



OPEN

Compute Project

SUE-T: Base Specification

1.0.0

Effective 8 October 2026

Author: Karen Schramm*, Broadcom

Author: Eric Spada, Broadcom

Author: Lowell D. Lamb, Broadcom

Author: Mark Debbage, Arista

Author: Tom Emmons*, Arista

**Corresponding Authors*

Table of Contents

1.	License	4
1.1.	Open Web Foundation (OWF) CLA	4
1.2.	Acknowledgements	5
2.	Compliance with OCP Tenets	5
2.1.	Openness	5
2.2.	Efficiency	6
2.3.	Impact	6
2.4.	Scale	6
2.5.	Sustainability	6
3.	Introduction	7
4.	Scope	8
5.	Deployment model	8
6.	SUE-T functional overview	10
7.	SUE-T processing flow	12
7.1.	Memory Model and Semantics	13
7.1.1.	Delivery ACK Mode	14
7.1.2.	Operational ACK mode	14
7.2.	Semantic mapping	16
7.2.1.	SUE-T using UET Unified Local	16
7.2.2.	Retransmission support	17
7.2.3.	Wire efficiency target	17
7.3.	Number of queues and scheduling	18
7.4.	Packing	19
7.5.	Reliable packet delivery	19
7.5.1.	RODL mapping	20
7.6.	Load balancing	21
7.7.	Control channel	21
7.8.	Options and tradeoffs	22
8.	SUE-T interfaces	22
8.1.	SUE-T management interface	23
8.2.	XPU command interface	23

8.3.	Network interface.....	23
8.4.	Example XPU to SUE-T command interfaces.....	25
8.4.1.	XPU to SUE-T Signal Command interface.....	25
8.4.2.	XPU to SUE-T AXI 4 Command Interface – AXI semantic mapping.....	26
8.4.2.1	Application interface.....	26
8.4.2.2	SUE-T AXI interface.....	27
8.4.3.	Request / Response	29
9.	Summary.....	30
10.	References	32
11.	Glossary	33

1. License

THE UPDATED DEFAULT CONTRIBUTOR LICENSE AGREEMENT (CLA) IS [OWFa 0.9](#). PLEASE VERIFY THE CORRECT CLA/FSA IS USED AND EXECUTED FOR THIS CONTRIBUTION.

1.1. Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in **Modified Open Web Foundation Agreement 0.9 (OWFa 0.9)**. (As of October 16, 2024) (“Contribution License”) by:

Broadcom and Arista

Usage of this Specification is governed by the terms and conditions set forth in **Modified OWFa 0.9 Final Specification Agreement (FSA)** (As of October 16, 2024) (“**Specification License**”).

You can review the applicable Specification License(s) referenced above by the contributors to this Specification on the OCP website at <https://www.opencompute.org/contributions/templates-agreements>.

For actual executed copies of either agreement, please contact OCP directly.

Notes:

The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS

SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.2. Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback:

Fuwad Khan, Siemens
Mani Subramaniyan, matrix.ai

2. Compliance with OCP Tenets

2.1. Openness

The specification for OCP SUE-T (Scale-Up Ethernet Transport) defines the requirements for scale-up network endpoints, clearly outlining the features necessary for at-scale deployment in AI infrastructure. By publishing these details, we expect silicon and system vendors to develop SUE-T-compliant devices, thus eliminating the need for custom development.

This specification represents an initial version that establishes a functional baseline architecture for SUE-T. As operational experience grows and other OCP members contribute feedback, the specification will evolve, with a commitment to maintaining backward compatibility with version 1.0 to protect prior investments.

Openness is measured by the ability of third parties to build, modify, or personalize the device, platform, or network from this contribution. The SUE-T specification provides a baseline architecture that enables a diversity of implementations, supporting differentiated traffic, hierarchical topologies, and robust congestion management. Openness is achieved by removing barriers that prevent an open networking platform. In this context, the SUE-T specification aims to remove such barriers by providing a consistent definition for scale-up network endpoints.

2.2. Efficiency

The OCP specification for OCP SUE-T (Scale-Up Ethernet Transport) advances efficiency by standardizing the definition of scale-up endpoints, it eliminates costly custom solutions and streamlines deployment. The specification also preserves existing hardware and software investments by maintaining backward compatibility and providing a consistent framework for resilient, differentiated networking. This approach reduces adaptation costs and avoids unnecessary duplication of effort, directly supporting the Open Compute Project's tenet of efficiency by enabling sustainable, cost-effective growth in AI infrastructure.

2.3. Impact

The SUE-T specification aligns with the Impact tenet of the Open Compute Project by facilitating the introduction of a new networking technology poised to transform the industry. For modern AI platforms, there is a pressing need for more efficient and scale-up domains. SUE-T addresses this need by standardizing the definition of scale-up endpoints that support scale-up specific enhancements to Ethernet, including lossless hierarchical topologies, differentiated traffic handling, and robust congestion management.

Our approach leverages widely adopted Ethernet technologies, making scale-up networking practical and accessible without requiring proprietary solutions. This standardization achieves the transformative impact sought by the Open Compute Project, empowering diverse vendors and customers to deploy high-performance AI infrastructure at scale.

2.4. Scale

This specification adheres to the Scale tenet published by the Open Compute Project, which requires that contributions be capable of serving Fortune 100 and large hyperscale customers. With the increasing demands for high-performance, low-latency connectivity in AI and compute platforms, SUE-T meets this scale tenet by defining a standardized approach to scale-up networking using widely available Ethernet technologies.

This approach unlocks compute scaling for hyperscale environments, ensuring that network infrastructure can grow seamlessly alongside compute resources to meet the needs of the largest customers in the industry.

2.5. Sustainability

This specification for OCP SUE-T (Scale-Up Ethernet Transport) complies with the Open Compute Project's Sustainability tenet in several ways.

By providing a standardized scale-up endpoint definition, SUE-T enables data centers to optimize network resources and reduce the need for excess hardware. Enhanced congestion

management, differentiated traffic handling, and lossless hierarchical topologies minimize compute idle time and maximize utilization, thereby limiting the environmental impact associated with overprovisioning and underutilized infrastructure.

SUE-T's robust and flexible specification leverages widely adopted Ethernet technologies, allowing for incremental scaling and interoperability across diverse compute platforms. This flexibility ensures that operators can better utilize existing hardware, extend the lifecycle of network equipment, and avoid unnecessary replacement or customization, supporting responsible resource management.

Overall, the SUE-T specification offers a clear path toward sustainable and efficient data center operations. By reducing operational overhead, enabling scalable growth, and promoting the use of open, interoperable technologies, SUE-T drives innovation while supporting environmental stewardship in the broader technology ecosystem.

3. Introduction

As machine-learning and AI workloads expand in complexity, the need for parallel processing increases. Scaling up an XPU cluster to a rack or multi-rack level is needed to enhance its ability to process larger datasets, train deeper neural networks, handle more simultaneous tasks, reduce execution time, and improve overall system efficiency.

Ethernet offers many advantages for designing such a fabric, from industry leading high-speed links and high-capacity switches to a well-developed ecosystem and well-understood operational methods. Multiple communications industry groups are currently developing networking technologies for AI based networks that extend Ethernet or use some of its components as building blocks.

The Scale-up Ethernet Transport (SUE-T) Workstream in the Open Compute Project (OCP) defines a framework that provides low-latency, high-bandwidth connectivity for XPU scale-up networks based on Ethernet. A SUE-T instance moves memory transactions between XPUs. This transport solution is suitable for use with the following Ethernet datalinks: ESUN as defined in the OCP [1], UFH as defined by the UEC [2], and UDP/IP as defined by the IETF [3-7]. This work references UEC 1.1 (draft), which is technically complete and in the final pre-publication phase. UEC 1.1 is scheduled for publication in mid-October 2026.

4. Scope

SUE-T defines a network endpoint that supports low-latency and high-bandwidth network connections in a manner that allows highly efficient implementations, enabling multiple instances to be instantiated with efficient area and power. Key aspects of this architecture are summarized in Table 1.

	SUE-T Scope	Notes
Number of XPU's	Up to 16K	Single hop for lowest latency Extensible to multi-hop networks
Transaction types	One-sided operations ¹ Memory load-store-atomics Small transfers	Large operations are broken into smaller transactions by the XPU, with a maximum data size of 256B per transaction
Memory architecture	Shared memory	E.g., Partitioned Global Address Space (PGAS) Registration outside of the scope of SUE-T
E2E latency	< 2us RTT	Round trip time (1500B packet, single-hop network)
Traffic Classes	2 to 4	Independent traffic classes, minimum of two traffic classes required to avoid deadlock scenarios

Table 1 – SUE-T Scope

5. Deployment model

The SUE-T specification defines a method for transporting generic command / response transactions over Ethernet. Figure 1 illustrates the SUE-T stack.

¹ In this case, one-sided means that the operation does not involve awareness at the source of the process(es) at the destination. The operation is acknowledged so that the source knows the operation successfully reached the destination.

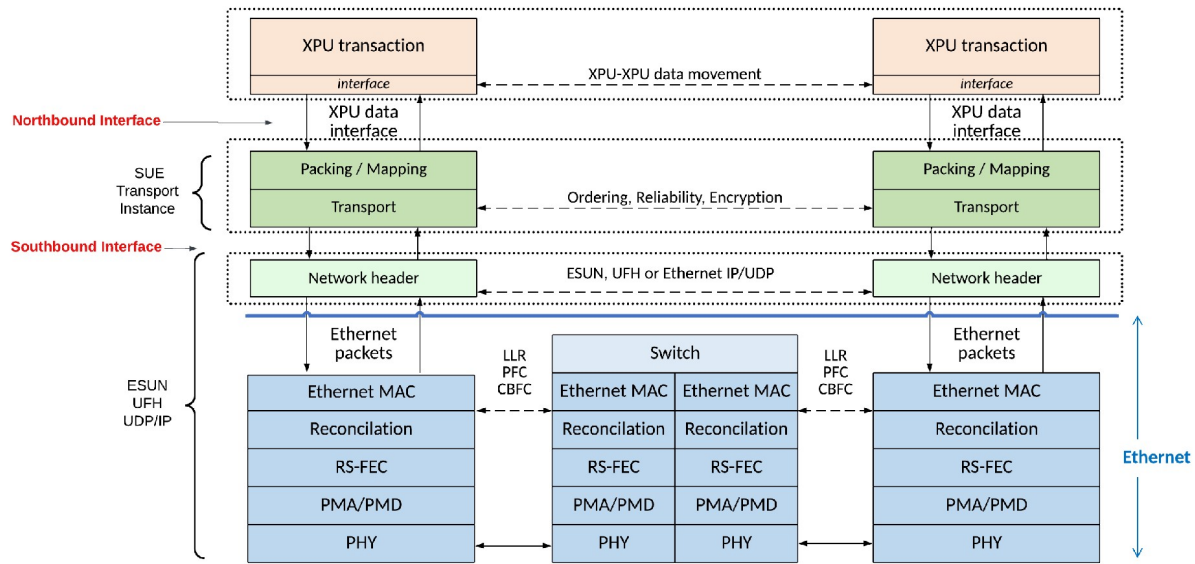


Figure 1 – SUE-T Stack

SUE-T is optimized for a network configuration with a single switch hop, taking advantage of the constraints on latency and congestion that come with this configuration. SUE-T can support larger networks with an associated increase in area and complexity.

SUE-T provides communication between XPUs in a multi-XPU system. Each SUE-T instance provides network connectivity with one or more ports. It is anticipated that multiple instances of SUE-T will be integrated into an XPU, thus scaling to very high aggregate bandwidth.

For example, consider a 1.6T SUE-T instance that supports 2x 800 Gb/s ports or 4x 400 Gb/s ports. An XPU may integrate six of these SUE-T instances for a total of 9.6 Tb/s scale-up bandwidth, as illustrated in Figure 2.

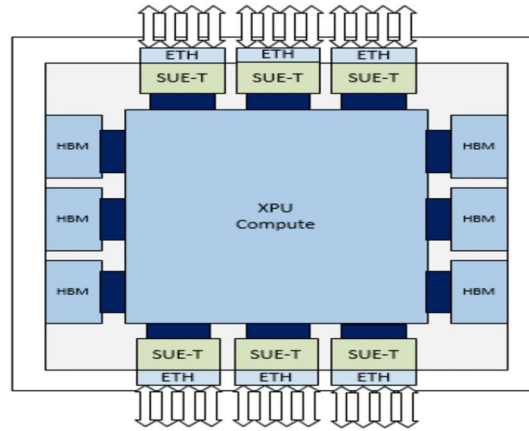


Figure 2 – Example SUE-T Integration

6. SUE-T functional overview

The northbound interface between the XPU and SUE-T varies depending on the XPU architecture. SUE-T defines this interface based on the AMBA® AXI 4 specification [8]. Alternatively, a generic signal interface can be used.

Three options are defined for the layer 2/3 protocol based on OCP ESUN [1], UEC UFH [2] and IEEE Ethernet [3-7]. The layer-4 transport is based on the UEC Unified Local profile [2].

In outline form, the operation of a SUE-T instance is as follows. The XPU interface sends transactions to the SUE-T instance. Each successful transaction includes a request and a response from the destination XPU. Each request includes control fields and, depending on the operation, may include data. For example, the request may be a store operation accompanied by 256B of data or may be a load operation and have no associated data.

Similarly, each response includes control fields and, depending on the operation, may include data. For example, if the request was a store operation, then the response is not accompanied by data. If the request was a load operation, then there will be associated data.

Each XPU-interface request and response includes the remote XPU identifier, opcode, address, length, ordering mode, and other control fields. These are mapped into a command header. SUE-T maps the XPU commands into Ultra Ethernet Transport (UET) AI-local semantic headers.

The mapping layer accepts XPU requests and responses and formats the control and data into a command payload data unit (CMD_PDU). During this mapping process, address translation may be performed using the XPU identifier, address and control fields. The

address-translation service is XPU specific and out-of-scope of SUE-T. An IOMMU or PCIe™ ATS are representative examples of an applicable address translation service.

The destination XPU (dstXPU) and traffic class (TC) are used to assign each CMD_PDU to a queue in the transmit packet buffer and then buffered per destination XPU (dstXPU) and traffic class (TC). A scheduling algorithm is used to select which Packet Delivery Context (PDC) to service next.

Once a PDC is selected, the packing layer pulls up to 16 CMD_PDUs from the corresponding buffer. These are concatenated with the packing headers updated accordingly. The resulting packet PDU is passed to the reliability layer.

The PDU is encapsulated in a reliability header which includes the PDC identifier and a packet sequence number (PSN). The UET defined RODL header, which is 8 bytes in length, is used. Reliability, data integrity and (optionally) encryption are provided at the packet level. Retransmission is optionally supported using GoBackN.

The network layer builds the L2/L3/L4 header stack based on the destination XPU and traffic class. This can be a standard Ethernet header or use an optimized format such as ESUN or UFH. Link Layer Retransmission (LLR) and priority flow control (PFC) or credit-based flow control (CBFC) are used to provide lossless service.

At the destination, the reliability layer tracks expected PSN and generates acknowledgements (ACKs/NACKs) accordingly. After reliability processing, the payload PDU is split into individual command PDUs. The control fields are mapped to the XPU interface as needed, including any required address translation. Each command and its associated data is passed to the XPU.

7. SUE-T processing flow

Figure 3 illustrates the processing flow for SUE-T.

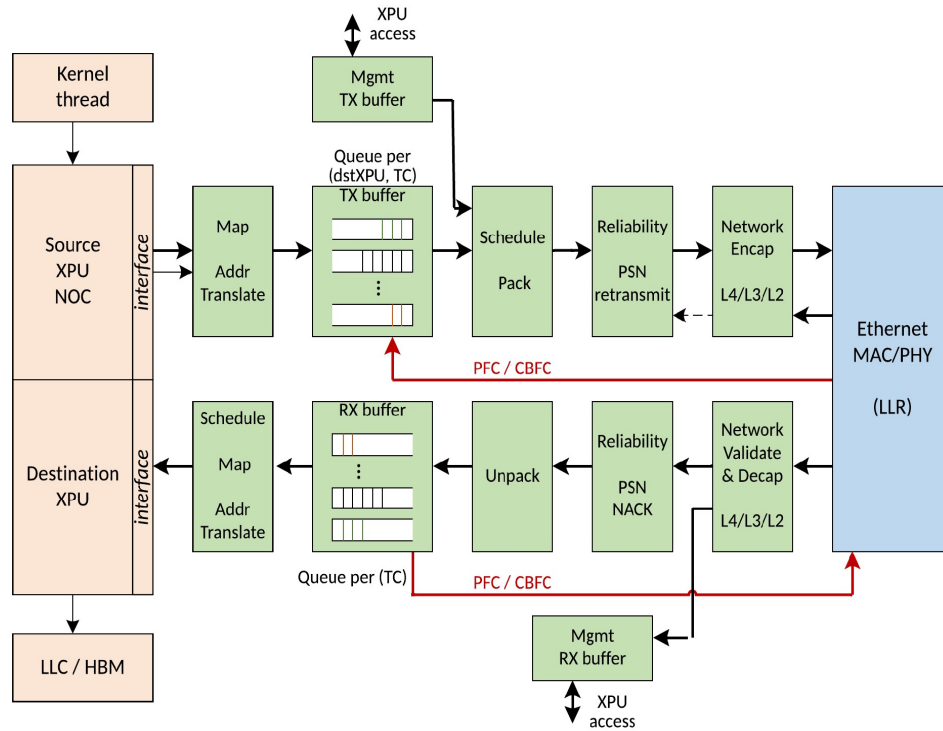


Figure 3 – SUE-T Processing Flow

The SUE-T processing flow follows these steps:

1. XPU kernel issues a request to SUE-T over the XPU transaction interface.
2. SUE-T accepts requests with any associated data, builds a CMD_PDU and passes it to the buffer stage.
3. The buffer stage manages the available buffer as per {destination XPU, traffic class} queues. It determines when a queue is eligible to be serviced based on available buffer, packing level per queue, and flow control state (for example, CBFC input). This state is provided to the scheduler.
4. The scheduler is used (e.g., weighted round robin) between the traffic classes, using arrival order to select among eligible queues in each TC.
5. CMD_PDUs are read from the selected buffer queue and packed into a packet PDU, with packing state in the headers updated accordingly. The resulting PDU is passed to the reliability layer.

6. The packet PDU is mapped to a port based on {dstXPU, TC}. The reliability header, per UET RODL, is added, with the next PSN for that connection.
 - a. If there is an ACK or NACK to be sent for the reverse direction, this state is added to the RODL header in ack_state field.
 - b. The UET CRC is added to the packet to create the SUE-T PDU.
7. A network header is added to the SUE-T PDU. The destination XPU and traffic class are used to look up the required network address fields.
8. The packet is transmitted and arrives at the destination.
9. Ethernet validity checks are performed and any packets destined to the control interface are demultiplexed.
10. The reliability layer checks if the expected PSN for that connection is received. If yes, the PDU is put in the RX buffer and scheduled to be sent to the XPU.
 - a. If an unexpected PSN is received, a NACK is sent to the source and any arriving packets from the source are dropped until the expected PSN arrives.
 - b. If the expected PSN is received, the packet is acknowledged – either immediately or after the XPU response is received based on the ACK mode in use.
11. The RX buffer receives requests/responses, which are then scheduled, unpacked, and sent to the XPU NOC.

7.1. Memory Model and Semantics

SUE-T uses a shared-memory model. Address translation, when required, is handled in the mapping function or may be handled within the XPU.

SUE-T uses one-sided memory semantics. Each command is processed unilaterally under the control of the source, with no indication provided to the destination. Memory registration and management, such as protection domains, are handled outside of SUE-T. Each command generated by the source is completed by SUE-T at the source.

Acknowledgement of XPU commands must be handled in one of two modes:

- (1) Delivery ACK: the ACK means the CMD_PDU was delivered to the destination XPU; the destination XPU response is transported back to the source independently.
- (2) Operational ACK: the ACK means the destination XPU processed the operation; the ACK carries the destination XPU response; This means SUE-T tracks and associates each XPU response with the corresponding request.

There is a trade-off between SUE-T complexity and wire efficiency to consider when selecting the ACK mode. UEC Unified Local profile supports both ACK modes.

7.1.1. Delivery ACK Mode

Delivery ACK mode is simpler for SUE-T: all commands and responses are completely opaque. SUE-T accepts a command, sends it and the acknowledgement indicates the command was delivered. In this way, the transmit and receive path in SUE-T are largely independent. If the XPU protocol expects a response (e.g., at the AXI 4 level) that ‘response’ is carried as a command from the remote XPU. The XPU interface then associates the ‘response’ with the original command. See Figure 4.

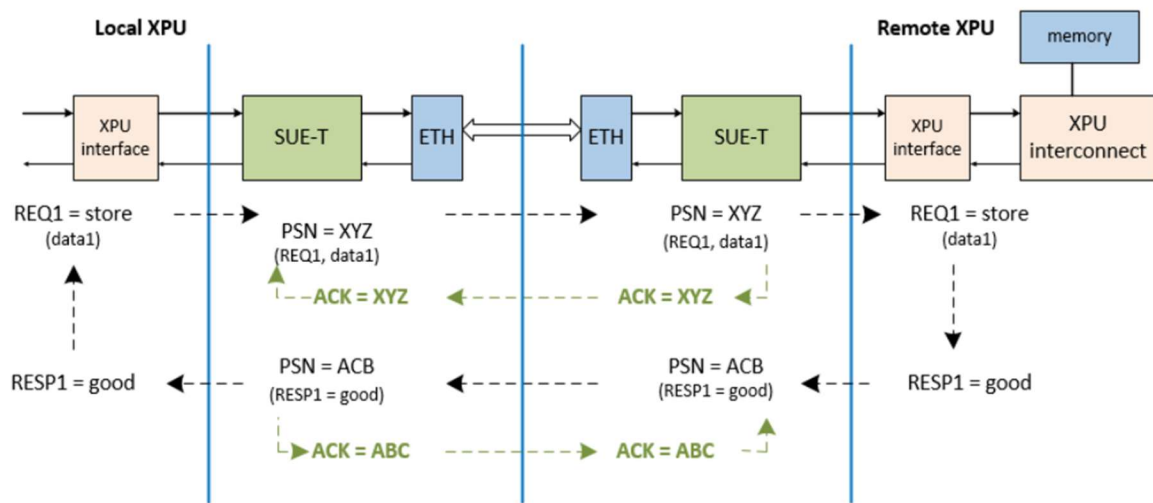


Figure 4 – Flow for Store using Delivery ACK.

ACK = XYZ indicates the remote SUE-T delivered the request to the XPU. This does not generate a response to the XPU interface. The remote XPU interface generates the XPU-layer response which is carried as a command by SUE-T. The RESP1 payload in the packet with PSN = ABC is the XPU-layer response to REQ1. SUE-T acknowledges that packet with ACK = ABC.

Note: when bidirectional traffic is present, the ACK is carried in the header of request packets sent in the reverse direction, resulting in minimal bandwidth overhead.

7.1.2. Operational ACK mode

Operational ACK mode adds state in SUE-T to track each command and associate a response from the XPU with the original command. This can be done using a label or by requiring ordering over the XPU-to-SUE-T interface. For example, in AXI 4-based

implementations, this is accomplished by using the AxID signal. This approach is very efficient for write and non-fetching atomics where the majority of responses are 'success' and carrying 8 to 16 bytes of command state for each response is not bandwidth optimized. Operational ACKs compress these responses into a single ACK PSN. The ACK state is carried in the header of request packets in the reverse direction. SUE-T then generates the well-known 'success' response at the source.

If the response carries data, e.g., for load and fetching atomic operations, a dedicated ACK packet carries the data. If an error occurs, an ACK packet carries the error indication.

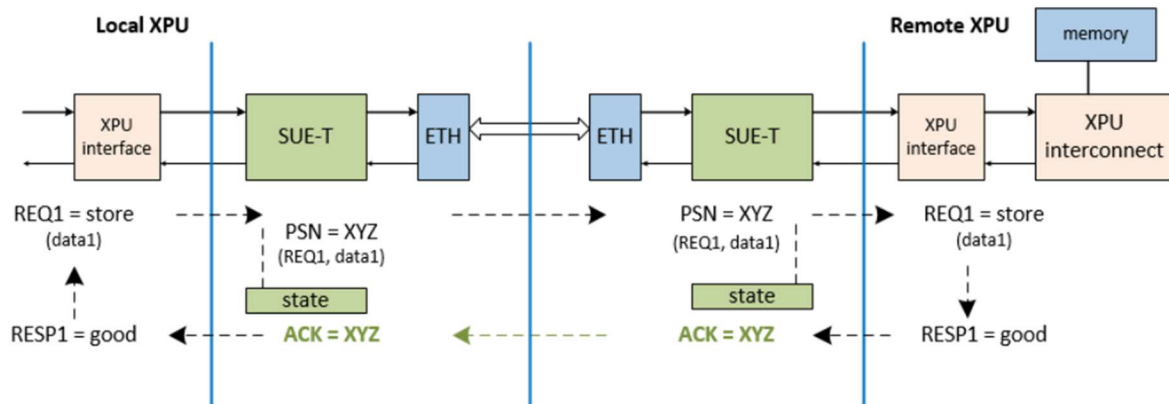


Figure 5 – Flow for Store using Operational ACK

The SUE-T at the local XPU saves state associated with each command in PSN = XYZ. When SUE-T at the remote XPU receives PSN = XYZ, it unpacks the CMD_PDU and sends the command and data to the XPU interface, saving state per {PSN, CMD_PDU}. The ACK is held until the XPU interface generates a response for all commands in the packet. If all the commands are 'success' with no data, the ACK = XYZ is sent. When the local SUE-T receives the ACK, it checks the state for all CMD_PDUs that are acknowledged and generates the appropriate response to the XPU interface for each. Storing the state in SUE-T effectively replaces sending that state over the wire.

Note, when bidirectional traffic is present, the ACK is carried in the header of request packets sent in the reverse direction, resulting in minimal bandwidth overhead. This flow results in a wire efficiency greater than 90% when two more CMD_PDUs are packed in a packet over UFH with 16-bit addresses, as compared to less than 80% wire efficiency using delivery ACKs with a 16-byte response command.

The state stored per command depends on the XPU interface. The identifier for the operation as well as any XPU specific information that is unique per transaction response. If an error occurs or if the response includes data, a dedicated ACK packet is used (versus carrying the ACK PSN in the header).

The complexity (amount of state) and wire efficiency of both delivery ACK and operational ACK modes are affected by choices in the XPU interface and the extent to which SUE-T is aware of the XPU transactions. For example, if the remote XPU interface immediately generates a response = success for all store and non-fetching atomic operations, the amount of state required in the SUE-T block is minimal.

If global observability is supported, the XPU may request the transaction be acknowledged after the data is placed in memory. This is done by setting the 'global observability' control field in the command. If this field is set, the XPU response is held until the data is placed. Global observability can be support with either ACK mode.

7.2. Semantic mapping

Semantic mapping refers to the set of operations supported, and how the associated control and data fields are mapped on the wire. SUE-T uses semantics per the UEC Unified Local profile. This enables use of open-source network management and monitoring tools and interop with other devices, such as network attached storage.

The reliability header, as described in Section 7.5, carries a next header field (`next_hdr`) which defines the type and format of the semantic command fields. Refer to the UEC 1.1 specification [2] for the definition of UET. The following text outlines the UET semantic headers used in SUE-T.

7.2.1. SUE-T using UET Unified Local

The Unified Local profile defines two header formats to carry load, store and atomic operations and the associated responses:

`UET_HDR_REQUEST_LCL` is a 12-byte or 8-byte semantic header for each transaction. The 8-byte header is a compressed variant that carries a 32-bit address defined as an offset to the previous address. `UET_HDR_REQUEST_LCL` provides the option to carry up to 16 bytes of additional XPU-specific state. Up to 16 transactions may be packed into a single packet.

Unified Local request opcodes support the following operations:

- 0x0 nop
- 0x1 write (store)
- 0x2 read (load)
- 0x3 atomic
- 0x4 fetching atomic

An implementation may extend the set of operations using vendor-specific opcodes. UET reserves UET_VENDOR_DEFINED opcodes 0x30 through 0x3E for vendor-specific operations. This approach uses the base UET format for packing (mm), version, global observability (dc), jobid (VM isolation) and data length. The remaining XPU specific control fields are mapped to the remaining bytes in the semantic header.

UET_HDR_RESPONSE_LCL uses 8-bytes of response state for each transaction. Up to 16 responses may be packed in a single acknowledgement packet.

A single Unified Local response opcode is used:

- 0x2 Response with data; set payload_length = 0 if no data

Unified Local supported return codes are:

- 0x1 RC_OK, the operation was successful
- 0x1F Undeliverable, indicating an error at the destination
- 0x30 – 0x37 vendor defined – these can be customized

The lowest complexity SUE-T semantic implementation supports:

- Write and non-fetching atomic only
- Completion ACK mode where XPU interface immediately ACKs all operations

SUE-T maps between the signals on the XPU-SUE interface and the semantic header format on both the transmit and receive paths.

7.2.2. Retransmission support

When retransmission is supported, CMD_PDUs are held until acknowledged and the buffer must be sized for round trip and remote processing time.

7.2.3. Wire efficiency target

More buffering increases the opportunity to pack multiple CMD_PDUs in a single packet, which increases wire efficiency. An implementation could choose to minimize buffering by only

packing CMD_PDUs if they arrive at the SUE-T instance consecutively. Alternatively, an implementation may choose to include buffering for up to N CMD_PDUs per queue.

7.3. Number of queues and scheduling

A solution that scales to N XPU's with K traffic classes may include buffering for $N * K$ queues. This allows a dedicated queue per {destXPU, TC}. Alternatively, the buffer may be designed for fewer queues and dynamically allocate buffer resources. The dynamic solution may be smaller but may experience blocking if the traffic pattern is highly distributed small messages.

SUE-T requires two traffic classes to support a single virtual connection (VC) because requests and responses must be mapped to different TCs to avoid deadlock. Implementations requiring multiple traffic classes for transaction (i.e., high and low priority transactions) require more than two TCs.

The selection of the scheduler discipline is left to the system designer. Depending upon the choices the designer has made for system scale the scheduling discipline needs to be selected.

The XPU to SUE-T interface may be higher bandwidth than the SUE-T to Ethernet interface. Flow control is used on a per traffic class basis to prevent over running the buffer in SUE-T. This speed mismatch may cause CMD_PDUs to accumulate in the queues.

The transmit scheduler monitors the queues to determine which to service next. If a destination queue has a single transaction, the transaction is sent. Transactions are not delayed for reasons of packing. SUE-T will pack up to the configured limit:

- 16 CMD_PDUs
- Maximum payload length – sum of all store/atomic payload (2K, 4K or 8K)
- Maximum response payload length – sum of all load/fetching atomic payloads (2K, 4K or 8K)

Figure 6 illustrates the queues and scheduler. The XPU sends five transactions A through E. The transactions are queued per {destination, traffic class}. The scheduler must be work conserving; it transmits the CMD_PDUs in next available queue. In this example, the first available queue is the one with transaction A. After selecting the queue with A, the scheduler selects the queue containing B. The scheduler then selects the queue with C in it. Transaction E is in the same queue, so C and E are both selected. The scheduler then selects the queue with D. The scheduler does not wait for a queue to accumulate transactions to a particular size, but opportunistically packs transactions together to optimize packet-on-wire efficiency and minimize latency of transactions.

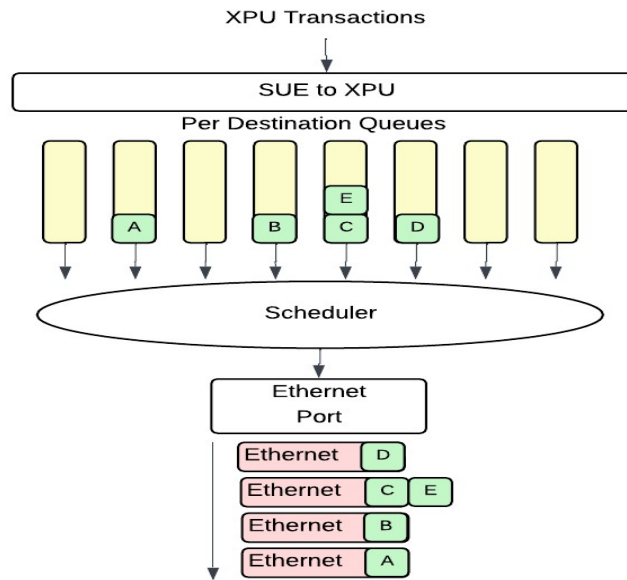


Figure 6 – Transmit queues and scheduler.

7.4. Packing

Refer to the UET 1.1 specification for a detailed description of packing multiple CMD_PDUs (called messages in UET) into a single packet.

On the transmit path, when the scheduler selects a queue, the set of CMD_PDUs that are available and fit within the configured maximum payload of a single packet are packed together. The semantic headers are updated, setting the mm and ac fields accordingly. The resulting packet PDU is passed to the reliability layer.

On the receive path, the payload is split into individual CMD_PDUs which are stored in the receive buffer.

7.5. Reliable packet delivery

Aligned with the UEC Unified Local profile, SUE-T uses the RODL reliability protocol in UET. RODL provides reliable, ordered delivery and is well suited to scale-up with a single hop network.

Ordered delivery is chosen for area and complexity savings. Unordered reliable services require packet-tracking bit maps, selective acknowledgements and selective retransmission.

RODL retransmission is defined as GoBackN. The state for each connection is significantly smaller for ordered connections using GoBackN.

Retransmission is optional to allow very compact SUE-T instances. Removing retransmission reduces the required buffering and state. The trade-off is loss of a single packet is fatal to the connection. Removing retransmission may be a reasonable trade-off in networks supporting link level retry (LLR).

SUE-T requires a lossless network service. The penalty for packet loss is high, potentially requiring significant retransmission bandwidth (GoBackN) or causing a fatal error on the connection (if no retransmission). PFC or CBFC is used to provide lossless service on all SUE-T traffic classes.

A multi-port SUE-T instance may provide additional network resiliency with integrated failover. When a network port fails, a connection can be moved to a different port. This concept is left as future work for the SUE-T project.

Congestion control is not required for single switch hop networks. If the XPU cluster size requires a three-switch hop network, congestion control may be added using CSIG, or by moving to UET ROD and using an algorithm like NSCC or RCCC. Congestion control for multi-tier scale-up is left for future work.

RODL uses dynamic connections called Packet Delivery Contexts (PDCs). If the XPU cluster size is too large to dedicate a PDC to each {dest XPU, TC}, then dynamic connections are efficient. In smaller systems with one PDC per remote XPU, the PDCs are effectively static. RODL can be directly configured as static connections using 'reserved PDCs'.

7.5.1. RODL mapping

This section describes a method for mapping packets to RODL Packet Delivery Contexts (PDCs). A PDC is a connection between two XPUs. State is maintained per PDC, including next packet sequence number (PSN), acknowledgement PSN, timeout timers, etc. Each PDC carries a PDCID field which uniquely identifies the connection.

XPU transactions are queued in the transmit buffer per {dstXPU, TC} with each queue associated with a RODL PDC. Refer to Section 10.3 for a description of the AXI fields used to determine {dstXPU, TC}.

The method for allocating PDCIDs and associating a PDC with a transmit buffer queue is implementation specific. If there are dedicated PDCs for each {destXPU, TC} then the PDCID could be set to equal {dstXPU, TC}. Alternatively, the PDCID can be dynamically assigned.

If the XPU cluster size is too large to dedicate a RODL connection to each {dest XPU, TC}, dynamic connections can be used to enable area-efficient designs. In smaller systems with one RODL connection per remote XPU, the connections are effectively static.

7.6. Load balancing

It is presumed that there will be multiple instances of SUE-T connected to an XPU. Load balancing across the SUE-T instances can be handled in software or by including a hardware block in the XPU that distributes the operations across the SUE-T modules. Each SUE-T instance provides congestion state to enable this option. Refer to Figure 7.

Within the the SUE-T, load balancing can be enabled when using two or more ports. In this case, SUE-T will dynamically assign each packed group of operations to a port based on available bandwidth.

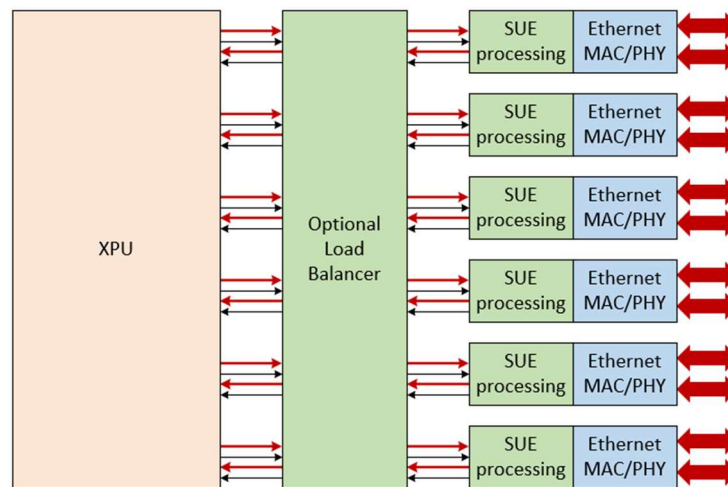


Figure 7 – Example XPU Load Balancer

7.7. Control channel

This optional capability allows the XPUs to exchange low-rate messages, primarily intended to support diagnostics and basic queries.

7.8. Options and tradeoffs

SUE-T defines a framework for building an efficient scale-up network over Ethernet. As mentioned in the previous section, there are a set of trade-offs between features and area/complexity of the SUE-T instance. This section summarizes the decisions in defining a specific SUE-T implementation.

Primary feature choices for SUE-T:

- Semantics operations
 - Limit operations to write and non-fetching atomics, simplify implementation
 - Include read and fetching atomics for added functionality and flexibility
- Transmit CMD_PDU buffer size
 - Include larger buffer to enable more packing and improve wire efficiency
 - Minimize buffer size to reduce area, reducing opportunity to pack with trade-off on wire efficiency
- Transport retransmission (GoBackN) or no retransmission (rely on LLR)
 - Retransmission requires additional buffering, larger area for improved resiliency
- ACK mode
 - Delivery ACK mode for less managed state, lower wire efficiency
 - Operational ACK mode for higher wire efficiency, more local state
- Dynamic connections
 - Maintain state for every remote XPU for effectively static connections
 - Support fewer PDCs than remote XPUs to save area using dynamic connections, e.g. 1K PDCs for 8K XPUs

Consider two examples: SUE-lite and SUE-full.

- SUE-lite: no retransmission, LLR, write only, delivery ACK
 - Provides smallest area implementation
- SUE-full: retransmission, operational ACKs, dynamic connections
 - Provides highest wire efficiency and reliability

8. SUE-T interfaces

Each SUE-T instance has three interfaces, as illustrated in Figure 8:

- XPU command interface
- XPU management interface
- Ethernet interface

A description of each of these is provided in the following subsections.

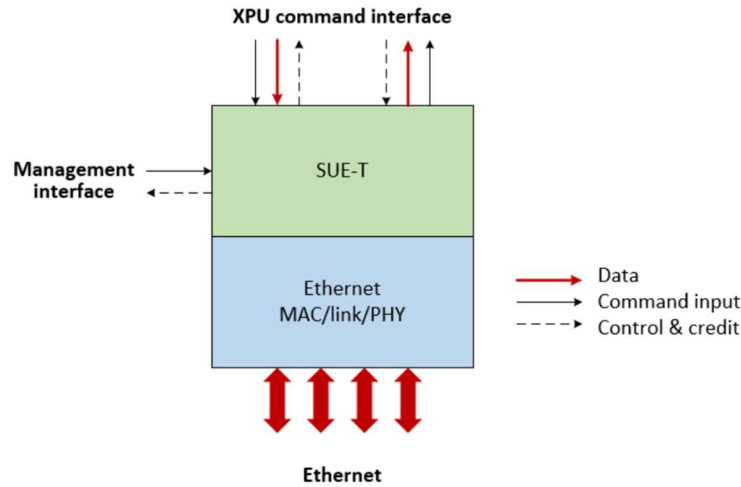


Figure 8 – SUE-T Interfaces.

8.1. SUE-T management interface

The management interface is used to configure and query the SUE-T instance. An example protocol for this is an AXI target interface. The set of control and status registers is unique per SUE-T implementation.

In addition to configuration and status, the control interface may provide the option to send and receive packets. The XPU builds the full packet and triggers transmission. SUE-T adds the Ethernet FCS. When a control packet arrives, it is stored in a buffer which can be queried by the XPU. This control packet transport is unreliable.

8.2. XPU command interface

The XPU to SUE-T interface is used to transfer commands. The protocol can be adapted to the XPU architecture. Two example interfaces are:

- Signal Interface using a FIFO interface with credits
- AXI 4 interface

SUE-T expects XPUs to use these directly or adapt them to meet specific needs.

8.3. Network interface

SUE-T is defined to run over Ethernet networks. SUE-T may run over:

- Ultra Ethernet Consortium's Unified Forwarding Header (UFH)
- OCP Ethernet for Scale Up Networks (ESUN)
- UDP over IP over Ethernet

The destination network address is typically configured in the SUE-T instance and looked up using a tag from the XPU. For example, the field which identifies the destination XPU ID may be used to lookup the destination MAC or IP address in ESUN or Standard Ethernet. In UFH, the destination XPU ID may be used directly.

When using UDP/IP/Ethernet, the presence of the UEC transport is identified using the UDP destination port. UDP destination port = 4793 identifies UEC.

When using UET, the present of the UEC transport is identified using the Ethertype. An Ethertype = 0xE790 indicates UEC.

When using ESUN, the presence of the UEC transport is inferred by network wide configuration.

The type of UEC transport (UET), e.g. RODL, is determined by the pds.type field in the UET header.

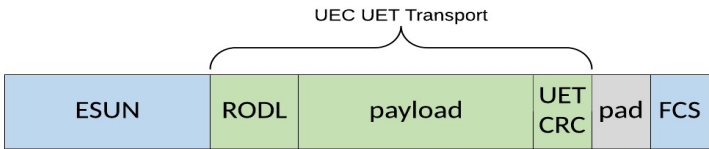


Figure 9 – SUE over ESUN

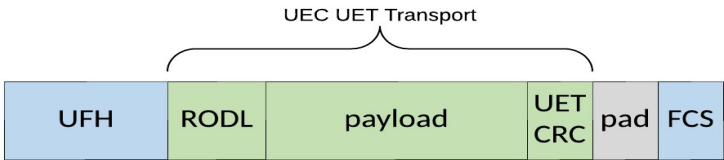


Figure 10 – SUE over UFH-16b

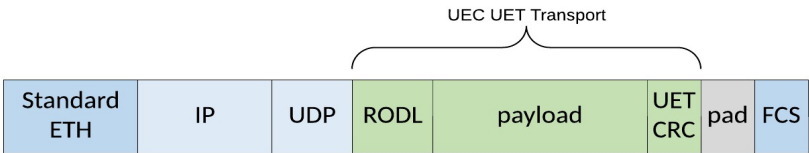


Figure 11 – Standard IEEE 802.3 Ethernet UDP/IP Format

8.4. Example XPU to SUE-T command interfaces

This section discusses the various command interfaces between the XPU and the SUE-T instance.

8.4.1. XPU to SUE-T Signal Command interface

The signal command interface is essentially a FIFO interface using credits. The XPU writes a command to the SUE-T, as well as any associated data. SUE-T generates flow control on a per {destination, VC} basis and on an aggregate rate basis using a credit mechanism. To initiate transfer of data, the XPU sends the information listed in Table 2 to SUE-T over the command interface. Input refers to signals from XPU to SUE-T. As packets arrive, SUE-T sends the transactions to the XPU over the command interface. Table 3 lists the SUE-T to XPU signals on this interface.

Field Name	I/O Type	Size (bits)	Description
tx_cycle	Input	4	Data cycle input number, transfer up to 256B of data over up to 16 clock – number of cycles is implementation specific depending on data bus width
tx_valid	Input	2	Type of operation: none, command only, command with data
tx_vc	Input	2	Virtual channel
xpuid	Input	10 - 16	Remote XPU identifier, support 1K to 16K XPUs
tx_control	Input	M	Control fields, processed and packed into transaction to be sent to remote XPU – this carries command and length information. The size, M, is implementation specific. The relevant fields may be mapped to the semantic extension header (aka, AXI AxUSER).
tx_data	Input	N	Data, serialized over multiple cycles depending on N; for example, if N = 256 bits (32B) then 8 cycles are used to transfer 256B.
tx_err	Input	1	Uncorrectable error on interface; for example, used to support ECC generation and checking when die to die interface is used.
tx_credit	Output	6	Credit & valid signals per VC
tx_reset_in	Input	1	Reset – clear FIFO and reset credits
tx_reset_out	Output	1	Indicates SUE-T instance is in reset

Table 2 – XPU to SUE-T Signal Command Interface

Field Name	I/O Type	Size (bits)	Description
rx_cycle	Output	4	Data cycle input number, transfer up to 256B of data – number of cycles is implementation specific depending on data bus width
rx_valid	Output	2	Type of operation: none, command only, command with data
rx_vc	Output	2	Virtual channel
xpuid	Output	10 - 16	Remote XPU identifier
rx_control	Output	M	Control fields, processed and packed into transaction to be sent to remote XPU – this carries command and length information. The size, M, is implementation specific. The relevant fields may be mapped to the semantic extension header (aka, AXI AxUSER).
rx_data	Output	N	Data, serialized over multiple cycles depending on N; for example, if N = 256 bits (32B) then 8 cycles are used to transfer 256B.
rx_credit	Input	6	Credit & valid for cmd and cmd+data
rx_reset_in	Input	1	Reset
rx_reset_out	Output	1	Indicates SUE-T is in reset

Table 3 – SUE-T to XPU Signal Interface

8.4.2. XPU to SUE-T AXI 4 Command Interface – AXI semantic mapping

This section defines how AXI transactions are mapped to SUE-T network semantics.

8.4.2.1 Application interface

The application may use any appropriate memory read/write transactional interface at the SOC/NOC level. A typical interface is AXI 4/AXI 5 using any desired features from the relevant AXI specification.

The SUE-T IP provides a simplified AXI 4 compliant interface with the ability to convey additional semantic data through the signal AxUSER (defined in the following section). When integrating the SUE-T IP, the integrator must define a mapping from the application interface to the SUE-T AXI4 Interface. This typically consists of simple combinatorial logic on the manager side to construct AxUSER and reverse logic on the subordinate side to deconstruct AxUSER.

In more detail this mapping requires:

- Mapping to signal widths for AxADDR (60 bits), AxID (16 bits) and AxUSER (128 bits), all of which are defined in the following section. The SUE-T interface defines maximum sizes for these signals, but an application may use a subset of the bits. AxUSER size can have a significant impact on packet encoding efficiency for small transactions (such as 256B read/write or smaller).
- Mapping logic to encapsulate application-defined signals into the SUE-T AxUSER signal at the sender and to decapsulate application-defined signals from the SUE-T AxUSER signal at the receiver. AxUSER is delivered by SUE-T from source to destination without modification, and the contents of AxUSER are opaque to the SUE-T IP (i.e. not interpreted by SUE-T). AxUSER is used for additional semantic information comprehended only by the application layer.

8.4.2.2 SUE-T AXI interface

This is a minimal AXI 4 Interface. Most AXI features are opaque to SUE-T and simply conveyed from source to destination without interpretation. AXI AW, W, B, AR and R channels are supported. The key signals are:

- ACLK, ARESETn, AxVALID, AxPENDING, AxCREDIT, AxSIZE, AxLEN, xLAST - these are local AXI signals and are not required in the network protocol.
- WSTRB - this is required as an input for AXI4 compliance. The approach is to support WSTRB on both manager and subordinate interfaces but constrain WSTRB to only support a contiguous range of enabled bytes. This matches the use case and aligns with conventional remote memory write semantics defined using a base address and length. This limitation means that WSTRB is not required in the packet representation of the AXI transaction. Transactions with unsupported WSTRB values (i.e. non-contiguous) are treated as local errors.
- xRESP - AXI response codes are mapped to/from SES response codes.
 - OKAY: RC_OK
 - EXOKAY: not supported
 - SLVERR (subordinate error): RC_VENDOR_DEFINED[0]
 - DECERR (decode error): RC_VENDOR_DEFINED[1]
- AxADDR - 60 bits.
- AxID - 16 bits, the number of used bits is configurable. AxID is used to determine the connection and is sized for the maximum number of connections/destinations. AxID represents XPU ID in the AXI interface. As specified by AXI, ordering is provided per AxID value, but

separately for reads and writes. In other words, writes using the same AxID are ordered, reads using the same AxID are ordered, there is no guaranteed ordering between reads and writes (whether using the same AxID value or not), and there is no guaranteed ordering when using different AxID values. The maximum AxID width is constrained to the number of connections supported by the SUE-T IP implementation. An implementation may use a mapping table from AxID to PDCID on the send side and use connection state to map from PDCID to AxID on the receive side.

- AxUSER - 128 bits, the number of used bits is configurable with supported sizes of 0, 32, 64 and 128 bits. SUE-T does not interpret AxUSER values. The AXI interface does not convey the number of AxUSER bits used. Instead, this should be configured into the SUE-T via an implementation-specific mechanism. One option is to configure this separately for write requests, read requests, write responses and read responses so that AxUSER can be sized appropriately for each of these.
- Only AXI read and AXI write are supported - additional transaction types can be defined at application level, as required, and mapped down to read or write and transported through SUE-T using opaque AxUSER fields. For example, fetching atomics can be mapped to read and non-fetching atomics can be mapped to writes and atomic opcode and datatype can be coded using AxUSER fields.
- AXI burst transactions support up to 4KB of data. SUE-T supports the full range of AXI transaction lengths for efficiency and to simplify IP integration. An AXI transaction must not cross a 4KB page multiple.
- The SUE-T AXI interface does not provide direct support for caching attributes, protection attributes, regions, QoS, etc. but an application can support these and encode them in AxUSER.

The mapping onto UEC SES transactions is as follows:

- Only UET_HDR_REQUEST_LCL and UET_HDR_RESPONSE_HDR_LCL header formats are used.
- Only UET_READ, UET_WRITE and UET response opcodes are used.
- AxID is mapped to PDCID, and the AxID value is not present in the semantic header. The mappings between AxID and PDCID are bijective - this means that the mapping is reversible so that the connection state on the receive side can be used to map PDCID back to AxID.
- The 8-bit jobid is supported. This value can be configured into PDC connection state and is inserted on the send side and validated on the receive side.

- The version, delivery complete (dc) and initiator error (ie) fields are used in the standard way defined by UEC.
- The SES 60-bit address is used to carry AxADDR. When the application requires fewer than 60 bits for AxADDR, there are two options for the upper bits:
 - They can be set to 0.
 - They can be used to transport opaque information from sender to receiver (additional to AxUSER). The application logic is responsible for constructing and deconstructing the 60-bit AxADDR value. The split between opaque data and address is not known to SES and not encoded in the SES wire protocol.
- The number of AxUSER bits used is configured as 0, 32, 64 or 128 bits. The used bits are present in the least significant bits of the 128-bit AxUSER signal with any unused most significant bits having reserved behavior. AxUSER maps into the Extension Header feature in the SES LCL header format. The Extension Header follows the address and precedes the payload per transaction or per packed transaction. The supported AxUSER bit sizes match the 0B, 4B, 8B and 16B Extension Header Lengths provided by UEC.
- SES transaction packing using the “mm” bit is supported.
- SES address compression using the “ac” bit is supported.
- Payload length is from 0 to 4096 bytes and held in the 14-bit extended request_length field of write requests or the 14-bit payload_length field of read responses. The ac=1 format supports only a 12-bit request_length with the range of 0 to 4095 bytes. For lengths of 4096 bytes, the ac=0 format must be used which adds 4 bytes of overhead (about 0.1% inefficiency).
- The ac=1 format does not support an Extension Header Length field and reuses the value from the prior ac=0 format in that packet. Therefore, the ac=0 format must be used when a different Extension Header Length is required.

8.4.3. Request / Response

Each AXI transaction leads to a SUE-T request from initiator to target, and a SUE-T response from target to initiator. Note that requests may be packed, responses may be packed and responses may be indicated using ACKs/NACKs, but this has no effect on functionality or order, only on timing.

For AXI read transactions, the SUE-T response carries the read response data and/or read response code that is needed to complete the AXI read.

For AXI write transactions, the SUE-T response carries the write response code that is needed to complete the AXI write. An implementation may make optimizations on the AXI write completion depending on the desired properties for transaction ordering (across connections), transaction visibility, and error handling. Some possible options for write completions, from most conservative to most aggressive, are:

1. The full network round-trip latency to XPU memory and back is covered, and the AXI write completion guarantees that the write has reached XPU memory successfully and/or reports any error encountered on the full path to XPU memory and back.
2. The XPU memory subsystem completes the write once it has been committed into a queue destined to the XPU memory. This preserves order and there may not be any actionable error cases beyond this point, so this is a common optimization.
3. The target SUE-T implementation completes the write as soon as it has been committed into a queue destined to the AXI interface to the XPU. The write has not reached memory from the perspective of other observers (e.g. using other NICs), and there may be XPU error cases (e.g. translation errors) that are not reported in the response.
4. The initiator SUE-T implementation completes the write as soon as it has been committed into a queue destined to the initiator's network port. Similarly, the write has not reached memory (in fact, may not even have left the source NIC) and there may be network error cases (e.g., corrupted or lost packet) that are not reported in the response.

The details of such optimizations are implementation-specific. They could be configured as astatic mode, or in some cases can be signaled using the SES "delivery complete" (DC) bit where appropriate. SUE-T does not define a mechanism to select this behavior.

However, in all cases a SUE-T write response or ACK/NAK is required in order to clean up resources at the initiator related to the outstanding write request.

9. Summary

The SUE-T Base Specification provides an open, flexible framework suitable for the design of low-latency, high-bandwidth endpoints attached to scale-up networks based on IEEE Ethernet. This framework includes solutions for memory transactions, packet assembly/disassembly, load-balancing, and re-transmission. SUE-T based designs can readily be optimized to meet specific performance, cost, and power constraints, and are suitable for scale-up networks relying on Ethernet datalinks defined by OCP ESUN, UFH defined by the UEC, or UDP/IP defined by the IETF.

It is expected that this Specification will provide a clear blueprint for those developing next-generation xPUs, and that design programs will be streamlined by the elimination of duplicative work. As the adoption of Ethernet-based scale-up networking accelerates, SUE-T will become a cornerstone of the AI/ML architecture.

10. References

1. *OCP ESUN - Network Operator Requirements - Base Specification*, **Open Compute Project**, 2026, <https://www.opencompute.org/documents/ocp-esun-network-operator-requirements-base-specification-rev-1-0-final-pdf> .
2. *Ultra Ethernet™ Specification v1.1*, **Ultra Ethernet Consortium**, in preparation.
3. *IEEE Standard for Ethernet*, **IEEE Std 802.3-2022**, 2022, <https://standards.ieee.org/ieee/802.3/10422/> .
4. *Internet protocol, DARPA Internet program, protocol specification*, **IETF RFC 791**, 1981, <https://datatracker.ietf.org/doc/html/rfc791> .
5. *User Datagram Protocol*, **IETF RFC 768**, 1980, <https://datatracker.ietf.org/doc/html/rfc768> .
6. *Internet Protocol, Version 6 (IPv6) Specification*, **IETF RFC 8200**, 2017, <https://datatracker.ietf.org/doc/html/rfc8200> .
7. *UDP Usage Guidelines*, **IETF RFC 8085**, 2017, <https://datatracker.ietf.org/doc/html/rfc8085> .
8. *AMBA® AXI and ACE Protocol Specification*, **ARM Limited**, 2021, <https://support.arm.com/documentation/ih0022/hc/?lang=en>

11. Glossary

CMD_PDU	Command Payload Data Unit
DW	Data Word, 32bits
dstXPU	destination XPU
ESUN	Ethernet Scale-Up Network
IOMMU	Input Output Memory Management Unit
LLR	Link Layer Retransmission
PCIe ATS	PCIe address translation services
PDC	Packet Delivery Context
PDCID	PDC Identifier
PFC	Priority Flow Control
PSN	Packet Sequence Number
ROD	Reliable Ordered Delivery
RODL	Reliable Ordered Delivery for Local networks
SES	Semantic Sublayer
SUE-T	Scale-Up Ethernet Transport
TC	Traffic Class
UET	Ultra Ethernet – Transport
UFH	Unified Forwarding Header
VC	Virtual Connection
VDH	Vendor Defined Header
WSTRB	Write Strobe
XPU	Generic term for special-purpose AI accelerator