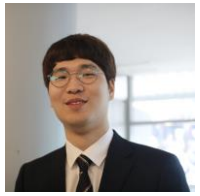
Georgia Tech School of Electrical and Computer Engineering
College of Engineering

http://synergy.ece.gatech.edu

ASTRA-SIM Description

Network Layer

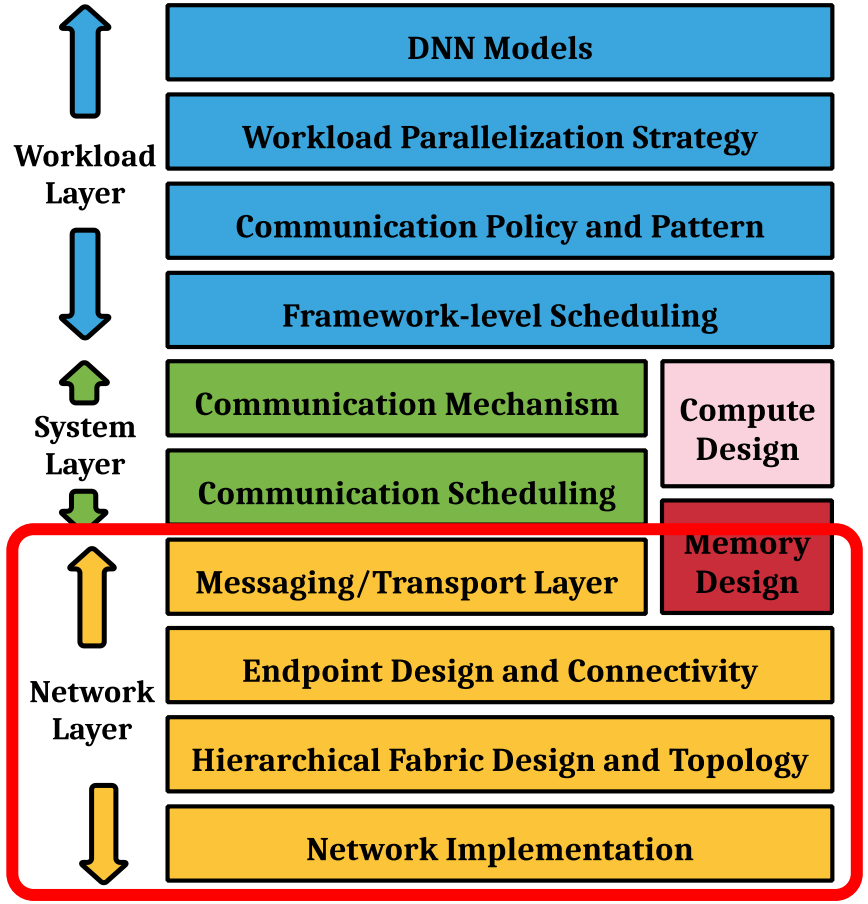


William Won
Ph.D. Student, School of Computer Science
Georgia Institute of Technology

Acknowledgments: Saeed Rashidi (Georgia Tech), Jinsun Yoo (Georgia Tech), Srinivas Sridharan (Meta), Sudarshan Srinivasan (Intel)

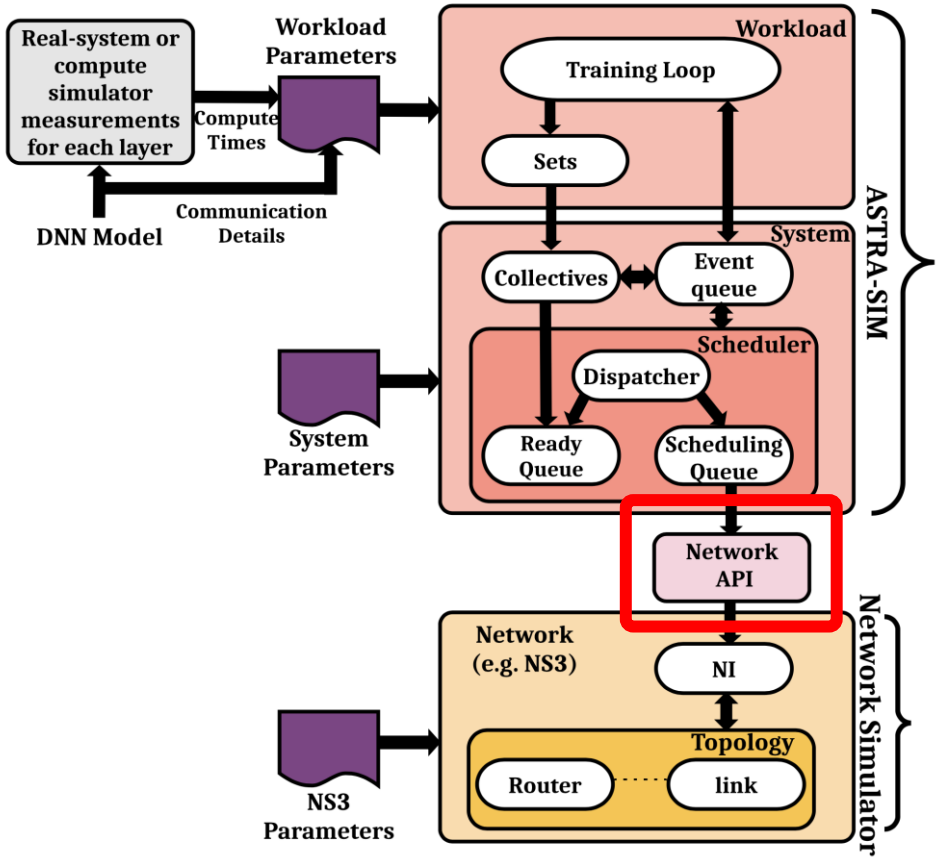
Network Layer

- Workload layer manages workload, parallelization, training loop
- System layer initiates collective communication
- Network layer simulates **actual network behaviors**
  - Communication protocols (TCP, RDMA, etc.)
  - Network topology
  - BW/latency per link
  - In-network collective communication
  - NIC offloading
  - Compression
  - Buffering, Arbitration



NetworkAPI

- **Interface between System layer and Network backend**
- Any network simulator implementing the NetworkAPI could be used as ASTRA-sim backend



(HOTI '20) Scalable Distributed Training of Recommendation Models: An ASTRA-SIM + NS3 case-study with TCP/IP transport

Example NetworkAPIs

- **`sim_send(msg_size, src, dest, callback)`**
  - **Simulate sending a message** of size msg_size from src through dest and **invoke callback function** once transmission has finished
- **`sim_recv(msg_size, src, dest, callback)`**
  - **Simulate receiving a message** of size msg_size from src through dest and **invoke callback function** once transmission has finished
- **`sim_schedule(delta, callback)`**
  - Invoke callback function after delta time
- **`sim_get_time()`**
  - Return current time of simulation to the frontend

Simulation Control Flow

- The main file (simulation entry point) is implemented inside network layer.
- Network layer creates corresponding **System and NetworkAPI** instances.
- Each system layer instance **internally** creates its workload layer instance.

Where do instantiations happen?
Analytical backend (analytical): analytical/analytical/main.cc
Analytical backend (congestion): analytical/congestion/main.cc
Garnet backend: garnet/gem5_astra/src/mem/ruby/network/garnet2.0/NetworkInterface.cc
ns3 backend: ns3-interface/simulation/scratch/AstraSimNetwork.cc

Network Layer

```
void main(){
        ⋮
Instantiate NetworkAPI[];
Instantiate System[];
        ⋮
for(auto &s:system){
    s->workload.fire();
}
process_all_events();
return;
}
```

Available Network Backends

- Network backends are maintained separately and are imported as **submodule**.

- We currently have **4 network backends** which implement NetworkAPI

Backend	Purpose	Notable Feature
analytical/analytical	analytical equation-based simulation	fast simulation, hierarchical topologies
analytical/congestion	congestion-aware analytical simulation	first-order congestion (queueing) modeling
Garnet	on-chip/scale-up network simulation	packetization, flow control, congestion
ns-3	inter-network simulation	large parallel GPU clusters

Analytical Backend

- Leverages **analytical equation** to estimate communication delay
- `delay(msg_size, src, dest) =`
  `(#hops)×(link latency) + (msg_size)/(link BW)`

  link delay — serialization delay
- `sim_send(msg_size, src, dest, callback)`
  - Estimate communication delay
  - Assign callback to event queue after delay
- No congestion modeling
  - Appropriate for topology-aware collectives without network congestion
- **Fast simulation** for large-scale systems

(ISPASS '23) ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-model Training at Scale

Congestion-aware Analytical Backend

- First-order congestion modeling by per-link queueing
- Per-link delay is calculated using analytical eqaution

- e.g., `send(msg_size: 1 MB, route: [1, 2, 3, 4, 5])`
  - `send(1 MB, 1 → 2)`
  - `send(1 MB, 2 → 3)`
  - `send(1 MB, 3 → 4)`
  - `send(1 MB, 4 → 5)`

  - **each send can be queued per each link**
  - **link processes pending chunks in-order**

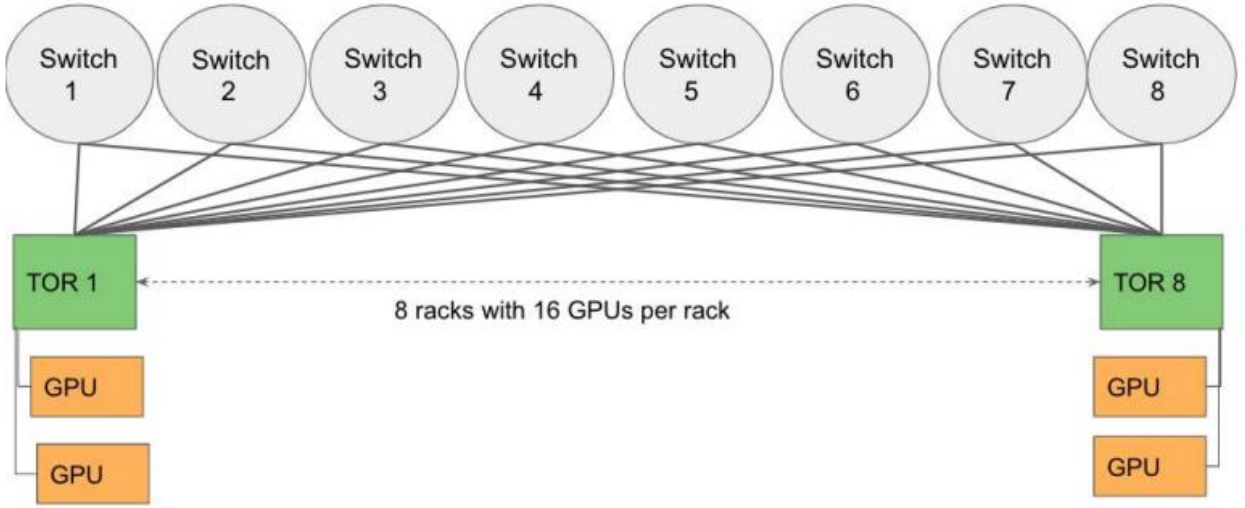
- **Fast simulation** for large-scale systems **with network congestion**

Garnet Backend

- Leverages **Garnet (interconnection network) simulator** as backend
- Appropriate for on-chip/scale-up networks
- Simulates interconnection network behaviors:
  - Message Packetization
  - Credit-based flow control
  - Congestion modeling
  - etc.
- Slower than analytical backend for large systems/models
- Supports switch-based/torus-based topologies

NS3 Backend

- Network simulator for **internet (inter-node) communication**
- Used to model ML training in **largely parallel GPU clusters**
- 1 NPU in Analytcial/Garnet becomes **1 GPU** in NS3, connected with ToR/spine switch, etc.



(HOTI '22) Current RoCE congestion control methods have little impact on ML training workloads

Slide courtesy of: Jinsun Yoo (Georgia Tech)

Sample Network Input

Garnet network input

sample_torus.txt

```
num-npus: 12
num-packages: 6
package-rows: 3
topology: Torus3D
local-rings: 2
vertical-rings: 1
horizontal-rings: 1
flit-width: 2048
local-packet-size: 4096
package-packet-size: 4096
tile-link-width: 256
package-link-width: 256
vcs-per-vnet: 50
routing-algorithm: Ring_XY
router-latency: 1
local-link-latency: 90
package-link-latency: 200
buffers-per-vc: 5000
local-link-efficiency: 1.0
package-link-efficiency: 1.0
```

Analytical network input

sample_Torus3D.json

```
{
  "topology-name": "Hierarchical",
  "topologies-per-dim": ["Ring", "Ring", "Ring"],
  "dimension-type": ["N", "N", "N"],
  "dimensions-count": 3,
  "units-count": [2, 2, 3],
  "links-count": [2, 2, 2],
  "link-latency": [10, 100, 100],
  "link-bandwidth": [32, 16, 16],
  "nic-latency": [0, 0, 0],
  "router-latency": [0, 0, 0],
  "hbm-latency": [500, 500, 500],
  "hbm-bandwidth": [370, 370, 370],
  "hbm-scale": [0, 0, 0]
}
```