Scalable Data Plane Architecture for BIER

draft-li-bier-scalable-data-plane-00

Zhiqiang Li ,Zongpeng Du,
Wei Cheng,Junjie Wang,Guoying Zhang

IETF 125, Mar 19th,2026

Why Revisit BIER Forwarding Now?

- BIER forwards multicast traffic without building explicit multicast trees in the transit network.
- The BFIR encodes the destination set directly in the BitString, so the core stays stateless on a per-flow basis.
- That model is attractive for large domains, multi-tenant data centers, and AI/ML clusters where multicast fan-out can be bursty and short-lived.
- The challenge is that larger deployments often require long BitStrings—256 bits or more—which stresses high-speed forwarding hardware.

Quick Retrospect for BIER Forwarding

- Each bit in the BitString represents one egress BFR, or BFER.
- The BFIR sets the bits for the intended destination set and imposes the BIER encapsulation.
- Transit BFRs consult the Bit Index Forwarding Table, or BIFT, to determine next hops and forwarding masks.
- The BIFT-id identifies the forwarding context; in non-MPLS BIER it uniquely maps to <SD, SI, BSL> across the domain.

Problem Statement

RFC 8279 procedure

1 Scan each set bit in the received BitString

2 Look up the bit in the BIFT and identify the BFR neighbor

3 Accumulate the interface Forwarding BitMask

4 Emit one copy per relevant egress interface

Latency pressure

- Processing cost grows with BSL.
- Scanning 256 or more positions is expensive at 100GE / 400GE line rate.
- A direct per-bit pipeline can stretch stage budget and determinism.

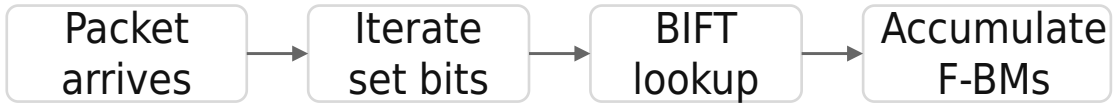
Table pressure

- A standard BIFT keeps one entry per BFR-id.
- Each entry stores a next-hop identifier plus a full-length F-BM.
- For BSL = 256, that means 256 entries each carrying a 256-bit mask for one <SD, SI> context.

Design Object and Key Idea

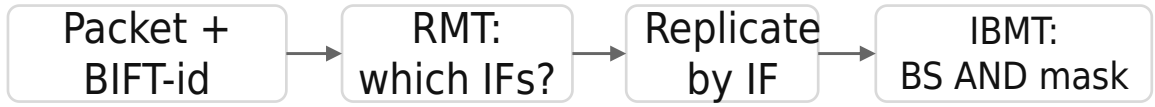
Pre-compute interface grouping once, then filter the BitString per interface.

Conceptual bit-centric flow



Cost follows BSL

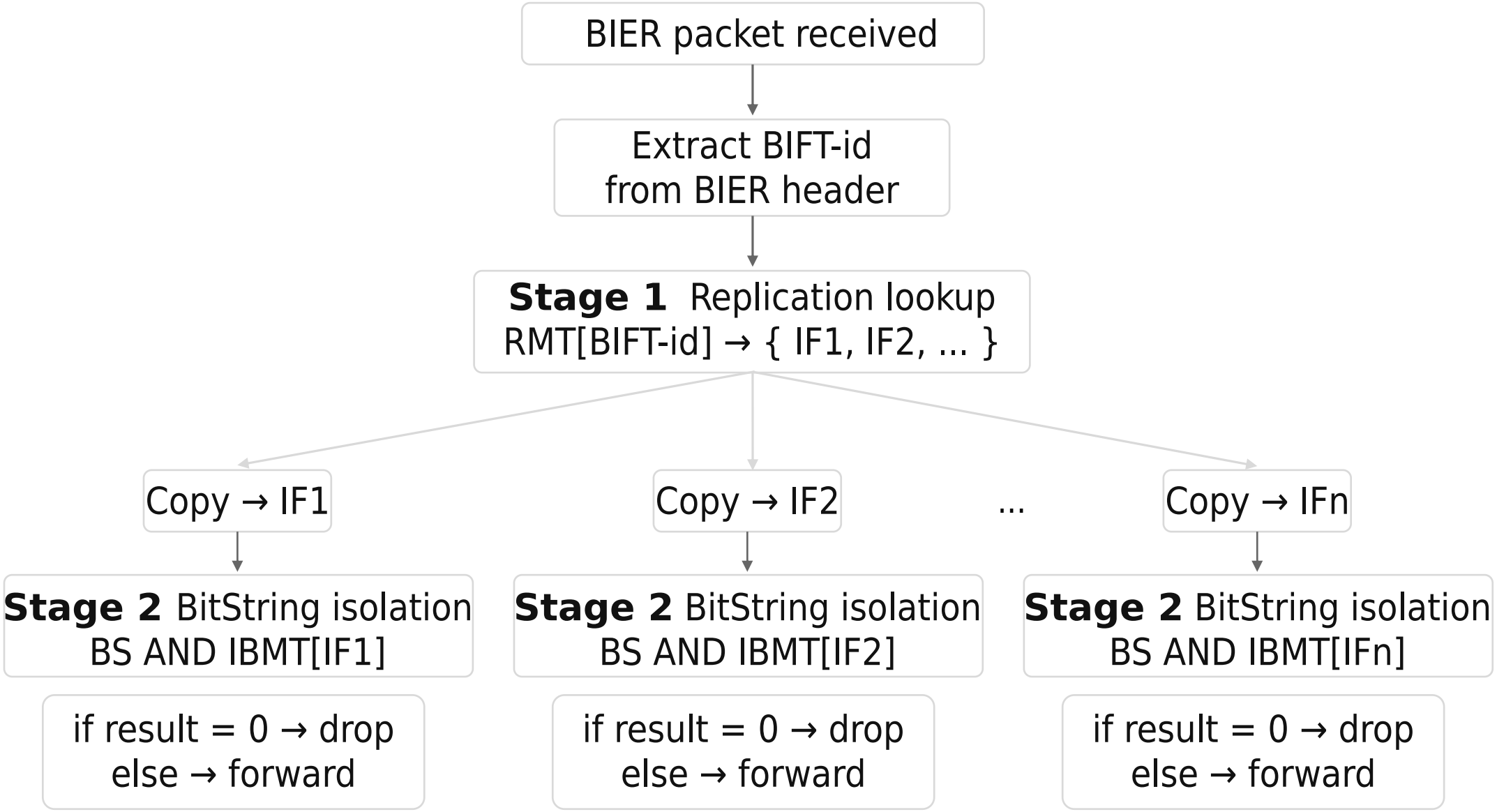
Proposed interface-centric flow



Cost follows number of interfaces

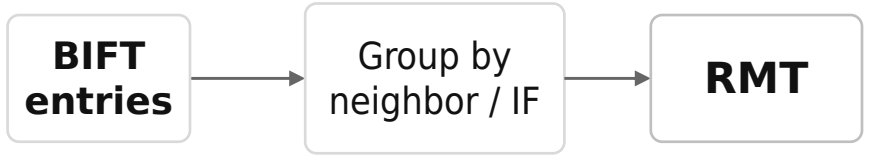
- Semantic equivalence is mandatory: the same replicas and the same per-interface BitStrings must be produced.
- No change to the BIER header, BIFT-id meaning, or protocol signaling.
- The improvement comes from decoupling “who gets a copy” from “which bits survive on that copy.”

Architecture Overview



Replication Member Table (RMT)

- The RMT maps a Replication Group Identifier to the set of egress interfaces that have at least one downstream BFER.
- In practice, the Replication Group Identifier may be derived directly from the BIFT-id.
- The control plane populates the RMT by grouping BIFT entries according to their unique BFR neighbors and their associated interfaces.
- The table size scales with the number of forwarding groups, not with every individual bit position.



Replication Group ID	Egress Interface List
RG_1	{ IF_a, IF_b }
RG_2	{ IF_a, IF_c, IF_d }
RG_3	{ IF_b }

RMT lookup is independent of the packet BitString

Interface BitMask Table (IBMT)

- The IBMT maps each egress interface to one Forwarding BitMask, or F-BM.
- That F-BM is the logical OR of all downstream BFER bit positions reachable via that interface.
- At run time, the BFR computes BitString_new = BitString_original AND F-BM(IF_k).
- Entries are updated whenever BIER topology or BIFT results changed.

Egress IF	Forwarding BitMask
IF_a	11000000 ... 00
IF_b	00110000 ... 00
IF_c	00001100 ... 00
IF_d	00000011 ... 00

Optional scoping:
per <SD, BSL> or aggregated
if mapping is stable

F-BM(IF_k) = OR { bit(BFR-id_ j) }
for all downstream BFER_ j via IF_k

Forwarding Procedure

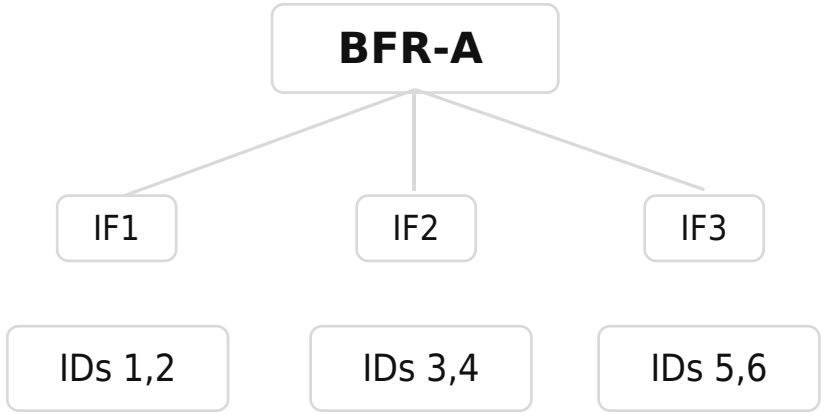
1 Resolve group → **2 Replicate by IF** → **3 Isolate BitString** → **4 Decide and send**

1 Resolve group	2 Replicate by IF	3 Isolate BitString	4 Decide and send
Extract the BIFT-id, derive the Replication Group Identifier, and query the RMT	Create one packet replica for each egress interface returned by the RMT, Do not edit the BitString yet	For each replica, retrieve IBMT[IF_k] and compute BitString_new = BitString_original AND F-BM(IF_k)	If BitString_new is zero, discard. Otherwise overwrite the BitString and transmit on IF_k

Invariant: the local optimization MUST produce exactly the same per-interface copies as a correct RFC 8279 implementation

- If no RMT entry matches the BIFT-id, the packet is discarded and the draft recommends logging an error
- All other BIER header processing, such as TTL handling or label updates, follows the normal RFC procedures

Example A All Three Interfaces Remain Active



RMT[RG_X] = { IF1, IF2, IF3 }

IBMT
IF1 = 00000011
IF2 = 00001100
IF3 = 00110000

Input BitString = 00101010
(destinations 2, 4, 6)

Replica 1 on IF1
00101010 AND 00000011 = 00000010
→ Forward with BS = 00000010

Replica 2 on IF2
00101010 AND 00001100 = 00001000
→ Forward with BS = 00001000

Replica 3 on IF3
00101010 AND 00110000 = 00100000
→ Forward with BS = 00100000

Example B Zero-result Causes a Clean Discard

Input BitString = 00101000
(destinations 4 and 6 only)

IF1: 00101000 AND 00000011 = 00000000
→ Discard

IF2: 00101000 AND 00001100 = 00001000
→ Forward

IF3: 00101000 AND 00110000 = 00100000
→ Forward

Why this matters operationally

- No bandwidth is wasted on interfaces that have no relevant downstream recipients for this packet.
- The zero-result rule preserves correctness even though replication happens before interface-specific filtering.
- This is the key isolation property: replication scope is broad first, then narrowed exactly by the AND operation.

Broad candidate set from RMT
+ exact per-interface mask from IBMT
= same output as per-bit forwarding

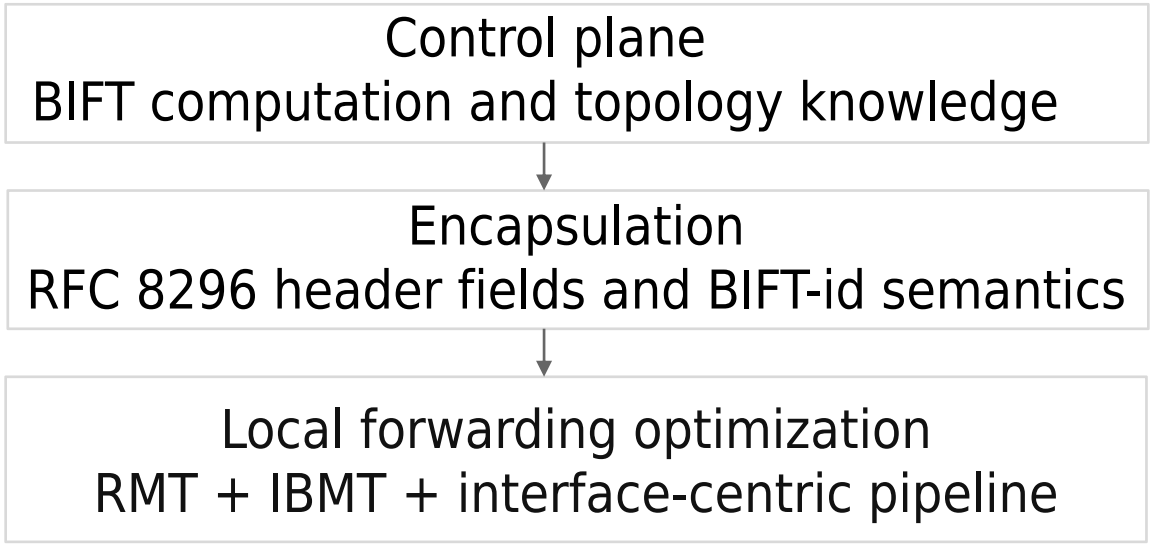
Relationship with RFC 8279

What stays the same

- BIER header format and encapsulation remain unchanged.
- BIFT-id meaning and ordinary BIER processing stay unchanged.
- The output replicas and per-interface BitStrings must be identical to RFC 8279 forwarding.

What changes locally

- The router may organize local forwarding state as RMT + IBMT.
- The data plane iterates over interfaces instead of bit positions.
- This is a local implementation optimization, not a new protocol.

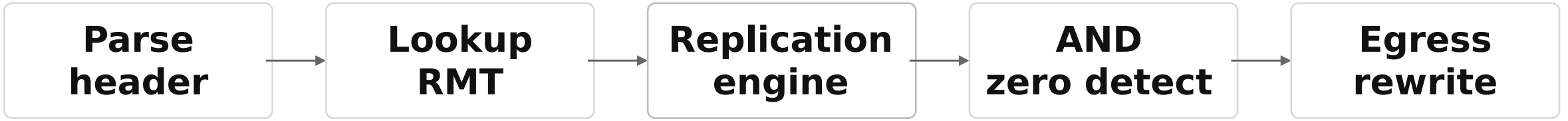


Design Principles

Deployable without any new signaling on the wire

Illustrative Hardware Mapping for Data Planes

A practical forwarding pipeline view(Illustrative only)



Fixed-stage work per packet copy is driven mainly by interface fan-out, not by bit-string length

Likely implementation benefit

- Deterministic stage count
- Compact interface-oriented tables
- Natural fit for replication hardware

Key assumption behind the gain

- A typical BFR has far fewer egress interfaces than available BIER bit positions.
- That is why grouping by interface reduces the amount of per-packet work.

Operational interpretation

- RMT and IBMT are updated by the same topology/control-plane changes that already affect the BIFT.
- No changes are required on remote nodes to benefit locally.

Deployment Considerations

Control-plane’s role

- RMT entries are derived from BIFT computation results.
- IBMT entries must be refreshed whenever topology or BIFT state changes.
- Protection of RMT and IBMT should follow the same security model as the BIFT.

Packet handling edge cases

- No matching RMT entry, discard and log.
- Zero result after AND, discard that replica only.
- Non-zero result,continue with ordinary RFC 8279 / RFC 8296 processing.

Where the gain is strongest

- Large BitString length
- Relatively small interface fan-out
- Hardware data planes that prefer fixed, simple bitwise operations

Summary & Next Step

- The proposal keeps BIER semantics, encapsulation, and signaling unchanged.
- It replaces per-bit forwarding work with an interface-centric two-phase pipeline built on RMT and IBMT.
- The resulting data plane scales with egress interfaces rather than with BitString length, which is a better fit for high-speed hardware.

Any comments are welcomed, Thanks!