https://astra-sim.github.io

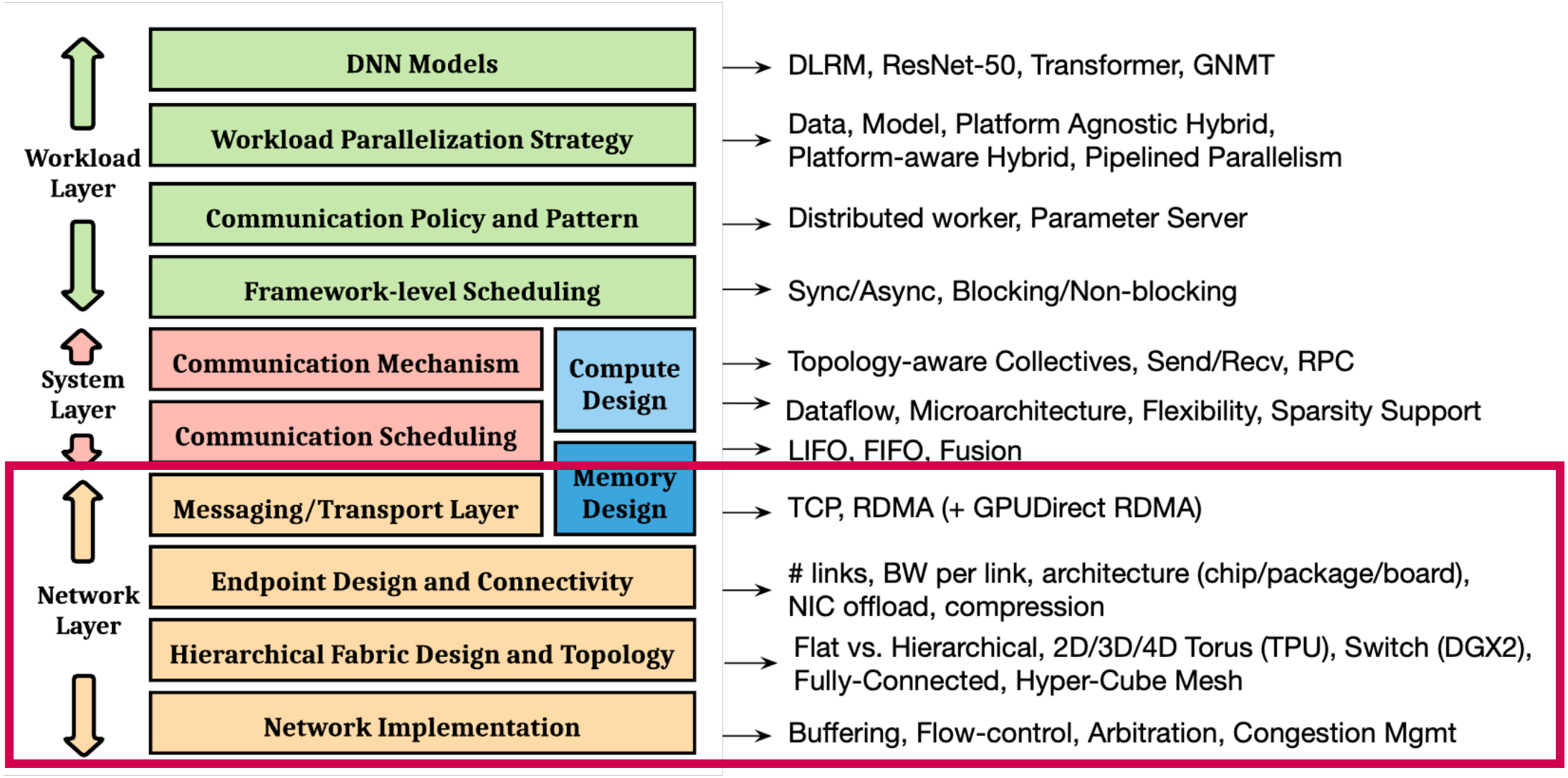
https://github.com/mlcommons/chakra

ASTRA-sim Tutorial
MICRO 2024
Nov 3, 2024

ASTRA-sim and Chakra Tutorial:
Network Layer

Will Won
Ph.D. Candidate
School of CS, Georgia Institute of Technology
william.won@gatech.edu

Design Space of Distributed ML

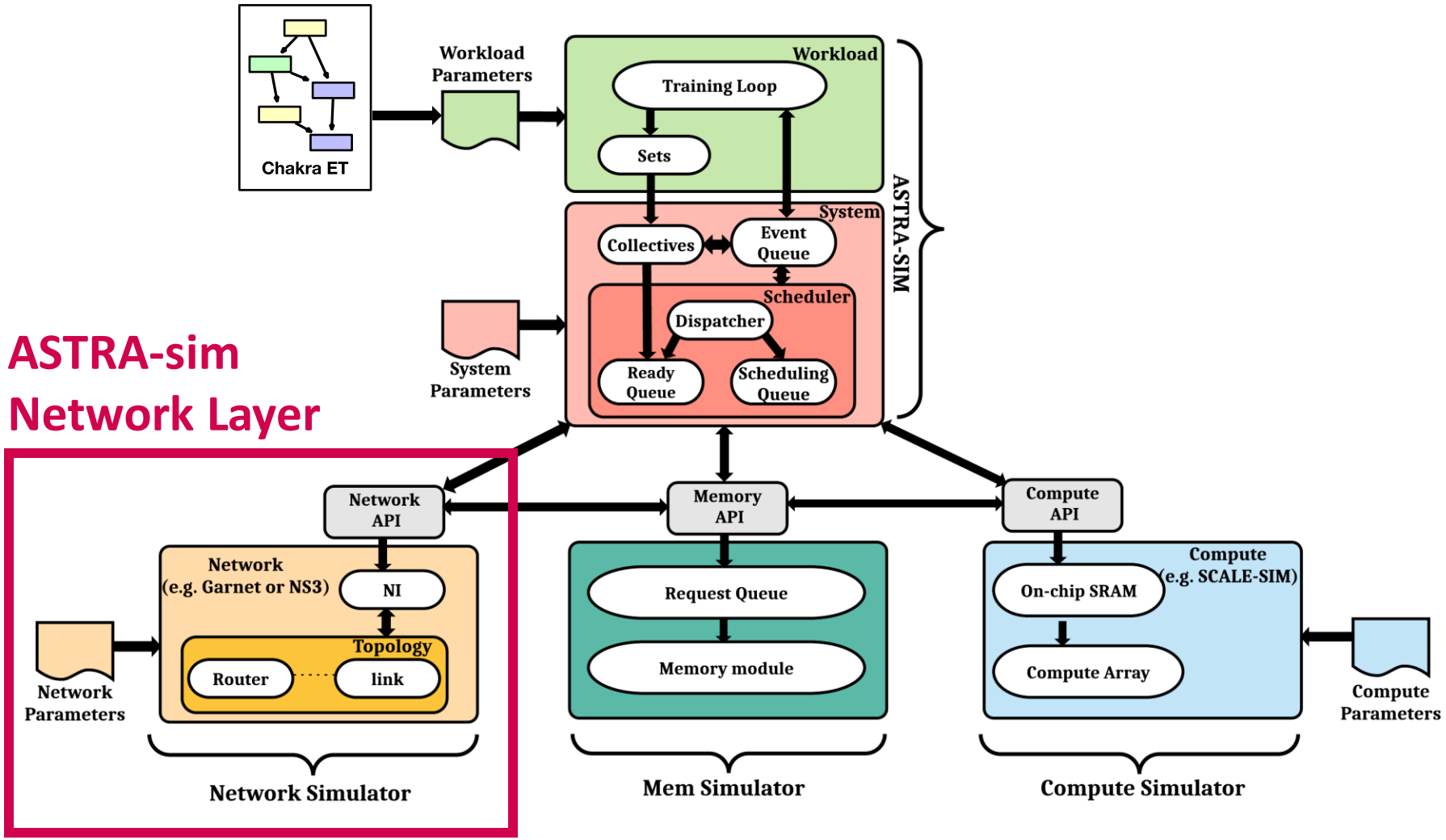


Network Layer

- 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

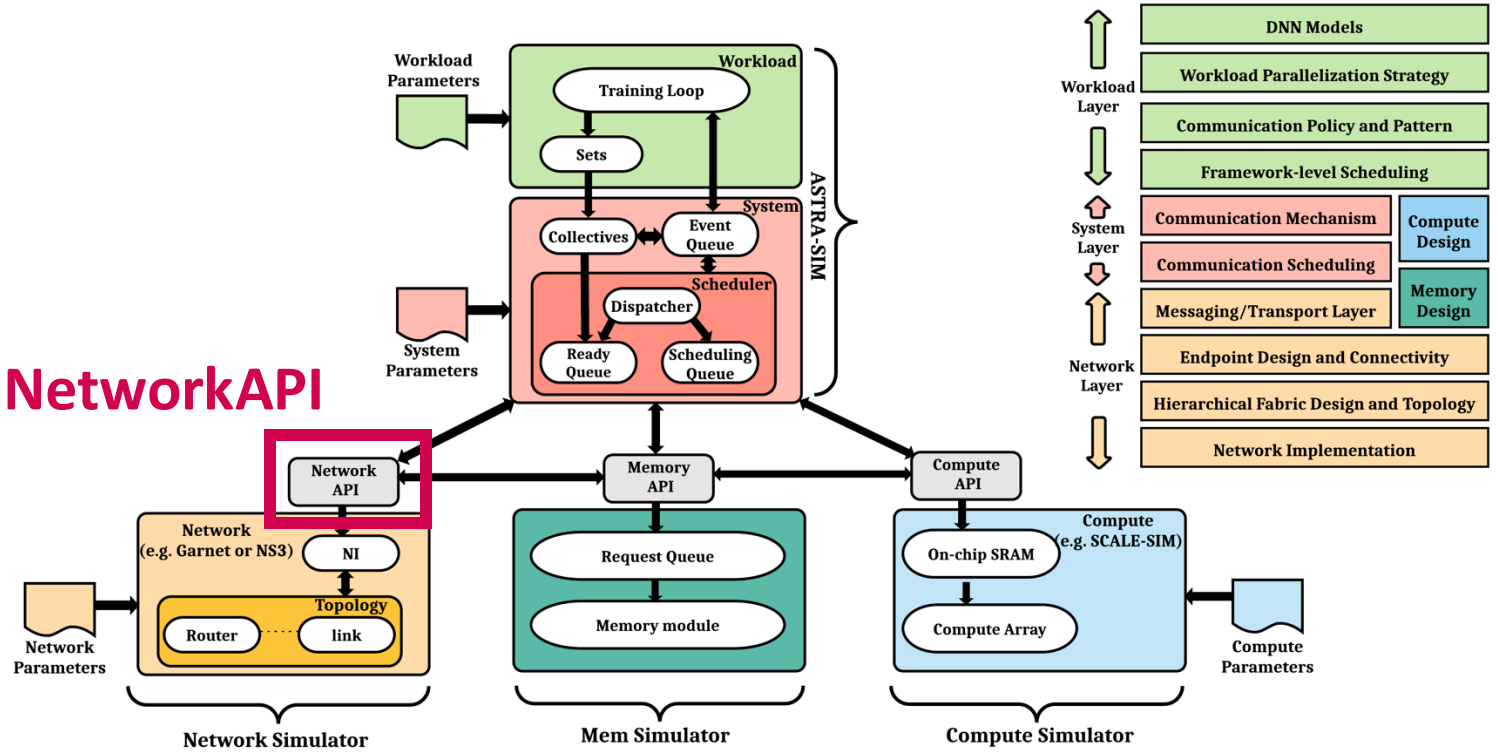
- Through easy **plug-and-play** of any **network simulators**
  - Enabled via **NetworkAPI**

ASTRA-sim: Network Layer



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, 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, 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

Declared at: astra-sim/common/AstraNetworkAPI.hh

NetworkAPI at System Layer

- Ring All-Reduce algorithm implementation

```
bool Ring::ready() {
        (...)
        stream->owner->sim_send(0, Sys::dummy_data, msg_size, UINT8, packet.preferred_dest, stream-
        >stream_id,
        &snd_req, &Sys::handleEvent, nullptr);

        (...)
        stream->owner->sim_recv(0, Sys::dummy_data, msg_size, UINT8, packet.preferred_src, stream-
        >stream_id,
        &rcv_req, &Sys::handleEvent, ehd);

        reduce();
        return true;
}
```

Send a chunk

Receive a chunk

Example NetworkAPI Implementation

```
timespec_t CommonNetworkApi::sim_get_time() {
(...)
    const auto current_time = event_queue->get_current_time();
    return current_time;
}
```

query the network backend of current (simulation) time

NetworkAPI Implementation varies by network simulation backend

Example: sim_schedule

```
void sim_schedule(
    timespec_t delta,
    void (*fun_ptr)(void* fun_arg),
    void* fun_arg);
```

delta time to schedule this event

event handler to be invoked

event handler argument

Example NetworkAPI Implementation

```
void sim_schedule(const timespec_t delta,
void (*fun_ptr)(void*),
void* const fun_arg) {
        (...)
        const auto current_time = sim_get_time();
        const auto event_time = current_time.time_val + delta.time_val;

        event_queue->schedule_event(event_time_ns, fun_ptr, fun_arg);
}
```

compute event time

schedule the event

Example: sim_send

```
int sim_send(
        void* buffer,
        uint64_t count,
        int type,
        int dst,
        int tag,
        sim_request* request,
        void (*msg_handler)(void* fun_arg),
        void* fun_arg);
```

message size (in Bytes)

destination NPU ID

chunk identifier

event handler

event handler argument

NetworkAPI Implementation: Example

- Ring All-Reduce algorithm implementation

```
int CongestionAwareNetworkApi::sim_send(...) {
        (...)
        // create chunk
        auto chunk_arrival_arg = std::tuple(tag, src, dst, count, chunk_id);
        auto arg = std::make_unique<decltype(chunk_arrival_arg)>(chunk_arrival_arg);
        const auto arg_ptr = static_cast<void*>(arg.release());
        const auto route = topology->route(src, dst);
        auto chunk = std::make_unique<Chunk>();

        // initiate transmission from src -> dst.
        topology->send(std::move(chunk));
}
```

Select a static route

Trigger actual network simulation

NetworkAPI Implementation varies by network simulation backend

NetworkAPI Implementation: Example

```
void process_chunk_arrival(...) {
    (...)

    if (entry.value()->both_callbacks_registered()) {
        entry.value()->invoke_send_handler();
        entry.value()->invoke_recv_handler();

        // remove entry
        tracker.pop_entry(tag, src, dest, count, chunk_id);
    }
    (...)
}
```

when the chunk arrives dest

run the callback functions (to notify ASTRA-sim)

NetworkAPI Implementation varies by network simulation backend

Example: sim_schedule

```
int sim_send(
        void* buffer,
        uint64_t count,
        int type,
        int dst,
        int tag,
        sim_request* request,
        void (*msg_handler)(void* fun_arg),
        void* fun_arg);
```

message size (in Bytes)

destination NPU ID

chunk identifier

event handler

event handler argument

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

Caveat: Garnet currently only works with ASTRA-sim 1.0 (deprecated)

Analytical Backend

- Leverages **analytical equation** to estimate communication delay

- `delay(msg_size, src, dest) =`
  `(#hops)×(link latency) + (msg_size)/(link BW)`

- 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

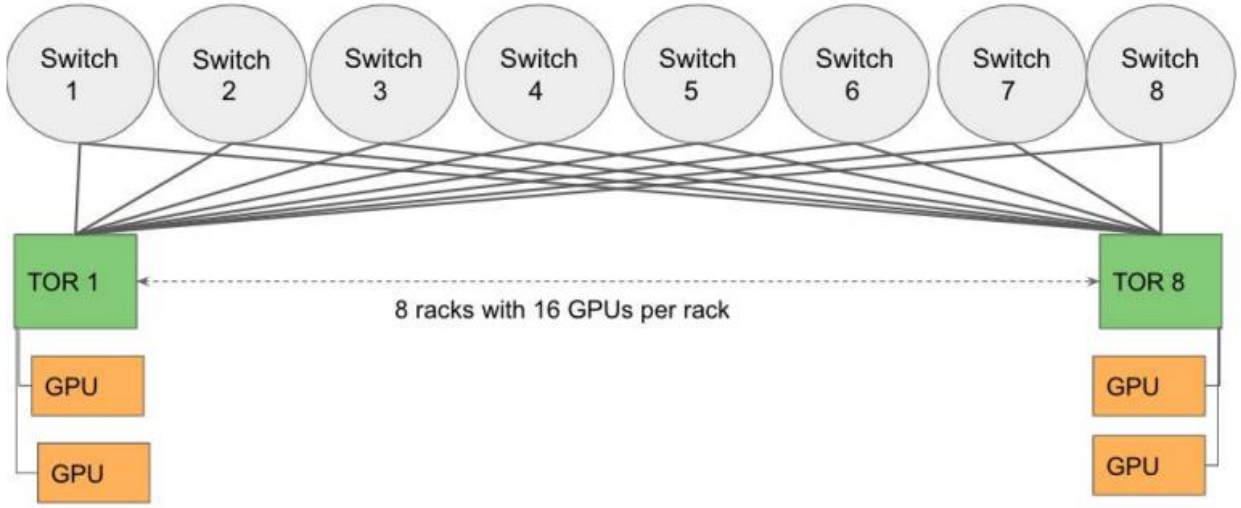
- e.g., `send(msg_size: 1 MB, route: [1, 2, 3, 4, 5])`
  - `send(1 MB, 1 → 2)`
  - `send(1 MB, 2 → 3)`
  - `send(1 MB, 3 → 4)`
  - `send(1 MB, 4 → 5)`

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

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

ns-3 Backend

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



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

Slide courtesy: Jinsun Yoo <jinsun@gatech.edu>

ns-3 Network Configurations

- Detailed internetwork behavior modeling/simulation

PACKET_PAYLOAD_SIZE	packet size
CC_MODE	Congestion control algorithm
BUFFER_SIZE	switch buffer size
	0: ACK has same priority with data packet
ACK_HIGH_PRIO	1: prioritize ACK
RATE_BOUND	Bound rate to a limited rate
ENABLE_QCN	Whether QCN (Quantized Congestion Notification) is enabled
L2_BACK_TO_ZERO	(Go-Back-N protocol) Layer 2 go back to zero transmission
L2_CHUNK_SIZE	(Go-Back-N protocol) Layer 2 chunk size
L2_ACK_INTERVAL	(Go-Back-N protocol) Layer 2 Ack intervals
HAS_WIN	Whether to use a window
	0: different server pairs use their own RTT as T
GLOBAL_T	1: use the max base RTT as the global T
VAR_WIN	Whether the window size is variable
RATE_BOUND	Use rate limitor
ACK_HIGH_PRIO	Prioritize acknowledgement packets
KMAX_MAP	a map from link bandwidth to ECN threshold kmax
KMIN_MAP	a map from link bandwidth to ECN threshold kmin
PMAX_MAP	a map from link bandwidth to ECN threshold pmax
RATE_AI	Rate increment unit in AI period
RATE_HAI	Rate increment unit in hyperactive AI period
MIN_RATE	Minimum rate of a throttled flow

ns-3 Network Topology

```
9 1 8
8
8 0 400Gbps 0.0005ms 0
8 1 400Gbps 0.0005ms 0
8 2 400Gbps 0.0005ms 0
8 3 400Gbps 0.0005ms 0
8 4 400Gbps 0.0005ms 0
8 5 400Gbps 0.0005ms 0
8 6 400Gbps 0.0005ms 0
8 7 400Gbps 0.0005ms 0
```

#total deices, #switches, #links

switch device IDs

link information

Slide courtesy: Jinsun Yoo <jinsun@gatech.edu>

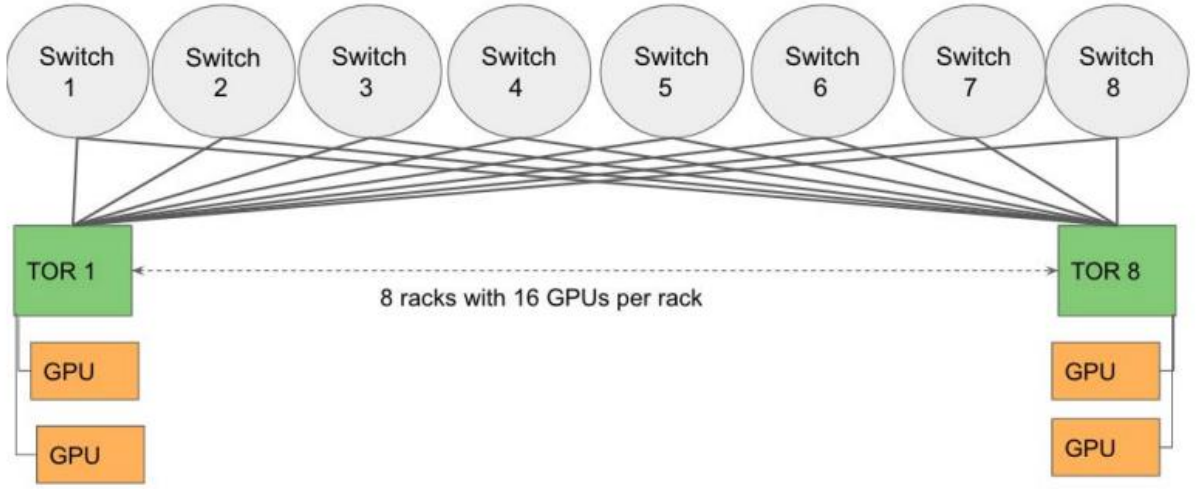
ns-3 Network Topology

```
144 16 192
128 129 130 … 142 143
0 128 200Gbps 0.005ms 0
1 128 200Gbps 0.005ms 0
2 128 200Gbps 0.005ms 0
3 128 200Gbps 0.005ms 0
128 136 200Gbps 0.0125ms 0
128 137 200Gbps 0.0125ms 0
128 138 200Gbps 0.0125ms 0
```

#total deices, #switches, #links

switch device IDs

link information



Slide courtesy: Jinsun Yoo <jinsun@gatech.edu>