AdEle+: An Adaptive Congestion-and-Energy-Aware Elevator Selection for Partially Connected 3D Networks-on-Chip

Ebadollah Taheri, *Student Member, IEEE*, Ryan Gary Kim, *Member, IEEE*, and Mahdi Nikdast, *Senior Member, IEEE*

***Abstract*—Vertical die stacking of 3D Networks-on-Chip (3D NoCs) is enabled using inter-layer Through-Silicon-Via (TSV) links. However, TSV technology suffers from low reliability and high fabrication costs. To mitigate these costs, Partially Connected 3D NoCs (PC-3DNoCs), which use fewer TSV links, have been introduced. Nevertheless, with fewer vertical links (a.k.a. elevators), elevator-less routers will have to send their traffic to nearby elevators for inter-layer traffic, increasing the traffic load and congestion at these elevators and potentially reducing performance. Therefore, it is important that elevator-less routers choose elevators that balance the traffic load among the available elevators. To address this problem, we present an adaptive congestion- and energy-aware elevator-selection algorithm, called AdEle+. AdEle+ employs an offline multi-objective simulated-annealing-based optimization to find good elevator subsets for routers. During high traffic loads, AdEle+ uses an adaptive and online elevator selection algorithm to select an elevator from the elevator subset to dynamically manage traffic congestion on elevators. Moreover, in low congestion circumstances, AdEle+ switches to a distance-based selection to improve energy efficiency. Compared to state-of-the-art selection algorithms under various PC-3DNoC configurations and traffic patterns, AdEle+ reduces the average latency by 9.5% on average and up to 11.2% while reducing the hardware overhead by 10.1%**

***Index Terms*—Adaptive routing, elevator selection, multi-objective optimization, partially connected 3D networks-on-chip, simulated annealing, through-silicon via.**

I. Introduction

NETWORK-ON-CHIP (NOC) has become the prevailing solution to enable scalable on-chip communication in manycore systems [1]. Moreover, with the advances in three-dimensional (3D) integration technologies, systems with stacked dies are interconnected using Through-Silicon Vias (TSVs), further improving NoC scalability, integration density, and system heterogeneity [2], [3], [4].

To create vertical links in TSV-based 3D NoCs, multiple TSVs are grouped into a bundle. However, due to the large TSV interconnect pitch and keep-out-zone requirements [5], these TSV bundles result in large area overhead. Moreover, TSVs are particularly susceptible to electromigration and capacitive crosstalk-induced issues [6], [7]. Therefore, it is very costly to have TSV-based vertical links at every router [2], [3]. To address these challenges, Partially Connected 3D NoCs (PC-3DNoCs) with TSV-based vertical links at only some routers have been proposed [3], [8], [9]. In addition to reduced fabrication costs, the PC-3DNoC paradigm can be utilized to handle missing vertical links due to TSV-based faults [10] and to improve manufacturing yields.

Since PC -3DNoCs remove some of the vertical links (a.k.a. elevators), the remaining elevators must be shared among multiple routers. A well-known routing solution in PC-3DNoCs, Elevator-First routing [8], naïvely select the nearest elevator without considering the overall network traffic, potentially creating traffic hotspots at certain elevators and increasing the network latency [3]. To mitigate the effects of these traffic hotspots, we can balance the traffic across all elevators using an adaptive routing technique that selects elevators based on elevator utilization. For example, Congestion-aware Dynamic elevator Assignment (CDA) [11] uses global traffic information to improve the elevator load distribution during runtime.

Elevator-selection algorithms in PC-3DNoCs can be broadly classified into design-time and runtime approaches. For example, in [8], [12], each router is assigned one elevator for all vertical traffic at design time and to optimize network performance (e.g., the network latency). On the other hand, online approaches like [13] monitor traffic and select elevators during runtime to help balance the elevator load. Since design-time approaches do most of the calculations offline, only simple look-up tables are required, oftentimes resulting in simpler and faster implementations compared to an online approach. However, these offline approaches [8], [12], [14], [15], [16] rely only on the traffic information available during design time and cannot adapt to new runtime traffic scenarios or changing system conditions, e.g., faults. Although online solutions can help adjust to these changes, they can impose large overhead. For example, in CDA [11], [13], gathering global traffic information from distant routers and determining the selection at runtime impose significant hardware and latency overhead.

Manuscript received 15 April 2022; revised 27 January 2023; accepted 11 February 2023. Date of publication 23 February 2023; date of current version 11 July 2023. This work was supported by the National Science Foundation (NSF) under Grant CNS-2046226. Recommended for acceptance by H. Matsutani. *(Corresponding author: Mahdi Nikdast.)*

The authors are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA (e-mail: ebad.taheri@colostate.edu; rygkim@gmail.com; mahdi.nikdast@colostate.edu).

Digital Object Identifier 10.1109/TC.2023.3248260

This paper addresses the elevator-selection problem in PC-3DNoC routing techniques by developing, a novel, congestion- and energy-aware adaptive elevator-selection scheme called AdEle+. The main contributions of AdEle+ are summarized in the following:

- We propose AdEle+, a two-stage elevator-selection approach that takes advantage of both design-time and runtime benefits: AdEle+ integrates a design-time elevator-set optimization and a runtime elevator-selection policy to balance traffic with minimal overhead.
- We utilize a Multi-Objective Simulated-Annealing-based optimization algorithm (AMOSA [17]) offline for the design-time elevator-set optimization. AMOSA chooses, for each router, an optimized subset of elevators that will be used during runtime selection to simplify the online elevator selection.
- We develop a low-cost runtime elevator selection that has two modes: one for high traffic loads using local traffic information and a simple modified round-robin technique to improve the elevator load; and another for low traffic loads using a distance-based elevator selection that considers the elevator distance to *both* source and destination.
- An algorithm is presented to find the optimized threshold for switching between the different runtime elevator selection modes.

Our simulation results, under different synthetic and real-application traffics with various PC-3DNoC configurations, show the promise of AdEle+ compared to state-of-the-art. For instance, AdEle+ improves the network latency by 10%, on average, and by up to 13.9%. AdEle+ also reduces the energy consumption in low traffic scenarios. Moreover, due to the local traffic monitoring of AdEle+, hardware overhead is reduced, compared to CDA selection, in which global traffic information is shared among routers.

The rest of the paper is organized as follows. We present the background and review some prior related work on PC-3DNoCs in Section II. Section III discusses the elevator-selection problem and its complexity in PC-3DNoCs. Moreover, it details our proposed technique (AdEle+) and its implementation. In Section IV, we explore the parameters of AdEle+ to optimize the elevator selection. In addition, Section IV presents our simulation results including latency, energy, power, and hardware. Finally, Section VII concludes the paper.

II. Background and Related Work

This section discusses PC-3DNoCs and related challenges in routing, elevator placement, and elevator selection.

A. Partially Connected 3D Networks-on-Chip

The total number of TSVs highly affects the overall cost of 3D chips [18], [19]. Since each elevator link includes tens to hundreds of TSVs (e.g., 2×128 bits in [20]), limiting elevator links to only some routers—a Partially Connected 3D NoC (PC-3DNoC)—can help significantly improve the fabrication cost [2], [8], [9], [21]. For example, a PC-3DNoC with elevators at 25% of the routers significantly improves the chip fabrication

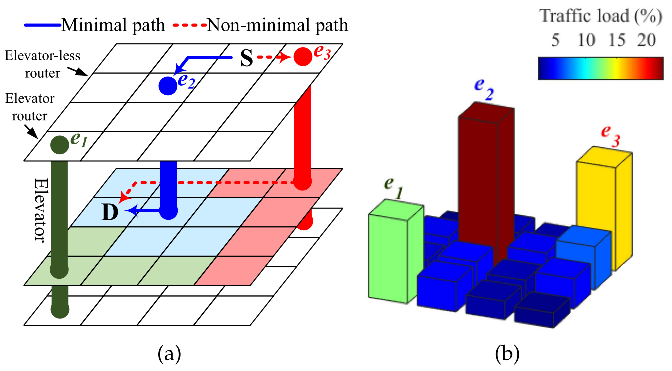


Fig. 1. (a) An example PC -3DNoC with three elevators (e_1, e_2, and e_3). The routing path from S to D based on Elevator-First algorithm [8] (dotted-red line) and the minimal path (blue-solid line) are shown. The middle-layer routers are colored based on their Elevator-First selected elevator. (b) Traffic load on each router in the middle layer: the e_2 elevator is highly congested because of the inefficient elevator selection in Elevator-First algorithm (7 out of 16 routers use this elevator router).

cost with only 18.7% overhead in the network latency, compared to a fully connected 3D NoC [22]. Moreover, PC-3DNoC schemes can be better extended to deal with TSV faults as they are designed to work in a partially connected network with missing TSV links [10]. Note that our work assumes a given PC-3DNoC topology and thus can handle manufacturing TSV faults. Considering runtime TSV faults are beyond the scope of this paper.

As shown in Fig. 1(a), PC-3DNoCs consist of two types of routers: elevator-less routers with five ports (east, south, west, north, and local), and elevator routers that include two additional ports (up and down) for a total of seven ports. As elevator-less routers are not directly connected to the other layers, their inter-layer packets must first route to an elevator router.

B. Elevator Placement and Routing Algorithms

Similar to elevator selection during routing in PC -3DNoCs (see Section 2.3), elevator placement also plays an important role in the network performance. In [22], the authors proposed an elevator placement that minimizes the hop count from all routers to nearby elevators. It is assumed that elevator-less routers utilize their closest elevator in the layer. Considering this assumption, an optimized placement is found to minimize the average router to elevator distance. However, in runtime network operation, different elevators might be selected for elevator-less routers, which is not consistent with the assumption in the placement optimization. To this end, [11] uses a Genetic Algorithm (GA) to place elevators while considering elevator selection to minimize the average distance and load variance. However, these approaches will likely end up with a non-uniform elevator placement, after manufacturing faults, which can lead to non-uniform elevator usage. Even with uniform elevator placement, traffic is typically non-uniform leading to imbalanced elevator utilization. Fortunately, an adaptive elevator selection algorithm can compensate for these effects and balance the traffic over the elevators.

Many routing algorithms have been proposed for PC-3DNoCs [2], [8], [9], [10], [15], [21], [23], [24], [25]. They are mainly different in their approach to guarantee deadlock freedom. For instance, in Elevator-First [8], the most popular deadlock-free routing for PC-3DNoCs, upward-bound packets are routed in one virtual channel while downward-bound packets are routed in another virtual channel to break the cyclic dependency and maintain deadlock freedom. However, Elevator-first is a deterministic routing algorithm and it suffers from a low adaptation in path selection. To this end, LEAD [15] offers an adaptive routing algorithm while guaranteeing deadlock freedom by defining some subnetworks. However, all of these routing algorithms employ a simple selection, where the closest elevator to the source router is used for inter-layer routing. Although simple, this selection results in an unbalanced traffic load over elevators and, therefore, unnecessary network congestion and performance loss.

C. Elevator Selection Algorithms

In PC-3DNoCs, the elevator selection process can highly impact the traffic distribution across the elevators and consequently the performance of the entire network. Unfortunately, elevator selection in PC-3DNoCs has been barely studied in related work. In conventional routing algorithms, usually the closest elevator is selected to route inter-layer packets [8], [23], [24]. However, this selection ignores the elevators' load distribution and can lead to network congestion. This can be especially harmful for PC-3DNoCs with non-uniform elevator placements, small number of elevators, or non-uniform traffic distribution. Adaptive elevator-selection techniques have been proposed [2], [9], [21], but they mainly focus on designing fault-tolerant approaches to handle elevator failures. These strategies select the closest non-faulty elevator to the source without considering the elevator's load and congestion.

To improve the traffic distribution in PC-3DNoCs, [12] proposed an offline optimized elevator-selection algorithm using the Tabu search algorithm to distribute the load over elevator links and reduce network latency. However, this approach does not consider network energy and cannot capture the dynamics of the runtime network traffic due to its offline nature. In [15], a simple online selection strategy was presented where routers select one elevator randomly. This random selection improves the elevator traffic distribution compared to the closest elevator selection. However, it increases the average hop count and energy consumption as some packets will have to travel much farther to distant elevators. In [11] and [13], an online Congestion-aware Dynamic elevator Assignment (CDA) was proposed. CDA selects the elevator that minimizes the sum of the delay and the buffer utilization of the routers between the packet's source and the elevator. However, CDA requires online global information of the routers' delay and buffer utilization, sharing of which imposes high overhead.

Considering the aforementioned efforts, an efficient elevator-selection solution is yet to be realized for PC-3DNoCs. In [26], we proposed an Adaptive congestion- and energy-aware Elevator-selection algorithm (AdEle) to optimize a subset of elevators for each router, then leverages such subsets in runtime to dynamically avoid congested elevators and improve traffic while relying only on local information. Employing a set of elevators instead of one elevator per router enables the online selection to adapt to new traffic patterns, hence improving the network performance. However, AdEle [26] does not always use the shortest-path elevator and it suffers from high energy consumption under low network traffic. In this paper, we extend our prior work in [26] to realize a low-cost Distance-Based (DB) elevator selection for the low congestion scenarios to improve the energy efficiency. In addition, an algorithm is proposed for switching between the traffic-aware and distance-based selections to enable AdEle+ to account for a dynamic traffic load. Moreover, we explore design parameters of AdEle+ and optimize them to significantly improve the performance. We also show the effectiveness of the distance-based elevator selection, under low traffic load scenarios, and the dynamic switching between the distance-based and congestion aware selections.

III. Proposed Elevator-Selection Algorithm: AdEle+

This section discusses the main challenges for elevator selection in PC-3DNoCs and details our proposed adaptive congestion- and energy-aware elevator-selection algorithm, AdEle+. As an overview, Fig. 2 shows the building blocks of the proposed algorithm: AdEle+ uses an offline multi-objective simulated-annealing-based algorithm (AMOSA) to find an optimal subset of elevators for each router and an online elevator-selection algorithm to then select the best elevator from the subset in the presence of runtime traffic.

A. Motivation: Routing in PC-3DNoCs

In PC-3DNoCs, the routing process requires three main steps because of the irregular topology: 1) selecting a vertical link (elevator) for each packet in the source router and then routing the packet to that elevator on the source layer; 2) vertically routing the packet to the destination layer; and 3) routing the packet from the elevator to the destination node on the destination layer. In PC-3DNoCs, the elevator selection (the first step) is critical as elevators quickly become the network bottleneck due to the smaller number of elevators. As we will show, AdEle+ optimizes the elevator selection in the first step to balance the traffic over the elevators, thereby reducing network traffic hot-spots.

Fig. 1(a) shows an example of a PC -3DNoC with three elevators (e_1–e_3). Router tiles are colored with the elevator's color they would use under the Elevator-First policy (i.e., the closest elevator to the source router is selected) [2], [8], [9], i.e., four routers will use the green (e_1) elevator, seven will use the blue (e_2) elevator, and five will use the red (e_3) elevator. Unfortunately, such an unbalanced elevator selection can put severe traffic pressure on certain elevators (e_2 in this example). Ideally, some of the load on e_2 should be assigned to e_1 or e_3, making e_2 less congested. Fig. 1(b) demonstrates the utilization of the middle-layer routers with Elevator-First selection policy under uniform traffic. This confirms that e_2 is highly congested due to the unbalanced elevator selection. In terms of energy efficiency in low traffic loads, the best elevator selection is on

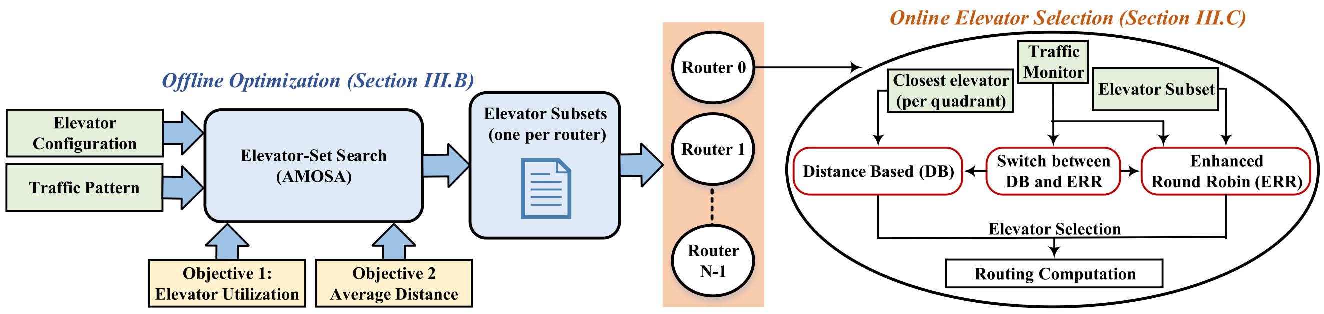

Offline Optimization (Section III.B)
Elevator Configuration
Traffic Pattern
Elevator-Set Search (AMOSA)
Elevator Subsets (one per router)
Objective 1: Elevator Utilization
Objective 2 Average Distance
Router 0
Router 1
Router N-1
Online Elevator Selection (Section III.C)
Closest elevator (per quadrant)
Traffic Monitor
Elevator Subset
Distance Based (DB)
Switch between DB and ERR
Enhanced Round Robin (ERR)
Elevator Selection
Routing Computation

Fig. 2. An overview of our proposed elevator-selection algorithm: AdEle+.

the minimal path between the source and destination. However, for the path between S and D in Fig. 1(a), policies that ignore the location of the destination during the elevator selection, e.g., Elevator-First, may end up with longer paths (red-dotted line) than the minimal path (blue-solid line).

AdEle+ considers both the elevator utilization and energy efficiency to select elevators with distributed traffic load and minimize source-destination distance. To the best of our knowledge, AdEle+ is the first congestion- and energy-aware elevator-selection algorithm in PC-3DNoCs that includes dynamic elevator selection to accommodate dynamic traffic behavior while relying only on local router information.

B. Offline Optimization in AdEle+

To find the optimal subset of elevators for each router, AdEle+ performs an offline optimization to distribute the expected traffic load across all elevators and minimize the average inter-node (source to destination) distance. To do this, we first define two optimization objectives: 1) elevator-utilization variance to improve the traffic load distribution, and 2) average inter-node distance to minimize the energy consumption. Leveraging these objective functions, we will use a multi-objective simulated-annealing-based algorithm (AMOSA [17]) to find the optimal elevator subsets.

1) Objective 1—Elevator Utilization: To balance the traffic on the elevators, AdEle+ attempts to minimize the elevator-utilization variance. As discussed above, it is important to evenly distribute the traffic over elevators to avoid highly congested elevators. To calculate the utilization variance, let us consider an N-node or N-router network with a set of elevators $\mathcal{E} = \{e_1, e_2, \dots, e_E\}$, where E is the total number of elevators. Moreover, assume that in runtime, each router i can select its elevator from a subset $A_i \subseteq \mathcal{E}$. For the time being, let us assume that each router selects each elevator from its elevator subset (A_i) uniformly (e.g., using a round-robin policy). Therefore, the utilization of elevator e (U_e) is:

$$U_e = \sum_{i=1}^{N} \frac{1}{|A_i|} \sum_{j=1}^{N} f_{ij} \cdot P_{ije}, \tag{1}$$

where f_{ij} is the communication frequency (i.e., traffic) between routers i and j, and $P_{ije} = 1$ when the routing between routers i and j uses the elevator e, otherwise $P_{ije} = 0$. Leveraging (1), the average traffic over all the elevators (μ) can be defined as:

$$\mu = \frac{1}{E} \sum_{i=1}^{E} U_i. \tag{2}$$

Using (1) and (2), elevator-utilization variance (σ^2) is:

$$\sigma^2 = \frac{1}{E} \sum_{i=1}^{E} (U_i - \mu)^2. \tag{3}$$

Minimizing the elevator-utilization variance will result in a better distribution of traffic load on the elevators and eventually lower the network latency by reducing network congestion and traffic hot-spots.

2) Objective 2—Average Inter-Layer-Node Distance: To improve network energy efficiency, AdEle+ attempts to minimize the average inter-layer-node distance when selecting the elevator subsets. The distance (D_{ij}^e) between inter-layer nodes i and j using elevator e can be defined as:

$$D_{ij}^e = \begin{cases} 0, & i \text{ and } j \text{ are on the same layer} \\ d_{se} + d_e + d_{ed}, & \text{otherwise.} \end{cases} \tag{4}$$

Here, d_{se}, d_e, and d_{ed} are the Manhattan distances between the source and elevator, the source and destination layers (through the elevator), and the elevator and destination, respectively. Based on (4), the average inter-layer-node distance (AD) in an L-layer network can be calculated as:

$$AD = \frac{1}{N \cdot (\frac{L-1}{L} \cdot N)} \sum_{i=1}^{N} \frac{1}{|A_i|} \sum_{e \in A_i} \sum_{j=1}^{N} D_{ij}^e. \tag{5}$$

3) Multi-Objective Optimization: We use a multi-objective simulated annealing-based optimization algorithm (AMOSA [17]) to find a set of optimal elevator subsets for all routers in the network ($\mathcal{A} = \{A_1, \dots, A_N\}$) while minimizing the objective functions (3) and (5). Similar to Simulated Annealing (SA) [27], AMOSA performs a broad exploration at the start of the search process and gradually chooses more greedy moves to select the best solutions to help approximate the global optima in a large solution space. Differently than SA, AMOSA outputs a *set* of solutions that lie on the Pareto front of the optimization objectives. In AdEle+, AMOSA provides solutions with different tradeoffs in terms of elevator-utilization variance and average inter-layer-node distance. From these solutions, a designer can choose the appropriate tradeoff (see

Fig. 6). Selection of solutions are discussed in detail in Section IV.

C. Adaptive Online Elevator Selection

Here, we discuss how a router i can efficiently select an elevator during runtime from its elevator subset (A_i) identified in the previous subsection. A common method is to use a simple Round Robin (RR) approach where each elevator is selected equally in a sequential order. However, solutions such as RR do not consider traffic patterns and congestion that occur during runtime. As we are interested in an even distribution of traffic load over all elevators to improve traffic congestion during runtime, we propose an Enhanced RR (ERR) algorithm for selecting elevators. Our proposed ERR approach includes a probability of skipping P_{Sik} a congested elevator k for each router i. P_{Sik} is adjusted based on the average latency imposed by the elevator k, i.e., higher latencies seen using elevator k increases the probability of skipping it in the future. Accordingly, AdEle+ can adaptively manage dynamic traffic loads and congestion.

To find P_{Sik}, AdEle+ first estimates the cost of selecting a particular elevator by considering the time between when the first flit (the header flit) and when the last flit (the tail flit) leave the source router. The latency (T_k) imposed by selecting elevator k is:

$$T_k = \frac{t_{tail} - t_{head} - l_p}{l_p}, \tag{6}$$

where t_{tail} and t_{head} denote the time when the tail flit and the header flit leave the source router, respectively. Also, l_p is the length of the packet. After each packet leaves a router i, the elevator-selection cost (C_{ik}) is updated:

$$C_{ik} \leftarrow (a \times T_k) + ((1-a) \times C_{ik}), \quad 0 \leq a \leq 1 \tag{7}$$

where a is used to adjust the impact of the new cost versus the old one. This allows us to keep some past information about the congestion at elevator k while updating it with the most recent transmission. We have experimentally found that $a = 0.2$ produces good results in AdEle+.

Leveraging (7), AdEle+ can estimate the latency cost at the source router with *only local information*, while the state-of-the-art [11], [13] requires global information with high overhead. With wormhole switching, any blocking in an elevator can be propagated along the path from the elevator to the source router. Since we expect the elevators to be the predominant source of congestion in these PC-3DNoCs, blocking at a source router can be interpreted as blocking in the elevator. Note that incorporating global-network information into AdEle+ would improve the selection policy but will impose high hardware area, energy consumption, and latency costs.

Using (7), we define router i's relative cost when selecting elevator k among other elevators in its elevator set A_i:

$$C_{ik}^{rel} = \frac{C_{ik}}{\sum_{p=1}^{|A_i|} C_{ip}}. \tag{8}$$

Where C_{ip}, is the cost of elevator p in router i (See (7)). Using the relative cost of a particular elevator selection, the possibility of skipping that elevator in our ERR approach is:

Algorithm 1: Enhanced Round Robin in AdEle+.

```
$R$: Round robin counter
  for all elevators $k$ in elevator set of router $i$ do
      $C_{ik}^{rel}$ ← Calculate the relative cost ( (8))
      $P_{Sik}$ ← Calculate skip probability ( (9))
  end for
  while elevator is not selected do
      Skip elevator $R$ with probability $P_{SiR}$
      Go to next selection ($R$++)
  end while
  Update $C_{iR}$ after sending tail flit
```

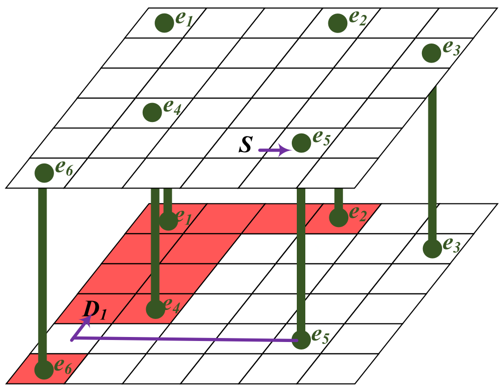
$$P_{Sik} = \begin{cases} 1-\xi, & \text{if } C_{ik}^{rel} \geq \frac{2}{|A_i|} \\ |A_i| \cdot \left(C_{ik}^{rel} - \frac{1}{|A_i|}\right) \cdot (1-\xi), & \text{if } \frac{2}{|A_i|} > C_{ik}^{rel} \geq \frac{1}{|A_i|} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Here, ξ is included to allow a small fraction of packets to be sent to highly congested routers so that the cost can be updated to reflect the current state ($\xi = 0.05$ in our experiments). To clarify the use of ξ, let us assume that $\xi = 0$. This would allow a value of 1 for P_{Sik} when there is high congestion. In this case, elevator k will not be selected in the ERR sequence at all and have no chance to update its elevator-selection cost (C_k). Even if the congestion at elevator k is resolved, we would constantly skip elevator k unless the cost for the other elevators rise and reduces C_{ik}^{rel} (see (8)). To address such an update failure, ξ allows every elevator to be selected with a low probability regardless of their P_{Sik}, so the cost function has a chance for updating. In (9), the expected load—i.e., when the load is evenly distributed over elevators—is $\frac{1}{|A_i|}$. Therefore, when the elevator load is below the expected load, the skip possibility is zero (the third line in (9)). On the other hand, if the load of an elevator is twice of the expected load ($\frac{2}{|A_i|}$) or more, the elevator is skipped with a high possibility (the first line in (9)). For the loads between $\frac{1}{|A_i|}$ and $\frac{2}{|A_i|}$, the elevator is skipped with respect to the extra load (the second line in (9)). The proposed ERR is described in Algorithm 1.

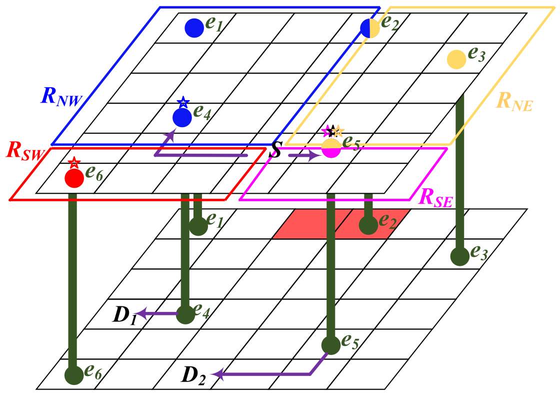
D. Elevator Selection in AdEle+ Under Low Traffic

AdEle+ employs ERR elevator selection to balance elevator utilization and improve network congestion. However, under low traffic loads, congestion is not a concern and employing ERR can increase both the latency and the energy by taking non-minimal paths. Therefore, we propose AdEle+ to use a distance-based routing when the congestion at the elevators is expected to be low. We also proposed a mechanism, which will be elaborated in the next subsection, to dynamically switch between the ERR and distance-based selections.

Unlike a regular mesh topology, finding the minimal path in PC-3DNoCs requires knowledge of the source, destination, and nearby elevators' locations. The minimal path is not necessarily through the source's nearest elevator. Let us consider an example



(a) Closest elevator selection example



(b) AdEle+'s Distance-Based (DB) elevator selection

Fig. 3. (a) Closest elevator selection example shows that 11 out of 36 destination routers in the bottom layer use non-minimal paths from source S (red destinations receive packets non-minimally). (b) Distance-Based (DB) elevator selection in AdEle+ shows example quadrants (R_{NW}, R_{NE}, R_{SW}, R_{SE}) based on S. Each elevator is color-coded based on their quadrant (with some elevators with two colors), and the regional closest elevator (RCE) for each region is marked with a color-coded star. Similarly, the closest elevator (CE) is marked with a black star (e_5). Our DB elevator selection will consider the paths through the RCE in the destination quadrant and the overall closest elevator. Our DB elevator selection only routes 2 out of 36 destinations non-minimally from source S.

shown in Fig. 3(a). Here, the conventional closest elevator selection (i.e., Elevator-First selection) would choose e_5 from S to D_1 instead of using the shortest path through e_4. To find the minimal path in PC-3DNoCs, we can apply an exhaustive search across all the elevators and save the results in each router. However, this is unscalable and imposes a rather large overhead. Instead, our efficient distance-based elevator selection takes advantage of the observation that the shortest path between many source-destination pairs goes through either 1) the closest elevator to the source, or 2) the closest elevator to the source in the same quadrant as the destination, assuming the source as the origin (0,0). We divide the source layer into 4 quadrant regions considering the source as the origin: northwest (R_{NW}), northeast (R_{NE}), southwest (R_{SW}), and southeast (R_{SE}) regions. In each of the 4 regions, we find one elevator as the closest elevator to the source in the quadrant. We define CE_{NW}, CE_{NE}, CE_{SW}, and CE_{SE} as the closest elevator in R_{NW}, R_{NE}, R_{SW}, and R_{SE} quadrants, respectively.

For example in Fig. 3(b), we show the R_{NW}, R_{NE}, R_{SW}, and R_{SE} quadrants using S as the source. In this example, $R_{NW} = \{e_1, e_2, e_4\}$, $R_{NE} = \{e_2, e_3, e_5\}$, $R_{SW} = \{e_6\}$, and $R_{SE} = \{e_5\}$. Therefore, $CE_{NW} = e_4$, $CE_{NE} = e_5$, $CE_{SW} = e_6$, and $CE_{SE} = e_5$. Note that the elevators on the border of each quadrant belong to both quadrants for the purpose of our distance-based elevator selection. For example, e_2 belongs to both R_{NW} and R_{NE} quadrants and e_5 is in both R_{NE} and R_{SE} quadrants. However, in the case of S, e_2 is not the quadrant's closest elevator in neither R_{NW} nor R_{NE}. Also, if there is no elevator in a quadrant, the closest elevator to the source (CE) is considered as that quadrant's regional closest elevator (RCE). For destination D_1, based on our observation earlier, we consider e_5 (closest to the source) and e_4 (closest in the destination quadrant), and select e_4 which is the shortest-path elevator. Similarly, for the path from S to D_2, we consider e_5 and e_6 and choose e_5, again the shortest-path elevator. Although this finds the shortest-path elevator in many situations, there are some cases that may not be minimal. For example, from S to the red routers—two routers out of 36 routers in the bottom layer—distance-based AdEle+ finds elevators with two hops more than the shortest path. However, using the closest elevator is even worse as it will fail to find the shortest-path elevator for all the routers in red shown in Fig. 3(a) (11 out of 36 routers).

To analyze the benefit of the proposed approach, we evaluated the average inter-layer distance of our approach for many network configurations (100 random elevator patterns with layer sizes of 4×4 and 8×8). Fig. 4 shows that the proposed distance-based selection in AdEle+ is able to achieve an average distance that is always better than the closest elevator and very close to the shortest-path selection. Fig. 4(a) and (c) show the average across 100 random elevator patterns' average distances, while Fig. 4(b) and (d) show the worst case among the 100 random elevator patterns' average distances. In all scenarios, DB is able to nearly achieve the same distance as the shortest path selection at every number of elevators. In these experiments, the average rate of non-minimal routing in DB AdEle+ was smaller than 4.1% and 7.5% for 4×4 and 8×8, which resulted in an increase of 3.2% and 2.7% in the average distance for 4×4 and 8×8, respectively. Due to its simplicity, the distance-based elevator selection only needs to store five different elevators regardless of the size of the network or the number of elevators: one for each quadrant and the closest elevator. Therefore, this approach is a very scalable and has a low overhead to help improve network latency and energy efficiency. Algorithm 2 details the distance-based elevator selection in AdEle+.

E. Adaptive Selection Between Distance-Based or Enhanced Round-Robin

To determine when to switch between the Distance-Based (DB) elevator selection and ERR, AdEle+ leverages the congestion information already gathered by the relative cost C^{rel} of elevators defined in (8). When C^{rel} is below a threshold (tr_{DB}) for all the elevators, AdEle+ will switch to the distance-based elevator selection and minimize hop counts. Otherwise, ERR is used to balance network congestion.

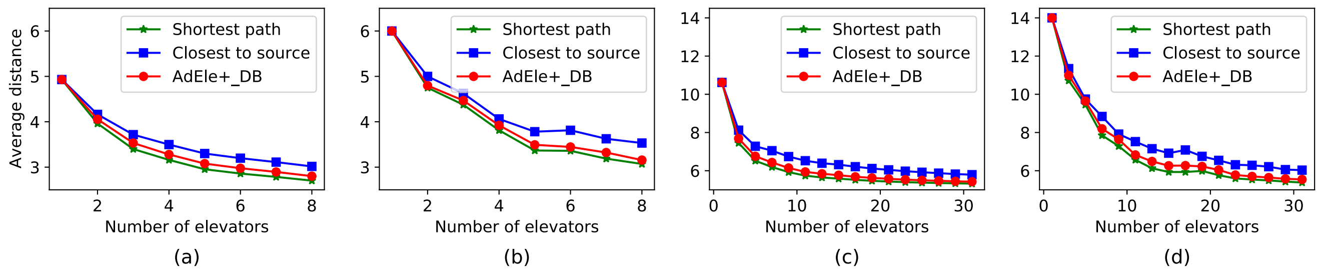


Fig. 4. Comparison of different elevator selection policies in terms of average distance under different network sizes and number of elevators. For each network size and number of elevators, 100 random elevator-placement patterns are evaluated. Here, we report both the average (average case) and the maximum (worst case) of all the 100 random elevator patterns' average distance. (a) 4×4 layer size - average case; (b) 4×4 layer size - worst case; (c) 8×8 layer size - average case; and (d) 8×8 layer size - worst case.

Algorithm 2: AdEle+'s Distance-Based Elevator Selection.

```
$RCE$: Regional closest elevator
$CE$: Closest elevator to source
$CE_{NE}, CE_{SE}, CE_{NW}$ and $CE_{SW}$: Closest elevator in
 north-east, south-east, north-west and south-west quadrant
 of source router
  if $S.x \geq D.x$ and $S.y \geq D.y$ then
      $RCE \leftarrow CE_{NE}$
  else if $S.x \geq D.x$ and $S.y \leq D.y$ then
      $RCE \leftarrow CE_{SE}$
  else if $S.x \leq D.x$ and $S.y \geq D.y$ then
      $RCE \leftarrow CE_{NW}$
  else
      $RCE \leftarrow CE_{SW}$
  end if
  Select $CE$ or $RCE$ based on source-destination distance
```

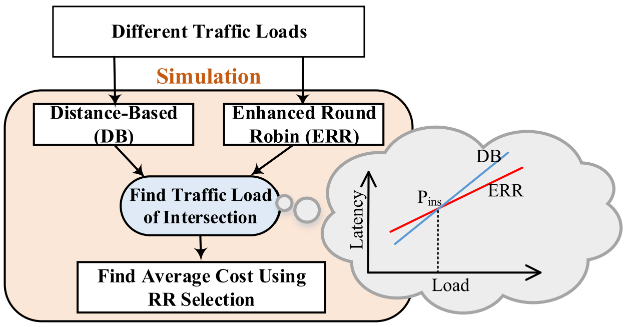

Different Traffic Loads
Simulation
Distance-Based (DB)
Enhanced Round Robin (ERR)
Find Traffic Load of Intersection
Find Average Cost Using RR Selection
DB
ERR
P_ins
Latency
Load

Fig. 5. Proposed framework to find tr_{DB}. ERR and distance-based elevator selections are simulated under different injection loads. The average cost (8) of the load where latency of both ERR and distance-based selections are equal is used as the minimal threshold (tr_{DB}).

The proposed framework to find tr_{DB} is shown in Fig. 5. In order to find the point at which ERR starts to outperform the distance-based selection (tr_{DB}), we perform network simulations with uniform traffic under increasing injection loads. At a high enough injection load, congestion starts to build up at the elevators and ERR begins to outperform the distance-based elevator selection. Then, we find the injection load (P_{ins}) when distance-based and ERR have the same latency and use the average cost of elevators (C) at the injection load (P_{ins}) to find tr_{DB}:

$$tr_{DB} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|A_i|}\sum_{k\in A_i} C_{ik}. \tag{10}$$

Although we use different traffic patterns in runtime, we will show in the next section that the average cost extracted using uniform traffic is close to the optimal one under different traffic patterns. Also, note that the proposed framework models the NoC at design time (here we used our simulator to model it) and calculate tr_{DB} to be used at design time. Therefore, there would not be any performance and area cost to estimate tr_{DB} at runtime.

IV. Experimental Results and Evaluations

In this section, we describe our simulation setup and present some parameter explorations in AdEle+ and its simulation results compared to the state-of-the-art elevator-selection algorithms.

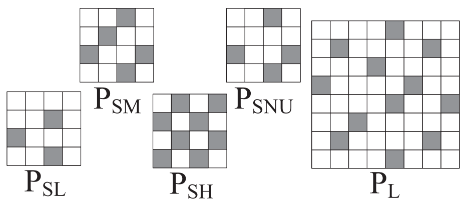
A. Simulation Setup

We compare the proposed AdEle+ with the state-of-the-art elevator-selection algorithms using Access Noxim [28], which is an extended version of Noxim [29], GEM5 [30] (for real-application traffic extraction), and Cadence Genus [31] (for hardware area analysis). We compare AdEle+ with two well-known elevator-selection algorithms: Elevator-First [8] and CDA [11]. Table I summarizes the simulation setup. We used Elevator-First [8] routing algorithm for deadlock freedom in our simulation (albeit, with a different elevator selection for CDA, AdEle, and AdEle+), although AdEle+ can be added to any other routing algorithms in PC-3DNoC.

In PC-3DNoCs, the number and location of elevators can be affected by different performance-cost trade-off considerations [3]. Therefore, AdEle+ is evaluated using different PC-3DNoC elevator-placement patterns to show that its efficacy is independent of any such patterns. Four elevator patterns are considered for a $4\times4\times4$ network (see Table I): P_{SL}, P_{SM}, and P_{SH}, which are patterns with low, medium, and high density of elevators, respectively; and a non-uniform pattern P_{SNU}, which is designed based on P_{SM} where one of the elevators is faulty (i.e., one elevator is removed due to, for example,

TABLE I
SIMULATION SETUP

Simulator	Access Noxim [28] (for latency & energy analyses)
Network size	4×4×4 and 8×8×4
Routing and VC selection (w/o elevator selection)	Elevator-First [8] (used to avoid deadlock)
Switching	Wormhole switching
Buffer depth	4 flits
flit width	32 bits
Packet size (flits)	10–30 (uniform) for synthetic 18 for real application
Traffic pattern	Uniform, Shuffle, and Real
PC-3DNoC Elevator Placements	



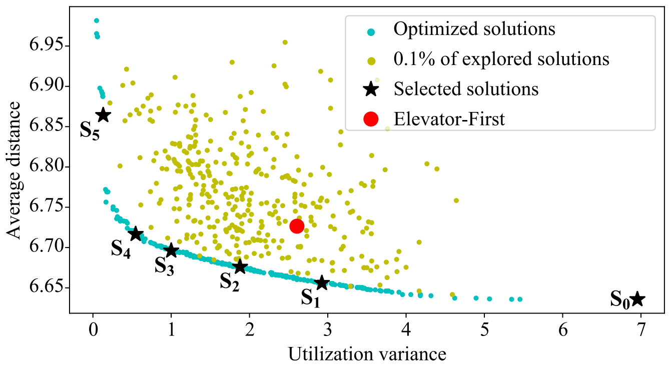


Fig. 6. Elevator-subset solutions found by AMOSA in AdEle+.

TABLE II
PERFORMANCE OF SELECTED SOLUTIONS FROM FIG. 6

	Elev. First	Optimized solutions					
		S_0	S_1	S_2	S_3	S_4	S_5✓
Latency*	161.4	396	209	156.6	76.9	67.4	56.6
Energy#	94.4	93.1	94.2	94.6	94.4	94.8	98.3

*Avg. Latency (cycles) #Energy/flit (nj) ✓Selected

TSV manufacturing faults [7]). These patterns are optimized using Simulated Annealing (SA) to minimize the overall average distance. A large network elevator pattern P_L (8×8×4) with optimized average distance is also considered in our evaluation to show the scalability of AdEle+.

AdEle+'s offline optimization (see Section III-B) is implemented in Python to extract the elevator subsets for each router. These subsets are then added to the AdEle+ router implemented in Access Noxim simulator [28]. As it is hard to predict online traffic accurately at the design time, for the offline optimization we considered uniform traffic, the most pessimistic assumption (i.e., traffic is not known *a priori*), while the network evaluations are done using different synthetic and real-application traffic patterns. Our analyses will demonstrate that AdEle+ does not require runtime traffic in its offline optimization as its online selection policy will adjust to runtime traffic (see AdEle_RR versus AdEle+ in evaluations presented in Figs. 10 and 11). However, if the traffic is known, AdEle+ can use the runtime traffic during elevator-subset selection to offer further latency and energy improvements.

B. Parameter Exploration and Optimization

1) AMOSA Elevator-Subset Exploration: As discussed in Section III-B, AMOSA finds various solutions with different elevator utilization variances and average distances. To show the solution-selection process in AdEle+ and as an example, the optimization for P_L is detailed here. A small sample of AMOSA's explored solutions is shown in Fig. 6. As AMOSA explores the solution space, it makes its way towards the Pareto front (blue curve) to find the optimal trade-offs between utilization variance (see (3)) and average distance (see (5)). Given the final set of solutions, a desired solution can be selected depending on the importance of energy efficiency (average distance) and latency (utilization variance). For brevity, we simulated several solutions spread along the Pareto front (S_0 to S_5) and summarized the results in Table II. As expected, lower utilization variance and lower average distance improves the latency and energy consumption, respectively. As we are able to significantly reduce the latency with fairly minimal increases in energy, we select S_5 for further analysis. Moreover, as we discussed in Section III-D, AdEle+ will dynamically switch to DB elevator selection to save energy when the traffic load is low and congestion is not a concern. We follow the same procedure for P_{SL}, P_{SM}, P_{SH}, and P_{SNU} to find an optimized set of elevator subsets.

In the elevator subset optimization, we establish a minimum ($ESet_{min}$) and maximum ($ESet_{max}$) number of elevators that each elevator subset may have. This allows AMOSA to choose different elevator subset sizes for different routers. Although having a large number of elevators in the subset helps the router adapt to different traffic scenarios, considering a very large number of elevators also increases the chance of selecting distant elevators, which impacts the energy efficiency.

To analyze the impact of elevator set size, average latency and energy-per-flit for P_L is shown in Fig. 7 using uniform and shuffle traffic patterns. Two ranges of elevator subset sizes are considered: $ESet_{min} = 1$ and $ESet_{max} = 2$ ($ESet =$1–2); and $ESet_{min} = 2$ and $ESet_{max} = 3$ ($ESet =$2–3). For elevator routers, we force them to only use their local elevator because the cost of rerouting outweighs the load balancing benefits. Since the elevator subsets are optimized based on uniform traffic, adding more elevators in the subset to help adapt to runtime traffic does only little to improve the latency under the uniform traffic. On the other hand, in an unseen traffic pattern like shuffle, $ESet =$2–3 shows better performance because it can benefit from a larger elevator subset size to adapt to the new traffic. We also evaluated $ESet_{min} = ESet_{max} = 3$, and $ESet_{min} = 3$ and $ESet_{max} = 4$. However, we did not observe any significant improvement in the latency, but we observed

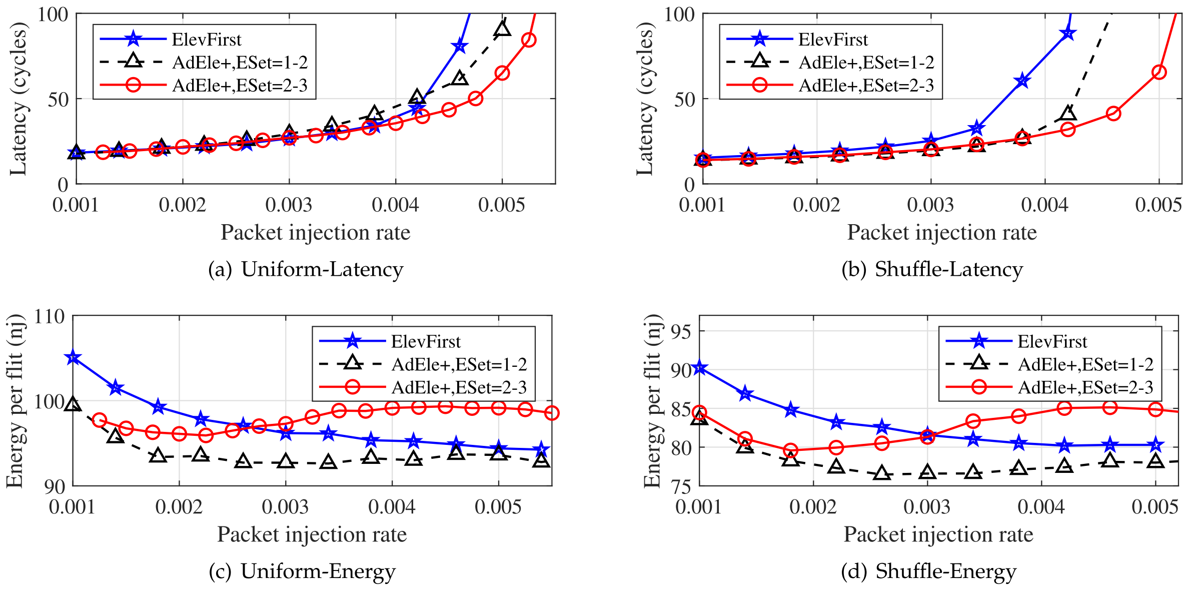


Fig. 7. Impact of elevator set size on (a and c) uniform traffic and (b and d) shuffle traffic.

energy overhead. Similar to these results, we found that, for different network sizes, traffic patterns, and elevator patterns (See Table I), $ESet_{min} = 2$ and $ESet_{max} = 3$ will result in a better performance, and therefore we use this configuration.

2) Distance-Based Selection Threshold (tr_{DB}): As discussed in Section III-E, we find the DB selection threshold (tr_{DB}) by varying the traffic injection rates and examining the latency. Results for P_L are shown in Fig. 8. As expected, DB elevator selection has a lower latency when injection rates are low, while ERR improves network congestion and outperforms DB selection at higher injection rates. However, as ERR can use non-minimal paths, it consistently has worse energy consumption compared to DB (see Fig. 8(b)). To improve latency, ideally we would like to use DB when the packet injection rate is below 0.00225 (the intersection occurred when packet injection rate is 0.00225), and ERR otherwise. Therefore, at an injection rate of 0.00225, we calculate the average cost of elevator selections (for all selections in all the routers) and using (10) tr_{DB} is calculated, which is 0.15 for P_L.

To show the impact of different values of tr_{DB}, we show the latency and energy results across a range of tr_{DB} for P_L in Fig. 9. As it can be seen, the threshold value we earlier determined (i.e., $tr_{DB} = 0.15$) is able to achieve a relatively low latency at both high and low injection rates, demonstrating the efficacy of our approach. Moreover, as shown in Fig. 9(b), energy consumption is improved in low injection rates, compared to ERR, due to switching to DB selection.

Following the same procedure, we find tr_{DB} for P_{SL}, P_{SM}, P_{SH}, and P_{SNU} across different traffic patterns (see Table III). Although there is a small difference in optimal tr_{DB} values, we do not expect to see a large impact on the overall results if we use the tr_{DB} computed based on the uniform traffic for other traffic patterns. From Fig. 9, we can see that even moving from $tr_{DB} = 0.15$ to $tr_{DB} = 0.5$—a range much larger than what can be

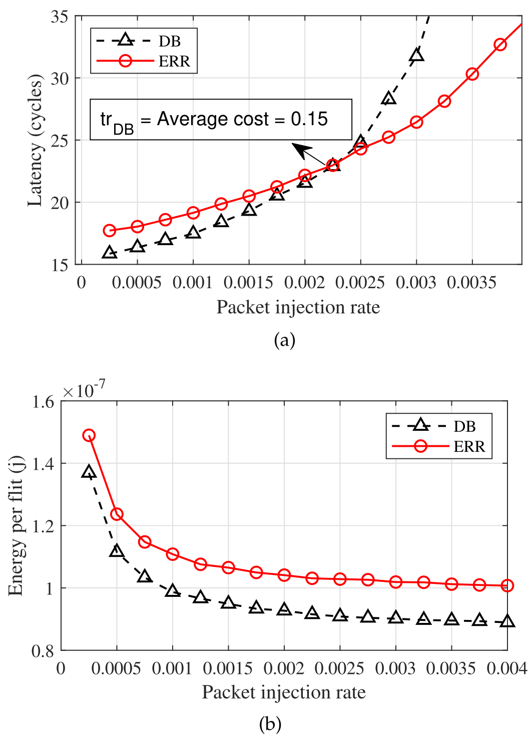
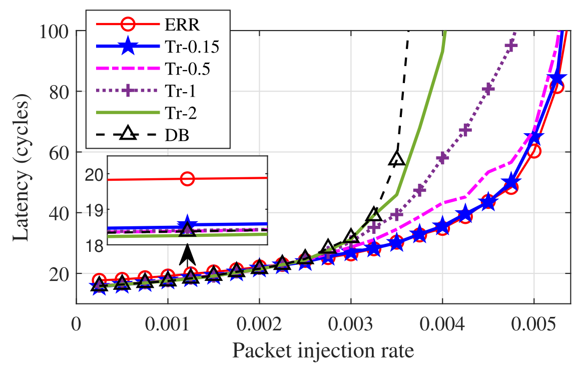


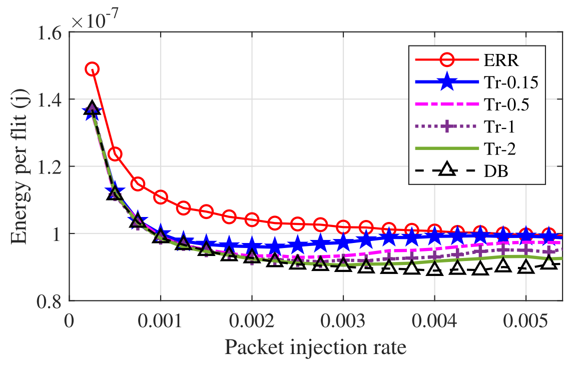
Fig. 8. Comparison of minimal versus ERR elevator selection under Uniform traffic: (a) average latency and (b) energy per flit. This is used to find tr_{DB}.

TABLE III
tr_{DB} FOR DIFFERENT ELEVATOR PLACEMENTS

	P_{SL}	P_{SM}	P_{SH}	P_{NU}	P_L
Uniform	0.08	0.07	0.06	0.07	0.15
Shuffle	0.13	0.06	0.1	0.07	0.3
Transpose	0.07	0.05	0.04	0.05	0.06



(a)



(b)

Fig. 9. Comparison of ERR selection under Uniform traffic with different values of tr_{DB} for P_L in (a) average latency and (b) energy per flit.

seen in Table III—has a relatively small impact on the latency and energy. Therefore, for each PC-3DNoC configuration, we assume the optimal tr_{DB} found under the uniform traffic for all the traffic patterns.

C. Evaluation Results

In this subsection, we evaluate AdEle+ compared to the state-of-the-art selection algorithms (Elevator-First [8] and CDA [11]) in terms of latency, energy, and area efficiency. Our evaluation includes both synthetic and real traffic patterns.

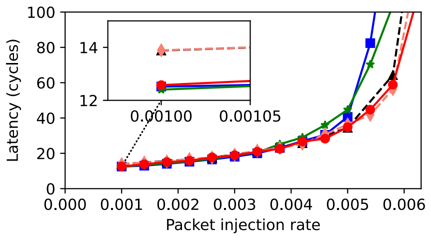
1) AdEle+ Performance Under Synthetic Traffic: We first compare the average latency in AdEle+ under uniform and shuffle synthetic traffic patterns, with Elevator-First, CDA and AdEle in Figs. 10 and 11. Under all the elevator placements (see Table I) and both traffic patterns, AdEle+ achieves the lowest latency and highest saturation throughput. Even though CDA employs global traffic information, AdEle+ still shows better performance while only relying on local information. In this work, we do not consider the high cost of CDA's global information sharing and optimistically assume that the information is instantaneously received at every router. In reality, CDA will likely perform even worse with stale information or include significant implementation overhead.

With a higher elevator density (e.g., P_{SH}) or larger horizontal dimensions (e.g., P_L), the intra-layer traffic will become more critical. AdEle+ still shows better performance in these cases. To demonstrate the efficacy of our DB approach, we compare AdEle+ with the policy that uses the ERR approach at all injection rates (AdEle [26]). As it can be seen, the DB approach is able to significantly improves the average latency at low injection rates (see in-set plots). Since the DB approach is able to utilize the shortest path when traffic congestion is not an issue, AdEle+ is able to lower the latency compared to the ERR approach. At high injection rates, AdEle+ switches over to ERR and has a negligible latency overhead.

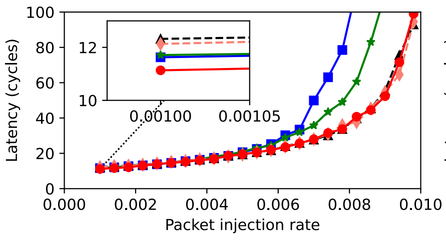
In addition, recall that AdEle+'s offline optimization is performed using uniform traffic only. Yet, as Fig. 11 shows, while the shuffle traffic is new for AdEle+, it still achieves the lowest latency because its online selection policy can monitor runtime congestion and select better elevators. We also include the average latency of AdEle with conventional round-robin selection (called AdEle_RR). As can be seen, AdEle+'s proposed online skipping policy achieves higher improvements in latency compared to RR under both uniform and shuffle traffic patterns. Notably, AdEle+ with ERR (shown as AdEle+ in the results) has more improvement with the unseen traffic (i.e., shuffle) than the one used in its offline optimization (i.e., uniform). This demonstrates that ERR can successfully adapt to new traffic patterns.

To further explain the latency improvement in AdEle+, we show the elevator traffic load distribution for P_{SL} normalized to the average router load in Fig. 12(a). The white bar shows the average load over elevator-less routers. The other colored bars show the load over different elevators as indicated in the inset. As it can be seen, AdEle+ better balances the load across the three elevators and reduces the load on the highest utilized elevator. To help quantify the congestion at an elevator, we examine the average residency of flits on elevators (i.e., the time that a flit is waiting at an elevator) in Fig. 12(b). Due to the high traffic load on the blue elevator in Elevator-First and CDA, we see a high flit residency on the blue elevator. By balancing the traffic load, AdEle+ shows almost the same flit residency on the three elevators. Therefore, balancing the traffic load on the elevators results in less waiting time at the elevators, and hence the average latency is improved. We also analyzed flit residency on elevators for other patterns in Fig. 12(c). The same trend is seen: AdEle+ significantly improves the average flit residency because of better load balancing on the elevators. In Fig. 12, packet injection rate is the injection rate at which latency is 3× zero-load latency: P_{SL}: 0.0054, P_{SM}: 0.0078, P_{SM}: 0.011, P_{SNU}: 0.0062, and P_L: 0.005.

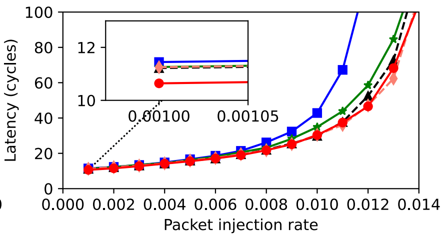
We compare the energy consumption in different elevator-selection algorithms and under various elevator placements in Fig. 13. Under low injection rates, AdEle+ always has the lowest energy consumption with an average of 5.6% energy improvement over AdEle, because it switches to DB selection and uses the minimal paths. As it can be seen, DB selection of AdEle+ offers significantly lower energy compared to AdEle [26]. However, because AdEle+ takes non-minimal paths at higher injection rates to improve traffic loads across elevators, it typically imposes a small energy overhead. When using P_{SL}, the low density of elevators causes more pressure on each elevator and increases the chance of taking non-minimal



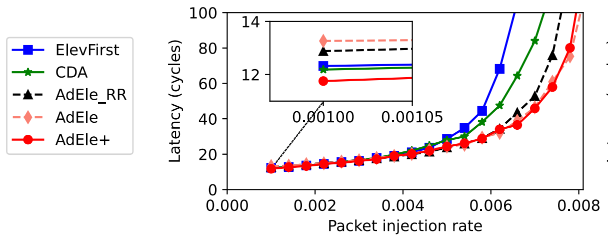
(a) P_{SL}



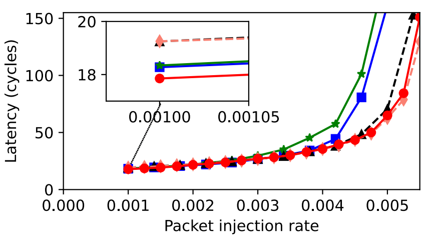
(b) P_{SM}



(c) P_{SH}

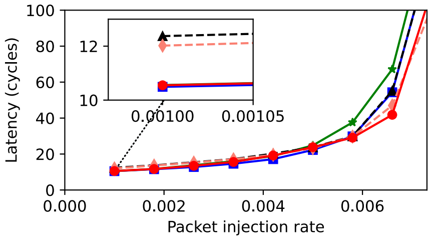


(d) P_{SNU}

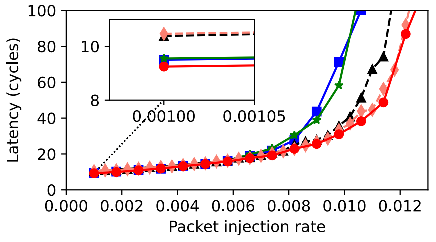


(e) P_{L}

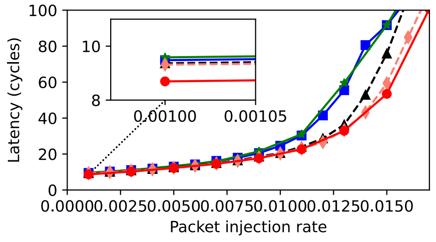
Fig. 10. Average latency for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under uniform traffic and using different elevator placements.



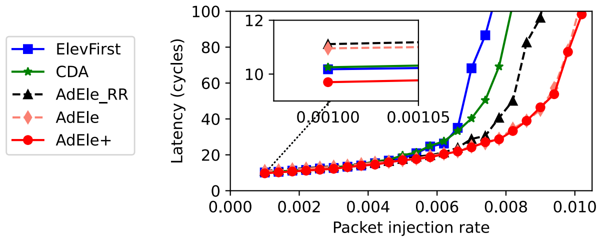
(a) P_{SL}



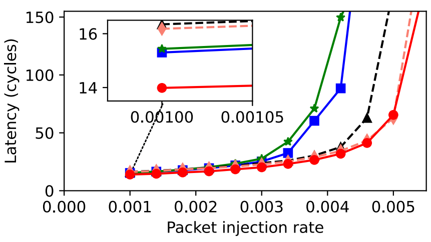
(b) P_{SM}



(c) P_{SH}

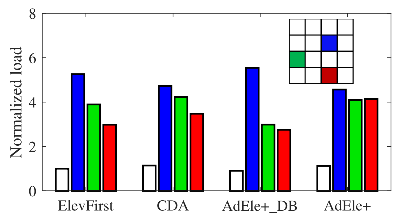


(d) P_{SNU}

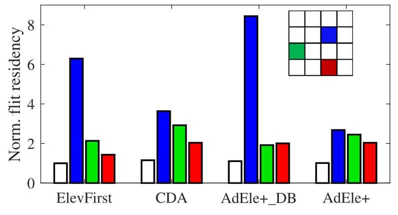


(e) P_{L}

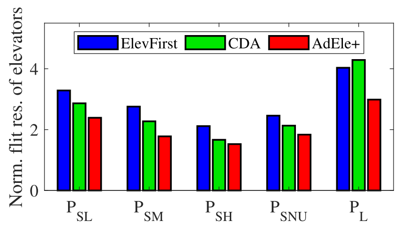
Fig. 11. Average latency for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under shuffle traffic and using different elevator placements.



(a)

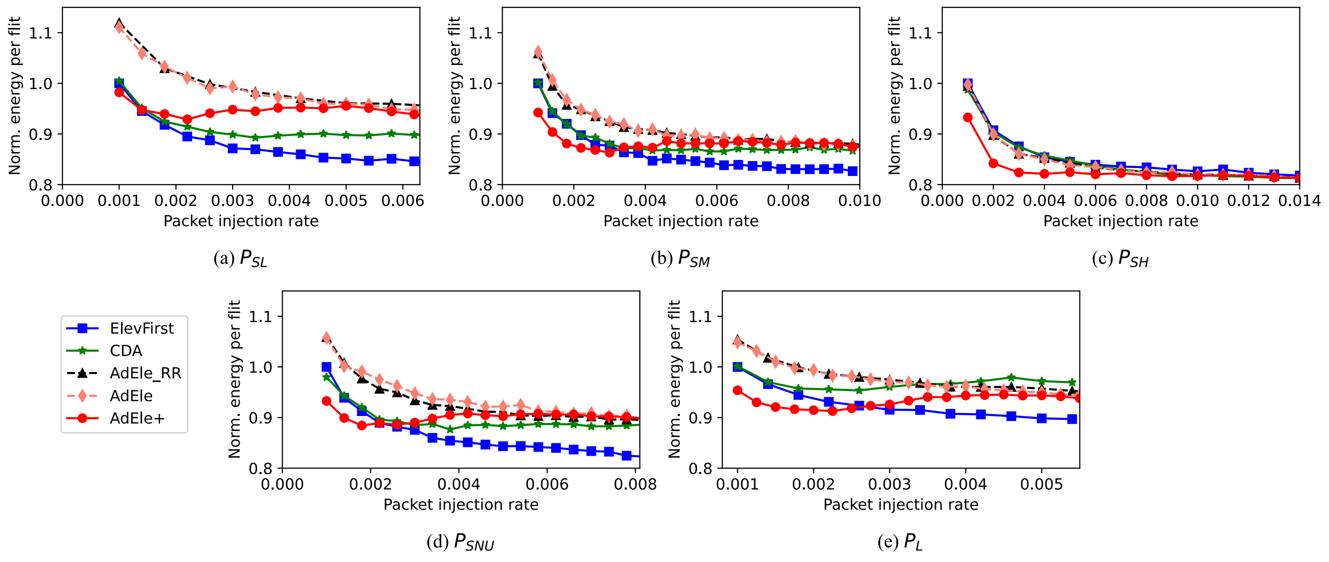


(b)



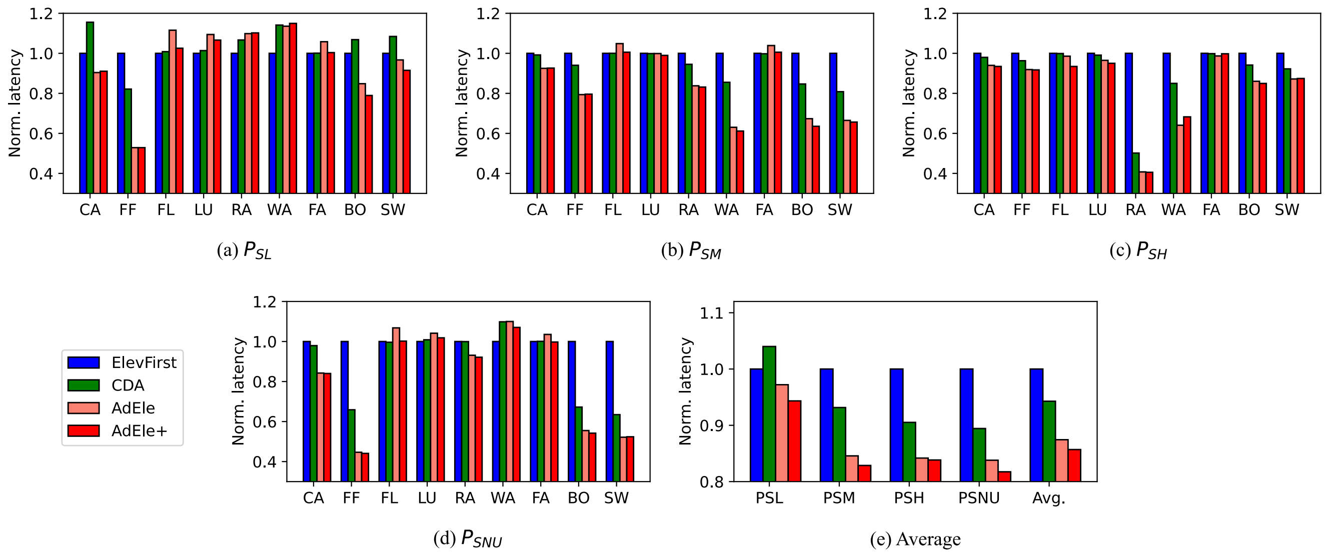
(c)

Fig. 12. Comparison of elevator traffic distribution for Elevator-First (ElevFirst), CDA, and AdEle+ in terms of (a) traffic load over elevators (blue, green, and red) normalized to the average load over horizontal links (white bars) of Elevator-First; (b) average flit residency over elevators; and (c) average flit residency of all elevators normalized to average flit residency of horizontal links. AdEle+_DB shows DB selection of AdEle+.



(a) P_{SL} (b) P_{SM} (c) P_{SH} (d) P_{SNU} (e) P_L

Fig. 13. Normalized energy per flit for Elevator-First, CDA, AdEle, AdEle_RR, and AdEle+ under uniform traffic with different injection rates.



(a) P_{SL} (b) P_{SM} (c) P_{SH} (d) P_{SNU} (e) Average

Fig. 14. Latency for Elevator-First (ElevFirst), CDA, AdEle and AdEle+ normalized to ElevFirst under real-application traffic with different elevator placements.

paths, incurring at most 6.4% energy overhead compared to CDA. However, distributing the traffic properly is especially important with a low number of elevators as the traffic pressure is already high and unbalanced loads can greatly affect the performance. If maximum energy efficiency is desired, then AdEle+ can use configurations with lower energy but higher latency (see Table II). For PC-3DNoC configurations with a higher density of elevators (P_{SM}, P_{SH}, and P_{SNU}), there is almost no energy overhead compared to CDA and negligible overhead compared to Elevator-First at higher injection rates. Although in a non-uniform elevator placement like P_{SNU} congestion aware selection of elevators can potentially result in a larger average distance, AdEle+ has a negligible energy overhead due to the efficient offline optimization. Finally, using P_L, AdEle+ shows energy consumption improvement compared to CDA. The reason is that AdEle+ selects an elevator among a set of nearby elevators, while CDA is free to select any elevator including those farther away.

2) AdEle+ Performance Under Real-Application Traffic: We extracted the traffic of several SPLASH-2 [32] and PARSEC [33] benchmarks using Gem5 [30] for real-application simulations. We obtained the real-application traffic traces using Gem5 simulations in full-system mode with 64 x86 cores, four coherence directories, four shared L2 cache banks (each core also has a private L1 cache), 64 threads, and simmedium input size. Because Gem5 is limited to 64 cores, we demonstrate our results for P_{SL}, P_{SM}, P_{SH} and P_{SNU}. As shown in Fig. 14(a)–(e), AdEle+ improves the network latency by 9.3%,

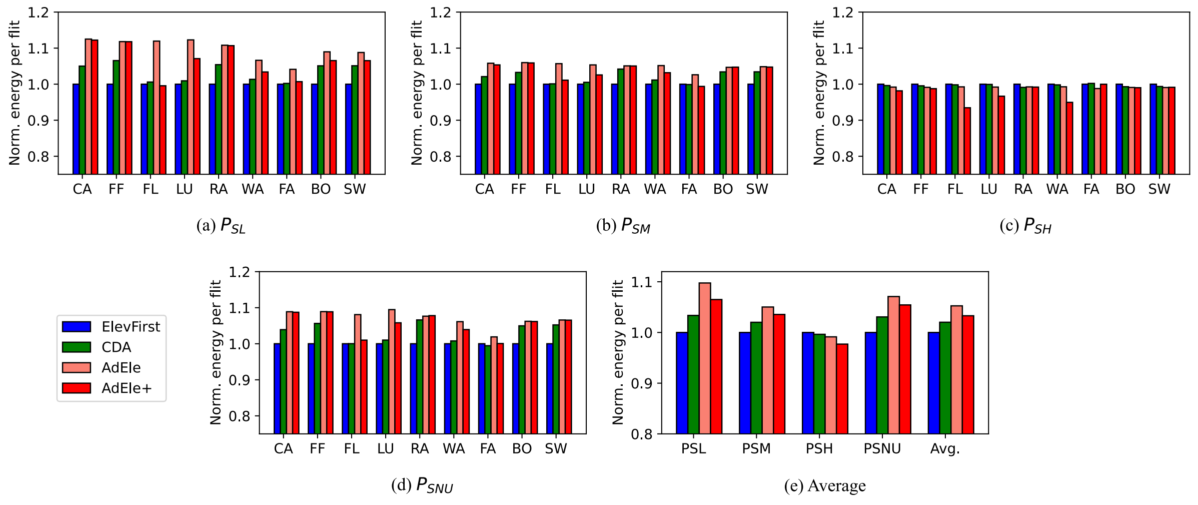


Fig. 15. Energy for Elevator-First (ElevFirst), CDA, AdEle, and AdEle+ normalized to ElevFirst under real-application traffic with different elevator placements.

11.2%, 8.4%, and 8.9% (9.5% on average) compared to CDA; by 5.7%, 17.2%, 16.2% and 18.3% (14.3% on average) compared to Elevator-First; and by 3%, 2.1%, 0.5% and 2.5% (2% on average) compared to AdEle using P_{SL}, P_{SM}, P_{SH} and P_{SNU}, respectively. Note that the first two letters of application are used on the x-axis in the figures—CA: canneal, FF: fft, FL: fluidanimate, Lu: lu, RA: radix, WA: water, facesim: FA, bodytrack: BO and swaption: SW. In particular, AdEle+ has more improvements in applications with higher traffic loads (canneal, fft, radix, water, bodytrack and swaption), as there is more opportunity to reduce the resulting elevator congestion. In applications with lower traffic loads (fluidanimate, lu and facesim), AdEle+ maintains almost similar performance to the other approaches as there is little contention on the elevators and the latency is close to zero-load latency. Although P_{SL} still shows some improvements for AdEle+, the lower number of elevators (three in P_{SL}) results in minimal opportunity for AdEle+ to redirect traffic and improve latency. Compared to AdEle, AdEle+ shows lower latency (2% on average) especially in low load traffic (e.g., 5.9% averaged across network configurations for fluidanimate), due to its efficient DB elevator selection.

Fig. 15 shows the average energy-per-flit for each elevator-placement pattern (P_{SL}, P_{SM}, P_{SH}, and P_{SNU}), normalized to Elevator-First. AdEle+ imposes small overhead because it routes packets over non-minimal paths in case of congestion to improve the latency. Compared to CDA, AdEle+ has a negligible overhead using P_{SL}, P_{SM}, and P_{SNU}, while it slightly improves the energy consumption under P_{SH}. On average, AdEle+ has improved energy consumption by 2%, in comparison with AdEle, due its efficient DB selection.

To see how sensitive AdEle+ is to packet size, we perform the same analysis for a packet size of five flits. In Fig. 16, we present the average latency and energy-per-flit over all network configurations (P_{SL}, P_{SM}, P_{SH}, and P_{SNU}) and all nine real applications. To evaluate the effect of packet size, we maintain the same flit injection rate in both 5-flit and 18-flit cases. Although flit injection rate is the same, as the chance of congestion is reduced under the small packet size, both CDA and

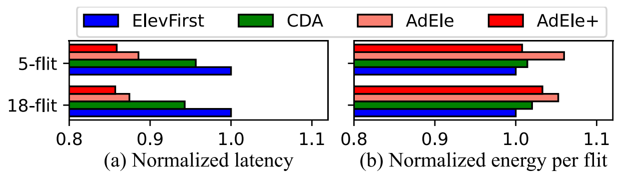


Fig. 16. Packet size effect on latency and energy-per-flit.

AdEle, which are congestion-aware approaches, have slightly less improvement in comparison with Elevator-First under the small packet size. On the other hand, AdEle+ offers slightly lower latency (0.2% improvement) and energy-per-flit (2.5% improvement) in the small packet size. AdEle+ can update the cost function faster with a higher packet frequency (i.e., smaller packet size) because, unlike CDA, AdEle+ updates its cost function every packet instead of using a time interval for the update (i.e., an event driven approach instead of a time driven one). This allows AdEle+ to more quickly adapt to traffic dynamics and improve performance.

3) Hardware-Area Analysis and Comparison: The hardware of Elevator-First, AdEle+, and CDA are implemented and analyzed using Cadence Genus [31] in 45 nm technology. Here, we consider a 1 GHz clock. For AdEle and AdEle+, we consider 8-bit precision for (6) and (7) and 5-bit precision for (8) and (9). We find that 8-bit precision is sufficient to cover the range of latency values before the network reaches saturation and maximum latency. However, we chose a lower precision for relative cost to save area overhead due to the division operation. We found that at 5 bits, the approximation resulting from lower precision resulted in a negligible effect (less than 0.1%) on AdEle+'s latency and energy. Therefore, we use these level of precision assumptions in our latency and energy evaluation presented in Section IV. The results are shown in Table IV. Compared to CDA, AdEle+ has smaller area overhead. Since AdEle+ only requires local traffic information, AdEle+ does not affect router frequency and AdEle+ calculations can be done in 1 cycle. However, CDA's area overhead is an optimistic

TABLE IV
AREA OVERHEADS FOR THE DIFFERENT ELEVATOR SELECTION ALGORITHMS

	Cycles	Router area (μm^2)	
Base (ElevFirst)	1	35550	Overhead
CDA*	2	41088	14.4%
AdEle	1	36875	3.7%
AdEle+	1	36954	3.9%

*global information sharing is not included.

assumption here as it does not include any overhead related to the actual sharing of information. Therefore, real CDA will likely impose higher area and latency overhead. Also, AdEle+ does not affect the router stages and will scale well with the network size, while CDA requires an additional cycle (or more for larger networks) to update its tables. As mentioned in Section III-D, AdEle+ will, at most, only need to store five elevator locations regardless of the system size and elevator density to support the DB selection. Compared to AdEle, AdEle+ imposes a negligible area overhead (i.e, 0.2%) due to DB selection, but DB selection improves both energy and latency significantly. In summary, the results presented in this section using different traffic patterns and network configurations show the promise of AdEle+ to manage congestion on elevators with low hardware overhead.

V. CONCLUSION

Elevator selection plays a crucial role in the network latency and energy efficiency of partially connected 3D NoCs. This paper has combined an offline elevator subset optimization process with an online elevator selection, to create a lightweight adaptive congestion- and energy-aware elevator-selection algorithm, called AdEle+, that addresses the traffic congestion on elevators while delivering packets with less hop counts in low traffic circumstances. By employing a set of elevators instead of one elevator for each source router, AdEle+ is able to adapt to runtime traffic loads and select the best elevator during runtime. Moreover, AdEle+ only requires local router information and is able to improve average latency in various scenarios under both synthetic and real traffic. AdEle+ also improves the energy consumption in some applications, especially under low traffic loads where there is a low chance of congestion. Results indicate the promise of AdEle+ to improve the network latency in PC-NoCs and prevent network hot-spots. Therefore, the work presented in this paper can help in designing low-latency and energy-efficient networks for high performance 3D manycore systems.

REFERENCES

[1] L. Benini and G. De Micheli, “Networks on chips: A new SoC paradigm,” *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] A. Coelho, A. Charif, N.-E. Zergainoh, and R. Velazco, “FL-RuNS: A high-performance and runtime reconfigurable fault-tolerant routing scheme for partially connected three-dimensional networks on chip,” *IEEE Trans. Nanotechnol.*, vol. 18, pp. 806–818, 2019.

[3] A. I. Arka, S. Gopal, J. R. Doppa, D. Heo, and P. P. Pande, “Making a case for partially connected 3D NoC: NFIC versus TSV,” *ACM J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 4, pp. 1–17, 2020.

[4] M. Bahmani, A. Sheibanyrad, F. Pétrot, F. Dubois, and P. Durante, “A 3D-NoC router implementation exploiting vertically-partially-connected topologies,” in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2012, pp. 9–14.

[5] F. Wang, Z. Zhu, Y. Yang, X. Yin, X. Liu, and R. Ding, “An effective approach of reducing the keep-out-zone induced by coaxial through-silicon-via,” *IEEE Trans. Electron Devices*, vol. 61, no. 8, pp. 2928–2934, Aug. 2014.

[6] T. Frank et al., “Resistance increase due to electromigration induced depletion under TSV,” in *Proc. Int. Rel. Phys. Symp.*, 2011, pp. 3F.4.1–3F.4.6.

[7] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, “Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip,” *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3591–3604, Dec. 2015.

[8] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, “Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3D NoCs,” *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 609–615, Mar. 2013.

[9] E. Taheri, M. Isakov, A. Patooghy, and M. A. Kinsy, “Addressing a new class of reliability threats in 3-D network-on-chips,” *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 39, no. 7, pp. 1358–1371, Jul. 2020.

[10] B. Niazmand, S. P. Azad, J. Flich, J. Raik, G. Jervan, and T. Hollstein, “Logic-based implementation of fault-tolerant routing in 3D network-on-chips,” in *Proc. IEEE/ACM 10th Int. Symp. Netw.-Chip*, 2016, pp. 1–8.

[11] Y. Fu et al., “Optimizing vertical link placement and congestion aware dynamic elevator assignment for partially connected 3D-NoCs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 10, pp. 1957–1970, Oct. 2021.

[12] S. Foroutan, A. Sheibanyrad, and F. Petrot, “Assignment of vertical-links to routers in vertically-partially-connected 3-D NoCs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 8, pp. 1208–1218, Aug. 2014.

[13] Y. Fu et al., “Congestion-aware dynamic elevator assignment for partially connected 3D-NoCs,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2019, pp. 1–5.

[14] E. Taheri, S. Pasricha, and M. Nikdast, “DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet networks,” in *Proc. Des., Automat. Test Europe Conf.*, 2022, pp. 1047–1052.

[15] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, “LEAD: An adaptive 3D-NoC routing algorithm with queuing-theory based analytical verification,” *IEEE Trans. Comput.*, vol. 67, no. 8, pp. 1153–1166, Aug. 2018.

[16] E. Taheri, S. Pasricha, and M. Nikdast, “ReSiPI: A reconfigurable silicon-photonic 2.5D chiplet network with PCMs for energy-efficient interposer communication,” in *Proc. IEEE/ACM 41st Int. Conf. Comput.-Aided Des.*, 2022, pp. 1–9.

[17] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, “A simulated annealing-based multiobjective optimization algorithm: AMOSA,” *IEEE Trans. Evol. Computat.*, vol. 12, no. 3, pp. 269–283, Jun. 2008.

[18] M. Jiang, I. A. Papistas, and V. F. Pavlidis, “Cost modeling and analysis of TSV and contactless 3D-ICs,” in *Proc. Great Lakes Symp. VLSI*, 2020, pp. 519–524.

[19] D. Velenis, M. Stucchi, E. J. Marinissen, B. Swinnen, and E. Beyne, “Impact of 3D design choices on manufacturing cost,” in *Proc. IEEE Int. Conf. 3D Syst. Integration*, 2009, pp. 1–5.

[20] B. Li, X. Wang, A. K. Singh, and T. Mak, “On runtime communication and thermal-aware application mapping and defragmentation in 3D NoC systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2775–2789, Dec. 2019.

[21] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, “A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs,” *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3265–3279, Nov. 2016.

[22] T. C. Xu, G. Schley, P. Liljeberg, M. Radetzki, J. Plosila, and H. Tenhunen, “Optimal placement of vertical connections in 3D network-on-chip,” *J. Syst. Archit.*, vol. 59, no. 7, pp. 441–454, 2013.

[23] A. Charif, A. Coelho, M. Ebrahimi, N. Bagherzadeh, and N.-E. Zergainoh, “First-last: A cost-effective adaptive routing solution for TSV-based three-dimensional networks-on-chip,” *IEEE Trans. Comput.*, vol. 67, no. 10, pp. 1430–1444, Oct. 2018.

[24] J. Lee, K. Kang, and K. Choi, “REDELF: An energy-efficient deadlock-free routing for 3D-NoCs with partial vertical connections,” *ACM J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 3, pp. 1–22, 2015.

[25] F. Vahdatpanah, M. Elahi, S. Kashi, E. Taheri, and A. Patooghy, “3DEP: A efficient routing algorithm to evenly distribute traffic over 3D network-on-chips,” in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2019, pp. 237–241.

[26] E. Taheri, R. G. Kim, and M. Nikdast, “AdEle: An adaptive congestion-and-energy-aware elevator selection for partially connected 3D NoCs,” in *Proc. IEEE/ACM 58th Des. Automat. Conf.*, 2021, pp. 67–72.

[27] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[28] K.-Y. Jheng, C.-H. Chao, H.-Y. Wang, and A.-Y. Wu, "Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip," in *Proc. Int. Symp. VLSI Des., Automat. Test*, 2010, pp. 135–138.

[29] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Proc. 26th Int. Conf. Appl.-Specific Syst., Architectures Processors*, 2015, pp. 162–163.

[30] N. Binkert et al., "The Gem5 simulator," *ACM SIGARCH Comput. Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[31] Cadence genus synthesis tool. 2015. [Online]. Available: www.cadence.com

[32] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *ACM SIGARCH Comput. Architecture News*, vol. 23, no. 2, pp. 24–36, 1995.

[33] C. Bienia and K. Li, *Benchmarking modern multiprocessors. Princeton*, Princeton, NJ, USA: Princeton Univ., 2011.

Ebadollah Taheri (Student Member, IEEE) received the MSc degree in electronic engineering from the Iran University of Science and Technology, in 2016. He is currently working toward the PhD degree in computer engineering with the Department of Electrical and Computer Engineering, Colorado State University, USA. His research interests are in the field of Reliable and High-Performance Computer Architectures with a focus on On-Chip Interconnection Networks.

Ryan Gary Kim (Member, IEEE) received the PhD degree in electrical engineering from Washington State University. He is an assistant professor with the Electrical and Computer Engineering Department, Colorado State University, Fort Collins. Afterwards, he was a postdoctoral researcher with Carnegie Mellon University. His current research interests include electronic design automation techniques for scalable, fully-adaptive manycore systems and domain-specific architectures.

Mahdi Nikdast (Senior Member, IEEE) received the PhD degree in electronic and computer engineering from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2014. He is an assistant professor with the Department of Electrical and Computer Engineering, Colorado State University (CSU), Fort Collins, CO. From 2014 to 2017, he was a postdoctoral fellow with McGill University and Polytechnique Montreal, Canada. He is the director of the Electronic-PhotoniC System Design (ECSyD) Laboratory, CSU. His primary research goals are focused on the design methodologies and development of high-performance computing and data-communication systems employing emerging technologies while emphasizing energy efficiency and robustness. He and his students have received multiple Best Paper awards for their work in the area of integrated photonics and design for manufacturability.