清華大學 交叉信息研究院

Institute for Interdisciplinary Information Sciences, Tsinghua University

清華大學 Tsinghua University

Evaluating Chiplet-based Large-Scale Interconnection Networks via Cycle-Accurate Packet-Parallel Simulation

Yinxiao Feng@, Yuchen Wei, Dong Xiang, Kaisheng Ma*

Tsinghua University

Thursday, July 11, 2024

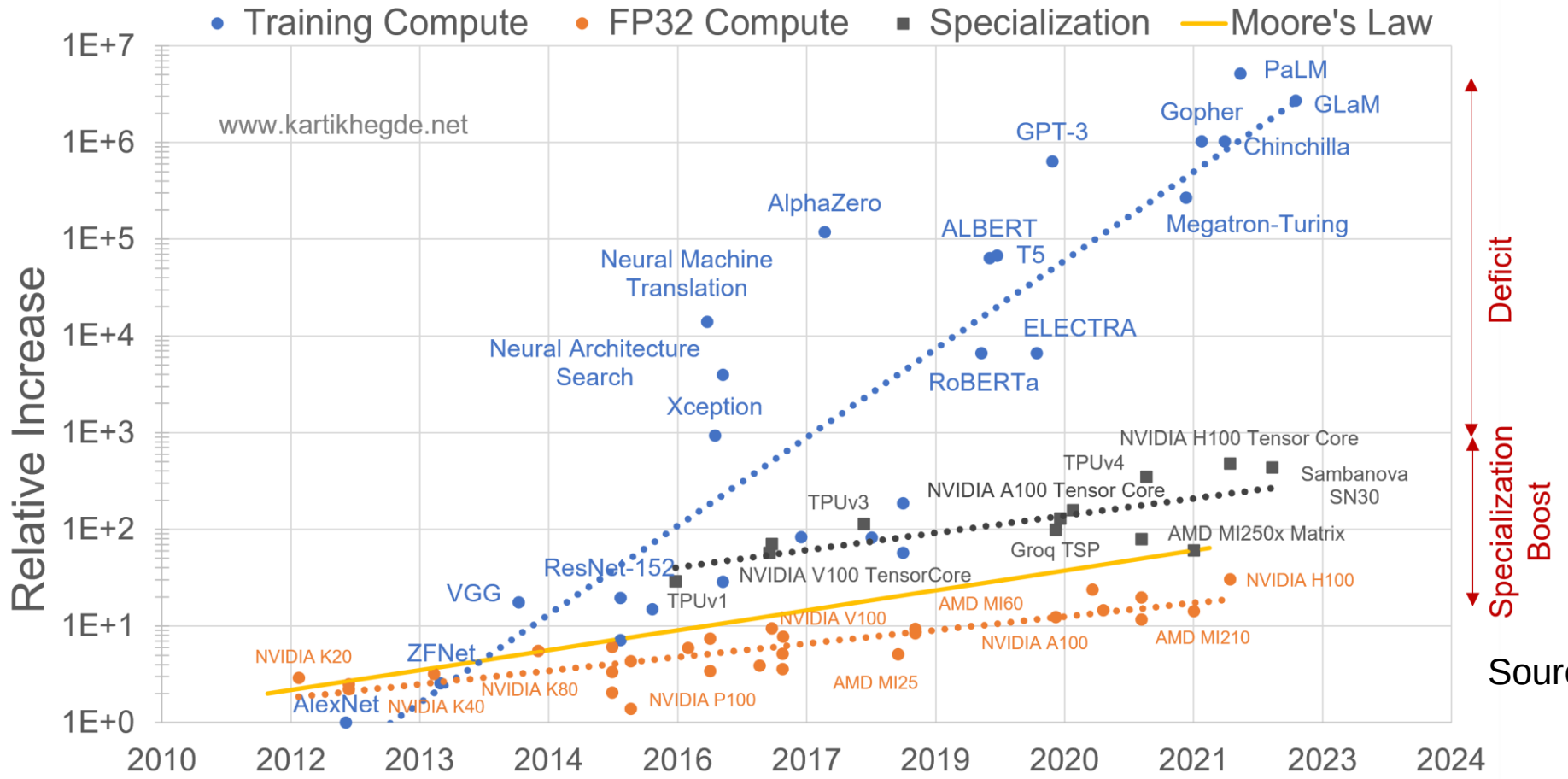
* Corresponding Author

@ Presenter

Outline

- Background
  - Trends, status, imitations, and challenges of **chiplet-based networks**
- Chiplet Network Simulator
  - Packet-centric architecture
  - Atomic-based packet-parallel simulation
  - Hyper-threading inconsistency
  - Other features
- Evaluation Results
- Discussion

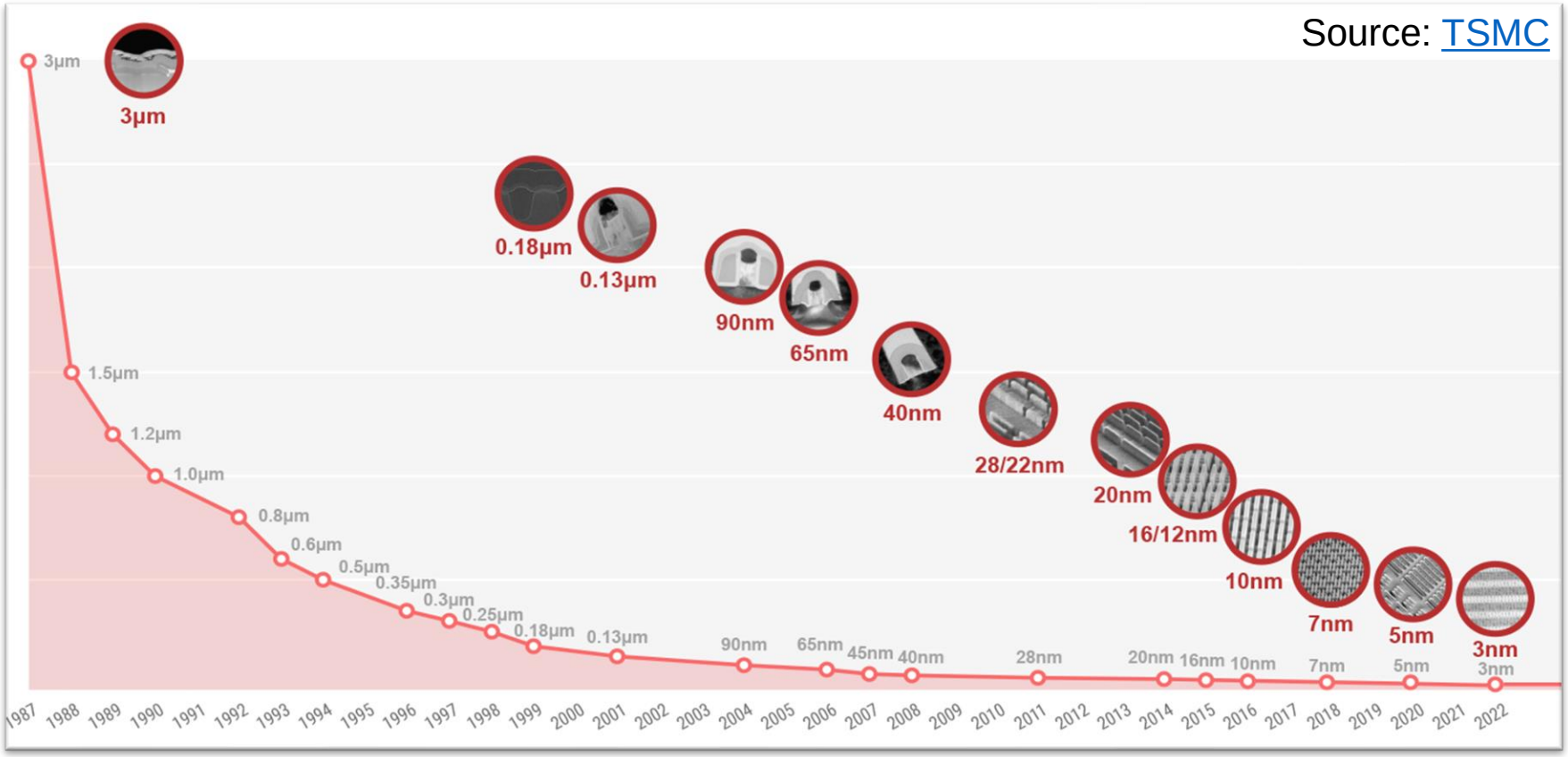
Background: Trends



Source: [Kartik Hegde]

Workloads (especially AI) grow much faster than the hardware.
The large gap is calling for a more scalable scale-up method.

Background: Post-Moore Era



Graphcore (2021)
GC200
TSMC 7nm
823 mm²

Nvidia (2022)
GH100
TSMC 4nm
814 mm²

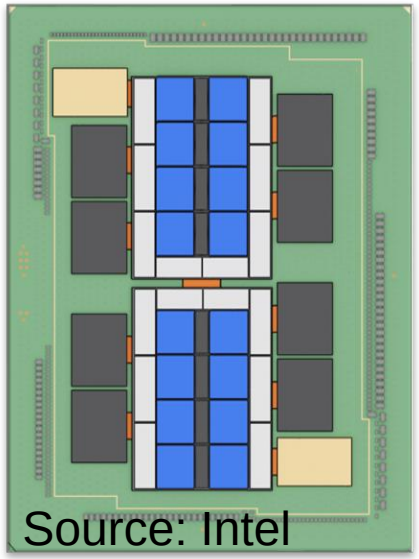
Lithographic reticle area limit
26mm×33mm=858 mm²

Transistors = Density × Area

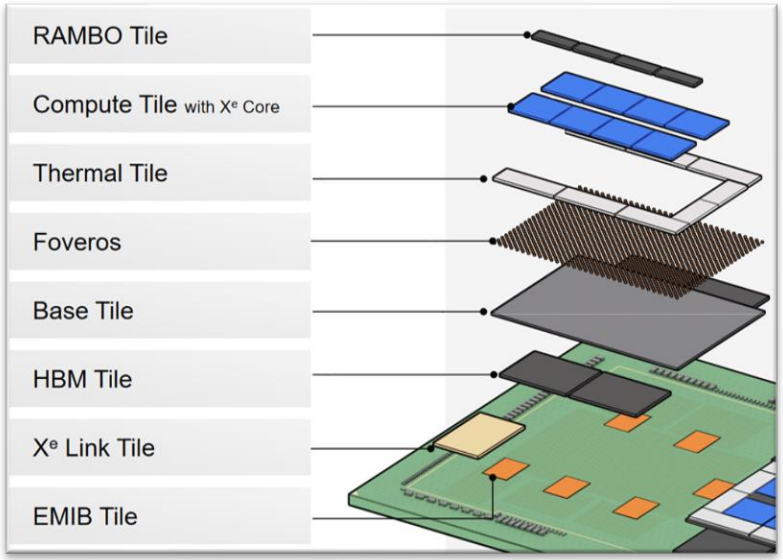
Device density and die size approach physical limits

Advanced Packaging/Interface Technologies

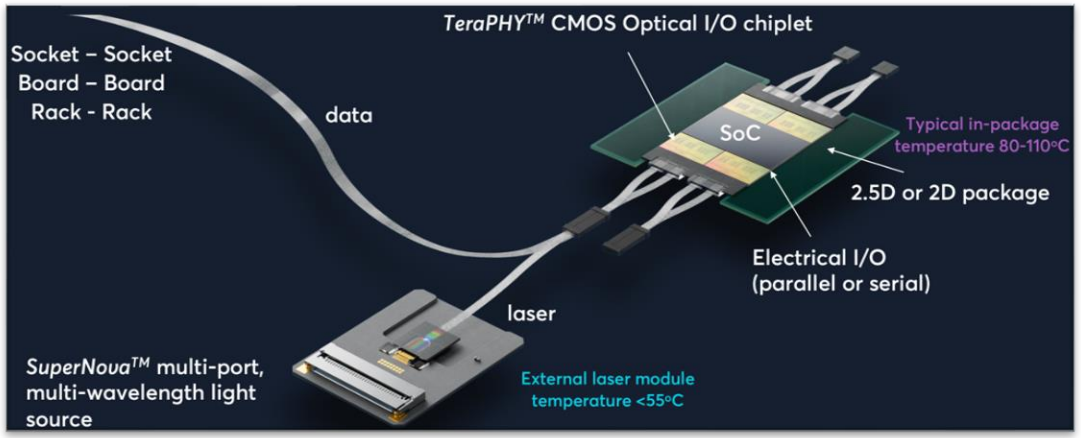
- 2.5D/3D Packaging



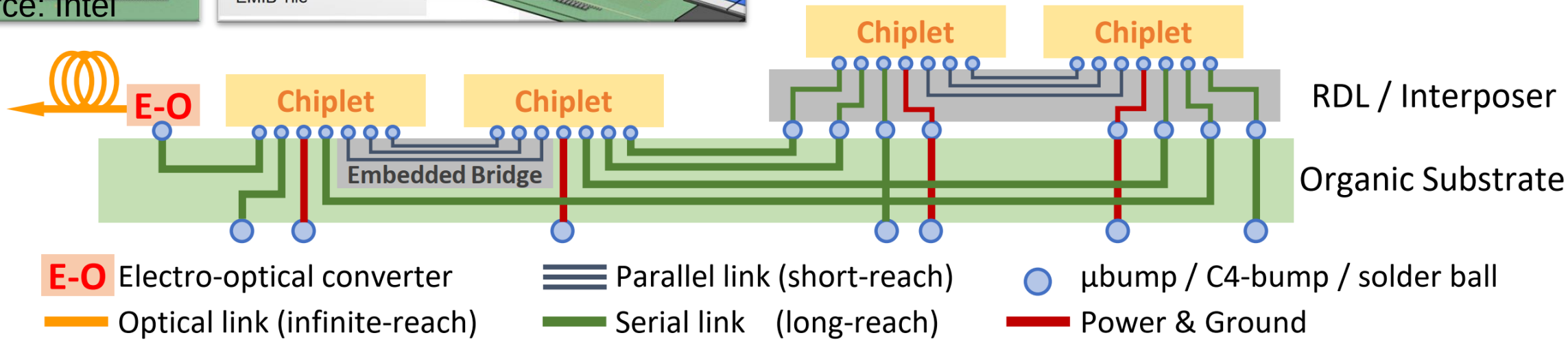
Source: Intel



- Co-packaged Optics

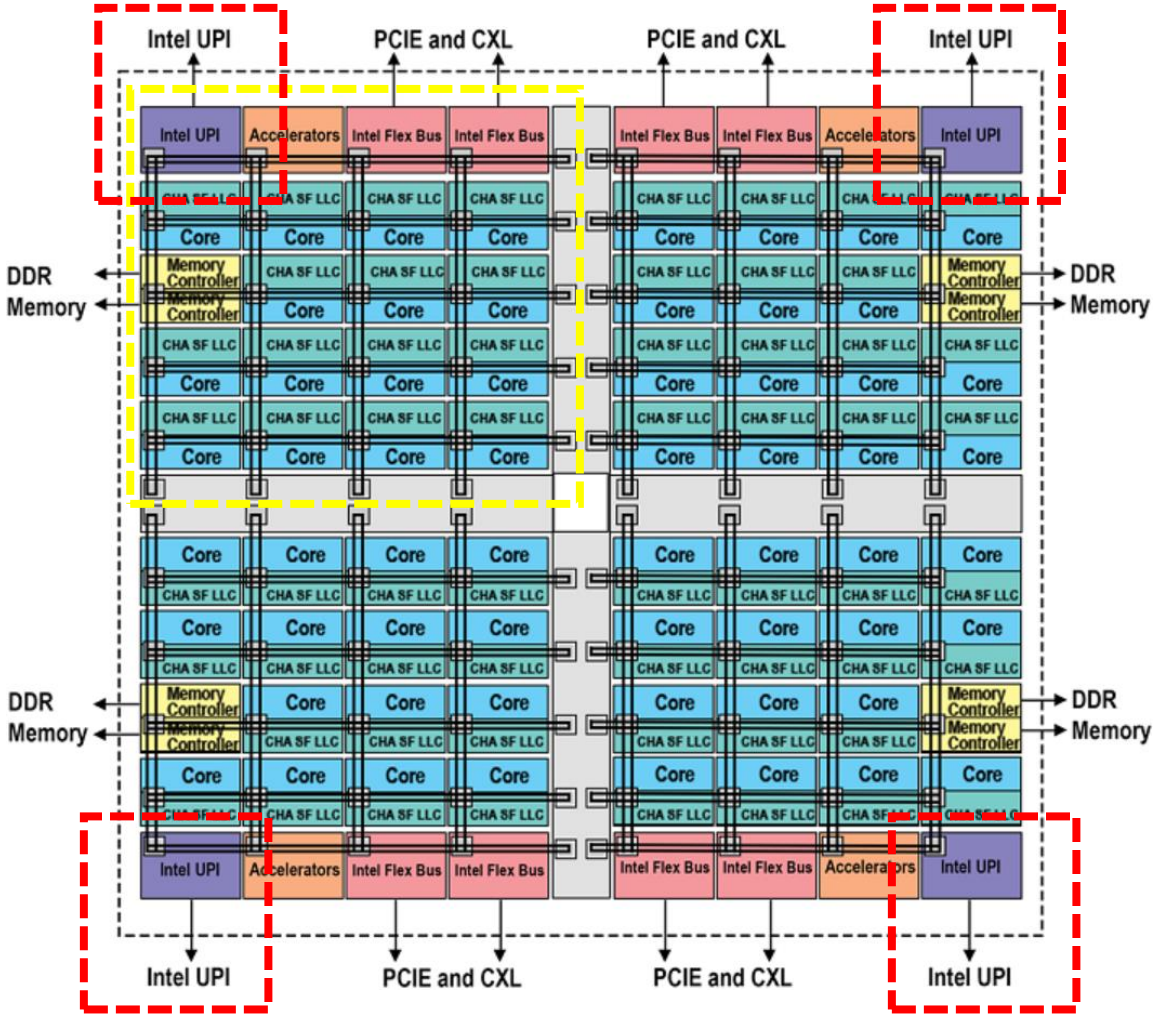


Source: Ayar Labs

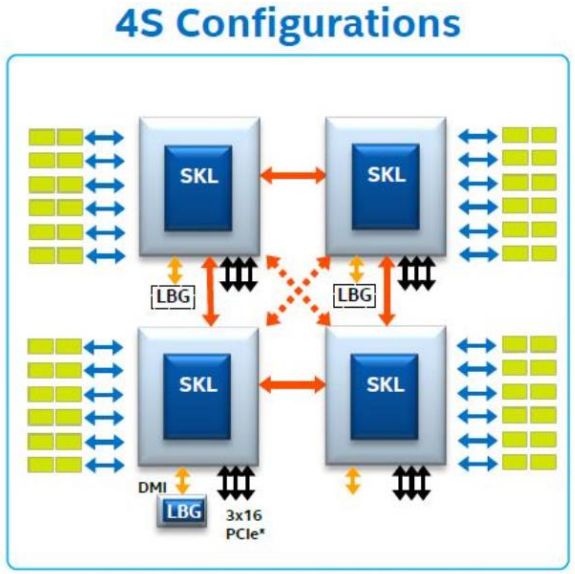


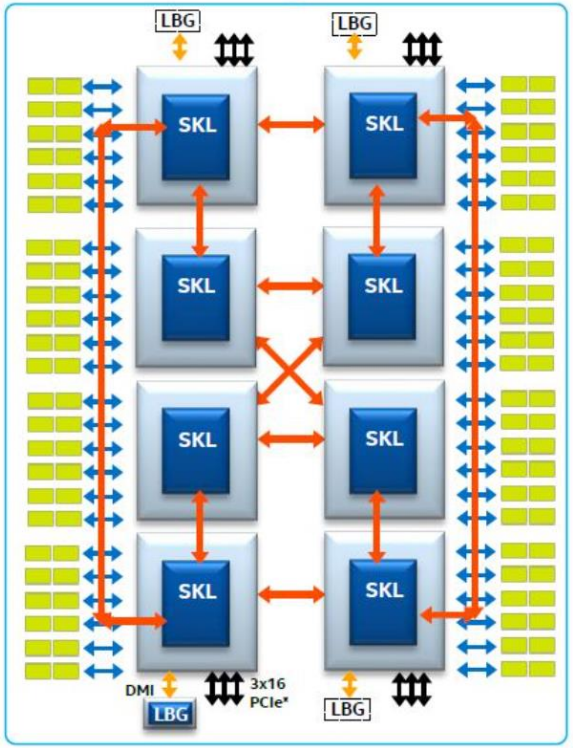
Chiplet-based NoC-Scale-Out Networks

- Intel Sapphire Rapids



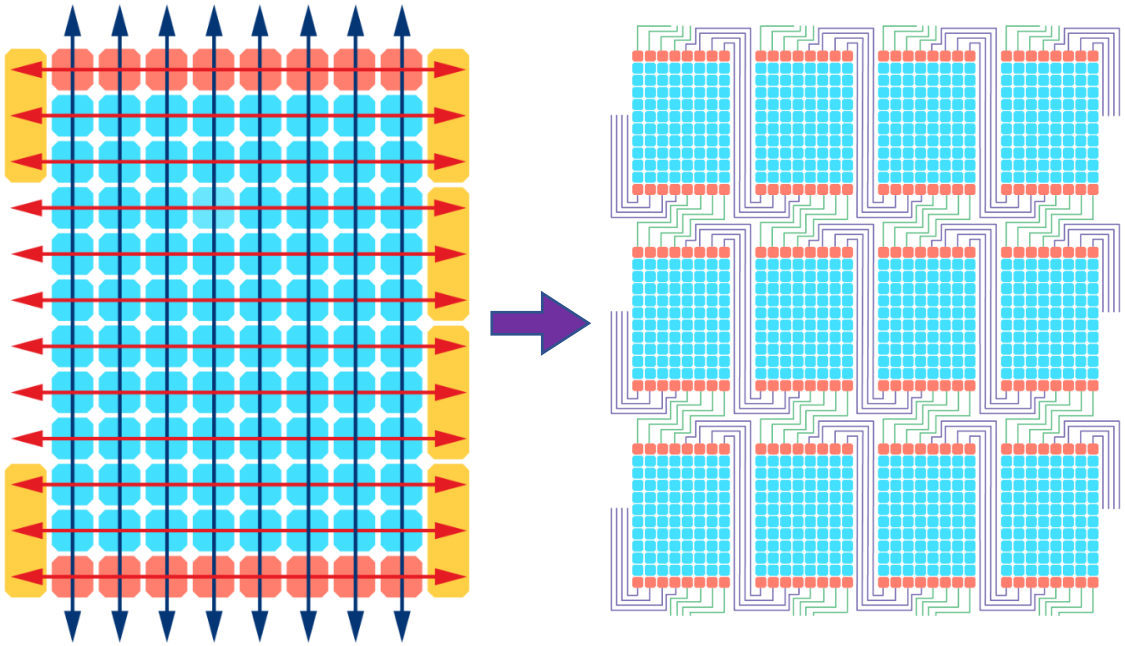
- Intel UPI network





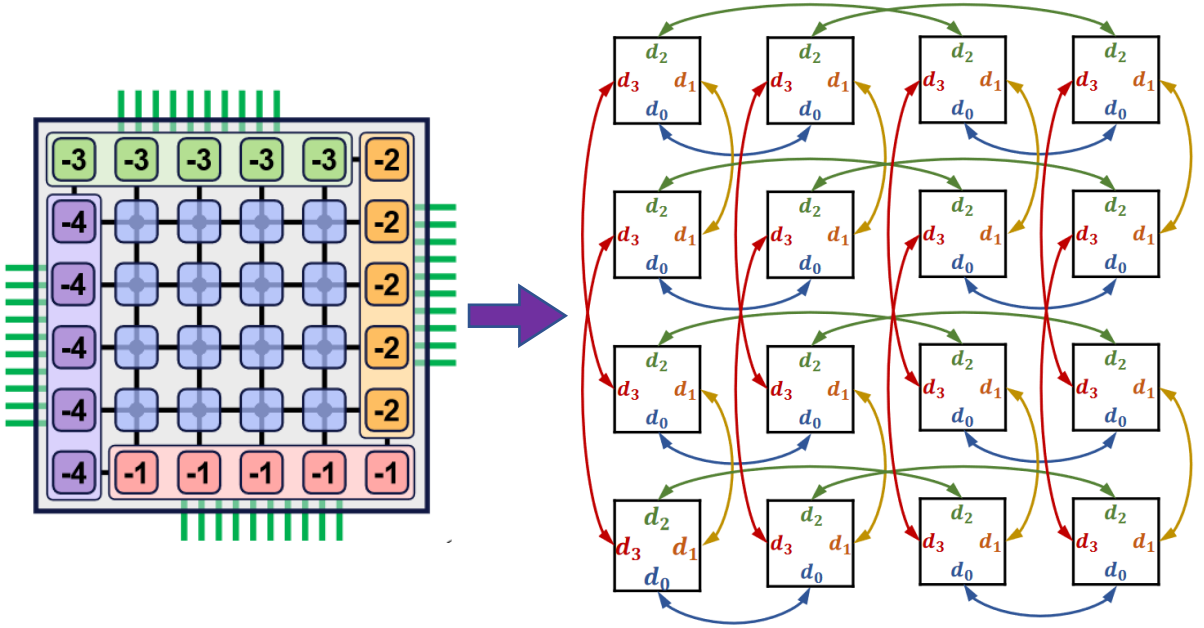
Chiplet-based NoC-Scale-Out Networks

- Tenstorrent training processor
  - Folded 2D-torus on chip
  - 2D-mesh of chip

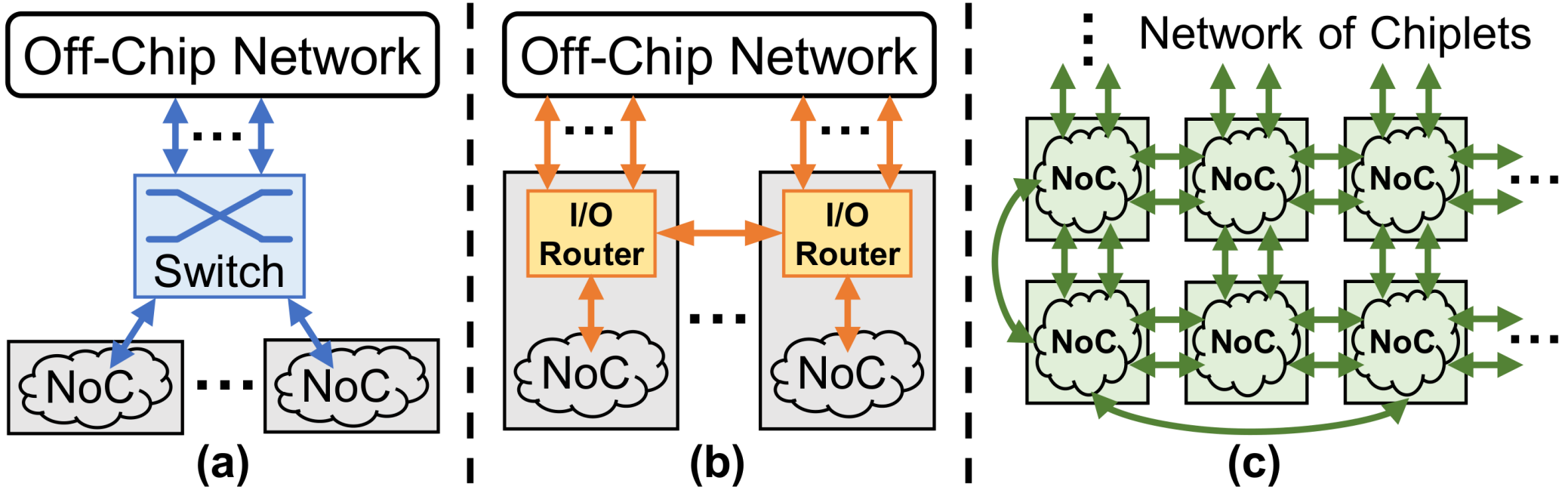


ISSCC2022, The Wormhole AI Training Processor

- Scalable chiplet-based networks
  - 2D-mesh on chip
  - Hypercube of chip



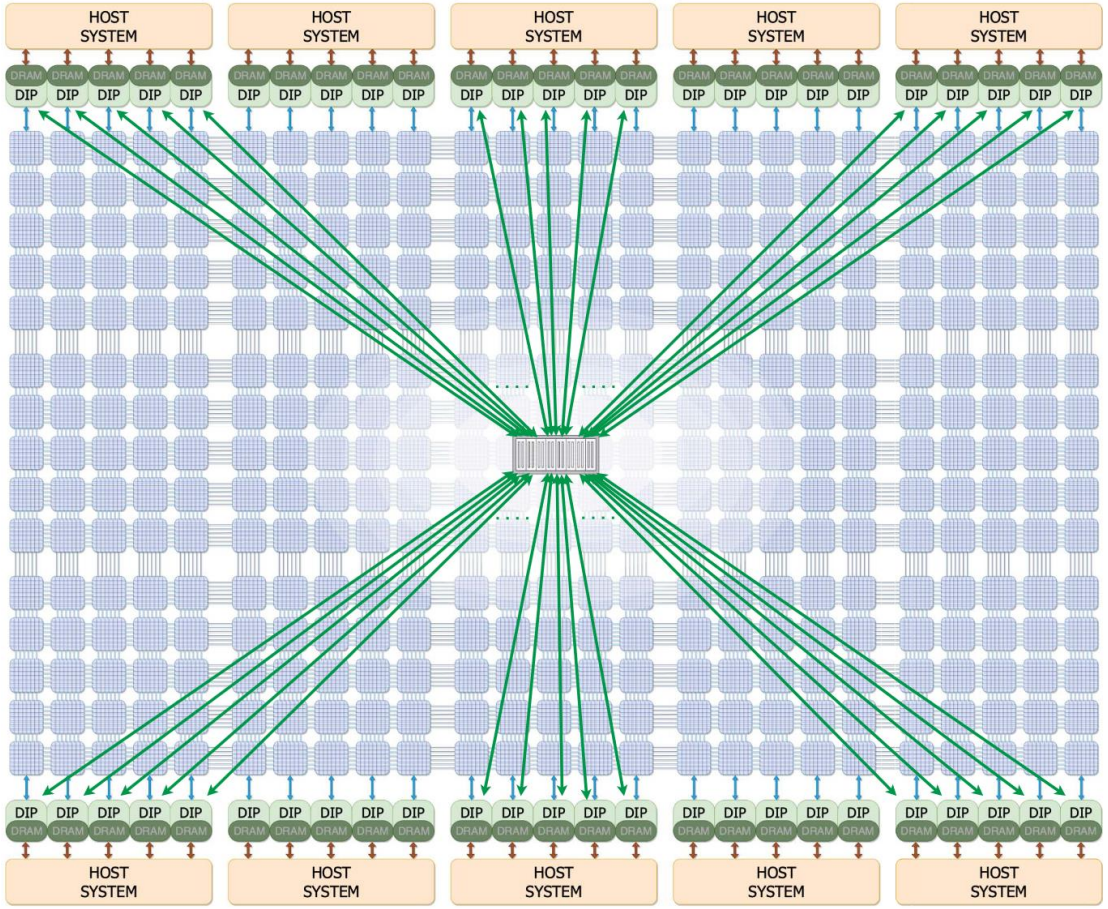
What is the difference?



- Traditionally, on-chip network and off-chip network are decoupled by switches, NICs, or I/O routers, so, they can be designed and evaluated **separately**
- As for network-of-chiplet, on-chip network and off-chip network are tightly coupled, the entire network must be designed and evaluated **jointly** rather than separately

Chiplet-based Large-Scale Networks

- Tesla DOJO system



IEEE Micro, 2023, The Microarchitecture of DOJO

- Up to **3000** D1 dies
- **1 million** on-chip cores

- Core-to-core topology
- Core-to-core routing
- Core-level scheduling
- Core-level fault-tolerance
- Flit-level transmission

Challenges for Evaluation

- On-chip network
  - Core-to-core
  - **Tens** of nodes (cores)
  - Cycle based simulation
- Off-chip network
  - NIC-to-NIC or switch-to-switch
  - **Thousands** of nodes (chips)
  - Discrete event based simulation
  - **Distributed parallel simulation**
- Chiplet-based large-scale network
  - Core-to-core
  - **Millions** of nodes (cores)
  - **Shared-memory**
  - Billions of cycles of simulation time
  - Large design space
  - To evaluate performance
  - To explore microarchitectures
  - To find potential deadlocks
  - To model **fine-grained** behaviors

Why Existing Simulators Are not Sufficient

- Cycle-accurate Simulators: BookSim, Garnet (Gem5), Noxim
  - **Single-thread**
  - Too much running time is consumed by unimportant circuit stage, e.g. the round-robin allocation solution

BookSim Profiling Results

Cycle Estimation (%)	2D-Mesh	Dragonfly
`IQRouter::_internalStep()`	92.18	71.92
Call: `_VCAllocEvaluate()`	48.08	17.65
Call: `SparseAllocator::Clear()`	28.86	17.93
Call: `_SWAllocEvaluate()`	5.71	13.43
Call: `_SWAllocUpdate()`	4.59	11.26

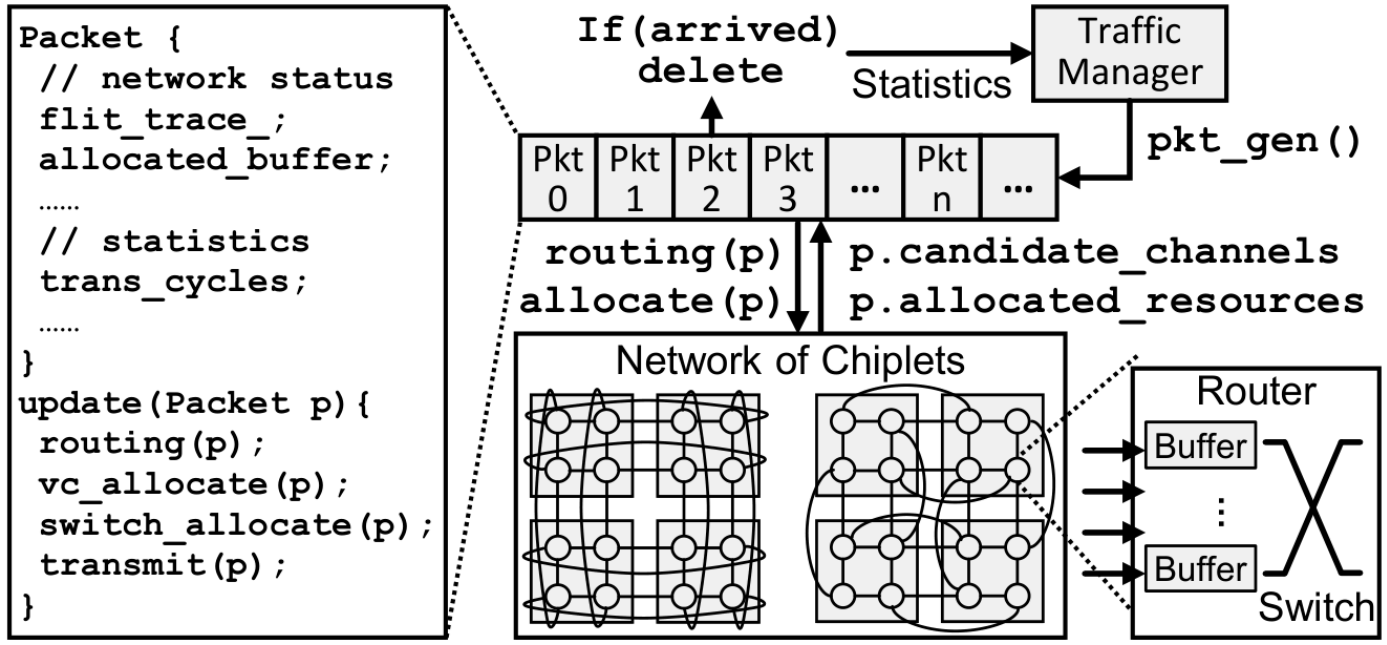
- Discrete event simulators: OMNeT++, NS-3, SST
  - Coase-grained, not cycle-accurate
  - Support parallel simulation, but require manual partitioning and MPI communication protocol
  - **Programming overhead**
  - Not suitable for chiplet-based networks

- **An ideal simulator is suppose to be:**
  - **Cycle-accurate**
  - **Very efficient**
  - **Not requiring extra programming**

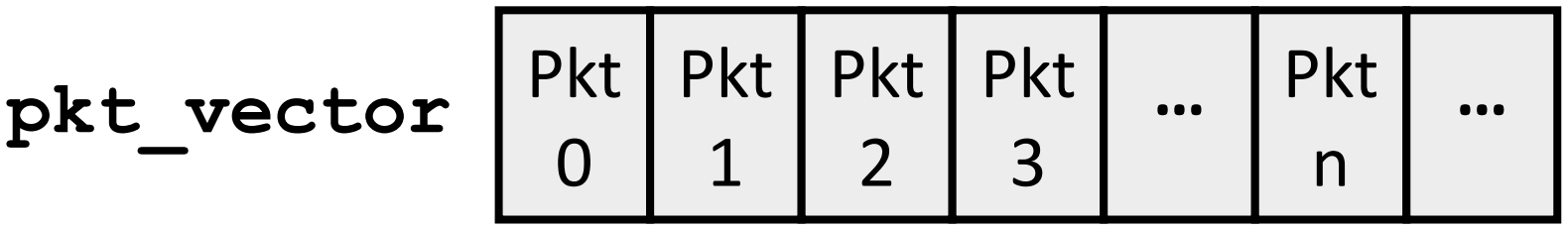
Chiplet Network Simulator (CNSim) Architecture

- Network-centric: the key status values (e.g., packet occupation) are stored in **`Network`** instance
  - Node 0:
    - Buffer i: [pkt x, pkt y]
    - Buffer j: [ ]
    - ……
  - Node 1:
    - Buffer k: [pkt z]
- …..

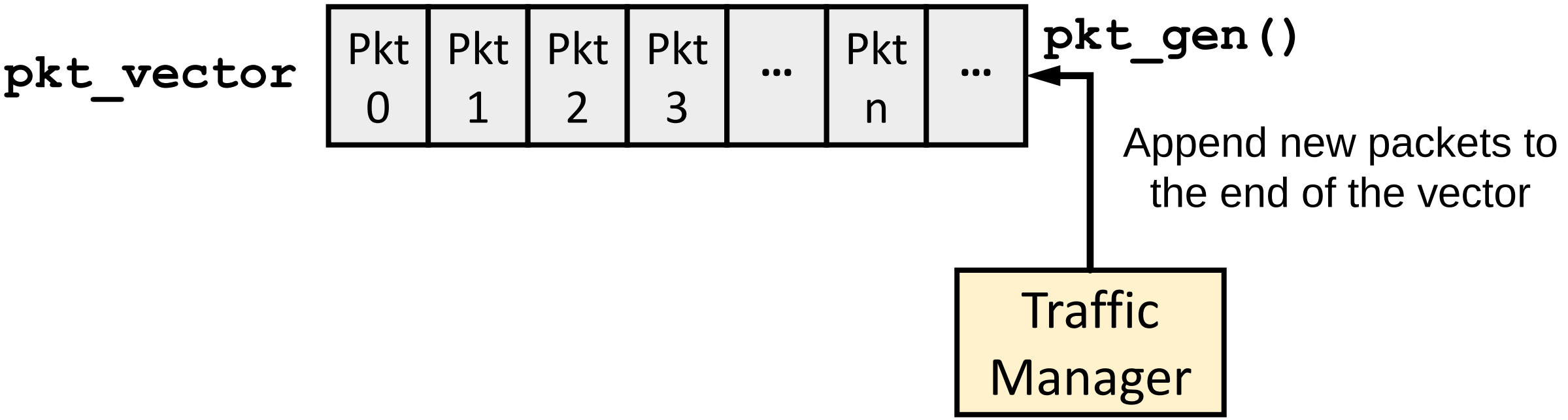
- Packet-centric: the key status values are stored in **`Packet`** instance
  - Packet x: Node 0, Buffer i, 0
  - Packet y: Node 0, Buffer i, 1
  - Packet z: Node 1, Buffer k, 0



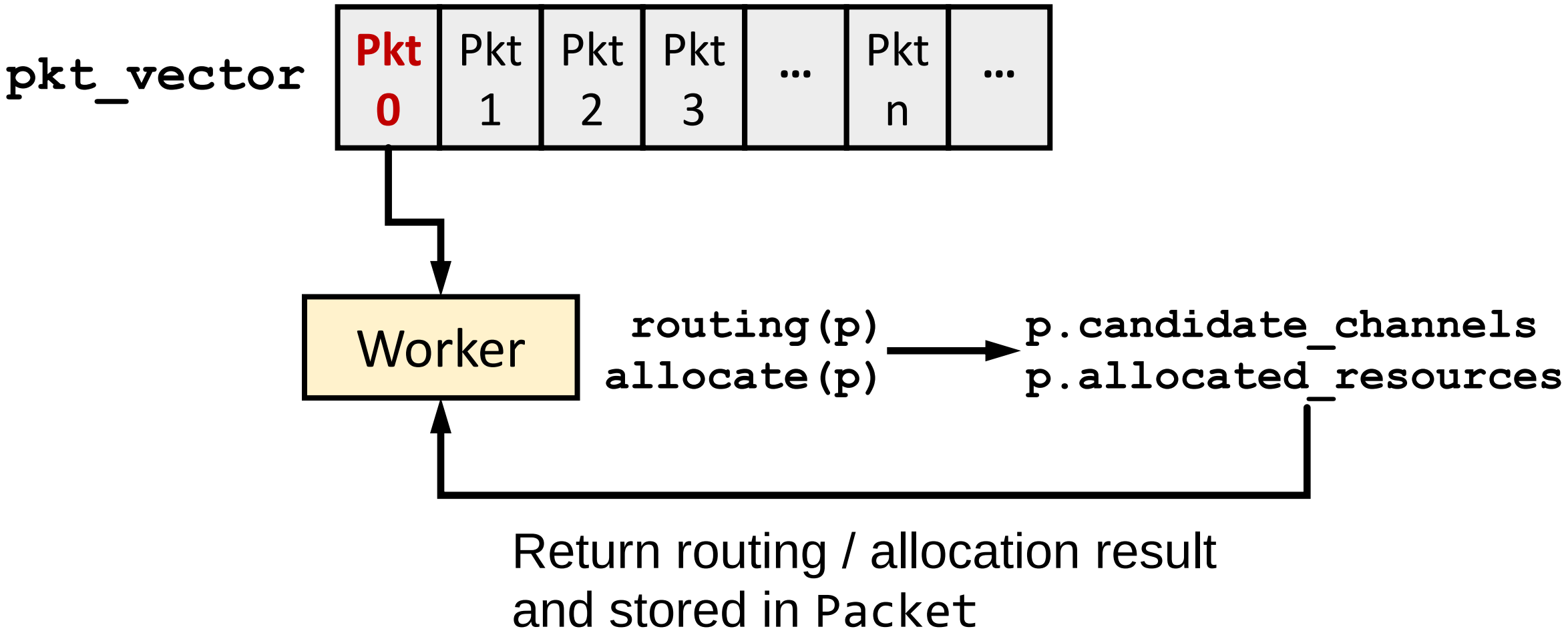
Packet Queue



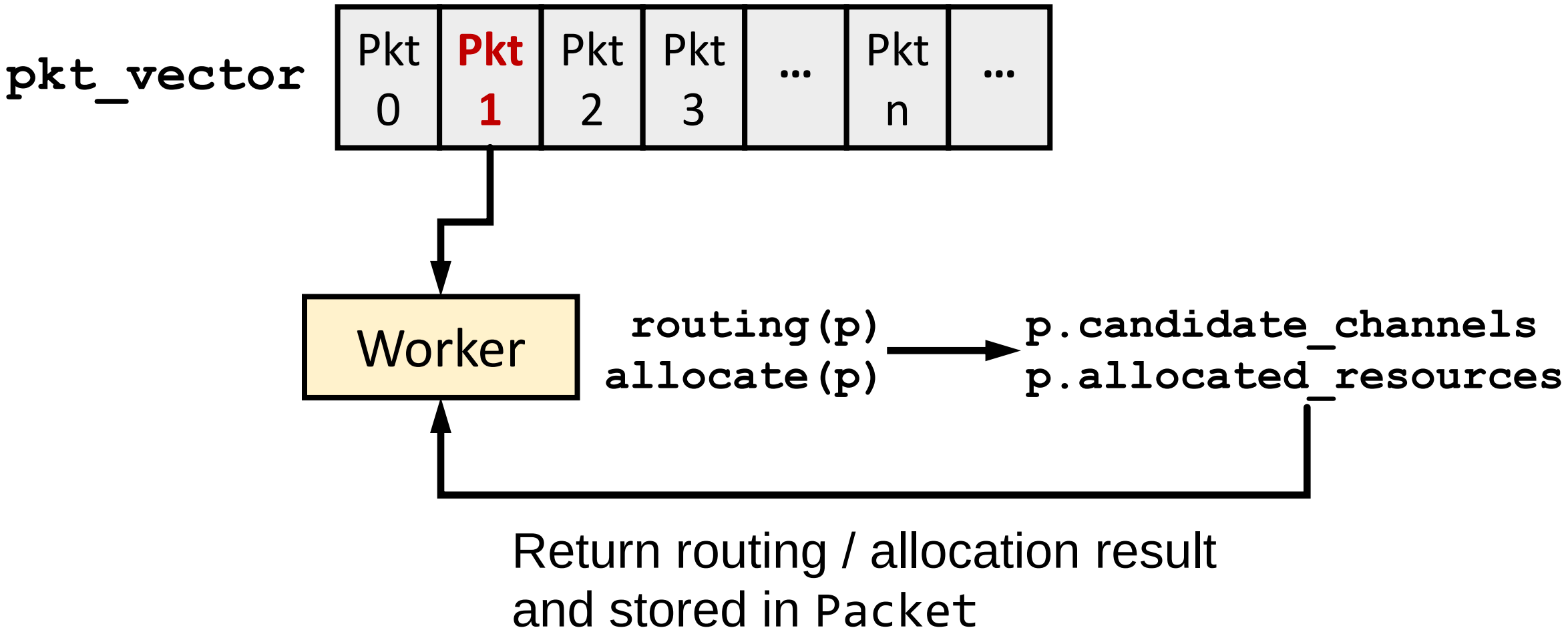
Packet Queue



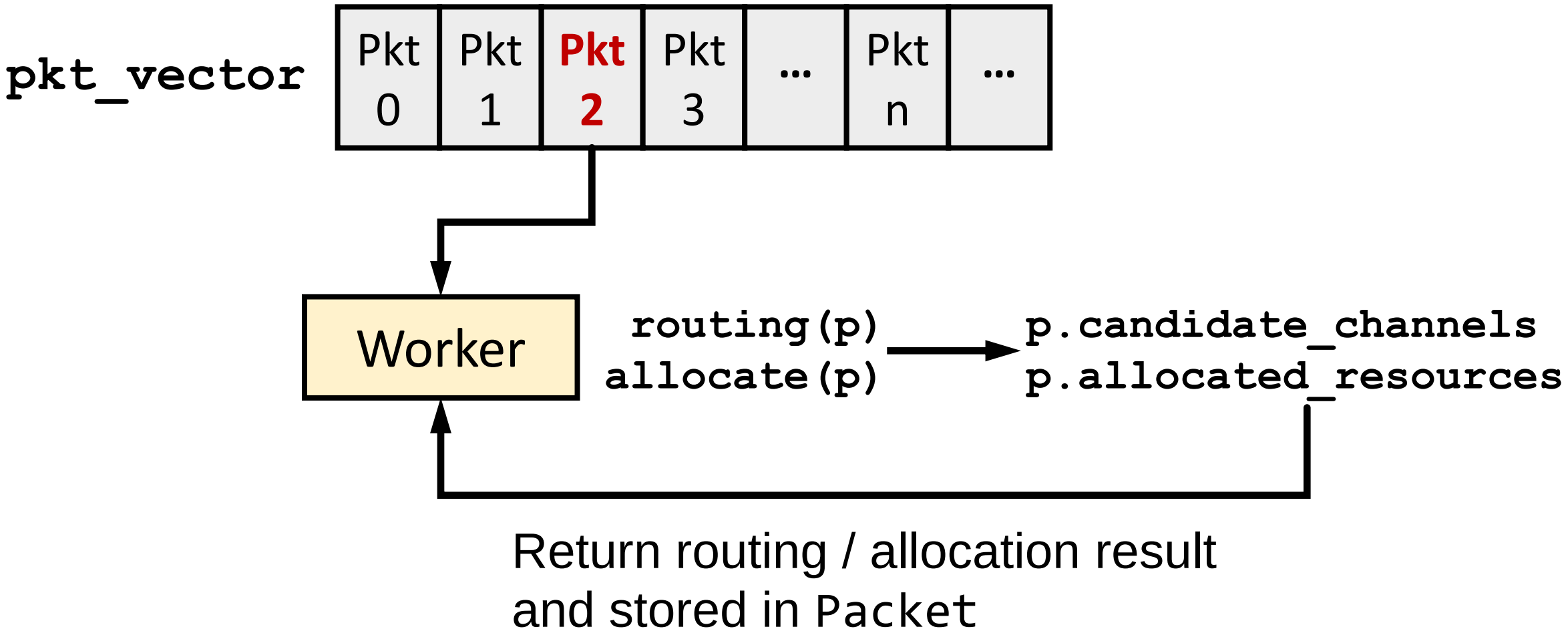
Worker


pkt_vector
Pkt 0
Pkt 1
Pkt 2
Pkt 3
…
Pkt n
…
Worker
routing(p)
allocate(p)
p.candidate_channels
p.allocated_resources
Return routing / allocation result
and stored in Packet

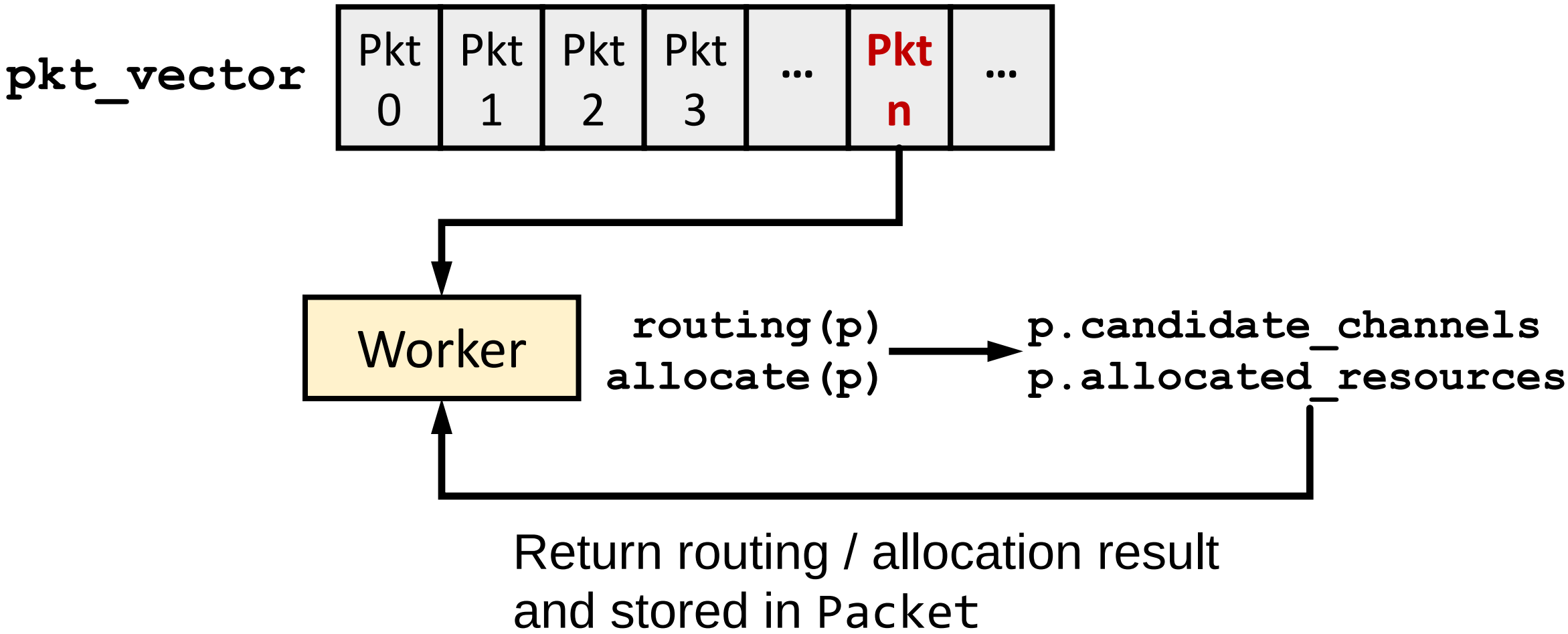
First Inject First Serve



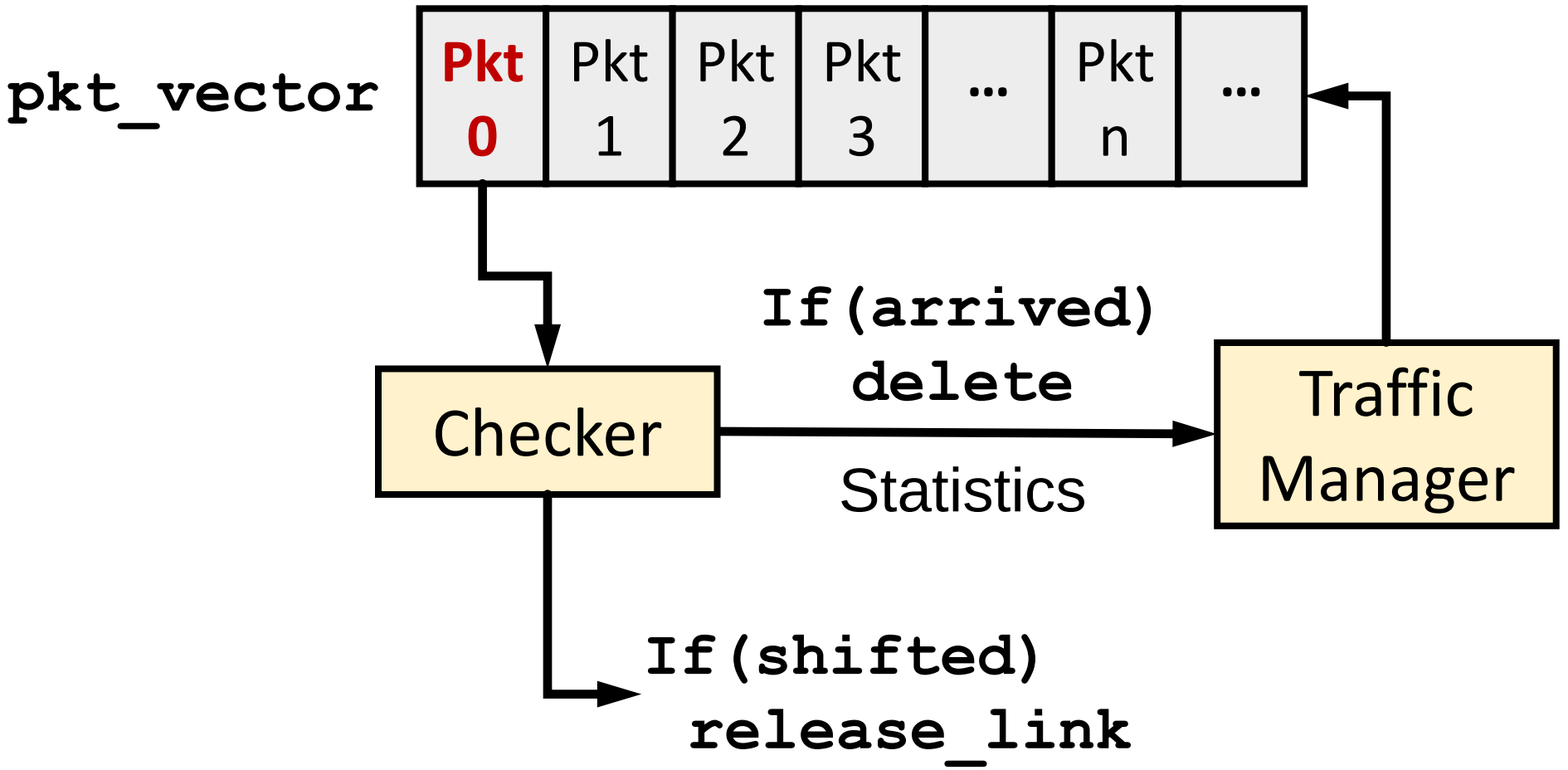
First Inject First Serve



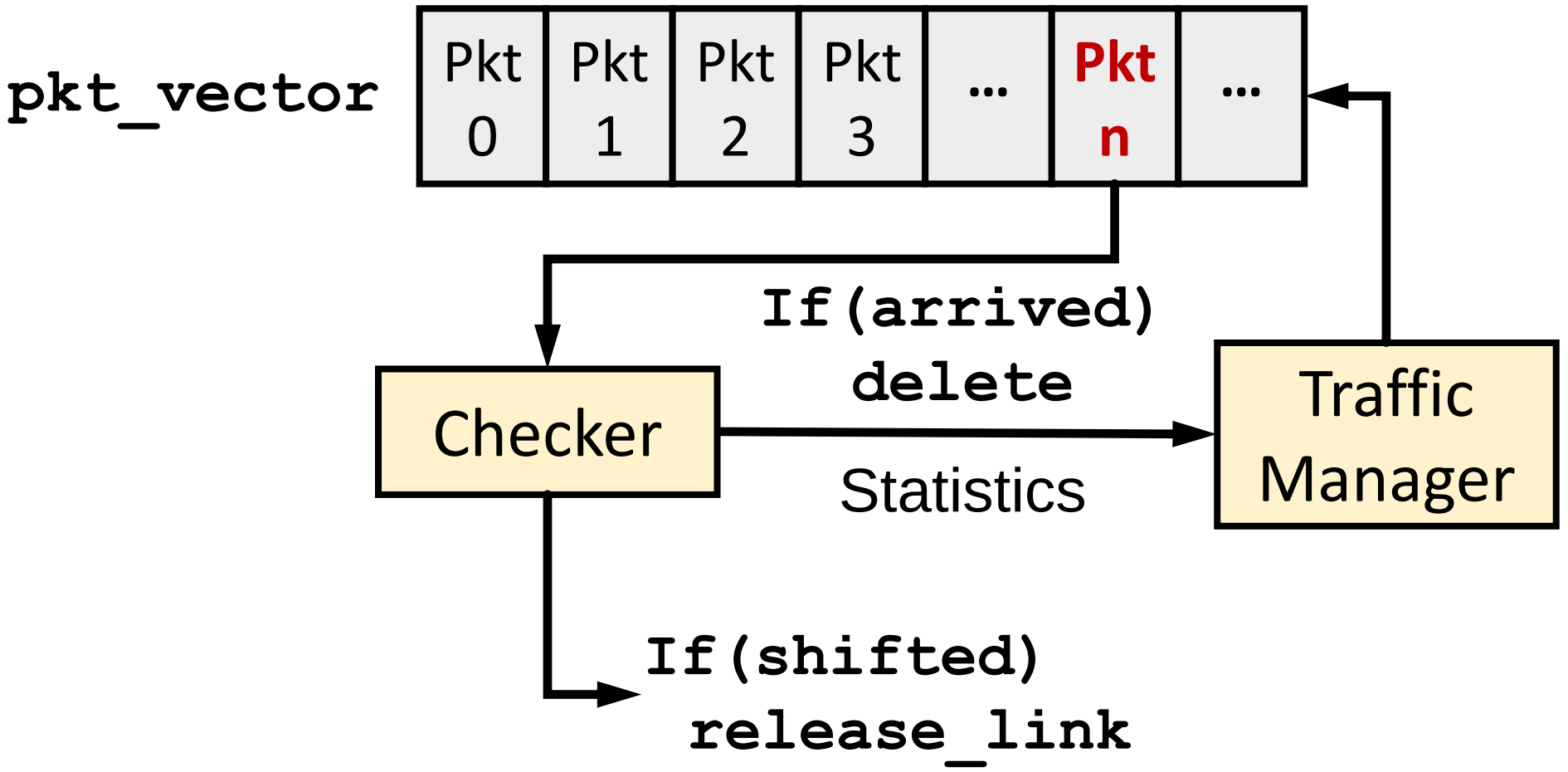
First Inject First Serve



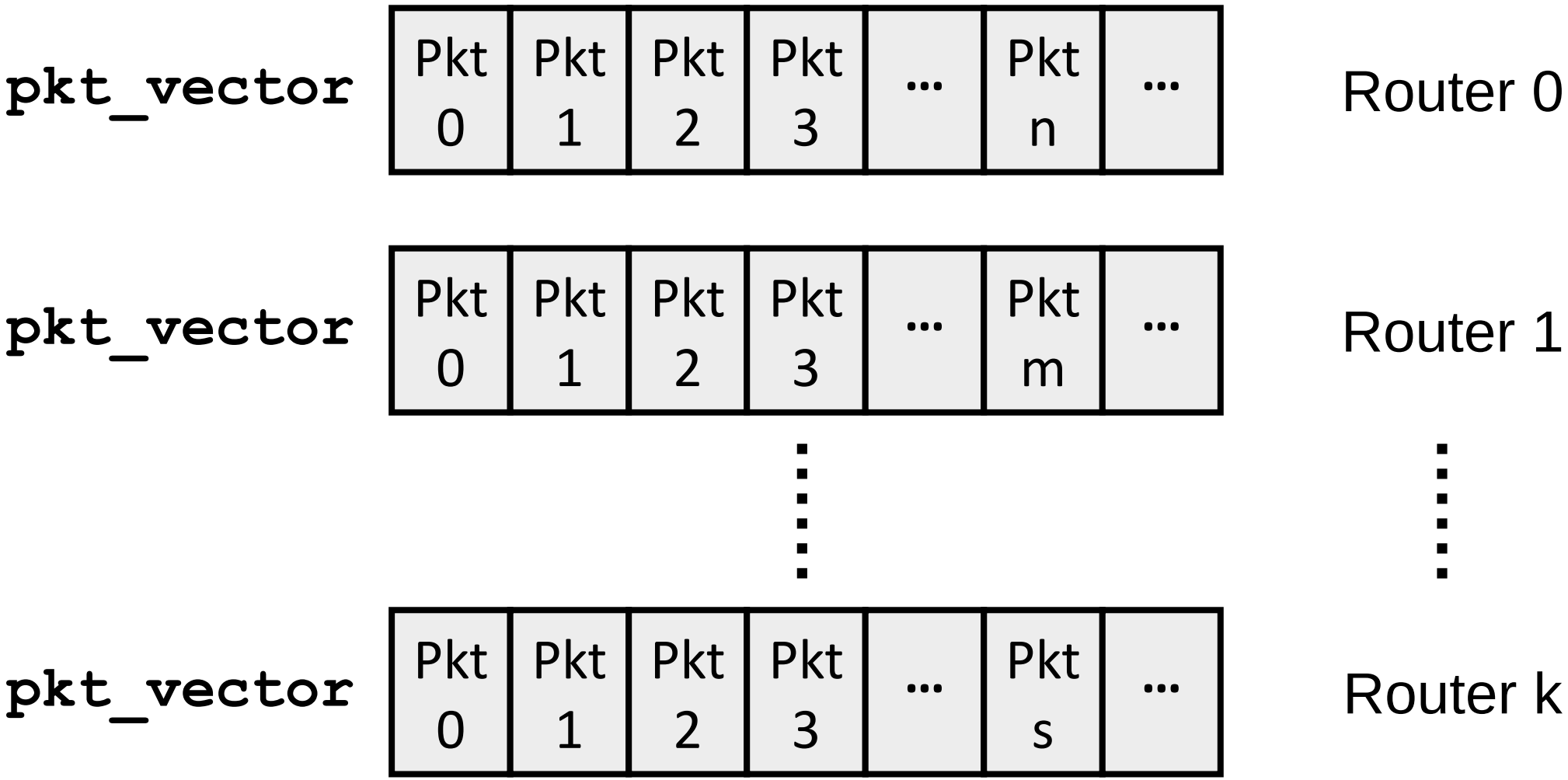
Resource Releasing



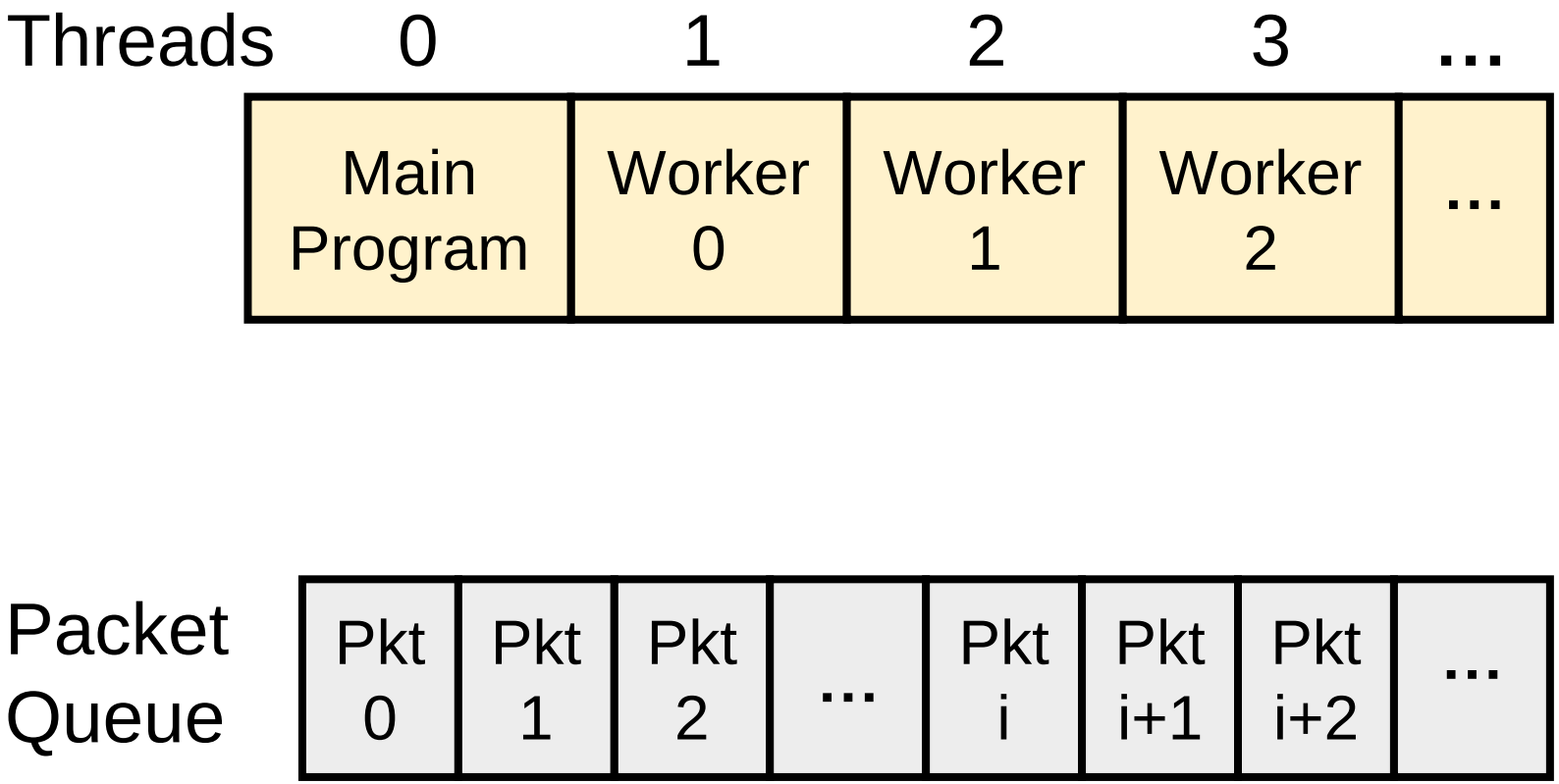
Resource Releasing



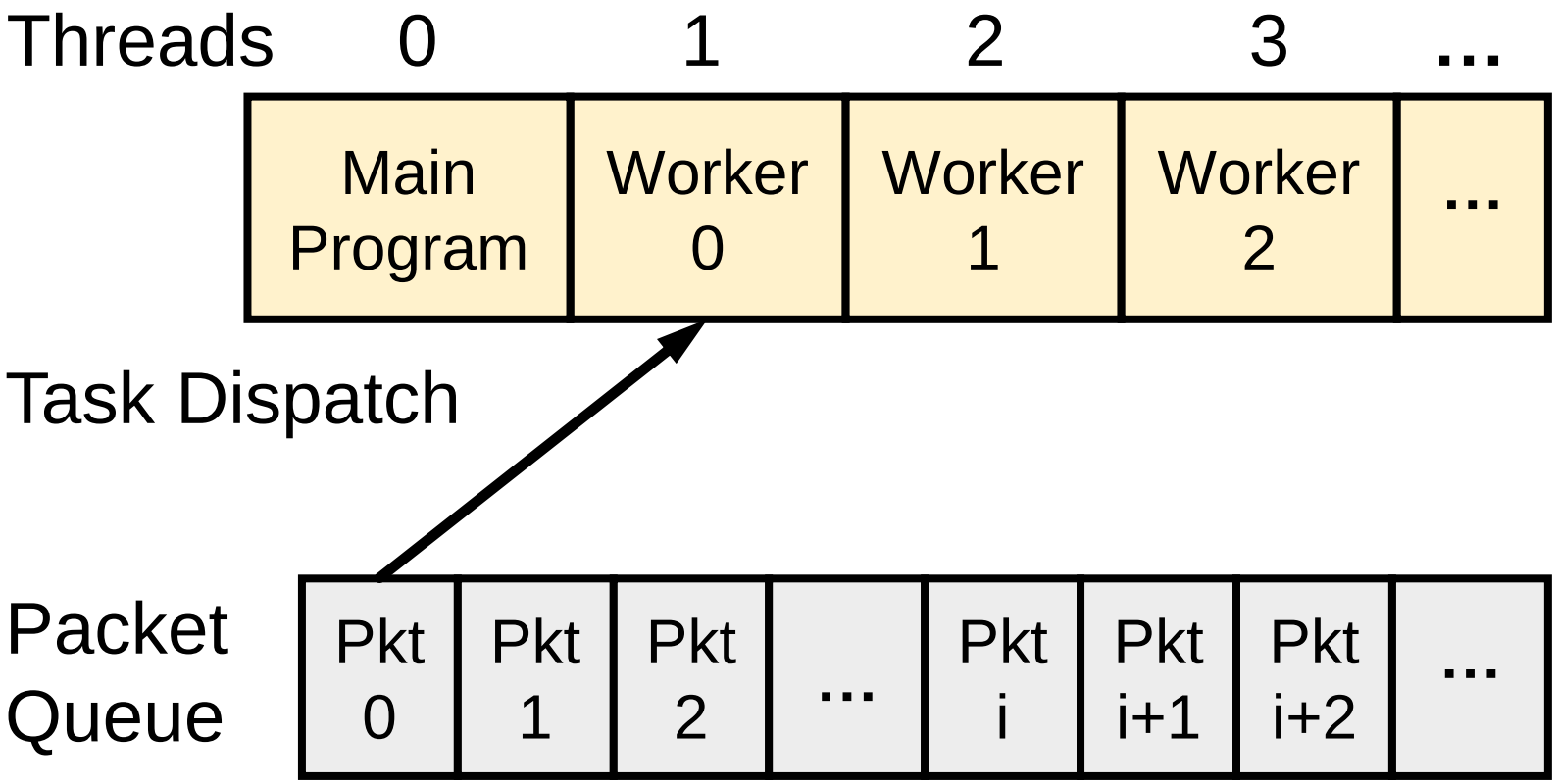
Network Parallelism / First **Come** First Serve



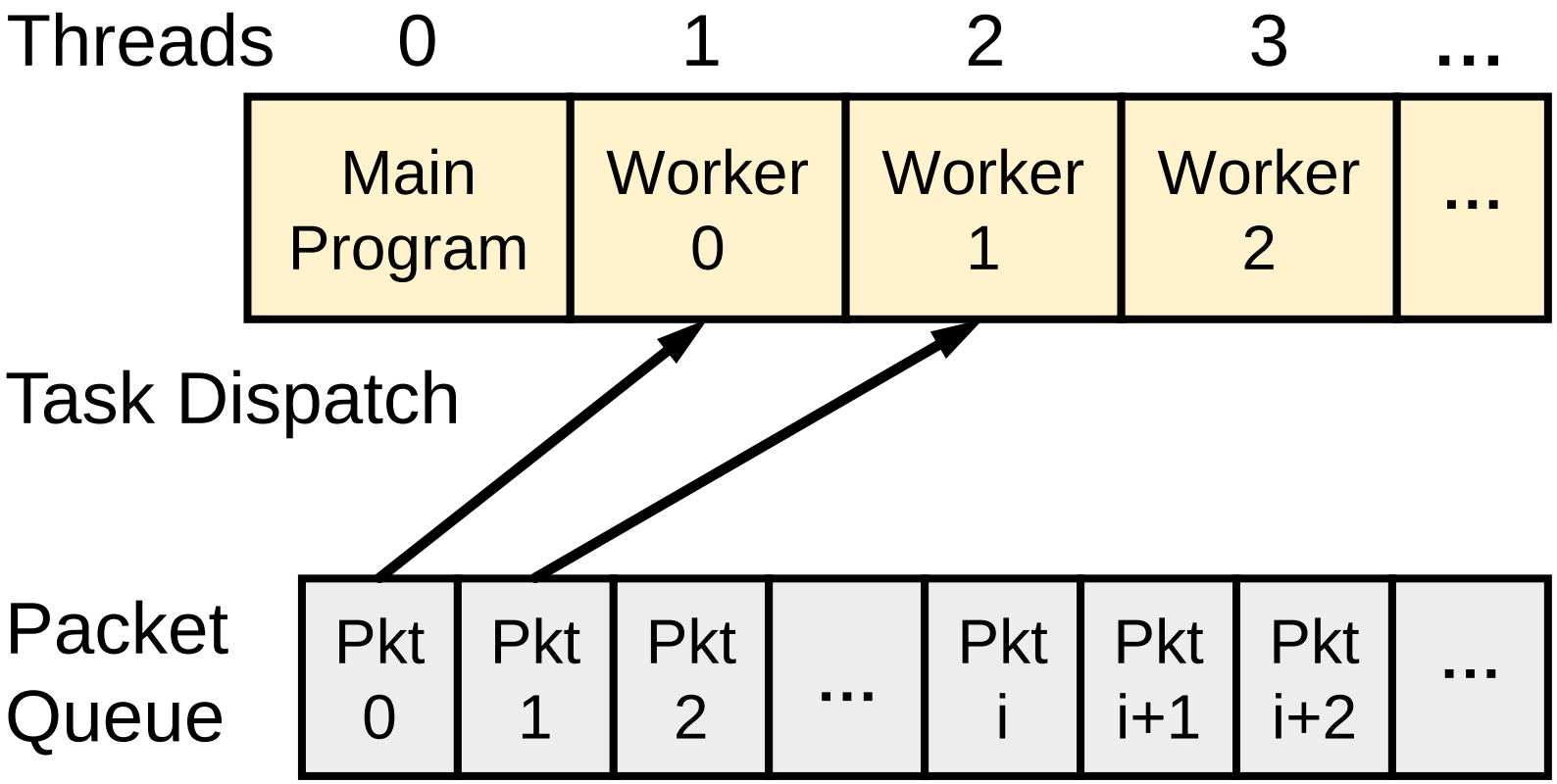
Packet-Parallel Simulation



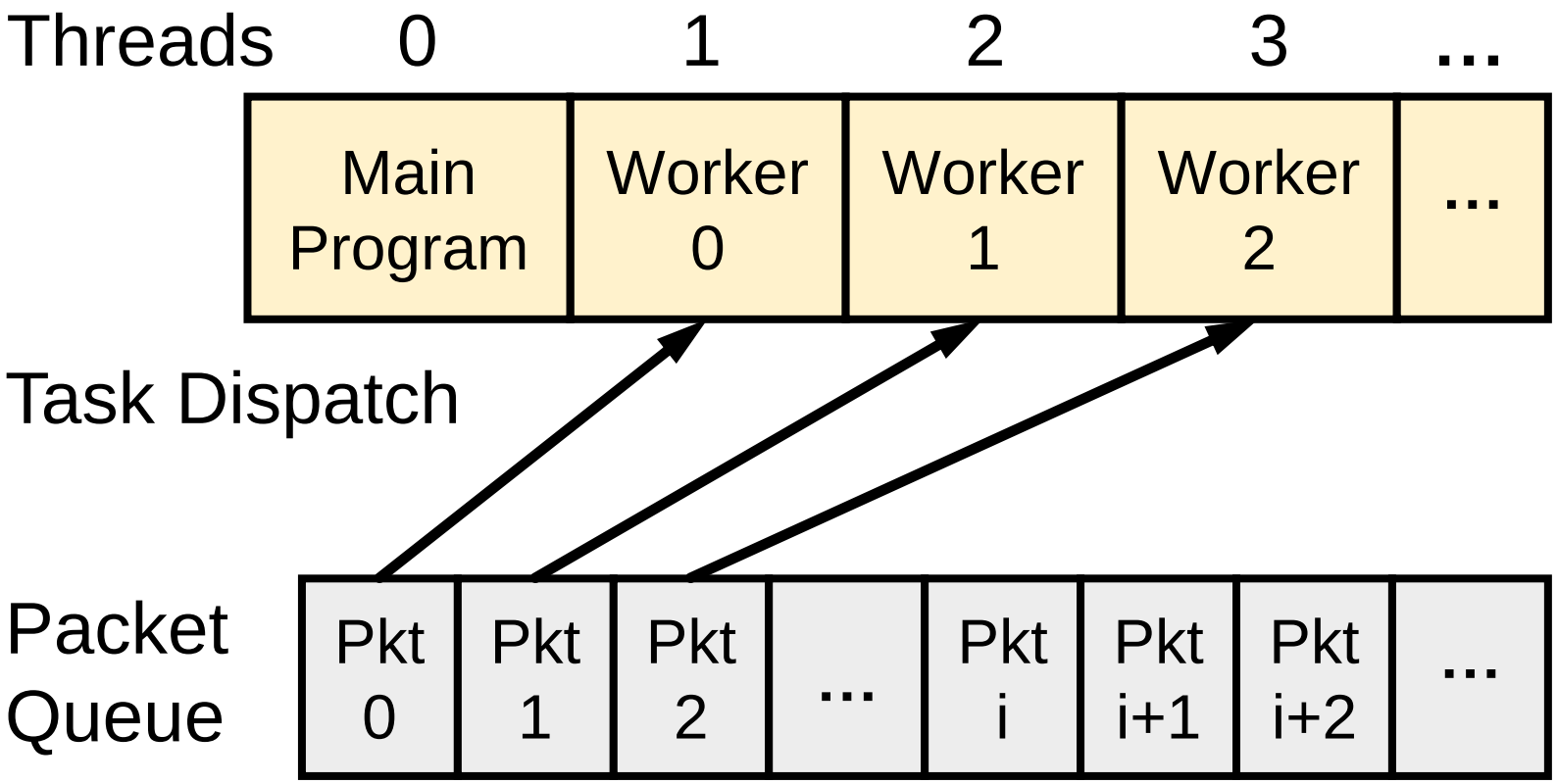
Packet Parallelism Simulation



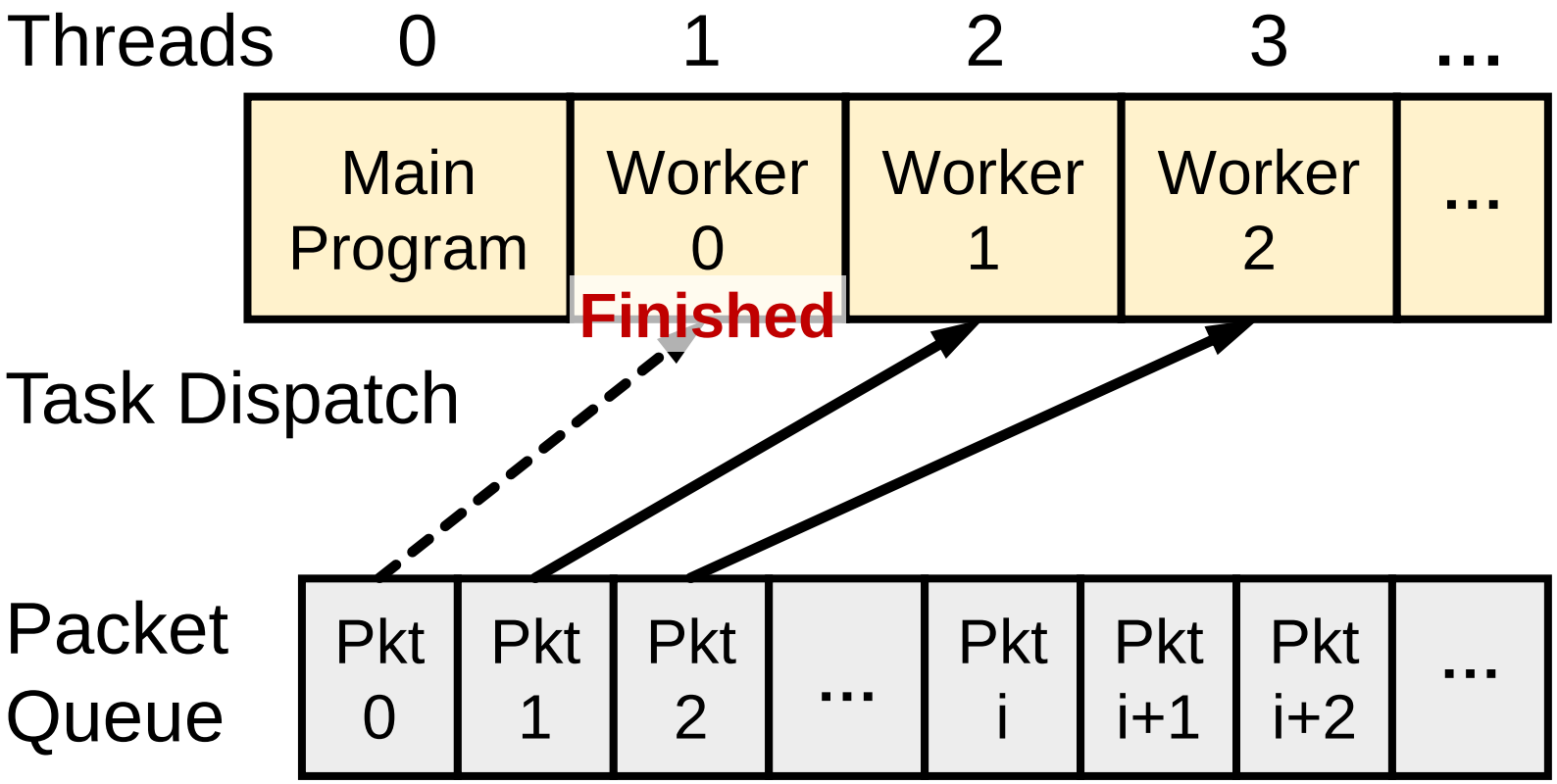
Packet Parallelism Simulation



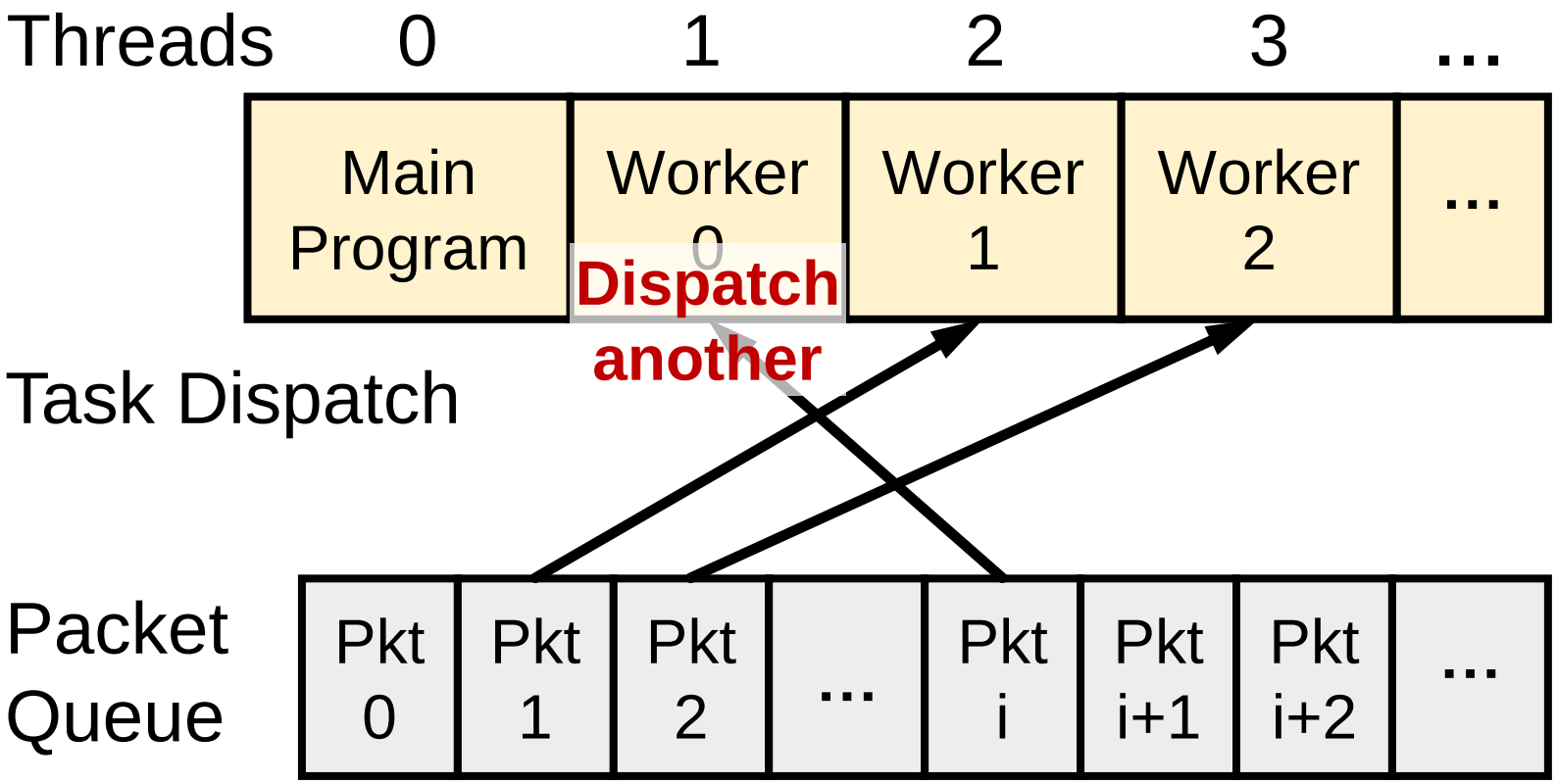
Packet Parallelism Simulation



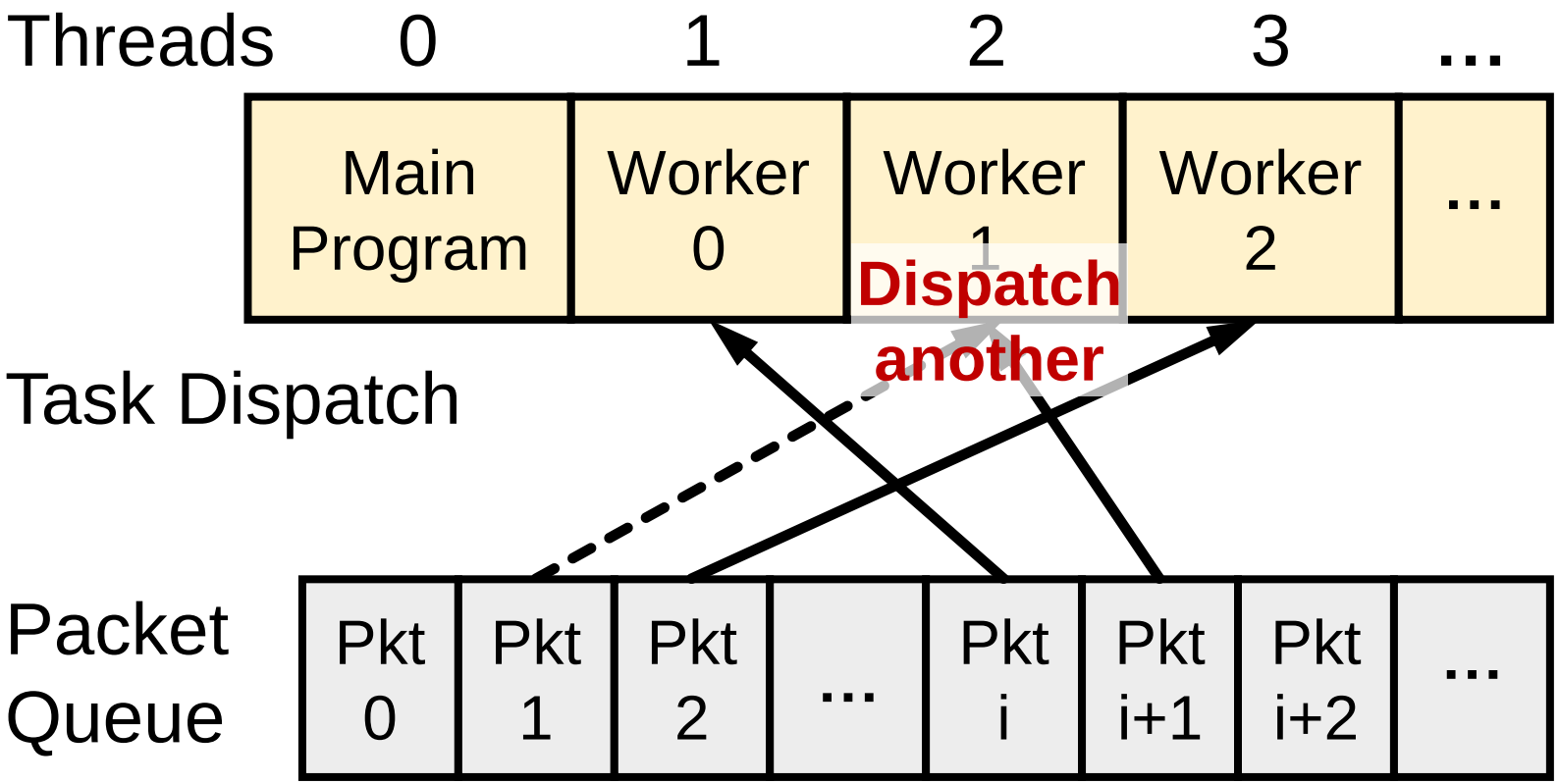
Packet Parallelism Simulation



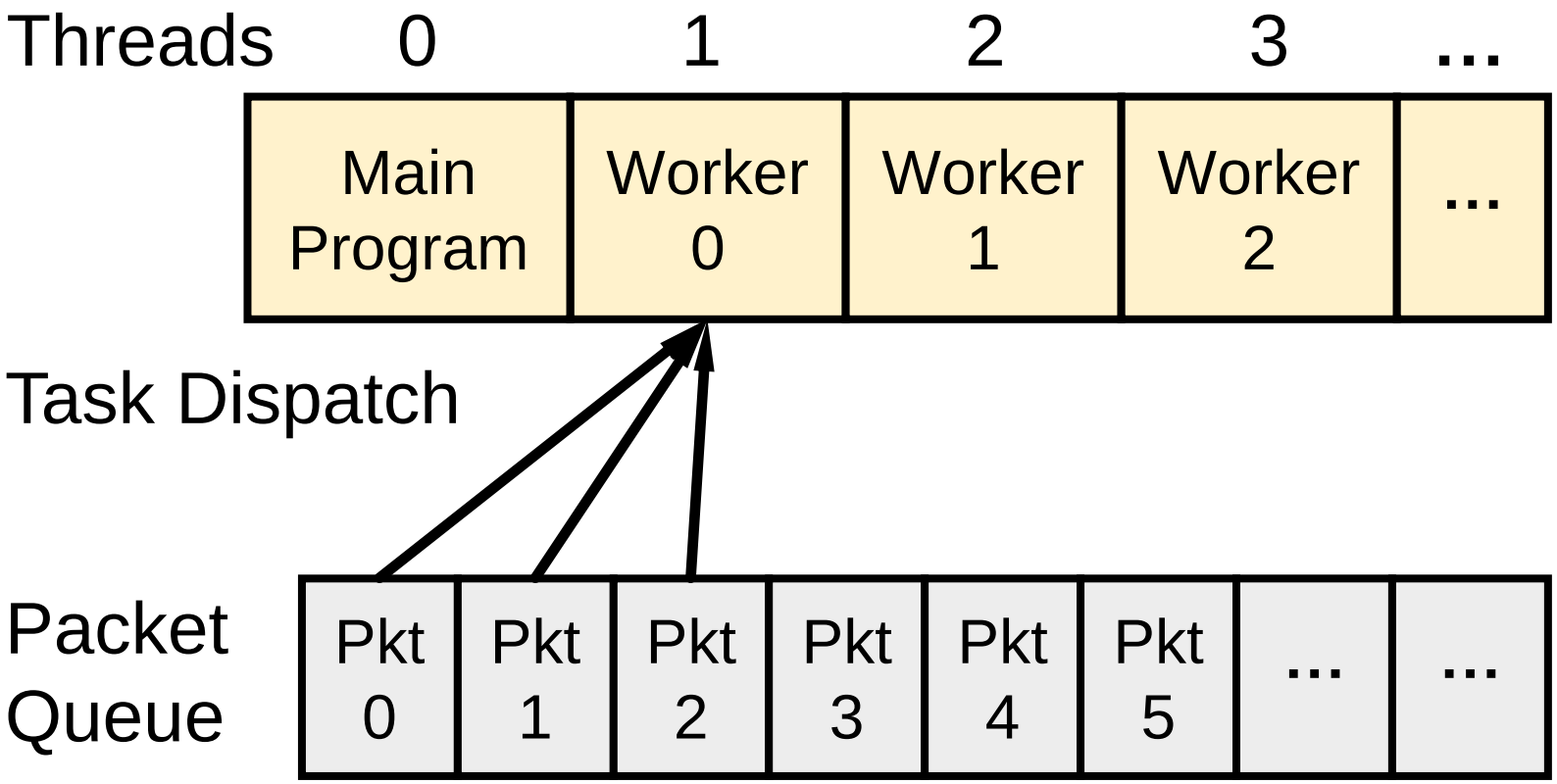
Packet Parallelism Simulation



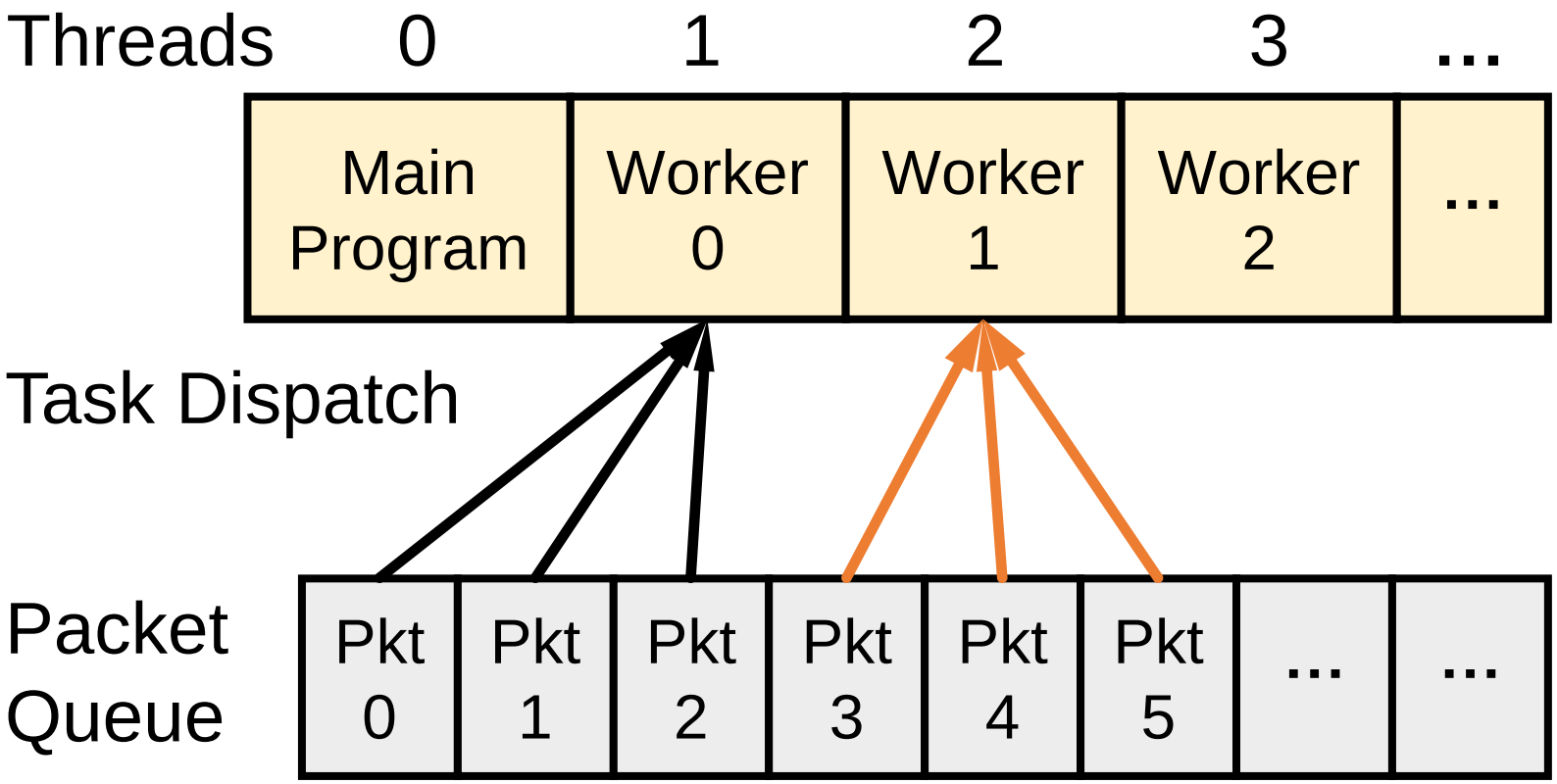
Packet Parallelism Simulation



Packet Parallelism Simulation (Multi Issue)



Packet Parallelism Simulation (Multi Issue)



Atomic-based Programming

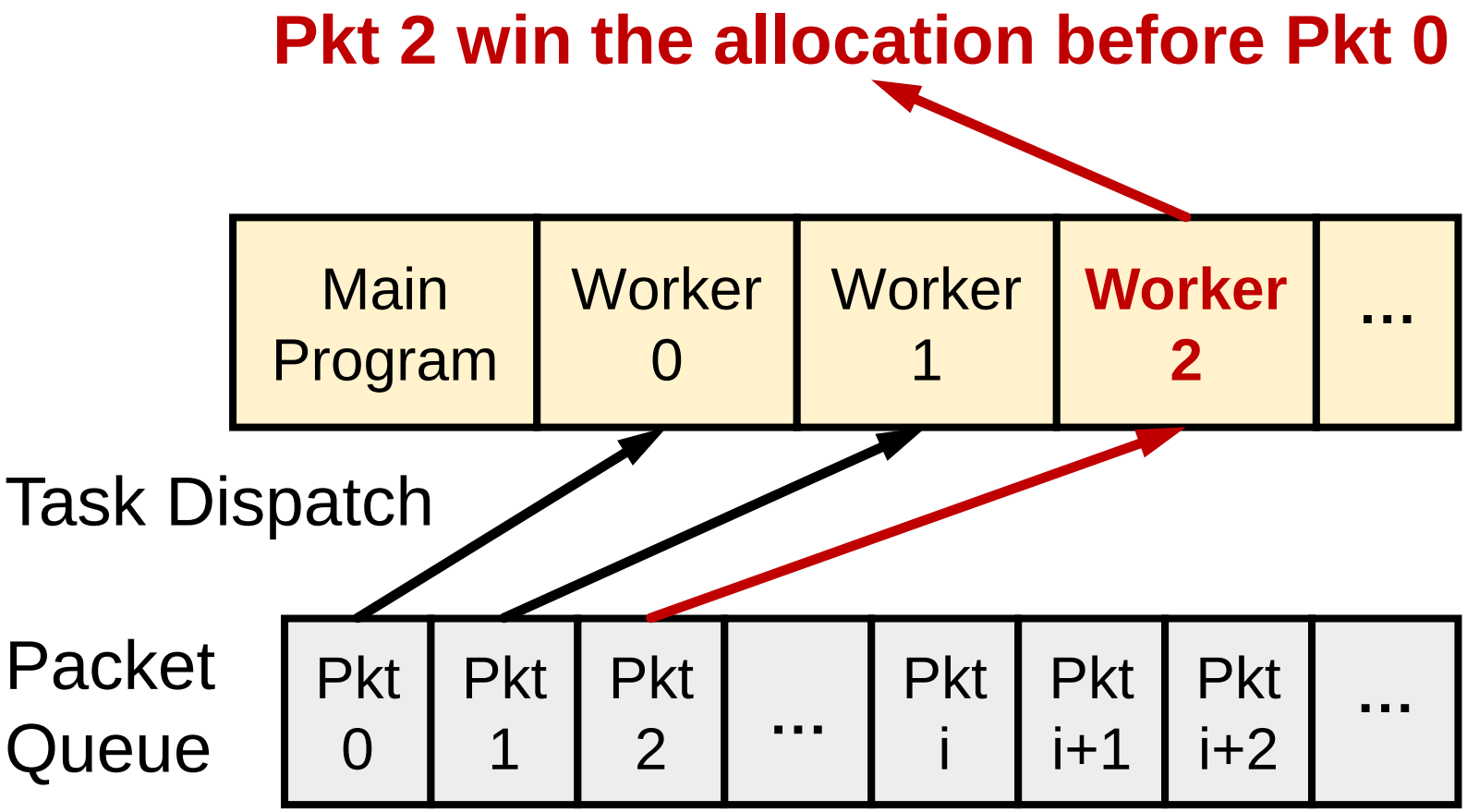
```
// current state of the physical link
std::atomic_bool in_link_used_;
std::atomic_bool sw_link_used_;

// current free buffer of each VC
std::atomic_int* vc_buffer_;

// record the packet order (and the head
packet) in each VC
std::queue<Packet*>* vc_queue_;
std::atomic<Packet*>* vc_head_packet;
```

```
bool Buffer::allocate_in_link(Packet& p) {
  int vcb = p.next_vc_.vcb;
  bool link_used_state = in_link_used_.load();
  if (link_used_state)
    return false;
  else if (in_link_used_.compare_exchange
              (link_used_state, true)) {
    // allocation succeeded
    if (node_->id_ != p.destination_) {
      push_pkt(&p, vcb);
    }
    return true;
  } else // allocation failed
      return false;
}
```

Multi-Thread Inconsistency



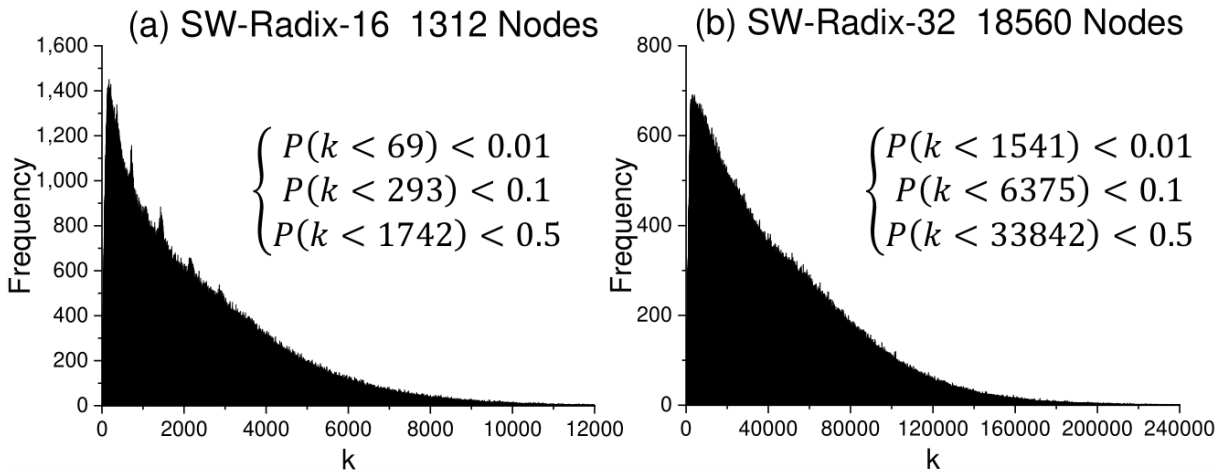
Is Multi-Thread Inconsistency Fatal?

- Allocation policy is not we are really interested in. Different policies have advantages and disadvantages, but they do not seriously affect the overall performance

- All packets are dispatched to workers (threads) in injection order. Therefore, early packets are still more likely to win the contention

$$P_{\text{inverted allocation}} \leq \sum_i P(k=i) e^{-\lfloor \frac{i}{c} \rfloor \frac{1}{M}}$$

- k: packet distance (index difference)
- c: multi-issue width
- M: worker (thread) number



Other Features

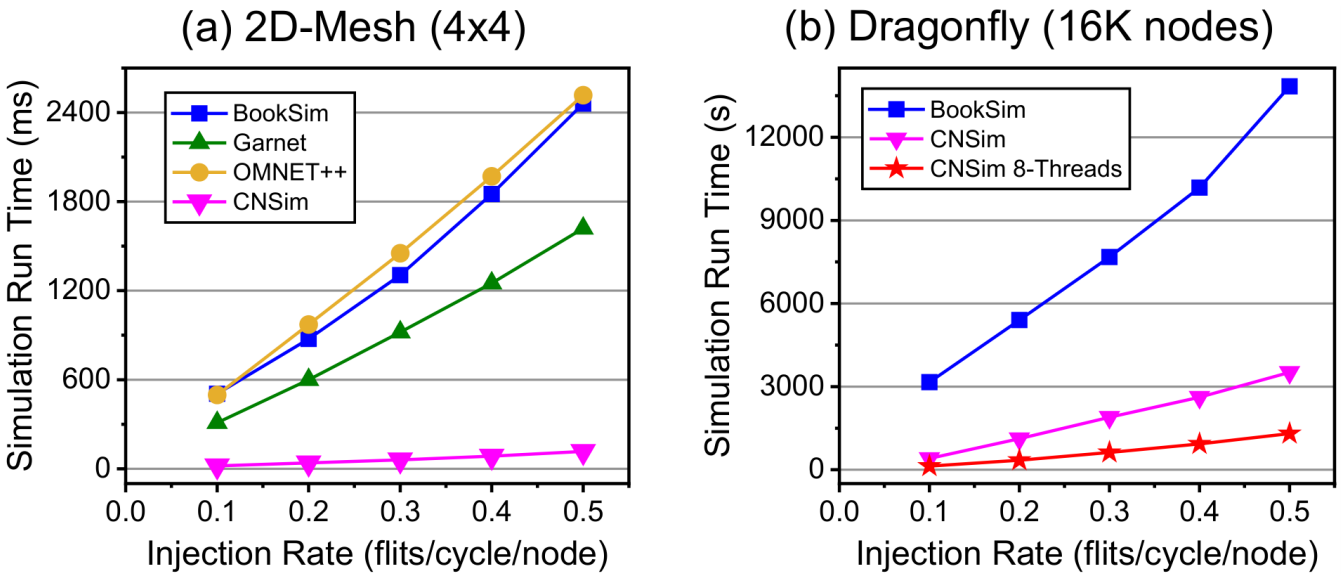
- Heterogeneous Router & Link
  - On-chip & off-chip networks are still different
  - Heterogeneity is a significant feature that existing simulators lack modeling.

- Cache the results of repetitive routing computations
  - By caching routing results, the simulation speed can be significantly improved.
  - But only part of the results, thus the memory usage is not significant

- Integration with real workload traces
  - Permutation patterns and AllReduce patterns
  - Real workload traces (PARSEC)

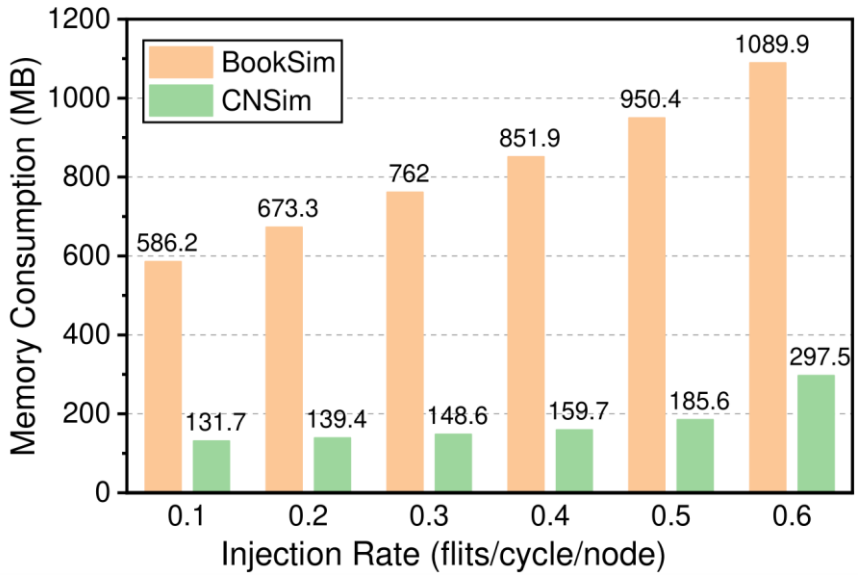
Evaluation Result

- Simulation Speed



Simulation speed is 11× ~ 14× faster than existing cycle-accurate simulators.

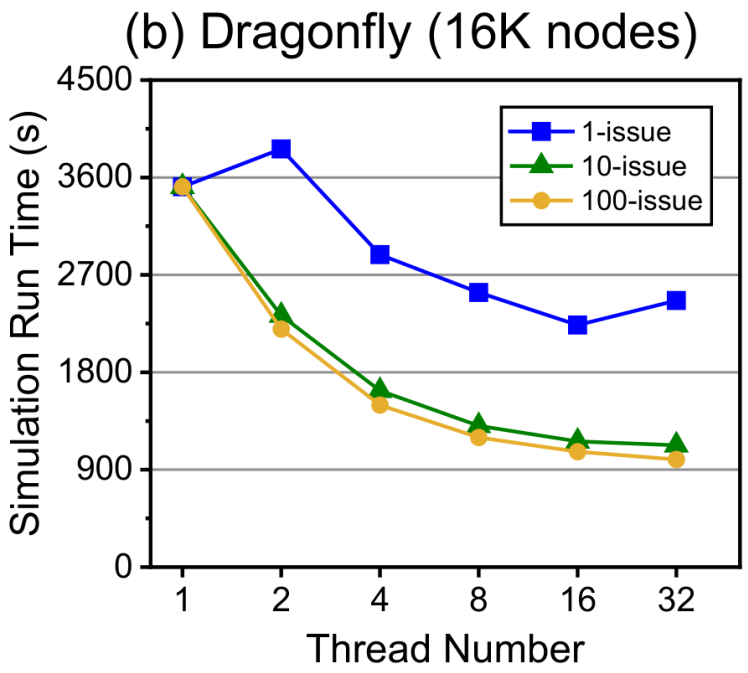
- Memory Consumption

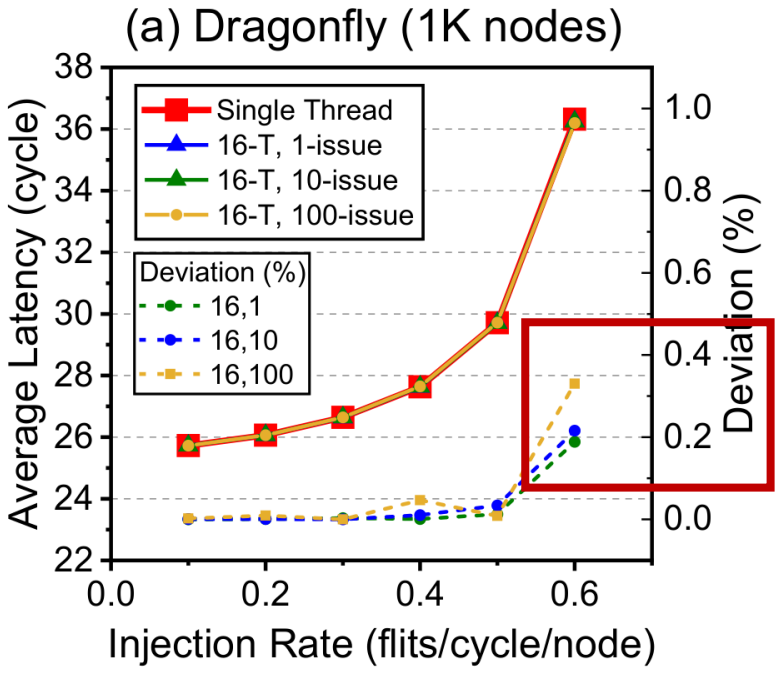


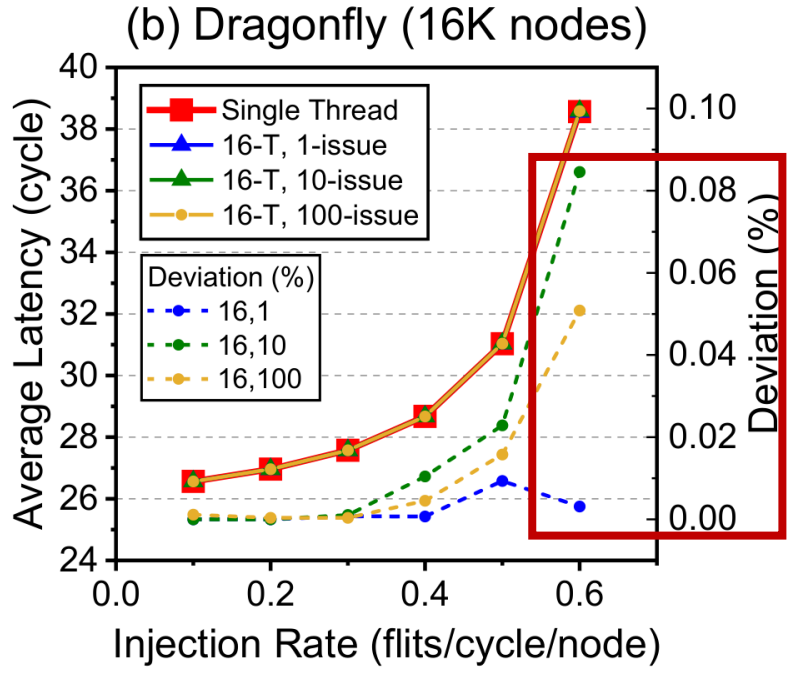
Memory consumption is very small.

Evaluation Result

- Hyper-threading Speedup
- Hyper-threading Inconsistency



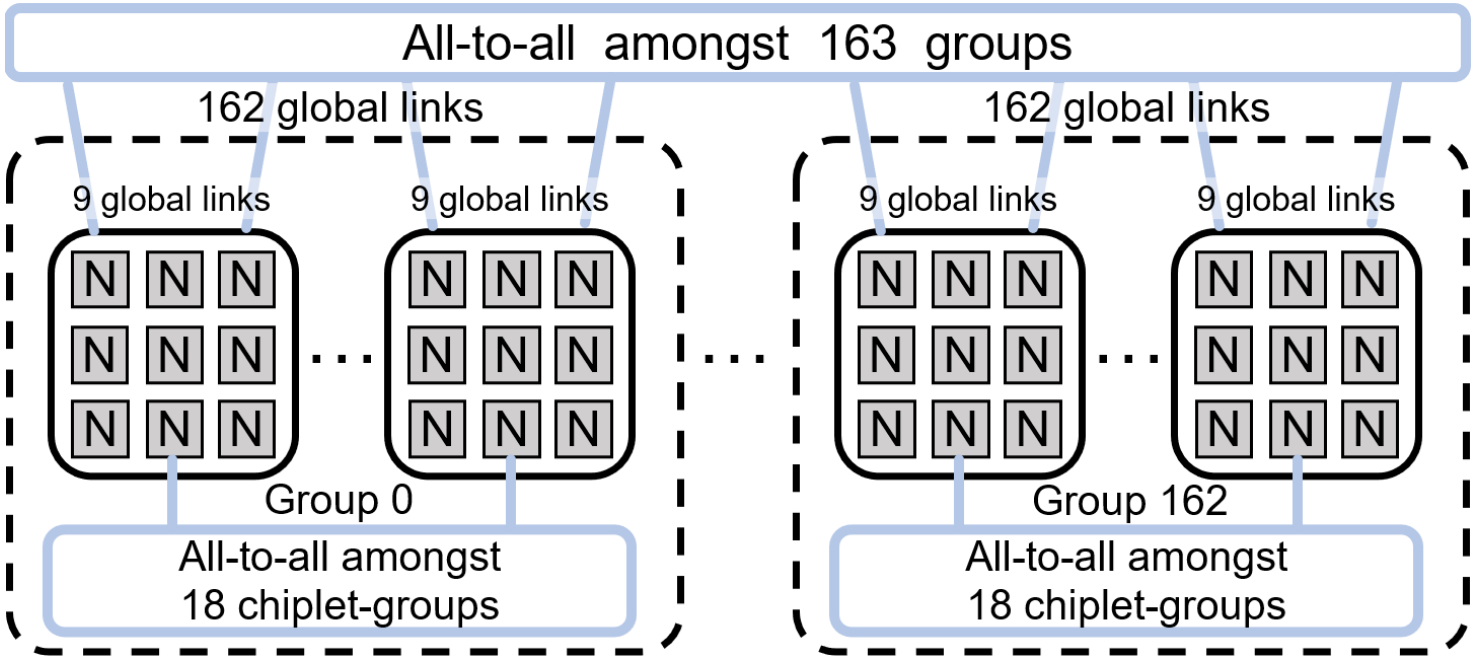




CNSim achieves significant parallel speedup with negligible hyper-threading inconsistency

Use Cases & Application Scenarios

- Evaluation of large-scale of network of chiplet
  - Performance: average packet latency, network throughput, etc.
  - Routing algorithms, flow-control policies, collective communication algorithms, etc.



Chiplet-based Switch-less Dragonfly on wafers

Run Time for Evaluation Real Networks

- Heterogeneous-Link-based Networks (MICRO 2023)
  - The entire PARSEC traces includes over **100B cycles and 3B messages**
  - One complete evaluation requires to run the traces for more than **10 times**
  - BookSim2: **> 10 days**
  - CNSim: **< 20 hours**

- Chiplet-based Switch-less Dragonfly (accepted by SC2024)
  - The entire network can have more than **200K chiplets (1M nodes)**
  - Each latency curve requires running from **0.1 to 1 flit/cycle/node** injection rate
  - BookSim2: **> 1 day**
  - CNSim: **a few hours**

Summary

- Cycle-accurate, packet-centric architecture
- Packet parallel-hyper-threading
- Sacrifice the simulator's ability to simulate the allocation policy in exchange for performance.
- High simulation speed and scalability, about 11× ~ 14× faster than other cycle-accurate simulators.
- Many functions and features
  - Integration of real workload traces
  - Heterogeneous link bandwidth and latency
  - Adaptive routing
  - **Open-sourced to the community**

清華大學
Tsinghua University

Thank You