ZEROSTACK TECHNICAL WHITEPAPER SERIES · PART 1

SMLIGHT

Scaling Zigbee 3.0 / R23 Beyond the 100-Node Wall

Deterministic *O(1)* Architecture vs. Legacy *O(N)* Stack Scaling on ARM Cortex-M4F

Published by: SMLIGHT Engineering
Target Hardware: TI SimpleLink™ CC1352P7 (48 MHz Cortex®-M4F, 144 KB SRAM, 704 KB Flash)
Dataset: 31 October 2026 — Legacy Reference Stack vs. ZeroStack, same board, 115,200 baud, medians

1. Executive Summary

Most commercial Zigbee coordinators share a hidden architectural ceiling. While marketing datasheets routinely advertise support for 200 to 300+ direct and routed devices, the underlying firmware stacks were architected two decades ago for MCUs with 8 KB of RAM and 20-node networks. To conserve scarce RAM in that era, reference stacks pushed cryptographic keys, link-key tables, and network state into append-only Non-Volatile Memory (NVM) flash logs, and relied on unindexed linear scans and dynamic first-fit heaps for runtime packet processing.

In small meshes (10–30 devices), these *O(N)* algorithms hide behind the raw clock speed of a modern 48 MHz Cortex-M4F microcontroller. However, as a network scales to 150–300 nodes and the non-volatile journal accumulates routine state updates, these linear costs compound multiplicatively on the critical packet path:

- **Per-Packet Flash Log Scans:** Processing a single Network (NWK) secured frame on a mature legacy coordinator can stall the CPU for **over 11 milliseconds** (`533,156` CPU cycles) simply walking flash headers to retrieve the active network key.
- **Join & Interview Storms:** Locating a Trust Center Link Key (TCLK) in a 300-device table takes up to **12.9 milliseconds** (`620,396` cycles) for a key whose record is old in the flash log, and **31 milliseconds** (`1,486,228` cycles) to establish that a device has no key yet—blocking the stack task during multi-device commissioning.
- **Flash Wear & Inline Compaction Stalls:** Redundant state writes append continuously to flash, triggering **163.9-millisecond** inline page-compaction freezes every ~69 writes.

ZeroStack is a ground-up architectural re-engineering of the coordinator runtime designed to eliminate *O(N)* flash and memory bottlenecks on the hot packet path. By replacing per-packet flash walks, unindexed table scans, and linear heap traversals with deterministic *O(1)* runtime primitives, ZeroStack delivers **134x–4,559x latency reductions** on the deepest hot paths, on identical silicon, while producing byte-identical ciphertext, MICs and frame counters in every security benchmark cell.

Headline Hardware Benchmark Summary (48 MHz Cortex-M4F)

Subsystem / Hot-Path Operation	Legacy Reference Stack	ZeroStack Architecture	Measured Improvement
Trust Center Link Key (TCLK) Lookup *(300-entry table, key not present)*	`1,486,228` cycles (`30.96 ms`)	**`326` cycles (`6.8 µs`)**	**4,559x faster** (flat wrt flash log age)
Per-Frame NWK Security (Encrypt) *(Mature flash log, D=150)*	`533,156` cycles (`11.11 ms`)	**`3,968` cycles (`82.7 µs`)**	**134.4x faster** (*O(1)*, 0 flash reads)
Per-Frame NWK Security (Decrypt) *(Mature flash log, D=150)*	`532,182` cycles (`11.09 ms`)	**`3,824` cycles (`79.7 µs`)**	**139.2x faster** (*O(1)*, 0 flash reads)
Per-Frame APS TX Security *(300-peer table, across flash depth)*	`52,909` – `409,301` cycles (`8.53 ms`)	**`5,410` – `5,424` cycles (`0.113 ms`)**	**Up to 75.5x faster** (Immune to log depth)
NVM Key Read Latency *(Depth D = 0 → 150 records)*	`2,347` → `64,817` cycles (`1.35 ms`)	**`302` – `303` cycles (`6.3 µs`)**	**213.9x faster**
Idempotent NVM State Write *(Unchanged 100 B record)*	`100,714` cycles (Appends to flash)	**`3,770` cycles (0 bytes written)**	**26.7x faster + 0% flash wear**
NVM Compaction Frequency *(Identical write workload)*	First at **write 69** (`163.9 ms` stall)	First at **write 1,013** (`13.7 ms` stall)	**14.7x lower compaction rate**
Dynamic Memory Allocation (`malloc`) *(64 heap holes)*	`2,550` cycles (Linear hole walk)	**`218` cycles (Constant time)**	**11.7x faster** (*O(1)* allocation)

Subsystem / Hot-Path Operation	Legacy Reference Stack	ZeroStack Architecture	Measured Improvement
Timer Start Latency *(39 active timers, fragmented heap)*	`10,034` cycles (`209.0 µs`)	**`1,448` cycles (`30.2 µs`)**	**6.9x faster** (heap-independent)
Inter-Task Event Enqueue (`msgSend`) *(64 queued events)*	`935` cycles (*O(Q)* list walk)	**`322` cycles (*O(1)* flat)**	**2.9x faster**

2. Measurement Methodology & Zero-Trust Hardware Harness

Microcontroller benchmarks are frequently distorted by host-side serial jitter, synthetic stubs, or compiler dead-code elimination. To ensure every measurement in this paper reflects real production execution, all tests were conducted on physical **TI CC1352P7** hardware running at **48.0 MHz** under a strict differential harness:

1. **Cycle-Accurate Hardware Counter (`DWT->CYCCNT`):** Every timing measurement is captured directly from the ARM Cortex-M4 Data Watchpoint and Trace (`DWT->CYCCNT`) 32-bit hardware cycle counter (`20.83 ns` resolution at 48 MHz) immediately bracketing the target stack operation.
2. **Paired Single-Board Differential Testing:** Both the unmodified Legacy Reference Stack and ZeroStack execute the exact same stimulus sequence across identical multi-dimensional axes:
   - *S* **(Peer Slot Index,** *0 … 299***):** Position of the target device in the 300-node coordinator tables.
   - *D* **(NVM Log Depth,** *0 … 150+***):** Number of newer non-volatile records appended to the flash journal after initial key provisioning.
   - *k* **(Heap Fragmentation,** *0 … 64* **holes):** Number of non-coalesced small allocations fragmenting the runtime heap.
   - *T, Q* **(Active Timers** *0 … 39* **& Queue Depth** *0 … 64***):** Concurrent load on the scheduler.
3. **Cryptographic & Wire Equivalence Oracles:** Performance optimizations are meaningless if they alter protocol behavior. For every security benchmark cell across all slots *S* and flash depths *D*, our external test oracle verifies that ZeroStack and the reference stack produce **byte-for-byte identical IEEE 802.15.4 / Zigbee PRO R23 ciphertexts, 32-bit Message Integrity Codes (MICs), and outgoing frame counters**. In the latest ZeroStack run, every ciphertext and MIC also matched an independent AES-CCM* implementation running outside the firmware.

3. Deep-Dive Results by Subsystem

3.1 Non-Volatile Memory (NVM) & Per-Packet Key Retrieval

THE LEGACY BOTTLENECK: LOG-STRUCTURED FLASH WALKS ON THE PACKET PATH

Embedded flash memory cannot be overwritten in place without erasing an entire hardware sector (typically 8 KB). Consequently, Zigbee stacks use a log-structured flash file system: every update to a network attribute, frame counter, route record, or device table entry appends a new record header and payload to the end of the active flash page. To locate an item, a standard driver scans backward from the write cursor, reading and validating flash record headers one by one until it encounters the matching identifier.

In legacy reference architectures, **each packet's security processing triggers a flash lookup for the active key**. As normal coordinator operation appends newer records to the flash log (depth *D*), every secured packet pays a growing tax to scan past unrelated records.

THE ZEROSTACK APPROACH

ZeroStack separates cold persistence from hot packet execution:

1. **No Flash Reads on the Packet Path:** Per-packet encryption and decryption run in constant time, regardless of flash log depth.
2. **Bounded NVM Resolution:** General non-volatile items resolve in bounded time, independent of how many records the journal holds.
3. **Zero-Wear Unchanged Writes:** Writing state that has not changed leaves flash untouched, preserving endurance and preventing log growth.

HARDWARE RESULTS: NVM OPERATIONS (CYCLES @ 48 MHZ)

Operation / Workload	Legacy Reference Stack	ZeroStack	Analysis
Stack Key Read (*D* **= 0, fresh log)**	`2,347`	**`302`**	**7.8x faster** even when the record sits at the very top of flash.

Operation / Workload	Legacy Reference Stack	ZeroStack	Analysis
Stack Key Read (*D = 150* newer records)	`64,817` (`+416` /rec)	**`303`**	**213.9x faster.** ZeroStack is strictly *O(1)* (`6.3 µs`); legacy cost scales linearly with flash traffic.
Write Unchanged Data (`2 B / 20 B / 100 B`)	`19.1k / 35.3k / 100.7k`	**`2.0k / 2.3k / 3.8k`**	**Up to 26.7x faster.** ZeroStack writes **0 bytes** to flash when data is unchanged (48/48 writes; legacy appended on 48/48).
Write Changed Data (`2 B / 20 B / 100 B`)	`19.1k / 35.4k / 102.2k`	`20.1k / 37.0k / 104.8k`	Physical flash programming time dominates when bytes genuinely mutate (`+3–5%` for integrity verification).
General Item Read* (Across Table Density)*	`581` – `14,303` *(flash walk; cost follows record position)*	**`1,696` – `1,724`**	Flat across *D = 0 … 256* regardless of log position; the legacy stack walks the flash log instead.
First Flash Page Compaction Stall	**Write #69** (`163.9 ms`)	**Write #1,013** (`13.7 ms`)	**14.7x more mutations** before compaction; none on idempotent writes. Stall times are not like-for-like: less live data was held in NV on the ZeroStack run.

3.2 Trust Center Link Key (TCLK) Lookup Scaling

THE LEGACY BOTTLENECK: CONTENT SEARCH OVER THE WHOLE FLASH LOG

In Zigbee 3.0 and R23 networks, the Coordinator acts as the Trust Center, maintaining a unique link key entry for up to 300 devices (`Slot 0` through `Slot 299`). Whenever an APS-encrypted command (such as `Transport-Key` , `Verify-Key` , `Update-Device` , or APS-encrypted application traffic) is processed, the stack must resolve the peer's 64-bit IEEE Extended Address (`extAddr`) to its corresponding TCLK slot.

The legacy stack finds the entry with a content match over the non-volatile log: it walks record headers from the newest record backwards until a link-key record carries the requested address. The cost therefore follows the **age of the device's record in the log** (how many records were written after it), not its slot number—and a device with no key yet costs a walk of the entire log.

THE ZEROSTACK APPROACH

On ZeroStack, lookup latency is independent of flash log depth and of record age, and stays within a narrow microsecond window across the entire 300-device table.

HARDWARE RESULTS: TCLK RESOLUTION (300-ENTRY TABLE)

Lookup Scenario	Legacy Reference Stack (Cycles / Time)	ZeroStack (Cycles / Time)	Speedup
Hit, newest record (Slot 0)	`2,961` – `2,963` (`61.7 µs`)	**`417` (`8.7 µs`)**	**7.1x faster**
Hit, newest record (Slot 299)	`2,962` – `2,963` (`61.7 µs`)	**`324` (`6.75 µs`)**	**9.1x faster**
Hit, oldest record measured (age 383)	`620,396` (`12.92 ms`)	**`324` (`6.75 µs`)**	**1,915x faster** (same record)
Miss, full 300-entry table (first contact)	**`1,486,228` (`30.96 ms`)**	**`326` (`6.8 µs`)**	**4,559x faster**

ZeroStack TCLK timings vary more between builds than other rows; the full-table miss ranged from `188` to `326` cycles, and the table reports the conservative value.

> **Why Commissioning Stalls on Large Legacy Networks**
>
> Each frame of a BDB 3.1 key exchange (`Request-Key` → `Transport-Key` → `Verify-Key` → `Confirm-Key`) resolves the device's key. At up to **12.9 milliseconds** per lookup on an old record, and **31 milliseconds** for a device seen for the first time, a legacy coordinator spends tens of milliseconds of CPU per joining device just finding key records. On ZeroStack every lookup measured stays at or below **9 microseconds**, independent of how much was written to flash before it.

3.3 Per-Frame Network (NWK) and Application (APS) Security

THE REAL-WORLD PACKET PATH

Every Zigbee packet on a secured mesh undergoes AES-128-CCM* authentication and encryption at the Network (NWK) layer, and sensitive control or application payloads add a second layer of AES-128-CCM* protection at the Application Support (APS) layer.

On the CC1352P7, the dedicated hardware AES crypto accelerator executes the raw 16-byte block cipher in microseconds. Yet on legacy stacks, the end-to-end security functions (`NWK Apply/Remove` and `APS TX/RX Security`) are dominated by flash log walks to fetch keys and linear scans across address management tables rather than by the cipher itself.

Furthermore, our hardware sweep exposed a severe **cliff** in the legacy stack between *D = 100* and *D = 150*: per-packet NWK encryption grows by about 900 cycles per newer record up to *D = 100* (`178 µs` → `2.05 ms`), then jumps to **`11.11 ms`** at *D = 150*—434k cycles for 50 records, far above that slope. The mechanism behind the jump has not been traced yet.

HARDWARE RESULTS: END-TO-END PER-FRAME SECURITY (MEDIAN CYCLES @ 48 MHZ)

Security Workload	Legacy Reference Stack	ZeroStack	Operational Impact
NWK Encrypt (*D = 0*)	`8,558` (`178.3 µs`)	**`3,980` (`82.9 µs`)**	**2.15x faster** at zero flash depth.
NWK Encrypt (*D = 150*)	**`533,156` (`11.11 ms`)**	**`3,968` (`82.7 µs`)**	**134.4x faster.** ZeroStack is strictly flat (`0` flash reads/packet).
NWK Decrypt (*D = 0*)	`8,091` (`168.6 µs`)	**`3,840` (`80.0 µs`)**	**2.11x faster** at zero flash depth.
NWK Decrypt (*D = 150*)	**`532,182` (`11.09 ms`)**	**`3,824` (`79.7 µs`)**	**139.2x faster.** Legacy coordinator caps at `~45 routed pkt/s`.
APS TX Encrypt (Slot *0 → 299*, *D=0..100*)	`52,909` → **`409,301` (`8.53 ms`)**	**`5,410` → `5,424` (`0.113 ms`)**	**Up to 75.5x faster** and invariant to flash log order.
Device Authentication Check (Slot *0 → 299*)	`1,656` → `42,243` (`880 µs`)	**`181` → `179` (`3.7 µs`)**	**236x faster** at 300 nodes; flat by slot (legacy `~136` cycles/slot).

> **Throughput Ceiling**
>
> Forwarding a single routed packet requires one NWK Decrypt on receive and one NWK Encrypt on transmit. At *D = 150*, the legacy stack spends **22.19 milliseconds of CPU time per routed hop** (`532k + 533k` cycles), limiting theoretical coordinator forwarding capacity to **45 packets per second** before the CPU hits 100% utilization on NWK security alone. ZeroStack completes the same NWK RX + TX security cycle in **162 microseconds** (`7,792` cycles)—supporting **over 6,100 packets per second** of NWK security throughput.

3.4 Dynamic Memory Allocation (`malloc` / `free`) Under Fragmentation

THE LEGACY BOTTLENECK: UNINDEXED FIRST-FIT HEAP WALKS

In bursty wireless networks, packets of varying sizes (12-byte ACKs, 40-byte sensor reports, 100-byte source-routed ZCL frames, and timer records) are allocated and freed asynchronously. A traditional single-list first-fit heap allocator scans linearly from the base of the heap on every allocation, stepping over allocated blocks and small unusable holes until a sufficiently large free block is found. Under sustained traffic with long-lived indirect buffers or timers pinning intermediate addresses, *k* small holes accumulate at the front of the heap.

THE ZEROSTACK APPROACH

ZeroStack's allocator completes requests in constant time, regardless of how many non-contiguous holes exist in the heap.

HARDWARE RESULTS: HEAP ALLOCATION LATENCY ACROSS FRAGMENTATION (*K* HOLES)

Heap State	Legacy Reference Stack (`malloc` / `free`)	ZeroStack (`malloc` / `free`)	Speedup
Unfragmented (*k = 0* holes)	`310` / **`84`** cycles	**`225`** / `200` cycles	**1.38x faster** baseline allocation.
Light Fragmentation (*k = 16* holes)	`869` / **`82`** cycles	**`218`** / `175` cycles	**3.99x faster.**
Moderate Fragmentation (*k = 32* holes)	`1,430` / **`82`** cycles	**`220`** / `175` cycles	**6.50x faster.**
Heavy Fragmentation (*k = 64* holes)	**`2,550`** / **`82`** cycles	**`218`** / `175` cycles	**11.7x faster** `malloc` (`~35` vs `0` cycles/hole).

`free` is slightly cheaper on the legacy stack (`82` – `84` vs `175` – `200` cycles) because the legacy allocator defers its bookkeeping to the next `malloc` walk, which is where its *O(k)* cost appears.

3.5 RTOS Timers & Event Queue Determinism

ELIMINATING HEAP-BACKED TIMERS AND LINEAR QUEUE SCANS

Every active Zigbee transaction (APS ACK retries, broadcast passive-ACK windows, route discovery expirations, and end-device poll timeouts) relies on software timers and inter-task message queues.

1. **Heap-Independent Timers:** In the reference stack, starting a software timer allocates its timer entry and its kernel clock object from the general heap (64 bytes per timer, measured). Consequently, starting a timer on a fragmented heap inherits the full *O(k)* heap scan penalty—twice. ZeroStack timer latency is unaffected by heap fragmentation.
2. *O(1)* **Inter-Task Event Queues:** When radio, MAC, NWK, and host serial tasks exchange messages, the reference stack walks the target task's linked list on every `msgSend` (*O(Q)*). ZeroStack enqueue latency stays flat regardless of queue depth.

HARDWARE RESULTS: TIMER & QUEUE SCALING (CYCLES @ 48 MHZ)

Scheduler / Timer Operation	Legacy Reference Stack	ZeroStack	Key Takeaway
Timer Start (*T = 0 → 39*, clean heap)	`2,652` → `5,844`	`912` → `1,447`	**4.04x faster** at 39 active timers (legacy performs `2` heap allocations per start).
Timer Start (*T = 0 → 39*, 64 heap holes)	`6,894` → **`10,034`**	`930` → `1,448`	**6.93x faster.** ZeroStack is unaffected by heap state.
Timer Stop (*T = 0 → 39* active timers)	`515` → `1,370`	`450` → `1,050`	**1.30x faster** at high timer load; clean deterministic teardown.
Timer Restart (*T = 0 → 39* active timers)	**`733`** → **`1,314`**	`779` → `1,396`	Legacy **1.06x faster**; same `~15–16` cycles/active timer slope.
Task `msgSend` (*Q = 0 → 64* queued msgs)	`497` → **`935`** (`+7` /msg)	`364` → `322` **(Flat)**	**2.90x faster** at *Q=64*; strictly *O(1)* enqueue.
Task `msgReceive` (*Q = 0 → 64* queued msgs)	`383` → `628`	`168` → `333`	**2.28x faster** at *Q=0*, **1.89x** at *Q=64* (dequeue & task wake path).

3.6 Host-to-Coordinator Serial Pipeline (Host Serial Path)

For gateways communicating with the coordinator over UART, host request-response turnaround governs how quickly the host application (such as Zigbee2MQTT or ZHA) can pipeline commands. Measured at 115,200 bps, from the last received byte's interrupt to the start of the reply transmission:

- **Synchronous Round-Trip (`PING`):** `9,790` cycles on the legacy stack vs **`4,618`** on ZeroStack (**53% fewer cycles**); the reply path after dispatch drops from `4,921` to **`816`** cycles (**83% fewer**). At 240-byte frames: `24,404` vs **`12,784`** (**48% fewer**).

4. Architectural Conclusion & Road Ahead

The data from Part 1 demonstrates a fundamental engineering truth: **the 100–200 device wall in modern Zigbee networks is not an RF spectrum limit or a Cortex-M4 silicon limit—it is an algorithmic complexity limit.** When a coordinator spends over 11 milliseconds of CPU securing a single packet and up to 31 milliseconds searching for a device's link key, the CPU starves the radio long before the 2.4 GHz channel saturates.

By enforcing *O(1)* runtime complexity across non-volatile storage, security key resolution, memory allocation, and task scheduling, ZeroStack turns flat microsecond latencies into a hard architectural guarantee—whether the network has 5 devices on day one or 300 devices after two years of continuous operation.

Coming Next in This Series

- **Part 2 — Adversarial Fault Injection, Fuzzing & Memory-Safe Protocol Engines:** How we verified strict memory-safe protocol and state-machine invariants across hundreds of millions of automated fuzz executions, compile-time type verification gates, and malformed over-the-air attack frames.
- **Part 3 — Multi-Node Mesh & End-to-End Hardware-in-the-Loop (HIL) Soak Testing:** Real-world throughput, packet delivery ratio (PDR), many-to-one source routing stability, and 300-node commissioning storms under continuous RF load.