The Chapel communication layer is a runtime component for exchanging data between locales typically on different physical nodes of a system. The interfaces it exports are used by the compiler and runtime to bootstrap execution and perform data transfer. Currently, the most commonly used Chapel communication layer is GASNet, a portable networking interface designed specifically for PGAS languages [1]. GASNet is easy to use and invaluable for portability via its conduit implementations, and it can deliver good performance. That said, for low-latency networks like the Aries interconnect in Cray XC systems\textsuperscript{TM}, the GASNet communication layer often performs significantly worse than Cray’s native communication layer written directly to uGNI (User-level Generic Network Interface) [2], the low-level Aries API. There are a number of reasons for this, some of which may just be a matter of implementation choices, but one notable problem is that GASNet’s API does not include a way to take advantage of hardware specific features such as Aries atomic memory operations.

So what’s a language runtime implementer to do? Forego performance for portability on high end systems or hope that the vendor will invest in a native communication layer? Perhaps there’s another way.

The OpenFabrics Interfaces (OFI) project is a community effort aimed at creating high performance, hardware-agnostic networking APIs. The first product of the OFI working group is the user-level networking API called libfabric. Libfabric was designed by a broad group of experts from across industry, academia, and government with the goal of providing a standardized interface for HPC middleware clients such as PGAS language runtime libraries [3]. The result is an open source implementation (hosted on GitHub) targeting upwards of seven different networking fabrics via pluggable implementations called providers. Because libfabric was designed from the start with performance in mind, the interfaces are quite rich and also include an extension mechanism for exposing for provider-specific functionality. The libfabric implementation is relatively new, but it holds great promise for a happy middle ground between that of an easy-to-use, purpose-specific library like GASNet and a extremely low-level, vendor-specific library like uGNI.

In this talk, I will describe the design and implementation of an OFI libfabric
communication layer for the Chapel runtime. I will start with an overview of OFI libfabric, followed by an overview of the Chapel communication layer functions and how they are mapped to the libfabric API. I will conclude with some of the general caveats and lessons learned from using libfabric as well as implementing a Chapel communication layer.

References

