Toward Practical 3D NoCs: Topology Design, Routing Bias, and Cost–Performance Trade-offs

A gem5/Garnet 3.0 Study of 3D Mesh and Sparse-Vertical Variants

Your Name Your Collaborator

Institution / Course / Term

September 4, 2025

Abstract

This paper studies *practical* 3D Network-on-Chip (NoC) design under manufacturing and area constraints. Using `gem5`/Garnet 3.0 (Garnet 3.0), we implement dense 3D Mesh and several sparse-Z variants (pillars, hub, torus augmented) and evaluate them against a same-router-count 2D Mesh under uniform traffic. We use the throughput *knee* as our primary metric.

Key findings. (i) With equal link latencies, a 4×4×4 3D Mesh sustains a ∼35% higher threshold throughput than an 8×8 2D Mesh. (ii) Enabling an *escape virtual channel* (VC_0 for an acyclic route set) amplifies the 3D advantage to ≥50–60%. (iii) In a *process-aware* setting that penalizes dense 3D with slow Z links (Z-latency=4) but allows sparse-Z designs to use faster TSVs (Z=1), Torus3D offers the largest headroom; Mesh3D_XYZ approaches Mesh_XY because *Z-latency dominates*, whereas 2D is constrained by *area/diameter*. (iv) A TSV-latency sweep shows a consistent *torus multiplier* of ≈2.2–2.4× in the knee for Sparse3D vs. Sparse3D_Pillars_torus (Z$\in \{1, 2, 4\}$: ∼0.10/0.16/0.22 vs. ∼0.22/0.38/0.50).

Guidance. When TSV budget is limited, prefer sparse-Z to recover process margin and invest it in reducing effective Z-latency; if floorplan and timing allow wrap-around, prioritize torus before micro-topology tweaks; and use an escape VC to safely harvest 3D path diversity.

1 Introduction

2D NoCs are mature and easy to implement, but their average distance and bisection bandwidth ultimately limit throughput. 3D organizations add the Z dimension to shorten paths and raise bandwidth density, yet dense per-router TSVs stress yield, thermals, and timing. In practice, designers trade Z *density* for TSV *quality*, and decide whether wrap-around (torus) is feasible in area and timing.

This paper takes a *process-aware* view of 3D NoCs. We model TSV quality via a Z-link latency parameter and explore (i) dense 3D Mesh, (ii) sparse-Z pillars and hub variants that can plausibly use faster TSVs, and (iii) torus augmentations. We also evaluate the role of an *escape VC*, which guarantees deadlock-freedom while letting other VCs exploit 3D path diversity.

Questions. (1) How large is the 3D advantage over a same-router-count 2D Mesh under equal latencies, and how does an escape VC change it? (2) When TSV process penalizes dense Z but benefits sparse-Z designs, which organization retains more of the 3D benefit? (3) How strong is the torus effect across TSV qualities?

- **Implementations.** Configurable topologies in `configs/topologies`: dense 3D Mesh, Cluster3D_Hub, Sparse3D_Pillars, SW3D_Express, Hier3D_Chiplet, and torus variants, with weight-based TABLE routing and explicit ports.
- **Reproducible method.** Injection sweeps on `gem5`/Garnet 3.0 with a knee detector; fairness controls (same VC count/buffers, link/router latencies unless stated); and a *process-aware* mixed-latency setup.
- **Findings.** Equal-latency baseline: 3D $>$ 2D by $\sim$35%; adding an escape VC raises the 3D gap to $\geq$50–60%. When dense Z is penalized (Z=4) but sparse-Z uses faster TSVs (Z=1), torus provides the largest headroom and Mesh3D_XYZ $\approx$ Mesh_XY. Z-latency sweeps show a stable torus multiplier of $\approx$2.2–2.4$\times$ in the knee.
- **Guidance.** If area allows wrap-around, enable torus first; else, use sparse-Z to gain TSV process margin and reduce effective Z-latency. In either case, keep an escape VC to safely exploit 3D path diversity.

2 Background and Related Work

Routing models. Dimension-order routing (XY/XYZ) is deadlock-free by construction. TABLE routing assigns per-port weights to bias directions. Reserving one *escape VC* (VC_0) for an acyclic route set guarantees progress while other VCs exploit adaptivity.

TSV economics $\rightarrow$ Z-latency. Dense per-router TSVs raise yield and thermal risk; sparse-Z lowers TSV count and often enables better process/stacking. We proxy TSV quality with a Z-link latency parameter.

3D organizations. Prior work spans dense meshes, pillar/hub sharing, express/long links, chiplet hierarchies, and torus wrap-arounds. We evaluate representative instances of these families with consistent router/link settings and a throughput-knee metric tailored to process-aware comparisons.

3 Topologies Implemented

This work adds Python topologies under `configs/topologies/`. All use `IntLink`/`ExtLink` wiring with explicit port names (e.g., `East`, `UpGW`, `EastExp`) and *weights* to guide TABLE routing.

3.1 3D Mesh (XYZ DOR baseline)

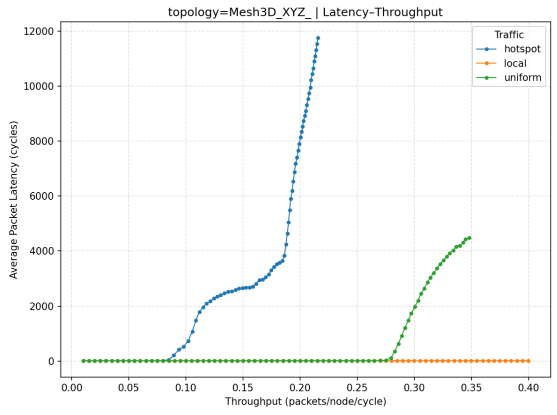
We use a standard 3D Mesh (e.g., 4×4×4) with fully populated Z links and DOR (XYZ) or TABLE routing with $W_X < W_Y < W_Z$.

- **Goal**: establish the upper-bound efficiency for a given router count.
- **parameters**: link/router latency=1; vc = 4 (default); sim-cycle = 20000; link-width=128; GlobalFrequency=2GHz.

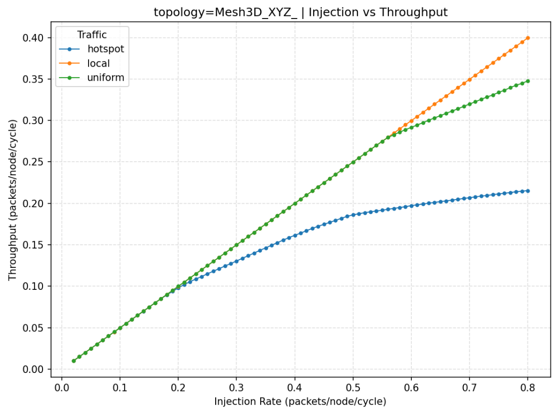
3.2 Cluster3D_Hub

A 2×2 submesh per layer shares a central hub; hubs connect vertically. The hub is transit-only (no endpoint injection). We bias routes via weights to funnel local traffic through its hub for Z.

- **Design knobs**: cluster size (e.g., 2×2×1, 2×2×2), vertical speedup, hub router latency reduction.
- **Rationale**: cut TSV count while preserving short detours to vertical resources.



(a) Throughput of the 3D Mesh topology under varying injection rates across different traffic patterns. Uniform traffic achieves the highest sustained throughput, while hotspot traffic saturates earlier due to localized congestion. Local traffic benefits from short paths and maintains high efficiency.



(b) Average packet latency versus throughput for the 3D Mesh network. At low loads, latency remains low; however, as the network approaches saturation, hotspot traffic experiences a sharp rise in latency, while uniform traffic maintains stability up to higher throughput levels.

Figure 1: Performance evaluation of the 3D Mesh topology under three canonical traffic patterns: (a) throughput vs. injection rate, and (b) latency-throughput trade-off. Results demonstrate that although 3D Mesh offers strong baseline performance, its behavior varies significantly with traffic distribution, especially under congestion.

3.3 Sparse3D_Pillars

Place vertical pillars every (P_x, P_y) tiles. Only routers aligned to a pillar get Up/Down links. Traffic uses XY detours to the nearest pillar, then moves vertically.

- **Design knobs**: P_x, P_y spacing, aligned vs. staggered patterns (future work).
- **Routing**: TABLE with $W_X < W_Y < W_Z$ and no direct Up/Down from non-pillar routers.

3.4 SW3D_Express

Add a budget of long-range *express* links (in-plane or cross-layer) on top of 3D Mesh or sparse-Z base. We use special port names (`EastExp`, `UpExp`, …) and assign them the lowest weight (W_{EXP}).

- **Design knobs**: grid-based placement period K (rule-based); total budget B; (optional) randomized placement.
- **Routing**: TABLE prefers express edges first, then normal links.

Table 1: Topology summary and intended routing bias (lower weight is preferred). Edit with your final parameters.

Topology	**Z budget**	**Extra links**	**Routing mode**	**Weights (X/Y/Z/extra)**
3D Mesh	dense	none	TABLE or XYZ DOR	1/2/3/–
Cluster3D_Hub	sparse hubs	hub vertical	TABLE	1/2/3/– (hub Z sped-up)
Sparse3D_Pillars	pillars (P_x, P_y)	none	TABLE	1/2/3/–
SW3D_Express	dense/sparse	express	TABLE	1/2/3/0 (exp)
Hier3D_Chiplet	per-gateway	backbone	TABLE	1/2/3/– (+GW tier)

3.5 Hier3D_Chiplet

Partition the die into chiplets (e.g., 4×4×1 each). Within a chiplet: local mesh. Gateways (1–2 per chiplet) connect to a backbone (2D in-layer and/or vertical). Gateways use `*GW` ports and higher weights at the backbone to avoid overuse.

- **Design knobs**: chiplet dimensions, #gateways, backbone topology (ring/mesh), vertical gateway connectivity.
- **Routing**: TABLE with tiered weights: $W_{\text{intra}} < W_{\text{backbone}} < W_Z$.

4 Routing and Deadlock Considerations

We primarily use TABLE-based routing in Garnet 3.0 with weights to induce preferences:

- **Strict ordering**: Distinct weights per dimension/tier maintain a partial order and avoid cycles typical in oblivious shortest-path selection.
- **Express edges**: assign the smallest weight (e.g., $W=0$) to let the router choose them deterministically before normal links.
- **Gateway tiers**: use intermediate weights for backbone/GW hops to avoid accidental oscillations.

We validated reachability via `config.ini` inspections and by eliminating “No Route exists from this Router” fatals during warmup runs.

5 Methodology

5.1 Simulator and Network

We use `gem5` with Garnet 3.0 (Garnet 3.0). Unless noted, system and Ruby clocks are 2GHz. Link width 128 bits, link latency 1, router latency 1.

5.2 Metrics

We collect average packet/flit latency, average hops, total injected/received packets (to derive accepted rate), and average link utilization. Saturation is indicated by the knee where latency diverges and accepted rate flattens.

6 Experimental Results

In following experiments, we use transpose traffic patterns to simulate hotspot traffic, and use neighbor traffic patterns to simulate local traffic, and uniform-random is identical to uniform traffic.

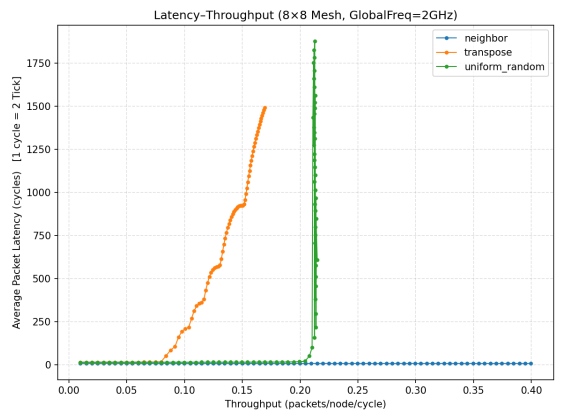
6.1 3D Mesh vs. 2D Mesh (same router count)

We compare a 4×4×4 3D Mesh with an 8×8 2D Mesh, both using 64 routers and identical link/router latencies. The 3D topology is expected to achieve lower average hop counts and higher throughput due to shorter paths and improved bisection bandwidth.

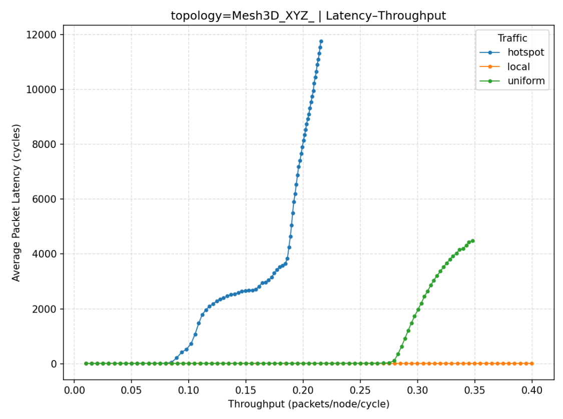
As shown in Figure 2, the 3D Mesh supports significantly higher throughput before saturation under uniform and transpose traffic (up to ∼0.35 packets/node/cycle), compared to the 2D Mesh's saturation at ∼0.21. Moreover, the 3D design reduces average hop count—especially under uniform traffic—from ∼5.3 in 2D to ∼3.75 in 3D—by leveraging vertical shortcuts.

6.2 Effect of Escape VC on 3D vs. 2D

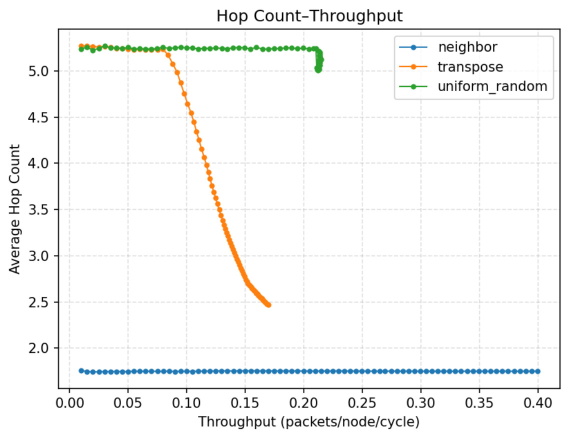
We repeat the uniform-random sweep while enabling an *escape virtual channel* (VC_0) reserved for a strictly ordered, deadlock-free route set (e.g., DOR). The remaining VCs use the standard (normal/adaptive) policy. **All other parameters are kept identical** across 2D and 3D



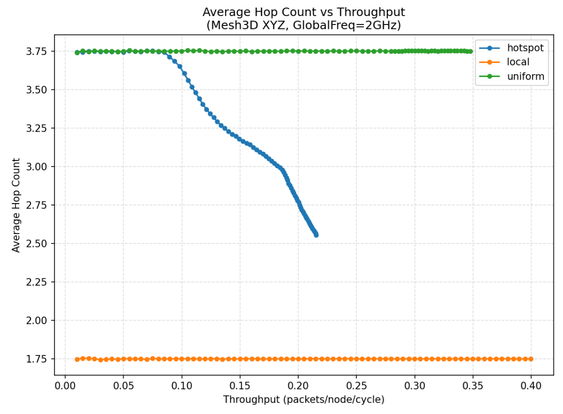
(a) Latency-throughput curve for the 8×8 2D Mesh under uniform, transpose, and neighbor traffic patterns. The network saturates at approximately 0.21 packets/node/cycle, with latency diverging sharply under uniform and transpose traffic due to global congestion. Neighbor traffic remains stable due to short paths.



(b) Latency-throughput curve for the 4×4×4 3D Mesh under hotspot, local, and uniform traffic. Despite higher average hop counts in some cases, the 3D topology achieves significantly higher throughput before saturation (up to ∼0.35), especially under uniform traffic, thanks to reduced path lengths and better bisection bandwidth.



(c) Average hop count versus throughput for the 8×8 2D Mesh. Neighbor traffic maintains a constant low hop count (∼1.7) due to short distances. Transpose traffic shows a sharp drop in hop count as injection rate increases, indicating more efficient routing through balanced load distribution. Uniform traffic remains near the theoretical maximum of 5.3 hops across all loads, reflecting long, random paths.

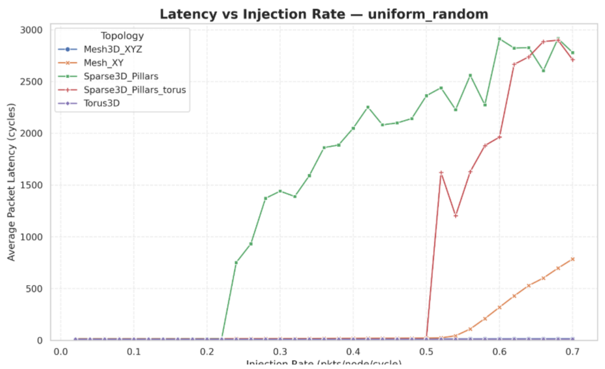


(d) Average hop count versus throughput for the 4×4×4 3D Mesh. Local traffic maintains a minimal hop count (∼1.75) throughout, consistent with its short-range nature. Hotspot traffic exhibits a gradual decrease in hop count with increasing load, suggesting adaptive routing or load-balancing effects. Uniform traffic stabilizes at ∼3.75 hops, which is significantly lower than the 2D baseline, demonstrating the benefit of Z-dimension shortcuts in reducing average path length.

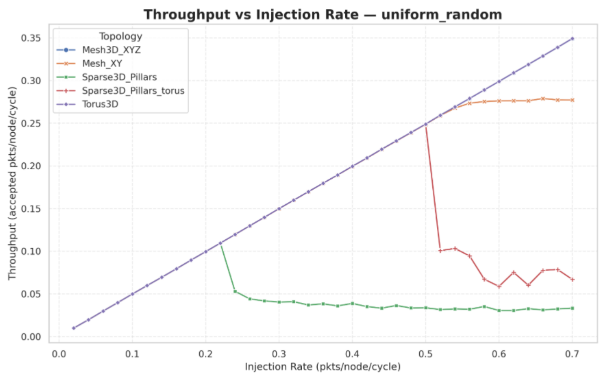
Figure 2: Comparison of performance metrics between 2D Mesh (8×8) and 3D Mesh (4×4×4) with identical router count (64). Top row: latency-throughput curves show that the 3D Mesh supports higher throughput and earlier saturation under global workloads. Bottom row: hop count trends reveal that the 3D Mesh reduces average path length, especially under uniform traffic, by leveraging vertical links for shorter routes.

(`vcs-per-vnet`=4 with one reserved for escape, identical buffers, link/router latencies, clocks, and traffic).

Empirically, the 2D Mesh exhibits a clear knee (threshold throughput) around ∼0.21 pkts/node/cycle, whereas the 3D Mesh shows no visible knee up to our maximum offered load of 0.7, achieving accepted throughput ≥0.35 in the sweep. Using the *threshold throughput* metric, the relative advantage of 3D increases from about 35% (without escape VC) to **≥50–60%** with escape VC (e.g., 0.35/0.21 ≈ 1.67×).



(a) Latency vs. injection under uniform-random with escape VC.



(b) Accepted throughput vs. injection under uniform-random with escape VC.

Figure 3: Escape VC (VC_0) amplifies the 3D advantage: 2D shows a knee near ∼0.21, while 3D shows no knee up to ≥0.35 within our sweep range.

Intuition. The escape VC guarantees progress, allowing the other VCs to exploit additional turn freedom and transient path diversity. In 3D, the Z dimension supplies more minimal and near-minimal alternatives, so adaptive VCs more effectively relieve head-of-line blocking and allocator contention than in 2D.

Fairness and limits. Both 2D and 3D reserve exactly one escape VC; buffer sizes and VC counts per input port are identical. Our current sweep upper bound is 0.7 offered load; future work will extend the range to locate the 3D knee more precisely.

6.2.1 TSV-Latency Sensitivity: Sparse3D vs. Sparse3D_Pillars_torus

To quantify manufacturing choices, we sweep TSV settings with Z-latency ∈ $\{1, 2, 4\}$ for two families: *Sparse3D_Pillars* and *Sparse3D_Pillars_torus*. The measured *throughput knees* under uniform traffic are:

Topology	Z=1	Z=2	Z=4
Sparse3D_Pillars	∼0.10	∼0.16	∼0.22
Sparse3D_Pillars_torus	∼0.22	∼0.38	∼0.50

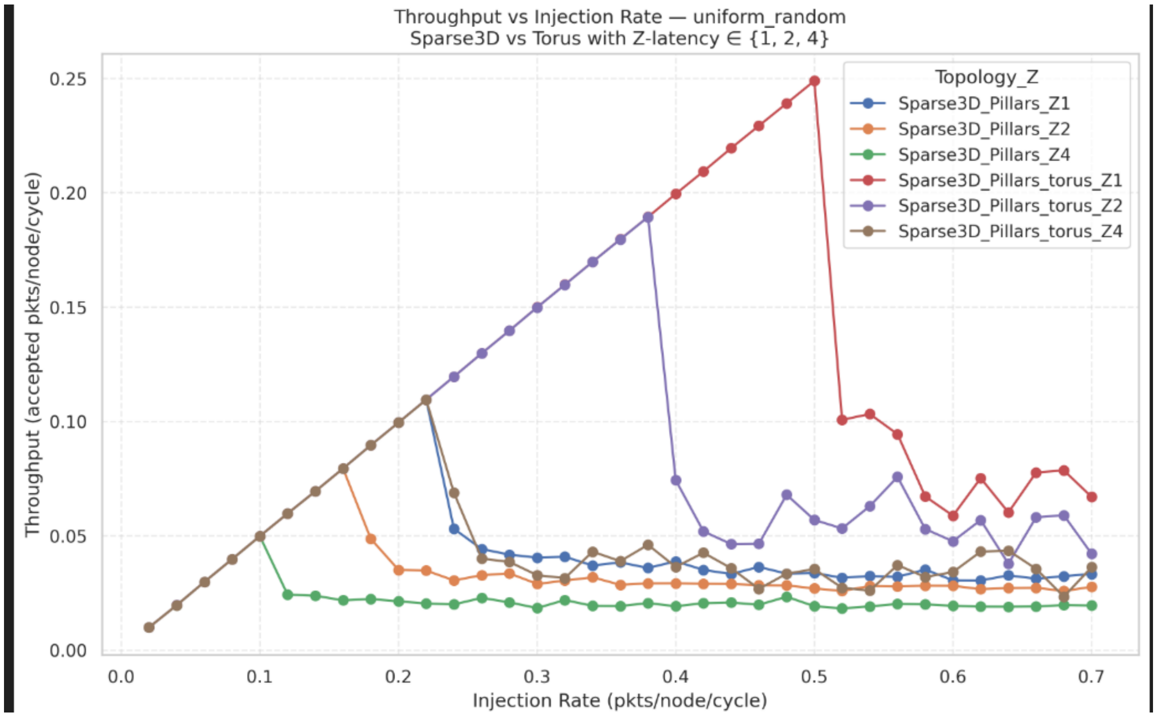


Figure 4: Z-latency sensitivity for Sparse3D vs. Sparse3D_Pillars_torus under uniform traffic. The torus-augmented design consistently pushes the knee higher (about 2.2×–2.4× across the three Z settings in our runs).

Design takeaways.

- **Torus multiplier.** For the same Z setting, adding torus roughly doubles the knee (e.g., 0.22/0.10≈2.2, 0.38/0.16≈2.4, 0.50/0.22≈2.3).
- **Process-first budget use.** If TSV process can be improved for a *sparse* Z budget, prioritizing lower effective Z-latency yields a larger payoff than micro-level tweaks on a non-torus grid.
- **Where to spend area.** If floorplan permits wrap-around, torus offers the highest headroom per TSV; otherwise, investing TSV quality on a sparse grid still moves the knee materially.

6.3 Process-Aware Case Study: Mixed Z-Link Latency

Setup. To emulate realistic manufacturing trade-offs, we keep XY link latency at 1 for all topologies. We penalize the dense 3D mesh by setting its Z links to latency 4 (to reflect TSV difficulty when every router has a TSV), while *sparser-Z* topologies use Z-latency = 1 (better TSV process made possible by lower TSV density).

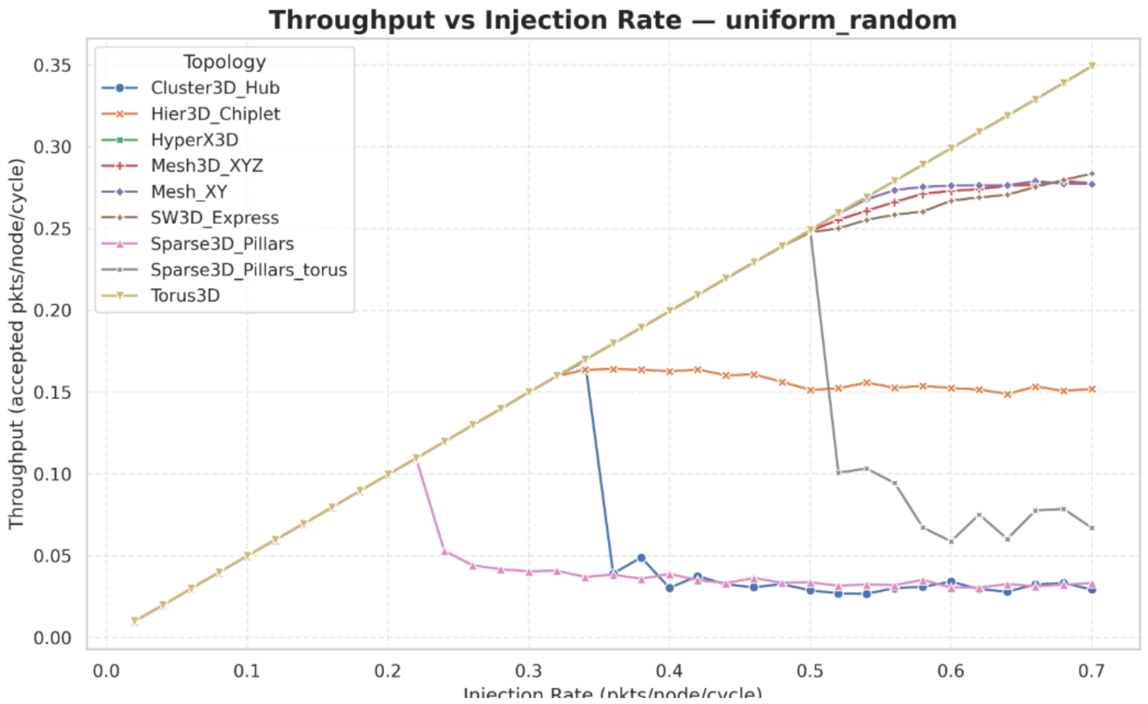


Figure 5: Process-aware comparison under uniform traffic: dense 3D mesh uses Z-latency 4; all other sparse-Z designs use Z-latency 1. Torus3D shows the largest headroom; Mesh_XY and Mesh3D_XYZ become close because the latter is bottlenecked by slow Z.

Observations.

- **Torus impact is substantial.** Torus3D continues scaling to the highest accepted rate (cf. Fig. 5); whether such wrapping can be realized within the floorplan and timing is a first-order implementation question.

- **Mesh3D_XYZ ≈ Mesh_XY under slow Z.** This contrasts with our equal-latency baseline where 3D > 2D. Here the 3D advantage is dominated by TSV process (Z-latency), whereas 2D is constrained by *area/diameter*. *Process vs. area* is therefore a central chip-production trade-off.

- **Generalization to other 3D variants.** Topology evolution alone yields limited gains if the Z process is poor; with improved TSV (enabled by sparsity), benefits reappear. Hence, *manufacturing and floorplanning constraints often dominate over micro-topology tweaks.*

7 Discussion

7.1 Manufacturing-driven guidance.

Our process-aware study (Secs. 6.3, 6.2.1) shows that (i) Z-latency (TSV quality) can erase or amplify the theoretical 3D gain; (ii) torus raises the throughput knee by about 2.2–2.4× across Z settings in our runs; (iii) when TSV density is reduced (sparse-Z), the saved process margin can be reinvested to achieve lower Z-latency and reclaim most of the 3D advantage. In short: *process & area choices dominate topology micro-variants*—optimize TSV quality or enable torus first, then tune the grid.

7.2 Routing Simplicity vs. Performance

TABLE routing with weights gives strong baselines without custom algorithms; we note when custom adaptive schemes might help (left as future work).

7.3 Limitations

8 Conclusion

A Topology Implementation Details

We place our Python files in `configs/topologies/`:

- `Mesh3D_XYZ_.py`: 3D Mesh baseline (XYZ DOR or TABLE).
- `Cluster3D_Hub.py`: cluster hub + vertical hub links (with configurable hub speedup).
- `Sparse3D_Pillars.py`: pillar spacing (P_x, P_y); optional staggered layout (future).
- `SW3D_Express.py`: rule-based express placement (period K) with W_{EXP}=0.
- `Hier3D_Chiplet.py`: chiplet size, #gateways, backbone wiring.

We followed gem5's `Mesh_XY` style: create routers, wire `ExtLink`s to NIs, then build `IntLink`s with port names and weights.

B Command Lines and Sweeps

Single Run Example

```
./build/NULL/gem5.opt configs/example/garnet_synth_traffic.py \
  --network=garnet --topology=Sparse3D_Pillars \
  --num-cpus=64 --num-dirs=64 --mesh-rows=4 \
  --routing-algorithm=0 \
  --synthetic=uniform_random --injectionrate=0.05 \
  --sim-cycles=2000000 \
  --sys-clock=1GHz --ruby-clock=1GHz \
  --link-width-bits=128 --link-latency=2 --router-latency=2 \
  --mem-type=SimpleMemory --mem-channels=64 --mem-size=8192MB
```

Sweep Script (bash)

```
for r in $(seq 0.01 0.01 0.25); do
  ./build/NULL/gem5.opt \
    -d runs/pillars_uniform_r${r} \
    configs/example/garnet_synth_traffic.py \
    --network=garnet --topology=Sparse3D_Pillars \
    --num-cpus=64 --num-dirs=64 --mesh-rows=4 \
    --routing-algorithm=0 \
    --synthetic=uniform_random --injectionrate=${r} \
    --sim-cycles=2000000 \
    --sys-clock=1GHz --ruby-clock=1GHz \
    --link-width-bits=128 --link-latency=2 --router-latency=2 \
    --mem-type=SimpleMemory --mem-channels=64 --mem-size=8192MB
done
```

C Data Collection and Plotting

Collect CSV.

```
./collect.sh runs results/pillars_uniform.csv
```

Plot curves.

```
python3 plot_metrics.py \
  --out results/plots \
  --series "Pillars:results/pillars_uniform.csv" \
  --series "3D Mesh:results/mesh3d_uniform.csv"
```

D Sanity Checks

We grep `config.ini` to confirm topology name, routing mode, #IntLinks, port names (e.g., `UpGW`, `EastExp`) and weight histograms; we ensure every router has outgoing `IntLinks`.

References

References