



UE Collective Communications Benchmark Performance Specification

v1.0

October 28, 2026

Copyright © 2026 Ultra Ethernet Consortium.

The Ultra Ethernet Consortium (UEC) is a consensus-based standards organization operating under the Joint Development Foundation. Its members collaborate openly to define and promote high-performance Ethernet technologies for modern computing environments. This specification ("Specification") contains Approved Deliverables of UEC as such term is defined in the UEC Charter.

This document is made available under the terms of the Creative Commons Attribution-NoDerivatives 4.0 International License (CC BY-ND 4.0). As stated in the license, you may copy and redistribute the document, provided that attribution is given to the Ultra Ethernet Consortium. However, if you create a derivative work from the material, you may not distribute that work. A copy of the license is available at <https://creativecommons.org/licenses/by-nd/4.0/>.

This Specification is provided on an "AS IS" basis. UEC and The Joint Development Foundation (the "Parties") make no warranties of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose or noninfringement of any third party intellectual property rights, or compliance with applicable laws. Use of the information in this Specification is at your own risk. The entire risk as to the results and performance of this Specification is assumed by the user. In no event will the Parties (as defined above) be liable to any other party for lost profits or any form of indirect, special, incidental, or consequential damages of any character from any causes of action of any kind with respect to this Specification or its governing document, whether based on breach of contract, tort (including negligence), or otherwise, and whether or not the recipient of this Specification has been advised of the possibility of such damage.

This Specification may include references to technologies for which patent rights might be claimed. UEC does not make any representation regarding the existence or status of any such rights. Implementers of this Specification are solely responsible for obtaining any necessary licenses or permissions from the appropriate rights holders.

UEC reserves the right to adopt any changes or alterations to this Specification as it deems necessary or appropriate.

Ultra Ethernet, Ultra Ethernet Consortium, and the Ultra Ethernet Consortium logo are the unregistered trademarks of Ultra Ethernet Consortium in the United States and other countries. All rights reserved. Other brands, names or logos contained herein are the property of their respective owners.

For more information about the Ultra Ethernet Consortium, visit: <https://ultraethernet.org>.

Contents

1	Introduction	8
1.1	How to Use This Document.....	8
1.2	Philosophy of This Document.....	8
1.3	Terminology	9
1.4	Not in Scope	11
2	Collective Communications Benchmarks	11
2.1	Collective Communications Benchmark Parameters & Settings	11
2.2	Collective Workload Parameters.....	12
2.3	Curated List of Collective Communications Benchmarks	13
2.3.1	Curated Benchmarking Sets	13
3	Performance Metrics, Measurement & Reporting.....	16
3.1	Measured Values	16
3.2	Computed and Derived Values.....	17
3.3	Benchmark Report	17
3.3.1	Benchmark Results Summary	17
3.3.2	Testbed setup details	18
3.3.3	Custom collective algorithm description	19
3.4	Benchmarking Profile ID.....	19
3.4.1	Profile ID Structure	19
3.4.2	Profile ID Fields	20
3.4.3	Profile ID Tags	20
3.4.4	Profile ID Examples.....	21
3.4.5	Profile ID Notes	22
4	Terminology of Collective Communications [Informative].....	22
4.1	Application Layer	22
4.2	Transport Semantics	23
4.3	Transport Protocol	24
4.4	Transport Metrics	25
5	References	26

Tables

Table 2-1 - Primary parameters	12
Table 2-2 - Additional/Optional parameters	12
Table 2-3 - Latency Baseline Benchmarks	14
Table 2-4 - Bandwidth Baseline Benchmarks	14
Table 2-5 - Congestion-Control Stress Tests	14
Table 2-6 - Load Balancing Tests	15
Table 3-1 - Measured Values	16
Table 3-2 - Computed and Derived Values	17
Table 3-3 - Benchmark Results Summary	18
Table 3-4 - Profile ID Fields	20
Table 3-5 - Profile ID Tags	20

Figures

Figure 4-1 - CCL Layer	23
Figure 4-2 - Transport Semantics Sublayer	24
Figure 4-3 - Transport Packet Delivery Sublayer	25

Acknowledgments

This specification is developed under the Process, Procedures, and Guidelines of the Ultra Ethernet Consortium (UEC). The organization would like to thank all that are involved. A special acknowledgment goes out to the following individuals for their leadership, authoring, and technical contributions to this specification.

UEC Steering Committee at Time of Publication

Role	Name	Affiliation
Chair	J Metz	AMD
Vice Chair	Abdul Kabbani	Microsoft
TAC Chair	Tom Emmons	Arista
TAC Vice Chair	Brad Matthews	Marvell

UEC Working Group Leadership at Time of Publication

Working Group	Name	Affiliation
Physical Layer WG	Jannik Hammel Nielsen	Huawei
	Adee Ran	Cisco
Link Layer WG	Bob Alverson	HPE
	Yan Zhuang	Huawei
Transport Layer WG	Torsten Hoefler	Microsoft
	Karen Schramm	Broadcom
Software Layer WG	Rip Sohan	Broadcom
	Jianxin Xiong	Intel
Management WG	Ravindranath Kanakarajan	Marvell
	Hemal Shah	Broadcom
Compliance & Test WG	Dave Estes	Viavi Solutions
	Cliff Tavares	Keysight
Performance & Debug WG	Rajalaxmi Angadi	Intel
	Nathan Hanford	LLNL
Storage WG	Venkat Goli	Broadcom
	Claudio DeSanti	Dell

Specification Development

Role	Name	Affiliation
Author	Chris Sommers	Keysight Technologies
Author	Yanfang Le	AMD Inc
Author	Rajalaxmi Angadi	Intel Corporation
Contributor	Jiaqi Gao	Alibaba Inc

Role	Name	Affiliation
Contributor	Nusrat Islam	AMD Inc
Contributor	Randy Zhang	Cisco Systems
Contributor	Irfan Ashraf	Marvell Technology
Contributor	Alex Bortok	Keysight Technologies
Contributor	Ulf Hanebutte	Marvell Technology
Contributor	Rahul Sachdev	Marvell Technology
Contributor	Maria Garzaran	Intel Corporation
Contributor	Bob Lantz	HPE
Contributor	Nathan Hanford	LLNL
Editor	Paul Congdon	Linux Foundation
Program Manager	Rob Craig	Linux Foundation

1 Introduction

This UE Collective Communications Performance Benchmark Specification addresses performance requirements for assessing the Ultra Ethernet Transport (UET) stack.

This specification defines a suite of benchmarks for evaluating collective communications over Ultra Ethernet. The collective communications benchmarks highlight the essential features advocated by UEC from the application perspective. This document aims to alleviate significant challenges during performance benchmarking such as inconsistent performance evaluations and complexities in optimizing network configurations.

Due to the nature of rapid technological development and lack of available physical hardware for testing, the benchmarks will accommodate testing both behavioral models and physical hardware. As a result, this specification will provide a uniform standard to evaluate the utility and performance of different alternatives across a variety of platforms, such as simulator, kernel implementation, software emulator, or real hardware.

This specification describes recommended topologies and workload parameters when running the benchmarks for the purposes of establishing and publishing performance for the UE-compliant implementations. These settings and parameters may also be considered as best practice examples for documenting the environment for benchmarking other DUTs. Optional parameters are also provided for more fair comparison and reproducibility.

1. How to Use This Document

This specification can be thought of as a sort of technical “cookbook.” It contains “recipes” to follow to produce a finished product – standardized test results. Like any good cookbook, it starts out with a discussion of fundamental concepts to establish a common vocabulary and educate the “cook,” or test practitioner.

The reader may consume this document from start to finish, but more experienced and/or eager readers can jump directly to section 1.5.

Readers may generate their own “recipes,” study the “ingredients” in section 1.5 to come up with their own creations.

Master Chefs are encouraged to contribute more recipes and ingredients to this cookbook, by participating in the Ultra Ethernet performance and debug working group.

2. Philosophy of This Document

This specification focuses on the *what*, more than the *how*. By *what*, the specification means the most important test case parameters and report contents. By *how*, the specification means the precise methodology and testbed setup details which are too numerous to prescribe and must be flexible to accommodate differing implementations and test environment. Nevertheless, the final outcome should be test performance results and setup details which can be comprehended, reproduced, and compared amongst different implementations.

Therefore, all test reports should contain sufficient details about the test setup, environment, device under test configuration, custom algorithms, measurement methodology, etc. such that the results, and how they were obtained, are sufficient, transparent and meaningful. This specification strives to enumerate some needful details here but cannot account for everything. When in doubt, over-document, rather than under-document!

3. Terminology

The following terms are used by the UE Performance specifications.

Term	Description
Algorithmic Bandwidth (algbw)	Metric to compute bandwidth based on Data Size (S) and Collective Completion Time (CCT), It is computed as the ratio of total data size S over collective completion time CCT, typically in GB/s. See section 9
Allgather	Collective operation, where Data of size S on each rank is gathered and distributed to all ranks, resulting in each rank having the complete set of data
Allreduce	Collective operation, where Data of size S on each rank is combined by applying a reduction operator (sum, max, etc.) and result is broadcast to all ranks
AlltoAll	Collective operation, where each rank sends same amount of data to all other ranks. In contrast, AlltoAllv has variable per rank data size (not considered in this benchmark)
Bus Bandwidth (busbw)	Metric to measure the effective hardware throughput or bottleneck of the underlying interconnect (bus) for a particular Collective Communications operation. It is often indirectly computed by adjusting the algbw with a factor that is dependent on the collective operation: $\text{busbw} = \text{algbw} * \text{factor}$, where the factor depends on the collective algorithm. As example, this factor is $2 * (p-1) / p$, for allreduce, and $(p-1)/p$ for allgather, where p is the number of ranks that participate in the collective
Broadcast	Collective operation, where Data of size S is broadcast from one rank to all other ranks
Algorithmic Rank-Chunk (Rank-Chunk)	A logically defined portion of a rank's Data Size (S), determined by the collective algorithm, that is scheduled for transfer with its peer ranks as part of executing the collective. It represents the algorithm-level unit of data movement, independent of pipelining, transport message sizing, or the number of flows used for its transfer.
Collective Algorithms (Strategies)	Collective algorithms or strategies are used to increase efficiencies when working with collectives. These are logical architectures for point-to-point message passing at the application layer
Collective Completion Time (CCT)	Time it takes for a Collective Operation to complete (from first rank to start to the last rank to finish); useful to compare Collective Algorithm performance on small data sizes.
Collective Operators	Communication patterns that involve a group of processes to exchange data. Elemental blocks of network communication in scale-out AI/ML clusters that move data between the GPUs.
Collective group Size	Number of ranks in the collective operation.
Custom Algorithm	Collective Algorithm which deviates from the collective algorithms and its parameter settings as defined in the benchmark document. Custom-

Term	Description
	algorithms are permitted; however they have to be clearly identified when results are reported.
Data Size (S)	Size of the data one rank has as input to the operation
DUT	Device under test
Double binary tree	Collective Algorithm where all nodes are formed in two distinctive binary trees, with both including all the ranks participating in the collective.
Fabric Utilization	Average utilization across all the ingress/egress ports to the fabric used by the ranks during the CCT window. Formula: (Total bytes on the ports in one direction) / (Sum of ports bandwidth * CCT). See section 9.
Flow Completion Time (FCT)	Elapsed time for a single flow—defined as a serialized sequence of transport messages—measured from the transmission of the first message in that flow to the completion of its last message at initiator. A single algorithmic rank-chunk may be transferred over multiple concurrent flows; FCT applies per flow, not per chunk.
Incast-generating-algorithm	This is a pathological case, with a non-optimal strategy for a collective, where all ranks send to the same target rank simultaneously
In-network computation	A computing paradigm where certain collective communication operations are offloaded and executed within the networking infrastructure itself rather than solely at the end hosts.
In-place	It usually refers to the case where both the send and the receive buffer use the same memory location. This approach is primarily used to optimize memory usage and reduce data movement.
Message	Message is a Transport work request. Message size is the payload size described in one work request.
N-Parallel	A parameter used in the AlltoAll algorithm, representing the number of destinations a sender communicates concurrently.
Rank	Identifiers of endpoints exchanging messages in the collective operation. Often, a rank represents a GPU. In some cases, a GPU can have many ranks.
Rank-Chunk Completion Time (RCCT)	Time it takes to transfer an algorithmic rank-chunk between two ranks. Measured for every rank-chunk for every pair of ranks. Measuring RCCT value for each rank-chunk transfer in a collective operation and comparing or binning the values helps understand tail latency in the system
Reduce	Collective operation, where Data of size S on each rank is combined by applying a reduction operator (sum, max, etc.).
ReduceScatter	Collective operation, where Data of size S on each rank is combined by applying a reduction operator (sum, max, etc.) and the result is scattered to the ranks (each rank receives a segment of the result)
Ring Algorithm	Collective Algorithm, where at any given time, each rank sends one message to its logical neighbor within the ring. When the collective group size is p, the previous step is performed p-1 times.
Root / Root Rank	A designated process that plays a special and central role in certain collective operations. The root rank is the rank that initiates data (in the Broadcast or Scatter operations). In a reduce operation, this root rank receives and aggregates data from all other ranks
SUT	System under Test

Term	Description
Warmup Iterations	Iterations, sometimes used to prime the system (load caches, memories, etc.)

4. Not in Scope

The following are to be considered in the future, either as enhancements to this specification, or another specification.

- Hierarchical collectives and concurrent collectives
- Scale-up is not considered. All curated tests assume scale-out only.
- Background traffic
- Impairments

2 Collective Communications Benchmarks

UE Transport can be used to improve the performance of many applications including machine learning (ML) and Artificial Intelligence (AI), High-Performance Computing (HPC), and others. This document focuses on Collective Communications as used in ML training and inference.

Collective communications (CC) are designed to enable efficient communication and synchronization among large number of compute nodes in HPC and AI applications. Collectives are high-level communication abstractions, which are being mapped to individual communication flows by collective algorithms. Libraries developed for CC help scale training processes, optimize resource use, and reduce communication overhead. Because different collectives may have significant implications on performance, such as latency, scalability, etc. it is crucial to document and select the right set of collectives and their specifics to achieve optimal performance.

This document specifies a family of benchmark tests derived from the combinations of various test options, where “options” means combination of the following:

- Scale, topology, collective group size, collective algorithms, collective operators
- Profile and optional features
- Range of parameters to sweep through

5. Collective Communications Benchmark Parameters & Settings

This section defines the different parameters and settings which are eventually chosen in some combination, to create a specific test-case. Because a full description of all possible permutations of a potential test-matrix would be prohibitively large, this document instead defines a way to describe the exact configuration of a particular test as a combination of these options, in both human-readable and machine-readable formats.

The benchmark restricts the set of collective algorithms to a fixed set of well-established algorithms, when reporting results for making comparisons. However, it allows for use of custom algorithms, as long as details of the custom algorithm are provided and the results are clearly marked as such.

This chapter recommends individual parameters and settings. These are intended as guidelines rather than stipulated requirements. A selected subset of recommended combinations is provided in the next section to serve as a guideline and common ground.

6. Collective Workload Parameters

These parameters define the size of the collective, the amount of data transferred, and the collective algorithm and operator used. Bold-faced values indicate recommended subsets.

Table 2-1 - Primary parameters

Collective group size (# of Ranks)	Data size	Collective operation
2, 8, 16, 32, 256, 1024, 2048	256 B, 4 KB, 32 KB, 64 KB, 512 KB, 1 MB , 8 MB, 32 MB , 128 MB, 512 MB, 1 GB	AlltoAll
		Broadcast
		Allgather
		Allreduce (operation)
		ReduceScatter (operation)
		Reduce (operation)

Table 2-2 - Additional/Optional parameters

Additional Parameters	values
Collective Algorithm (algorithms are applicable to specific collectives)	32-parallel, 64-parallel, 128-parallel , user specified, Custom for AlltoAll Double binary tree, Tree for Broadcast Double binary tree, Ring, Halving-doubling[3] , in-network computation, user specified, Custom for Allgather, Allreduce, ReduceScatter, Reduce
Data Type	int(8) (Signed/unsigned) int(32) (Signed/unsigned) int(64) (Signed/unsigned) float(16) float(32) float(64) bfloat16 float8e4m3 float8e5m2

Additional Parameters	values
Collective Operation Reduction type	Sum, Min, Max, Avg
Threads	The number of total threads used to perform a collective.
Num_css	The number of identical sub-collectives, whose combined data size equals the total collective bytes. Some implementations employ multiple sub-collectives to maximize performance.
In-place	True/False
Warmup Iterations	Number of iterations to run before beginning measurements, e.g. 3
Iterations	Number of test cycles (e.g., 5)

7. Curated List of Collective Communications Benchmarks

The preceding sections explained collective communications benchmarks and detailed the various parameters and configurations from which specific test-cases can be declared. This section encompasses a range of reference sets of test scenarios to highlight comparative performance trends while remaining practical for typical research and production settings.

Users are encouraged to submit curated sets that reflect their unique requirements and emphasize the performance dimensions most relevant to their environments.

2.1.1 Curated Benchmarking Sets

Curated sets are compact, vendor-neutral collections of Profile IDs that provide a standardized, reproducible basis for benchmarking collective communications in AI-scale data-center networks. A concise, well-defined configuration set narrows scope, controls variability, and standardizes measurement, analysis, and reporting across implementations and environments.

The curated sets presented here serve as initial reference configurations focused on key performance dimensions. They offer a practical starting point but are not exhaustive; the catalog will evolve as the Performance & Debug Working Group and broader community contribute additional sets aligned to emerging use cases and critical functionalities.

The parameters defined for these benchmarking tests are chosen to reflect the scale and characteristics of data-center infrastructure at the time of publication and will be revised as architecture and requirements change. See section 1.11 for description of Profile ID

2.1.1.1 Latency Baseline

This set evaluates baseline latency in low-contention scenarios, focusing on round-trip time (RTT) and setup costs for small to medium message sizes.

Table 2-3 - Latency Baseline Benchmarks

Profile ID	Collective Op	Ranks	Data Size	Topology	Notes
AR-8R-100KB-1T	All-Reduce	8	100 KB	Single-tier	
AR-16R-1MB-1T	All-Reduce	16	1 MB	Single-tier	
A2A-16R-1MB-1T	All-to-All	16	1 MB	Single-tier	

2.1.1.2 Bandwidth Baseline

This set assesses peak bandwidth saturation in balanced configurations, using large message sizes to stress link capacities.

Table 2-4 - Bandwidth Baseline Benchmarks

Profile ID	Collective Op	Ranks	Data Size	Topology	Notes
AR-16R-256MB-1T	All-Reduce	16	256 MB	Single-tier	
A2A-16R-256MB-1T+p8	All-to-All	16	256 MB	Single-tier	p8 tag represents 8-Parallel mode in All-to-All.
Note: <ul style="list-style-type: none"> Configure iterations to maintain link saturation for at least 1 second. 					

2.1.1.3 Congestion-Control Stress Tests

This set probes congestion handling in high-in cast scenarios, including pathological cases for queue buildup.

Table 2-5 - Congestion-Control Stress Tests

Profile ID	Collective Op	Ranks	Data Size	Topology	Notes
A2A-128R-64MB-2T+p16+OS2:1	All-to-All	128	64 MB	Two-tier	p16 tag represents 16-Parallel mode in All-to-All. The OS2:1 tag represents Oversubscription 2:1.
A2A-128R-64MB-2T+p128+OS2:1	All-to-All	128	64 MB	Two-tier	p128 tag represents 128-Parallel mode in All-to-All.

Profile ID	Collective Op	Ranks	Data Size	Topology	Notes
					The OS2:1 tag represents Oversubscription 2:1.
A2A-128R-64MB-2T+p16+OS10:1	All-to-All	128	64 MB	Two-tier	p16 tag represents 16-Parallel mode in All-to-All. The OS10:1 tag represents Oversubscription 10:1.
A2A-128R-64MB-2T+p128+OS10:1	All-to-All	128	64 MB	Two-tier	p128 tag represents 128-Parallel mode in All-to-All. The OS10:1 tag represents Oversubscription 10:1.
A2A-64R-8MB-1T	All-to-All	64	8 MB	Single-tier	
Note: <ul style="list-style-type: none"> p128 in above cases are intentionally pathological 					

2.1.1.4 Load Balancing Tests

This set of tests is designed to evaluate the transport-layer multipath algorithm. Accordingly, it requires a fully bisectional multi-tier network topology that offers multiple paths between any pair of endpoints. The tests are expected to route at least half of the traffic through the second tier of the network topology. This set tests the UE transport's capability to use path diversity as a load-balancing technique.

Table 2-6 - Load Balancing Tests

Profile ID	Collective Op	Ranks	Data Size	Topology (must have multiple paths)	Notes
RS-64R-1MB-2T	Reduce-Scatter	64	1 MB	Two-tier	Note 1
RS-64R-32MB-2T	Reduce-Scatter	64	32 MB	Two-tier	Note 2

Profile ID	Collective Op	Ranks	Data Size	Topology (must have multiple paths)	Notes
RS-64R-1GB-2T	Reduce-Scatter	64	1 GB	Two-tier	Note 3
Note: <ol style="list-style-type: none"> For data transfers of this small size, the Adaptive Packet Spray mechanism cannot gather sufficient network feedback to dynamically adjust its strategy. Therefore, its performance is expected to be similar to (or slightly better than) Oblivious Packet Spraying, but both will be significantly faster than a single-path transport method. Oblivious packet spray expected to operate. This transfer size provides more network feedback than smaller data sizes yet remains near the limit for fully responding to path congestion. Therefore, UET's adaptive packet spray is expected to be only marginally better than oblivious packet spray. Adaptive & oblivious packet spray may perform similarly. This data size allows the adaptive packet spray to gather enough congestion information to react to path congestion and select less congested paths. Therefore, UET's adaptive packet spray is expected to perform better than oblivious packet spray. Adaptive packet spray should outperform oblivious packet spray. 					

3 Performance Metrics, Measurement & Reporting

8. Measured Values

The following will be measured during a test run:

Table 3-1 - Measured Values

Metric	Comment	Units/Values (if applicable)
Collective Completion Time (CCT)	End-to-end completion time, measured as the maximum time interval from the starting time of any process participating in the collective to the ending time of the entire collective	Milliseconds (ms)
Rank-chunk Completion Time (RCCT)	Measured for every rank-chunk for every pair of ranks. Used for computing the values to be reported by the benchmark and for debug purposes. It is optional to report.	Milliseconds (ms)
Flow Completion Time (FCT)	Measured for every flow for every pair of ranks. Used for computing the values to be reported by the benchmark and for debug purposes. It is optional to report.	Milliseconds (ms)

Since Collective Completion Time is measured end-to-end, the ending time **MUST** be measured after *all* participating processes have finished. This can be enforced via synchronization, such as a barrier before the end timestamp, an ordered atomic update, or taking the maximum of end timestamps from synchronized time sources.

Test reports should clearly describe the measurement methodology. Comparisons of results obtained via differing measurement approaches, or across systems with differing measurement overhead or accuracy, are not accurate and may lead to incorrect conclusions. *Care should be taken to only compare values collected using the same measurement approach, overhead, and accuracy.*

9. Computed and Derived Values

The following will be computed or derived from the metrics gathered during a test run.

Table 3-2 - Computed and Derived Values

Metric	Comment	Units/Values (if applicable)
Algorithm Bandwidth (algbw)	Canonic collective operation performance metric, especially useful on small data sizes when CCT tends to flatline	Gigabytes per Second (GB/s)
Bus Bandwidth (busbw)	Normalized collective operation performance metric that helps to compare performance of collectives with different number of ranks	Gigabytes per Second (GB/s)
P99 RCCT and FCT	Tail values of rank-chunk and flow completion times in one run	Milliseconds (ms)
P50 RCCT and FCT	Median values of rank-chunk and flow completion times in one run	Milliseconds (ms)
P99/P50 RCCT and FCT	Ratios that characterize tail latencies of the rank-chunk and flow completion times in one run	Ratio, >1
Fabric Utilization	Characterizes capability of the system to utilize available bandwidth of the network fabric. Note that in the presence of packet loss this value maybe high while busbw stays low due to excessive retransmissions.	Per-cent (%)

10. Benchmark Report

The test results shall include the following:

- Benchmark results summary
 - Measured results in printed (charts, tables, etc.) and machine-readable form (e.g. spreadsheet, JSON, file, etc.)
- Testbed setup details – topology, hardware and software of equipment and system under test
 - Profile details about DUT configuration
 - Optimizations and other techniques, where relevant and appropriate
- Custom collective algorithm description, if applicable

3.1.1 Benchmark Results Summary

For each workload and test case results shall be reported as the example table below:

Table 3-3 - Benchmark Results Summary

Parameter/Testcases	Iteration 1	Iteration 2	Iteration 3
Testcase ID (ProfileID + additional tags)	AR-16R-4KB-2T-iter1	AR-16R-4KB-2T-iter2	AR-16R-4KB-2T-iter3
Rank	8	8	8
Message size	4 k	4 k	4 k
Collective Operation	Allreduce	Allreduce	Allreduce
Collective Operation parameter (Reduction Type)	Sum	Sum	Sum
CCT (ms)	78.007	78.008	78.107
Algbw (GB/s)	55.06	54.02	55.2
Busbw (GB/s)	48.18	48	48.5
P99 RCCT	1.005	1.006	1.01
P99 FCT			
P50 RCCT			
P50 FCT			
P99/P50 RCCT			
P99/P50 FCT			
Collective algorithm	Ring	Ring	Ring
Threads	1	1	1
Num_css	n/a	n/a	n/a
Num of connections per source and destination pair ¹			
Fabric Utilization (%)	78%	77%	78.5%
Note: 1. A network connection is uniquely defined by a 5-tuple: source IP, destination IP, IP protocol, source port, and destination port.			

3.1.2 Testbed setup details

The following information should be recorded and archived together with any test results:

- Host computer CPU architecture, Operating System distro and version
- Fabric topology:
 - Structure (tiers, # hops, etc.)
 - Line rates between all tiers and hosts
 - Measured baseline RTT (optional - breakdown by switch latency, fiber interconnect delays, etc.)
- NIC (device under test) firmware and software packages and versions, if applicable
- UE transport settings information
- Network switch type and software versions
- Optimizations and other techniques
- CPU/Accelerator information

- The type of, and configuration parameters for, the traffic replay tool, including (but not limited to) request size, CPU thread count, and number of connections for each source-destination pair

3.1.3 Custom collective algorithm description

If a custom algorithm is used, describe it sufficiently well to results can be comprehended. For example:

- Name
- Key features
- Advantages
- Tradeoffs

11. Benchmarking Profile ID

This section defines the **Benchmarking Profile ID** ("Profile ID"), which is a compact identifier for a logical benchmark profile. A profile captures the workload and the abstract fabric class under which it is evaluated, independent of the specific hardware implementation.

The Profile ID is used to:

- Refer to benchmark profiles in curated sets,
- Index and correlate results across implementations, and
- Provide a stable naming scheme for documentation and automation

3.1.4 Profile ID Structure

A Profile ID consists of:

- Four mandatory base fields, and
- Zero or more optional tags.

The general structure is:

```
<Collective>-<Ranks>-<DataSizePerRank>-<Topology>[+<Tag1>+<Tag2>+...+<TagN>]
```

- Base fields are separated by hyphens (-).
- Tags are appended and separated by plus signs (+).

Tags are used to express additional standardized parameters (e.g., algorithm, oversubscription, rails) and optional free-form hints.

Profile ID Examples:

- Minimal, no tags:

```
AR-16R-4KB-1T
```

- With fabric behaviour and algorithm tags:

AR-256R-64MB-2T+OS2:1+ring

- With more details (parallelism degree, rails):

A2A-512R-32MB-2T+OS2:1+ring+p8+dualrails

3.1.5 Profile ID Fields

The field descriptions below include illustrative examples, but they are not exhaustive. Field encodings are not restricted to these examples. When defining Profile IDs, ensure that the encodings are clearly explained so readers can interpret them accurately.

Table 3-4 - Profile ID Fields

Field	Description	Encodings (examples)
Collective Operation	Specifies the type of collective operation being benchmarked.	AR (All-Reduce), A2A (All-to-All), AG (Allgather)
Ranks	Indicates the number of participating ranks in the collective group.	16R, 32R, 256R, 1024R, 4096R
Data Size per Rank	Defines the amount of data processed per participating rank.	4KB, 100KB, 64MB, 256MB, 1GB
Topology	Describes the scale-out network topology class used in the benchmark (cluster / pod fabric).	1T (one-tier), 2T (two-tier), 3T (three-tier), DF (Dragonfly), 2DT (2-D torus), 3DT (3-D torus)

3.1.6 Profile ID Tags

Tags are appended after the base fields and separated by plus signs (+). They provide additional details to the Profile ID without altering the core base fields.

Table 3-5 - Profile ID Tags

Field	Description	Encodings (examples)
Tags (optional)	Atomic descriptors appended with + for additional details such as algorithm choice, N-parallel, oversubscription class, rails, or data type.	Algorithms: ring, tree, dbt, incast Parallelism: p1, p4, p8, p16, p128 Oversubscription: OS2, OS2:1, OS2:1:1 Rails: singlerail, dualrails Data type: fp32, fp16, bf16, fp8_e4m3

3.1.6.1 Tag Examples:

Oversubscription tags:

For tiered topologies, oversubscription tags are expressed per tier:

- 1T – One tier: +OSx
 - x = host-to-Tier-1 oversubscription ratio (x:1).
 - Example: OS2 = 2:1 oversubscription at the single switching tier.
- 2T – Two tiers: +OSa:b
 - a = host $\leftrightarrow$ Tier-1 (host–leaf) oversubscription.
 - b = Tier-1 $\leftrightarrow$ Tier-2 (leaf–spine) oversubscription.
 - Example: OS2:1 = 2:1 host–leaf, 1:1 leaf–spine.
- 3T – Three tiers: +OSa:b:c
 - a = host $\leftrightarrow$ Tier-1 (leaf).
 - b = Tier-1 $\leftrightarrow$ Tier-2 (aggregation).
 - c = Tier-2 $\leftrightarrow$ Tier-3 (core).
 - Example: OS2:1:1 = 2:1 at leaf, 1:1 above.

Rails tags:

The number of independent rails used by a benchmarking profile can be encoded as:

- Singlerail
- Dualrail,
- Multirail, and further.

N-Parallel tags:

N-parallel (often written as p_n) describes how many different destination ranks a given rank is allowed to talk to at the same time in each algorithm step.

- p_1 (N-parallel = 1):
 - Each rank sends to one destination at a time (highly serialized, lower instantaneous load, easier on the network but slower).
- p_8 (N-parallel = 8):
 - Each rank can send to 8 different ranks concurrently in each algorithm step (more outstanding flows, higher bandwidth potential, but more contention).

3.1.7 Profile ID Examples

A few complete examples combining base fields with topology-related and other tags:

1. Single-rail 2T leaf–spine, Ring All-Reduce

AR-256R-64MB-2T+OS2:1+ring

2. Dual-rail 2T leaf–spine, All-to-All with $p=8$

A2A-256R-64MB-2T+OS2:1+ring+p8+dualrails

3. 3T fabric, DBT All-Reduce, FP16

AR-1024R-64MB-3T+OS2:1:1+dbt+fp16

4. Dragonfly-based All-to-All, dual-rail, stress profile

A2A-2048R-32MB-DF+ring+p16+dualrails

5. 3-D torus All-Reduce, non-oversubscribed

AR-2048R-16MB-3DT+T8x8x4+dbt+fp32

3.1.8 Profile ID Notes

- The Profile ID is intended to describe the **benchmarking scenario**, not run configurations, environment, or detailed setup information.
- **Units and Abbreviations:** Use implicit units (e.g., B/KB/MB/GB for data size); keep consistent to avoid ambiguity.
- **Expanding Encodings:** The encodings for fields (e.g., collective operations, topologies, and other parameters) will evolve as new scenarios are incorporated into curated benchmarking sets or additional runs. The examples provided in the table below serve as an initial reference and are not exhaustive, allowing flexibility for future additions as new configurations or requirements emerge.
- TestcaseID = Profile_ID + additional_Testbed_tags

4 Terminology of Collective Communications [Informative]

When benchmarking collective operations, the focus is on measuring KPIs and application-level terminology independent of the underlying transport. This begins by defining the key concepts used at the application level to establish terminology, and then relates those concepts to the underlying transport implementation to clarify their relationships.

12. Application Layer

Collective operations in an AI cluster are performed by a Collective Communication Library (CCL). In HPC clusters, a similar role is performed by MPI; however, this document focuses on CCL, whose terminology may differ.

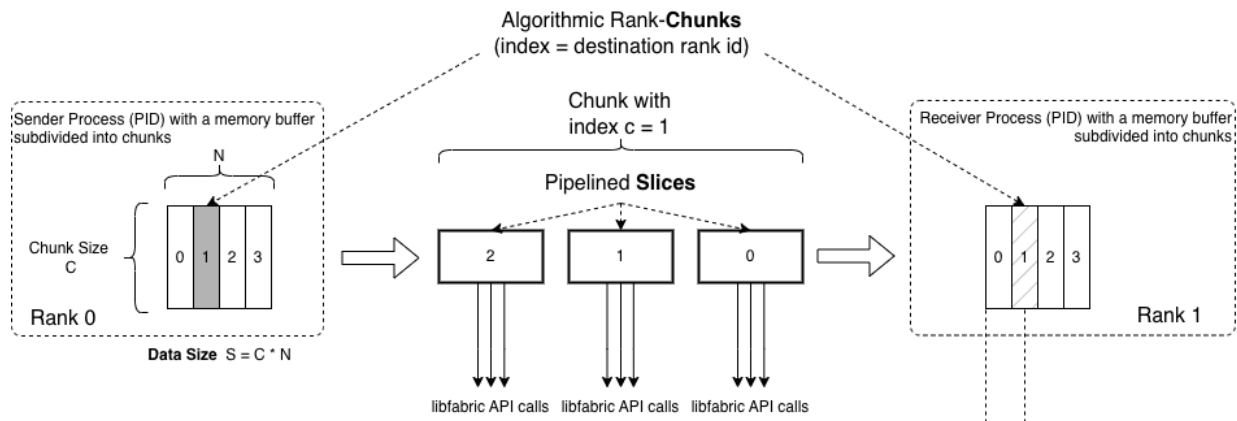


Figure 4-1 - CCL Layer

A CCL's job is to perform a **Collective Operation** (AllReduce, etc.) on data that resides in memory on multiple GPUs. Each CCL instance has an identifier in a cluster, called a **Rank**. The total number of ranks participating in the collective operation could be called **Collective Size N**. The amount of data in the memory of a rank is called **Data Size S**.

Depending on the collective algorithm, parts of that data will need to be moved between the ranks in a fashion choreographed by the CCL. For example, for AlltoAll, it would be divided into N parts, and each part (chunk) of size $C = S/N$ would be sent by each rank to the corresponding N destination rank. These parts can be called **Algorithmic Rank-Chunks**. The number of chunks is determined by the algorithm, and chunks are sequentially numbered on each rank. For AllReduce Ring, chunks of size $C = S/N$ move according to the ring schedule. In these examples, the number of algorithmic rank-chunks per rank is equal to N. This property is algorithm specific.

For operations that perform data reduction, the CCL must not only to receive the chunks from other ranks but also use compute (GPU) to perform the reduction function. To improve efficiency, the chunks could be subdivided into smaller, fixed-size slices, and as long as a slice is received, it could be reduced locally and then forwarded to the next rank. This approach is called a data pipeline, and the fixed-size slices are called pipelined slices.

Pipelined Slice: A fixed-size subdivision of an algorithmic rank-chunk introduced by the CCL implementation to enable overlap of communication and computation. The slice size is library-specific and can vary depending on the algorithm and other parameters.

13. Transport Semantics

The CCL application running on each rank calls into a library (e.g. libfabric) to efficiently copy the slices to the memory of other ranks, using hardware acceleration. The libraries map to the semantic layer of providers such as UE Transport. A key function of the transport semantics layer is to enable addressing of the memory regions of the receiver so that data ends up in the correct place. The way memory addressing works depends on the library and underlying transport protocol. There are common

elements among them, including **buffer offset** that represents displacement from the start of the receiver's collective operation memory buffer.

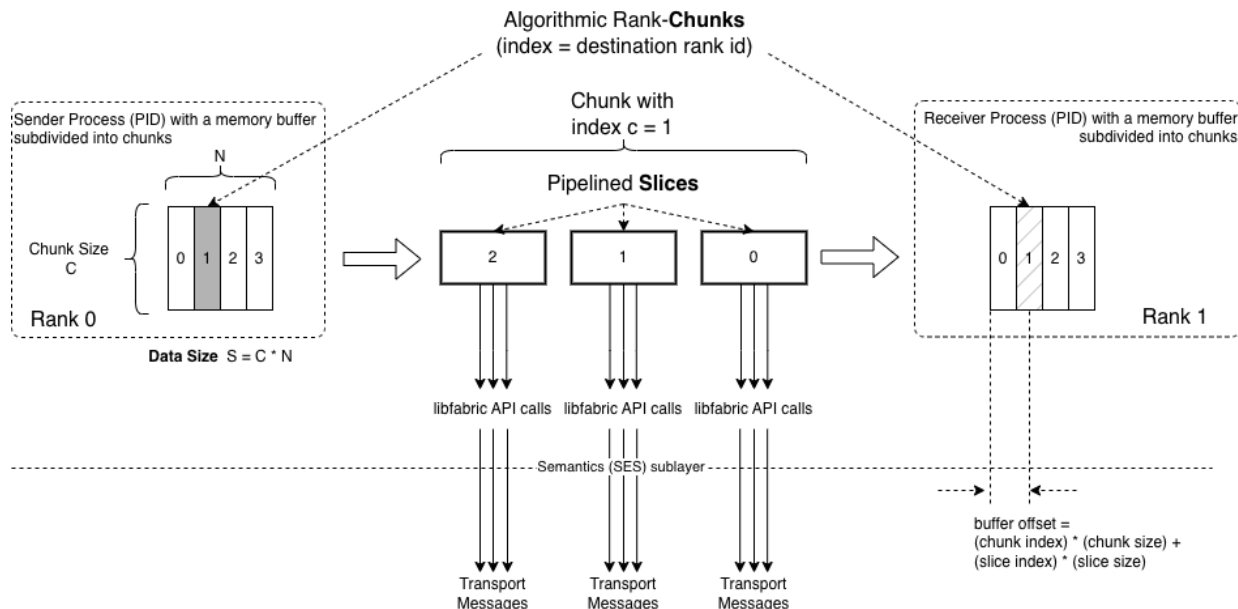


Figure 4-2 - Transport Semantics Sublayer

The block of data transmitted by a single transport API call is defined as a **Transport message** (RDMA message in RDMA-based transports). The amount of data transmitted by a single transport message can be called a transport message size (RDMA message size). An important consideration is if there is one-to-one mapping between pipelined slices and transport messages. Even though it should be possible to transmit each slice with a single message, there are performance optimization techniques observed on practice that split the slices into multiple messages.

14. Transport Protocol

UE Transport's Packet Delivery Sublayer (PDS) uses Packet Delivery Contexts (PDC) connections between sender and receiver. Notable characteristics of PDCs include their ephemeral nature and lack of connection setup or teardown, a short lifespan, and, depending on the transport mode (RUD, ROD and so on), randomization of the UDP source ports by the sender on packet-by-packet basis. This is in contrast to other RDMA protocols like RoCEv2, which relies on long-lived QPairs that need initial setup, re-used for multiple transfers, and enforce in order packet arrival by keeping a fixed 5-tuple combination of IP addresses, a protocol and UDP ports.

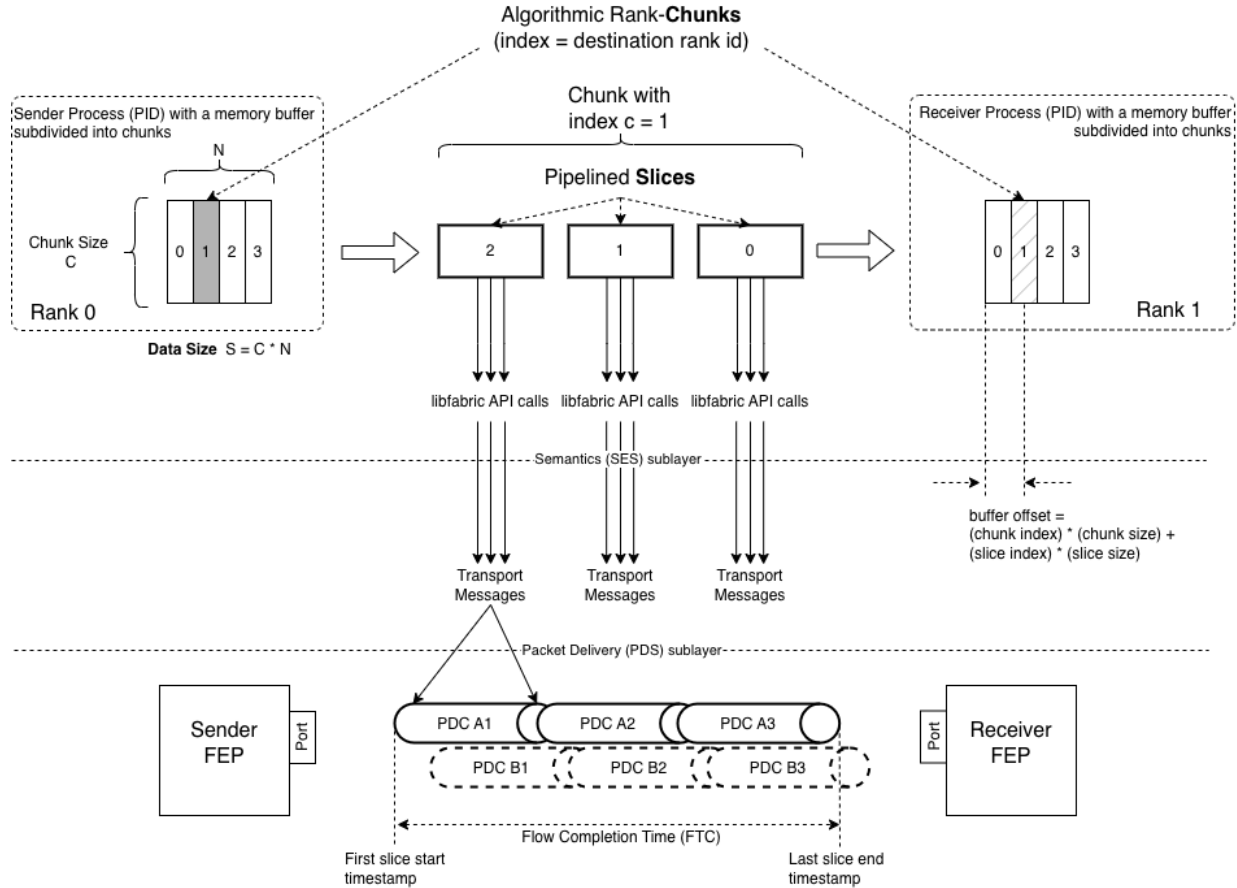


Figure 4-3 - Transport Packet Delivery Sublayer

This difference provides an insight into pipelined slices mapping into transport messages. In RoCEv2 each RDMA message is mapped to a specific QPair. Due to difficulty for a network fabric to load balance traffic with a small number of QPairs with high bandwidth, it is usual to see the slices being split into smaller blocks. That way a higher number of QPairs could be used to make load balancing easier.

Whether similar slice subdivision techniques are required for UE Transport is an implementation consideration and is outside the scope of this terminology definition. However, the definition of transport metrics should accommodate such behavior.

15. Transport Metrics

Since collective operations require all the ranks to finish until the operation can be considered complete, identification of root causes for straggler ranks is critical. To characterize impact of the transport performance, a metric is required to determine whether transfer times of different chunks are approximately uniform or vary significantly. Such a metric helps identify stragglers as well as ranks that take disproportional bandwidth needlessly. Given that certain optimization techniques employ multiple parallel data transfers to address load-balancing limitations, the resulting variability must be considered; individual transfers may experience longer delays depending on the specific path or path set used.

Traditionally, due to in-order delivery requirements, transfers that use the same 5-tuple combination follow the same path. In network engineering, transfers that are mapped into unique 5-tuples are called **network flows**. Flow Completion Time, or FCT, is a well-established metric used to describe duration of such transfers. In RoCEv2, a network flow could be defined as a sequence of transport messages that transfer data from a single algorithmic rank-chunk over the same QPair.

It is recommended to use FCT as a key metric that provides insights into rank-chunk transfer performance between different pairs of ranks.

For benchmarking purposes, to be able to observe improvements provided by the UET, the definition of a flow needs to be expanded such that it is not tied to QPairs but still remains compatible with RoCEv2. The recommendation is to define a **Benchmark Flow** as the **sequence of transport messages that transfer data from a single algorithmic rank-chunk in a serialized manner**. This definition matches behavior of RoCEv2 messages serialized over a single QPair, as well as accommodates a series of UET transfers over one or many PDCs, as long as transport messages belonging to the same algorithmic rank-chunk do not overlap in time with each other. This definition should be practically useful as long as there is serialization of data transfers.

5 References

- [1] GitHub, "ultraethernet / pd-performance," 2026. [Online]. Available: <https://github.com/ultraethernet/pd-performance>. [Accessed 2026].
- [2] GitHub, "NVIDIA / nccl-tests," 2026. [Online]. Available: <https://github.com/ultraethernet/pd-performance>. [Accessed 2026].
- [3] J. Dong, Z. Cao, T. Zhang, J. Ye, S. Wang, F. Feng, L. Zhao, X. Liu, L. Song, L. Peng, Y. Guo, X. Jiang, L. Tang, Y. Du, Y. Zhang, P. Pan and Y. Xie, "EFLOPS: Algorithm and System Co-Design for a High Performance Distributed Training Platform," in 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020.