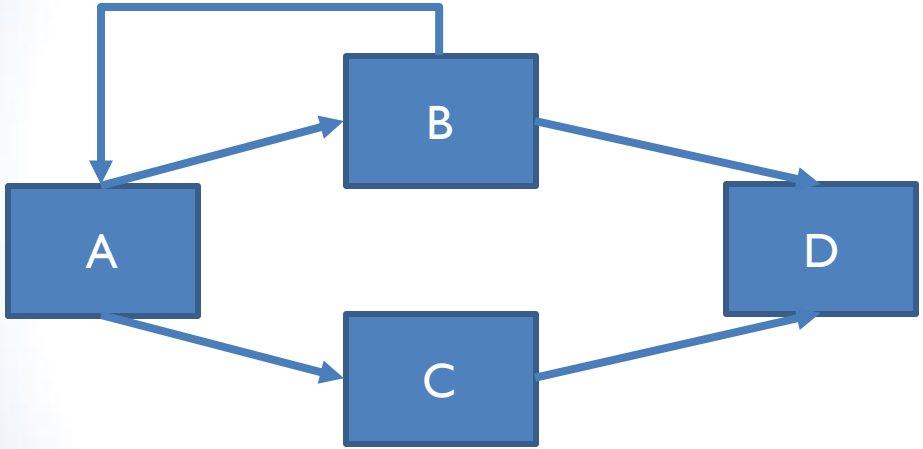
A Heterogeneous Stack for Deploying Clusters

Naif Tarafdar and Paul Chow

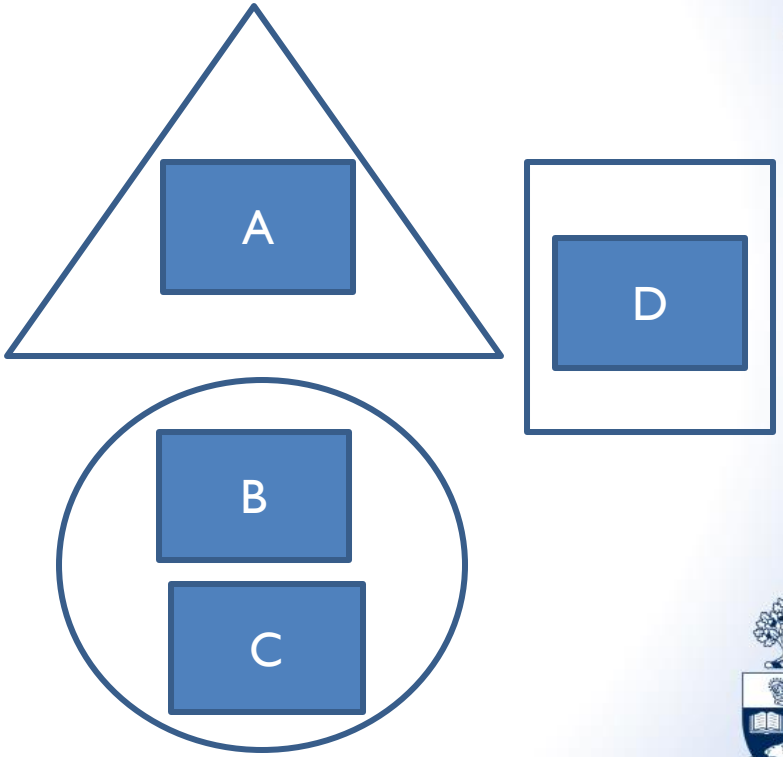
High-Performance Reconfigurable Computing Group
Department of Electrical and Computer Engineering
University of Toronto

The Holy Grail

Distributed Flow Diagram



Heterogeneous Data Center



Data Centers are Heterogeneous

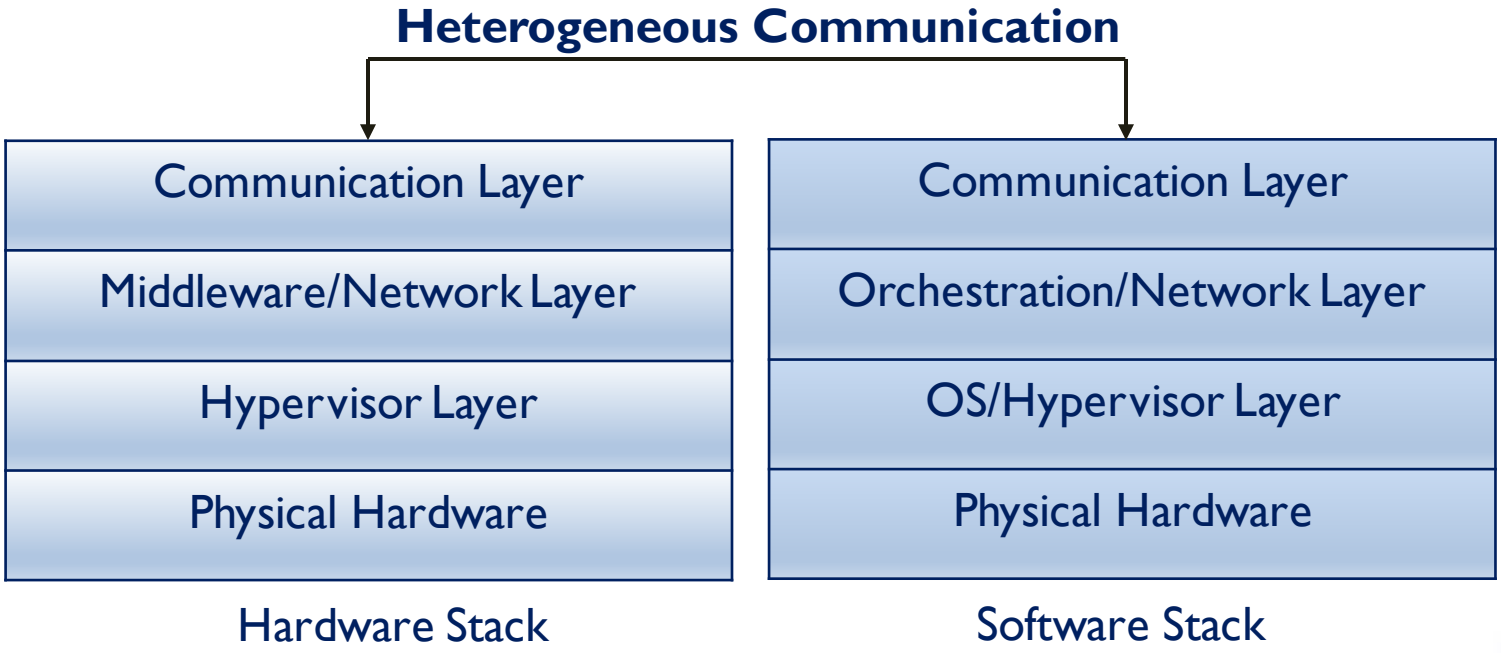
- Take advantage of unique resources in data center
- Need back doors or abstractions to communicate between different devices
- Scaling this up is difficult as different devices have different programming models, abstractions etc.
- Communicating across these different devices even more complicated

Contributions

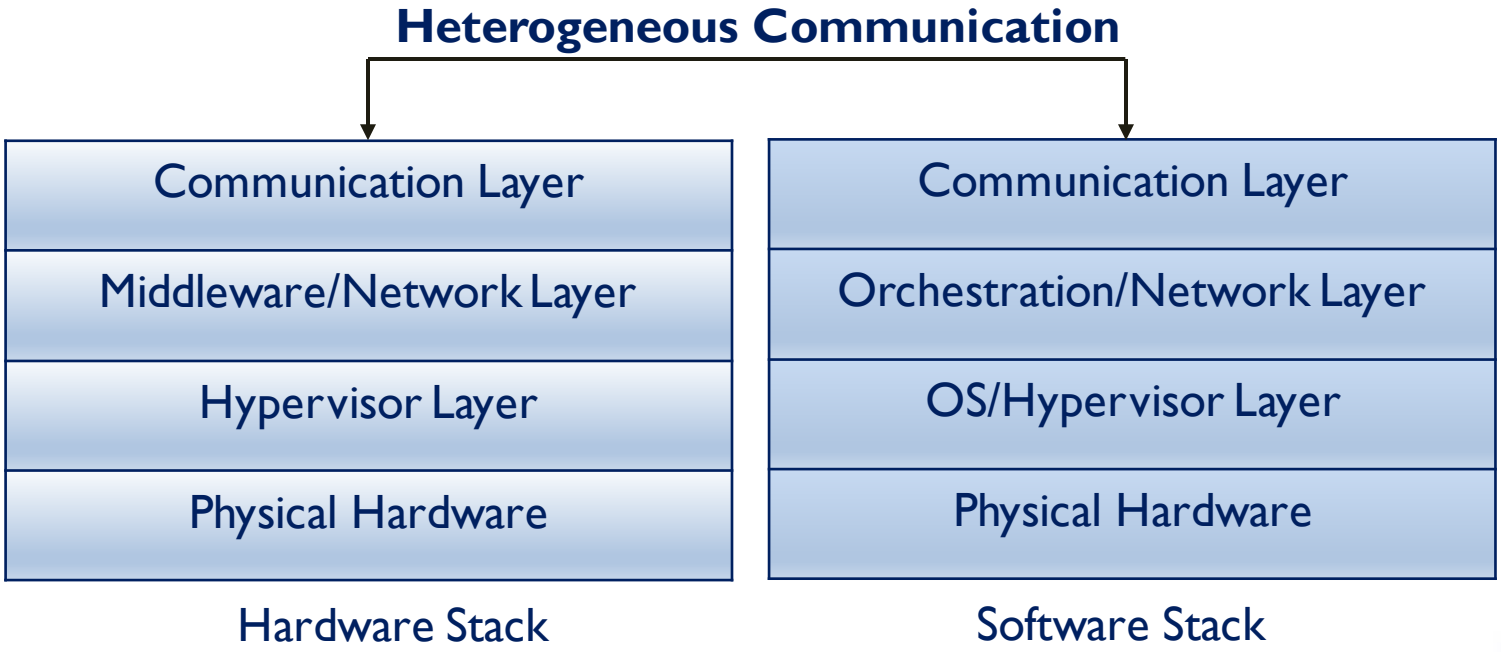
- Built a heterogeneous communication layer across multiple devices
- Built on top of heterogeneous stack
  - Allows to communicate to any device that is abstracted in stack
- Functional portability allows the prototyping of cluster in software before deploying on accelerator

BACKGROUND: GALAPAGOS STACK

Heterogeneous Abstraction Stack

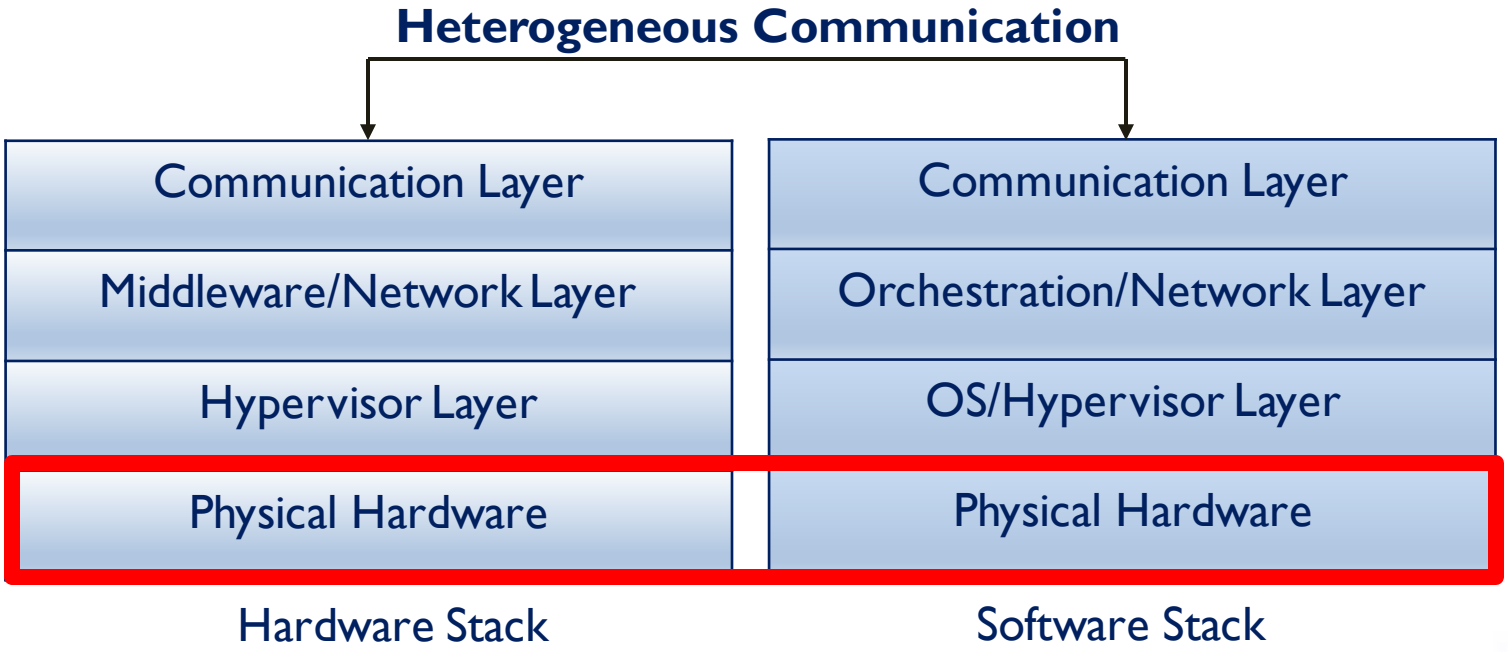


Heterogeneous Abstraction Stack



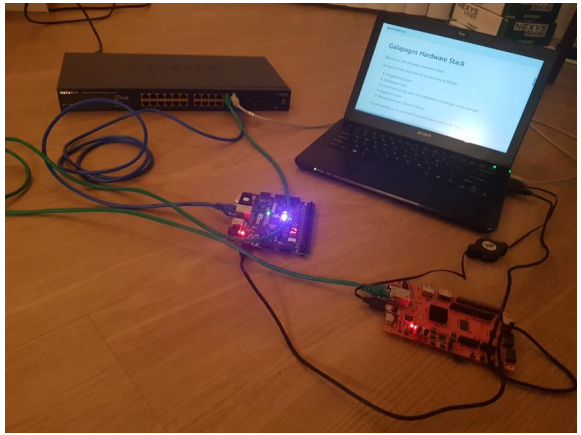
Galapagos

Heterogeneous Abstraction Stack

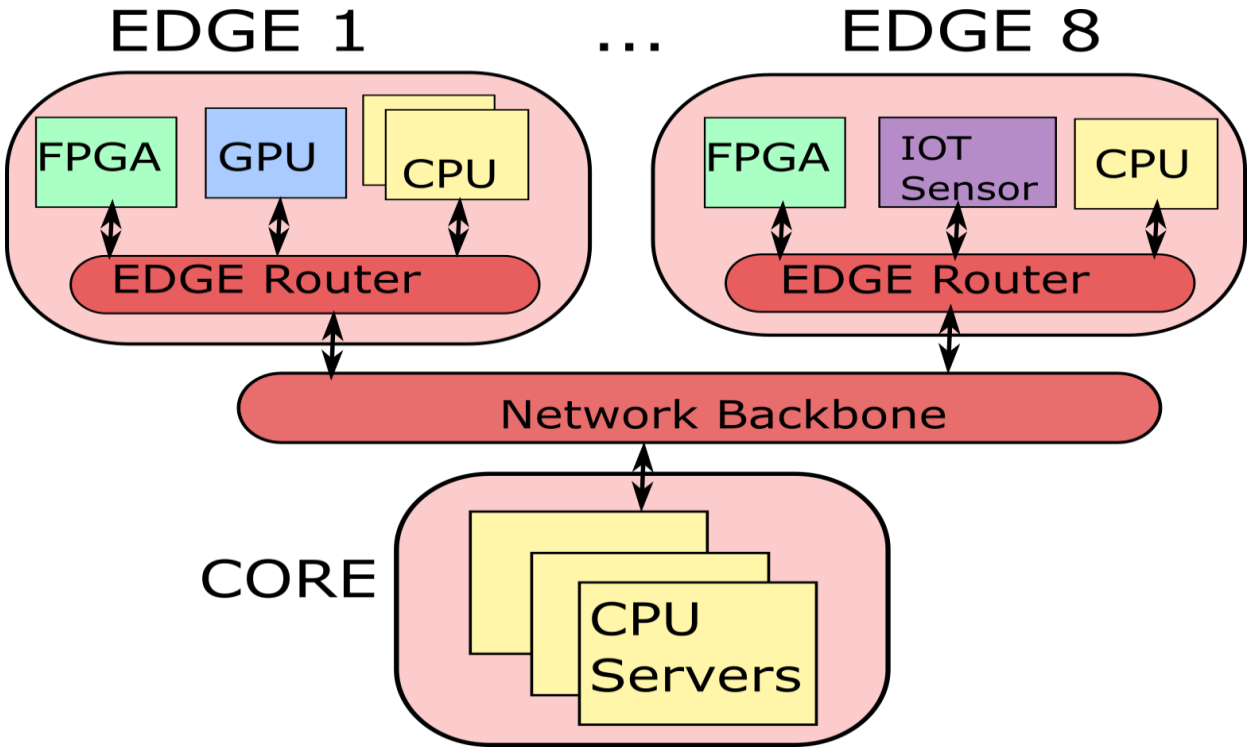


Physical Hardware

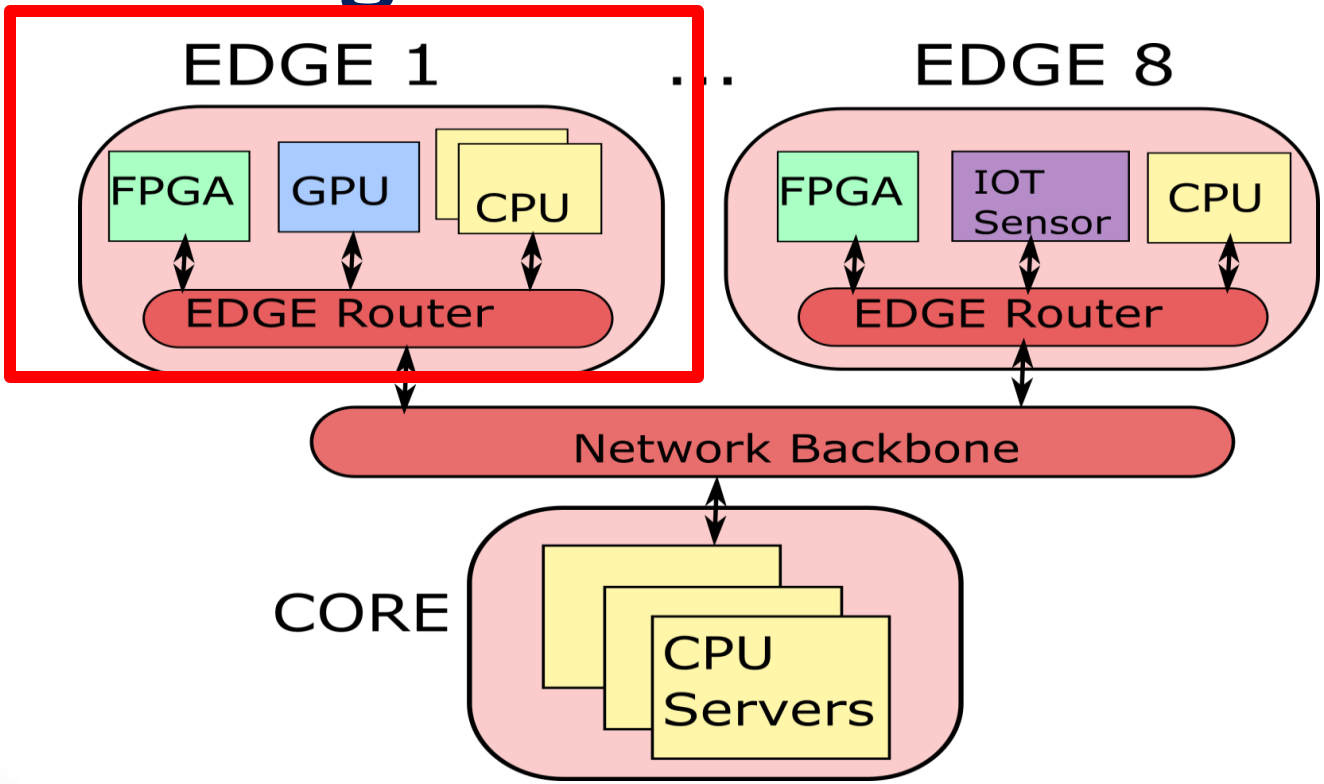
- Refers to the physical devices and how they are connected
  - E.g: FPGA, CPU, Sensors
  - Wireless , Infiniband
- Example: Nexys 4 Board, Pynq Board, Laptop attached via Ethernet cables to 1G network



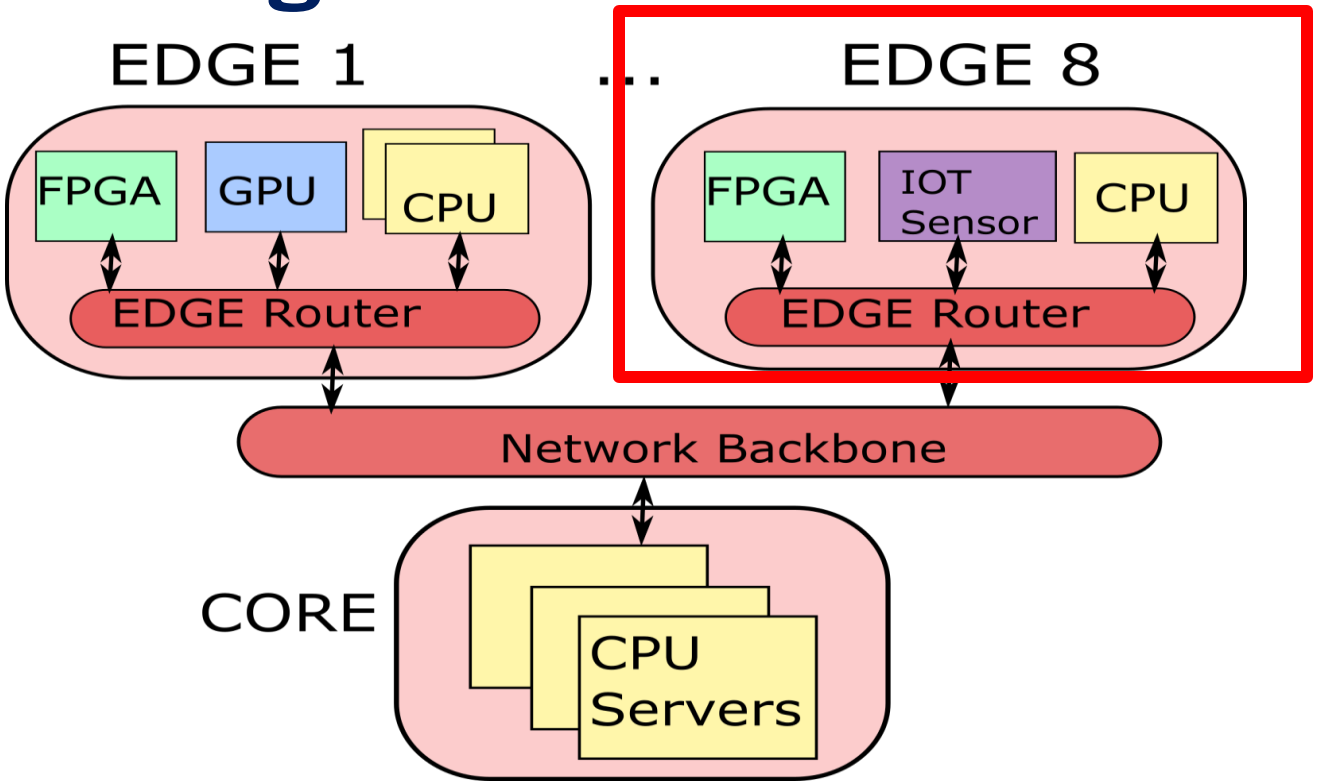
Heterogeneous Multi-Tier Cloud



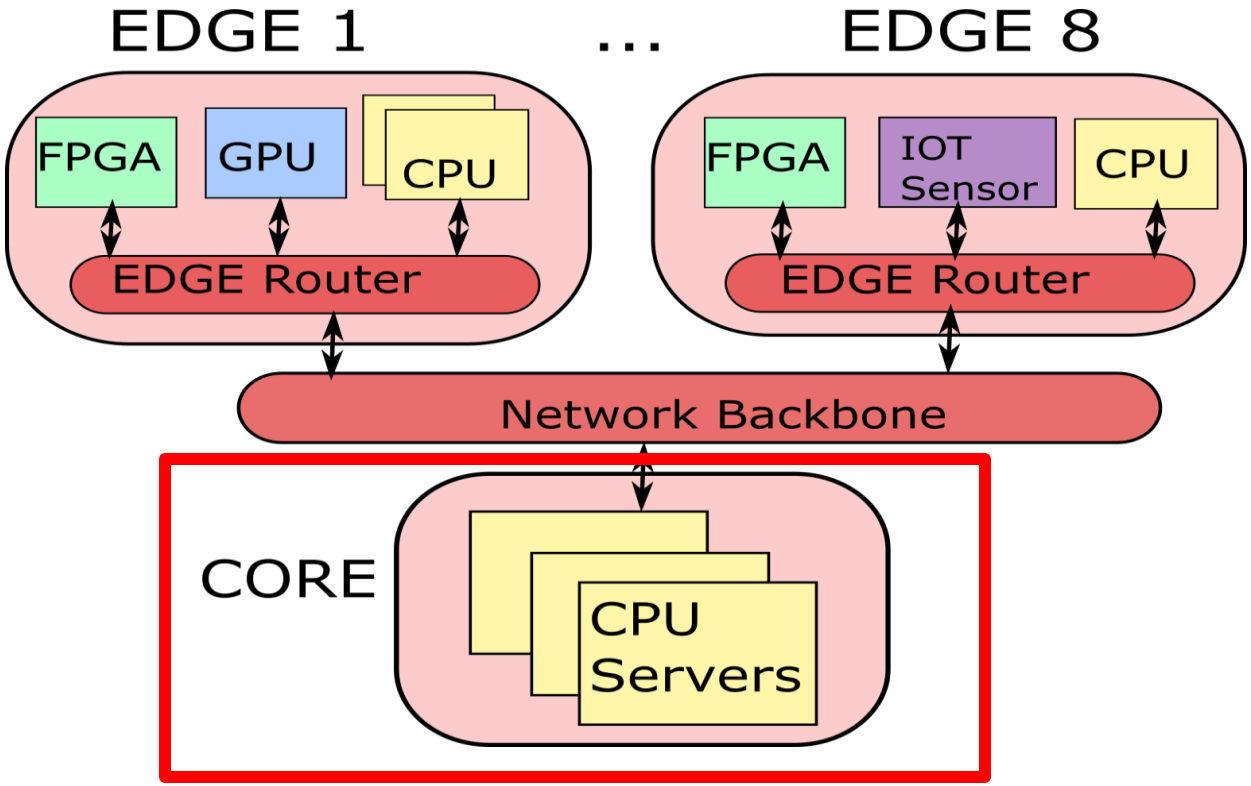
Heterogeneous Multi-Tier Cloud



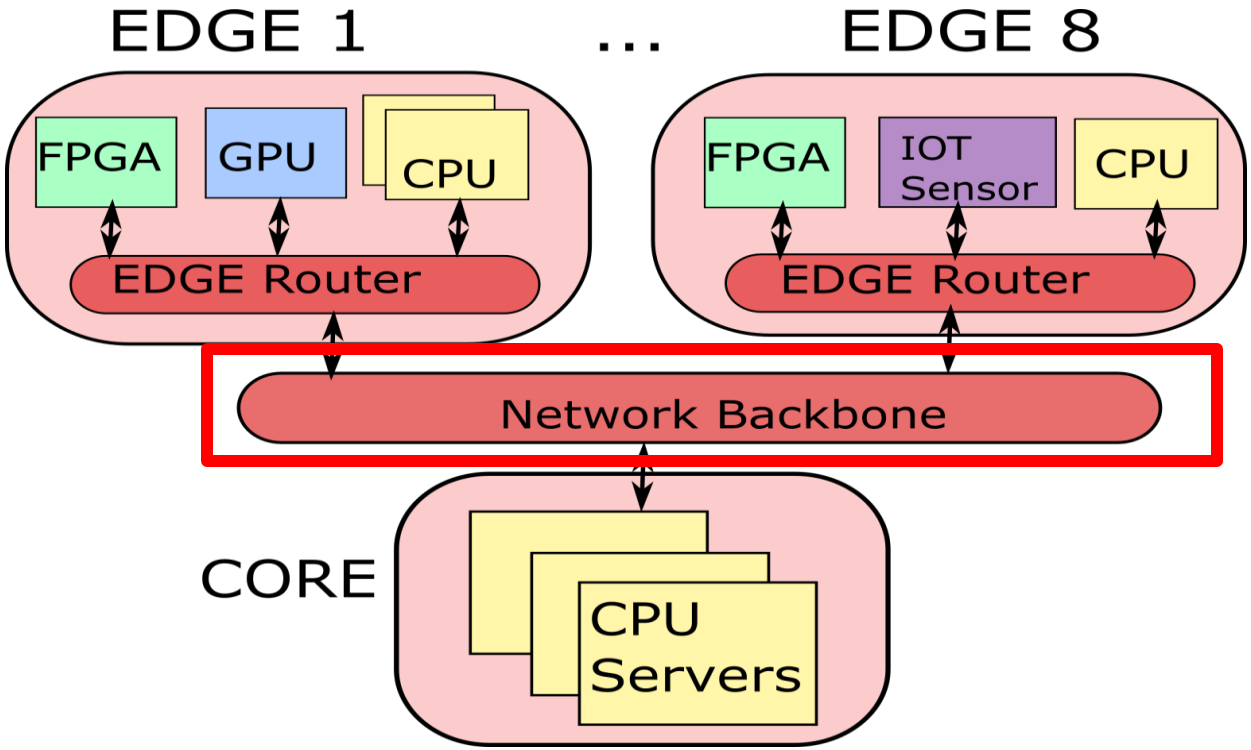
Heterogeneous Multi-Tier Cloud



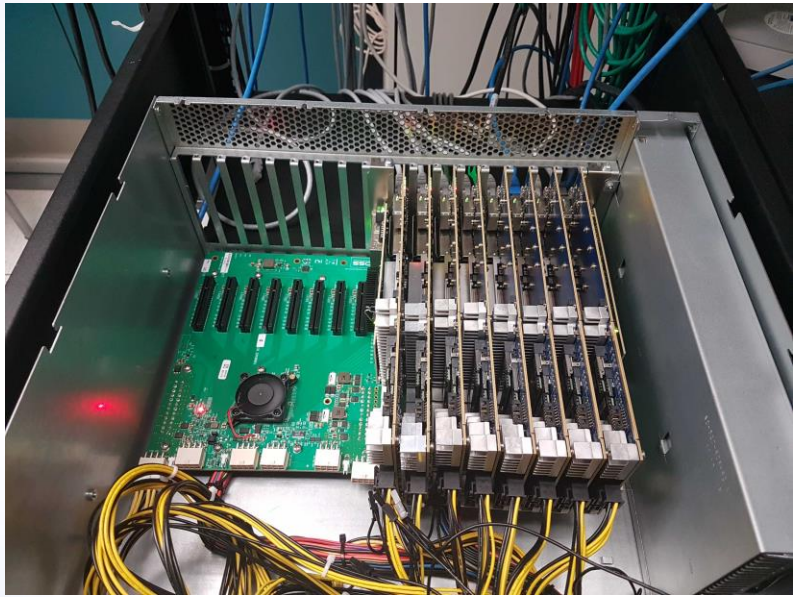
Heterogeneous Multi-Tier Cloud



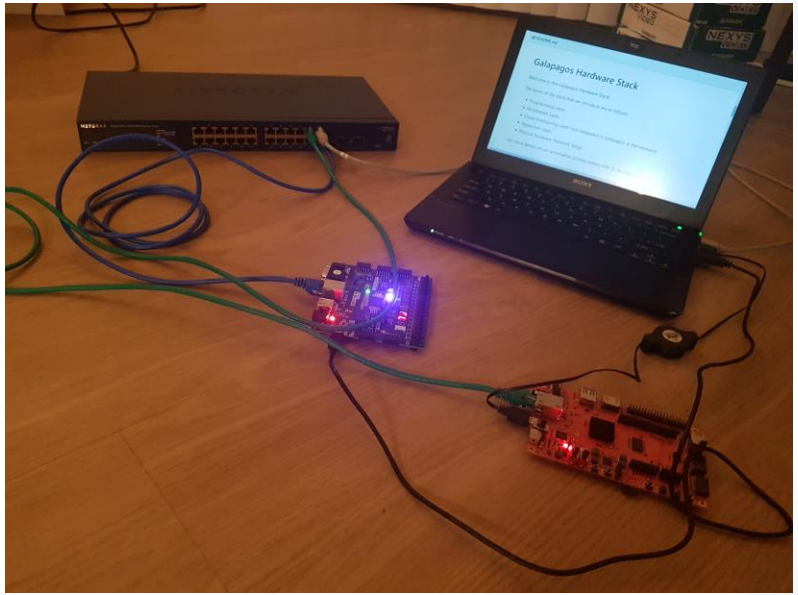
Heterogeneous Multi-Tier Cloud



Our Datacenters

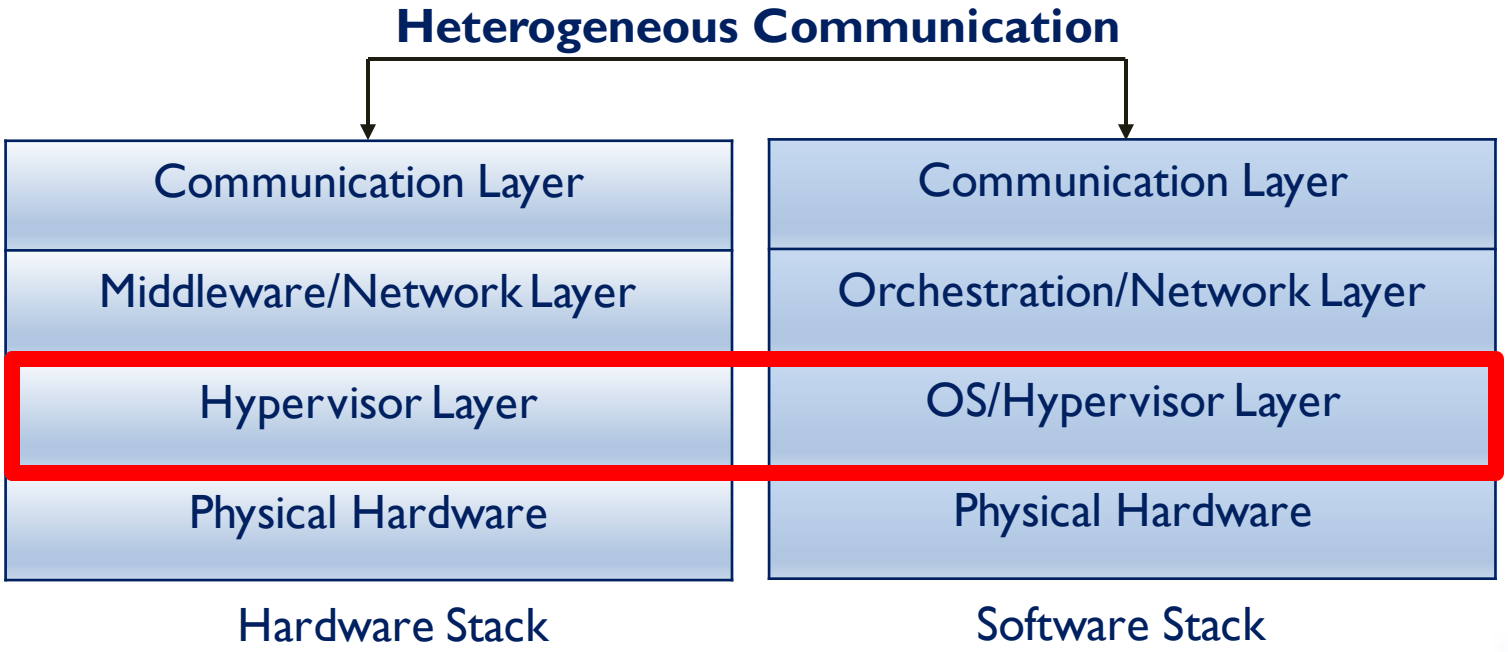


MPSoC Boards Connected via 100 GB/s Cables
~$100 000



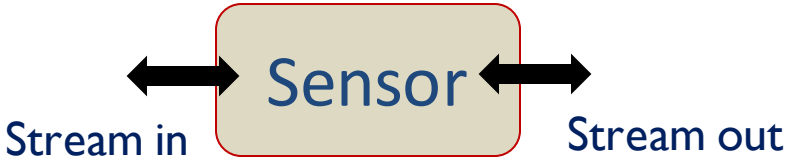
Nexys, Pynq, Laptop, 1G switch
~$2000

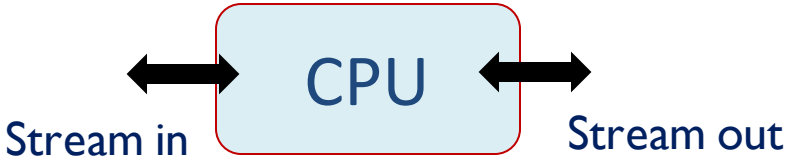
Heterogeneous Abstraction Stack



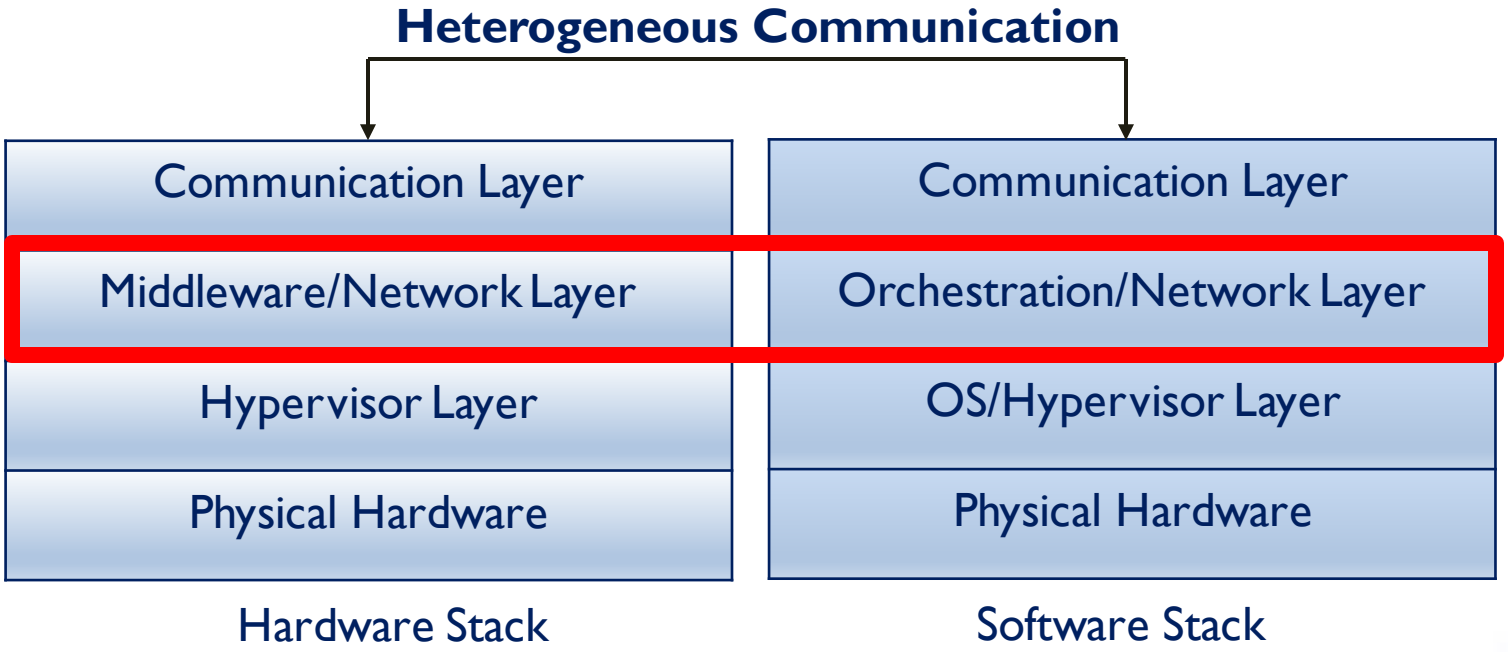
Galapagos: Hypervisor

- Abstracts device into a streaming device
- All devices with hypervisor now looks like the same streaming device





Heterogeneous Abstraction Stack

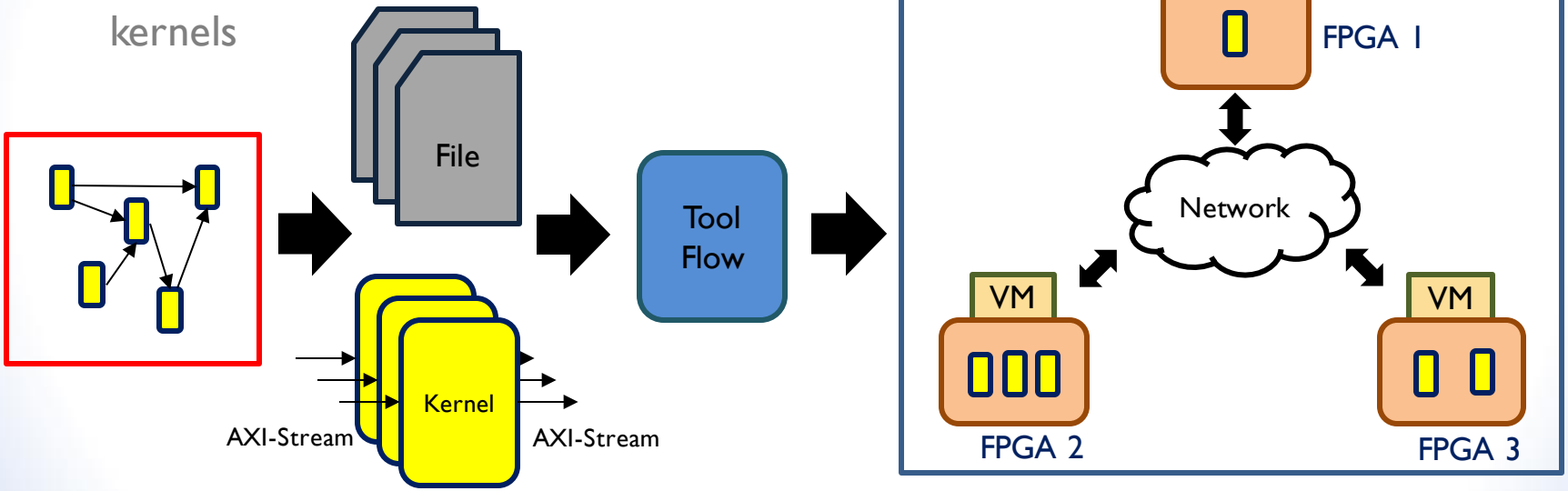


Galapagos: Middleware

- Allows streaming kernels independent of placement and implementation to communicate to any other kernel within a heterogeneous cluster
- This layer refers to how we orchestrate clusters of resources
  - Includes FPGAs and CPUs
- Orchestration includes automating the connections between resources and providing handle to entire cluster

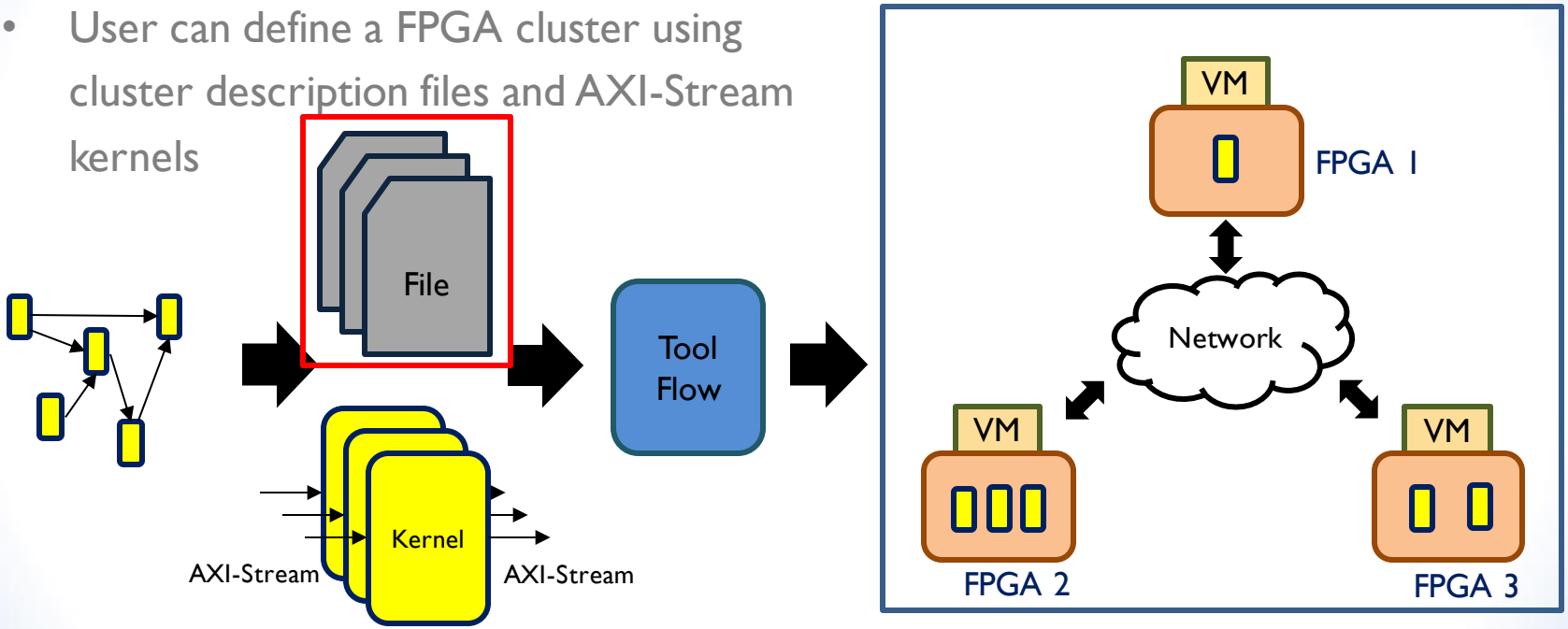
Galapagos: Middleware Layer

- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



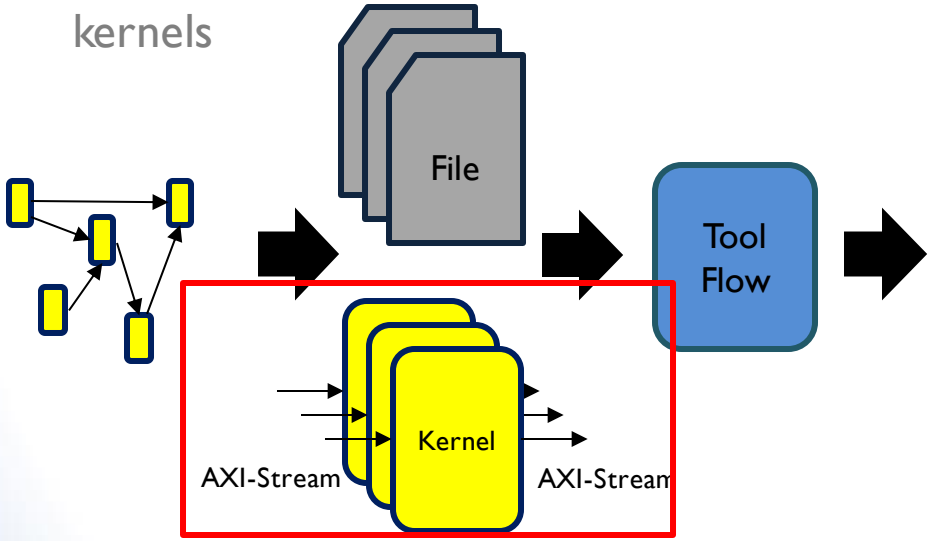
Galapagos: Middleware Layer

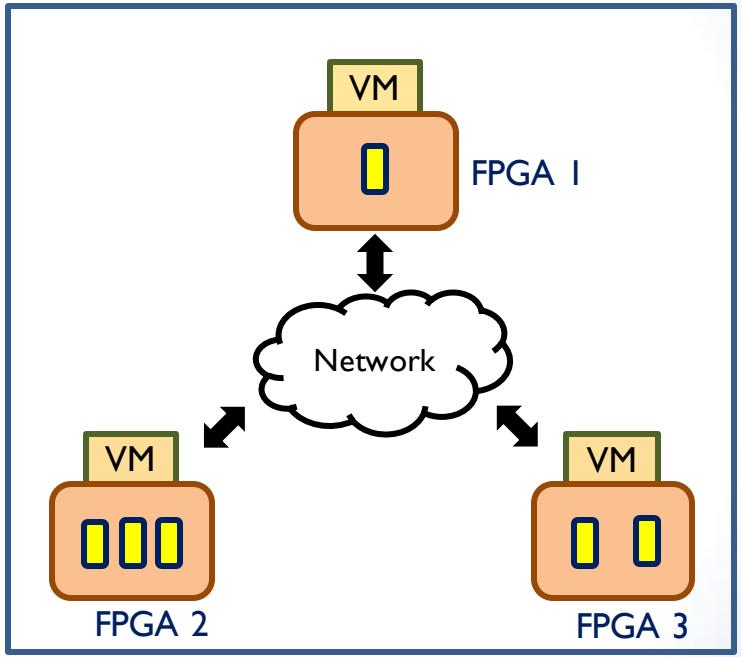
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels



Galapagos: Middleware Layer

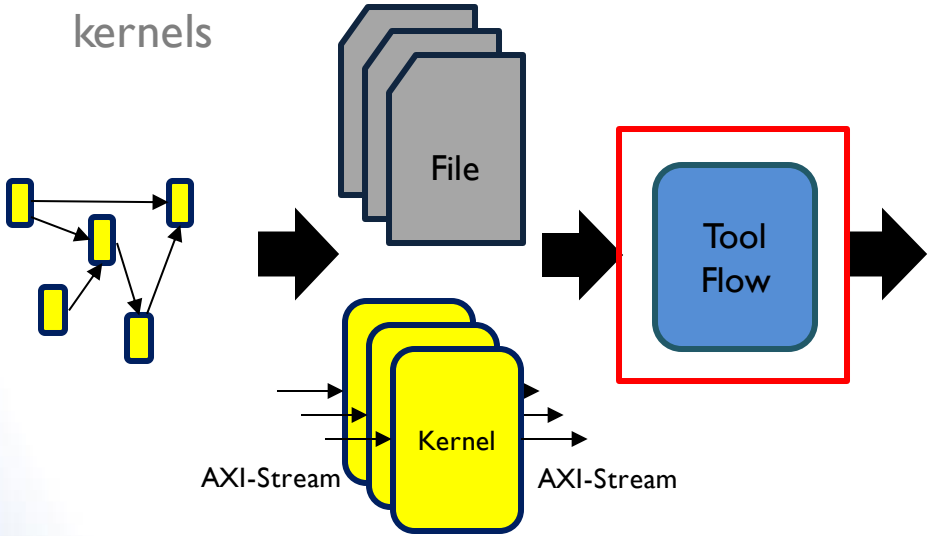
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels

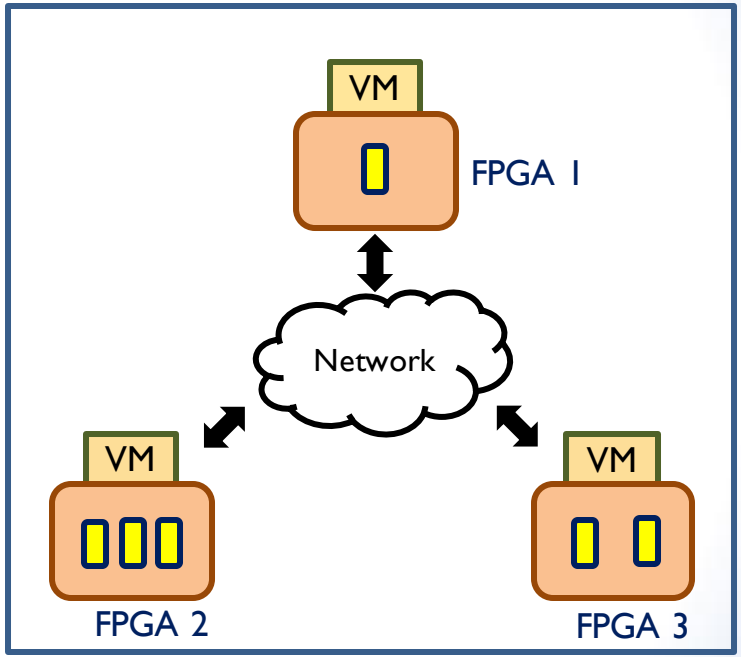




Galapagos: Middleware Layer

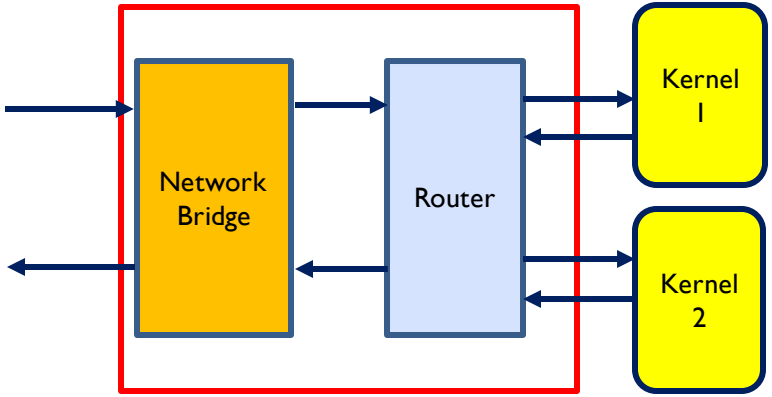
- User can define a FPGA cluster using cluster description files and AXI-Stream kernels





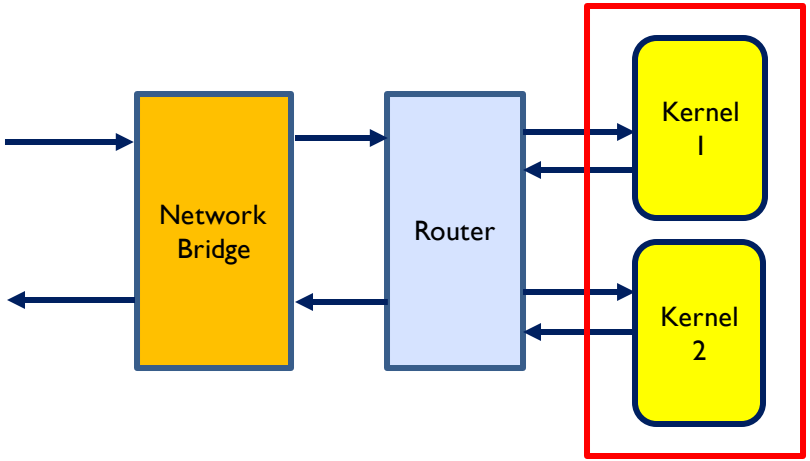
Galapagos: Middleware IP Blocks

Middleware generates additional IP blocks



Galapagos: Middleware IP Blocks

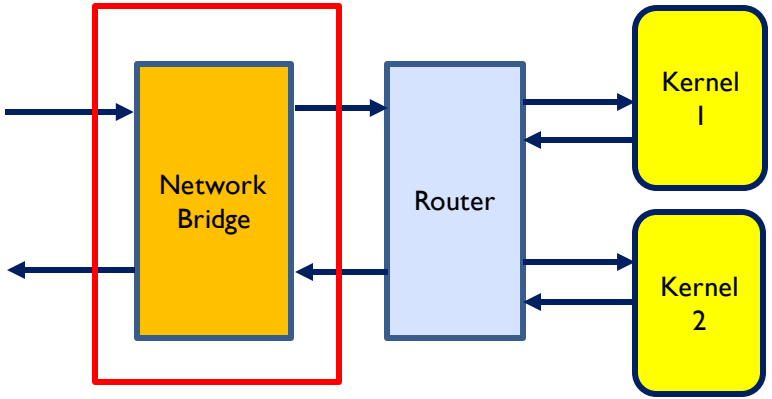
Connects to user provided kernels within application region of hypervisor



Galapagos: Middleware IP Blocks

Network Bridge

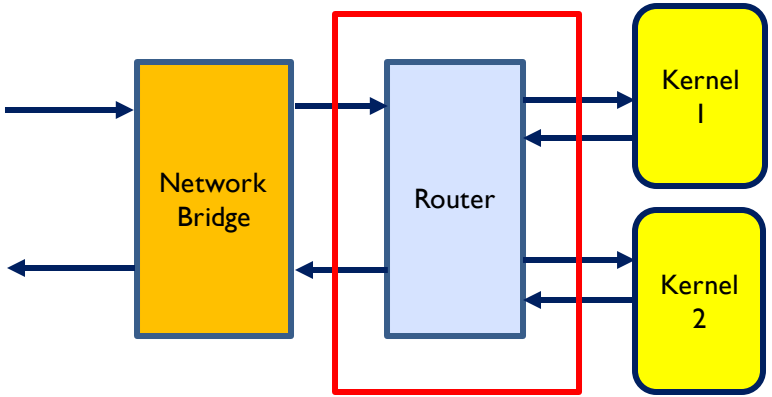
- Translates network packets into AXI stream packets
- Current support for TCP and L2 ethernet



Galapagos: Middleware IP Blocks

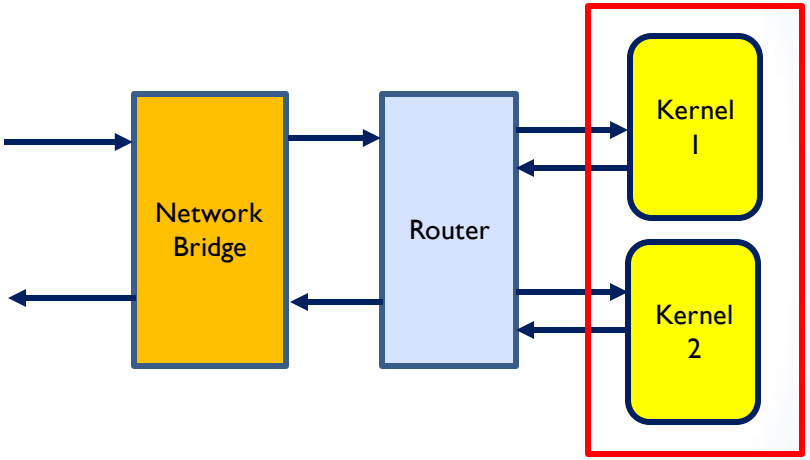
Router

- Routes packets on chip and off-chip
- Routing table maps AXI dest to FPGA network address

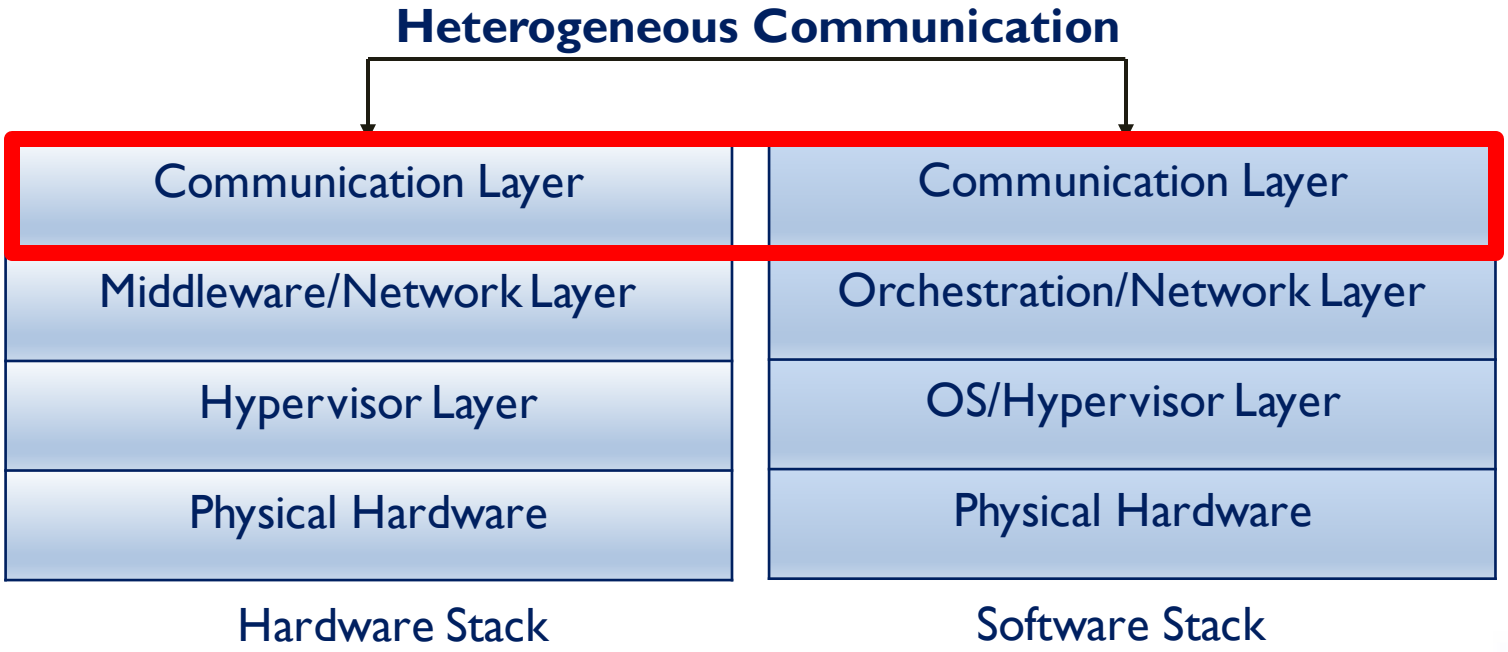


Galapagos: Middleware IP Blocks

Connects to user provided kernels within application region of hypervisor

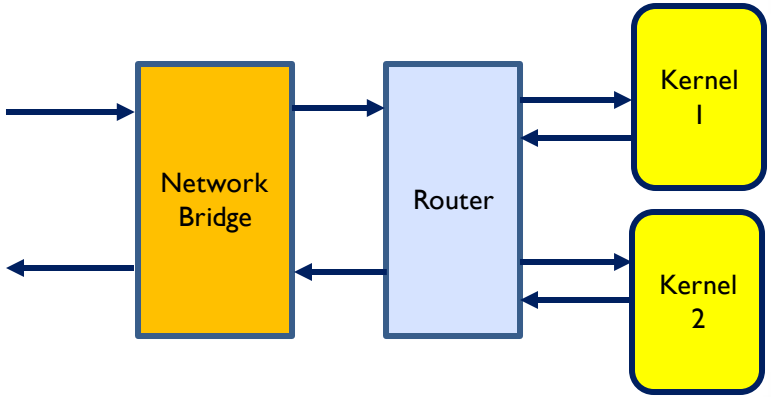


Heterogeneous Abstraction Stack



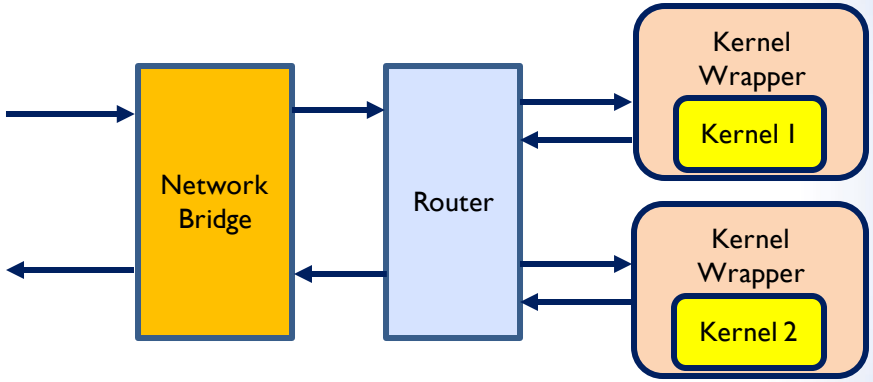
Communication Layer: libGalapagos

- Create software model of each component
- Galapagos software kernel object wrapper for HLS module
  - Functionally portable, uses same HLS code for software
- Kernels are spawned as individual threads



Communication Layer: libGalapagos

- Create software model of each component
- Galapagos software kernel object wrapper for HLS module
  - Functionally portable, uses same HLS code for software
- Kernels are spawned as individual threads

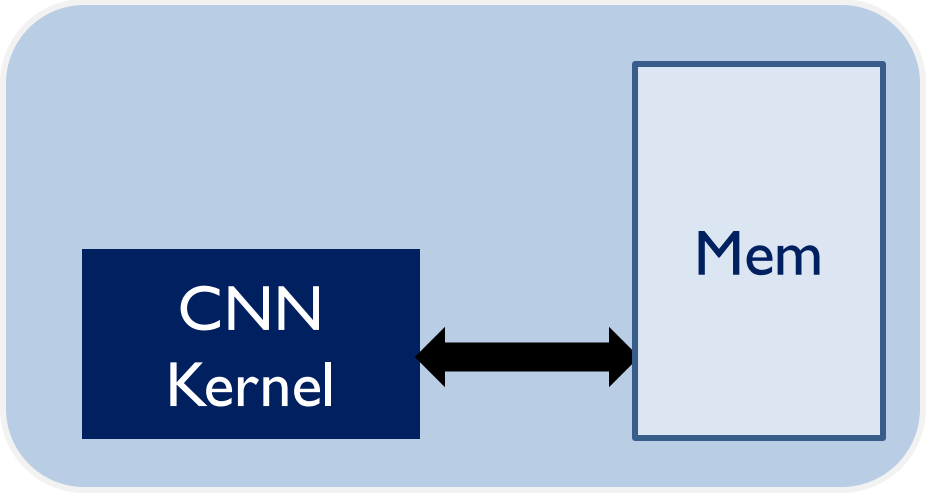


HETEROGENEOUS MACHINE LEARNING PLATFORM

September 10, 2019

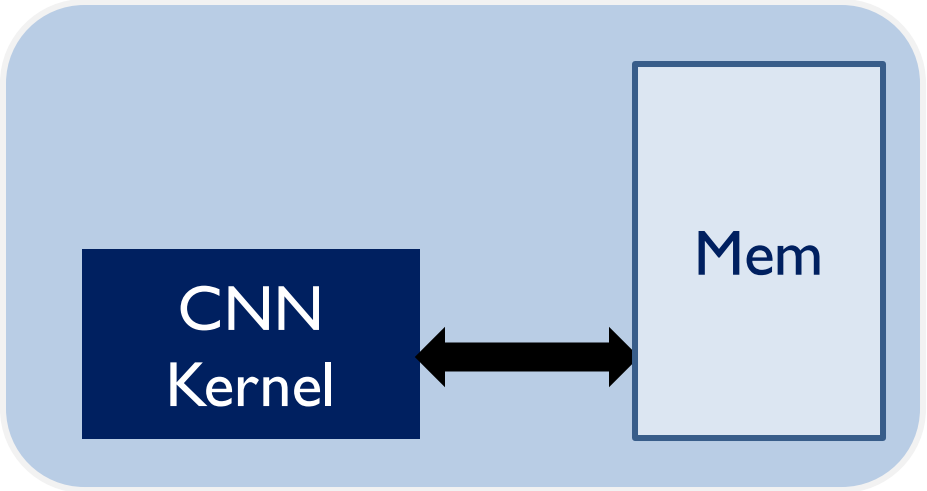
Single Node Inference Engine

- Xilinx CNN Kernel
- Kernel Instruction provides layer information and addresses for input, weights and memory
- Not streaming based, can't connect to cluster



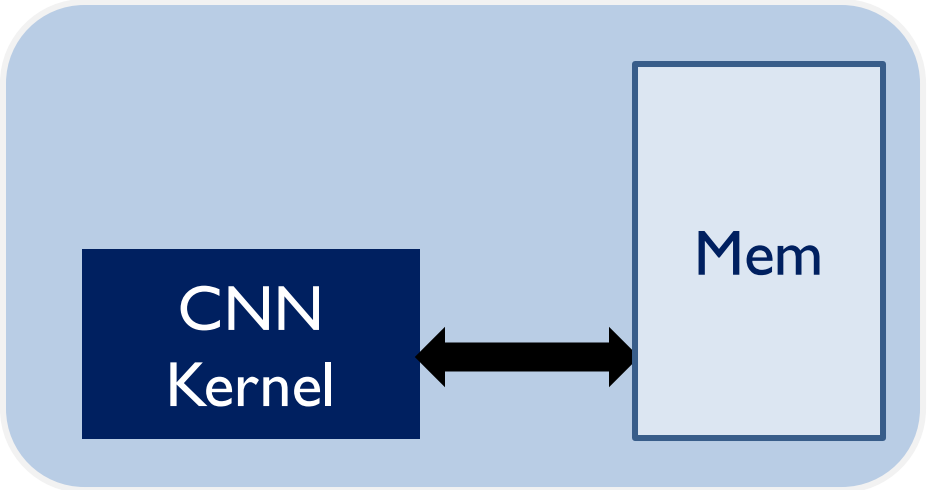
Single Node Inference Engine

- Xilinx CNN Kernel
- Kernel Instruction provides layer information and addresses for input, weights and memory
- Not streaming based, can't connect to cluster



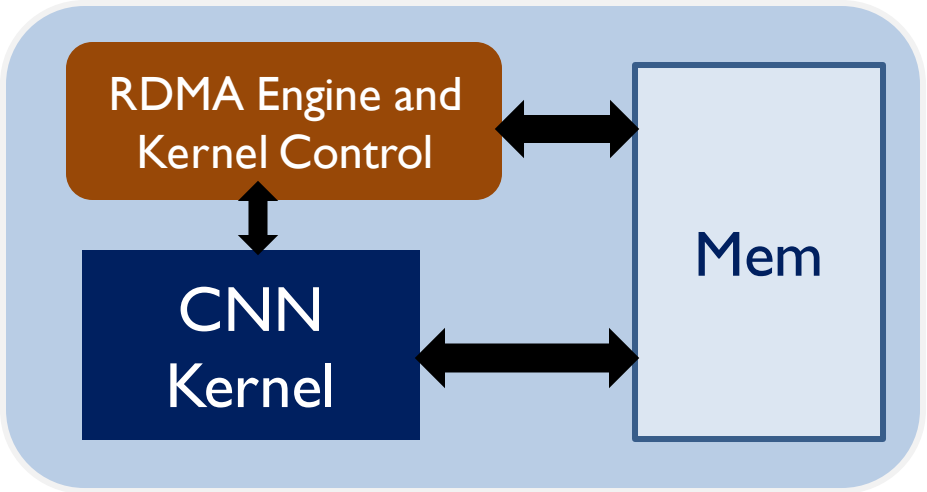
Single Node Inference Engine

- Xilinx CNN Kernel
- Kernel Instruction provides layer information and addresses for input, weights and memory
- Not streaming based, can't connect to cluster



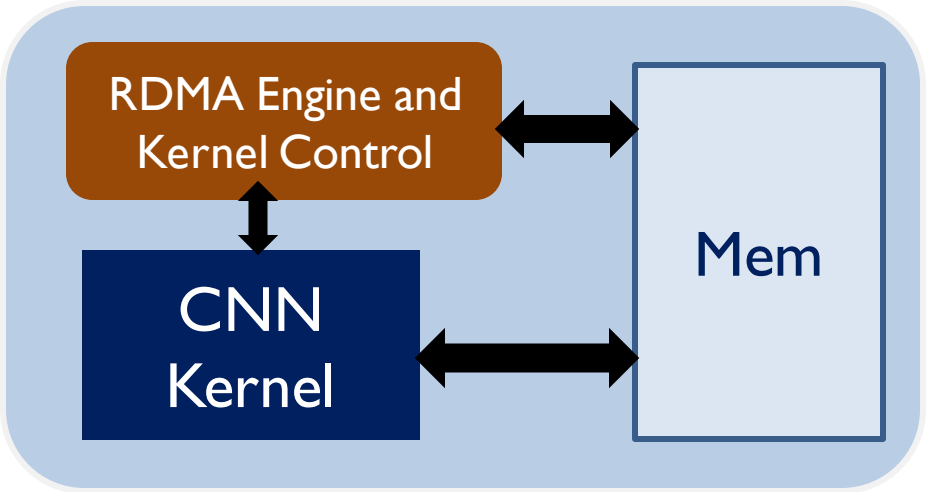
Single Node Inference Engine

- Controller written in HLS
- Stream in/stream out
- Streams in insns for CNN kernel and DMA instructions for memory



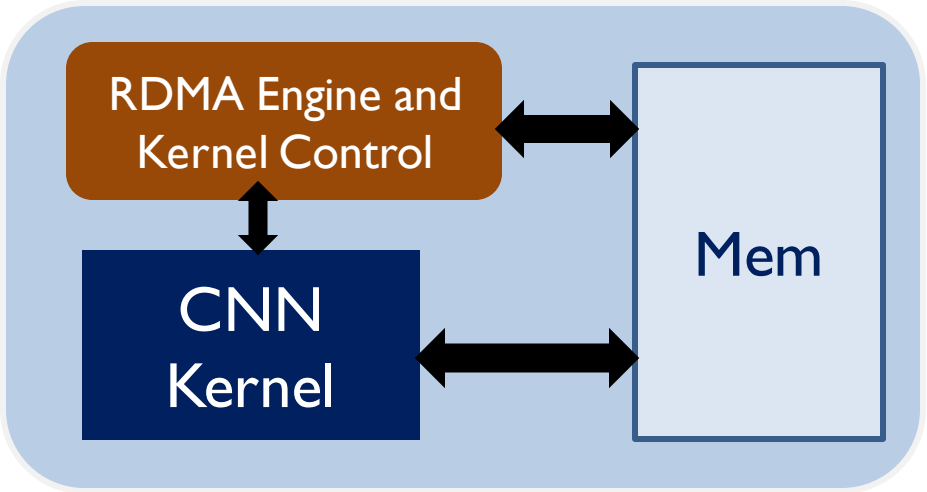
Single Node Inference Engine

- Controller written in HLS
- Stream in/stream out
- Streams in insns for CNN kernel and DMA instructions for memory



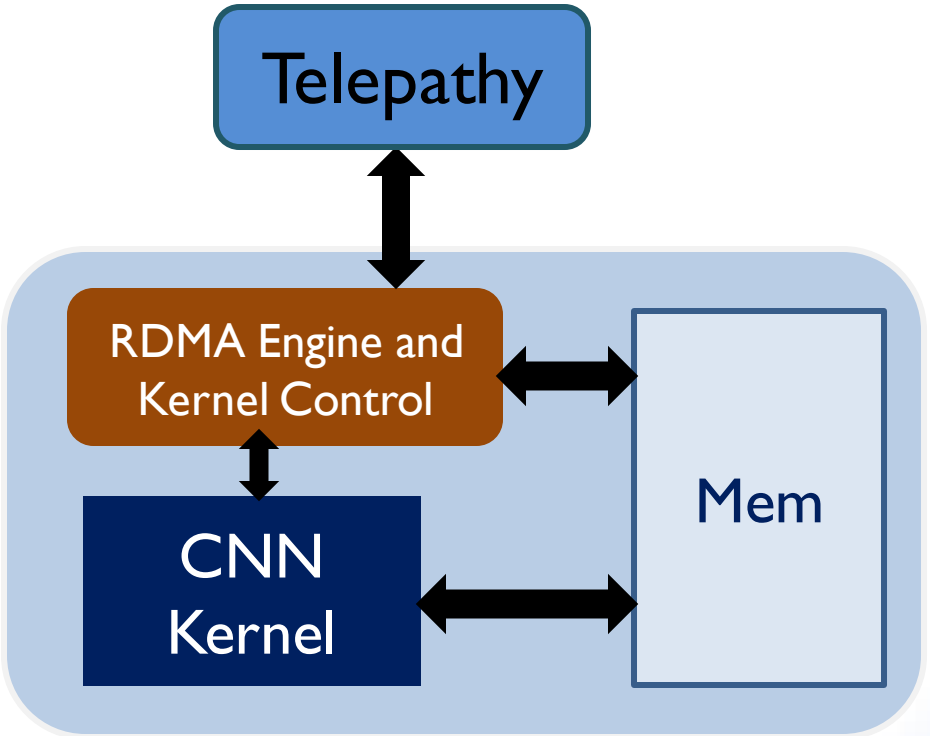
Single Node Inference Engine

- Controller written in HLS
- Stream in/stream out
- Streams in instructions for CNN kernel and DMA instructions for memory



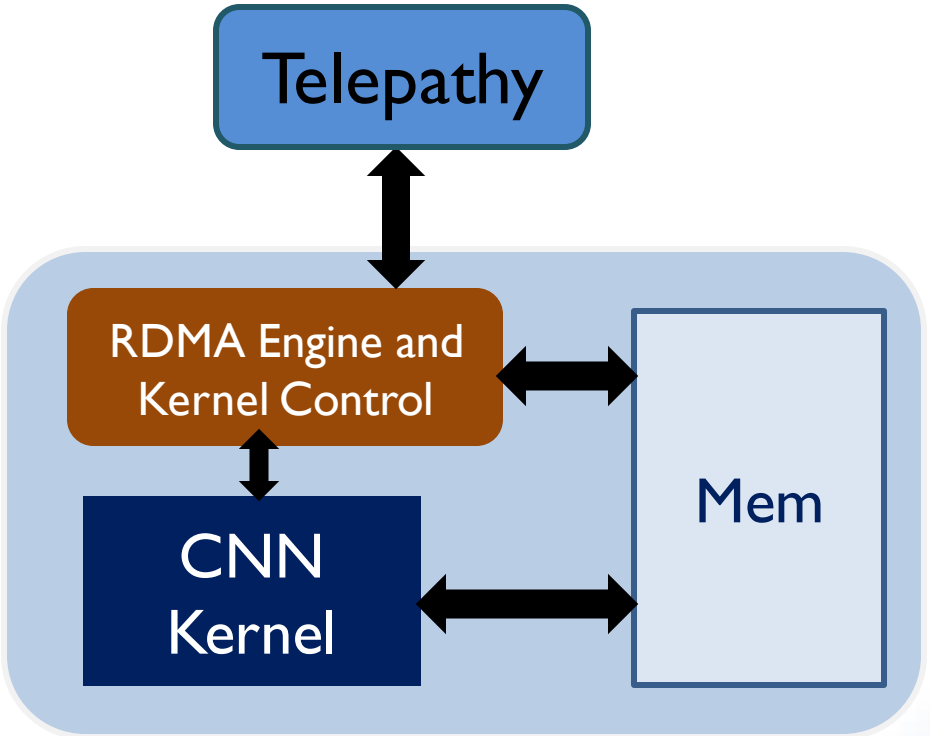
Single Node Inference Compiler

- Using neural net graph description generate the instructions for CNN kernel and DMA addresses
- Our compiler is called Telepathy



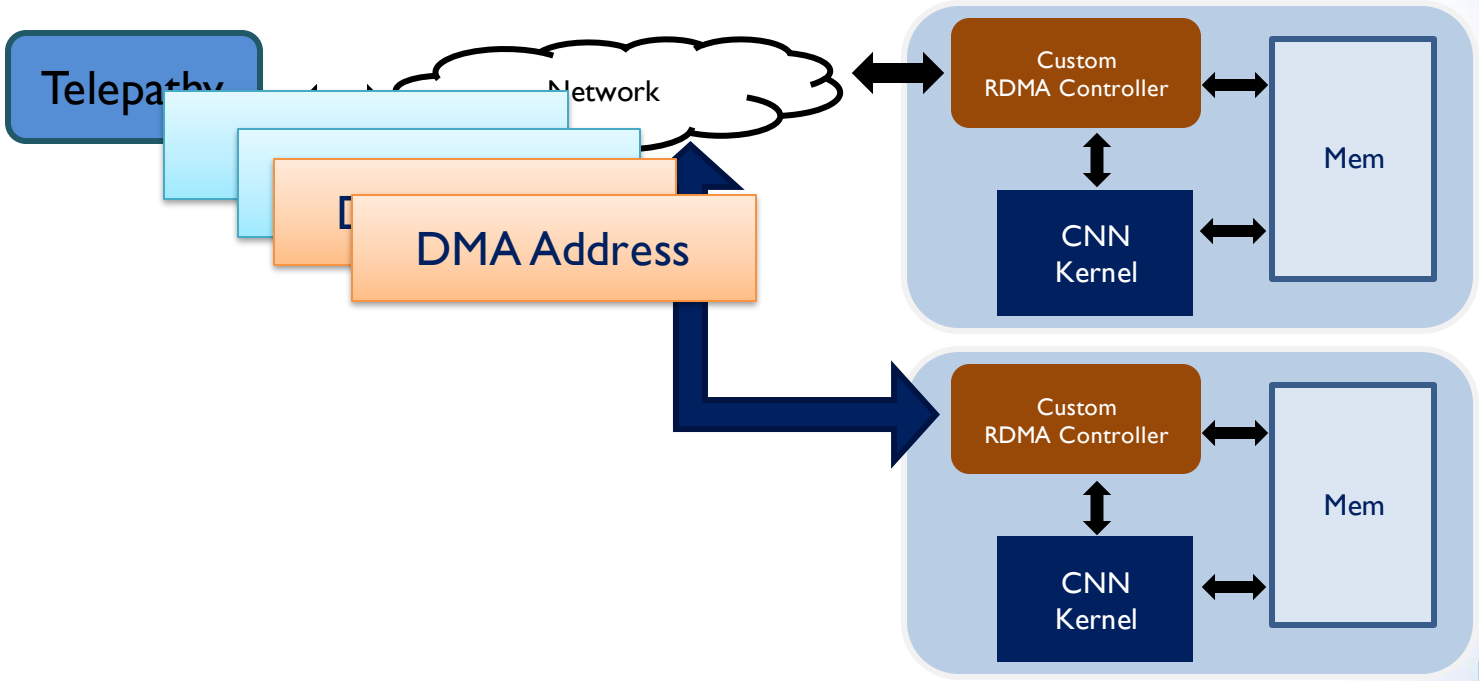
Telepathy Details

- Telepathy responsible for partitioning neural and keeping track of addresses of all nodes in cluster
  - Agnostic to CNN kernel implementation
- Generates instructions
  - Specific to CNN kernel implementation



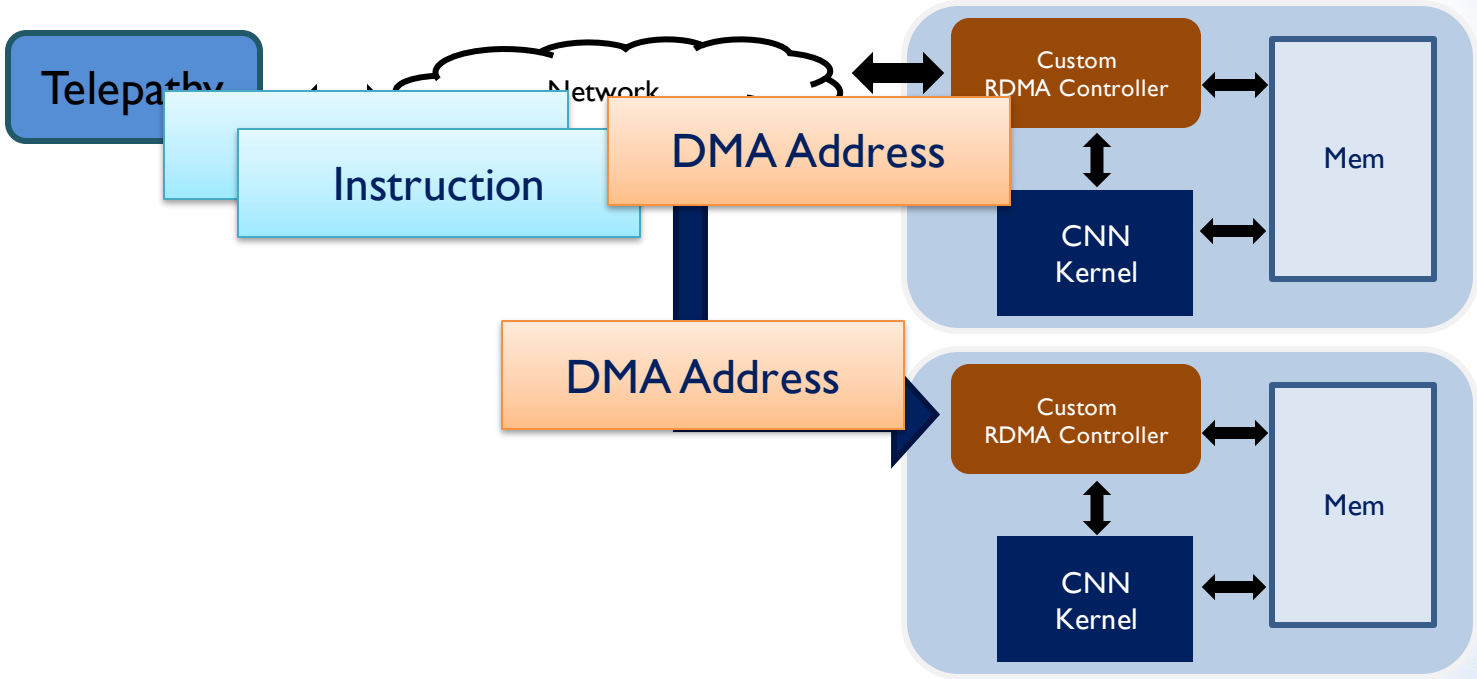
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



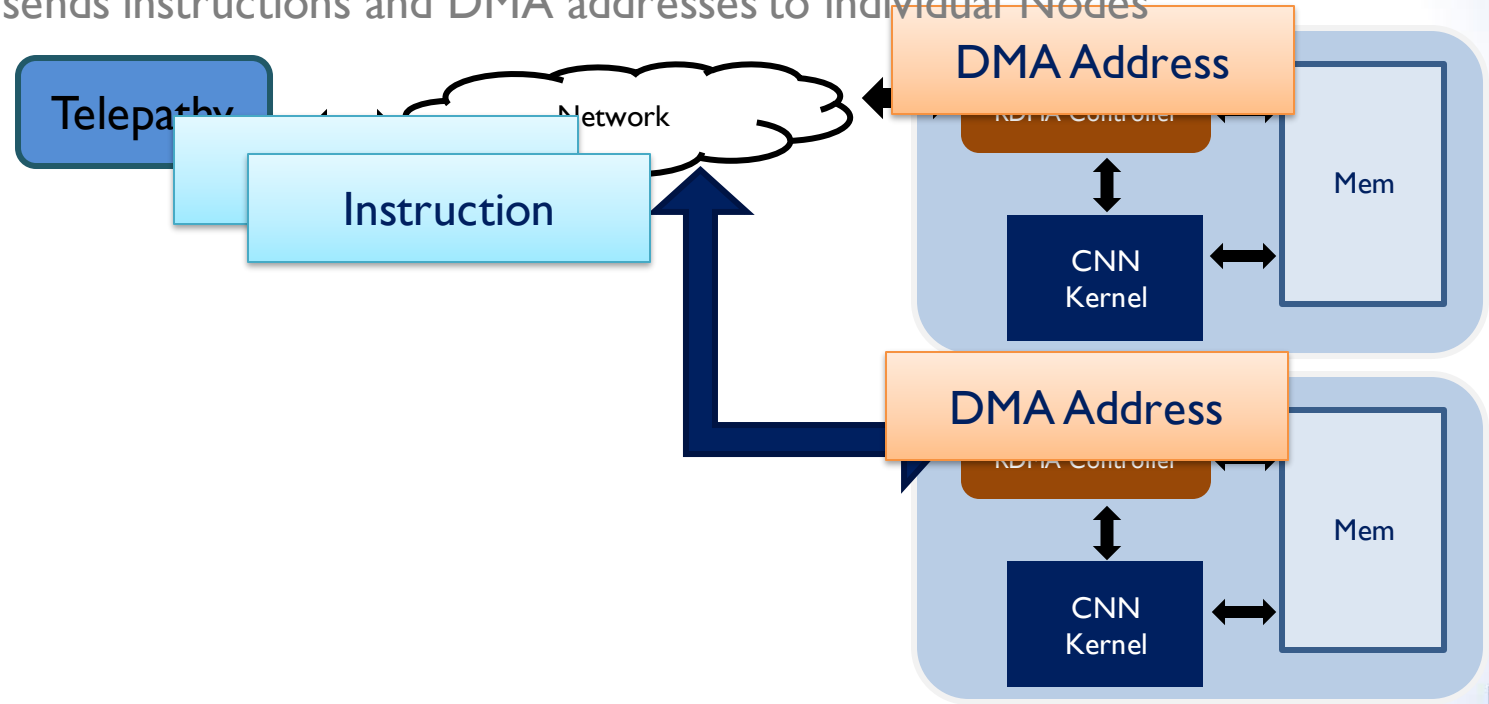
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



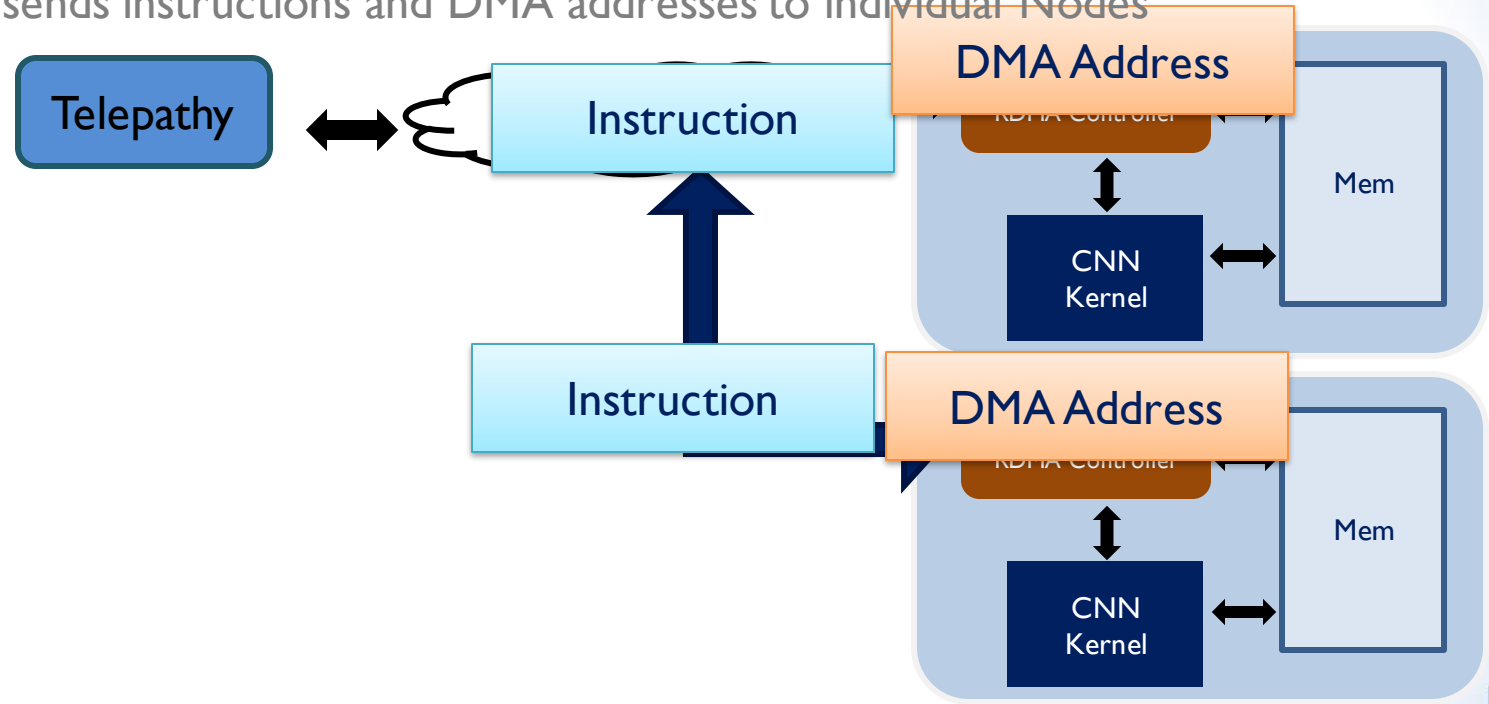
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



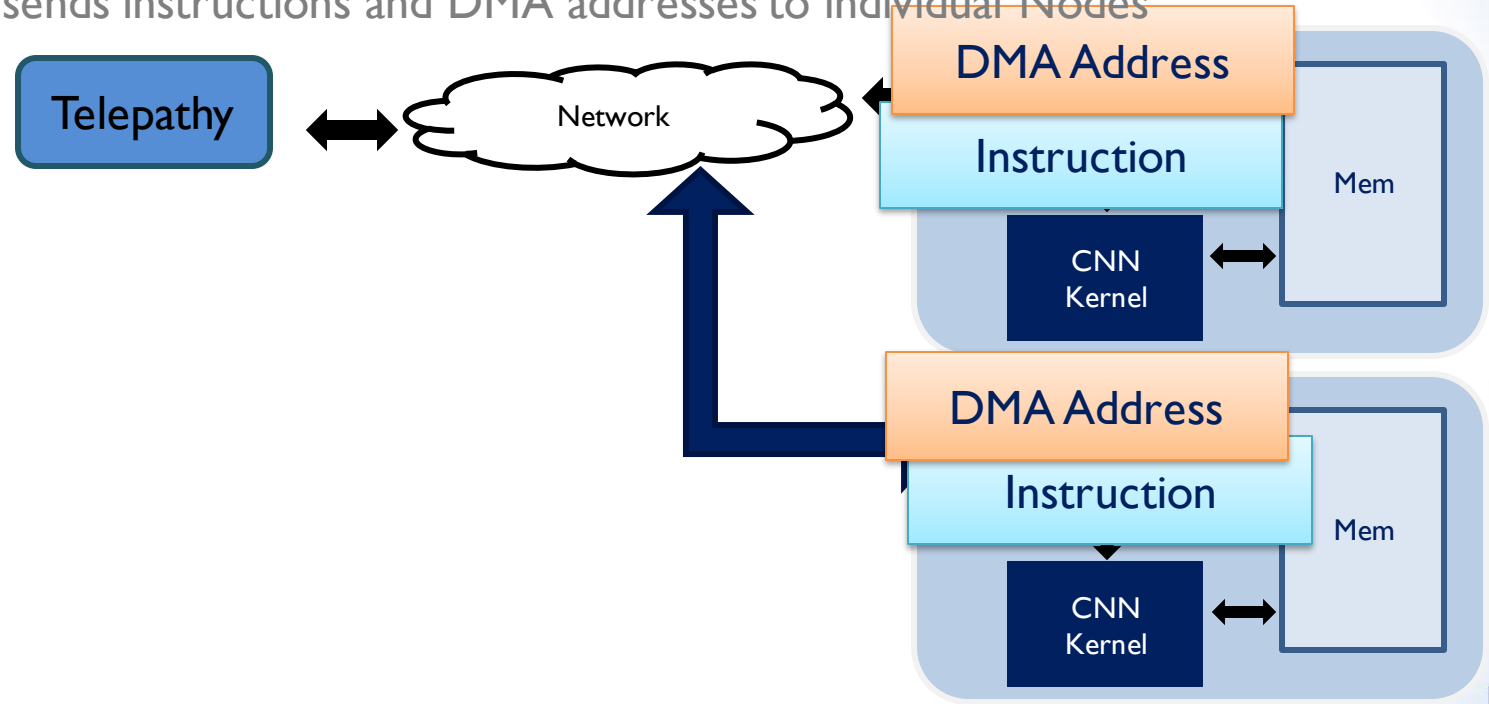
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



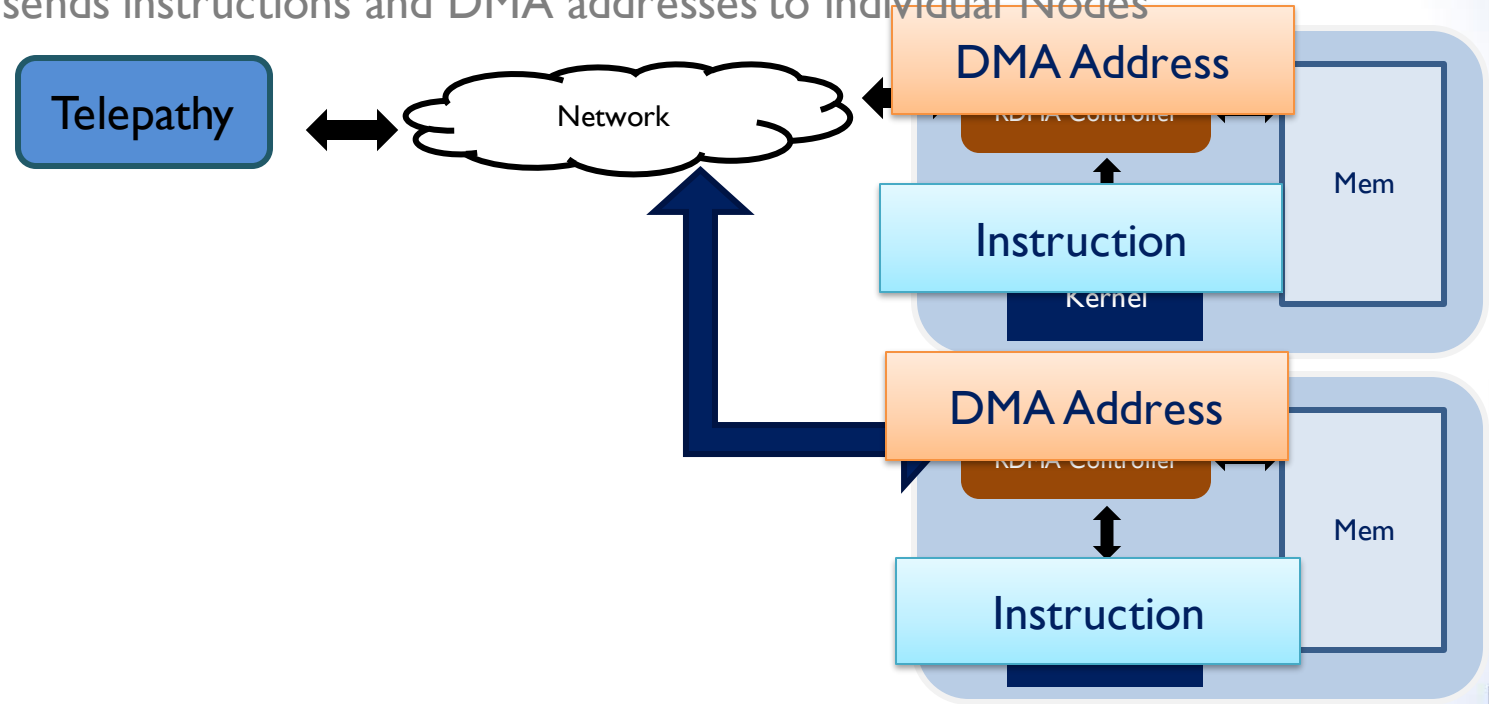
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



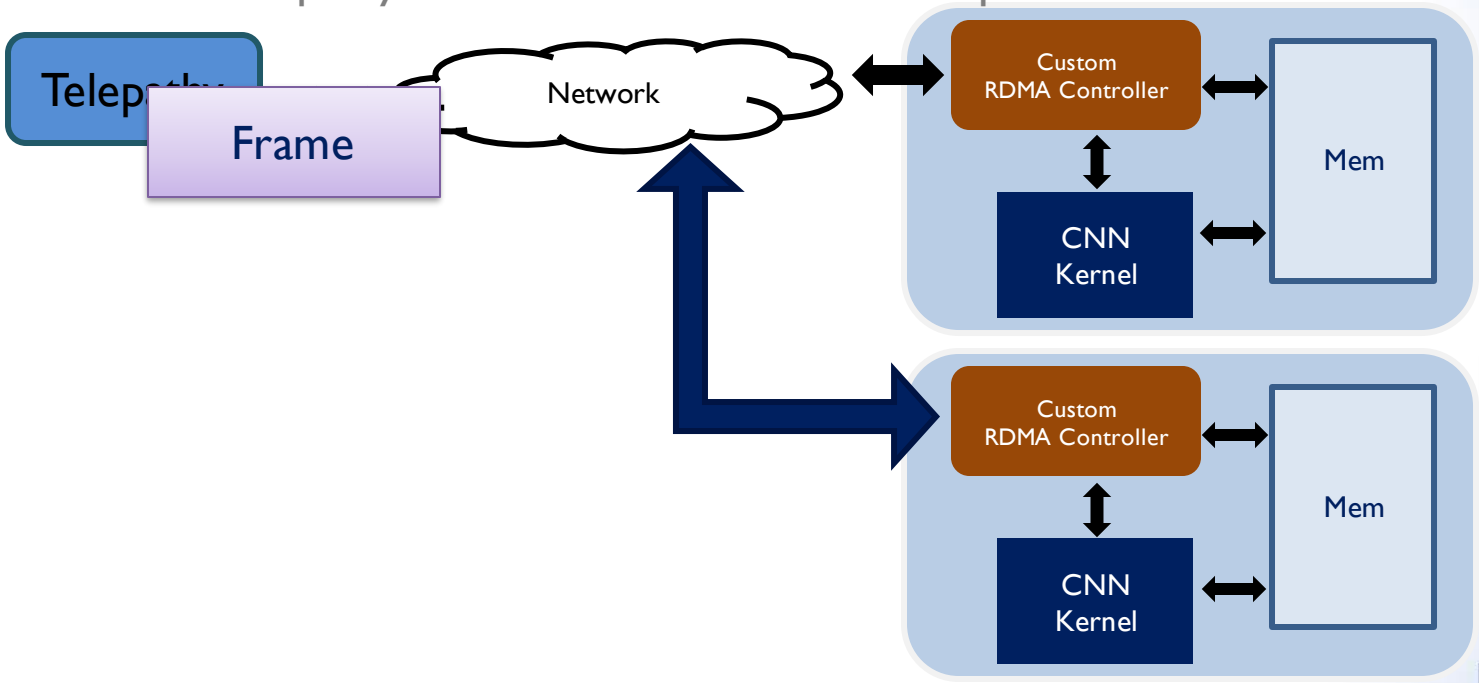
Multi-Node Inference

Telepathy sends instructions and DMA addresses to individual Nodes



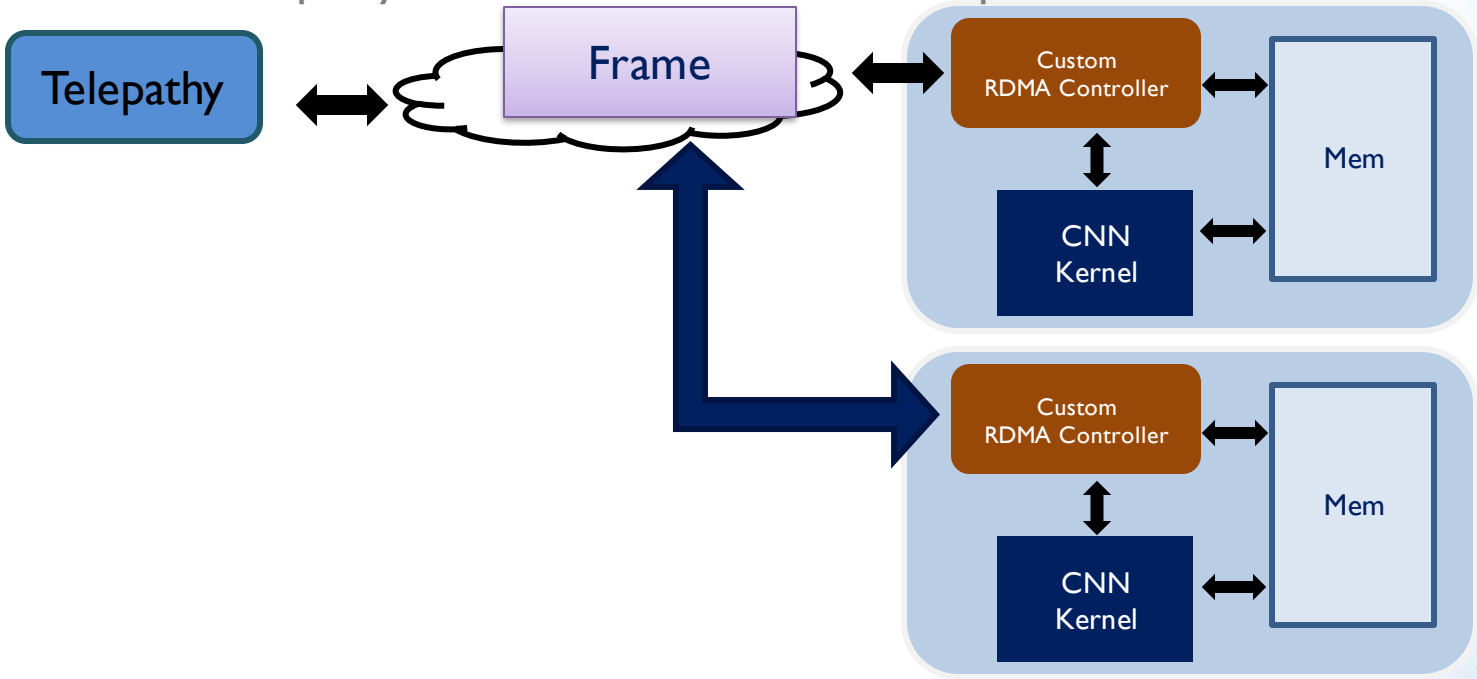
Multi-Node Inference

First frame sent from Telepathy to first Node via RDMA and processed



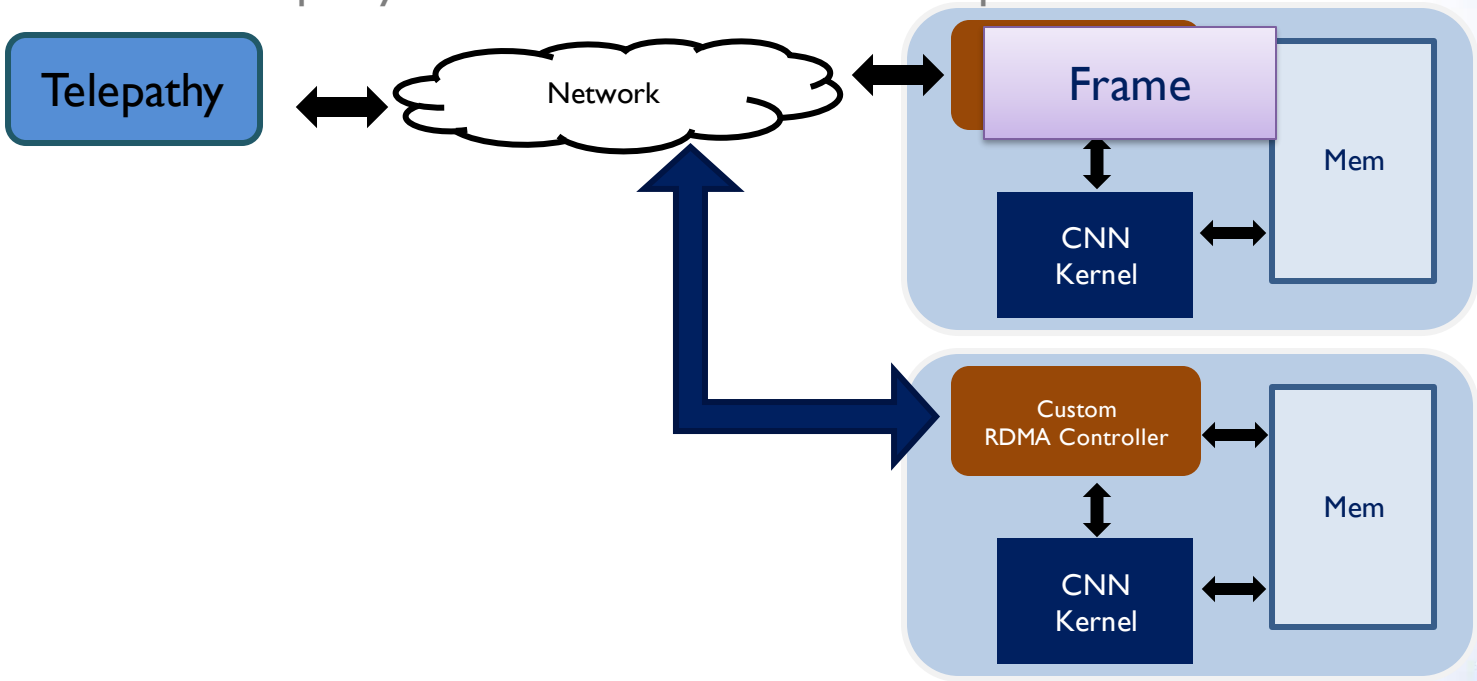
Multi-Node Inference

First frame sent from Telepathy to first Node via RDMA and processed



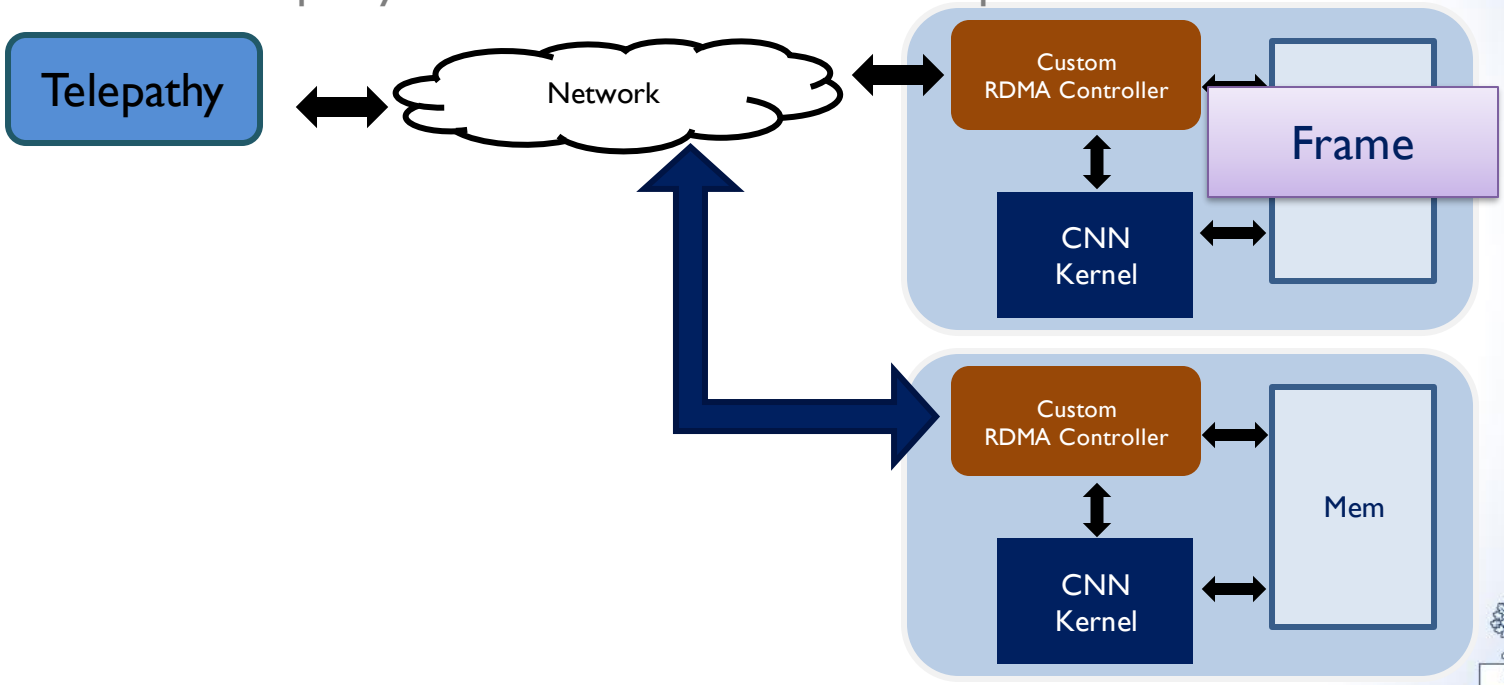
Multi-Node Inference

First frame sent from Telepathy to first Node via RDMA and processed



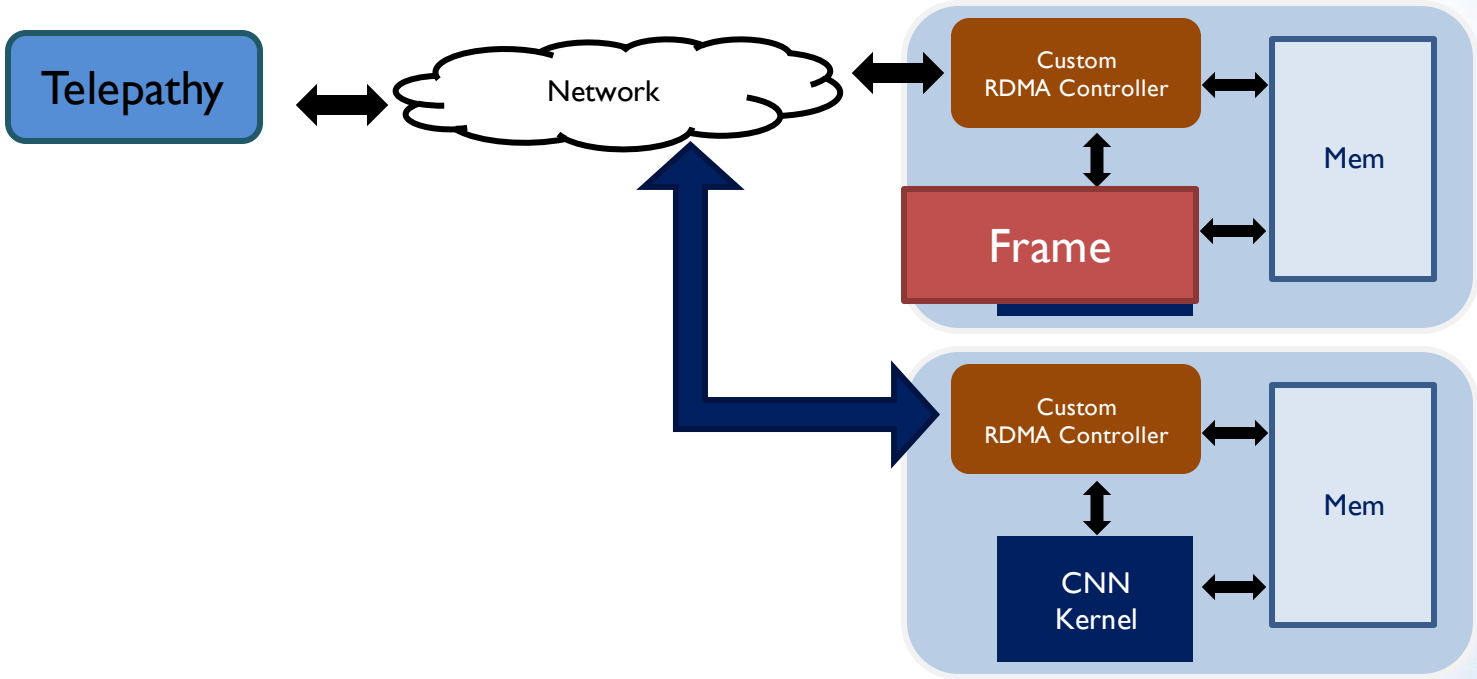
Multi-Node Inference

First frame sent from Telepathy to first Node via RDMA and processed



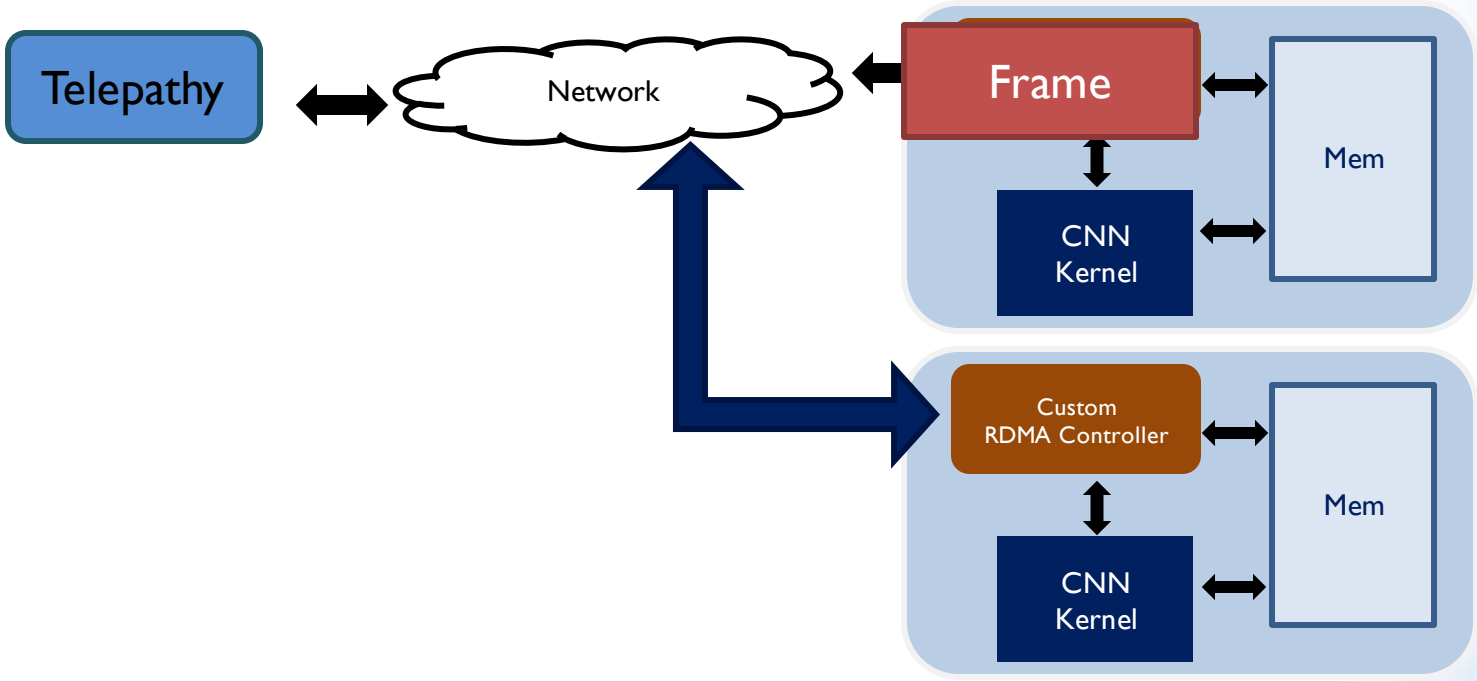
Multi-Node Inference

Processed frame sent to next Node



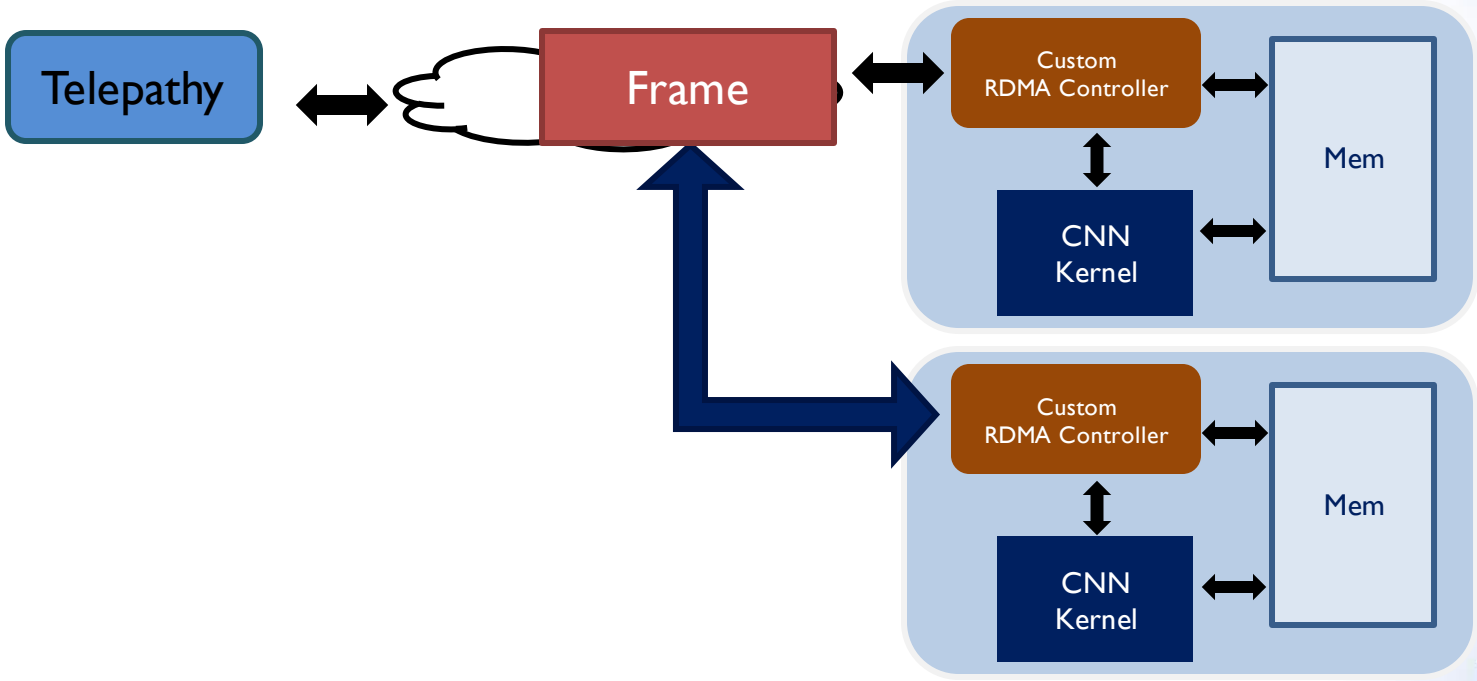
Multi-Node Inference

Processed frame sent to next Node



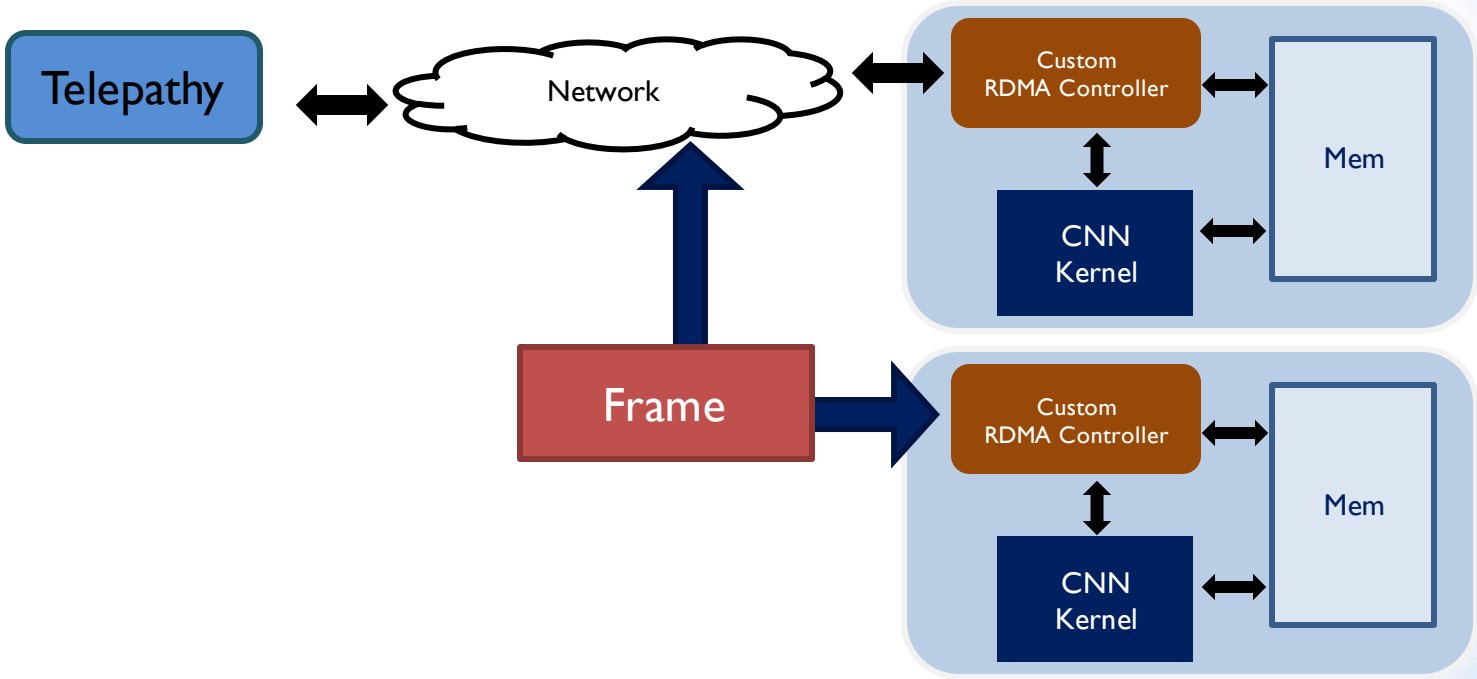
Multi-Node Inference

Processed frame sent to next Node



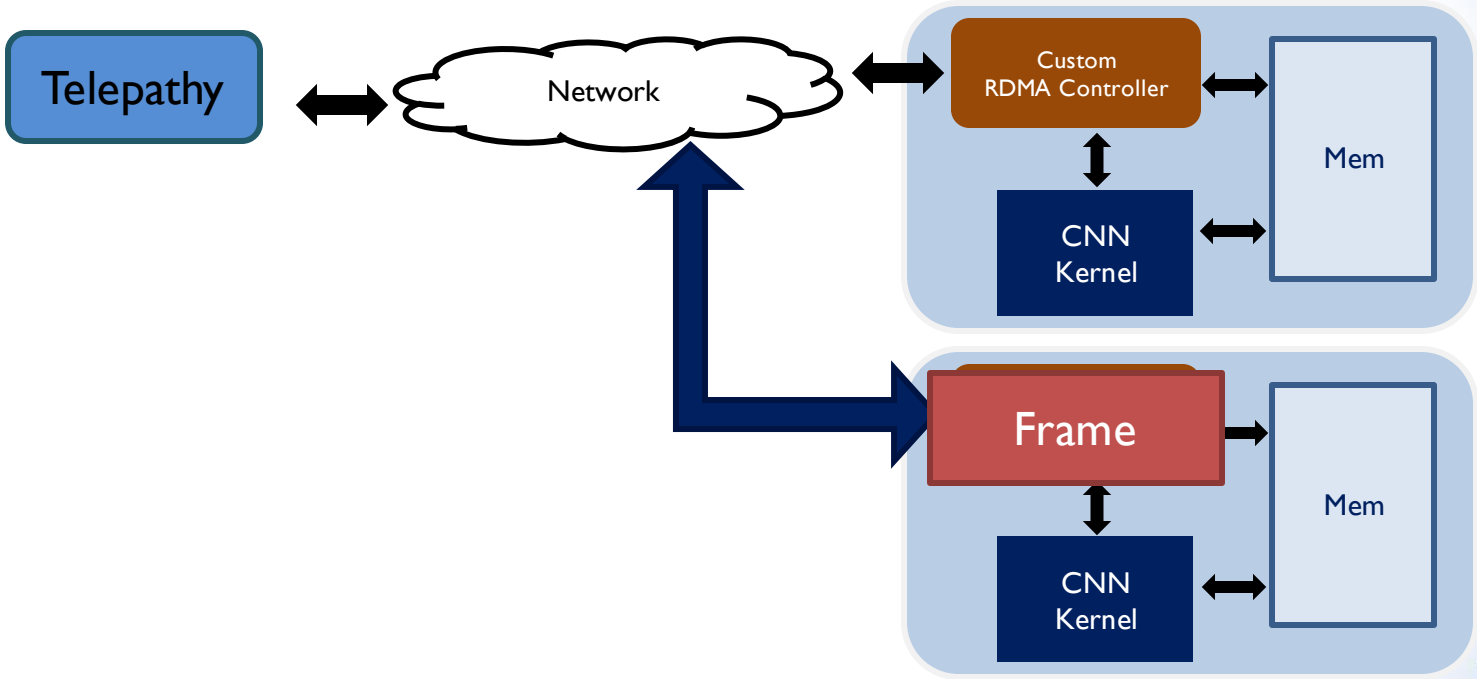
Multi-Node Inference

Processed frame sent to next Node



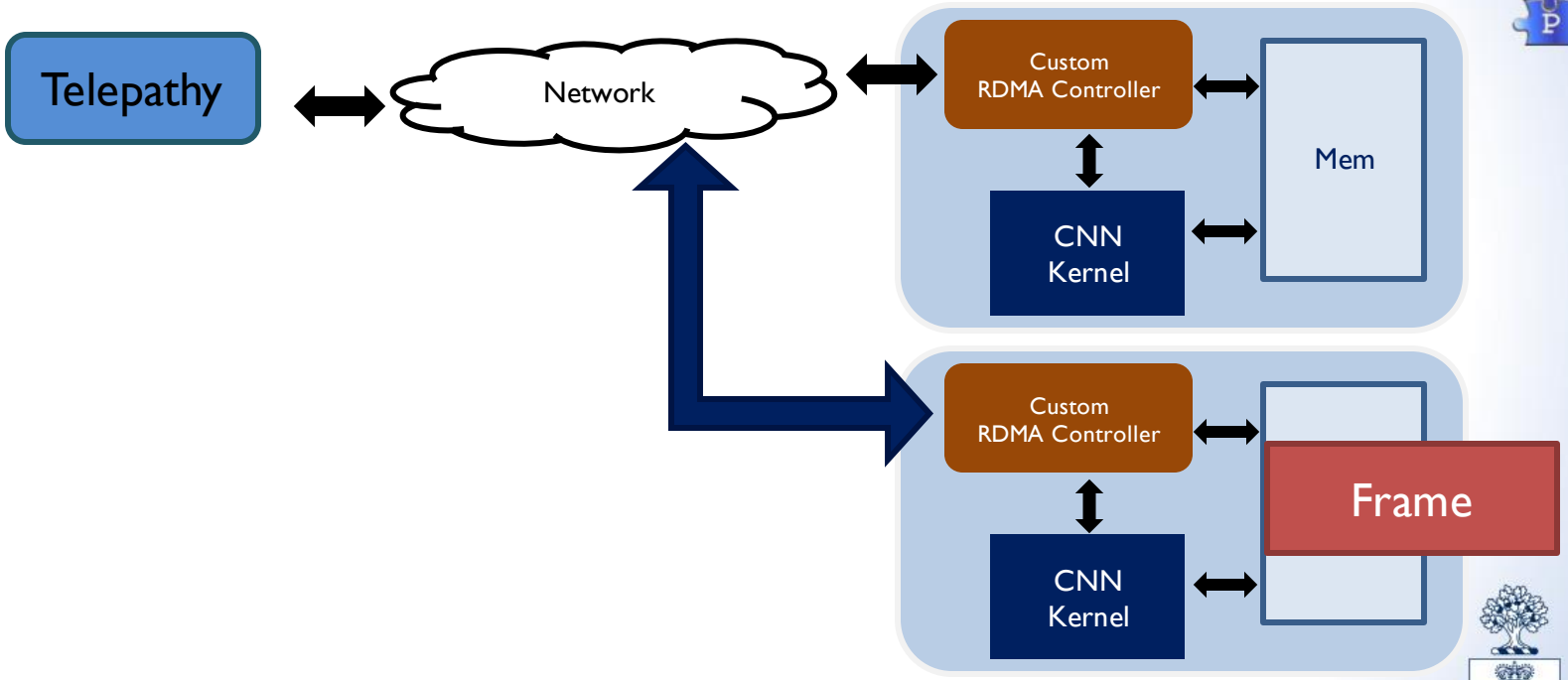
Multi-Node Inference

Processed frame sent to next Node



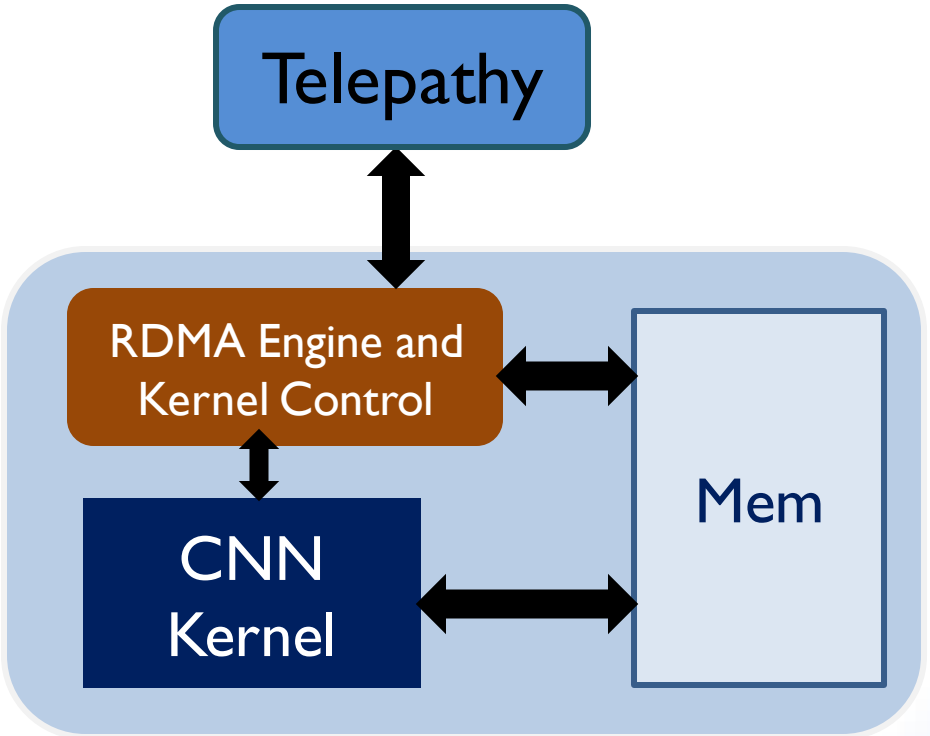
Multi-Node Inference

Processed frame sent to next Node



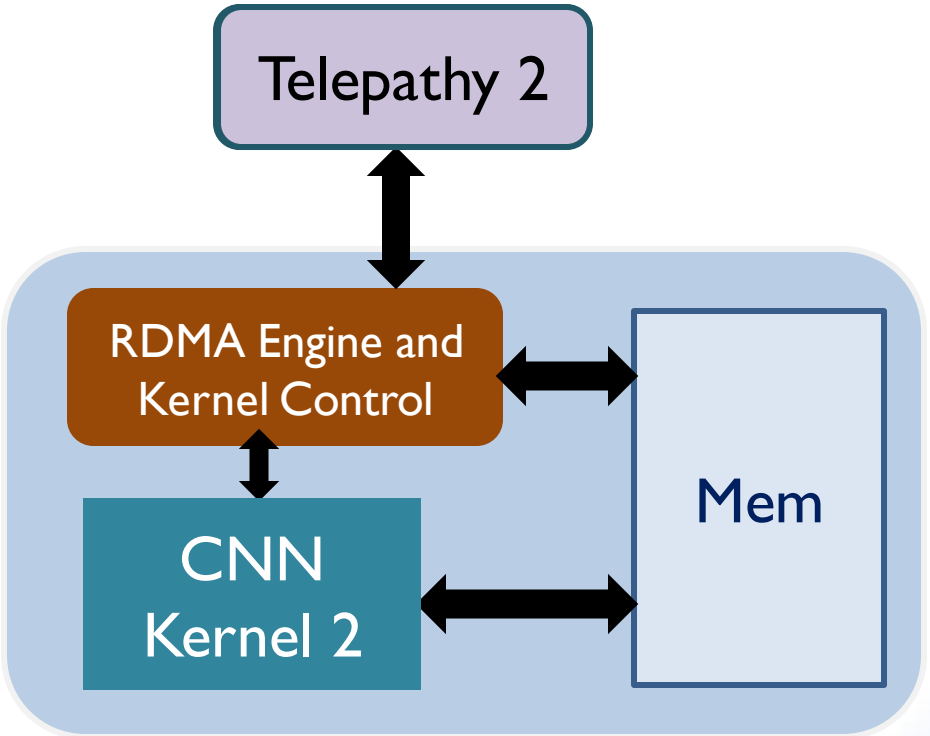
Different Use Case, Different Kernel

- The CNN kernel can be swapped with other CNN kernel implementations as long as Telepathy's instruction generator is modified



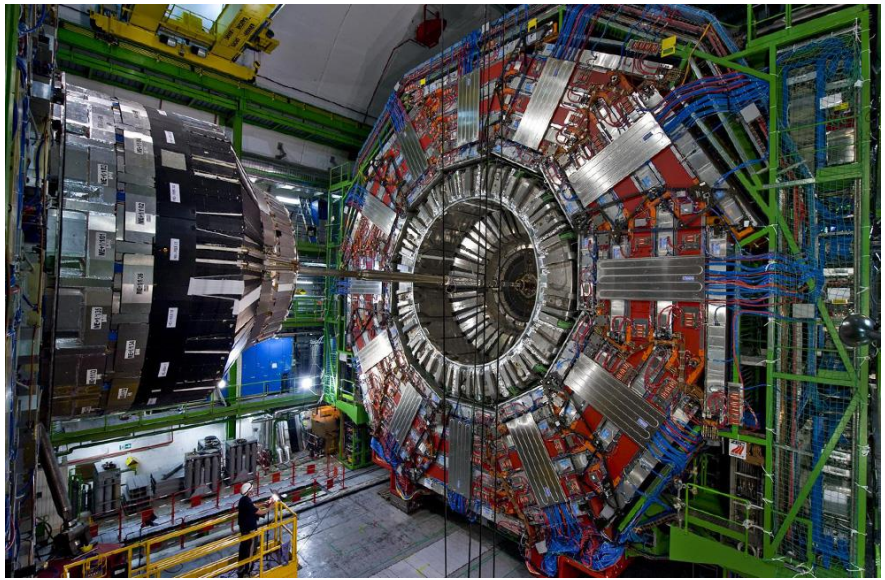
Different Use Case, Different Kernel

- The CNN kernel can be swapped with other CNN kernel implementations as long as Telepathy's instruction generator is modified



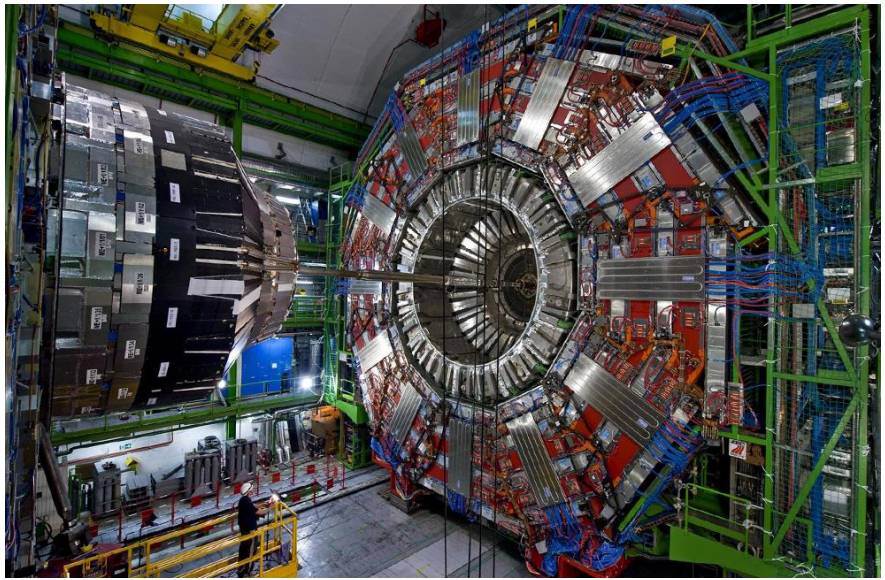
Current ML Applications

- Currently in collaboration with HLS4ML team use their kernel as a Galapagos kernel
- Second ML app using Xilinx kernel for data center work loads
- Same abstraction layers!



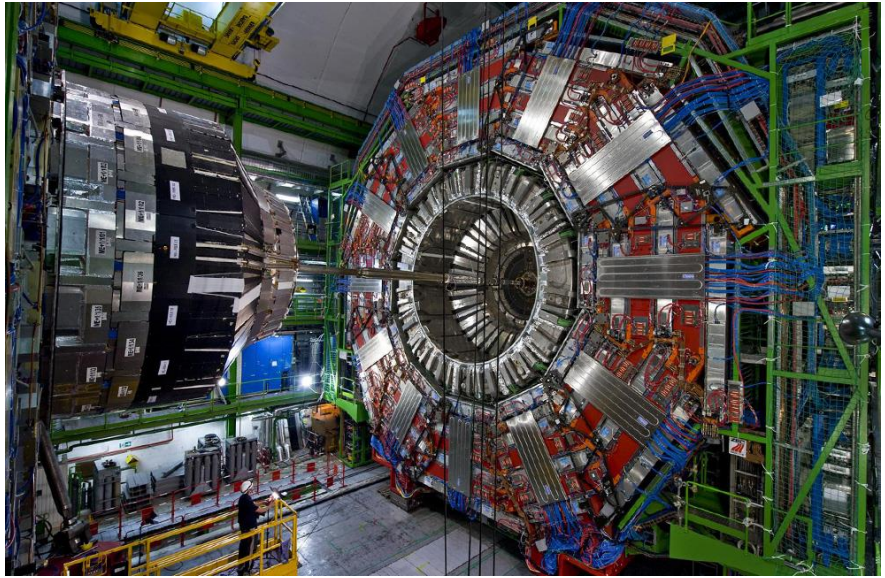
Current ML Applications

- Currently in collaboration with HLS4ML team use their kernel as a Galapagos kernel
- Second ML app using Xilinx kernel for data center work loads
- Same abstraction layers!



Current ML Applications

- Currently in collaboration with HLS4ML team use their kernel as a Galapagos kernel
- Second ML app using Xilinx kernel for data center work loads
- Same abstraction layers!



CONCLUDING REMARKS

Conclusion

- Created heterogeneous abstraction layer stack
- Makes it easy to connect different types of devices
- Makes formation of clusters more modular

Conclusion

- Created heterogeneous abstraction layer stack
- Makes it easy to connect different types of devices
- Makes formation of clusters more modular

Conclusion

- Created heterogeneous abstraction layer stack
- Makes it easy to connect different types of devices
- Makes formation of clusters more modular

Thanks!

Questions?

Email: naif.Tarafdar@mail.utoronto.ca