Formal Verification of RDMA Failover Impossibility

We formalize and mechanically verify three impossibility theorems for transparent RDMA failover using the Rocq proof assistant (formerly Coq). All proofs are available at github.com/taooceros/shift-verification.

Theorem 1: Indistinguishability of Packet Loss and ACK Loss

Definition (Sender View)**.** Let $\mathcal{T}$ be an execution trace. The *sender view* $\sigma(\mathcal{T})$ is the projection containing only sender-observable events: operation sends, completions, and timeouts.

Definition (Transparent Overlay)**.** A failover mechanism is *transparent* if its retransmission decision $D : \sigma(\mathcal{T}) \times \text{Op} \to \{0, 1\}$ depends only on the sender view.

Theorem (Impossibility of Safe Retransmission)**.** For any transparent overlay D, there exist executions $\mathcal{T}_1$ (packet lost) and $\mathcal{T}_2$ (ACK lost, memory reused) such that:

$$\sigma(\mathcal{T}_1) = \sigma(\mathcal{T}_2) \tag{1}$$

but safety requires $D(\sigma(\mathcal{T}_1)) = 1$ (retransmit) while $D(\sigma(\mathcal{T}_2)) = 0$ (do not retransmit).

Proof. We construct two traces with identical sender views but opposite correctness requirements:

$\mathcal{T}_1$: $[\text{Send}(W_D), \text{PacketLost}(W_D), \text{Timeout}(W_D)]$
$\mathcal{T}_2$: $[\text{Send}(W_D), \text{Receive}, \text{Execute}, \text{AppConsume}, \text{AppReuse}(V'), \text{AckLost}, \text{Timeout}(W_D)]$

Both produce sender view $[\text{ObsSent}(W_D), \text{ObsTimeout}(W_D)]$. In $\mathcal{T}_1$, the operation was never executed (liveness requires retry). In $\mathcal{T}_2$, the operation executed and memory was reused with value $V' \neq V_1$ (safety forbids retry). Since D is a function, $D(\sigma(\mathcal{T}_1)) = D(\sigma(\mathcal{T}_2))$, contradicting the requirements. □

Theorem 2: Non-Idempotency of Atomic Operations

Theorem (FADD Non-Idempotency)**.** For any $\delta > 0$ and memory state m, FADD is not idempotent:

$$\text{exec}_{\text{FADD}\left(\text{exec}_{\text{FADD}(m,a,\delta)},a,\delta\right)} \neq \text{exec}_{\text{FADD}(m,a,\delta)} \tag{2}$$

Proof. Let $m[a] = v$. After one FADD: $m'[a] = v + \delta$. After retry: $m''[a] = v + 2\delta$. Since $\delta > 0$, we have $v + \delta \neq v + 2\delta$. □

Theorem (CAS Retry Violation)**.** Under concurrent modification, a CAS retry can succeed twice, violating at-most-once semantics.

Proof. Consider sender S with $\text{CAS}(a, 0, 1)$ and concurrent process P with $\text{CAS}(a, 1, 0)$:

State 0: $m[a] = 0$
State 1: $S.\text{CAS}(0, 1)$ succeeds $\to m[a] = 1$
State 2: $P.\text{CAS}(1, 0)$ succeeds $\to m[a] = 0$
State 3: S retries $\text{CAS}(0, 1) \to$ succeeds again!

S's single CAS executed twice, and P's successful modification was silently overwritten. □

Theorem 3: Consensus Hierarchy Barrier

We prove that failover coordination is equivalent to 2-process consensus, which read-only verification cannot solve.

Unified Observation Constraint Framework

Definition (Observation Constraint)**.** Each synchronization primitive defines a constraint on what protocols can observe:

Primitive	**Constraint**
Register	valid_{rw} : $\text{obs}(\text{exec}, i)$ depends only on writes before i
FADD	$\text{valid}_{\text{fadd}}$: $\text{obs}(\text{exec}, i)$ depends only on $\{j : j \text{ before } i\}$ (set, not order)
CAS	$\text{valid}_{\text{cas}}$: $\text{obs}(\text{exec}, i) = \text{winner}(\text{exec})$ (first process)

The constraints are *derived* from primitive semantics:

- **Register**: Reads are invisible; only writes affect observable state
- **FADD**: Returns sum of prior deltas; $\delta_0 + \delta_1 = \delta_1 + \delta_0$
- **CAS**: First CAS to sentinel wins; all subsequent fail; all read same value

Consensus Number Verification

Definition (Consensus Number)**.** $\text{CN}(X) = n$ iff X can solve n-consensus but not $(n+1)$-consensus.

Lemma (Register CN = 1)**.** For any observation function satisfying valid_{rw}, solo executions [0] and [1] produce identical observations (both have empty prior write history) but require decisions 0 and 1 respectively.

Lemma (FADD CN = 2)**.** For any observation function satisfying $\text{valid}_{\text{fadd}}$, executions $[0, 1, 2]$ and $[1, 0, 2]$ are indistinguishable to process 2 (both see $\{0, 1\}$ ran before), but require decisions 0 and 1.

Lemma (CAS CN = ∞)**.** Any observation function satisfying $\text{valid}_{\text{cas}}$ gives $\text{obs}(\text{exec}, i) = \text{winner}(\text{exec})$. Different winners $\rightarrow$ different observations $\rightarrow$ always distinguishable.

Failover as 2-Consensus

Definition (Verification Mechanism)**.** A verification mechanism $V : \text{Memory} \rightarrow \{\text{Commit}, \text{Abort}\}$ decides whether to retry based on reading remote memory.

Theorem (Failover-Consensus Isomorphism)**.** The failover problem is structurally isomorphic to 2-process consensus:

2-Consensus	**Failover**
Process 0 input	CAS executed (Commit)
Process 1 input	CAS not executed (Abort)
Observation	Memory state m
Decision function	Verification mechanism V
Solo executions indistinguishable	ABA: both histories yield same m

Theorem (Transparent Failover Impossibility)**.** No verification mechanism $V : \text{Memory} \rightarrow \text{bool}$ can solve failover.

Proof. By the ABA problem, there exist histories H_0 (CAS not executed) and H_1 (CAS executed, then ABA reset) with identical final memory: $\text{mem}(H_0) = \text{mem}(H_1) = m$.

Correctness requires $V(m) = \text{Abort}$ for H_0 and $V(m) = \text{Commit}$ for H_1. But V is a function, so $V(\text{mem}(H_0)) = V(\text{mem}(H_1))$. Contradiction.

This matches the register $\text{CN} = 1$ proof: V satisfies valid_{rw} (it only reads), and the ABA histories correspond to solo executions with identical "prior write state." □

Theorem (Main Result)**.** Transparent RDMA failover for atomic operations is impossible because:
1. Failover requires solving 2-consensus
2. Transparency limits verification to read-only operations
3. $\text{CN}(\text{Register}) = 1 < 2$
4. By Herlihy's hierarchy, CN=1 primitives cannot solve 2-consensus

Mechanization

Component	Lines	Key Theorems
Core definitions	400	Memory model, RDMA operations, traces
Theorem 1	200	`impossibility_safe_retransmission`
Theorem 2	300	`fadd_not_idempotent`, `cas_double_success`
Theorem 3	1200	`register_cn_1_verified`, `fadd_cn_2_verified`, `valid_cas_no_ambiguity`, `transparent_cas_failover_impossible`

Table 1: Rocq formalization statistics

All proofs are constructive and fully mechanized in Rocq 9.0. The consensus number framework provides a unified treatment where each primitive's limitation is derived from its operational semantics, and the failover impossibility follows as a direct consequence of Herlihy's hierarchy applied to the structural isomorphism between failover and 2-consensus.