A Formal Foundation of Transparent Failover Limits

This appendix provides the formal proofs supporting the design decisions in Section 3. The formalization distinguishes between what is achievable via transparent mechanisms (SHIFT's best-effort path) and what requires non-transparent coordination (SHIFT's atomic rejection and 2-way handshake).

A.1 A.1 System Model and Transparency

We model the RDMA system as a state machine.

Definition (State)**.** The system state $S = (M, Q_R)$ consists of:
- M: The receiver's memory state (Addr $\to$ Val).
- Q_R: The state of the Receive Queue (list of available Receive WQEs).

Definition (Operations)**.**
- Write(addr, val): Updates $M[\text{addr}] \leftarrow \text{val}$. Idempotent if val is constant.
- Send(val): Consumes head(Q_R), writes val to the buffer specified by the WQE.
- Atomic(addr, op): Updates $M[\text{addr}]$ based on current value.

Definition (Transparency)**.** A failover mechanism D is *transparent* if its decision to retransmit an operation op is a function solely of the sender's local history H_S:

$$D(H_S, \text{op}) \to \{\text{Retry}, \text{Abort}\}$$

It cannot observe the receiver's state S directly.

A.2 A.2 The Queue Sliding Problem (Two-Sided Operations)

While non-idempotency of Atomics is well-understood, we formalize a subtle but critical failure mode for two-sided operations: "Queue Sliding."

Theorem (Queue Consumption Non-Idempotency)**.** *Retrying a generic* Send *operation is not idempotent with respect to Resource Consumption, even if the data payload is idempotent.*

Proof. Let the receiver queue be $Q_R = [R_1, R_2, ...]$, where R_i points to buffer B_i. Consider the trace for sending message m_1:

Trace 1 (Success):
1. Sender sends m_1.
2. Receiver consumes R_1, writes m_1 to B_1. $Q_{R'} = [R_2, ...]$.
3. ACK is generated but lost.

Trace 2 (Retry):
1. Sender times out (Transparency Barrier: looks like packet loss).
2. Sender retries m_1.
3. Receiver receives m_1.
4. Receiver consumes R_2, writes m_1 to B_2. $Q_{R''} = [R_3, ...]$.

Result:
- The message m_1 is duplicated in B_1 and B_2.
- Crucially, R_2 (intended for message m_2) is consumed by the stale m_1.
- When m_2 finally arrives, it will land in B_3 (via R_3) or fail if Q_R is empty.
- The data stream is permanently shifted relative to the buffer stream ("Queue Sliding").

Thus, transparent retry of SEND operations violates correctness by corrupting the mapping between messages and receive buffers. □

Implication for SHIFT: This proves that transparently retrying SEND/RECV is impossible without creating data alignment corruption. This necessitates SHIFT's **2-way Handshake** (Section 3.2), which

explicitly resynchronizes the sequence numbers and queue indices, breaking transparency to ensure correctness.

A.3 A.3 The Consensus Barrier

We prove that verifying whether an operation executed (to decide on safe retry) is equivalent to solving consensus.

Theorem (Execution Verification is 2-Consensus)**.** *Let P_S be the sender (Verifier) and P_E be the environment (deciding if packet loss occurred). Distinguishing "Packet Lost" from "ACK Lost" is equivalent to 2-process Consensus.*

Proof. By Herlihy's hierarchy:

1. Read/Write registers have Consensus Number $\text{CN} = 1$.
2. To solve 2-Consensus (i.e., agree on "Did it commit?"), one needs a primitive with $\text{CN} \geq 2$ (like FADD, CAS, or a Queue).
3. Transparent verification allows only **Reading** remote state (or observing timeouts).
4. Since Reads have $\text{CN} = 1$, the sender cannot unilaterally determine the system state consistent with the receiver's view in the presence of faults.

Therefore, no algorithm based solely on reading remote memory or waiting for timeouts can correctly resolve the ambiguity between "Execution Failed" and "Reporting Failed" for non-idempotent operations. □

A.4 A.4 Mapping Theory to SHIFT Design

The impossibility theorems define the boundary in Figure 1 of the paper.

Operation	**Theoretical Property**	**SHIFT Handling**
WRITE	Idempotent Data, No Queue Side-effect	**Transparent Retry**. Safe because memory overwrite with same data is harmless (assuming protocol coordination).
ATOMIC	Non-Idempotent Data	**Abort**. Retry corrupts state (Thm 2). Verification impossible (Thm 3).
SEND	Queue Side-effect (Queue Sliding)	**Handshake**. Transparent retry causes sliding (Thm "Queue Sliding"). Requires explicit synchronization between SHIFTLib instances (hidden from app).

The **RNR (Receive Not Ready)** error does not solve the consensus problem. An RNR error confirms "Packet Arrived, No Buffer." However, the absence of RNR does not confirm "Packet Arrived, Buffer Consumed" versus "Packet Lost." The ambiguity remains, sustaining the validity of Theorem 1.