]]$

By InvL,

(4) $\Gamma' \vdash_{\mathsf{UCC}} started(\mathsf{s}(n_l)) = \mathsf{true} \wedge started(\mathsf{s}'(n_l)) = \mathsf{false} \Rightarrow$
$(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))$

The contra-positive of [4],

(5) $\Gamma' \vdash_{\mathsf{UCC}} \neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l')) \Rightarrow$
$started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true}$

That is,

(6) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$started(\mathsf{s}'(n_l)) = \mathsf{false} \ \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}(started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true})]]$

That is,

(7) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$
$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$
$\hat{\Diamond}[[[started(\mathsf{s}(n_l)) = \mathsf{false} \wedge (prop(\mathsf{s}(n_l)) \neq \bot)] \vee$
$[started(\mathsf{s}(n_l)) = \mathsf{true} \wedge (prop(\mathsf{s}(n_l)) \neq \bot)]] \wedge$

$$\Box(started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true})]$$

By PE,

(8) $\Gamma' \vdash_{\mathsf{UCC}} n_l \in \mathsf{Correct} \wedge prop(\mathsf{s}(n_l)) \neq \bot \wedge started(\mathsf{s}(n_l)) = \mathsf{false} \Rightarrow$
$$(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})$$

By INVSA,

(9) $\Gamma' \vdash_{\mathsf{UCC}} started(\mathsf{s}(n_l)) = \mathsf{true}) \Rightarrow$
$$\hat{\Diamond}(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})$$

From [7], [8] and [9],

(10) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$$
$$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$$
$$\hat{\Diamond}[[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \vee$$
$$\hat{\Diamond}[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})]] \wedge$$
$$\Box(started(\mathsf{s}(n_l)) = \mathsf{false} \vee started(\mathsf{s}'(n_l)) = \mathsf{true})]$$

By using induction and POSTPRE on [10] (considering the conjunction for the base case of induction),

(11) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$$
$$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$$
$$\hat{\Diamond}[[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \vee$$
$$\hat{\Diamond}[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})]] \wedge$$
$$\Box(started(\mathsf{s}'(n_l)) = \mathsf{true})]$$

By adding the premise again,

(12) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$$
$$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$$
$$\hat{\Diamond}[[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \vee$$
$$\hat{\Diamond}[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})]] \wedge$$
$$\Box(started(\mathsf{s}'(n_l)) = \mathsf{true}) \wedge \Box\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l))]$$

By INVL,

(13) $\Gamma' \vdash_{\mathsf{UCC}} (n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}(n_l)) = \mathsf{true} \wedge \mathsf{self}) \Rightarrow$
$$(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l))$$

The contra-pasitive of [13],

(14) $\Gamma' \vdash_{\mathsf{UCC}} \neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \Rightarrow$
$$\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self}) \vee (started(\mathsf{s}(n_l)) = \mathsf{false})$$

By using [12] and [14]

(15) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$$
$$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$$
$$\hat{\Diamond}\hat{\Diamond}[[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \wedge$$
$$\Box(started(\mathsf{s}'(n_l)) = \mathsf{true}) \wedge$$
$$\Box[\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self}) \vee (started(\mathsf{s}(n_l)) = \mathsf{false})]]$$

That is,

(16) $\Gamma' \vdash_{\mathsf{UCC}} \exists ts, n_l.\ n_l \in \mathsf{Correct} \wedge$
$$[(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts, n_l)) \wedge (prop(\mathsf{s}(n_l)) = \bot) \wedge \hat{\Diamond}(prop(\mathsf{s}(n_l)) \neq \bot) \wedge$$
$$\hat{\Box}\neg(n_l \bullet 0 \uparrow \mathsf{startEpoch}_{\mathsf{ech}}(ts', n_l'))] \Rightarrow$$
$$\hat{\Diamond}\hat{\Diamond}[[[(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge started(\mathsf{s}'(n_l)) = \mathsf{true} \wedge \mathsf{self})] \wedge$$

$\Box\neg(n_l \bullet (1, \mathsf{epoch}_{\mathsf{ec}}(n_l, ts)) \in \mathsf{ors} \wedge \mathsf{self})]$

Theorem 27 (UC_2: Validity)**.**
If a node decides v, then v was proposed by some node.
$\Gamma \vdash_{\mathsf{UC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \rightsquigarrow \exists n'.\ (n \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))$
where Γ is defined in Definition 22.

Proof.
By OI′,

(1) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Diamond}(n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self})$

By InvL,

(2) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)$

By EC'_1

(3) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\exists n'.\ \Diamond(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v))$

By OR′,

(4) $\Gamma' \vdash_{\mathsf{UCC}} (n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Diamond}(n' \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self})$

By InvL,

(5) $\Gamma' \vdash_{\mathsf{UCC}} (n' \bullet (1, \mathsf{propose}_{\mathsf{ec}}(v)) \in \mathsf{ors} \wedge \mathsf{self}) \Rightarrow$
$(n' \bullet 1 \downarrow \mathsf{propose}_{\mathsf{uc}}(v))$

From [1] to [5] and Lemma 86,

(6) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\exists n'.\ \Diamond(n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))$

That is,

(7) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \rightsquigarrow$
$\exists n'.\ (n' \bullet \top \downarrow \mathsf{propose}_{\mathsf{uc}}(v))$

Theorem 28 (UC_3: Integrity)**.**
No node decides twice.
$\Gamma \vdash_{\mathsf{UC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow \hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v'))$
where Γ is defined in Definition 22.

Proof.
By OI′,

(1) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Diamond}(n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self})$

By InvL,

(2) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \mathsf{occ}(\mathsf{ois}, \mathsf{decide}_{\mathsf{uc}}(v)) \leq 1$

From [2],

(3) $\Gamma' \vdash_{\mathsf{UCC}} \neg(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$(\mathsf{n} = n \wedge \mathsf{self} \rightarrow \neg(\mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois}))$

From EC'_3, we have

(4) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

By Lemma 108 on [4], we have

(5) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \wedge$
$\hat{\boxminus}\neg(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))$

From [1]-[5], we have

(6) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow \hat{\Diamond}[$
$\mathsf{occ}(\mathsf{ois}, \mathsf{decide}_{\mathsf{uc}}(v)) \leq 1 \wedge$
$\hat{\Box}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow \neg(\mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois})) \wedge$
$\hat{\boxminus}(\mathsf{n} = n \wedge \mathsf{self} \rightarrow \neg(\mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois}))]$

From UniOR on [6]

(7) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow \hat{\Diamond}[$
$(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge$
$\hat{\boxminus}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v))]$

By Lemma 109 on [7], we have

(8) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Box}\neg(n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v))$

Theorem 29 (UC_4: Uniform Agreement)**.**
No two nodes decide differently.
$\Gamma \vdash_{\mathsf{UC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$(v = v')$
where Γ is defined in Definition 22.

Proof.

By OI′,
(1) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Leftrightarrow}(n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self})$
By InvL,
(2) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \mathsf{decide}_{\mathsf{uc}}(v) \in \mathsf{ois} \wedge \mathsf{self}) \Rightarrow$
$n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)$
From [1] and [2],
(3) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \Rightarrow$
$\hat{\Leftrightarrow}(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v))$
From [3] and instantiating v to v' and n to n',
(4) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$\hat{\Leftrightarrow}(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Diamond(\hat{\Leftrightarrow}(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v')))$
By Lemma 112 on [4],
(5) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$\hat{\Leftrightarrow}(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge$
$[\Leftrightarrow(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v')) \vee \Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$
That is,
(6) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$[\hat{\Leftrightarrow}(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Leftrightarrow(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \vee$
$[\hat{\Leftrightarrow}(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$
By Lemma 105 on [6],
(7) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$\Leftrightarrow[(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Leftrightarrow(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \vee$
$\Leftrightarrow[\Leftrightarrow(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \vee$
$\Leftrightarrow[(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$
By Lemma 119 on [7],
(8) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$\Leftrightarrow[\Diamond(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge (n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \vee$
$\Leftrightarrow[(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))] \vee$
$\Leftrightarrow[(n \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v)) \wedge \Diamond(n' \bullet 1 \uparrow \mathsf{decide}_{\mathsf{ec}}(v'))]$
By EC_2' and [8],
(9) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow$
$\Leftrightarrow[v = v']$
That is,
(10) $\Gamma' \vdash_{\mathsf{UCC}} (n \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v)) \wedge \Diamond(n' \bullet \top \uparrow \mathsf{decide}_{\mathsf{uc}}(v')) \Rightarrow (v = v')$

5.4 Temporal Logic

The following subsections show axioms, rules and lemmas from the basic temporal logic of Manna and Pnueli [1] that we use.

5.4.1 Axioms and Inference Rules

Future Axioms.

Axiom 1 (FX0).
$\Box p \to p$

Axiom 2 (FX1).
$\bigcirc \neg p \Leftrightarrow \neg \bigcirc p$

Axiom 3 (FX2).
$\bigcirc (p \to q) \Leftrightarrow (\bigcirc p \to \bigcirc q)$

Axiom 4 (FX3).
$\Box(p \to q) \Rightarrow (\Box p \to \Box q)$

Axiom 5 (FX4).
$\Box p \to \Box \bigcirc p$

Axiom 6 (FX5).
$(p \Rightarrow \bigcirc p) \to (p \Rightarrow \Box p)$

Axiom 7 (FX6).
$p \ \omega \ q \Leftrightarrow [q \vee (p \wedge \bigcirc (p \ \omega \ q))]$

Axiom 8 (FX7).
$\Box p \Rightarrow p \ \omega \ q$

Past Axioms.

Axiom 9 (PX1).
$\ominus p \Rightarrow \tilde{\ominus} p$

Axiom 10 (PX2).
$\tilde{\ominus}(p \to q) \Leftrightarrow (\tilde{\ominus} p \to \tilde{\ominus} q)$

Axiom 11 (PX3).
$\boxminus(p \to q) \Rightarrow (\boxminus p \to \boxminus q)$

Axiom 12 (PX4).
$\Box p \to \Box \tilde{\ominus} p$

Axiom 13 (PX5).
$(p \Rightarrow \tilde{\ominus} p) \to (p \Rightarrow \boxminus p)$

Axiom 14 (PX6).
$p \ \beta \ q \Leftrightarrow (q \vee [p \wedge \tilde{\ominus}(p \ \beta \ q)])$

Axiom 15 (PX7).
$\tilde{\ominus} F$

Mixed Axioms.

Axiom 16 (FX8).
$p \Rightarrow \bigcirc \ominus p$

Axiom 17 (PX8).
$p \Rightarrow \tilde{\ominus} \bigcirc p$

Rule 1 (Generalization).
GEN
$$\frac{p}{\vdash \Box p}$$

Axiom 18 ($\forall \bigcirc$-Comm (Universal commutation)).

$\forall x. \ \bigcirc p(x) \Leftrightarrow \bigcirc \forall x. \ p(x)$

5.4.2 Derived Rules and Lemmas

Lemma 76 (CM1).
$\forall x : \ominus p(x) \Leftrightarrow \ominus \forall x : p(x)$

Lemma 77 (CM2).
$\exists x : \ominus p(x) \Leftrightarrow \ominus \exists x : p(x)$

Lemma 78 (CM3).
$\exists x : \bigcirc p(x) \Leftrightarrow \bigcirc \exists x : p(x)$

Lemma 79 (Rule E-MP (for n = 1)).
$p \Rightarrow q, \Box p \vdash \Box q$

Lemma 80 ($\Rightarrow$T).
$p \Rightarrow q, q \Rightarrow r \vdash p \Rightarrow r$

Lemma 81 ($\bigcirc$M)**.**
$p \Rightarrow q \vdash \bigcirc p \Rightarrow \bigcirc q$

$p \Leftrightarrow q \vdash \bigcirc p \Leftrightarrow \bigcirc q$

Lemma 82 (CI)**.**
$p \Rightarrow \bigcirc p \vdash p \Rightarrow \Box p$

Lemma 83 (T4)**.**
$\Box p \Rightarrow p$

Lemma 84 (T9)**.**
$\Box(p \wedge q) \Leftrightarrow \Box p \wedge \Box q$

Lemma 85 (T25)**.**
$\boxminus(p \wedge q) \Leftrightarrow (\boxminus p \wedge \boxminus q)$
$\hat{\boxminus}(p \wedge q) \Leftrightarrow (\hat{\boxminus} p \wedge \hat{\boxminus} q)$

Lemma 86 (T27)**.**
$\diamondminus p \Leftrightarrow \diamondminus\diamondminus p$
$\hat{\diamondminus}\hat{\diamondminus} p \Rightarrow \hat{\diamondminus} p$
$\diamondminus\hat{\diamondminus} p \Rightarrow \hat{\diamondminus} p$
$\hat{\diamondminus}\diamondminus p \Rightarrow \hat{\diamondminus} p$

Lemma 87 (T11)**.**
$\diamond p \Leftrightarrow \diamond\diamond p$

Lemma 88 ($\diamondminus$T)**.**
$p \Rightarrow \diamondminus q, q \Rightarrow \diamondminus r \vdash p \Rightarrow \diamondminus r$

Lemma 89 ($\diamond$T)**.**
$p \Rightarrow \diamond q, q \Rightarrow \diamond r \vdash p \Rightarrow \diamond r$

Lemma 90 (T26)**.**
$p \Rightarrow \diamondminus p$

Lemma 91.
$\neg\Box p \Leftrightarrow \diamond\neg p$
$\neg\diamond p \Leftrightarrow \Box\neg p$

Lemma 92 (T6)**.**
$\Box p \rightarrow \bigcirc p$

Lemma 93 (CN1)**.**
$\bigcirc(\neg p) \Leftrightarrow \neg\bigcirc p$

Lemma 94 (FP8)**.**
$p \rightarrow \tilde{\ominus}\bigcirc p$

Lemma 95 ($\diamond$M)**.**
$p \Rightarrow q \vdash \diamond p \Rightarrow \diamond q$

Lemma 96 ($\diamondminus$M)**.**
$p \Rightarrow q \vdash \diamondminus p \Rightarrow \diamondminus q$

5.4.3 Extra Temporal Logic Lemmas

Lemma 97.
$((p \Rightarrow q) \wedge p) \rightarrow q$

Proof.
By Axiom 1.

Lemma 98.
$\Diamond\!\!\!-\, p \rightarrow \bigcirc \Diamond\!\!\!-\, p$

Proof.
We assume
(1) $\Diamond\!\!\!-\, p$
We prove
$\bigcirc \Diamond\!\!\!-\, p$
From rule T28 and [1],
$\bigcirc \Diamond\!\!\!-\, p$

Lemma 99.
$p \Rightarrow \Diamond q \ \wedge \ q \Rightarrow r \ \rightarrow$
$p \Rightarrow \Diamond r$

$p \Rightarrow \Diamond\!\!\!-\, q \ \wedge \ q \Rightarrow r \ \rightarrow$
$p \Rightarrow \Diamond\!\!\!-\, r$

$p \Rightarrow \hat{\Diamond}\!\!\!-\, q \ \wedge \ q \Rightarrow r \ \rightarrow$
$p \Rightarrow \hat{\Diamond}\!\!\!-\, r$

Proof.
Immediate from Rule $\Diamond$M ($\Diamond\!\!\!-$M) and Rule $\Rightarrow$T.

Lemma 100.
$p \Rightarrow \Box q \ \wedge \ q \Rightarrow r \ \rightarrow$
$p \Rightarrow \Box r$

Proof.
Immediate from Rule $\Box$M and Rule $\Rightarrow$T.

Lemma 101.
$\Diamond\!\!\!-\, \Box p \Rightarrow p$
$\hat{\Diamond}\!\!\!-\, \hat{\Box} p \Rightarrow p$
$\hat{\Diamond}\!\!\!-\, \Box p \Rightarrow p$

Lemma 102.
$\Diamond\!\!\!-\, \hat{\Box} p \Rightarrow \hat{\Box} p$
$\hat{\Diamond}\!\!\!-\, \Box p \Rightarrow \Box p$
$\Diamond\!\!\!-\, \Box p \Rightarrow \Box p$

Lemma 103.
$p \rightarrow (\hat{\boxminus}(p \Rightarrow \bigcirc p) \Rightarrow p)$

Lemma 104.
$p \Rightarrow q \Rightarrow r$
$\leftrightarrow$
$(\Diamond\!\!\!-\, p \wedge q) \Rightarrow r$
$\leftrightarrow$
$(p \wedge \Diamond q) \Rightarrow \Diamond r$

Lemma 105.
$(\Diamond p \wedge \Diamond q) \Rightarrow$
$\quad \Diamond(p \wedge \Diamond q) \vee$
$\quad \Diamond(q \wedge \Diamond p)$

$(\Diamond\!\!\!-\, p \wedge \Diamond\!\!\!-\, q) \Rightarrow$
$\quad \Diamond\!\!\!-\,(p \wedge \Diamond\!\!\!-\, q) \vee$
$\quad \Diamond\!\!\!-\,(q \wedge \Diamond\!\!\!-\, p)$

$\Diamond\!\!\!-\, p \wedge \Diamond q \Rightarrow$
$\quad \Diamond\!\!\!-\,(p \wedge \Diamond q)$

Lemma 106.
$(\Diamond p \wedge \hat{\Box} \neg p) \Rightarrow p$
$(\Diamond\!\!\!-\, p \wedge \hat{\boxminus} \neg p) \Rightarrow p$

Lemma 107.
$\Diamond\!\!\!-\, p \Rightarrow \Box \Diamond\!\!\!-\, p$

Lemma 108.
$(p \Rightarrow \hat{\Box}\neg p) \rightarrow$
$(p \Rightarrow \hat{\boxminus}\neg p)$
and
$(p \Rightarrow \hat{\boxminus}\neg p) \rightarrow$
$(p \Rightarrow \hat{\Box}\neg p)$

Proof.
We assume
(2) $(p \Rightarrow \hat{\Box}\neg p)$
We prove
$(p \Rightarrow \hat{\boxminus}\neg p)$
That is,
$\diamondminus p \Rightarrow \neg p$

From [2]
(3) $(\hat{\diamondminus} p \Rightarrow \diamondminus\hat{\Box}\neg p)$
By Lemma 101,
(4) $\hat{\diamondminus}\hat{\Box}\neg p \Rightarrow \neg p$
From [3] and [4],
(5) $(\hat{\diamondminus} p \Rightarrow \neg p)$

Lemma 109.
$p \Rightarrow \diamondminus(p \Rightarrow q) \rightarrow$
$(p \Rightarrow q)$

Lemma 110.
$\hat{\boxminus} p \Rightarrow p \rightarrow$
$\Box p$

Lemma 111.
$\hat{\boxminus} p \wedge \hat{\boxminus} q \Rightarrow p \wedge q \rightarrow$
$\Box(p \wedge q)$

Proof.
Immediate from Lemma 110 and Lemma 93.

Lemma 112.
$\diamondminus\Diamond p \Leftrightarrow \Diamond p \vee \diamondminus p$
$\hat{\diamondminus}\Diamond p \Rightarrow \Diamond p \vee \diamondminus p$
$\Diamond\diamondminus p \Rightarrow \Diamond p \vee \diamondminus p$
$\Diamond\hat{\diamondminus} p \Rightarrow \Diamond p \vee \diamondminus p$
$\hat{\diamondminus}\hat{\Diamond} p \Rightarrow \Diamond p \vee \diamondminus p$

Lemma 113.
$\diamondminus p \vee \Diamond p \Leftrightarrow \hat{\diamondminus} p \vee \Diamond p$

Lemma 114.
$\diamondminus p \Leftrightarrow \bigcirc\hat{\diamondminus} p$

Lemma 115.
$\Diamond p \wedge \Box q \Rightarrow \Diamond(p \wedge \Box q)$

Lemma 116.
$\hat{\Diamond}\Diamond p \Rightarrow \Diamond p$

Lemma 117.
$\neg\hat{\diamondminus} p \Rightarrow \hat{\boxminus}\neg p$

Lemma 118.
$\hat{\boxminus} p \Leftrightarrow \boxminus\tilde{\ominus} p$

Lemma 119.
$p \wedge \hat{\diamondminus} q \Rightarrow$
$\hat{\diamondminus}(q \wedge \Diamond p)$

Lemma 120.
$\boxminus p \wedge \Box p \Rightarrow$
$\boxminus \hat{\boxminus} p \wedge \Box \hat{\boxminus} p$

Lemma 121.
$\hat{\diamondminus} \bigcirc p \Leftrightarrow \diamondminus p$

Lemma 122.
$p \wedge \Diamond \hat{\Box} \neg p \Rightarrow$
$\Diamond (p \wedge \hat{\Box} \neg p)$

Lemma 123.
$\Diamond p \wedge \hat{\Diamond} \Box q \Rightarrow$
$\Diamond (p \wedge \hat{\Diamond} \Box q) \vee$
$\Diamond (p \wedge \Box q)$

Lemma 124.
$\Diamond p \wedge \hat{\Diamond} \Box q \Rightarrow$
$\Diamond (\Diamond p \wedge \hat{\Diamond} \Box q) \vee$
$\Diamond (p \wedge \Box q)$
$\rightarrow$
$\Diamond p \wedge \hat{\Diamond} \Box q \Rightarrow \Diamond (p \wedge \Box q)$

Lemma 125.
$p \Rightarrow \Box p$
$\rightarrow$
$\neg p \Rightarrow \hat{\boxminus} \neg p$

Lemma 126.
$\diamondminus \Diamond \hat{\Box} p \rightarrow \Diamond \hat{\Box} p$

Lemma 127.
$\Diamond p \wedge \Diamond q \Rightarrow \Diamond (\diamondminus \Diamond p \wedge q)$

References

[1] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag New York, Inc., New York, NY, USA, 1992.