This presentation covers Gen-Z Collectives support and operations.
Disclaimer

This document is provided ‘as is’ with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of proprietary rights, relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Gen-Z is a trademark or registered trademark of the Gen-Z Consortium.
All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

All material is subject to change at any time at the discretion of the Gen-Z Consortium

http://genzconsortium.org/
Collectives are used in a wide-range of messaging applications, e.g., numerous MPI and shared memory applications use collectives. Collectives are often implemented in application or middleware software. They are used to coordinate computations or activities across a distributed application. Gen-Z specifies a set of operations to enable applications to construct a variety of collectives. Further, these operations can be used by collective accelerators within a switch topology. For example, in the above figure, one collective accelerator is provisioned per switch.
Gen-Z supports a variety of collectives. These can be implemented using collective request and response packets or in conjunction with Gen-Z read, write, and LDM (large data movement) read request packets.

An All-to-All collective can be implemented as a Scatter plus All-Gather collective. This sequence should provide equivalent performance.
This slide illustrates example steps to execute a broadcast collective using collective accelerators. In this example, the collective accelerators optimize communications to reduce fabric load, the potential of congestion events, and to improve application performance (e.g., eliminate software overheads, interrupts, etc. used in software-only implementations). A collective is initiated by an application or on behalf of an application using middleware software. If software has explicit knowledge of a specific collective accelerator, then it can directly communicate with it using a collective request packet. If it does not have explicit knowledge of a specific collective accelerator, then it can use multicast encapsulation (even if the switches do not support multicast packet replication) to transparently target the initial collective accelerator which takes over executing the collective across all collective accelerators. Multicast encapsulation enables a unicast packet to be encapsulated and distributed to a multicast group.
All collective packets contain a set of fields used to identify the collective group and the specific collective operation within that group. This enables a Gen-Z fabric to support multiple groups and outstanding collective operations.

Collective operations use the CTXID OpClass. This OpClass includes Requester and Responder context identifiers (application handles) to quickly and easily identify the application resources (simplifies hardware implementations)

Collectives can use a variety of data type sizes, signed / unsigned integer and floating point.

Though this example contains a payload field, not all collective packet formats do.

If a completion handler needs to be invoked, then the component sets CH = 1b. This enables the Requester to dynamically signal completion handling should be invoked to inform the application / middleware of the request packet’s arrival (solutions using collective accelerators may not require such invocations depending upon the implementation).

```
| Collective Group ID—identifies the set of components participating in a collective |
| Collective ID—identifies a specific collective operation |
| ReqCTXID / RspCTXID—Requester and Responder context identifiers used to locate resources |
| Data Type—size and type of data to target, e.g., 8 / 16 / 32 / 64 / 128 / 256-bit data, integer / FP, etc. |
| Payload (Data Type) |
| CH—Indicates if a completion handler is to be invoked upon receiving the packet |
```
This packet format is used by reduce collectives. It contains additional fields include a Reduce Operator (what reduction to be performed), and a Filter to indicate what comparison to perform, in the case of a Map-Reduce operation, the packet includes the Filter Reference Data.
A collective Abort impacts a single collective group and a single collective operation. A collective All-Abort impacts all outstanding collective operations associated with a specific collective group.
Unless an error was detected, a Collective Responder Count Response packet is returned for each collective request packet. The Collective Responder Count Response can represent one or more Responders, enabling hierarchical collective solutions and / or collective accelerators to aggregate the results, i.e., response packets, from multiple Responders.
Thank you

This concludes this presentation on collectives. Thank you.