Efficient Direct-Connect Topologies for Collective Communications

Liangyu Zhao, *University of Washington;* Siddharth Pal, *RTX BBN Technologies;* Tapan Chugh, *University of Washington;* Weiyang Wang, *MIT CSAIL;* Jason Fantl, Prithwish Basu, and Joud Khoury, *RTX BBN Technologies;* Arvind Krishnamurthy, *University of Washington*

https://www.usenix.org/conference/nsdi25/presentation/zhao-liangyu

This paper is included in the
Proceedings of the 22nd USENIX Symposium on
Networked Systems Design and Implementation.

April 28–30, 2025 • Philadelphia, PA, USA

978-1-939133-46-5

Efficient Direct-Connect Topologies for Collective Communications[*]

Liangyu Zhao[†] Siddharth Pal[‡] Tapan Chugh[†] Weiyang Wang[§] Jason Fantl[‡]
Prithwish Basu[‡] Joud Khoury[‡] Arvind Krishnamurthy[†]

[†]*University of Washington* [‡]*RTX BBN Technologies* [§]*MIT CSAIL*

Abstract

We consider the problem of distilling efficient network topologies for collective communications. We provide an algorithmic framework for constructing direct-connect topologies optimized for the latency vs. bandwidth trade-off associated with the workload. Our approach synthesizes many different topologies and communication schedules for a given cluster size and degree, then identifies the best option for a given workload. Our algorithms start from small, optimal base topologies and associated schedules, using techniques that can be iteratively applied to derive much larger topologies and schedules. Additionally, we incorporate well-studied large-scale graph topologies into our algorithmic framework by producing efficient communication schedules for them using a novel polynomial-time algorithm. Our evaluation uses multiple testbeds and large-scale simulations to demonstrate significant performance benefits from our derived topologies and schedules.

1 Introduction

Collective communication operations involve concurrently aggregating and distributing data on a cluster of nodes and are used in both machine learning (ML) and high-performance computing (HPC). With the improved computational capabilities of accelerators, collective operations are a significant overhead in large-scale distributed ML training [20,44,64,76].

An emerging approach to address these challenges has been to employ various forms of optical circuit switching to achieve higher bandwidth at reasonable capital expenditure and energy costs [29,30,40,43,73,77,81]. Hosts communicate using a limited number of optical circuits that can be reconfigured at timescales appropriate for the hardware, thus exposing network topology as a configurable component. We refer to this setting as *direct-connect* with circuits configured and fixed for an appropriate duration.

Existing optical-circuit-based ML systems [30,43,73,81] fit this direct-connect model but do not exploit the flexibility topology reconfiguration offers. Collective operations such as allreduce are still limited to a few well-known algorithms that can fit the degree constraints of the optical fabric (e.g., rings, multi-rings, tori) and accept the consequent performance tradeoffs. For example, ring allreduce, while bandwidth-efficient, has a high graph diameter, causing high total-hop latency. A double binary tree, on the other hand, has a logarithmic diameter but suffers from load imbalances and bandwidth inefficiencies. Conversely, the broader spectrum of well-known collective algorithms that achieve desired latency and bandwidth (e.g., recursive-doubling, Bruck algorithm) [72, 79] use dynamic communication patterns ideal for switch networks but are ill-suited for degree-constrained direct-connect networks.

To fill this gap, we seek to identify new custom-built topologies and communication schedules for direct-connect networks. We pose the following question: *How to efficiently construct high-performance direct-connect topologies and communication schedules for collectives given the network's performance characteristics and degree constraints?*

This question poses several challenges. First, jointly optimizing *both* the network topology and the corresponding communication schedule is intractable at a large scale. Prior efforts reduce the search cost by optimizing only one or the other (e.g., schedules for a given topology [10, 65, 76] or topology permutations while retaining a ring schedule [77]). The combination of topological structure and communication schedule as degrees of freedom explodes the search space, making this a seemingly intractable problem. Second, the optimization must carefully consider the workload and the network's performance characteristics when distilling a topology and schedule. For example, minimizing the topology's diameter is ideal not only for latency-sensitive allreduce at small data sizes but also for all-to-all throughput; however, this could come at the cost of load imbalance across links in bandwidth-sensitive allreduce at large data sizes. Finally, lowering the synthesized schedules to the underlying hardware and runtimes [26,49] in an efficient way requires careful scheduling to achieve the desired performance in practice.

Our work addresses these issues by developing an algorithmic toolchain for quickly synthesizing efficient topologies and schedules for collective communications.

1. We devise a range of **expansion techniques** for synthesizing custom large-scale network topologies and schedules. The expansions start with small, optimal topologies and communication schedules and expand them to achieve near-optimal large-scale topologies and schedules.

2. We devise a **polynomial-time schedule generation algorithm** to produce optimal collective communication schedules for large-scale topologies with specific symmetry properties. This exposes many well-known topologies as options for the direct-connect network fabric.

[*] Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

3. We devise a **topology enumeration and search algorithm** to identify the best option for a target cluster and workload by exploring the *Pareto-efficient* options that provide different tradeoffs for bandwidth efficiency, total-hop latency, and also all-to-all throughput.
4. We develop **compilers** to realize our optimized schedules. We offer efficient implementations for both GPUs and CPUs, integrating with ML frameworks (e.g., PyTorch) through the MSCCL [49] and oneCCL [26] runtimes.

We evaluate our approach using two testbeds: a 12-node GPU cluster capable of topology reconfiguration and torus clusters on Frontera [69] supercomputer with up to 54 CPU nodes. Our techniques reduce collective communication times by $>30\%$ for DNN training on the GPU testbed and up to $3.1\times$ in supercomputing settings. Simulations for large-scale DNN training show up to an order of magnitude reduction in total communication time from topology and schedule optimization. Our schedule generation algorithm is orders of magnitude faster than the state-of-the-art (e.g., SCCL [10] and TACCL [65]), capable of producing schedules for topologies with thousands of nodes in a minute.

2 Background & Related Work

2.1 Network Fabric

Our work identifies topologies and schedules helpful for a broad range of settings, such as *switchless physical circuits*, *patch-panel optical circuits*, and *optical circuit switches*. While these options differ in cost and reconfigurability [77], they are all significantly cheaper than packet-switch solution [29, 40, 77] and can benefit from our work.

Switchless physical circuits require the least amount of fabric hardware. However, the topology must remain reasonably static for long periods, as the reconfiguration is manual. *Patch-panel optical circuits* provide a higher degree of reconfigurability by using a mechanical solution (e.g., robotic arms) to perform physical reconfigurations through a patch panel. The reconfigurations occur on the scale of minutes, but the patch panel itself can scale to a large number of duplex ports and is reasonably cheap (e.g., 1008 ports at $100 per port [70]). Both options can benefit from a carefully curated topology optimized for the workload, but they require it to remain static for a job given the reconfiguration time.

Commercial *optical circuit switches (OCS)* can perform reconfigurations in $\approx$10ms, are more expensive than patch panels, and scale to fewer ports (e.g., Polatis 3D-MEMS switch has 384 ports at $520 per port). Though OCSes support faster reconfigurations, the delays are still too high to support the rewiring of the circuits *during* a typical collective operation.[1] Thus, they cannot take advantage of algorithms designed for full-bisection switches, such as recursive halving/doubling [72, 79], that exploit high logical degree over time to provide both latency and bandwidth optimality. Thus, OCSes can also benefit from the custom-built and low-degree topologies synthesized by our approach.

All of these optical technologies allow for a shared cluster to be split into multiple subclusters for running separate jobs [29], so each job can be configured with its own topology. Further, unidirectional topologies are technically feasible on optical testbeds. Unidirectionality gives greater freedom in topology design and can enable lower-diameter networks.

Evaluation Target: In this paper, we use a reconfigurable optical patch panel to configure and evaluate different topologies. Given the high reconfiguration costs for the patch panel, we identify an efficient topology that will remain static for the duration of a job. Nevertheless, our techniques could be used to derive topologies for finer reconfiguration timescales if the hardware can efficiently support frequent reconfigurations.

2.2 Related Work

Several existing optical-circuit-based ML systems [30, 43, 73, 77, 81] fit the direct-connect model; however, they rely on existing implementations of collectives. Typically, communication libraries for ML training [20, 56, 64] offer either ring collective for high-latency bandwidth-optimal transfers or tree collective, which has logarithmic latency but suffers from load-imbalances across links. Other topologies such as mesh, tori, hypercubes, etc., have also been explored in HPC systems [3, 8, 11, 12, 18, 23, 58], but their bandwidth-latency tradeoff choices are limited as well. Bandwidth and latency optimal collectives for switch networks such as recursive-doubling, Bruck algorithm [72, 79], BlueConnect [13], etc., are unsuitable for direct-connect networks, because their one-to-one communication patterns fail to utilize all available links, and they assume a fully connected network.

Our work uniquely considers joint optimization of *both* the network topology and the corresponding collective communication schedule at a large scale, while prior work either optimizes one or the other. For instance, TopoOpt [77] generates customized shifted-ring topologies to optimize concurrent collective and non-collective communications for hybrid data-parallel [15, 37, 38] and model-parallel [28, 52, 67] DNN training, respectively. The collective communications in TopoOpt still use existing ring collectives. Consequently, when data parallelism, for example, dominates the workload, TopoOpt's performance suffers from similar latency issues present in existing ring collectives. Our effort is complementary as it synthesizes new topologies and schedules for collectives that span the entire cluster, but we do not optimize sub-cluster communications for hybrid parallelism. Extending our work to jointly optimize topologies and schedules for hybrid parallelism is future work.

Recent work like Blink [76], SCCL [10], and TACCL [65] also focus on generating a collective schedule for a given

[1] Research prototypes [1, 48] support μs to ns reconfigurations using overlay-hop relays and Valiant load balancing (VLB). While this is valuable for generic workloads, collectives have structured communication patterns, and it would be ideal to realize them without incurring the VLB overheads.

M	total data size	α	single-hop latency
N	number of nodes	B	total egress bandwidth of a node
S	data shard ($\|S\| = \frac{M}{N}$)	B/d	bandwidth of a single link
C	data chunk ($C \subseteq S$)	$T_L(A)$	total-hop latency of schedule
d	degree of topology	$T_B(A)$	bandwidth runtime of schedule
V_G	vertex/node set of G	$T_L^*(N,d)$	Moore optimality (Def 10)
E_G	edge/link set of G	$T_B^*(N)$	bandwidth optimality $\frac{M}{B} \cdot \frac{N-1}{N}$
$D(G)$	graph diameter of G	$N_x^+(u)$	nodes at distance x from u
Table 9 for graph symbols		$N_x^-(u)$	nodes at distance x to u

Table 1: Summary of Important Notations

topology. However, they all involve NP-hard optimizations that severely limit their scalability. SCCL is capable of generating optimal schedules, but it fails to generate a schedule in a reasonable time when the topology size is beyond 30 nodes. TACCL improves the scalability of SCCL by using communication sketches and also handles switch networks, but it sacrifices schedule performance and is still limited in scalability. In our approach, we either synthesize the schedule along with the topology or rely on a polynomial-time schedule generation technique that is provably optimal for networks with certain symmetry properties.

Generic large-scale topologies are typically not optimized for collective communications but for general datacenter traffic [6, 31, 33, 68, 75, 80]. Our framework can incorporate any degree-constrained regular topology (e.g., low-diameter expander graphs [25, 61]) and generate candidate schedules.

2.3 All-to-All Throughput

While we optimize allreduce, reduce-scatter, and allgather, the performance of all-to-all communication is also crucial for training DNN models like Mixture of Experts (MoE) [19, 34, 36, 60] and Deep Learning Recommendation Model (DLRM) [45, 53, 54]. Unlike other collectives, the scheduling of all-to-all can be easily formulated and efficiently solved as a multi-commodity flow (MCF) problem [4, 21, 32, 75, 78]. However, the graph diameter of the underlying topology is critical for all-to-all throughput [4, 41, 42, 48, 77]. The intuition is that if the nodes are far from each other, then all-to-all flows cost more *bandwidth tax* [47, 77] (i.e., the bandwidth of the flow multiplied by the length of the flow). With a fixed *network capacity* (i.e., the total number of links times link bandwidth), longer flows reduce the available bandwidth for each flow, thus decreasing all-to-all throughput. In this work, we construct topologies and associated schedules that are high-performance in both allreduce-type collectives and all-to-all by **deriving efficient allreduce-type schedules on existing or our synthesized low-diameter topologies**.

3 Formal Model of Collective Communications

We provide a formal model of ***reduce-scatter***, ***allgather***, and ***allreduce*** collectives. In each operation, there are N nodes operating on a vector of data of total size M. The data can be divided into N **shards**. In *reduce-scatter*, each node i reduces the i-th shard from all other nodes; in *allgather*, each node i broadcasts the i-th shard to all other nodes; in *allreduce*, each node i ends up with the fully reduced vector of data.

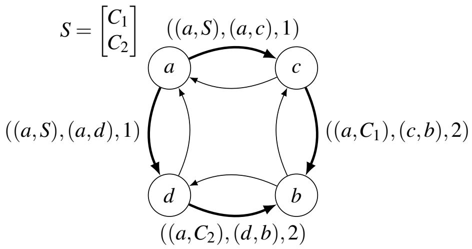


Figure 1: The allgather schedule of complete bipartite graph $K_{2,2}$. Shard S is divided into two half chunks C_1 and C_2. From a, at the 1st comm step, a sends the entire shard S to both c and d. At the 2nd comm step, c and d send the two half chunks C_1 and C_2, respectively, to b. Thus, every node receives the full shard from a. By applying similar broadcast from c,b,d in parallel, we have a complete BW-optimal *allgather* schedule with $T_L = 2\alpha, T_B = \frac{M}{B} \cdot \frac{3}{4}$.

Throughout the paper, we elaborate only on *allgather* schedule construction because the other two collectives are direct transformations. Since *allgather* and *reduce-scatter* are, respectively, simultaneous broadcasts and reductions for each node, we can **construct *reduce-scatter* schedules** in bidirectional topologies by simply reversing the communications in *allgather* schedules [11]. In unidirectional topologies, we utilize graph transposition to achieve a similar transformation (Appendix B).[2] **To construct an *allreduce* schedule,** we concatenate *reduce-scatter* and *allgather*.

3.1 Communication Topology & Schedule

The network topology is modeled as a directed graph (digraph) $G = (V,E)$, where V denotes the set of nodes ($|V| = N$) and E denotes the set of directed links/edges. The direct-connect network imposes a constraint that all nodes have degree d, which is the number of connection ports on each host and is typically low and independent of N.

A communication **algorithm** (G,A) uses the communication **schedule** A on topology G. Schedule A can be specified as what chunk C is communicated over which link in which **communication (comm) step** t. We define **chunk** C as a subset of shard S. Both C and S are specified as *index sets* of elements. Typically, S is interval $[0,1]$ representing the whole shard, and C is some subinterval. We denote v's chunk C as (v,C), which is a subset of v's starting shard (v,S). Let $((v,C),(u,w),t)$ denote that v's chunk C is sent by node u to its neighbor w at comm step t. Schedule A then is specified as a list of tuples $((v,C),(u,w),t)$. Figure 1 gives an example of an allgather schedule in such a tuple notation. Within a schedule, chunks can be different-sized subsets of S. Appendix B gives formal definitions of reduce-scatter/allgather schedules.

3.2 Cost Model

We use the well-known α-β cost model [24]. The cost of sending a message of size H over a link is $\alpha + \beta H$. This cost comprises two components: the constant single-hop latency α and a bandwidth component β, which is the inverse of link

[2] The main text of this paper focuses on high-level ideas of various techniques. We provide detailed mathematical analyses in the appendix.

bandwidth, i.e., $\beta = \frac{1}{b}$. This simple model has been shown to be appropriate for GPU interconnects [10, 35, 65]. In our analysis, we use node bandwidth B with $B = db$. In this paper, we focus on homogeneous networks, although some techniques also support heterogeneous ones (Appendix E.3).

The runtime of a schedule A can be broken down into a total-hop latency component and a bandwidth component. The **total-hop latency** component $T_L(A) = t_{\max}\alpha$, where $t_{\max}$ is the number of comm steps. $T_L(A)$ represents the cost of performing schedule A on an infinitesimal amount of data. The bandwidth component $T_B(A)$, or **bandwidth (BW) runtime**, is the sum of the BW runtime of each comm step, i.e., $\sum_t T_B(A_t)$. The BW runtime of comm step t is the max amount of data transmitted by a link within the comm step, divided by link bandwidth $b = B/d$. For N-node d-regular graphs, the optimal runtime of both *reduce-scatter* and *allgather* is approximately $\alpha \log_d N + \frac{1}{B} \cdot \frac{M(N-1)}{N}$. The 1st term represents the total-hop latency required for communicating across the diameter of a topology, while the 2nd term represents the transmission time for any node to send/recv $N-1$ shards in reduce-scatter/allgather. **One should not confuse total-hop latency with overall latency,** which is the sum of total-hop latency and BW runtime. We omit the computational time of reduction and discuss this in Appendix C.4.

We analyze the optimality of total-hop latency and BW runtime separately. An algorithm (G, A) is optimal in one component if no (G', A') with the same N, d can perform better. For BW runtime, an algorithm is **bandwidth (BW) optimal** iff its T_B equals $T_B^*(N) := \frac{M}{B} \cdot \frac{N-1}{N}$. For total-hop latency, given G, the lowest T_L achievable is $\alpha \cdot D(G)$, where $D(G)$ is the graph diameter of G. Thus, the optimal total-hop latency equals the smallest diameter of any N-node d-regular graph, which remains an open question in graph theory [50]. Therefore, we define *Moore optimality* based on *Moore bound*, which provides a lower bound for diameter given N, d and thus a well-defined $T_L^*(N, d)$. An algorithm is **Moore optimal** iff $T_L = T_L^*(N, d)$. Moore optimal topologies have the lowest diameters, which is also ideal for all-to-all throughput. Appendix C gives formal definitions of optimalities.

Ring allreduce has a total-hop latency that is linear in N, while the BW runtime is optimal. *Double binary trees* (DBT), on the other hand, offer the advantage of logarithmic total-hop latency but have suboptimal BW performance. Our work offers a range of topologies that are Pareto-efficient in total-hop latency and BW performance.

4 Overview of Our Approach

Direct-connect topologies can typically be categorized as either **low-hop** topologies, which have low diameters (e.g., expander graphs) suited for all-to-all throughput and small-data allgather/reduce-scatter/allreduce, or **load-balanced** topologies, which have simplistic structure (e.g., ring, torus) with easy load-balanced schedule for large-data allgather/reduce-scatter/allreduce (see Table 2). We seek to jointly identify network topologies and schedules that achieve high performance in both categories to the extent possible. Specifically, this entails the challenging task of constructing load-balanced allgather[3] schedules for low-hop topologies.

Topology Type	Small-Data Allreduce (Total-Hop Latency T_L)	Large-Data Allreduce (BW Perf T_B)	All-to-All Throughput
Low-Hop	✓	–	✓
Load-Balanced	–	✓	–

Table 2: The tradeoffs of low-hop topology vs. load-balanced topology. Reduce-scatter and allgather perform similarly to allreduce.

At a small scale, one could handpick a topology such as the complete bipartite graph $K_{2,2}$ defined at $N = 4, d = 2$. $K_{2,2}$ is both low-hop and load-balanced that a Moore- and BW-optimal allgather could be manually constructed (Figure 1). But how do we scale the topology and the schedule to larger sizes? Our work approaches this problem with two tools: *expansion techniques* (§5) and *BFB schedule generation* (§6).

Expansion Techniques: Given a base topology and its schedule, expansion techniques can expand them into a larger topology and associated schedule with minimal loss in performance. We call the resulting topologies **synthesized topologies**. The base topologies are small in scale, such as $K_{2,2}$ in Figure 1, for which straightforward schedules exist or an exhaustive search for the schedule is feasible. The line graph expansion, for example, can then expand $K_{2,2}$ and its schedule in Figure 1 to an allgather for $N = 4 \cdot 2^n$, for arbitrarily large n, while retaining a node degree of 2. Multiple expansion techniques can be composed to achieve the desired N and d.

Breadth-First-Broadcast (BFB) Schedule Generation: Besides synthesized topologies, we can use known topologies from graph theory (e.g., twisted torus, expander graphs). We call them **generative topologies** as they can be instantiated at various Ns and ds. Generative topologies are often low-hop, beneficial for total-hop latency and all-to-all throughput. The problem, though, is that efficient load-balanced collective schedules are not known for many of these topologies, and existing schedule generation methods [10, 65] are intractable even at moderate scales. Our work offers BFB, a polynomial-time schedule generation that can yield efficient schedules for large-scale topologies. For allgather, it performs a breadth-first broadcast from each node and uses linear programs to balance the workload on links. Although not always optimal, BFB schedules are provably optimal for many topologies exhibiting certain symmetries. For instance, BFB can generate a schedule with the lowest total-hop latency and BW optimality on any torus, including those with *unequal dimensions*.

With expansion techniques and BFB schedules, our *topology finder* (§5.4) assembles a large pool of topologies and schedules, identify *Pareto-efficient* ones from a low-hop vs. load-balanced perspective, and select from them for a given workload. When two options are Pareto-efficient, one must be better than the other in either low-hop (i.e., total-hop latency) or load-balanced (i.e., BW performance) but not in both. We

[3] We construct allreduce and reduce-scatter from allgather schedules.

choose low-hop options for workloads requiring all-to-all throughput and small-data allgather/reduce-scatter/allreduce, and load-balanced options for large-data allgather/reduce-scatter/allreduce. Finally, the *compiler* (§7) lowers the chosen topology and schedule to the runtime and hardware.

5 Expansion Techniques

We present three techniques that can be applied to construct near-optimal large-scale *synthesized topologies* and schedules by expanding small-scale topologies and associated schedules. The three techniques provide different options for increasing the size of the topology and the per-node degree while preserving either total-hop latency or BW optimality of the base graph and schedule (Table 3). While we describe the techniques in the context of allgather, corollary 1.2 in §B implies equivalent constructions for reduce-scatter and allreduce.

5.1 Line Graph Expansion

We borrow the line graph transformation from graph theory [22], which transforms an input graph G into a larger graph $L(G)$ as follows: every edge in G becomes a node in $L(G)$, and two nodes in $L(G)$ are adjacent if the corresponding edges are adjacent in G (Definition 12).

Intuition: Line graph expands an N-node degree-d topology into a dN-node topology. The degree d remains the same, which is crucial since the degree is often limited by hardware constraints like the number of available ports. While the number of nodes grows by d-fold, the diameter of the topology only increases by one, which is also optimal for total-hop latency and all-to-all performance. In addition, the paths in the base topology are mapped into the expanded topology, allowing the communication schedule for the base to be expanded as well. Line graph expansion can be applied repeatedly to scale the topology and schedule to arbitrarily large sizes.

Figure 2 gives an example of the line graph expansion of the complete bipartite graph $K_{2,2}$. Any (shortest) path $w_0 \rightarrow w_1 \rightarrow \ldots \rightarrow w_n$ in $K_{2,2}$ can be mapped to a (shortest) path $w_{-1}w_0 \rightarrow w_0w_1 \rightarrow \ldots \rightarrow w_{n-1}w_n \rightarrow w_nw_{n+1}$ in $L(K_{2,2})$ from $w_{-1}w_0$ to w_nw_{n+1}, for any w_{-1}, w_{n+1} provided that $w_{-1}w_0 \neq w_nw_{n+1}$.

Given an allgather schedule A_G for G, we construct schedule $A_{L(G)}$ for $L(G)$. Pick any node $v'v$ in $L(G)$. It needs to broadcast its shard to every other node in $L(G)$. Pick any other node, say, uu'. For each element x of $v'v$'s shard, we want to send x to uu'. Since v broadcasts x to every other node in A_G, there is a path $v \rightarrow w_1 \rightarrow \ldots \rightarrow w_{n-1} \rightarrow u$ in G along which x is sent to u in A_G. Thus, the path $v'v \rightarrow vw_1 \rightarrow w_1w_2 \rightarrow \ldots \rightarrow w_{n-1}u \rightarrow uu'$ can be utilized to send x from $v'v$ to uu' in $L(G)$.

Definition 1 (Schedule of Line Graph)**.** *Given an allgather schedule A_G for topology G, let $A_{L(G)}$ be the schedule for line graph $L(G)$ containing:*

1. $((v'v,S),(v'v,vu),1)$ *for each edge* $(v'v,vu) \in E_{L(G)}$ *with* $v'v \neq vu$. **[Insert the 1st comm step in $A_{L(G)}$.]**
2. $((v'v,C),(uw,ww'),t+1)$ *for each* $((v,C),(u,w),t) \in A_G$ *and* $v'v \neq ww'$. **[Adapt A_G to form $A_{L(G)}$.]**

At the 1st comm step, x is broadcasted by $v'v$ to every neighbor, including vw_1. Then, for every $((v,C),(w_i,w_{i+1}),t)$ in A_G with $x \in C$, there is $((v'v,C),(w_iw_{i+1},w_{i+1}w_{i+2}),t+1)$ in $A_{L(G)}$ that takes x from w_iw_{i+1} to $w_{i+1}w_{i+2}$ and, eventually, to uu' ($v = w_0, u = w_n$). Since x and uu' are picked arbitrarily, $v'v$ broadcasts every element of its shard to all nodes in $L(G)$. Figure 2c shows an example of schedule construction.

As for the performance of $A_{L(G)}$, we leave the mathematical details in Appendix D.1. In practice, one can apply line graph expansion repeatedly to scale the topology and schedule indefinitely. The more optimal the base topology and schedule are, the more optimal the expanded topology and schedule will be. Figure 3 shows how the performance evolves as we continuously apply line graph expansion to several Moore and BW optimal base graphs. The total-hop latency always remains Moore optimal. The BW performance deviates from optimality T_B^* but remains a constant factor away asymptotically. A key observation in Figure 3 is that the larger the size of the base graph is, the closer the expanded schedule is to BW optimality. Line graph expansion is notable for its ability to construct indefinitely large-scale topologies without increasing degree d. The expansion also maintains low-hop, making it ideal for synthesizing all-to-all topologies as well.

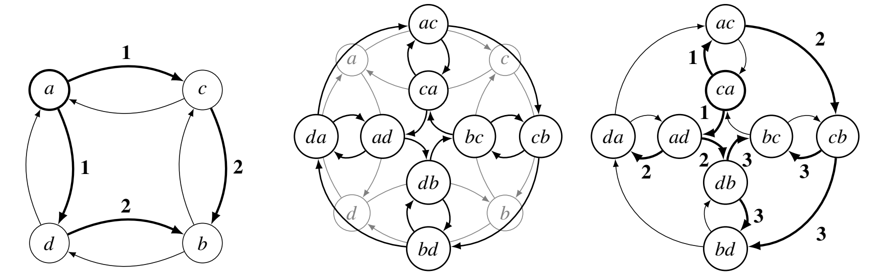
5.2 Degree Expansion

Intuition: While line graph expansion expands the number of nodes, degree expansion additionally expands the topology degree. Taking a base topology G, we make n copies of it and connect two nodes in different copies if they are adjacent in G. This process forms an expanded topology $G*n$, which multiplies both the number of nodes and the degree of G by n. Because the connections in $G*n$ are derived from G, similar to line graph expansion, we can map paths from G to $G*n$ to expand the communication schedule of G as well.

Figure 4 gives an example of expanding a 4-node unidirectional ring into an 8-node degree-2 topology (see formal definition of degree expanded topology in Definition 13). Based on the input schedule A_G for G, we construct a schedule A_{G*n} for $G*n$. For any data traveling along $v \rightarrow w^{(1)} \rightarrow \ldots \rightarrow w^{(m)} \rightarrow u$ in A_G, A_{G*n} has the data travel along $v_i \rightarrow w_i^{(1)} \rightarrow \ldots \rightarrow w_i^{(m)} \rightarrow u_j$ for all i, j. That is, data is transmitted within the i-th copy of G, except at the last step. With this construction, any node u_i has broadcasted the data to all other nodes except its own copies u_js. We add an additional comm step for u_j to collect the data from its in-neighbors (see Figure 4c).

Definition 2 (Degree Expanded Schedule)**.** *Given an allgather schedule A_G for G, construct A_{G*n} for $G*n$:*

1. *For all i,j including $i=j$ and for each* $((v,C),(u,w),t) \in A_G$, *add* $((v_j,C),(u_j,w_i),t)$ *to* A_{G*n}*;*
2. *Divide shard S into equal-sized chunks $C_1,\ldots,C_{nd}$. Given $u_i,u_j \in V_{G*n}$ with $i \neq j$, add $((u_i,C_\alpha),(v_\alpha,u_j),t_{\max}+1)$ to A_{G*n} for each $(v_1,u_j),\ldots,(v_{nd},u_j) \in E_{G*n}$, where $t_{\max}$ is the max comm step in A_G.*



(a) $K_{2,2}$ ($N\!=\!4, d\!=\!2$) (b) $L(K_{2,2})$ ($N\!=\!8, d\!=\!2$) (c) Broadcast from ca

Figure 2: The complete bipartite topology $K_{2,2}$ with its line graph $L(K_{2,2})$. Figure (a) shows the base topology and broadcast paths from a to c,b,d in $A_{K_{2,2}}$ (see Figure 1). The number next to edge shows the comm step using the edge. Figure (b) shows the expanded topology. Observe that every edge in $K_{2,2}$ becomes a vertex in $L(K_{2,2})$, and two vertices are connected if the corresponding edges in $K_{2,2}$ have one's head node being the other's tail node. Figure (c) shows the broadcast paths of node ca, transformed from the broadcast paths of a in figure (a). At the 1st comm step, by step 1 of Def 1, ca broadcasts its shard to all its neighbors: $((ca,S),(ca,ac),1)$, $((ca,S),(ca,ad),1)$. The rest of the broadcast paths are transformed from $A_{K_{2,2}}$ by step 2 of Def 1, e.g. $((a,C_1),(c,b),2) \mapsto \{((ca,C_1),(cb,bc),3),((ca,C_1),(cb,bd),3)\}$. Each of the nodes bc and bd receives C_1,C_2 from its two in-neighbors, just like b does in $A_{K_{2,2}}$.

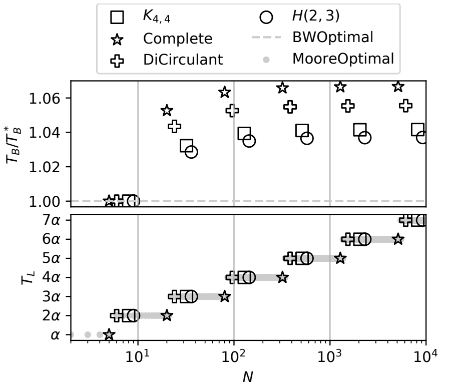
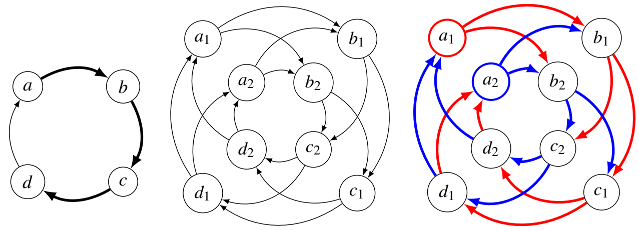


Figure 3: Line graph expansion on Moore and BW optimal degree-4 base graphs: complete bipartite graph $K_{4,4}$, complete graph, directed circulant graph, and Hamming graph $H(2,3)$. $T_B^* = \frac{M}{B} \cdot \frac{N-1}{N}$ is the optimal BW runtime.



(a) $G(N\!=\!4, d\!=\!1)$ (b) $G*2$ ($N\!=\!8, d\!=\!2$) (c) Broadcasts from a_1,a_2

Figure 4: 4-node unidirectional ring and its degree expansion to $d\!=\!2$. Figure (a) shows the base topology and broadcast path from a to b,c,d. Figure (b) shows the expanded topology. Figure (c) shows the broadcast paths from a_1 and a_2 to other nodes marked in red and blue, respectively. For any $u \neq a$, the path from a_i to u_j stays in i until the very last step when it jumps to u_j, e.g., $a_1 \to b_1 \to c_2$ and $a_2 \to b_2 \to c_2 \to d_1$. For a_i to a_j, each in-neighbor of a_j sends an equal portion of a_i's shard to a_j in the end; for example, d_1 and d_2 each send half of a_1's shard to a_2 and half of a_2's shard to a_1. The red and blue broadcast paths are disjoint, resulting in BW optimality.

Unlike line graph expansion, degree expansion preserves BW optimality. This is because the expanded broadcast paths from copies of an original node are totally disjoint from each other (Figure 4c). However, degree expansion does not preserve Moore optimality. While line graph expansion does not change degree, degree expansion increases it, reducing the number of comm steps required for Moore optimality.

5.3 Cartesian Product Expansion

The Cartesian product of two graphs G_1, G_2 is an expanded graph $G_1 \square G_2$ with size and degree equal to the product of G_1, G_2's sizes and the sum of their degrees, respectively.

Definition 3 (Cartesian Product)**.** *The Cartesian product digraph $G_1 \square G_2$ of digraphs G_1 and G_2 has vertex set $V_{G_1} \times V_{G_2}$ with vertex $\mathbf{u} = (u_1,u_2)$ connected to $\mathbf{v} = (v_1,v_2)$ iff either $(u_1,v_1) \in E_{G_1}$ and $u_2 = v_2$; or $u_1 = v_1$ and $(u_2,v_2) \in E_{G_2}$.*

This definition generalizes to the Cartesian product of n digraphs: $G_1 \square G_2 \square \ldots \square G_n$. When $G_1 = \ldots = G_n = G$, the product is denoted as Cartesian power $G^{\square n}$. We use Cartesian power and product in our topology and schedule expansion.

Intuition: The Cartesian product $G_1 \square G_2 \square \ldots \square G_n$ consists of n dimensions, with connections in dimension i identical to G_i. Taking the schedules of $G_1, G_2, \ldots, G_n$, we can balance the amount of traffic going through each dimension to achieve high BW performance. Cartesian product expansion greatly expands the set of topologies we construct by enabling the combination of existing topologies to form a new product topology with an efficient schedule.

Cartesian Power Expansion: Given a d-regular G and schedule A_G, we can construct a schedule $A_{G^{\square n}}$ for $G^{\square n}$, which is nd-regular and has $|V_G|^n$ nodes. This technique helps generate efficient topologies, including some well-known ones like hypercube and Hamming graph. We describe how to construct allgather schedules for Cartesian power graphs by using $\ell \times \ell$ torus (ℓ-ring$^{\square 2}$) as an example. A typical allgather schedule on an $\ell \times \ell$ torus is to perform the ℓ-ring allgather along rings in one dimension first and then the other dimension, as in hierarchical ring allreduce [74]. Consider two schedules: $A^{(1)}$ performs allgather on vertical rings first and then horizontal ones; $A^{(2)}$ performs allgather in the opposite order. $A^{(1)}, A^{(2)}$ use disjoint set of links at any comm step. Thus, we divide each data shard in $\ell \times \ell$ torus into two halves and let them be allgathered by $A^{(1)}, A^{(2)}$ separately. The combined schedule, where $A^{(1)}$ and $A^{(2)}$ are performed in parallel, is BW-optimal, with a total-hop latency of $2T_L(A)$.

The above torus schedule has appeared in previous literature [62]. It can be generalized to generate schedules for Cartesian power of arbitrary topologies (see Appendix D.3).

Cartesian Product Expansion: One can also construct a schedule for the Cartesian product of distinct topologies. For example, an $a \times b \times c$ 3D torus is the Cartesian product of three rings with lengths a,b,c. Constructing this schedule requires BFB schedule generation technique, which we introduce in §6. If individual topologies have BW-optimal BFB schedules, as in the case of any torus, then the schedule generated for the Cartesian product is BW-optimal (Table 3).

Expansion Techniques	# of Nodes	Deg	Moore	BW	Perf
Line Graph Exp $L^n(G)$	d^nN	d	✓	×	Thm 7.1
Degree Exp $G*n$	nN	nd	×	✓	Thm 11
Cartesian Power $G^{\square n}$	N^n	nd	×	✓	Thm 12
Cartesian Prod $G_1\square\ldots\square G_n$	$\prod_i N_i$	$\sum_i d_i$	×	✓	Thm 13

Table 3: Summary of expansion techniques. The table shows the characteristics of the resulting topology and schedule after applying expansion techniques to an N-node degree-d base topology. "✓, ×" show whether the expansion preserves Moore/BW optimality. The last column refers to the theorems that give the exact performance of expanded schedules. Appendix D presents more formal definitions and analyses of expansion techniques.

Topology	T_L	T_B	$2(T_L+T_B)$	$D(G)$	All-to-All
$\Pi_{4,1024}$	5α	$1.332M/B$	323.5us	5	409.1us
$L^3(C(16,\{3,4\}))$	6α	$1.020M/B$	291.0us	6	403.5us
$L^2(\text{Diamond}^{\square 2})$	8α	$1.004M/B$	328.4us	8	446.6us
$L(\text{DBJMod}(2,4)^{\square 2})$	11α	$1.000M/B$	387.8us	9	529.9us
$(\text{UniRing}(1,4)\square\text{UniRing}(1,8))^{\square 2}$	20α	$0.999M/B$	567.6us	20	1174.4us
Theoretical Bound	$\mathbf{5\alpha}$	$\mathbf{0.999M/B}$	**267.6us**	**5**	**382.3us**

Table 4: Pareto-efficient topologies at $N=1024$, $d=4$. The $2(T_L+T_B)$ column shows the allreduce runtimes for $\alpha=10$us and $M/B=1\text{MB}/100\text{Gbps}$. We multiply T_L+T_B by 2 because allreduce is performed by combining reduce-scatter and allgather. The all-to-all time is computed via multi-commodity flow (Appendix A.5) with each node having 1MB of data to send (i.e., sending $1/N$ MB to each node). For comparison, the baselines Shifted Ring and Double Binary Tree (§8.2) have allreduce times of 20640us and 1434us, and all-to-all times of 10738us and 21475us, respectively. Table 9 shows the details of the base topologies.

5.4 Topology Finder

The goal of Topology Finder is to produce the best topologies and schedules for a target N and d. If we aim for asymptotic performance ($N \to \infty$ with fixed d), we want **the base topology to be as large as possible and the base schedule to be as optimal as possible** (Theorem 9). However, for a target N and d, only base topologies with certain sizes (e.g., divisors of N) and degrees can be expanded to the target. Thus, we keep a collection of known base topologies and their schedules (Table 9). These topologies and schedules are highly optimized and cover a wide range of N and d.

Given base topologies, we perform a bottom-up search for the combinations of expansion techniques to reach the target N and d. We iteratively apply expansions to candidates. At intermediate sizes, we prune candidates with inferior performance and keep the best ones for further expansion. Because each expansion multiplies the topology size (Table 3), the number of expansions that can be applied before the size gets too large—and hence the number of possible combinations—is limited, making the search feasible.

While we expand the topologies, proved theorems (Table 3) allow us to predict the performance of expanded topologies. This is vital for the search because it is intractable to construct schedules for every topology and compare their performance. Using a simple formula for prediction enables us to quickly compare different topologies and prune inferior ones. We keep a Pareto frontier of topologies for each given N and d. A topology is inferior to another only if it is worse in both total-hop latency and BW runtime. Ultimately, the search finds all Pareto-efficient topologies for the target N and d. Depending on the testbed, we may convert unidirectional topologies to bidirectional ones (Appendix A.6). Then, we determine the best-performing topology for a given workload.

Table 4 shows the result for $N=1024$ and $d=4$. From top to bottom, the Pareto frontier exhibits an increasing T_L and a decreasing T_B, with the top and bottom being Moore and BW optimal, respectively. On the all-to-all side, the diameters of the topologies also follow the same trend as T_L because of $T_L \geq \alpha \cdot D(G)$ (Theorem 3). Table 4 also shows the allreduce and all-to-all times calculated based on specific α, M, B. Notably, the line graph of circulant graph $L^3(C(16,\{3,4\}))$ has both the lowest allreduce and all-to-all times, within 9% and 6% of the theoretical bounds. Table 7 in appendix contains more results for $N=32,64,\ldots,1024$.

While low-hop/diameter indicates high all-to-all throughput, other metrics like the average distance between nodes [41, 42] also play a role. Thus, despite having a lower diameter, $\Pi_{4,1024}$ underperforms $L^3(C(16,\{3,4\}))$. Including other metrics makes the search process more complex and computationally expensive. In practice, T_L and $D(G)$ are feasible and accurate enough for predicting all-to-all throughput.

In DNN training experiments, we use one topology for the entire training due to the high reconfiguration latency of our target patch panel platform. We select the best option based on the distribution of collective sizes Ms, which depends on the communication strategy of the training framework [38]. With faster reconfiguration, one could change topology to optimize for different collective runs during training.

Our implementation runs under a minute for all $d = 2,4,8,16$ and N up to 2000. While this can be sped up, we find it acceptable for now, given that the search is performed once for all Ns and ds, and results can be saved for future use.

6 Breadth-First-Broadcast (BFB) Schedule

We now present a scalable algorithm for generating schedules for *generative topologies*, which are directly borrowed from graph theory, as well as for topologies obtained through Cartesian Product expansion—the only expansion technique that does not yield a schedule. State-of-the-art schedule generations (e.g., Blink [76], SCCL [10], and TACCL [65]) can scale only to a modest number of nodes because they involve NP-hard optimization. To ensure polynomial-time generation, we impose a *breadth-first* broadcast order from each node such that (1) data always travels along the shortest paths between source and destination nodes; (2) the schedule is structured as a series of comm steps, where each comm step is responsible for eagerly transmitting data to a set of nodes that is one additional hop away. Our BFB schedule generation technique does not guarantee optimality in an arbitrary topology, given these constraints prohibit the use of longer paths or delayed (non-eager) transmissions along paths, but these constraints enable polynomial-time generation.

Despite these constraints, BFB schedules guarantee the following: (1) The schedules have the lowest total-hop latency as all data is eagerly communicated over the shortest paths.

(2) For Cartesian product topologies, BFB schedule generation yields a BW-optimal solution if the underlying product components admit BW-optimal BFB schedules (thus yielding optimal schedules for networks such as torus with arbitrary dimensions). (3) BFB schedules are also provably BW-optimal for many generative topologies with certain symmetries.

6.1 BFB Schedule Generation Linear Program

Intuition: Allgather is a simultaneous broadcast from every node in the topology. A BFB allgather schedule, as the name suggests, performs a *breadth-first* broadcast from every node. At each comm step, a node typically has the option to receive a chunk from multiple in-neighbors on the previous breadth-first frontier. To optimize BW performance, BFB uses a linear program to balance the traffic across the ingress links.

At each comm step t, a BFB schedule requires that for every node v, all nodes at a distance t from v, i.e., $N_t^+(v)$, receive v's data shard within the comm step. To achieve this, all nodes at distance $t-1$, i.e. $N_{t-1}^+(v)$, need to collectively multicast the data shard to nodes $N_t^+(v)$ in comm step t.

Given any v and $u \in N_t^+(v)$, u may have multiple in-neighbor ws in $N_{t-1}^+(v)$. All of them can provide v's data shard because they have received it in comm step $t-1$. Since the BW runtime of a comm step equals the transmission time of the most congested link, a question is **how to allocate the amount of data u receives from each w to balance the workload on links?** Figure 5 shows an example. Here, u_1 needs to get v_1's shard from w_1, w_2 and v_2's shard from w_2. The solution is simple: since u_1 can only get v_2's shard from w_2, we let w_1 send v_1's shard and w_2 send v_2's shard, achieving a perfectly balanced workload. For u_2, it is more complicated. We formulate such a problem as a linear program:

$$\begin{array}{lll} \text{minimize} & U_{u,t} & \\ \text{subject to} & \sum\limits_{v} x_{v,(w,u),t} \le U_{u,t}, & \forall w \in N^-(u) = N_1^-(u) \\ & \sum\limits_{w} x_{v,(w,u),t} = 1, & \forall v \in N_t^-(u) \\ & 0 \le x_{v,(w,u),t} \le 1. & \forall w, v \end{array} \tag{1}$$

$x_{v,(w,u),t}$ is the proportion of v's shard that is sent from w to u and is defined for every v, w such that $w \in N^-(u)$ and $d(v,u) = d(v,w)+1 = t$. $U_{u,t}$ is the max workload among links to u, i.e., $(w_1,u_2), (w_2,u_2), (w_3,u_2)$ in the case of u_2. Minimizing $U_{u,t}$ is equivalent to minimizing $\frac{M/N}{B/d} \cdot U_{u,t}$, the max transmission time among links to u at comm step t. The 1st and 2nd constraints ensure correct max workload and u receiving all data shards, respectively. Appendix E gives the specific LP for u_2, and the solution is shown in blue in Figure 5. The workload is also balanced with each link sending $2/3$ shard and hence BW runtime $\frac{M/N}{B/d} \cdot \frac{2}{3}$.

SCCL [10] and TACCL [65] use NP-hard optimizations with discrete variables used to ensure each chunk is received before being sent. In contrast, we do not need discrete variables. A key observation from Figure 5 is that because w_1, w_2, w_3 all receive the entire shard of v_1 at comm step $t-1$, the $x_{v_1,(w_1,u_2),t} = 2/3$ and $x_{v_1,(w_2,u_2),t} = 1/3$ in the solution can be any portions of the data shard, as long as their union is the entire shard. Assuming $[0,1]$ is the entire shard of v_1, no matter the $2/3$ sent by w_1 to u_2 is $[0, \frac{2}{3}]$ or $[\frac{1}{3}, 1]$, the $1/3$ sent by w_2 can simply be $[\frac{2}{3}, 1]$ or $[0, \frac{1}{3}]$ accordingly. Thus, **we only need to decide the amount of data sent on each link, which are continuous variables**, enabling polynomial-time schedule generation. To obtain a complete schedule, one needs to solve an LP (1) for each $u \in V_G$ and $t \in \{1, \ldots, D(G)\}$. The BW runtime of the generated schedule is

$$T_B = \frac{M/N}{B/d} \sum_{t=1}^{D(G)} \max_{u \in V_G} U_{u,t}. \tag{2}$$

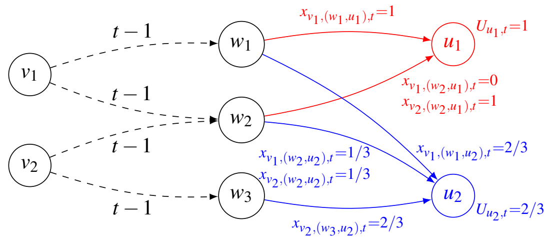


Figure 5: Example of BFB allgather schedule at comm step t. Here, $w_1, w_2 \in N_{t-1}^+(v_1)$ and $w_2, w_3 \in N_{t-1}^+(v_2)$. u_1, u_2 are at distance t from both v_1, v_2, so they both need to receive the data shards of v_1, v_2 in comm step t. Note that u_1 cannot get v_2's shard from w_3 because w_3 is not an in-neighbor of u_1. The figure also shows the solutions to LPs (1). The red and blue are independent LPs optimizing $U_{u_1,t}$ and $U_{u_2,t}$ respectively.

One could create an LP incorporating all $U_{u,t}$s and minimize T_B (2) *"globally"*. However, the result is equivalent to individually solving small LPs (1) for each u and t. This is because the LPs are independent of each other, e.g., the decisions made to minimize $U_{u_2,t}$ in Figure 5 do not affect $U_{u_1,t}$, and vice versa. The advantage of small LPs is that they can be solved in parallel. Due to the breadth-first nature of BFB, data always follows the shortest paths between source and destination. Thus, the number of comm steps of the BFB schedule always equals the graph diameter, i.e., $T_L = \alpha \cdot D(G)$, the lowest possible T_L given G.

Appendix E analyzes the BFB schedule and includes modifications to generate **discrete chunked schedules** (§E.2) and schedules for **heterogeneous link bandwidths** (§E.3). Corollary 1.1 implies how to generate **reduce-scatter** schedules.

6.2 Generative Topologies

Generative topologies, unlike synthesized ones, are large graphs directly borrowed from graph theory. They are too large for manual or NP-hard schedule generation. Thus, we use the BFB linear program to generate schedules. Since a BFB schedule always has the lowest T_L for a topology, if it is also BW-optimal, then it is *the optimal schedule* for that topology. Generative topologies often have symmetries that allow us to prove optimality mathematically. Their low diameters are also ideal for all-to-all throughput.

Torus is a widely used topology in parallel computing systems. Our torus schedule generated by BFB is theoretically optimal and represents a significant improvement over traditional torus schedules [62]. Given a $d_1 \times d_2 \times \ldots \times d_n$ torus, the traditional schedule, which performs parallel ring collectives on dimensions, only works (or is efficient) when dimensions are equal, i.e., $d_1 = d_2 = \ldots = d_n$, and has $T_L = \sum_i (d_i - 1)\alpha$. BFB torus schedule, however, is BW-optimal with any d_is and $T_L = \sum_i \lfloor d_i/2 \rfloor \alpha$. The BW optimality is due to torus being the *Cartesian product* of rings, each of which has a BW-optimal BFB schedule. BFB torus opens up many more construction possibilities since d_is can be any combination.

Generalized Kautz Graph (§F.2) and **Circulant Graph** (§F.4) are a pair of versatile graphs in our toolbox. The former can be constructed for any N and d, while the latter can be constructed for any N and even-value d. Furthermore, the BFB schedule of the former is at most one α away from Moore optimality, making it the topology with the lowest T_L, while the latter always has a BW-optimal BFB schedule. Thus, they can fill gaps in N and d that expansion techniques fail to cover (e.g., prime N) or provide good candidates.

Besides the aforementioned ones, the following graphs also have optimal schedules by BFB. **Distance-Regular Graph** (§F.3) is a family of large highly-symmetric graphs that are both BW-optimal and low-hop at the same time. The **Twisted Torus** [14] used by TPU v4 [29] is also computationally verified to be BW-optimal for at least $N \leq 10^4$. A **BFB Ring Schedule** with half the T_L of traditional one is shown in §F.1.

7 Schedule Compilation

We implemented two compilers for lowering communication schedules to both GPU and CPU clusters, given the significance of collective communication for both ML and HPC workloads. We lowered over 1K schedules for various topologies and configurations. The compilers enable us to evaluate the performance of our topologies and schedules on hardware and to validate our mathematical model.

For GPUs, our compiler lowers a mathematically defined schedule to an XML file that can be executed by the MSCCL runtime [49]. MSCCL is an open-source collective communication library that extends NCCL [56] with an interpreter providing the ability to program custom schedules. Communication schedules are defined in XML as instructions (send/receive/reduce/copy) for each GPU threadblock. Our compiler also performs certain optimizations, such as consolidating non-contiguous sends using a scratch buffer and evenly distributing the computational workload across threadblocks.

For CPU-based supercomputers, we use Intel oneCCL [26] + libfabric [39] to execute schedules. We extended oneCCL with an interpreter that executes XMLs. The mathematical schedules are lowered into instructions (send/recv/reduce/-copy/sync) for CPUs in an XML file and then executed.

8 Evaluation

We present performance evaluation results on a 12-node direct-connect optical GPU testbed and a supercomputing CPU torus cluster with up to 54 nodes. We also present analytical and simulation results at larger scales.

Collective Communication: On the 12-node testbed, our topologies consistently outperform baselines in allreduce, reduce-scatter, and allgather (§8.3, Fig 6, Fig 12). Analytical model shows order-of-magnitude improvements in allreduce and all-to-all performance at larger scales (Fig 7).

DNN Training: In data-parallel training experiments, our topologies achieve the best performance for both small models and GPT-2 [59] (§8.4, Fig 8). In simulated large-scale training, our topologies demonstrate order-of-magnitude improvements in all-to-all involved expert-parallel training (Fig 9).

Schedule Generation: While generating optimal schedules, BFB is orders of magnitude faster and more scalable than SCCL [10] and TACCL [65] (§8.5.1, Tab 6 & Fig 10). On supercomputing torus clusters, BFB schedules outperform traditional scheduling [62], SCCL, and TACCL (§8.5.2, Fig 11).

Finally, we also conducted experiments on our testbed to compare BFB against widely adopted solutions for switch networks (e.g., NCCL and recursive halving & doubling) (§A.1) and to validate the α-β cost model (§A.2).

8.1 Direct-Connect Optical Testbed

Our testbed consists of 12 servers, each with an NVIDIA A100-PCIe-40GB GPU [55] and an HPE Ethernet Adapter, configured as 4x25Gbps breakout interfaces. The NICs are directly connected via a Telescent optical patch panel [70]. Our testbed can realize topologies by reconfiguring the patch panel. We limit our evaluation to bidirectional topologies due to limitations in our testbed (discussed in Appendix A.6).

8.2 Experiment Setup

Baselines: We evaluate against the two baselines at $d = 4$: (1) *ShiftedRing*, which improves upon NCCL ring [56], is used by TopoOpt [77] for data-parallel training. The topology is a superposition of two bidirectional rings, each allreducing half of the data. (2) *Double Binary Tree* (DBT), implemented in NCCL [27], uses the topology and schedule from [63].

Methodology: We use the MSCCL runtime [49] to evaluate the topologies and schedules. We sweep through runtime parameters, such as the protocol (Simple or LL), number of channels (1, 2, 4, or 8), degrees of pipelining for the DBT baseline, etc., and choose the best-performing schedule for each data size. For DNN training, we run our schedules in PyTorch through MSCCL as the backend.

8.3 Collective Communication Evaluation

Figure 6 shows allreduce results for varying topology sizes N and data sizes M. Table 5 shows the topologies generated by our topology finder (§5.4). We also add *ShiftedBFBRing*, which is ShiftedRing topology but with our BFB generated schedule. We observe that in the small data regime ($M =$ 1KB), our topology beats ShiftedRing by a significant margin ($\sim$75% at $N = 12$) and also outperforms DBT ($\sim$20% at $N = 8, 10, 12$). Our ShiftedBFBRing beats ShiftedRing ($\sim$40%

N	**Topology**	T_L
5	Complete Graph: K_5	2α
6	Degree Expansion of Complete graph: $K_3 * 2$	4α
7	Circulant Graph: $C(7,\{2,3\})$	4α
8	Complete Bipartite Graph: $K_{4,4}$	4α
9	Hamming Graph: $H(2,3)$	4α
10	Degree Expansion of BFB augmented Bidirectional Ring: $\mathrm{BiRing}(2,5) * 2$	4α
11	Circulant Graph: $C(11,\{2,3\})$	4α
12	Circulant Graph: $C(12,\{2,3\})$	4α

Table 5: OurBestTopo at $d=4$ generated by topology finder (§5.4). All topologies listed above are BW-optimal.

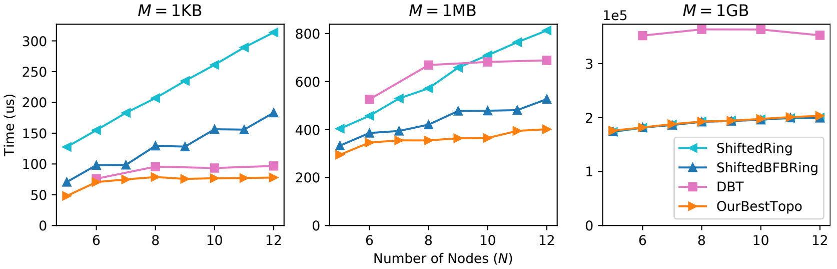


Figure 6: Allreduce experiment results on testbed at $M = 1$KB, 1MB, 1GB. "OurBestTopo" topologies are listed in Table 5. Reduce-scatter and allgather results are in Appendix Figure 12.

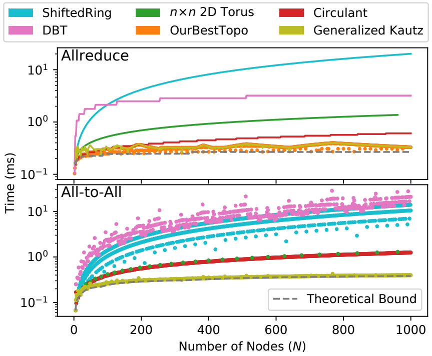
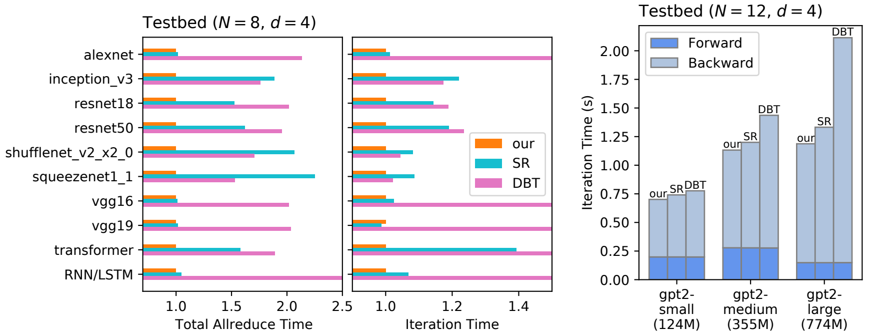


Figure 7: Comparing theoretical allreduce and all-to-all runtimes analytically at large N for $d=4$, $\alpha = 10$us, and $M/B = 1$MB/100Gbps. The all-to-all times are computed via multi-commodity flow (Appendix A.5) with each node having 1MB of data to send (i.e., sending $1/N$ MB to each node).



(a) Small Model testbed training results (normalized by our $K_{4,4}$)

(b) GPT2 testbed training results

Figure 8: Testbed data-parallel training results with different topologies. We compare our topologies (Table 5) against ShiftedRings (SR) and double binary trees (DBT) at 8- and 12-node scale. (a) shows results of training small models on 8 A100 GPUs of our testbed, all using batch size 64. The total allreduce time is the sum of the allreduce times for all layers in the model. (b) shows results of training GPT-2 with 12 A100 GPUs. The per-GPU batch sizes are selected to max out the 40GB GPU memory, with the small, medium, and large models having per-GPU batch sizes of 8, 4, and 1, respectively.

at $N = 12$) despite using the same topology. At small data sizes, the runtime is dominated by total-hop latency T_L, and hence, we can significantly outperform ShiftedRing, which has linear instead of logarithmic T_L growth with respect to N.

In the large data regime ($M = 1$GB), our topology beats DBT by a significant margin ($\sim$50% lower at $N = 8, 10, 12$) and matches the performance of ShiftedRing. At large data sizes, the runtime is dominated by BW runtime T_B. Since the ShiftedRing is BW-optimal, we can only match its performance. Due to the influence of both total-hop latency and BW runtime at intermediate data sizes ($M = 1$MB), our topology outperforms ShiftedRing ($\sim$50% at $N = 12$) and DBT ($\sim$45% at $N = 8, 10, 12$) in this regime. Our ShiftedBFBRing also outperforms ShiftedRing ($\sim$35% at $N = 12$). Note that although our gains over ShiftedRing diminish as M grows, future increases in hardware bandwidth will enhance gains at large M due to T_L playing a more significant role. Appendix Figure 12 shows the reduce-scatter and allgather results, which demonstrate trends and conclusions similar to those in Figure 6.

Figure 7 shows the allreduce and all-to-all runtime comparison for large N based on our analytical model. Topologies generated by our topology finder perform orders of magnitude faster in both allreduce and all-to-all. In allreduce, our best topologies outperform ShiftedRing and DBT by 56× and 10×, respectively, near $N = 1000$, due to the former's linear growth in T_L and the latter's poor BW performance. When compared against 2D torus, our topologies also achieve 4× better allreduce performance near $N = 1000$ (see §A.3 for a detailed analysis of our topologies at large N for different $\alpha, M/B$). As for all-to-all, we compare baseline topologies against our lowest-diameter topology, generalized Kautz, and our highest-diameter topology, circulant, from our Pareto-frontier for any N and d. These two represent our best and worst all-to-all topologies, respectively, while also serving as the worst and best BW-efficient allreduce topologies. Nevertheless, circulant still outperforms all baselines in all-to-all: 9× and 14× better than ShiftdRing and DBT, respectively, on average from $N = 900$ to 1000. It is barely matched by the 2D torus, which is limited to the square number Ns. Our lowest-diameter topology, generalized Kautz, outperforms ShiftedRing and DBT by 28× and 42×, respectively, and is within 5.2% of the theoretical bound from $N = 900$ to 1000.

8.4 DNN Training Evaluation

We compare our topologies against ShiftedRings and double binary trees in DNN training. On our small-scale testbed, we demonstrate improvements in data-parallel training across various small models and also GPT-2 [59]. For full-scale LLM training involving up to 1024 nodes and all-to-all com-

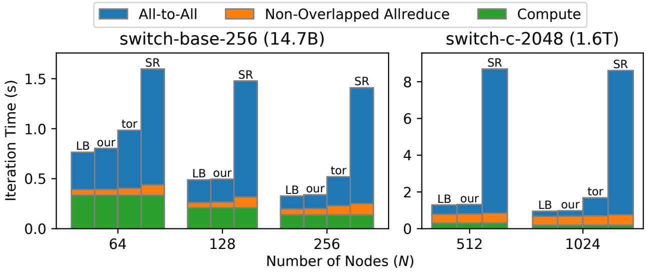


Figure 9: Simulated expert-parallel training of Switch Transformers across various topologies of different sizes. The simulation is conducted assuming $\alpha = 10$us, $B = 100$Gbps. All-to-all time is computed via multi-commodity flow (Appendix A.5). We detailed our setup in Appendix A.4.

munications, we simulate expert-parallel training of a Mixture of Experts (MoE) model to show our improvements at scale.

Testbed Training: We run PyTorch Distributed Data Parallel (DDP) [38] training experiments on our testbed. Figure 8 shows the results of training both small DNN models and GPT-2. We compare our topologies (from Table 5) against ShiftedRings and DBT. In training small models (Figure 8a), our topology improves total allreduce time by 30% and 50% on average against ShiftedRing and DBT, respectively. With optimizations such as compute-communication overlap, our topology still secures a 10% and 25% average improvement in iteration time over the baselines. In GPT-2 training (Figure 8b), despite the limited scale of our testbed, our topology enhances iteration time by 7% and 25% on average compared to ShiftedRing and DBT, respectively.

Large-Scale Simulation: While improvements over ShiftedRing and DBT have been shown in testbed training, full-scale LLM training is performed on much larger clusters. In Figure 9, we simulate expert-parallel training of Switch Transformers [19] on a much larger scale with parameter sizes up to 1.6 trillion. We collect execution timestamps from one A100 GPU to derive the compute times for each layer. Communication times are then added to simulate training, ensuring compute-communication overlap/dependency. Appendix §A.4 provides further details of the simulation.

Expert-parallel training involves not only data-parallel allreduce for non-expert layers but also all-to-all communications to transfer tokens to and from the routed experts, which are in the critical path of compute [19, 34, 36, 60]. In Figure 9, we break down the iteration time into compute time, non-overlapped allreduce time, and all-to-all time for a better understanding of performance. As previously analyzed in Figure 7, ShiftedRings (SR) exhibit all-to-all performance that is order-of-magnitude worse than our topologies. At 256-node training of 14.7B MoE model, ShiftedRing has 8× greater total all-to-all time, resulting in 4× longer iteration time compared to our topology. We also include 2D torus (tor) for comparison due to its relatively better all-to-all performance. However, it still has all-to-all and iteration times that are 2× and 1.5× greater, respectively, than our topology. The disparity is even larger at 1024-node training of 1.6T MoE model, where ShiftedRing and 2D torus show all-to-all times that are 27× and 3.3× greater, and iteration times that are 9× and 1.7× longer, respectively. At this scale, ShiftedRing and 2D torus spend 91% and 58% of iteration time on all-to-all communications, while our topology only spends 30%. We omit DBT in Figure 9 due to its significantly worse performance ($\sim$2× of ShiftedRing). Due to high performance in both allreduce and all-to-all, our topologies consistently remain within 5% of the theoretical lower bound (LB) for iteration time.

Since large models involve large allreduce sizes and both torus and ShiftedRing are BW-optimal, the allreduce performance is similar across these topologies. For a broader spectrum of all-to-all efficient low-hop topologies like expander graphs, the lack of efficient allreduce schedules prior to our work has prevented their use in allreduce-involved training.

8.5 BFB Schedule Evaluation

We evaluate schedule generation from two aspects: schedule generation runtime and the performance of generated schedules. In §8.5.1, we compare BFB with state-of-the-art schedule generations: SCCL [10] and TACCL [65], in both generation runtime and theoretical schedule performance. In §8.5.2, we compare, on supercomputing torus clusters, the performance of torus schedules generated by BFB, traditional torus scheduling [62], SCCL, and TACCL.

8.5.1 Schedule Generation

In schedule generation, SCCL and TACCL are the closest in spirit to BFB schedule generation. Table 6 shows the runtime comparison between SCCL, TACCL, and BFB when generating allgather schedules for hypercube and 2D torus. Both SCCL and TACCL use NP-hard optimization to generate schedules. SCCL, which uses an SMT solver, fails to generate a schedule within 10^4 seconds beyond $N = 30$. TACCL formulates the scheduling problem as a mixed integer linear program (MILP). It sets an 1800s time limit for its MILP solver, after which it will return the best solution found up to that point. However, for larger topologies, TACCL's solver fails to find a solution within the time limit, resulting in an error. In comparison, BFB schedule generation is faster by orders of magnitude due to its polynomial-time generation.

In terms of theoretical schedule performance, Figure 10 compares the total-hop latency and BW runtime of generated schedules. Given a topology, SCCL and TACCL need to perform a sweep across parameters such as the number of chunks and symmetry. They have to generate schedules for different parameter sets to identify the high-performance ones, unlike BFB, which requires no parameter. In Figure 10, the schedules of SCCL and BFB can both achieve exact optimality, but TACCL's have significantly worse performance, especially at large Ns. SCCL is uniquely capable of generating all Pareto-efficient schedules. However, due to the prohibitive runtime of parameter sweep, SCCL can only do so for very small Ns.

N	SCCL				TACCL w/o Symmetry				TACCL w/ Symmetry				BFB
	$c=1$	$c=2$	$c=3$	$c=4$	$c=1$	$c=2$	$c=3$	$c=4$	$c=1$	$c=2$	$c=3$	$c=4$	
Hypercube													
4	0.59	0.64	0.68	0.72	0.89	0.50	0.83	0.75	0.62	0.51	0.71	0.60	<0.01
8	0.86	1.22	1.86	2.48	96.9	807	63.2	1800	7.97	645	7.39	1801	<0.01
16	21.4	48.4	130	573	1801	1801	1801	1802	1801	n/a	n/a	n/a	<0.01
32	$>10^4$	$>10^4$	$>10^4$	$>10^4$	1802	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.03
64	$>10^4$	$>10^4$	$>10^4$	$>10^4$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.17
1024	$>10^4$	$>10^4$	$>10^4$	$>10^4$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	52.7
2D Torus ($n \times n$)													
4	0.61	0.63	0.67	0.76	0.68	0.50	0.82	0.72	0.45	0.51	0.76	0.64	<0.01
9	1.00	1.51	2.22	3.44	1801	189	67.8	262	88.6	71.1	67.8	105	<0.01
16	17.5	60	131	603	1801	1801	1801	1802	1801	1801	1801	n/a	<0.01
25	3286	5641	$>10^4$	$>10^4$	1802	1802	1803	n/a	1802	n/a	n/a	n/a	0.01
36	$>10^4$	$>10^4$	$>10^4$	$>10^4$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.03
2500	$>10^4$	$>10^4$	$>10^4$	$>10^4$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	61.1

Table 6: Comparing allgather schedule generation runtimes (in seconds) of SCCL, TACCL, and BFB. The setup of SCCL is to generate schedules with the least number of comm steps. Both SCCL and TACCL were run with chunks=1,2,3,4 (number of chunks per shard), and TACCL was run w/ and w/o manually set topology symmetry. "n/a" indicates where TACCL reports an error due to failure to generate a solution within its 1800s time limit for MILP solver.

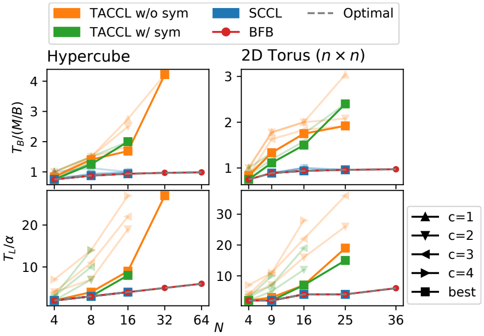


Figure 10: Comparing theoretical performances of schedules from Table 6. We show both T_L and T_B of the schedules, along with the theoretical optimal. For SCCL and TACCL, the solid lines show the best results from parameter sweeps. The inferior ones are dimmed.

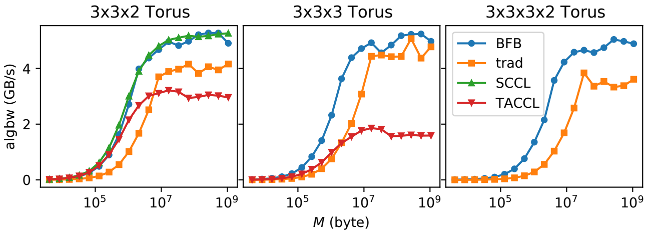


Figure 11: Comparing allreduce performances of torus schedules generated by BFB, traditional torus scheduling [62], SCCL, and TACCL on Frontera [69] supercomputer. The y-axis is algorithmic bandwidth (algbw), computed as M divided by end-to-end runtime. SCCL fails to generate a schedule for 3×3×3 and 3×3×3×2 tori, and TACCL fails to generate a schedule for 3×3×3×2 torus within the time limits.

8.5.2 Supercomputing Allreduce Experiments

In the supercomputing setting, we run torus schedules generated by BFB, traditional torus scheduling [62], SCCL, and TACCL on Frontera [69] supercomputer at the Texas Advanced Computing Center (TACC) [71]. The cluster consists of 396 nodes in a 6D torus direct-connect topology. Each node is equipped with an Intel Xeon Platinum 8280 CPU and a Rockport NC1225 network card, capable of delivering 25 Gbps per link, with degree 12. However, the total BW of a single node may be bottlenecked by the 100 Gbps host BW of PCIe Gen3 x16. Finally, the schedules are lowered and run using Intel oneCCL [26] + libfabric [39].

We run schedules on two types of sub-torus within the cluster: equal-dimension (3×3×3) and unequal-dimension (3×3×2 & 3×3×3×2). As shown in Figure 11, BFB schedules achieve the highest performance in all settings. As mentioned in §6.2, the traditional torus schedule can only achieve high BW performance in tori with equal dimensions. At large M, it matches BFB's performance in 3×3×3 torus but significantly underperforms in 3×3×2 and 3×3×3×2, where BFB has 29% and 42% higher algbw on average for $M \geq 100$MB. At small to intermediate M (< 100MB), BFB outperforms traditional schedules by 3.1× on average in all settings due to its 2× lower in total-hop latency and higher BW performance.

As for SCCL and TACCL, we adhere to the same time limits and parameter sweeps as in §8.5.1 and select the best result at each M from all parameter sets. In 3×3×2 torus, SCCL is able to generate an optimal schedule, matching BFB's performance across all M. However, it fails to generate a schedule within 10^4 seconds for other larger tori. TACCL can only generate schedules in 3×3×2 and 3×3×3, and its schedules underperform BFB's by a large margin. One additional observation is that the algbw of BFB at large M hardly changes from 18-node (3×3×2) to 54-node (3×3×3×2) torus. This can be explained by the fact that BFB schedules have theoretically achieved allreduce BW optimality ($\frac{2M}{B} \cdot \frac{N-1}{N}$), which remains nearly constant as N increases.

9 Concluding Remarks

Collective communications are critical to both ML training and HPC workloads. Current solutions often rely solely on existing topologies and schedules, resulting in high total-hop latency, bandwidth inefficiency, or low all-to-all throughput. We presented a general, highly scalable, and automated algorithmic framework for optimizing topology and schedule generation for collectives by leveraging scalable graph-theoretic approaches. Our evaluation demonstrates significant performance gains across multiple testbeds and large-scale simulations in both standalone collective communications and end-to-end ML training.

10 Acknowledgements

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) under Contract No.HR001120C0089. This work was also supported in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] Ballani, H., Costa, P., Behrendt, R., Cletheroe, D., Haller, I., Jozwik, K., Karinou, F., Lange, S., Shi, K., Thomsen, B., and Williams, H. Sirius: A flat datacenter network with nanosecond optical switching. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (New York, NY, USA, 2020), SIGCOMM '20, Association for Computing Machinery, p. 782–797.

[2] Bang, S., Dubickas, A., Koolen, J., and Moulton, V. There are only finitely many distance-regular graphs of fixed valency greater than two. *Advances in Mathematics 269* (2015), 1–55.

[3] Barnett, M., Shuler, L., van De Geijn, R., Gupta, S., Payne, D. G., and Watts, J. Interprocessor collective communication library (intercom). In *Proceedings of IEEE Scalable High Performance Computing Conference* (1994), IEEE, pp. 357–364.

[4] Basu, P., Zhao, L., Fantl, J., Pal, S., Krishnamurthy, A., and Khoury, J. Efficient all-to-all collective communication schedules for direct-connect topologies. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing* (New York, NY, USA, 2024), HPDC '24, Association for Computing Machinery, p. 28–41.

[5] Bermond, J.-C., Homobono, N., and Peyrat, C. Connectivity of kautz networks. *Discrete Math. 114*, 1-3 (apr 1993), 51–62.

[6] Besta, M., and Hoefler, T. Slim Fly: A cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), SC '14, IEEE Press, p. 348–359.

[7] Boesch, F., and Wang, J.-F. Reliable circulant networks with minimum transmission delay. *IEEE Transactions on Circuits and Systems 32*, 12 (1985), 1286–1291.

[8] Bokhari, S. H., and Berryman, H. Complete exchange on a circuit switched mesh. In *Proceedings Scalable High Performance Computing Conference SHPCC-92.* (1992), IEEE, pp. 300–306.

[9] Broadcom P2200G - 2x200GbE PCIe NIC. https://www.broadcom.com/products/ethernet-connectivity/network-adapters/200gb-nic-ocp/p2200g.

[10] Cai, Z., Liu, Z., Maleki, S., Musuvathi, M., Mytkowicz, T., Nelson, J., and Saarikivi, O. Synthesizing optimal collective algorithms. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (New York, NY, USA, 2021), PPoPP '21, Association for Computing Machinery, p. 62–75.

[11] Chan, E., Heimlich, M., Purkayastha, A., and van de Geijn, R. Collective communication: Theory, practice, and experience: Research articles. *Concurr. Comput. : Pract. Exper. 19*, 13 (sep 2007), 1749–1783.

[12] Chan, E., Van De Geijn, R., Gropp, W., and Thakur, R. Collective communication on architectures that support simultaneous communication over multiple links. In *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming* (2006), pp. 2–11.

[13] Cho, M., Finkler, U., and Kung, D. BlueConnect: Novel hierarchical all-reduce on multi-tired network for deep learning. In *Proceedings of the 2nd SysML Conference* (2019).

[14] Cámara, J. M., Moretó, M., Vallejo, E., Beivide, R., Miguel-Alonso, J., Martínez, C., and Navaridas, J. Twisted torus topologies for enhanced interconnection networks. *IEEE Transactions on Parallel and Distributed Systems 21*, 12 (2010), 1765–1778.

[15] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. Large scale distributed deep networks. *Advances in neural information processing systems 25* (2012).

[16] DistanceRegular.org. https://www.math.mun.ca/distanceregular/.

[17] Esfahanian, A.-H., Ni, L., and Sagan, B. The twisted n-cube with application to multiprocessing. *IEEE Transactions on Computers 40*, 1 (1991), 88–93.

[18] Faizian, P., Mollah, M. A., Yuan, X., Alzaid, Z., Pakin, S., and Lang, M. Random regular graph and generalized de bruijn graph with k-shortest path routing. *IEEE Transactions on Parallel and Distributed Systems 29*, 1 (2017), 144–155.

[19] Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research 23*, 120 (2022), 1–39.

[20] Gibiansky, A. Bringing hpc techniques to deep learning. *Baidu Research, Tech. Rep.* (2017).

[21] GRINER, C., ZERWAS, J., BLENK, A., GHOBADI, M., SCHMID, S., AND AVIN, C. Cerberus: The power of choices in datacenter topology design - a throughput perspective. *Proc. ACM Meas. Anal. Comput. Syst. 5*, 3 (dec 2021).

[22] HARARY, F., AND NORMAN, R. Z. Some properties of line digraphs. *Rendiconti del Circolo matematico di Palermo 9*, 2 (1960), 161–168.

[23] HO, C.-T., AND JOHNSSON, S. L. Distributed routing algorithms for broadcasting and personalized communication in hypercubes. In *ICPP* (1986), pp. 640–648.

[24] HOCKNEY, R. W. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel computing 20*, 3 (1994), 389–398.

[25] IMASE, AND ITOH. A design for directed graphs with minimum diameter. *IEEE Transactions on Computers C-32*, 8 (1983), 782–784.

[26] INTEL. oneAPI Collective Communications Library (oneCCL). https://github.com/oneapi-src/oneCCL.

[27] JEAUGEY, S. Massively scale your deep learning training with nccl 2.4. https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/, 2019.

[28] JIA, Z., ZAHARIA, M., AND AIKEN, A. Beyond data and model parallelism for deep neural networks. *Proceedings of Machine Learning and Systems 1* (2019), 1–13.

[29] JOUPPI, N., KURIAN, G., LI, S., MA, P., NAGARAJAN, R., NAI, L., PATIL, N., SUBRAMANIAN, S., SWING, A., TOWLES, B., YOUNG, C., ZHOU, X., ZHOU, Z., AND PATTERSON, D. A. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (2023).

[30] KHANI, M., GHOBADI, M., ALIZADEH, M., ZHU, Z., GLICK, M., BERGMAN, K., VAHDAT, A., KLENK, B., AND EBRAHIMI, E. Sip-ml: High-bandwidth optical network interconnects for machine learning training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (New York, NY, USA, 2021), SIGCOMM '21, Association for Computing Machinery, p. 657–675.

[31] KIM, J., DALLY, W. J., SCOTT, S., AND ABTS, D. Technology-driven, highly-scalable dragonfly topology. In *2008 International Symposium on Computer Architecture* (2008), pp. 77–88.

[32] KIM, J., GHAYOORI, A., AND SRIKANT, R. All-to-all communication in random regular directed graphs. *IEEE Transactions on Network Science and Engineering 1*, 1 (2014), 43–52.

[33] LAKHOTIA, K., BESTA, M., MONROE, L., ISHAM, K., IFF, P., HOEFLER, T., AND PETRINI, F. PolarFly: A cost-effective and flexible low-diameter topology. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (2022), SC '22, IEEE Press.

[34] LEPIKHIN, D., LEE, H., XU, Y., CHEN, D., FIRAT, O., HUANG, Y., KRIKUN, M., SHAZEER, N., AND CHEN, Z. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations* (2021).

[35] LI, A., SONG, S. L., CHEN, J., LI, J., LIU, X., TALLENT, N. R., AND BARKER, K. J. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems 31*, 1 (2019), 94–110.

[36] LI, J., JIANG, Y., ZHU, Y., WANG, C., AND XU, H. Accelerating distributed MoE training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)* (Boston, MA, July 2023), USENIX Association, pp. 945–959.

[37] LI, M., ANDERSEN, D. G., PARK, J. W., SMOLA, A. J., AHMED, A., JOSIFOVSKI, V., LONG, J., SHEKITA, E. J., AND SU, B.-Y. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on operating systems design and implementation (OSDI 14)* (2014), pp. 583–598.

[38] LI, S., ZHAO, Y., VARMA, R., SALPEKAR, O., NOORDHUIS, P., LI, T., PASZKE, A., SMITH, J., VAUGHAN, B., DAMANIA, P., AND CHINTALA, S. PyTorch distributed: experiences on accelerating data parallel training. *Proc. VLDB Endow. 13*, 12 (Aug. 2020), 3005–3018.

[39] libfabric Open Fabrics Interfaces (OFI). https://github.com/ofiwg/libfabric.

[40] LIU, H., URATA, R., YASUMURA, K., ZHOU, X., BANNON, R., BERGER, J., DASHTI, P., JOUPPI, N., LAM, C., LI, S., MAO, E., NELSON, D., PAPEN, G., TARIQ, M., AND VAHDAT, A. Lightwave fabrics: At-scale optical circuit switching for datacenter and machine learning systems. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA, 2023), ACM SIGCOMM '23, Association for Computing Machinery, p. 499–515.

[41] LOGUINOV, D., CASAS, J., AND WANG, X. Graph-theoretic analysis of structured peer-to-peer systems:

routing distances and fault resilience. *IEEE/ACM Transactions on Networking 13*, 5 (2005), 1107–1120.

[42] LOGUINOV, D., KUMAR, A., RAI, V., AND GANESH, S. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2003), SIGCOMM '03, Association for Computing Machinery, p. 395–406.

[43] LU, Y., GU, H., YU, X., AND LI, P. X-NEST: A scalable, flexible, and high-performance network architecture for distributed machine learning. *Journal of Lightwave Technology 39*, 13 (2021), 4247–4254.

[44] LUO, L., WEST, P., NELSON, J., KRISHNAMURTHY, A., AND CEZE, L. Plink: Discovering and exploiting locality for accelerated distributed training on the public cloud. In *Proceedings of Machine Learning and Systems* (2020), I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, pp. 82–97.

[45] LUO, L., ZHANG, B., TSANG, M., MA, Y., CHU, C.-H., CHEN, Y., LI, S., HAO, Y., ZHAO, Y., LAKSHMINARAYANAN, G., WEN, E., PARK, J., MUDIGERE, D., AND NAUMOV, M. Disaggregated multi-tower: Topology-aware modeling technique for efficient large scale recommendation. In *Proceedings of Machine Learning and Systems* (2024), P. Gibbons, G. Pekhimenko, and C. D. Sa, Eds., vol. 6, pp. 266–278.

[46] MEIJER, P. T. *Connectivities and diameters of circulant graphs*. PhD thesis, Theses (Dept. of Mathematics and Statistics)/Simon Fraser University, 1991.

[47] MELLETTE, W. M., DAS, R., GUO, Y., MCGUINNESS, R., SNOEREN, A. C., AND PORTER, G. Expanding across time to deliver bandwidth efficiency and low latency. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)* (Santa Clara, CA, Feb. 2020), USENIX Association, pp. 1–18.

[48] MELLETTE, W. M., MCGUINNESS, R., ROY, A., FORENCICH, A., PAPEN, G., SNOEREN, A. C., AND PORTER, G. RotorNet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017).

[49] Microsoft Collective Communication Library (MSCCL). https://github.com/microsoft/msccl.

[50] MILLER, M., AND SIRAN, J. Moore graphs and beyond: A survey of the degree/diameter problem. *Electronic Journal of Combinatorics 1000* (2013).

[51] MONAKHOVA, E. A. A survey on undirected circulant graphs. *Discrete Mathematics, Algorithms and Applications 04*, 01 (2012), 1250002.

[52] NARAYANAN, D., SHOEYBI, M., CASPER, J., LEGRESLEY, P., PATWARY, M., KORTHIKANTI, V., VAINBRAND, D., KASHINKUNTI, P., BERNAUER, J., CATANZARO, B., ET AL. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2021), pp. 1–15.

[53] NAUMOV, M., KIM, J., MUDIGERE, D., SRIDHARAN, S., WANG, X., ZHAO, W., YILMAZ, S., KIM, C., YUEN, H., OZDAL, M., NAIR, K., GAO, I., SU, B.-Y., YANG, J., AND SMELYANSKIY, M. Deep learning training in facebook data centers: Design of scale-up and scale-out systems, 2020.

[54] NAUMOV, M., MUDIGERE, D., SHI, H.-J. M., HUANG, J., SUNDARAMAN, N., PARK, J., WANG, X., GUPTA, U., WU, C.-J., AZZOLINI, A. G., DZHULGAKOV, D., MALLEVICH, A., CHERNIAVSKII, I., LU, Y., KRISHNAMOORTHI, R., YU, A., KONDRATENKO, V., PEREIRA, S., CHEN, X., CHEN, W., RAO, V., JIA, B., XIONG, L., AND SMELYANSKIY, M. Deep learning recommendation model for personalization and recommendation systems, 2019.

[55] NVIDIA A100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/a100/.

[56] NVIDIA Collective Communications Library (NCCL). https://github.com/NVIDIA/nccl.

[57] NVIDIA ConnectX-6 Dx Datasheet. https://www.nvidia.com/content/dam/en-zz/Solutions/networking/ethernet-adapters/connectX-6-dx-datasheet.pdf.

[58] PATARASUK, P., AND YUAN, X. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing 69*, 2 (2009), 117–124.

[59] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., SUTSKEVER, I., ET AL. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[60] RAJBHANDARI, S., LI, C., YAO, Z., ZHANG, M., AMINABADI, R. Y., AWAN, A. A., RASLEY, J., AND HE, Y. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning* (17–23 Jul 2022), K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and

S. Sabato, Eds., vol. 162 of *Proceedings of Machine Learning Research*, PMLR, pp. 18332–18346.

[61] ROLIM, J., TVRDIK, P., TRDLIČKA, J., AND VRTO, I. Bisecting de bruijn and kautz graphs. *Discrete Applied Mathematics 85*, 1 (1998), 87–97.

[62] SACK, P., AND GROPP, W. Collective algorithms for multiported torus networks. *ACM Trans. Parallel Comput. 1*, 2 (feb 2015).

[63] SANDERS, P., SPECK, J., AND TRÄFF, J. L. Two-tree algorithms for full bandwidth broadcast, reduction and scan. *Parallel Comput. 35*, 12 (dec 2009), 581–594.

[64] SERGEEV, A., AND BALSO, M. D. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799* (2018).

[65] SHAH, A., CHIDAMBARAM, V., COWAN, M., MALEKI, S., MUSUVATHI, M., MYTKOWICZ, T., NELSON, J., SAARIKIVI, O., AND SINGH, R. TACCL: Guiding collective algorithm synthesis using communication sketches. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)* (Boston, MA, Apr. 2023), USENIX Association, pp. 593–612.

[66] SHAZEER, N., MIRHOSEINI, A., MAZIARZ, K., DAVIS, A., LE, Q., HINTON, G., AND DEAN, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations* (2017).

[67] SHOEYBI, M., PATWARY, M., PURI, R., LEGRESLEY, P., CASPER, J., AND CATANZARO, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[68] SINGLA, A., HONG, C.-Y., POPA, L., AND GODFREY, P. B. Jellyfish: Networking data centers randomly. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)* (San Jose, CA, Apr. 2012), USENIX Association, pp. 225–238.

[69] STANZIONE, D., WEST, J., EVANS, R. T., MINYARD, T., GHATTAS, O., AND PANDA, D. K. Frontera: The evolution of leadership computing at the national science foundation. In *Practice and Experience in Advanced Research Computing* (New York, NY, USA, 2020), PEARC '20, Association for Computing Machinery, p. 106–111.

[70] Telescent G4 Network Topology Manager. https://www.telescent.com/products.

[71] Texas Advanced Computing Center (TACC). https://www.tacc.utexas.edu/.

[72] THAKUR, R., RABENSEIFNER, R., AND GROPP, W. Optimization of collective communication operations in MPICH. *The International Journal of High Performance Computing Applications 19*, 1 (2005), 49–66.

[73] TRUONG, T.-N., AND TAKANO, R. Hybrid electrical/optical switch architectures for training distributed deep learning in large-scale. *IEICE Transactions on Information and Systems E104.D*, 8 (2021), 1332–1339.

[74] UENO, Y., AND YOKOTA, R. Exhaustive study of hierarchical allreduce patterns for large messages between gpus. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (2019), pp. 430–439.

[75] VALADARSKY, A., SHAHAF, G., DINITZ, M., AND SCHAPIRA, M. Xpander: Towards optimal-performance datacenters. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, 2016), CoNEXT '16, Association for Computing Machinery, p. 205–219.

[76] WANG, G., VENKATARAMAN, S., PHANISHAYEE, A., DEVANUR, N., THELIN, J., AND STOICA, I. Blink: Fast and generic collectives for distributed ML. *Proceedings of Machine Learning and Systems 2* (2020), 172–186.

[77] WANG, W., KHAZRAEE, M., ZHONG, Z., GHOBADI, M., JIA, Z., MUDIGERE, D., ZHANG, Y., AND KEWITSCH, A. TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)* (Boston, MA, Apr. 2023), USENIX Association, pp. 739–767.

[78] WEBB, K. C., SNOEREN, A. C., AND YOCUM, K. Topology switching for data center networks. In *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 11)* (2011).

[79] WICKRAMASINGHE, U., AND LUMSDAINE, A. A survey of methods for collective communication optimization and tuning. *arXiv preprint arXiv:1611.06334* (2016).

[80] YOUNG, S., AKSOY, S., FIROZ, J., GIOIOSA, R., HAGGE, T., KEMPTON, M., ESCOBEDO, J., AND RAUGAS, M. SpectralFly: Ramanujan graphs as flexible and efficient interconnection networks. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2022), pp. 1040–1050.

[81] ZHU, Z., TEH, M. Y., WU, Z., GLICK, M. S., YAN, S., HATTINK, M., AND BERGMAN, K. Distributed deep learning training using silicon photonic switched architectures. *APL Photonics 7*, 3 (2022), 030901.

[82] ZOPH, B., BELLO, I., KUMAR, S., DU, N., HUANG, Y., DEAN, J., SHAZEER, N., AND FEDUS, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906* (2022).

Appendix

In this appendix, we give additional evaluation results, along with formal mathematical definitions and analyses of the various techniques and concepts discussed in the main text:

- §A provides supplementary materials to evaluation section.
- §B gives formal definitions of reduce-scatter/allgather schedule and how one can be transformed into another.
- §C gives formal definitions of total-hop latency and bandwidth optimality, along with discussions on optimal allreduce schedule and computational cost of reduction.
- §D provides formal definitions of expansion techniques and optimality analysis of their expanded schedules.
- §E provides optimality analysis of BFB schedule generation and discusses variant formulations that support generating schedules for a fixed number of chunks and for heterogeneous network topology.
- §F discusses various generative topologies and the performance of their generated BFB schedules.
- §G provides proofs of all theorems in this paper.
- §H contains supplementary tables and figures. In particular, Table 9 gives a summary of topologies in this paper.

A Evaluation Appendix

- §A.1 presents experiment results that compare BFB schedule generation with communication solutions for switch networks: NCCL [56] and recursive halving & doubling.
- §A.2 shows experiment results to validate α-β cost model.
- §A.3 gives an analysis of Pareto-efficient topologies/schedules under different hardware and workload specifications.
- §A.4 details setup of simulated DNN training and the topologies generated by our topology finder.
- §A.5 provides the multi-commodity flow (MCF) formulation used to compute all-to-all throughput.
- §A.6 shows how to convert unidirectional topologies/schedules into bidirectional ones.

A.1 Comparison Against Switch Solutions

NCCL [56] and recursive halving & doubling (RH&D) are widely adopted collective communication solutions on switch networks. We assess the schedule performance of BFB against these solutions over two direct-connect 8-node topologies: hypercube and twisted hypercube [17]. Hypercube is widely used in HPC settings, and its connections perfectly match the communication pattern of RH&D. Twisted hypercube is a variant of hypercube with a lower diameter.

Figure 13 compares the baselines against our BFB schedule when run over either hypercube or twisted hypercube with $N=8$, $d=3$ on the testbed. At small M, all schedules and topologies perform roughly the same, except BFB can take advantage of the lower diameter of twisted hypercube and

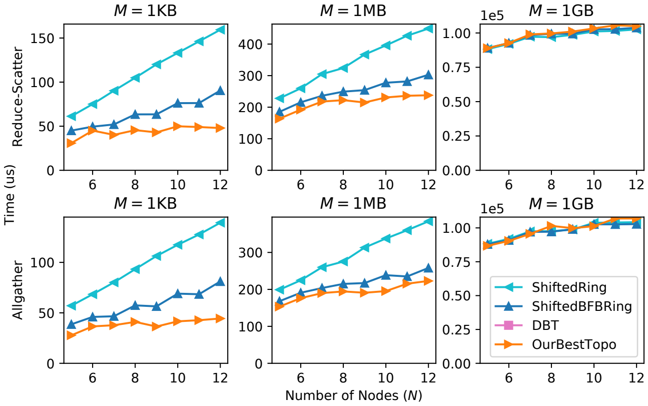


Figure 12: Comparing reduce-scatter and allgather runtimes of topologies. This figure shows the corresponding reduce-scatter and allgather results of Figure 6.

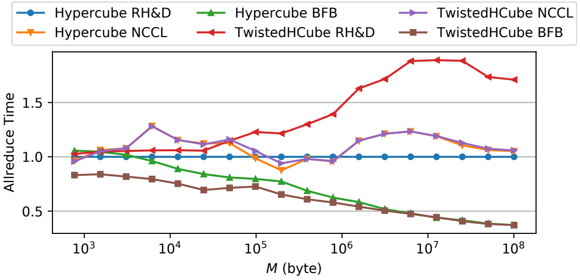
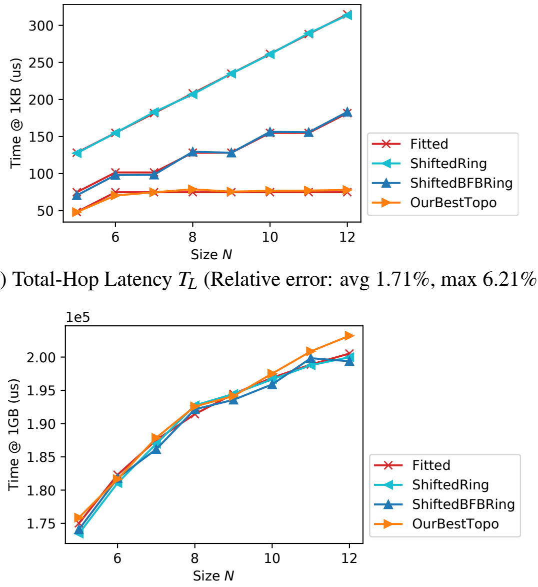


Figure 13: Comparing switch allreduce solutions (recursive halving & doubling (RH&D), NCCL) against BFB schedule on hypercube and twisted hypercube on $N = 8, d = 3$ testbed. The runtimes are normalized by the runtime of recursive halving & doubling on hypercube.

achieve $\sim$20% lower runtime. At large M, because BFB achieves BW optimality on both topologies, it performs even better with 60% lower runtime. RH&D and NCCL perform poorly as M grows because they cannot utilize all $d = 3$ links simultaneously. At every comm step of RH&D, a node only communicates with one of the three neighbors, utilizing at most $1/3$ of the total bandwidth (similarly with NCCL). Also, because the schedule is not matched to the twisted hypercube, some nodes communicate with nodes multiple hops away, occupying more links and causing congestion.

A.2 Cost Model Validation

Despite the wide acceptance of α-β cost model by previous literature [10, 11, 24, 63, 65], we also conducted a linear regression analysis to validate the cost model on our testbed. In particular, we want to verify that (1) total-hop latency follows $T_L = \alpha \cdot x + \varepsilon$ and (2) BW runtime follows $T_B = \frac{M}{B} \cdot y$, where x and y are the number of comm steps and bandwidth factor respectively ($y = 2 \cdot \frac{N-1}{N}$ if BW-optimal). ε is the constant latency[4] including time costs such as GPU kernel launching. Here, we use linear regression to derive the values of α, ε, and $1/B$, and compute the relative errors between the observed runtimes and expected runtimes. We fit the allreduce runtimes at 1KB to the total-hop latency, since BW runtime is negligible at such a small M. Similarly, we fit the runtimes at 1GB to the BW runtime, since total-hop latency is negligible at such a large M.



(a) Total-Hop Latency T_L (Relative error: avg 1.71%, max 6.21%)

(b) BW Runtime T_B (Relative error: avg 0.47%, max 1.32%)

Figure 14: Linear regression results.

Figure 14 shows our results of linear regression analysis to verify our cost model. For total-hop latency, we obtain estimates $\alpha \approx 13.33$us and $\varepsilon \approx 21.60$us with low errors (average and maximum relative errors of 1.71% and 6.21% respectively). As one can see from Figure 14a, ShiftedRing and ShiftedBFBRing have a straight and a stair-step shape of runtime growth respectively, which match the expected numbers of comm steps $2(N-1)$ and $2\lfloor N/2 \rfloor$ respectively. For BW runtime, we get an estimate $1/B \approx 1.018 \times 10^{-4}$us/byte or $B \approx 79$Gbps with low errors (average and maximum relative errors of 0.47% and 1.32% respectively). As one can see from Figure 14b, all three topologies follow the fitted curve $2\frac{1\text{GB}}{B} \cdot \frac{N-1}{N} = 2T_B^*(N)$ since they are all BW-optimal. However, there is a gap between $B \approx 79$Gbps and the hardware theoretical bandwidth 4x25Gbps=100Gbps. Besides inevitable loss of bandwidth in actual communication, the gap can also be explained by the fact that computational cost of reduction also accounts for part of $1/B$ as discussed in §C.4.

A.3 Pareto-Efficiency Analysis

There could exist multiple Pareto-efficient topologies at given N and d. For different α and M/B, the Pareto-efficient topology with minimum allreduce runtime is also different. To see how N affects the best choice of topologies, we use topology finder (§5.4) to generate Pareto-efficient topologies at $d = 4$ for N up to 2000 and pick the best one based on specific values of α and M/B.

[4] This part of latency is a fixed constant for all topologies and schedules, so it is omitted earlier.

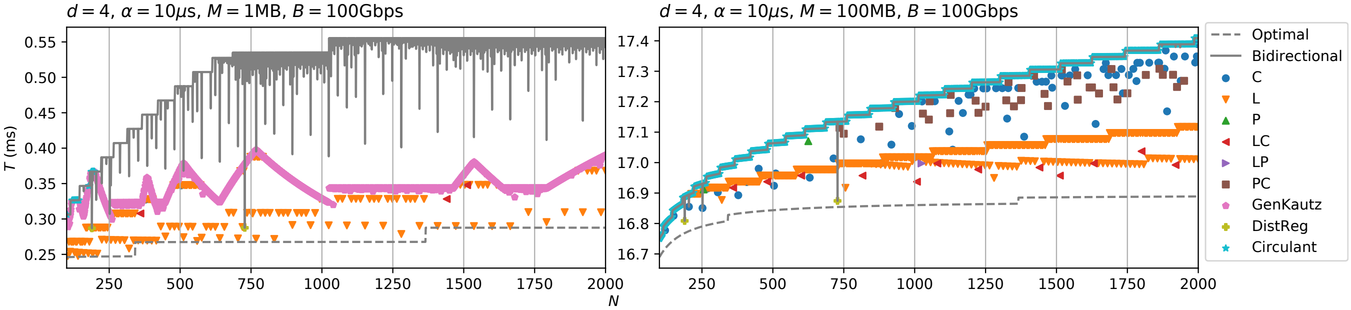


Figure 15: The minimum allreduce runtimes at different N for $d=4$, $\alpha=10$us, and $M/B=1$MB$/100$Gbps$,100$MB$/100$Gbps. "L", "P", and "C" stand for line graph, Cartesian power, and Cartesian product (of different graphs) respectively. For example, "LC" means the runtime is achieved by a topology whose construction involves line graph expansion and Cartesian product. "GenKautz", "DistReg", and "Circulant" stand for generalized Kautz graph (§F.2), distance regular graph (§F.3), and circulant graph (§F.4) respectively. The figures also show the best bidirectional topology known at different Ns. Degree expansion does not show up due to target $d=4$ being relatively small.

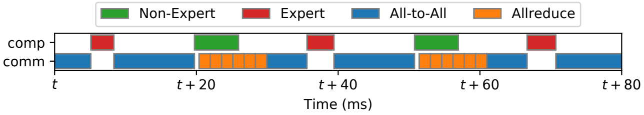


Figure 16: Example of a training timeline for Switch Transformers.

Figure 15 shows two examples of such analysis. At $M=1$MB, total-hop latency is more important than BW runtime. Thus, we see that generalized Kautz graph is the most popular one, being the best topology at many Ns. On the contrary, at $M=100$MB, BW performance becomes the dominant factor, and thus circulant graph becomes the most popular one. Line graphs are also popular in both settings; however, line graph expansion requires target N to be divisible by some power of d, so it does not work for any N.

A.4 Details of Simulated Distributed Training

We simulate distributed ML training by first collecting actual compute times for model layers, running the models on an NVIDIA A100-SXM-80GB GPU, and then adding communication times according to the specific parallelism, e.g., data or expert parallelism. The communication time is calculated using α-β model for allreduce (§A.2) and multi-commodity flow for all-to-all (§A.5), assuming $\alpha=10$us, $B=100$Gbps over $d=4$. Our simulation is designed to match the compute-communication overlap pattern of PyTorch Distributed Data Parallel [38]. As in PyTorch DDP, we bucket gradients that are ready for allreduce during backward propagation. Once the gradient volume reaches a predefined bucket capacity, an allreduce is performed. While a large bucket size results in less latency overhead, a small bucket size enhances compute-communication overlap. We choose the best bucket size by comparing the iteration times of bucket sizes {1MB, 10MB, 100MB, 1GB}. To ensure overlap, computation and communication are handled as independent streams, with the communication stream executing one collective at a time.

For the simulated training of Mixture-of-Experts (MoE) models, we follow the standard practice of expert parallelism [19, 34, 36, 60], where experts are sharded across all nodes while non-expert layers are replicated. All-to-all communications are needed before the expert layers to route tokens to the nodes of the assigned experts, and afterward to return tokens to the original nodes for the continuation of the forward/backward pass, thus blocking the computation stream. Furthermore, all-to-all and allreduce are not allowed to be overlapped as they occupy the same network bandwidth [36]. Figure 16 shows a timeline example of the simulated expert-parallel training. For simplicity, we assume a uniform token distribution among the experts, as MoE models are trained to balance expert load [19, 66, 82]. Consequently, the all-to-all communication is uniform across the nodes. We use the multi-commodity flow formulation (3) to compute the all-to-all communication time.

All hyperparameters, including sequence lengths and global batch sizes, are chosen according to the original paper of Switch Transformers [19]. The topology degree is fixed at 4, and the topology sizes are chosen such that the local batch size at each node is ≥ 1 and not so large as to run out of GPU memory. Table 7 includes all the Pareto-efficient topologies used in the simulation. For each model and topology size, we choose the topology that results in the smallest iteration time.

A.5 All-to-All Throughput

The problem of deriving the throughput of all-to-all communication on a topology can be nicely formulated as a multi-commodity flow (MCF) problem [4, 21, 32, 75, 78]. In an all-to-all MCF, each pair of nodes $(s,t)\in V_G^2$ acts as the source and sink of a commodity. The objective is to simultaneously route f units of flows from each s to t such that f is maximized, with flow allocation subject to flow conservation and edge capacities. In [4], the authors have devised an efficient LP formulation to compute the optimal f:

$$\begin{array}{lrll} \text{maximize} & f & & \\ \text{subject to} & \sum_s y_{s,(u,v)} \leq 1, & & \forall u,v \\ & f+\sum_v y_{s,(u,v)} \leq \sum_w y_{s,(w,u)}, & & \forall s,u: s\neq u \\ & y_{s,(u,v)} \geq 0. & & \forall s,u,v \end{array} \tag{3}$$

In LP (3), we assume the capacity/bandwidth of each link is 1 unit. Therefore, if the bandwidth of each link is B/d,

Topology	**Allreduce**		**All-to-All**	
$N=32$, $d=4$	T_L	T_B	$D(G)$	MCF
$L(K_{4,4})$	3α	$1.000{}^{M}/_{B}$	3	5.71e−2
$\mathrm{DistReg}(4,32)$	4α	$0.969{}^{M}/_{B}$	4	5.26e−2
Theoretical Bound	3α	$0.969{}^{M}/_{B}$	3	5.80e−2
$N=64$, $d=4$	T_L	T_B	$D(G)$	MCF
$\Pi_{4,64}$	3α	$1.312{}^{M}/_{B}$	3	2.17e−2
$L(\mathrm{DBJMod}(4,2))$	4α	$1.000{}^{M}/_{B}$	4	2.21e−2
$\mathrm{Diamond}^{\square 2}$	6α	$0.984{}^{M}/_{B}$	6	1.87e−2
Theoretical Bound	3α	$0.984{}^{M}/_{B}$	3	2.42e−2
$N=128$, $d=4$	T_L	T_B	$D(G)$	MCF
$L^2(K_{4,4})$	4α	$1.031{}^{M}/_{B}$	4	9.89e−3
$L(\mathrm{DistReg}(4,32))$	5α	$1.000{}^{M}/_{B}$	5	9.26e−3
$\mathrm{BiRing}(2,8)\square\mathrm{UniRing}(1,4)^{\square 2}$	10α	$0.992{}^{M}/_{B}$	10	5.21e−3
Theoretical Bound	4α	$0.992{}^{M}/_{B}$	4	1.00e−2
$N=256$, $d=4$	T_L	T_B	$D(G)$	MCF
$\mathrm{DBJ}(4,4)$	4α	$1.328{}^{M}/_{B}$	4	4.04e−3
$L^2(\mathrm{DBJMod}(4,2))$	5α	$1.016{}^{M}/_{B}$	5	4.10e−3
$L(\mathrm{Diamond}^{\square 2})$	7α	$1.000{}^{M}/_{B}$	7	3.62e−3
$\mathrm{DBJMod}(2,4)^{\square 2}$	10α	$0.996{}^{M}/_{B}$	8	2.94e−3
Theoretical Bound	4α	$0.996{}^{M}/_{B}$	4	4.39e−3
$N=512$, $d=4$	T_L	T_B	$D(G)$	MCF
$L^3(K_{4,4})$	5α	$1.039{}^{M}/_{B}$	5	1.88e−3
$L^2(\mathrm{DistReg}(4,32))$	6α	$1.008{}^{M}/_{B}$	6	1.78e−3
$L(\mathrm{BiRing}(1,4)^{\square 3}\square\mathrm{UniRing}(1,2))$	11α	$1.000{}^{M}/_{B}$	11	1.12e−3
$\mathrm{UniRing}(1,4)^{\square 3}\square\mathrm{UniRing}(1,8)$	16α	$0.998{}^{M}/_{B}$	16	5.58e−4
Theoretical Bound	5α	$0.998{}^{M}/_{B}$	5	1.90e−3
$N=1024$, $d=4$	T_L	T_B	$D(G)$	MCF
$\Pi_{4,1024}$	5α	$1.332{}^{M}/_{B}$	5	8.01e−4
$L^3(C(16,\{3,4\}))$	6α	$1.020{}^{M}/_{B}$	6	8.12e−4
$L^2(\mathrm{Diamond}^{\square 2})$	8α	$1.004{}^{M}/_{B}$	8	7.34e−4
$L(\mathrm{DBJMod}(2,4)^{\square 2})$	11α	$1.000{}^{M}/_{B}$	9	6.18e−4
$(\mathrm{UniRing}(1,4)\square\mathrm{UniRing}(1,8))^{\square 2}$	20α	$0.999{}^{M}/_{B}$	20	2.79e−4
Theoretical Bound	5α	$0.999{}^{M}/_{B}$	5	8.57e−4

Table 7: Pareto-efficient topologies at $N\in\{32,64,128,256,512,1024\}$**,** $d=4$**.** The results are generated from the topology finder (§5.4). For notations of the topologies, see Table 3 and 9. For distance regular graphs (DistReg), see Table 8. The MCF values are computed using LP (3).

then fB/d represents the rate at which every node can send to every other node simultaneously.

A.6 Unidirectional to Bidirectional

Unidirectional topologies are technically feasible on optical testbeds. The optical cable contains two fibers, one for each direction, and the fabric can link them to two distinct end-hosts, thus enabling unidirectional topologies at no additional hardware cost.

However, in our evaluation, we only use bidirectional topologies. While unidirectional topologies can be realized by configuring the patch panel in simplex mode, the requisite overlay routing for the reverse path traffic (acks, etc.) is currently only supported using routing rules performed by the host kernel as opposed to the NIC, leading to unpredictable RTTs. Therefore, we can functionally validate unidirectional topologies on our testbed, but we cannot accurately evaluate their performance. Note that newer NICs [9, 57] do support hardware offloading for these rules, which we will examine in future work.

While this paper considers unidirectional topologies a lot, many of the techniques can be conveniently applied to bidirectional topologies as well. For example, BFB schedule generation, degree expansion, and Cartesian product can all be used on bidirectional topologies by replacing each bidirectional edge with two opposite unidirectional edges. The resulting degree expanded and Cartesian product topologies still have unidirectional edges in opposite pairs. Although line graph expansion only works within unidirectional topologies, there is a way to convert unidirectional topology and schedule to bidirectional ones with zero performance sacrifice. In this section, we will show how to convert a reverse-symmetric (see Definition 6) d-regular unidirectional topology G and its allgather schedule A to a $2d$-regular bidirectional topology G' and its schedule A' such that $T_L(A)=T_L(A')$ and $T_B(A)=T_B(A')$.

Let $g: V_G \to V_{G^T}$ be the isomorphism from G to G^T, then it is trivial to see that $g(A)$ (see Definition 7) is an allgather schedule for G^T. Observe that $G' = G \cup G^T$ is a $2d$-regular bidirectional topology. Consider both A and $g(A)$ as allgather schedules for bidirectional topology G'. Schedules A and $g(A)$ use disjoint sets of edges, because they use opposite directions. Thus, we can divide each shard into two halves. Let one half follow schedule A and the other half follow $g(A)$. Let such a schedule be A'.

It is trivial to see that $T_L(A)=T_L(A')$. As for $T_B(A)=T_B(A')$, it follows the fact that the total data size is halved for each of A and $g(A)$, but the bandwidth per edge is also halved due to the doubling of degree. Note that if A is BW-optimal, then A' is BW-optimal; however, A' is not necessarily Moore optimal if A is Moore optimal.

B Reduce-Scatter & Allgather

We use tuple $((v,C),(u,w),t)$ to denote that u sends v's chunk C to w at comm step t. Node v is the source and destination node of chunk C in allgather and reduce-scatter respectively. A communication schedule is thus a collection of tuples.

Definition 4 (Allgather)**.** *An algorithm* (G,A) *is an allgather algorithm if for arbitrary* $x \in S$ *and distinct* $u,v \in V_G$*, there exists a sequence in* A*:*

$$((v,C_1),(w_0,w_1),t_1),((v,C_2),(w_1,w_2),t_2),\ldots$$
$$((v,C_n),(w_{n-1},w_n),t_n),$$

where $w_0=v$*,* $w_n=u$*,* $t_1<t_2<\ldots<t_n$*, and* $x\in C_1\cap C_2\cap\cdots\cap C_n$*.*

This sequence serves to broadcast x from v to u. A reduce-scatter algorithm has the same definition except $w_0=u$, $w_n=v$. In reduce-scatter, we assume any chunk received by a node is immediately reduced with the node's local chunk.

In this paper, many of the techniques are discussed under allgather only. We will show that anything holds in either reduce-scatter or allgather has an equivalent version for the other collective operation. To do so, we use the concept of *transpose graph* from graph theory and define *reverse schedule*. We say a schedule A is *for* topology G if every $((v,C),(u,w),t)\in A$ satisfies $u,v,w\in V_G$ and $(u,w)\in E_G$.

Definition 5 (Reverse Schedule)**.** *Suppose* A *is a schedule for* G*. A reverse schedule* A^T *of* A *is a schedule for trans-*

pose graph G^T such that $((v,C),(u,w),t_{\max}-t+1) \in A^T$ iff $((v,C),(w,u),t) \in A$, where $t_{\max}$ is the max comm step in A.

It is trivial to see that $T_L(A) = T_L(A^T)$ and $T_B(A) = T_B(A^T)$. Note that $(u,w) \in E_{G^T}$ if and only if $(w,u) \in E_G$ by definition of transpose graph.

Theorem 1. *If A is a reduce-scatter/allgather schedule for G, then A^T is an allgather/reduce-scatter schedule for G^T.*

Theorem 1 has the following two corollaries:

Corollary 1.1. *Suppose $G \mapsto f(G)$ is a function to construct reduce-scatter/allgather schedule given graph G, then $G \mapsto f(G^T)^T$ is a function to construct allgather/reduce-scatter schedule given graph G.*

Corollary 1.2. *Suppose $(G,A) \mapsto (f(G), f(A))$ is a mapping within reduce-scatter/allgather algorithms, then $(G,A) \mapsto (f(G^T)^T, f(A^T)^T)$ is a mapping within allgather/reduce-scatter algorithms.*

For example, the line graph expansion in §5.1 can be seen as a mapping within allgather, and the BFB linear program (1) can be seen as a function to construct allgather schedule. Thus, Corollary 1.1 and 1.2 have shown that they both have equivalent versions in reduce-scatter.

In undirected topology, it is well-known that reduce-scatter and allgather are a pair of dual operations such that one can be transformed into another by reversing the communication in schedule [11]. It is similar for directed topology but with extra requirement and more complicated transformation. We define the following property for directed graphs:

Definition 6 (Reverse-Symmetry)**.** *A digraph G is reverse-symmetric if it is isomorphic to its own transpose graph G^T.*

In graph theory, there is a similar concept called *skew-symmetric graph*. Reverse-symmetry is a weaker condition than skew-symmetry.

We define a way to transform the schedule for G into a schedule for G^T based on graph isomorphism:

Definition 7 (Schedule Isomorphism)**.** *Suppose G and G' are isomorphic. Let $f : V_G \to V_{G'}$ be the graph isomorphism and A be a schedule for G, then $f(A)$ is a schedule for G' that $((f(v),C),(f(u),f(w)),t) \in f(A)$ iff $((v,C),(u,w),t) \in A$.*

Theorem 2. *Suppose G is reverse-symmetric. Let G^T be the transpose graph, and let $f : V_{G^T} \to V_G$ be the isomorphism from G^T to G. If (G,A) is a reduce-scatter/allgather algorithm, then $(G, f(A^T))$ is an allgather/reduce-scatter algorithm with $T_L(f(A^T)) = T_L(A)$ and $T_B(f(A^T)) = T_B(A)$.*

Theorem 2 establishes that given any reverse-symmetric topology, if we have either reduce-scatter or allgather, then we can construct both reduce-scatter and allgather. Since allreduce can be achieved by applying a reduce-scatter followed by an allgather, we only need one of reduce-scatter and allgather to construct a complete allreduce algorithm. Furthermore, if the reduce-scatter or allgather algorithm has runtime T, then the resulting allreduce algorithm has runtime $2T$.

Most of our base topologies are reverse-symmetric (Table 9). In addition, all of our expansion techniques also preserve reverse-symmetry. Thus, one can almost always use Theorem 2 to derive reduce-scatter and allreduce schedules from allgather schedule on our synthesized topologies. For non-reverse-symmetric topologies like generalized Kautz graph, one can apply Corollary 1.1 or 1.2 to construct reduce-scatter and allgather separately.

C Topology-Schedule Optimality

Because our cost model is only concerned with total-hop latency and BW runtime, the optimality of reduce-scatter/allgather algorithm is only related to total-hop latency optimality and BW optimality in this paper. Note that we also consider topology as a dimension that can be optimized, so optimality is discussed in the space of all topology-schedule combinations, i.e., *algorithms* by our definition.

C.1 Total-Hop Latency Optimality

Definition 8 (Total-Hop Latency Optimal)**.** *Given an N-node degree-d reduce-scatter/allgather algorithm (G,A), if any other N-node degree-d reduce-scatter/allgather algorithm (G',A') satisfies $T_L(A') \geq T_L(A)$, then (G,A) is total-hop latency optimal.*

Because in reduce-scatter/allgather, every node needs to send a shard of data to every other node, the number of comm steps is lower bounded by the graph diameter:

Theorem 3. *Every reduce-scatter/allgather algorithm (G,A) satisfies $T_L(A) \geq \alpha \cdot D(G)$, where $D(G)$ is the diameter of G.*

Because we can always construct a BFB schedule A for topology G with $T_L(A) = \alpha \cdot D(G)$, it follows the corollary:

Corollary 3.1. *An N-node degree-d reduce-scatter/allgather algorithm (G,A) is total-hop latency optimal if and only if $T_L(A) = \alpha \cdot D(G) = \alpha \cdot \min\{D(G') : |V_{G'}| = N, \deg(G') = d\}$.*

The minimum diameter of a directed graph given a number of vertices and degree is still an open question. One can check *degree/diameter problem* [50] for more information. However, as a close upper bound of number of vertices given degree and diameter, the *Moore bound* for digraph is sufficient to tell the total-hop latency optimality in most cases.

Definition 9 (Moore Bound)**.** *Let G be any degree-d digraph of diameter k. The Moore bound is an upper bound on the number of vertices in G:*

$$M_{d,k} = \sum_{i=0}^{k} d^i = \frac{d^{k+1}-1}{d-1}.$$

Definition 10 (Moore Optimal)**.** *Let (G,A) be an N-node degree-d reduce-scatter/allgather algorithm with $T_L(A) = k\alpha$, then (G,A) is Moore optimal if $N > M_{d,k-1}$.*

Because for any degree-d digraph G, $D(G) \geq k$ must be true as long as $|V_G| > M_{d,k-1}$, Moore optimality is a stronger condition than total-hop latency optimality. We define a function T_L^* such that $T_L^*(N,d)$ equals the Moore optimal total-hop latency of N-node degree-d reduce-scatter/allgather algorithms.

C.2 Bandwidth Optimality

Definition 11 (Bandwidth Optimal)**.** *Given an N-node degree-d reduce-scatter/allgather algorithm (G,A), if any other N-node degree-d reduce-scatter/allgather algorithm (G',A') satisfies $T_B(A') \geq T_B(A)$, then (G,A) is BW-optimal.*

In reduce-scatter/allgather, each node needs to send/receive at least $M \cdot \frac{N-1}{N}$ amount of data. Thus, the following holds:

Theorem 4. *$\frac{M}{B} \cdot \frac{N-1}{N}$ is a lower bound of $T_B(A)$ for any N-node reduce-scatter/allgather algorithm (G,A).*

Note that one can always construct a ring of degree d by sending d parallel edges from one node to the next node. The trivial ring reduce-scatter/allgather schedule has $\frac{M}{B} \cdot \frac{N-1}{N}$ BW runtime. Therefore, we have:

Corollary 4.1. *An N-node reduce-scatter/allgather algorithm (G,A) is BW-optimal if and only if $T_B(A) = \frac{M}{B} \cdot \frac{N-1}{N}$.*

We define a function T_B^* such that $T_B^*(N) = \frac{M}{B} \cdot \frac{N-1}{N}$ is the optimal BW runtime of N-node reduce-scatter/allgather algorithms. From Corollary 4.1, we have the following necessary and sufficient condition for BW optimality:

Theorem 5. *An allgather algorithm (G,A) is BW-optimal if and only if:*

1. *$\frac{1}{B/d}\sum_{((v,C),(u,w))\in A_t}|C| = T_B(A_t)$ for all $(u,v) \in E_G$ and $t \in \{1,\ldots,t_{\max}\}$. A_t is the subschedule of A at comm step t.*
2. *Pick any distinct $u,v \in V_G$. For each $x \in S$, there exists a unique $((v,C),(w,u),t) \in A$ such that $x \in C$.*

Condition 1 ensures that at each comm step, every link of topology G has equal workload, so no link finishes early and results in waste of bandwidth. Condition 2 ensures that no piece of data is received twice by some node, so no duplicated send exists.

C.3 Allreduce Optimality

In this paper, we construct an allreduce algorithm through a reduce-scatter followed by allgather. In such construction, the lower bound of allreduce algorithm is $2(T_L^*(N,d) + T_B^*(N))$. To compare this with the lower bound of any allreduce construction, in [58], the authors have proved that $2T_B^*(N)$ is indeed the lower bound of BW runtime of any allreduce algorithm. As for total-hop latency, a reduce-scatter followed by allgather has at least $2D(G)$ number of comm steps, so $2T_L^*(N,d)$ is the lower bound of total-hop latency. Although one can use all-to-all to construct an allreduce with number of comm steps equal to one diameter $D(G)$ (lower bound being $T_L^*(N,d)$ instead of $2T_L^*(N,d)$), the lower bound of BW runtime for all-to-all is $\frac{M}{B} \cdot (N-1) = N \cdot T_B^*(N)$, which is much worse than $2T_B^*(N)$.

There is also another way of constructing allreduce: reduce followed by broadcast. In such an approach, the number of comm steps can be twice the radius of G instead of twice the diameter. However, the Moore bound for graph diameter also applies to graph radius, so $2T_L^*(N,d)$ is still a lower bound of allreduce via reduce+broadcast. By Theorem 16, the total-hop latency optimal allreduce via reduce+broadcast is at most 2α lower than the total-hop latency of generalized Kautz graph can do with reduce-scatter plus allgather. Furthermore, reduce+broadcast is usually poor in BW performance.

C.4 Computational Cost

In this paper, we omit the computational cost of reduction operation in performance analysis. While this approach is commonly adopted in previous literature [10, 63, 65, 76], we give a formal reasoning why this approach is legitimate. It is not only because computational cost is generally orders of magnitude lower than network cost, but also because computational cost can be incorporated into network cost.

Assume a cost model where computation and network communication do not overlap at each node.[5] In particular, at each comm step of reduce-scatter, the computation to reduce chunks happens immediately after the node receives all chunks and before the node starts to send out chunks for the next comm step. We adopt notations from [11], where γ denotes the computational time cost per size of data. Like total-hop latency and BW runtime, we also let $T_C(A)$ be the total time spent on computation by schedule A. As argued in [11], a lower bound of computational cost is $T_C \geq M \cdot \gamma \cdot \frac{N-1}{N}$ for both reduce-scatter and allreduce, which is identical to the BW optimality of reduce-scatter and half of that of allreduce. The following theorem shows that BW runtime of a schedule can act as an upper bound for the computational time.

Theorem 6. *Given a reduce-scatter algorithm (G,A), suppose $T_B(A) = \frac{M}{B} \cdot y$, then $T_C(A) \leq M \cdot \gamma \cdot y$.*

The rationale behind Theorem 6 is that the amount of computation for any node at a given comm step equals the amount of data the node receives during that comm step. Thus, **as we balance network transmission, it naturally leads to a more balanced computation.** With Theorem 6, if the BW runtime of some allreduce schedule A is $T_B(A) = 2\frac{M}{B} \cdot y$, then $T_B(A) + T_C(A) \leq M \cdot (\frac{2}{B} + \gamma) \cdot y$. We can thus simply define $B' = (\frac{1}{B} + \frac{\gamma}{2})^{-1}$, and then $2\frac{M}{B'} \cdot y$ can represent the sum of BW runtime and computational runtime altogether. The value of y is all that matters. The following corollary shows that if an algorithm is BW-optimal, then such representation is exact.

Corollary 6.1. *If allreduce algorithm (G,A) is BW-optimal, i.e., $T_B(A) = 2\frac{M}{B} \cdot \frac{N-1}{N}$, then $T_C(A) = M \cdot \gamma \cdot \frac{N-1}{N}$ and $T_B(A) + T_C(A) = 2M \cdot (\frac{1}{B} + \frac{\gamma}{2}) \cdot \frac{N-1}{N}$.*

When profiling a testbed, one can simply derive the value of $\frac{1}{B} + \frac{\gamma}{2}$ using BW-optimal topologies and use it as the new $1/B$ to apply the results of this paper. While it is still possible for two schedules with the same BW runtime to have different computational runtimes, such difference is bounded by the aforementioned theorems and orders of magnitude smaller than BW runtime.

[5] Otherwise, the computational cost would be even more negligible.

D Optimality of Expansion Techniques

In this section, we provide formal definitions and detailed performance analysis of expansion techniques.

D.1 Line Graph Expansion

Definition 12 (Line Graph)**.** *Given a directed graph (or digraph) G, each edge $(u,v) \in E_G$ corresponds to a vertex uv in the line graph $L(G)$. For every uv, vw pair in $V_{L(G)}$, there exists an edge $(uv, vw) \in E_{L(G)}$.*

In the case of multiedges between u and v, the line graph also contains multiple vertex uv.

Definition 1 (Schedule of Line Graph)**.** *Given an allgather schedule A_G for topology G, let $A_{L(G)}$ be the schedule for line graph $L(G)$ containing:*

1. *$((v'v, S), (v'v, vu), 1)$ for each edge $(v'v, vu) \in E_{L(G)}$ with $v'v \neq vu$.* **[Insert the 1st comm step in $A_{L(G)}$.]**
2. *$((v'v, C), (uw, ww'), t+1)$ for each $((v,C),(u,w),t) \in A_G$ and $v'v \neq ww'$.* **[Adapt A_G to form $A_{L(G)}$.]**

The following theorem gives the performance of the expanded schedule:

Theorem 7. *Given a d-regular topology G, if (G, A_G) is an N-node allgather algorithm, then $(L(G), A_{L(G)})$ is a dN-node allgather algorithm satisfying:*

$$T_L(A_{L(G)}) = T_L(A_G) + \alpha, \tag{4}$$

$$T_B(A_{L(G)}) \leq T_B(A_G) + \frac{M}{B} \cdot \frac{1}{N}. \tag{5}$$

From Theorem 7, one can see that the performance of the expanded schedule depends on that of the base schedule. Note that T_B also depends on M and B. For simplicity, we write $T_B(A_G)$ instead of $T_B(A_G, M, B)$ when there is no ambiguity. Theorem 7 makes an implicit assumption that $T_B(A_G, M, B) = \tau(M/B)$ for some constant τ. This assumption, suggesting that T_B scales linearly with data size and inversely with bandwidth, should hold for any reasonably designed schedule.

Consequently, if we apply line graph expansion n times, the performance of the expanded schedule is:

Corollary 7.1. *Given a d-regular topology G, if (G, A_G) is an N-node allgather algorithm with $T_B(A_G, M, B) = \tau(M/B)$ for some constant τ, then $(L^n(G), A_{L^n(G)})$ is a d^nN-node allgather algorithm satisfying:*

$$T_L(A_{L^n(G)}) = T_L(A_G) + n\alpha, \tag{6}$$

$$T_B(A_{L^n(G)}) \leq T_B(A_G) + \frac{M}{B} \cdot \frac{d}{d-1}\left(\frac{1}{N} - \frac{1}{d^nN}\right). \tag{7}$$

In terms of the optimality of line graph expansion:

Theorem 8. *$(L^n(G), A_{L^n(G)})$ is Moore optimal if and only if (G, A_G) is Moore optimal.*

Theorem 9. *If (G, A_G) is BW-optimal with N nodes, then $T_B(A_{L^n(G)})/T_B^*(d^nN) \leq 1 + [(d-1)N]^{-1}$ for all n.*

As mentioned in the main text, by Theorem 9, the key metric for the quality of base graph is how large it is while achieving both Moore and BW optimality. Currently, our largest such base graph that works for any even degree is Hamming graph $H(2, 1+d/2)$, which has $(1+d/2)^2 = \Theta(d^2)$ number of nodes. The corresponding line graph expanded topology is always Moore optimal and at most $O(1/d^3)$ away from BW optimality by Theorem 9.

Line graph expansion is closely related to BFB schedule for two reasons: (1) most of our base topologies like complete bipartite graph and Hamming graph use BFB schedule as the base schedule, and (2) the line graph expansion of BFB schedule is still a BFB schedule. To see the performance bound in Theorem 7 is tight, we have the following results in the context of BFB schedule:

Theorem 10. *Let A_G be a BFB allgather schedule for d-regular topology G with $|N^+(u)| > 1$ for all $u \in V_G$, then the expanded schedule $A_{L(G)}$ is a BFB allgather schedule for $L(G)$. In particular, if A_G is the optimal BFB schedule for G, then $A_{L(G)}$ is the optimal BFB schedule for $L(G)$ satisfying:*

$$T_B(A_{L(G)}) = T_B(A_G) + \frac{M}{B} \cdot \frac{1}{N}. \tag{8}$$

Corollary 10.1. *Let A_G be a BFB allgather schedule for d-regular topology G with $|N^+(u)| > 1$ for all $u \in V_G$, then the expanded schedule $A_{L^n(G)}$ is a BFB allgather schedule for $L^n(G)$. In particular, if A_G is the optimal BFB schedule for G, then $A_{L^n(G)}$ is the optimal BFB schedule for $L^n(G)$ satisfying:*

$$T_B(A_{L^n(G)}) = T_B(A_G) + \frac{M}{B} \cdot \frac{d}{d-1}\left(\frac{1}{N} - \frac{1}{d^nN}\right).$$

D.2 Degree Expansion

Definition 13 (Degree Expanded Topology)**.** *Given an N-node d-regular topology G without self-loops, construct the degree expanded nN-node nd-regular topology $G*n$:*

1. *For each vertex $v \in V_G$, add $v_1, \ldots, v_n$ to V_{G*n},*
2. *For each edge $(u,v) \in E_G$, add (u_i, v_j) to E_{G*n} for all i, j including $i = j$.*

Definition 2 (Degree Expanded Schedule)**.** *Given an allgather schedule A_G for G, construct A_{G*n} for $G*n$:*

1. *For all i,j including $i=j$ and for each $((v,C),(u,w),t) \in A_G$, add $((v_j, C), (u_j, w_i), t)$ to A_{G*n};*
2. *Divide shard S into equal-sized chunks $C_1, \ldots, C_{nd}$. Given $u_i, u_j \in V_{G*n}$ with $i \neq j$, add $((u_i, C_\alpha), (v_\alpha, u_j), t_{\max}+1)$ to A_{G*n} for each $(v_1, u_j), \ldots, (v_{nd}, u_j) \in E_{G*n}$, where $t_{\max}$ is the max comm step in A_G.*

Theorem 11. *Given a d-regular topology G without self loops, if (G, A_G) is an N-node allgather algorithm with $T_B(A_G, M, B) = \tau(M/B)$ for some constant τ, then $(G*n, A_{G*n})$ is an nN-node allgather algorithm satisfying:*

$$T_L(A_{G*n}) = T_L(A_G) + \alpha, \tag{9}$$

$$T_B(A_{G*n}) = T_B(A_G) + \frac{M}{B} \cdot \frac{n-1}{nN}. \tag{10}$$

Corollary 11.1. *If (G, A_G) is BW-optimal and $T_B(A_G, M, B) = \tau(M/B)$ for some τ, then $(G*n, A_{G*n})$ is BW-optimal.*

Degree expansion preserves BW optimality. As for total-hop latency of degree expanded topology, observe that $T_L^*(N,d) = \Theta(\log_d N)$ and $\log_{nd} nN < \log_d N$, so T_L^* decreases as we apply degree expansion. Since T_L increases in degree expansion, Moore optimality is not preserved.

D.3 Cartesian Product Expansion

Definition 3 (Cartesian Product)**.** *The Cartesian product digraph $G_1 \square G_2$ of digraphs G_1 and G_2 has vertex set $V_{G_1} \times V_{G_2}$ with vertex $\mathbf{u} = (u_1, u_2)$ connected to $\mathbf{v} = (v_1, v_2)$ iff either $(u_1, v_1) \in E_{G_1}$ and $u_2 = v_2$; or $u_1 = v_1$ and $(u_2, v_2) \in E_{G_2}$.*

Definition 3 generalizes to Cartesian product of multiple digraphs: $G_1 \square G_2 \square G_3 = (G_1 \square G_2) \square G_3$. The Cartesian product of n identical digraphs is denoted as Cartesian power $G^{\square n}$.

Definition 14 (Schedule of Cartesian Power)**.** *Given an allgather schedule A_G for topology G and $n \in \mathbb{N}$, construct the schedule $A_{G^{\square n}}$ for $G^{\square n}$:*

1. *Construct the schedule $A^{(1)}$ as follows:*
2. *For $j = 1, \ldots, n$, for each $((w,C),(u,v),t) \in A_G$, add*
   $$(((\mathbf{x}, w, \mathbf{z}), C), ((\mathbf{y}, u, \mathbf{z}), (\mathbf{y}, v, \mathbf{z})), t + (j-1)t_{\max})$$
   to $A^{(1)}$ for all $\mathbf{x}, \mathbf{y} \in V_G^{j-1}$ and $\mathbf{z} \in V_G^{n-j}$. $t_{\max}$ is the max comm step in A_G.
3. *Similarly, construct $A^{(i)}$ for $i = 2, \ldots, n$ that each vertex $\mathbf{v}$ in $A^{(1)}$ is shifted by $i-1$ to $(\mathbf{v}[n-i+2:n], \mathbf{v}[1:n-i+1])$.*
4. *Divide each shard into n equal-sized subshards. Construct schedule $A_{G^{\square n}}$ such that $A^{(i)}$ performs allgather over the i-th subshards of all nodes.*

Theorem 12. *Given a d-regular topology G, if (G, A_G) is an N-node allgather algorithm with $T_B(A_G, M, B) = \tau(M/B)$ for some constant τ, then $G^{\square n}$ is an nd-regular topology, and $(G^{\square n}, A_{G^{\square n}})$ is an N^n-node allgather algorithm satisfying:*

$$T_L(A_{G^{\square n}}) = n \cdot T_L(A_G), \tag{11}$$

$$T_B(A_{G^{\square n}}) = T_B(A_G) \cdot \frac{N}{N-1} \cdot \frac{N^n - 1}{N^n}. \tag{12}$$

We then have the following corollary:

Corollary 12.1. *If (G, A_G) is BW-optimal and $T_B(A_G, M, B) = \tau(M/B)$ for some τ, then $(G^{\square n}, A_{G^{\square n}})$ is BW-optimal.*

Like degree expansion, Cartesian power expansion does not preserve Moore optimality.

We use BFB schedule generation when dealing with Cartesian product of distinct topologies:

Theorem 13. *Let $G_1, G_2, \ldots, G_n$ be topologies that*

1. *$G_1, \ldots, G_n$ are nontrivial simple digraphs;*
2. *Every G_i has BW-optimal BFB allgather schedule.*

Then, the optimal BFB allgather schedule, i.e. the schedule generated by BFB LP (1), for $G_1 \square \ldots \square G_n$ is also BW-optimal. The total-hop latency of the schedule equals $\alpha \cdot D(G_1 \square \ldots \square G_n) = \alpha \cdot \sum_i D(G_i)$.

The BFB schedule generation can also be used when individual topologies do not have BW-optimal BFB schedules; however, in such a case, we do not have performance bound for the schedule of the Cartesian product.

E BFB Schedule Generation

The LP formulation for u_2 in Figure 5 is:

$$\begin{array}{lr}
\text{minimize} & U_{u_2,t} \\
\text{subject to} & x_{v_1,(w_1,u_2),t} \leq U_{u_2,t}, \\
 & x_{v_1,(w_2,u_2),t} + x_{v_2,(w_2,u_2),t} \leq U_{u_2,t}, \\
 & x_{v_2,(w_3,u_2),t} \leq U_{u_2,t}, \\
 & x_{v_1,(w_1,u_2),t} + x_{v_1,(w_2,u_2),t} = 1, \\
 & x_{v_2,(w_2,u_2),t} + x_{v_2,(w_3,u_2),t} = 1, \\
 & 0 \leq x_{v,(w,u_2),t} \leq 1. \qquad \forall v, w
\end{array}$$

Definition 15 (BFB schedule)**.** *An allgather schedule A for G is a BFB schedule if A satisfies: $((v,C),(w,u),t) \in A$ only if $d(v,u) = d(v,w) + 1 = t$.*

Theorem 14. *A schedule A for G is a BFB allgather schedule if and only if the following are satisfied:*

1. *If $((v,C),(w,u),t) \in A$, then $d(v,u) = d(v,w) + 1 = t$;*
2. *For any distinct $u, v \in V_G$, the collection of chunks $\mathcal{C}_v = \{C \mid ((v,C),(w,u),t) \in A\}$ satisfies $S = \bigcup_{C \in \mathcal{C}_v} C$.*

Condition 1 ensures the schedule follows the breadth-first broadcast order. Condition 2 ensures every node receives the entire shard from every other node and thus a valid allgather.

E.1 Optimality

Theorem 15. *If A is a BFB schedule for G, then the total-hop latency $T_L(A) = \alpha \cdot D(G)$.*

There may exist many BFB schedules for a given topology G. They all have the same T_L but may have different T_Bs. Thus, the optimal BFB schedule is the one with the lowest T_B. Since every BFB schedule can be expressed as a solution to linear program (1), we have the following result:

Theorem 16. *Given any topology G, linear program (1) gives the optimal BFB schedule of G.*

An important implication of Theorem 16 is that **if we can show a BW-optimal BFB schedule exists for a topology G, then linear program (1) is guaranteed to generate one.** This has become an important tool for us to prove that BFB schedule generation can always generate BW-optimal schedules for some families of topologies (see §F). For the rest of this section, we show conditions that, if met by a topology, ensure it has a BW-optimal BFB schedule.

The following theorem shows the necessary and sufficient conditions for a BFB allgather schedule to be BW-optimal:

Theorem 17. *Suppose (G,A) is a BFB allgather schedule. (G,A) is BW-optimal if and only if:*

1. *There exists a sequence $N_1^-, N_2^-, \ldots, N_{D(G)}^- \in \mathbb{N}$ such that for any $x \in \mathbb{N}$ and $u \in V_G$, $|N_x^-(u)| = N_x^-$.*
2. *For any $(w,u) \in E_G$, $\sum_{((v,C),(w,u)) \in A_t} |C| = \frac{M}{N} |N_t^-(u)|/d = \frac{M}{N} N_t^-/d$.*

We assume G is d-regular. Condition 1 and 2 together ensure that at each comm step, all links have perfectly balanced workloads. In Theorem 13, we have already proven that *a Cartesian product graph has BW-optimal BFB schedule if it is the product of graphs that each have a BW-optimal BFB schedule.* Here, Theorem 17 also leads to the following sufficient condition for a bidirectional topology to have a BW-optimal BFB schedule:

Theorem 18. *There exists a BW-optimal BFB schedule for undirected graph G if for every distance x, two of the following constants exist:*

1. $N_x = |N_x(u)|$ *for any* $u \in V_G$;
2. $a_x = |N_x(u) \cap N_{x-1}(w)|$ *for any* $u \in V_G$ *and* $w \in N(u)$;
3. $b_x = |N(u) \cap N_{x-1}(v)|$ *for any* $u \in V_G$ *and* $v \in N_x(u)$.

Moreover, if two of N_x, a_x, b_x *exist, then the third one must also exist with* $N_x = da_x/b_x$.

Note that in undirected graphs, we have $N_x^+(u) = N_x^-(u) = N_x(u)$. To understand these constants, N_x is the number of data shards u needs to receive at comm step x; a_x is the number of data shards that can be transmitted by each link (w,u) at comm step x; b_x is the number of link (w,u)s that each data shard can use to transmit the data to u at comm step x. These three constants collectively ensure that links are perfectly balanced with each link transmitting $\frac{M}{N}N_x/d = \frac{M}{N}a_x/b_x$ amount of data at comm step x.

Now, we give a *necessary and sufficient condition* for any topology to have a BW-optimal BFB schedule. The condition is derived based on the observation that the BFB optimization problem is equivalent to a job scheduling problem. In each comm step t, for each node u, we have a set of jobs $\{j_1, j_2, \ldots, j_m\}$ (data from the source nodes $v \in N_t^-(u)$) and a set of processors $\{p_1, p_2, \ldots, p_d\}$ (links from in-neighbors $w \in N^-(u)$). There exists a map f from any job to a set of processors that j_i can only be scheduled to the processors in $f(j_i)$ (in-neighbor ws satisfying $d(v,u) = d(v,w) + 1 = t$). Assuming jobs can be arbitrarily divided into subjobs for parallel execution on multiple processors, the problem is how to schedule these jobs to processors so that workloads are balanced across all processors. We have the following result:

Theorem 19. *The workloads can be balanced if and only if there exists no subset* $J \subseteq \{j_1, j_2, \ldots, j_m\}$ *such that*

$$\frac{|J|}{|\bigcup_{j \in J} f(j)|} > \frac{m}{d}.$$

Note that there is an independent scheduling problem for each comm step t and node u. Therefore, topology G has a BW-optimal BFB schedule if and only if:

1. At each comm step t, $|N_t^-(u)|$ is the same for all $u \in V_G$.
2. The scheduling problem w.r.t. each t and u satisfies the condition in Theorem 19.

E.2 Discrete Chunked BFB Schedule

The BFB LP (1) makes an assumption that shards can be divided arbitrarily and infinitesimally. However, to compile the schedule into an executable form, one may need a *discrete chunked schedule*, where each shard is divided into a fixed number of equal-sized chunks. In practice, $x_{v,(w,u),t}$s are usually solved to be rational numbers. We can divide each shard into a number of chunks equal to the LCM of $x_{v,(w,u),t}$s' denominators so that each $x_{v,(w,u),t}$ represents some integer number of chunks. This approach has worked for us in evaluations. However, there exists the case where each shard of the data can only be divided into P equal chunks (i.e., the whole data M can only be divided into PN equal chunks). In such a case, we show that *we can approximate the optimal discrete chunked BFB schedule in polynomial time*.

Consider the following integer program given u,t:

$$\begin{array}{lll} \text{min} & W_{u,t} & \\ \text{s.t.} & \sum_v y_{v,(w,u),t} \le W_{u,t}, & \forall w \in N^-(u) \\ & \sum_w y_{v,(w,u),t} = P, & \forall v \in N_t^-(u) \\ & y_{v,(w,u),t} \in \{0,1,\ldots,P\}, & \forall w, v. \end{array} \tag{13}$$

Compared with (1), one can easily see that the optimal solution of (13) gives the optimal BFB allgather schedule when each shard of the data can only be divided into P chunks. One can also easily solve the LP relaxation of (13) in polynomial time. Let T_B^{OPT} be the optimal BW runtime of the schedule obtained by directly solving integer program (13). Suppose the LP relaxation gives a schedule with BW runtime T_B^{LP}, then it holds that $T_B^{\text{LP}} \le T_B^{\text{OPT}}$.

Let $y_{v,(w,u),t}^{\text{LP}}$s be the solution to the LP relaxation of (13). We can obtain an integer solution $y_{v,(w,u),t}$s of (13) by rounding $y_{v,(w,u),t}^{\text{LP}}$s up or down to integers. For each v, we have

$$\sum_w \left\lfloor y_{v,(w,u),t}^{\text{LP}} \right\rfloor \le P \le \sum_w \left\lceil y_{v,(w,u),t}^{\text{LP}} \right\rceil.$$

Thus, it is trivial to round $y_{v,(w,u),t}^{\text{LP}}$s to integer $y_{v,(w,u),t}$s that $\sum_w y_{v,(w,u),t} = P$ and $y_{v,(w,u),t} < y_{v,(w,u),t}^{\text{LP}} + 1$. We give the following approximation bound for the resulting schedule:

Theorem 20. *Rounding LP gives a solution with BW runtime* $T_B \le T_B^{OPT} + \frac{M}{B} \cdot \frac{d(d^{D(G)}-1)}{(d-1)PN}$. *In addition, if topology G is Moore optimal, then* $T_B \le T_B^{OPT} + \frac{M}{B} \cdot \frac{d}{P}$.

The cost $\frac{M}{B} \cdot \frac{d}{P}$ is negligible since P can easily be hundreds or even thousands while degree d is usually a small integer.

E.3 Heterogeneous BFB Schedule

The BFB LP (1) assumes a homogeneous network. It turns out that with little modification, (1) can become an LP for

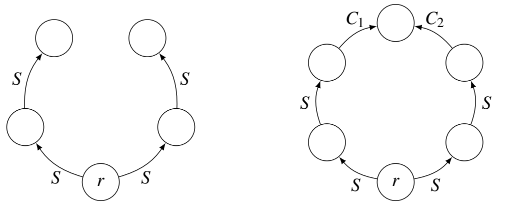


Figure 17: The broadcast paths of ring BFB allgather schedule. The left and right figures respectively show the broadcast patterns for odd- and even-sized bidirectional rings. Edges of the rings are omitted. C_1 and C_2 are two halves of shard S.

heterogeneous network too:

$$\begin{aligned}
\min \quad & U_{u,t} & & \\
\text{s.t.} \quad & \alpha_{w,u} + \tfrac{M/N}{B_{w,u}} \sum_{v} x_{v,(w,u),t} \le U_{u,t}, & & \forall w \in N^-(u) \\
& \sum_{w} x_{v,(w,u),t} = 1, & & \forall v \in N_t^-(u) \\
& 0 \le x_{v,(w,u),t} \le 1, & & \forall w, v.
\end{aligned} \tag{14}$$

$\alpha_{w,u}$ and $B_{w,u}$ are the hop latency and bandwidth of link (w,u). In some cases, the α of some link (w,u) is so high that $\alpha_{w,u}$ alone dominates $U_{u,t}$ in (14) even though $\sum_v x_{v,(w,u),t} = 0$. This is problematic because one should not pay $\alpha_{w,u}$ if link (w,u) is not used. However, such a scenario can be easily detected after solving LP (14). One can avoid the issue by simply removing link (w,u) and solving the LP again.

F Generative Topologies

In this section, we introduce several topologies for which applying BFB schedule generation yields high-performance communication schedules.

F.1 Bidirectional Ring

Ring is the most common topology for allreduce. The traditional schedule on ring is to make each shard go a full circle to do reduce-scatter/allgather. In a bidirectional ring, one can simply make half the shard go clockwise and the other half go counterclockwise to utilize both directions of the links. Such a reduce-scatter/allgather schedule is BW-optimal but poor in total-hop latency with $T_L = (N-1)\alpha$. With BFB schedule generation, we discovered a new ring reduce-scatter/allgather schedule that achieves half the total-hop latency ($T_L = \lfloor N/2 \rfloor \alpha$) while maintaining BW optimality. From each node, the BFB allgather schedule broadcasts the *entire* shard clockwise and counterclockwise in parallel. Thus, each direction only needs to go half a circle instead of a full circle. If N is even, then the farthest node across the ring receives each half of the shard from each of its two neighbors in the end. Figure 17 shows examples in odd- and even-sized rings respectively.

F.2 Generalized Kautz Graph

Generalized Kautz graph [5, 25] is a low-T_L unidirectional topology that can be constructed for every N and d.

Definition 16 (Generalized Kautz Graph)**.** *The $\Pi_{d,m}$ digraph has the set of integers modulo m as vertex set. Its arc set A is defined as follows:*

$$A = \{(x,y) \mid y \equiv -dx - a, 1 \le a \le d\}.$$

If $m = d^{n+1} + d^n$, then $\Pi_{d,m} = K(d,n)$, where $K(d,n)$ is the Kautz graph $L^n(K_{d+1})$.

We apply BFB schedule generation to generalized Kautz graph. The resulting schedule is not always Moore optimal, but the following theorem shows that it is at most one α away from Moore optimality, i.e., $T_L \le T_L^*(N,d) + \alpha$:

Theorem 21. *Suppose $D(\Pi_{d,m}) = k$, then $m > M_{d,k-2}$.*

Remember Moore optimality is stricter than total-hop latency optimality, so it is possible that generalized Kautz graph is total-hop latency optimal. The special case, Kautz graph $K(d,n)$, is always Moore optimal and is, in fact, the largest known digraph in *degree/diamter problem* for any degree $d > 2$ [50].

As for BW performance, from Figure 18, one can see that generalized Kautz graph is also close to BW optimality, especially at higher degrees.

F.3 Distance-Regular Graph

In graph theory, distance-regular graphs are a family of highly symmetric undirected graphs. We can show that there exists a BW-optimal BFB schedule for any distance-regular graph, and thus LP (1) can always generate one. We borrow the following definition from [2]:

Definition 17 (Distance-Regular Graph)**.** *A connected graph G is distance-regular if for any vertices $x, y \in V_G$ and integers i, j, the number of vertices at distance i from x and distance j from y depends only on i, j and $d(x,y)$.*

In other words, there exists a constant $s^h_{i,j}$ for every h, i, j such that $s^h_{i,j} = |N_i(x) \cap N_j(y)|$ whenever $x, y \in V_G$ satisfy $d(x,y) = h$. Thus, we can apply Theorem 18 with $N_x = s^0_{x,x}$, $a_x = s^1_{x,x-1}$, and $b_x = s^x_{1,x-1}$.

The significance of distance-regular graph is not only about BW optimality. Many of distance-regular graphs have low diameters, so their schedules are not only BW-optimal but also close to, and in some cases exactly, Moore optimal. Table 8 gives examples of distance-regular graphs at $d = 4$. In addition, many of the base graphs mentioned in this paper are also distance-regular like complete bipartite graphs (Figure 1) and Hamming graphs. One can refer to [16] for a repository of distance-regular graphs.

F.4 Circulant Graph

Circulant graph is a well-studied topology in both graph theory and network design. Many popular network topologies like shifted ring, chordal ring, and loop network are either subcategories of or closely related to circulant graphs. The definition of circulant graph is as follows:

Definition 18. *The circulant graph $C(n, \{a_1, \ldots, a_k\})$ is a bidirectional graph with vertex set $\{0, 1, \ldots, n-1\}$ and each node i is adjacent to nodes $i \pm a_1, \ldots, i \pm a_k \pmod{n}$.*

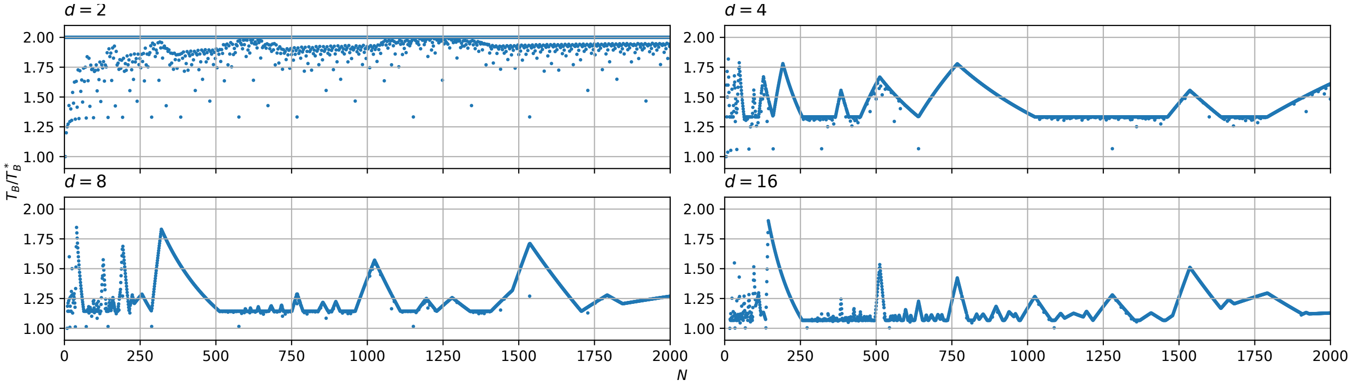


Figure 18: T_B/T_B^* of generalized Kautz graph $\Pi_{d,N}$ up to $N = 2000$. As shown, the BW runtime of $\Pi_{d,N}$ is less than or equal to $2T_B^*$ at all times for $d = 2,4,8,16$. In particular, the higher the degree is, the closer T_B is to optimal. As for total-hop latency, Theorem 21 shows that $T_L \leq T_L^*(N,d) + \alpha$.

Graph Name	N	T_L	T_L^*	$T_L - T_L^*$	T_L^{**}	$T_L - T_L^{**}$
Octahedron J(4,2)	6	2	2	0	2	0
Paley graph P9≅H(2,3)	9	2	2	0	2	0
K5,5-I	10	3	2	1	2	1
Distance-3 graph of Heawood graph	14	3	2	1	2	1
Line graph of Petersen graph	15	3	2	1	2	1
4-cube Q4≅H(4,2)	16	4	2	2	2	2
Line graph of Heawood graph	21	3	2	1	3	0
Incidence graph of PG(2,3)	26	3	3	0	3	0
Incidence graph of AG(2,4) minus a parallel class	32	4	3	1	3	1
Odd graph O4	35	3	3	0	3	0
Line graph of Tutte's 8-cage	45	4	3	1	3	1
Doubled Odd Graph D(O4)	70	7	3	4	4	3
Incidence graph of GQ(3,3)	80	4	3	1	4	0
Line graph of Tutte's 12-cage	189	6	4	2	5	1
Incidence graph of GH(3,3)	728	6	5	1	6	0

Table 8: Examples of distance-regular graphs at $d = 4$ [16]. T_L^{**} is the bidirectional Moore optimality.

Note that in this paper, we only consider connected circulant graphs, and $C(n,\{a_1,\dots,a_k\})$ is connected if and only if $\gcd(n,a_1,\dots,a_k) = 1$ [46, 51]. It is easy to see that $C(n,\{a_1,\dots,a_k\})$ is an n-node $2k$-regular topology.

We have found that the BFB schedule generation seems to give BW-optimal schedules for all circulant graphs. In particular, we have the following conjecture:

Conjecture 1. *For any circulant graph $C(n,\{a_1,\dots,a_k\})$, there exists a BW-optimal BFB schedule.*

While we leave a complete proof or disproof of this conjecture for future work, we have proved the conjecture holds when $k = 2$, which corresponds to the graph having degree 4.

Circulant graph revolutionized our Pareto frontier of topologies since it can be constructed for every N and even value d. It can provide a BW-optimal topology if our expansion techniques fail to produce one at some N and d. Since all circulant graphs seem to be BW-optimal, the question is what choices of $a_1,\dots,a_k$ result in minimum total-hop latency, or equivalently, minimum diameter for a given n and k. While this remains largely an open question in graph theory [51], the case of $k = 2$ has been solved in [7]:

Theorem 22. *Given $n > 6$ and $m = \lceil(-1+\sqrt{2n-1})/2\rceil$, circulant graph $C(n,\{m,m+1\})$ has a diameter equal to m, which is the minimum diameter over all circulant graphs $C(n,\{a_1,a_2\})$.*

We can certainly use multiedge to apply this construction for any even degree that is ≥ 4. The resulting topology has $\Theta(\sqrt{N})$ diameter, which is a significant improvement in terms of total-hop latency when BW optimality is required. Previously, the only topology that is known to be BW-optimal for any N and d is ring, which has $\Theta(N)$ diameter.

G Proofs

Theorem 7. *Given a d-regular topology G, if (G,A_G) is an N-node allgather algorithm, then $(L(G),A_{L(G)})$ is a dN-node allgather algorithm satisfying:*

$$T_L(A_{L(G)}) = T_L(A_G) + \alpha, \tag{4}$$

$$T_B(A_{L(G)}) \leq T_B(A_G) + \frac{M}{B} \cdot \frac{1}{N}. \tag{5}$$

Proof. Let $v'v, uw$ be arbitrary two distinct vertices in $L(G)$. We want to show there exists a sequence in $A_{L(G)}$ going from $v'v$ to uw like in Definition 4 for any $x \in S$. If $u = v$, then $((v'v,S),(v'v,uw),1)$ at the first comm step suffices. If $u \neq v$, because A_G is allgather, there exists a sequence in A_G:

$$((v,C_1),(v,w_1),t_1),((v,C_2),(w_1,w_2),t_2),\dots \\ ((v,C_n),(w_{n-1},u),t_n),$$

where $t_1 < t_2 < \cdots < t_n$ and $x \in C_1 \cap C_2 \cap \cdots \cap C_n$. Thus, by Definition 1, there exists a sequence in $A_{L(G)}$:

$$((v'v,S),(v'v,vw_1),1),((v'v,C_1),(vw_1,w_1w_2),t_1+1),\dots \\ ((v'v,C_n),(w_{n-1}u,uw),t_n+1),$$

as desired. The new algorithm $(L(G),A_{L(G)})$ has dM total data length, because the number of nodes has grown d-fold while the size of a shard remains the same.

As for $T_L(A_{L(G)})$ and $T_B(A_{L(G)})$, equality (4) trivially follows the Definition 1. Let $[A_{L(G)}]_t$ and $[A_G]_t$ be the subschedules of $A_{L(G)}$ and A_G at comm step t. Given $v \in V_G$, because G is d-regular, we have $|\{v'v \mid v'v \in V_{L(G)}\}| = |\{(v',v) \mid (v',v) \in E_G\}| = d$. Given any edge (uw,ww') and t, there are at most d number of $((v'v,C),(uw,ww'),t+1) \in A_{L(G)}$ for each

$((v,C),(u,w),t) \in A_G$ by Definition 1. Thus, given (uw,ww'),

$$\sum_{((v'v,C),(uw,ww')) \in [A_{L(G)}]_{t+1}} |C| \leq \sum_{((v,C),(u,w)) \in [A_G]_t} d \cdot |C|.$$

It follows that $T_B([A_{L(G)}]_{t+1}, dM, B) \leq d \cdot T_B([A_G]_t, M, B)$ and hence $\sum_{t=2}^{t_{\max}+1} T_B([A_{L(G)}]_t, dM, B) \leq d \cdot T_B(A_G, M, B)$. For the first comm step, we have

$$T_B([A_{L(G)}]_1, dM, B) = \frac{|S|}{B/d} = \frac{M/N}{B/d}.$$

Assuming $T_B(A_G,M,B) = \tau(M/B)$ for some constant τ, we have $d \cdot T_B(A_G,M,B) = T_B(A_G,dM,B)$. It follows that

$$T_B(A_{L(G)}, dM, B) = \sum_{t=1}^{t_{\max}+1} T_B([A_{L(G)}]_t, dM, B)$$
$$\leq \frac{M/N}{B/d} + d \cdot T_B(A_G,M,B) = T_B(A_G,dM,B) + \frac{dM}{B} \cdot \frac{1}{N}.$$

Replacing dM by M gives (5) as desired. □

Corollary 7.1. *Given a d-regular topology G, if (G,A_G) is an N-node allgather algorithm with $T_B(A_G,M,B) = \tau(M/B)$ for some constant τ, then $(L^n(G), A_{L^n(G)})$ is a d^nN-node allgather algorithm satisfying:*

$$T_L(A_{L^n(G)}) = T_L(A_G) + n\alpha, \quad (6)$$
$$T_B(A_{L^n(G)}) \leq T_B(A_G) + \frac{M}{B} \cdot \frac{d}{d-1} \left(\frac{1}{N} - \frac{1}{d^nN} \right). \quad (7)$$

Theorem 9. *If (G,A_G) is BW-optimal with N nodes, then $T_B(A_{L^n(G)})/T_B^*(d^nN) \leq 1 + [(d-1)N]^{-1}$ for all n.*

Proof. If (G,A_G) is BW-optimal, then $T_B(A_G) = \frac{M}{B} \cdot \frac{N-1}{N}$ and

$$T_B(A_{L^n(G)}) \leq \frac{M}{B} \left[1 + \frac{1}{d-1} \left(\frac{1}{N} - \frac{d}{d^nN} \right) \right]. \quad (15)$$

It is trivial to see that $(15)/T_B^*(d^nN) \nearrow 1 + [(d-1)N]^{-1}$ as $n \to \infty$. □

Theorem 1. *If A is a reduce-scatter/allgather schedule for G, then A^T is an allgather/reduce-scatter schedule for G^T.*

Proof. Suppose (G,A) is a reduce-scatter algorithm. For arbitrary $x \in S$ and distinct $u,v \in V_G$, there exists a sequence of tuples in A:

$$((v,C_1),(u,w_1),t_1),((v,C_2),(w_1,w_2),t_2),\ldots$$
$$((v,C_n),(w_{n-1},v),t_n),$$

where $t_1 < t_2 < \cdots < t_n$ and $x \in C_1 \cap C_2 \cap \cdots \cap C_n$. It follows that there exists a sequence of tuples in A^T:

$$((v,C_n),(v,w_{n-1}),t'_n),\ldots$$
$$((v,C_2),(w_2,w_1),t'_2),((v,C_1),(w_1,u),t'_1).$$

where $t'_i = t_{\max} - t_i + 1$, so $t'_n < \cdots < t'_2 < t'_1$. Since u,v,x are abitrary, A^T is an allgather schedule on G^T. One can similarly show that if (G,A) is an allgather algorithm, then (G^T,A^T) is a reduce-scatter algorithm. □

Corollary 1.1. *Suppose $G \mapsto f(G)$ is a function to construct reduce-scatter/allgather schedule given graph G, then $G \mapsto f(G^T)^T$ is a function to construct allgather/reduce-scatter schedule given graph G.*

Corollary 1.2. *Suppose $(G,A) \mapsto (f(G),f(A))$ is a mapping within reduce-scatter/allgather algorithms, then $(G,A) \mapsto (f(G^T)^T, f(A^T)^T)$ is a mapping within allgather/reduce-scatter algorithms.*

Theorem 2. *Suppose G is reverse-symmetric. Let G^T be the transpose graph, and let $f: V_{G^T} \to V_G$ be the isomorphism from G^T to G. If (G,A) is a reduce-scatter/allgather algorithm, then $(G, f(A^T))$ is an allgather/reduce-scatter algorithm with $T_L(f(A^T)) = T_L(A)$ and $T_B(f(A^T)) = T_B(A)$.*

Proof. By definition of $f(A^T)$,

$$((f(v),C),(f(w),f(u)),\, t_{\max} - t + 1) \in f(A^T)$$
$$\Leftrightarrow ((v,C),(w,u),t_{\max} - t + 1) \in A^T$$
$$\Leftrightarrow ((v,C),(u,w),t) \in A.$$

Note that $(u,w) \in E_G \Leftrightarrow (w,u) \in E_{G^T} \Leftrightarrow (f(w),f(u)) \in E_G$, so $f(A^T)$ is a valid schedule for G.

Suppose (G,A) is a reduce-scatter algorithm. For any $x \in S$ and distinct $u,v \in V_G$, there exists a sequence of tuples in A:

$$((v,C_1),(u,w_1),t_1),((v,C_2),(w_1,w_2),t_2),\ldots$$
$$((v,C_n),(w_{n-1},v),t_n),$$

where $t_1 < t_2 < \cdots < t_n$ and $x \in C_1 \cap C_2 \cap \cdots \cap C_n$. It follows that there exists a sequence of tuples in $f(A^T)$:

$$((f(v),C_n),(f(v),f(w_{n-1})),t'_n),$$
$$((f(v),C_{n-1}),(f(w_{n-1}),f(w_{n-2})),t'_{n-1}),$$
$$\vdots$$
$$((f(v),C_1),(f(w_1),f(u)),t'_1),$$

where $t'_i = t_{\max} - t_i + 1$, and $x \in C_n \cap C_{n-1} \cap \cdots \cap C_1$. Because f is a bijection, $(G, f(A^T))$ is an allgather algorithm. $T_L(A) = T_L(f(A^T))$ and $T_B(A) = T_B(f(A^T))$ are trivial, and one can similarly prove that if (G,A) is an allgather algorithm, then $(G, f(A^T))$ is a reduce-scatter algorithm. □

Theorem 3. *Every reduce-scatter/allgather algorithm (G,A) satisfies $T_L(A) \geq \alpha \cdot D(G)$, where $D(G)$ is the diameter of G.*

Proof. The proof is mentioned in text. □

Corollary 3.1. *An N-node degree-d reduce-scatter/allgather algorithm (G,A) is total-hop latency optimal if and only if*

$$T_L(A) = \alpha \cdot D(G) = \alpha \cdot \min\{D(G') : |V_{G'}| = N, \deg(G') = d\}.$$

Theorem 4. $\frac{M}{B} \cdot \frac{N-1}{N}$ *is a lower bound of $T_B(A)$ for any N-node reduce-scatter/allgather algorithm (G,A).*

Proof. The proof is mentioned in text. □

Corollary 4.1. *An N-node reduce-scatter/allgather algorithm (G,A) is BW-optimal if and only if $T_B(A) = \frac{M}{B} \cdot \frac{N-1}{N}$.*

Theorem 5. *An allgather algorithm (G,A) is BW-optimal if and only if:*

1. *$\frac{1}{B/d}\sum_{((v,C),(u,w))\in A_t}|C| = T_B(A_t)$ for all $(u,v)\in E_G$ and $t\in\{1,\dots,t_{\max}\}$. A_t is the subschedule of A at comm step t.*
2. *Pick any distinct $u,v\in V_G$. For each $x\in S$, there exists a unique $((v,C),(w,u),t)\in A$ such that $x\in C$.*

Proof. If $T_B(A)=T_B^*(N)=\frac{M}{B}\cdot\frac{N-1}{N}$, then the amount of data received by each vertex must be equal to $M\cdot\frac{N-1}{N}$, and the ingress bandwidth B must be fully utilized. If condition 1 does not hold, then some link (w,u) is not fully utilized. If condition 2 does not hold, then the amount of data received by some node is greater than $M\cdot\frac{N-1}{N}$.

If both 1 and 2 hold, then every vertex receives exactly $M\cdot\frac{N-1}{N}$ in total and bandwidth are fully utilized. Thus, $T_B(A)=T_B^*(N)$ and (G,A) is BW-optimal. □

Theorem 6. *Given a reduce-scatter algorithm (G,A), suppose $T_B(A)=\frac{M}{B}\cdot y$, then $T_C(A)\le M\cdot\gamma\cdot y$.*

Proof. At any comm step t, suppose the BW runtime is $T_B(A_t)=\frac{M}{B}\cdot y_t$. It follows at comm step t, the amount of data each node receives is at most $B\cdot T_B(A_t)=M\cdot y_t$, so $T_C(A_t)\le M\cdot\gamma\cdot y_t$. The theorem trivially follows $y{=}\sum_t y_t$. □

Corollary 6.1. *If allreduce algorithm (G,A) is BW-optimal, i.e., $T_B(A)=2\frac{M}{B}\cdot\frac{N-1}{N}$, then $T_C(A)=M\cdot\gamma\cdot\frac{N-1}{N}$ and $T_B(A)+T_C(A)=2M\cdot(\frac{1}{B}+\frac{\gamma}{2})\cdot\frac{N-1}{N}$.*

Theorem 8. *$(L^n(G),A_{L^n(G)})$ is Moore optimal if and only if (G,A_G) is Moore optimal.*

Proof. Suppose $T_L(A_G)=\alpha k$. Thus, (G,A_G) is Moore optimal if and only if

$$N > M_{d,k-1} = \sum_{i=0}^{k-1} d^i = \frac{d^k}{d-1} - \frac{1}{d-1}. \qquad (16)$$

$(L^n(G),A_{L^n(G)})$ is Moore optimal if and only if

$$d^nN > M_{d,k+n-1} \;\Leftrightarrow\; N > \frac{d^k}{d-1} - \frac{1}{d^n(d-1)}. \qquad (17)$$

Because (17) − (16) < 1 and (16) is an integer, (16) and (17) are equivalent. □

Theorem 10. *Let A_G be a BFB allgather schedule for d-regular topology G with $|N^+(u)|>1$ for all $u\in V_G$, then the expanded schedule $A_{L(G)}$ is a BFB allgather schedule for $L(G)$. In particular, if A_G is the optimal BFB schedule for G, then $A_{L(G)}$ is the optimal BFB schedule for $L(G)$ satisfying:*

$$T_B(A_{L(G)}) = T_B(A_G) + \frac{M}{B}\cdot\frac{1}{N}. \qquad (8)$$

Proof. It is trivial to see that $A_{L(G)}$ is a BFB allgather schedule on $L(G)$. For the sake of contradiction, suppose there exists a BFB schedule $A'_{L(G)}$ that $T_B(A'_{L(G)}) < T_B(A_G)+\frac{M}{B}\cdot\frac{1}{N}$. Let $x^*_{v'v,(wu,uu'),t}$s be the solution of BFB LP (1) corresponding to $A'_{L(G)}$. We build a schedule A'_G by constructing a solution of (1) such that

$$x_{v,(w,u),t} = \frac{1}{d}\sum_{v'\in N^-(v)} x^*_{v'v,(wu,uu'),t+1},$$

where $u'\in N^+(u)\setminus\{v\}$ is arbitrary. To verify the construction is a valid solution, given any $u\in V_G$ and $v\in N_t^-(w)$, the equality of (1) follows:

$$\begin{aligned}\sum_w x_{v,(w,u),t} &= \frac{1}{d}\sum_{v'}\sum_{w} x^*_{v'v,(wu,uu'),t+1}\\ &= \frac{1}{d}\sum_{v'}\sum_{wu} x^*_{v'v,(wu,uu'),t+1} = \frac{1}{d}\sum_{v'}1 = \frac{1}{d}\cdot d = 1.\end{aligned}$$

The third equality follows the equality constraint in (1). Now, given $(w,u)\in E_G$, observe that

$$\begin{aligned}\sum_v x_{v,(w,u),t} &= \frac{1}{d}\sum_{v}\sum_{v'} x^*_{v'v,(wu,uu'),t+1}\\ &= \frac{1}{d}\sum_{v'v} x^*_{v'v,(wu,uu'),t+1} \le \frac{1}{d}U^*_{uu',t+1}.\end{aligned}$$

Thus, $U_{u,t}=\max_w\sum_v x_{v,(w,u),t}\le\frac{1}{d}U^*_{uu',t+1}$ and hence

$$\max_{u\in V_G} U_{u,t} \le \frac{1}{d}\max_{uu'\in V_{L(G)}} U^*_{uu',t+1}.$$

Note that $U^*_{uu',1}=1$ for all $uu'\in V_{L(G)}$, as each node must send the full shard to every neighbor at the 1st comm step in any BFB allgather schedule. By (2), we have

$$\begin{aligned}T_B(A'_G) &\le T_B(A'_{L(G)}) - \frac{M/(dN)}{B/d}\\ &= T_B(A'_{L(G)}) - \frac{M}{B}\cdot\frac{1}{N} < T_B(A_G),\end{aligned}$$

contradicting A_G being the optimal BFB schedule. Thus, combined with inequality (5), we have proven $A_{L(G)}$ being optimal as well as the equality (8). □

Corollary 10.1. *Let A_G be a BFB allgather schedule for d-regular topology G with $|N^+(u)|>1$ for all $u\in V_G$, then the expanded schedule $A_{L^n(G)}$ is a BFB allgather schedule for $L^n(G)$. In particular, if A_G is the optimal BFB schedule for G, then $A_{L^n(G)}$ is the optimal BFB schedule for $L^n(G)$ satisfying:*

$$T_B(A_{L^n(G)}) = T_B(A_G) + \frac{M}{B}\cdot\frac{d}{d-1}\left(\frac{1}{N}-\frac{1}{d^nN}\right).$$

Theorem 11. *Given a d-regular topology G without self loops, if (G,A_G) is an N-node allgather algorithm with $T_B(A_G,M,B)=\tau(M/B)$ for some constant τ, then $(G*n,A_{G*n})$ is an nN-node allgather algorithm satisfying:*

$$T_L(A_{G*n}) = T_L(A_G)+\alpha, \qquad (9)$$

$$T_B(A_{G*n}) = T_B(A_G)+\frac{M}{B}\cdot\frac{n-1}{nN}. \qquad (10)$$

Proof. Let u_i,v_j be arbitrary two distinct vertices in $G*n$. Suppose $u\neq v$ in G, then for any $x\in S$, there exists a sequence in A_G:

$$((v,C_1),(v,w^{(1)}),t_1),((v,C_2),(w^{(1)},w^{(2)}),t_2),\dots$$

$$((v,C_n),(w^{(n-1)},u),t_n),$$

where $t_1 < t_2 < \cdots < t_n$ and $x \in C_1 \cap C_2 \cap \cdots \cap C_n$. By Definition 2, there exists a sequence in A_{G*n}:

$$((v_j,C_1),(v_j,w_j^{(1)}),t_1),((v_j,C_2),(w_j^{(1)},w_j^{(2)}),t_2),\ldots$$
$$((v_j,C_n),(w_j^{(n-1)},u_i),t_n),$$

as desired. Now, suppose $u = v$ in G. By previous proof, the shard of v_j reaches every in-neighbor u'_α of u_i by the end of comm step $t_{\max}$ since $u' \neq v$. Then, the last comm step $t_{\max}+1$ added in step 2 of Definition 2 delivers the shard to u_i with each edge (u'_α,u_i) delivering $1/nd$ of a shard. Thus, A_{G*n} is a complete allgather.

In step 1 of Definition 2, we have $T_B([A_{G*n}]_t,nM,nB) = T_B([A_G]_t,M,B)$ and hence $\sum_{t=1}^{t_{\max}} T_B([A_{G*n}]_t,nM,nB) = T_B(A_G,M,B)$. The nM and nB are due to the fact that both the number of nodes and degree have grown n-fold. Thus,

$$\begin{aligned}
T_B(A_{G*n},M,B) &= T_B(A_{G*n},nM,nB)\\
&= T_B(A_G,M,B) + T_B([A_{G*n}]_{t_{\max}+1},nM,nB)\\
&= T_B(A_G,M,B) + (n-1)\cdot\frac{(nM)/(nN)}{nd}\cdot\frac{1}{nB/(nd)}\\
&= T_B(A_G,M,B) + \frac{M}{B}\cdot\frac{n-1}{nN}.
\end{aligned}$$

The first equality follows the assumption that $T_B(A_G,M,B) = \tau(M/B)$ for some constant τ. □

Corollary 11.1. *If (G,A_G) is BW-optimal and $T_B(A_G,M,B) = \tau(M/B)$ for some τ, then $(G*n,A_{G*n})$ is BW-optimal.*

Theorem 12. *Given a d-regular topology G, if (G,A_G) is an N-node allgather algorithm with $T_B(A_G,M,B) = \tau(M/B)$ for some constant τ, then $G^{\square n}$ is an nd-regular topology, and $(G^{\square n},A_{G^{\square n}})$ is an N^n-node allgather algorithm satisfying:*

$$T_L(A_{G^{\square n}}) = n\cdot T_L(A_G), \tag{11}$$

$$T_B(A_{G^{\square n}}) = T_B(A_G)\cdot\frac{N}{N-1}\cdot\frac{N^n-1}{N^n}. \tag{12}$$

Proof. We will show that $A^{(1)}$ is a valid allgather schedule. Since $A^{(i)}$s are simply starting at different dimensions, this also shows that $A^{(i)}$s and hence $A_{G^{\square n}}$ are all valid allgather schedules for $G^{\square n}$.

Let $\mathbf{u}$ be arbitrary vertex in $G^{\square n}$. For any $x \in S$, we will show that schedule $A^{(1)}$ broadcasts x from $\mathbf{u}$ to all vertices in $G^{\square n}$. At $j=1$, $A^{(1)}$ performs an allgather over vertices $\{(v_1,\mathbf{u}[2:n]) \mid v_1 \in V_G\}$ which induce a subgraph of $G^{\square n}$ isomorphic to G. Thus, x has been broadcast to all vertices in $\{(v_1,\mathbf{u}[2:n]) \mid v_1 \in V_G\}$. At $j=2$, $A^{(1)}$ performs an allgather over vertices $\{(v_1,v_2,\mathbf{u}[3:n]) \mid v_2 \in V_G\}$ for each v_1. By the end of $j=2$, x has been broadcast to all vertices in $\{(v_1,v_2,\mathbf{u}[3:n]) \mid v_1,v_2 \in V_G\}$. By the end of $j=n$, x has been broadcast to all vertices in $\{\mathbf{v} \mid \mathbf{v} \in V_G^n\} = V_{G^{\square n}}$. Since $\mathbf{u}$ and x are arbitrary, $A^{(1)}$ is a valid allgather schedule for $G^{\square n}$.

As for performance, (11) is trivial. To prove (12), observe that at each j in $A^{(1)}$, allgather A_G is performed with a data size $N^{j-1}M/n$ over the subgraph induced by $\{(\mathbf{y},v,\mathbf{z}) \mid v \in V_G\}$ for each $\mathbf{y} \in V_G^{j-1}, \mathbf{z} \in V_G^{n-j}$. The bandwidth of each node within the subgraph is $1/n$ of that in $G^{\square n}$. It follows that

$$\begin{aligned}
T_B(A^{(1)},N^{n-1}M/n,nB) &= \sum_{j=1}^{n} T_B(A_G,N^{j-1}M/n,B)\\
&= \sum_{j=1}^{n}\frac{N^{j-1}}{n}T_B(A_G,M,B)\\
&= \frac{N^n-1}{n(N-1)}T_B(A_G,M,B).
\end{aligned}$$

Therefore,

$$\begin{aligned}
T_B(A_{G^{\square n}},M,B) &= \frac{n}{N^{n-1}}T_B(A_{G^{\square n}},N^{n-1}M,nB)\\
&= \frac{n}{N^{n-1}}T_B(A^{(1)},N^{n-1}M/n,nB)\\
&= \frac{n}{N^{n-1}}\cdot\frac{N^n-1}{n(N-1)}T_B(A_G,M,B)\\
&= T_B(A_G,M,B)\cdot\frac{N}{N-1}\cdot\frac{N^n-1}{N^n}.
\end{aligned}$$

□

Corollary 12.1. *If (G,A_G) is BW-optimal and $T_B(A_G,M,B) = \tau(M/B)$ for some τ, then $(G^{\square n},A_{G^{\square n}})$ is BW-optimal.*

Theorem 13. *Let $G_1,G_2,\ldots,G_n$ be topologies that*

1. *$G_1,\ldots,G_n$ are nontrivial simple digraphs;*
2. *Every G_i has BW-optimal BFB allgather schedule.*

Then, the optimal BFB allgather schedule, i.e. the schedule generated by BFB LP (1), for $G_1\square\ldots\square G_n$ is also BW-optimal. The total-hop latency of the schedule equals $\alpha\cdot D(G_1\square\ldots\square G_n) = \alpha\cdot\sum_i D(G_i)$.

Proof. To prove the theorem, it is sufficient to show that if G_1 and G_2 have BW-optimal BFB schedules, then $G_1\square G_2$ has a BW-optimal BFB schedule. By Theorem 16, let $x^*_{v_1,(w_1,u_1),t_1}$s and $x^*_{v_2,(w_2,u_2),t_2}$s be the solutions of (1) on G_1 and G_2 respectively. Let $\mathbf{u} = (u_1,u_2), \mathbf{v} = (v_1,v_2)$. Define $r \in [0,1]$, which we will decide later. We construct a solution of (1) for $G_1\square G_2$ such that:

$$\begin{aligned}
x_{\mathbf{v},((w_1,u_2),\mathbf{u}),t_1+t_2} &= \begin{cases} r\cdot x^*_{v_1,(w_1,u_1),t_1} & \text{if } u_2 \neq v_2,\\ x^*_{v_1,(w_1,u_1),t_1} & \text{if } u_2 = v_2,\end{cases}\\
x_{\mathbf{v},((u_1,w_2),\mathbf{u}),t_1+t_2} &= \begin{cases} (1-r)\cdot x^*_{v_2,(w_2,u_2),t_2} & \text{if } u_1 \neq v_1,\\ x^*_{v_2,(w_2,u_2),t_2} & \text{if } u_1 = v_1.\end{cases}
\end{aligned} \tag{18}$$

First of all, because $d_{G_1\square G_2}(\mathbf{v},\mathbf{u}) = d_{G_1}(v_1,u_1) + d_{G_2}(v_2,u_2)$, it is easy to verify that (18) gives a BFB schedule. In addition, for any distinct $\mathbf{u},\mathbf{v} \in G_1\square G_2$ with $u_1 \neq v_1$ and $u_2 \neq v_2$,

$$\begin{aligned}
\sum_{\mathbf{w}} x_{\mathbf{v},(\mathbf{w},\mathbf{u}),t_1+t_2} &= r\sum_{w_1} x^*_{v_1,(w_1,u_1),t_1} + (1-r)\sum_{w_2} x^*_{v_2,(w_2,u_2),t_2}\\
&= r + (1-r)\\
&= 1
\end{aligned}$$

satisfying the equality in (1). The $u_1 = v_1$ or $u_2 = v_2$ case is trivial. Because G_1 and G_2 have BW-optimal BFB schedule, by Theorem 17, for any $(w_1,u_1)\in E_{G_1}$,

$$\sum_{v_1\in N_t^{-G_1}(u_1)} x^*_{v_1,(w_1,u_1),t} = \frac{N_t^{-G_1}}{d_1}, \tag{19}$$

where $N_t^{-G_1}(u_1)$ is $N_t^-(u_1)$ in G_1. Define $N_{t_1,t_2}^{-G_1\square G_2}(\mathbf{u}) = N_{t_1}^{-G_1}(u_1)\times N_{t_2}^{-G_2}(u_2)$, then it holds that

$$N_t^{-G_1\square G_2}(\mathbf{u}) = \bigcup_{t_1=0}^{t} N_{t_1,t-t_1}^{-G_1\square G_2}(\mathbf{u})$$

$$= N_t^{-G_1}(u_1)\times\{u_2\}\cup\{u_1\}\times N_t^{-G_2}(u_2)\cup\bigcup_{t_1=1}^{t-1} N_{t_1,t-t_1}^{-G_1\square G_2}(\mathbf{u}).$$

Thus, (19) gives

$$\sum_{\mathbf{v}\in N_t^{-G_1\square G_2}(\mathbf{u})} x_{\mathbf{v},((w_1,u_2),\mathbf{u}),t} = \frac{N_t^{-G_1}}{d_1} + r\sum_{t_1=1}^{t-1}\frac{N_{t_1}^{-G_1}N_{t-t_1}^{-G_2}}{d_1}. \tag{20}$$

for any $((w_1,u_2),\mathbf{u})\in E_{G_1\square G_2}$. For G_2, one can similarly get

$$\sum_{\mathbf{v}\in N_t^{-G_1\square G_2}(\mathbf{u})} x_{\mathbf{v},((u_1,w_2),\mathbf{u}),t} = \frac{N_t^{-G_2}}{d_2} + (1-r)\sum_{t_2=1}^{t-1}\frac{N_{t-t_2}^{-G_1}N_{t_2}^{-G_2}}{d_2}. \tag{21}$$

The value of r is the solution to (20) = (21):

$$\frac{N_t^{-G_1}}{d_1} + r\sum_{t_1=1}^{t-1}\frac{N_{t_1}^{-G_1}N_{t-t_1}^{-G_2}}{d_1} = \frac{N_t^{-G_2}}{d_2} + (1-r)\sum_{t_2=1}^{t-1}\frac{N_{t-t_2}^{-G_1}N_{t_2}^{-G_2}}{d_2}.$$

To see there is always a solution $r\in[0,1]$, we have $N_t^{-G_1}\le d_1\cdot N_{t-1}^{-G_1}$ and $N_t^{-G_2}\le d_2\cdot N_{t-1}^{-G_2}$, so

$$\frac{N_t^{-G_1}}{d_1} - \frac{N_t^{-G_2}}{d_2} \le \frac{N_t^{-G_1}}{d_1} \le N_{t-1}^{-G_1} \le \sum_{t_2=1}^{t-1}\frac{N_{t-t_2}^{-G_1}N_{t_2}^{-G_2}}{d_2},$$

$$\frac{N_t^{-G_2}}{d_2} - \frac{N_t^{-G_1}}{d_1} \le \frac{N_t^{-G_2}}{d_2} \le N_{t-1}^{-G_2} \le \sum_{t_1=1}^{t-1}\frac{N_{t_1}^{-G_1}N_{t-t_1}^{-G_2}}{d_1}.$$

The last inequality follows that because G_2 is nontrivial simple digraph, $N_1^{-G_2} = d_2$ and hence $N_{t-1}^{-G_1} = N_{t-1}^{-G_1}N_1^{-G_2}/d_2$. Note that $a+rb = c+(1-r)d$ always has a solution $r\in[0,1]$ if $a-c\le d$ and $c-a\le b$. With (20) = (21), by Theorem 17, we have constructed a BW-optimal solution of (1) for $G_1\square G_2$. The theorem trivially follows by induction. □

Theorem 14. *A schedule A for G is a BFB allgather schedule if and only if the following are satisfied:*

1. *If $((v,C),(w,u),t)\in A$, then $d(v,u) = d(v,w)+1 = t$;*
2. *For any distinct $u,v\in V_G$, the collection of chunks $\mathcal{C}_v = \{C \mid ((v,C),(w,u),t)\in A\}$ satisfies $S = \bigcup_{C\in\mathcal{C}_v} C$.*

Proof. Let v_0, v_k be arbitrary two distinct vertices in V_G with $d(v_0,v_k) = k$. For any $x\in S$, we want to show that there exists a path taking x from v_0 to v_k. At comm step k, conditions 1 and 2 guarantee that there exists $v_{k-1}\in N^-(v_k)$ and $((v_0,C_k),(v_{k-1},v_k),k)\in A$ such that $d(v_0,v_{k-1}) = k-1$ and $x\in C_k$. At comm step $k-1$, similarly, it is guaranteed that there exists $v_{k-2}\in N^-(v_{k-1})$ and $((v_0,C_{k-1}),(v_{k-2},v_{k-1}),k-1)\in A$ such that $d(v_0,v_{k-2}) = k-2$ and $x\in C_{k-1}$. Thus, we have a sequence of tuples in A:

$$((v_0,C_1),(v_0,v_1),1),((v_0,C_2),(v_1,v_2),2),\ldots$$
$$((v_0,C_k),(v_{k-1},v_k),k),$$

where $x\in C_1\cap C_2\cap\cdots\cap C_k$ as desired. In the other direction, if condition 1 fails, then A is not a BFB schedule; if condition 2 fails, then A is not a valid allgather schedule. □

Theorem 15. *If A is a BFB schedule for G, then the total-hop latency $T_L(A) = \alpha\cdot D(G)$.*

Proof. The proof is trivial. □

Theorem 16. *Given any topology G, linear program (1) gives the optimal BFB schedule of G.*

Proof. The proof is mentioned in text. □

Theorem 17. *Suppose (G,A) is a BFB allgather schedule. (G,A) is BW-optimal if and only if:*

1. *There exists a sequence $N_1^-, N_2^-, \ldots, N_{D(G)}^-\in\mathbb{N}$ such that for any $x\in\mathbb{N}$ and $u\in V_G$, $|N_x^-(u)| = N_x^-$.*
2. *For any $(w,u)\in E_G$, $\sum_{((v,C),(w,u))\in A_t}|C| = \frac{M}{N}|N_t^-(u)|/d = \frac{M}{N}N_t^-/d$.*

Proof. At comm step t, each vertex needs to receive shards from vertices in $N_t^-(u)$. By condition 1 of Theorem 5, each in-edge of vertex u receives equal amount of data, so each in-edge receives $\frac{M}{N}|N_t^-(u)|/d$. In addition, condition 1 of Theorem 5 also forces every edge in G receiving equal amount of data at any given comm step, so BW optimality is achieved if and only if $\frac{M}{N}|N_t^-(u)|/d = \frac{M}{N}|N_t^-(v)|/d = \frac{M}{N}N_t^-/d$ for all $u,v\in V_G$. Note that condition 2 of Theorem 5 is automatically satisfied. Thus, the conditions of Theorem 17 lead to Theorem 5, and vice versa. □

Theorem 18. *There exists a BW-optimal BFB schedule for undirected graph G if for every distance x, two of the following constants exist:*

1. *$N_x = |N_x(u)|$ for any $u\in V_G$;*
2. *$a_x = |N_x(u)\cap N_{x-1}(w)|$ for any $u\in V_G$ and $w\in N(u)$;*
3. *$b_x = |N(u)\cap N_{x-1}(v)|$ for any $u\in V_G$ and $v\in N_x(u)$.*

Moreover, if two of N_x, a_x, b_x exist, then the third one must also exist with $N_x = da_x/b_x$.

Proof. It is easy to see $N_x = da_x/b_x$. Constant N_xs satisfy condition 1 of Theorem 17. As for 2 of Theorem 17, at comm step t, consider a BFB schedule such that for any $u,v,w\in V_G$ with $d(v,u) = d(v,w)+1 = t$, node w sends $1/b_t$ of v's shard to u. Thus, $\sum_{((v,C),(w,u))\in A_t}|C| = \frac{M}{N}a_t/b_t = \frac{M}{N}N_t/d$. □

Theorem 19. *The workloads can be balanced if and only if there exists no subset $J\subseteq\{j_1,j_2,\ldots,j_m\}$ such that*

$$\frac{|J|}{\left|\bigcup_{j\in J}f(j)\right|} > \frac{m}{d}.$$

Proof. Consider a flow network, where each j_a is connected to each $p_b \in f(j_a)$ with ∞ capacity. Source s is connected to each j_a with capacity 1, and each p_b is connected to sink t with capacity m/d. Thus, the workloads can be balanced if and only if the max flow is m. Given any subset J, consider the s-t cut $(A, \bar{A})$ that $A = s + J + f(J)$. The cut has capacity $m - |J| + \frac{m}{d}|f(J)|$, which is less than m if and only if the inequality is true. □

Theorem 20. *Rounding LP gives a solution with BW runtime* $T_B \le T_B^{OPT} + \frac{M}{B} \cdot \frac{d(d^{D(G)}-1)}{(d-1)PN}$. *In addition, if topology* G *is Moore optimal, then* $T_B \le T_B^{OPT} + \frac{M}{B} \cdot \frac{d}{P}$.

Proof. For any (w,u) at comm step t, since $|N_{t-1}^{-}(w)| \le d^{t-1}$,

$$\sum_v y_{v,(w,u),t} < \sum_v 1 + y_{v,(w,u),t}^{\mathrm{LP}} \le d^{t-1} + \sum_v y_{v,(w,u),t}^{\mathrm{LP}}.$$

Thus, we have $W_{u,t} \le W_{u,t}^{\mathrm{LP}} + d^{t-1}$. By (2),

$$\begin{aligned} T_B - T_B^{\mathrm{OPT}} &\le T_B - T_B^{\mathrm{LP}} \\ &\le \frac{M/N}{B/d} \cdot \frac{1}{P} \sum_{t=1}^{D(G)} d^{t-1} = \frac{M}{B} \cdot \frac{d(d^{D(G)}-1)}{(d-1)PN}. \end{aligned}$$

Note that we need to divide (2) by P, because $y_{v,(w,u),t} \in [0,P]$ in (13) while $x_{v,(w,u),t} \in [0,1]$ in (1). If G is Moore optimal (i.e., $N > M_{d,D(G)-1}$), it follows that

$$T_B - T_B^{\mathrm{OPT}} < \frac{M}{B} \cdot \frac{d(d^{D(G)}-1)}{(d-1)PM_{d,D(G)-1}} = \frac{M}{B} \cdot \frac{d}{P}.$$

□

Theorem 21. *Suppose* $D(\Pi_{d,m}) = k$, *then* $m > M_{d,k-2}$.

Proof. From [25], we know that $k \le \lceil \log_d m \rceil$. Then,

$$m \ge d^{k-1} > \frac{d^{k-1}-1}{d-1} = M_{d,k-2}.$$

□

Theorem 22. *Given* $n > 6$ *and* $m = \lceil(-1+\sqrt{2n-1})/2\rceil$, *circulant graph* $C(n, \{m, m+1\})$ *has a diameter equal to* m, *which is the minimum diameter over all circulant graphs* $C(n, \{a_1, a_2\})$.

Proof. See [7]. □

H Supplementary Tables and Figures

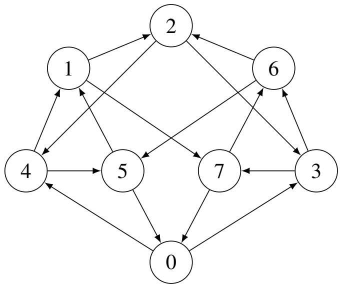
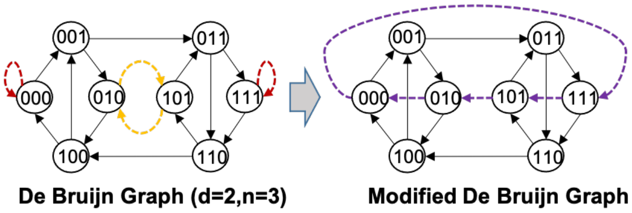


Figure 19: Diamond Topology $(N = 8, d = 2)$**.**



De Bruijn Graph (d=2,n=3)

Modified De Bruijn Graph

Figure 20: An Example of Modified de Bruijn Graph $(N = 8, d = 2)$**.** The modification rewires the self loops and 2-cycles in de Bruijn graph to form a single long cycle without violating degree constraint.

Topology	Notation	Degree	Size	Reverse-Symmetric	Bandwidth Optimal	Moore Optimal	BFB Schedule	Self-Loop	MultiEdge
Complete	K_m	$m-1$	m	✓	✓	✓	✓	×	×
Complete Bipartite (Fig 1)	$K_{d,d}$	d	$2d$	✓	✓	✓	✓	×	×
Hamming	$H(n,q) = K_q^{\square n}$	$n(q-1)$	q^n	✓	✓	$T_L = n$	✓	×	×
Kautz	$K(d,n) = L^n(K_{d+1})$	d	$d^n(d+1)$	✓	when $n = 0$	✓	✓	×	×
Generalized Kautz (§F.2)	$\Pi_{d,m}$	d	$m \geq d+1$	×	when $m = d+1$	$T_L \leq T_L^* + 1$	✓	when $m \bmod (d+1) \neq 0$	×
Circulant (§F.4)	$C(m, \{a_1, \ldots, a_d\})$	d	m	✓	✓	×	✓	×	×
Directed Circulant		d	$d+2$	✓	✓	✓	✓	×	×
Bidirectional Ring	BiRing(d,m)	even d	$m \geq 3$	✓	✓	$T_L = \lfloor m/2 \rfloor$	✓	×	when $d > 2$
Unidirectional Ring	UniRing(d,m)	d	m	✓	✓	$T_L = m-1$	✓	×	when $d > 1$
Diamond (Fig 19)		2	8	×	✓	✓	×	×	×
de Bruijn	DBJ(d,n)	d	d^n	✓	when $n \leq 1$	✓	✓	✓	×
Modified de Bruijn (Fig 20)	DBJMod$(2,3)$	2	8	✓	✓	$T_L = 4$	×	×	×
	DBJMod$(2,4)$	2	16	×	✓	$T_L = 5$	×	×	×
	DBJMod$(3,2)$	3	9	×	✓	$T_L = 3$	×	×	×
	DBJMod$(4,2)$	4	16	×	✓	$T_L = 3$	×	×	×
Distance-Regular Graphs (§F.3)	DistReg(d,m)	d	m	✓	✓		✓	×	×

Table 9: Summary of Important Topologies.