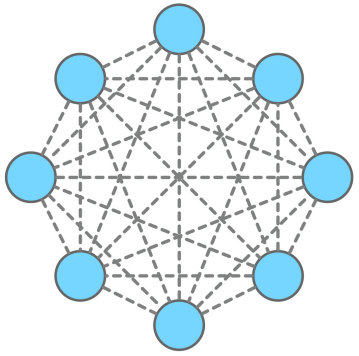


MAX PLANCK INSTITUTE
FOR INFORMATICS

OpenOptics: Enabling Open Research and Implementation of Optical Data Center Networks

Yiming Lei[1], **Federico De Marchi[1]**, Jialong Li[2], Raj Joshi[3], Shu-Ting Wang[4], Xiaoqi Chen[5], Balakrishnan Chandrasekaran[6], Yiting Xia[1]

Max Planck Institute for Informatics[1]

Shenzhen University of Advanced Technology[2], Red Hat / Harvard University[3],

UC San Diego[4], Purdue University[5], Vrije Universiteit Amsterdam[6]

Ever Growing Cloud Traffic

PyTorch

Model Training

Gemini

Claude

ChatGPT

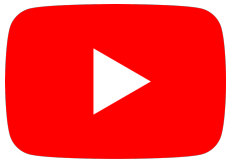
AI Workloads

Cluster Management

Hugging Face

GitHub

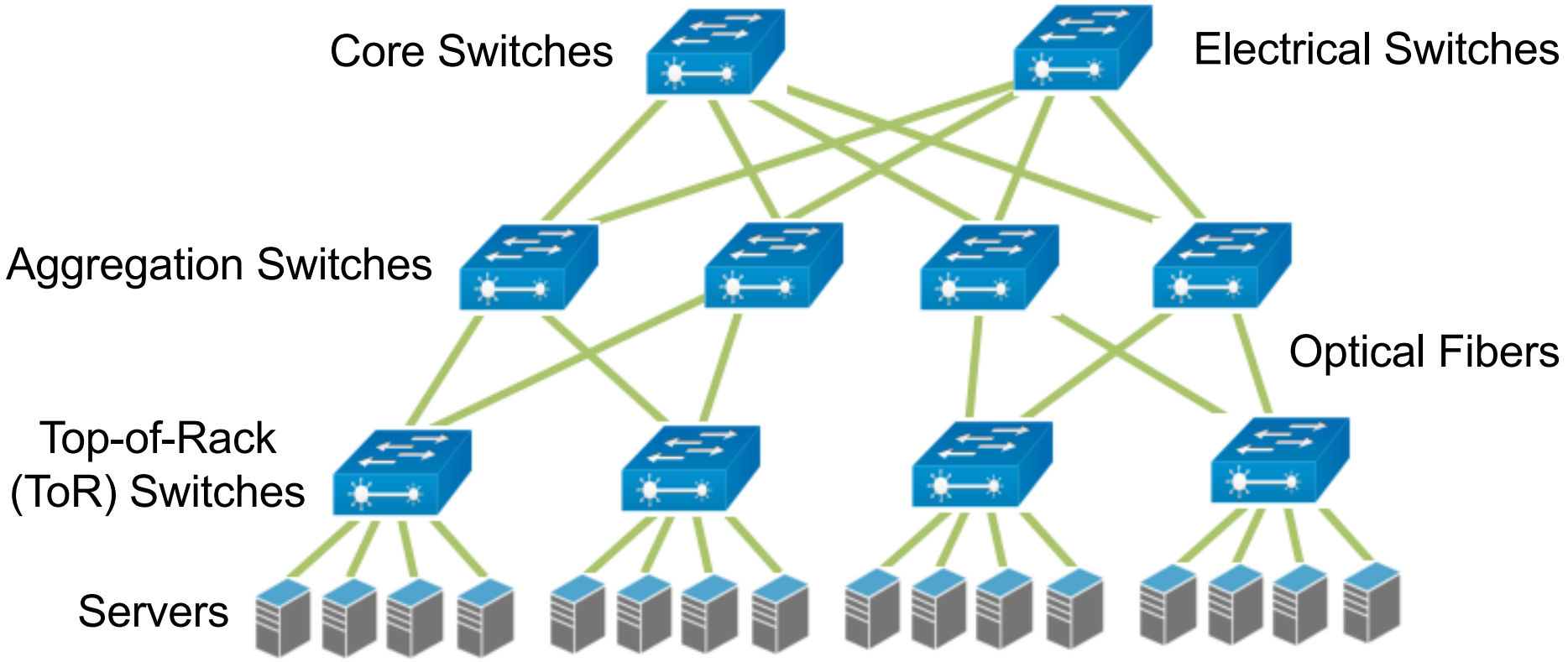
Community Platforms



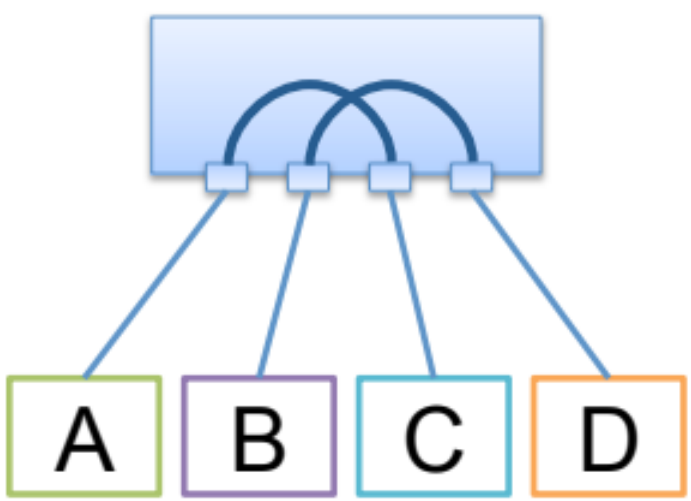
Streaming

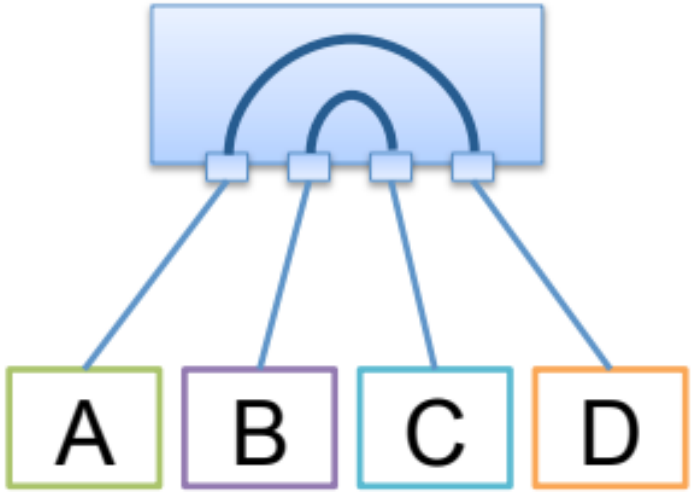
Moore's Law for Electrical Switches

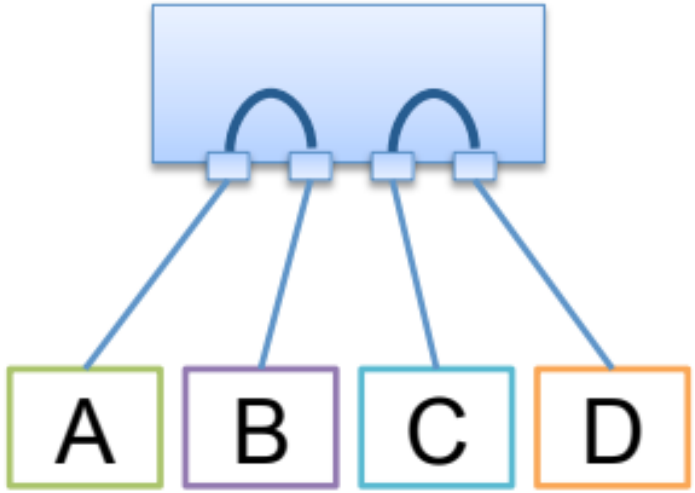
- Electrical switches double their bandwidth every two years at the same power and cost.



Optical Data Center Networks



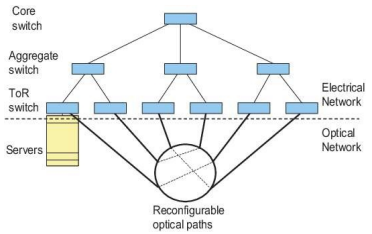




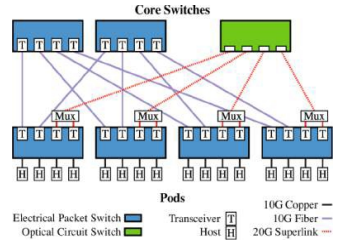
- Optical circuit switches
  - Dedicated links per configuration
  - An accurate transmission “schedule”
- Bufferless
  - No queuing
  - Need to buffer packets elsewhere

Optical Data Center Networks

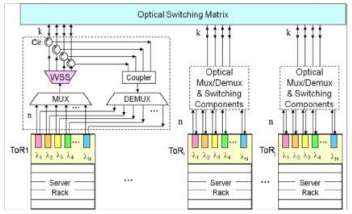
- **Power efficient**
  - Passive optical switches
- **Low cost**
  - Lower per-port cost than electrical switches
- **Future proof**
  - High bandwidth limit on optical switches
- **Flexible** network topology
  - Bring data and computation closer



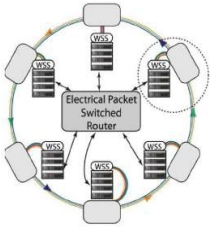
c-Through
[SIGCOMM 2010]



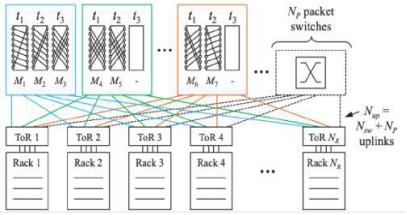
Helios
[SIGCOMM 2010]



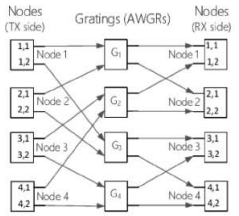
OSA
[NSDI 2012]



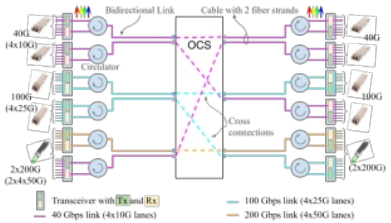
Mordia
[SIGCOMM 2013]



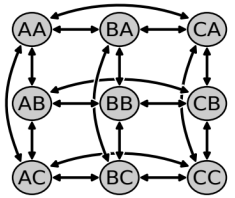
RotorNet
[SIGCOMM 2017]



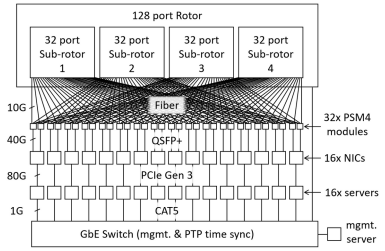
Sirius
[SIGCOMM 2020]



Jupiter
[SIGCOMM 2022]



Shale
[SIGCOMM 2024]



Realizing RotorNet
[SIGCOMM 2024]

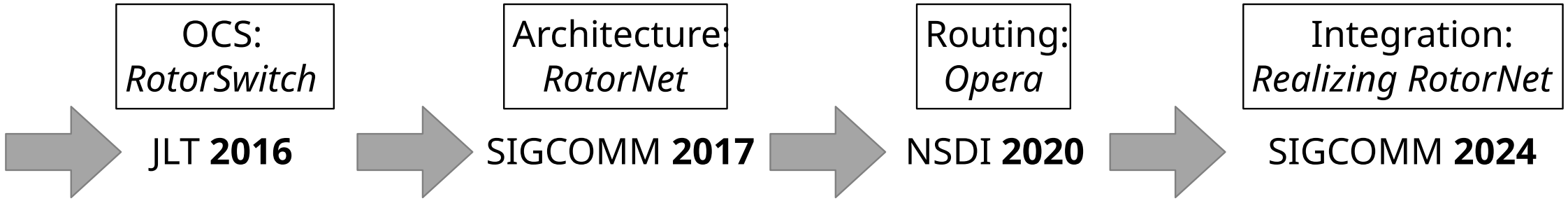
High Barrier of Entry

- **Closed ecosystem**
  - Specialized optical hardware + customized software stack
  - Home-grown simulation + small-scale testbed
- **Siloed research and innovation**
  - Software systems tied to the underlying hardware
  - Lack of a unified platform for implementation and evaluation

High Barrier of Entry

- Long Development Cycle

RotorNet

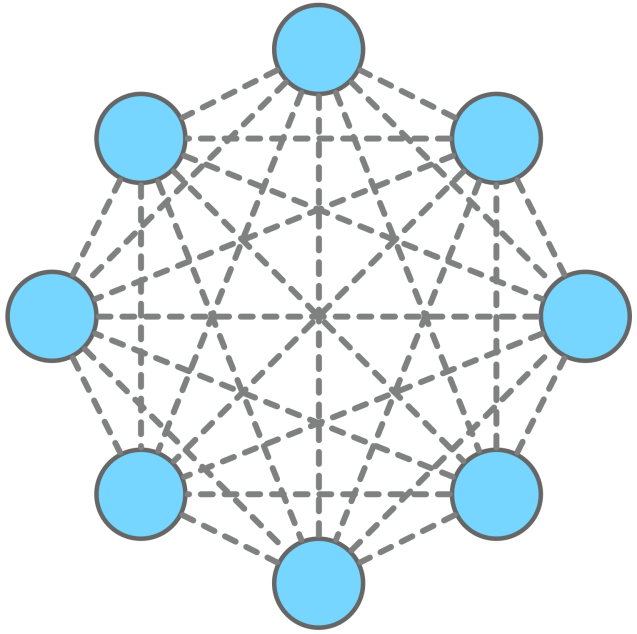


[JLT'16] A scalable, partially configurable optical switch for data center networks
[SIGCOMM'17] RotorNet: A Scalable, Low-complexity, Optical Datacenter Network
[NSDI'20] Expanding across time to deliver bandwidth efficiency and low latency
[SIGCOMM'24] Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking

High Barrier of Entry

- Architecture-Specific Design

	Jupiter	**Sirius**
Company	Google	Microsoft
OCS Type	MEMS	AWGR
Switch Interval	Minutes	Nanoseconds
Scheduling	Traffic-aware	Traffic-oblivious

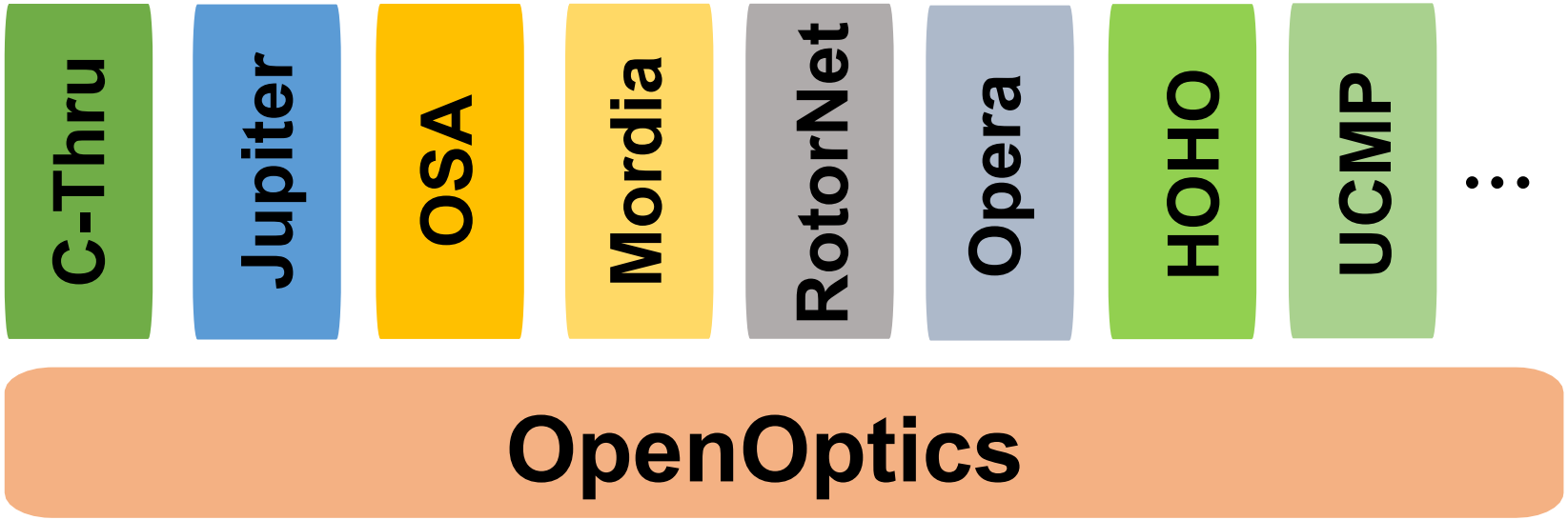


OpenOptics

Decouple Software from Hardware → Evolve Independently

What is in OpenOptics

- Optical DCN architecture plug-and-play
- A simple abstraction + user API
- Multi-backend support: Testbed, Simulation, Emulation
- To enable easy design and testing



Who is OpenOptics for

- **With physical hardware**
  - Shorten development cycle with built-in software
  - Software-stack evaluation with emulated optical hardware
- **Without physical hardware**
  - Single-laptop emulation or simulation
  - For initial research and exploration, for classroom education

How to Unite Different Architectures?

DCN Applications

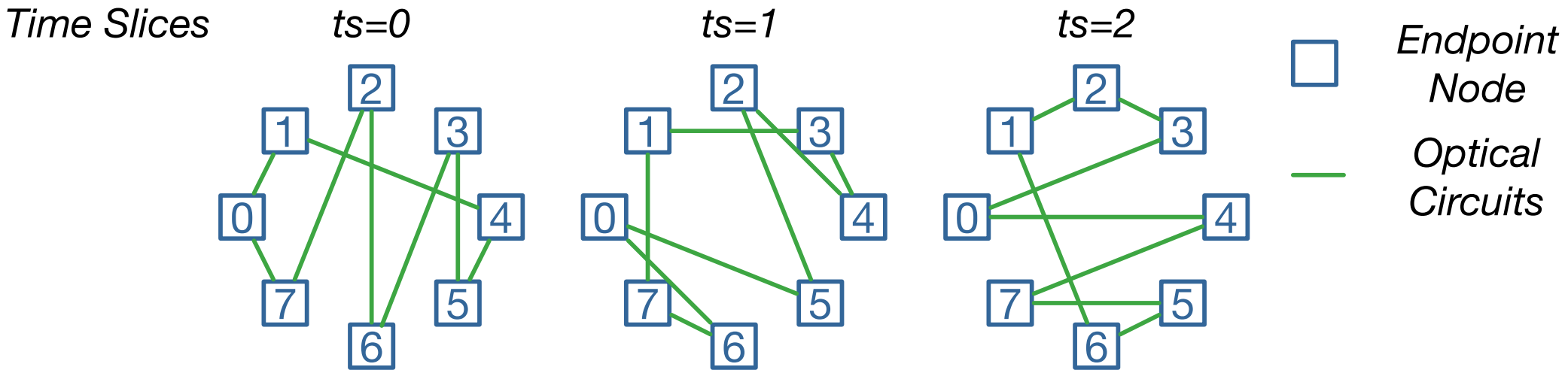
Transport Protocol

Routing

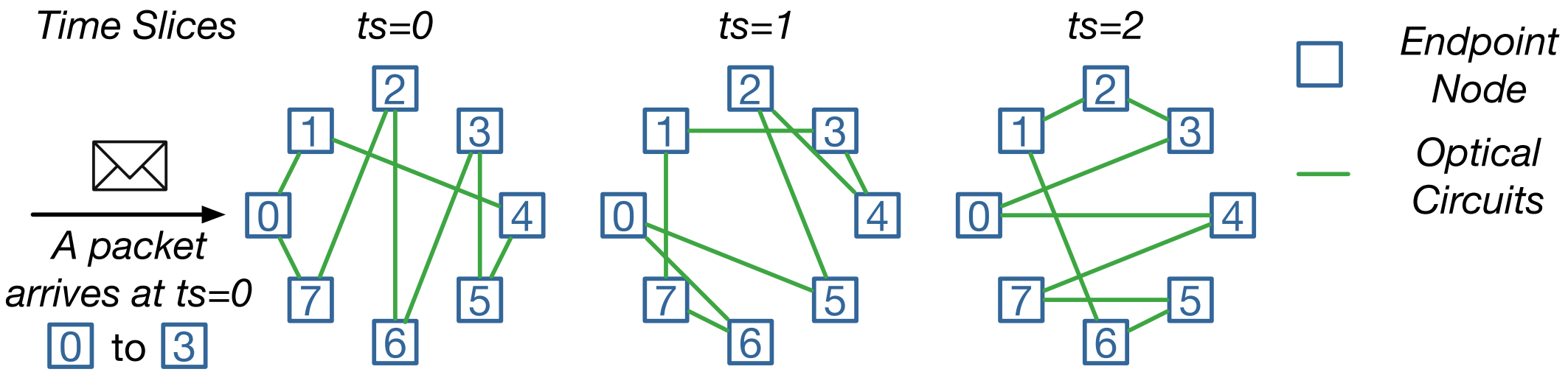
Packet Scheduling

Optical Architectures

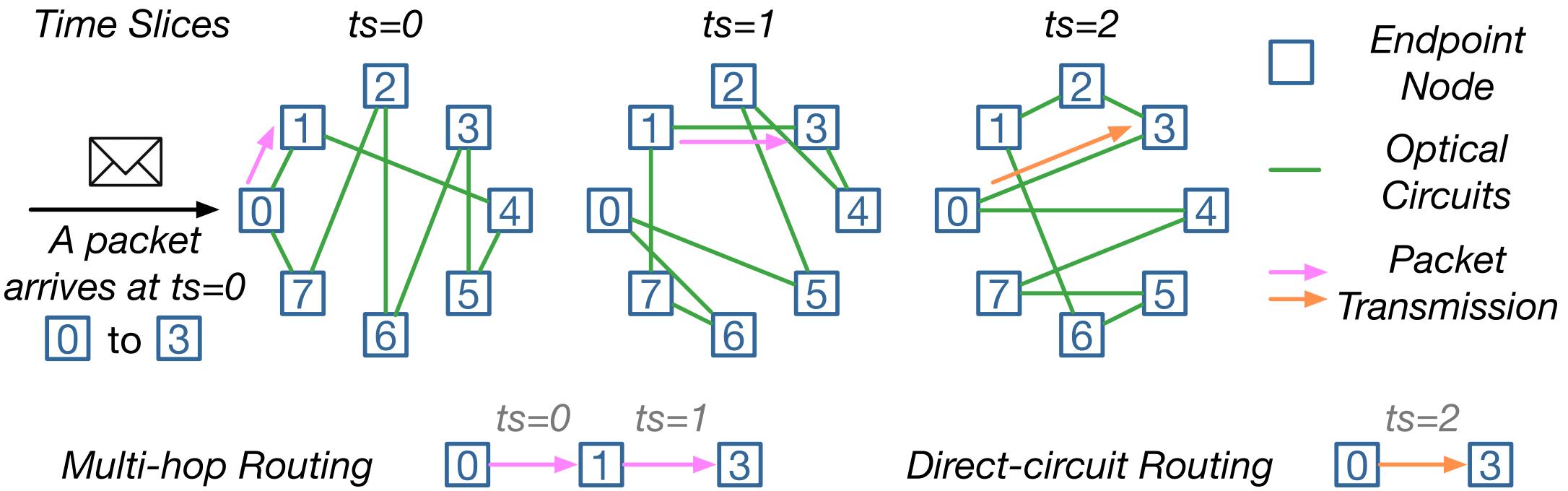
Routing in Optical DCNs



Routing in Optical DCNs



Routing in Optical DCNs



Routing is different across time slices

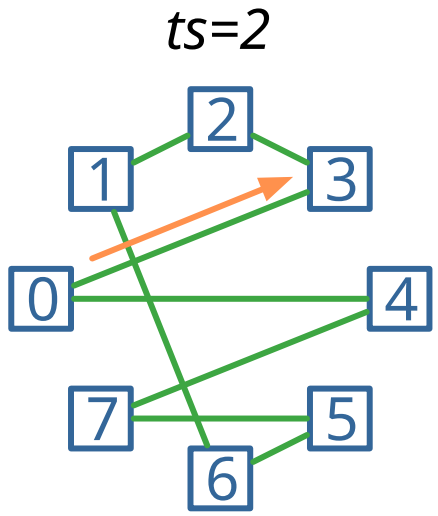
Packets may wait until a certain time to continue

Time-Flow Table

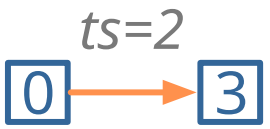
Node 0

Arrival Slice	Dst.	Egress Port	Departure Slice
0	3	2	2

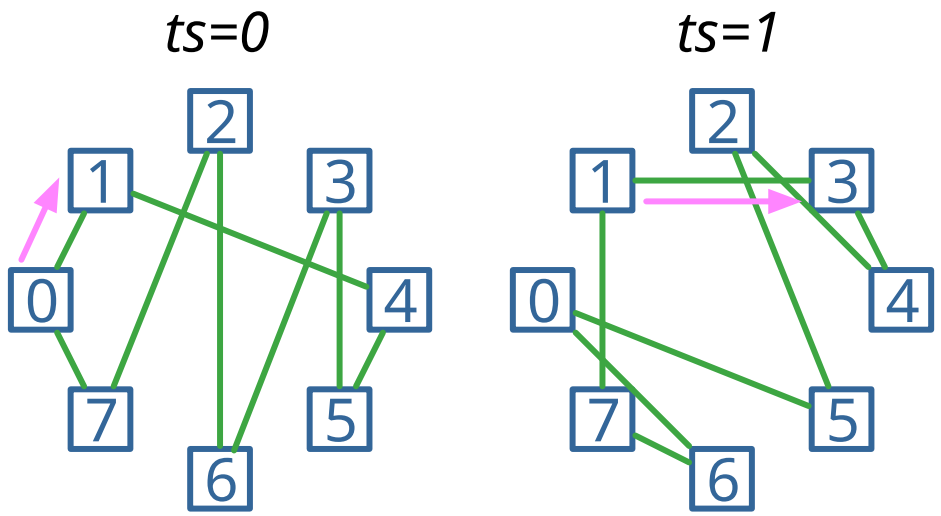
Direct path



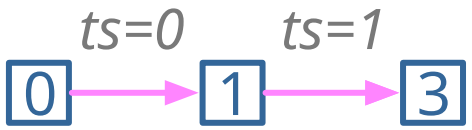
Direct-circuit Routing



Time-Flow Table



Multi-hop Routing



Node 0

Arrival Slice	*Dst.*	*Egress Port*	*Departure Slice*
0	*3*	*1*	*0*

Node 1

Arrival Slice	*Dst.*	*Egress Port*	*Departure Slice*
0	*3*	*2*	*1*

Multi-hop path (per-hop lookup)

Time-Flow Table

Node 0

Arrival Slice	*Dst.*	*Egress Port*	*Departure Slice*
*	3	2	*

Traditional flow table

Node 0

Arrival Slice	*Dst.*	*Source Routing Action* *Hops: < Egress Port, Departure Slice>*
0	3	<1,0>,<2,1>

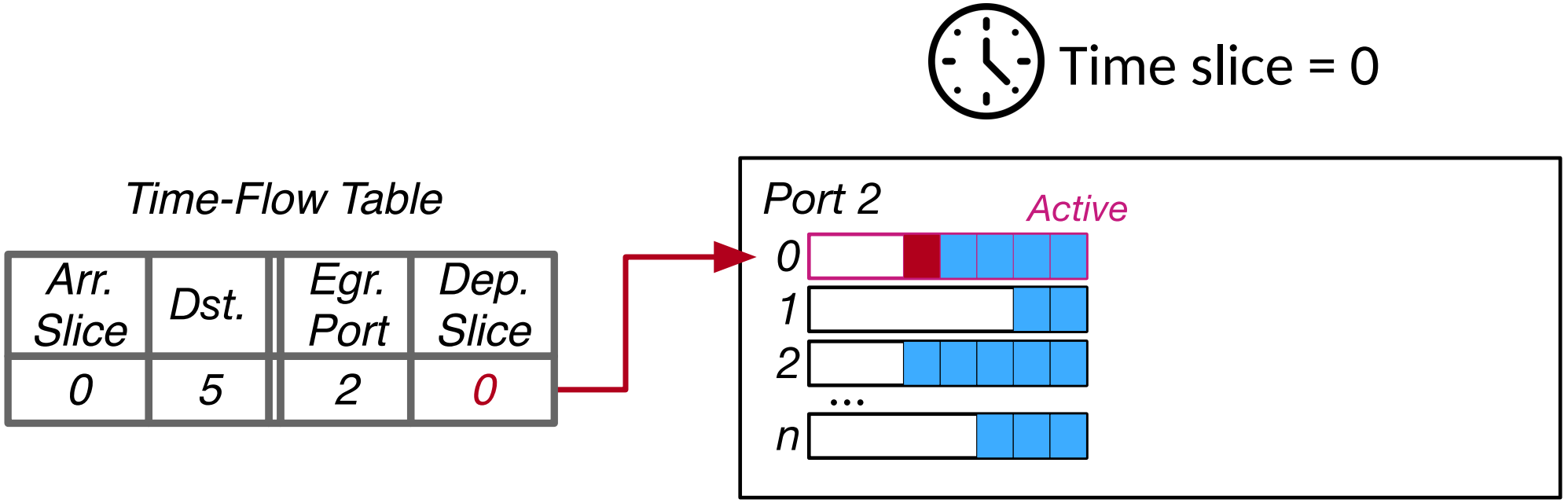
Multi-hop path (source routing)

Under Time-Flow Table: Time-Based Packet Scheduling

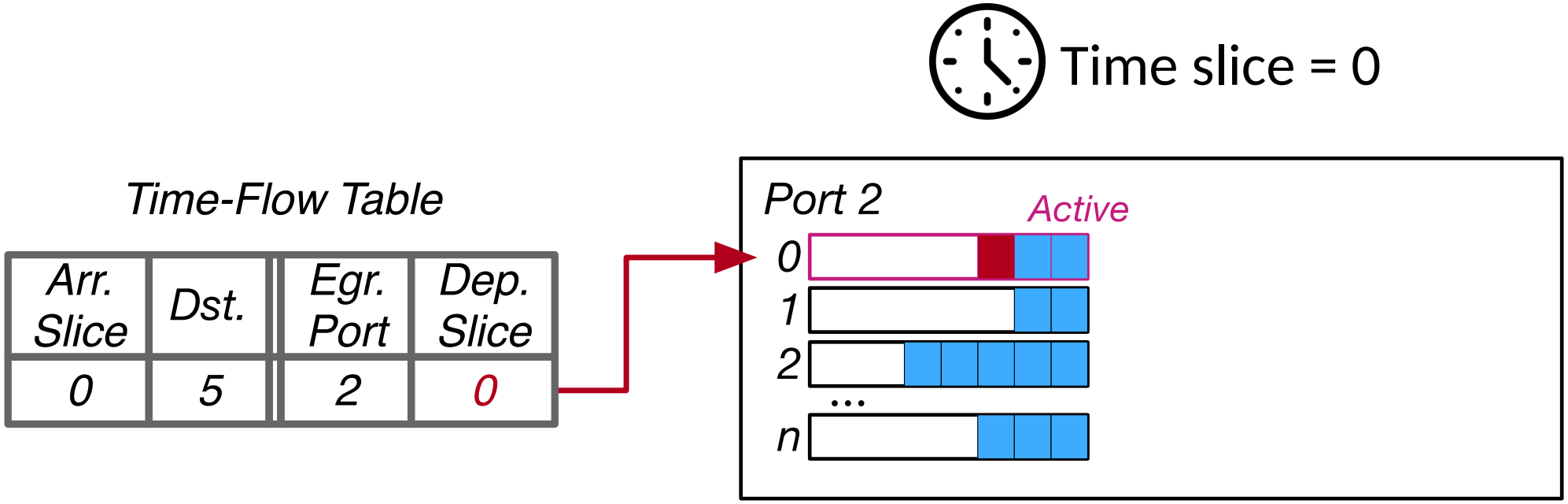
Time-Flow Table

Arr. Slice	*Dst.*	*Egr. Port*	*Dep. Slice*
0	*5*	*2*	*0*
1	*5*	*3*	*2*

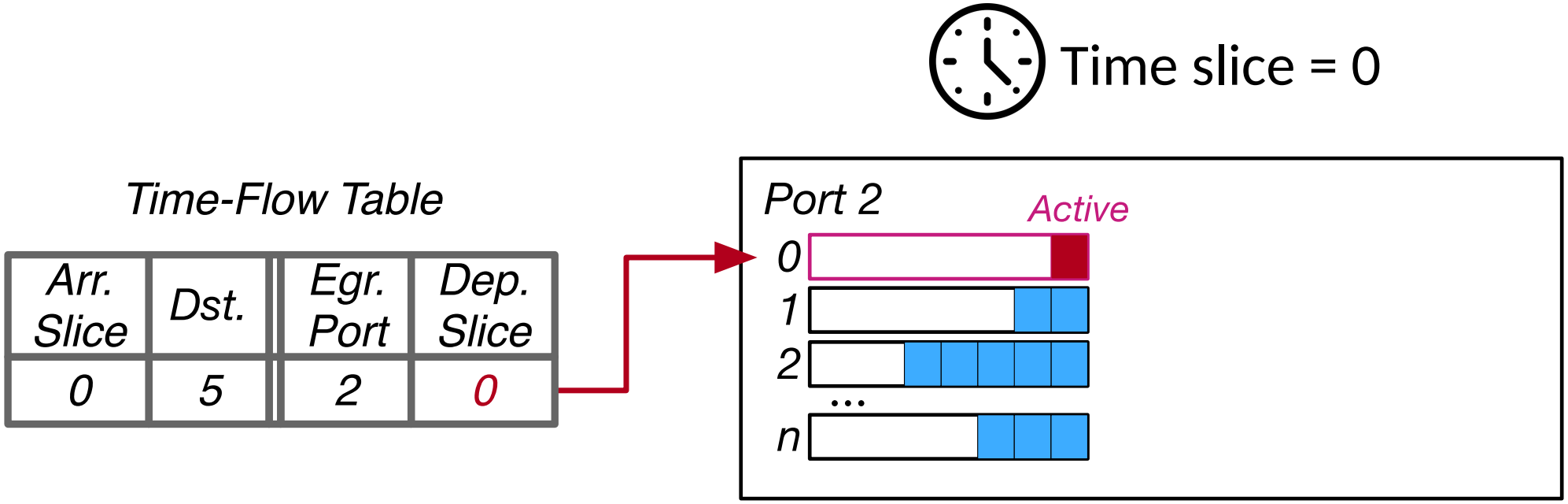
Under Time-Flow Table: Time-Based Packet Scheduling



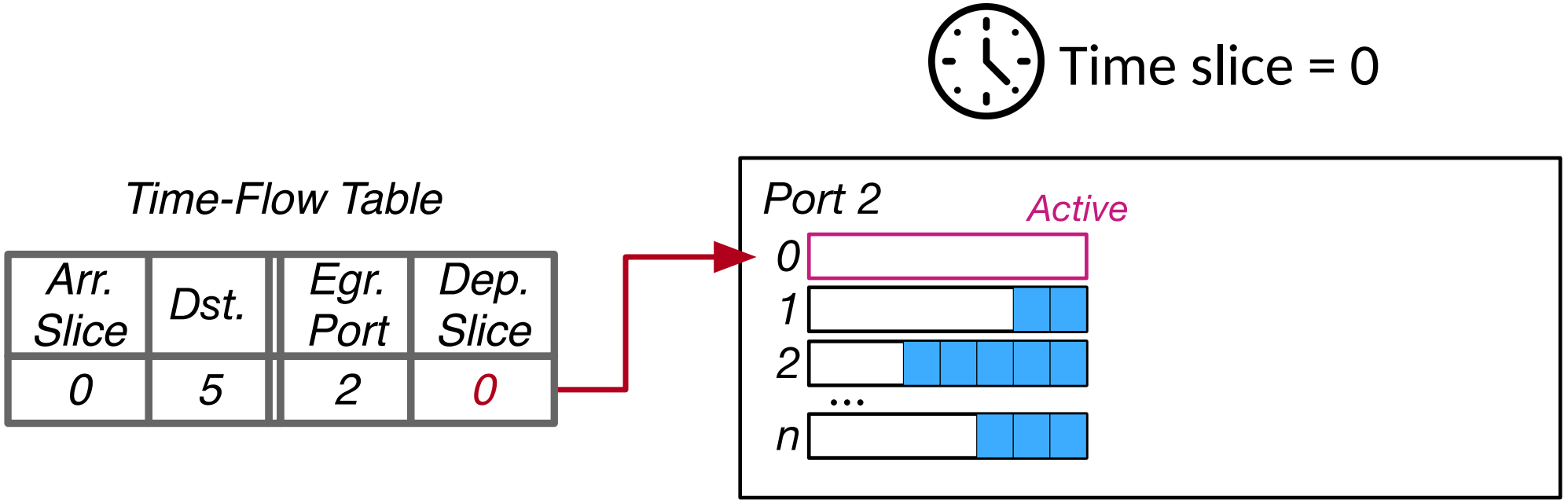
Under Time-Flow Table: Time-Based Packet Scheduling



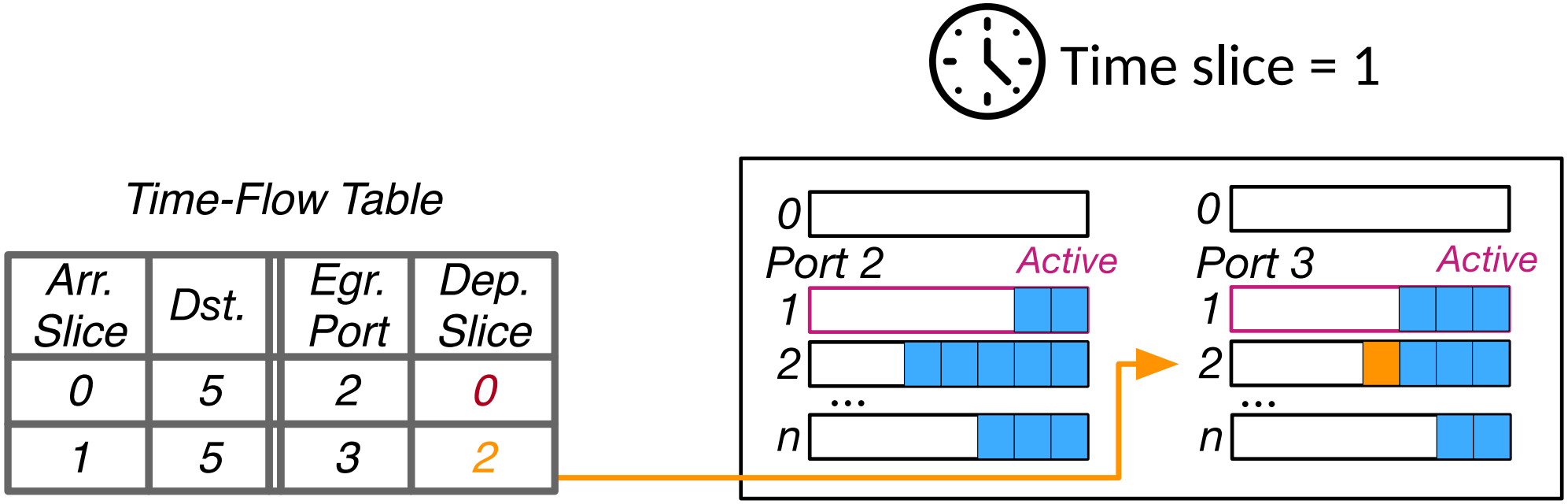
Under Time-Flow Table: Time-Based Packet Scheduling



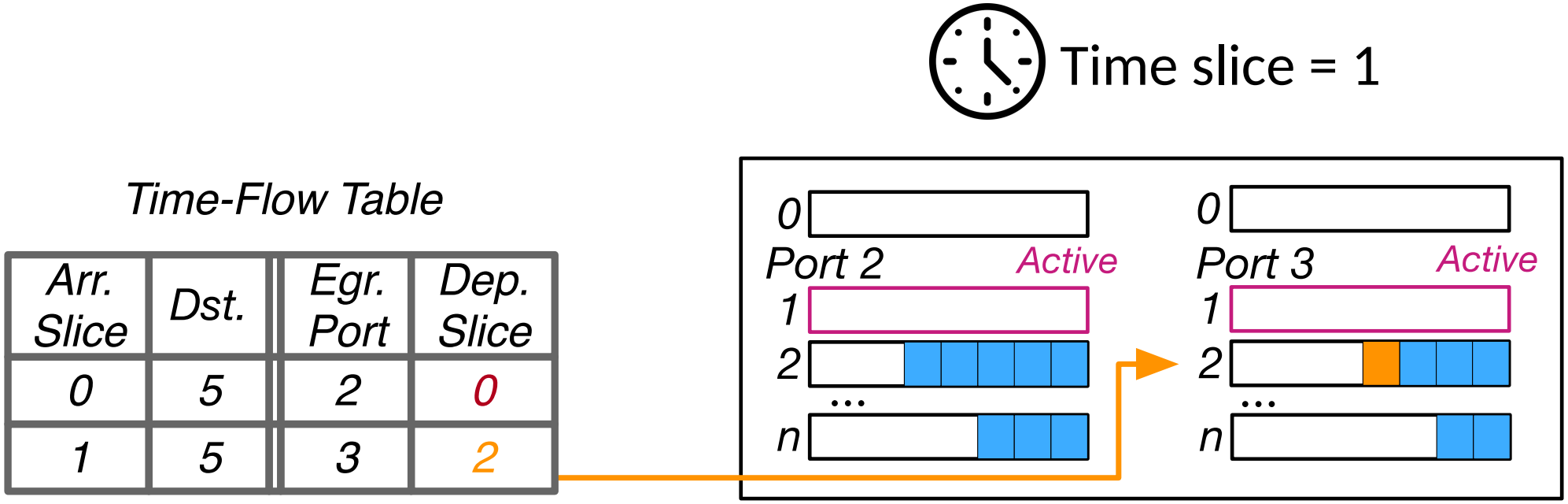
Under Time-Flow Table: Time-Based Packet Scheduling



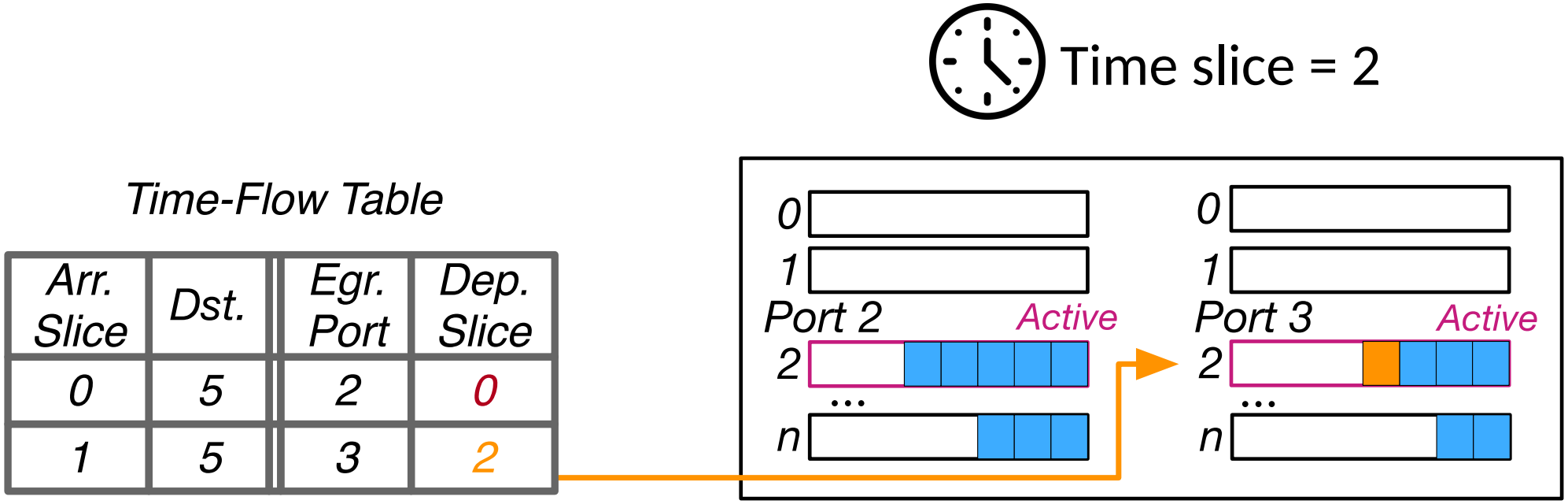
Under Time-Flow Table: Time-Based Packet Scheduling



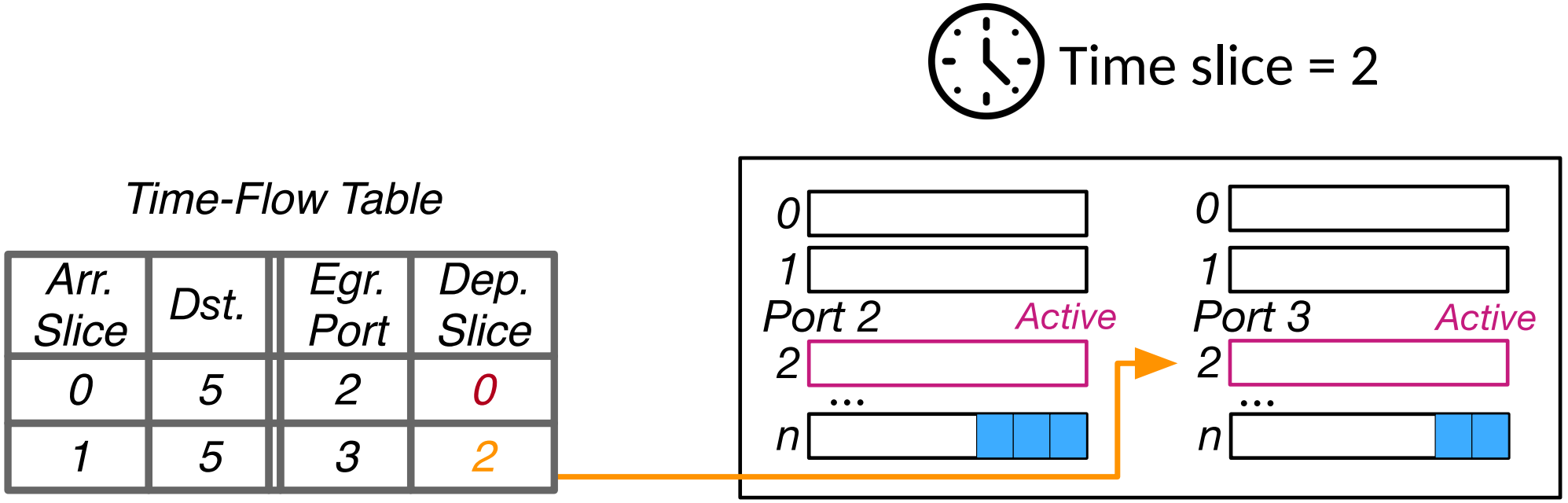
Under Time-Flow Table: Time-Based Packet Scheduling



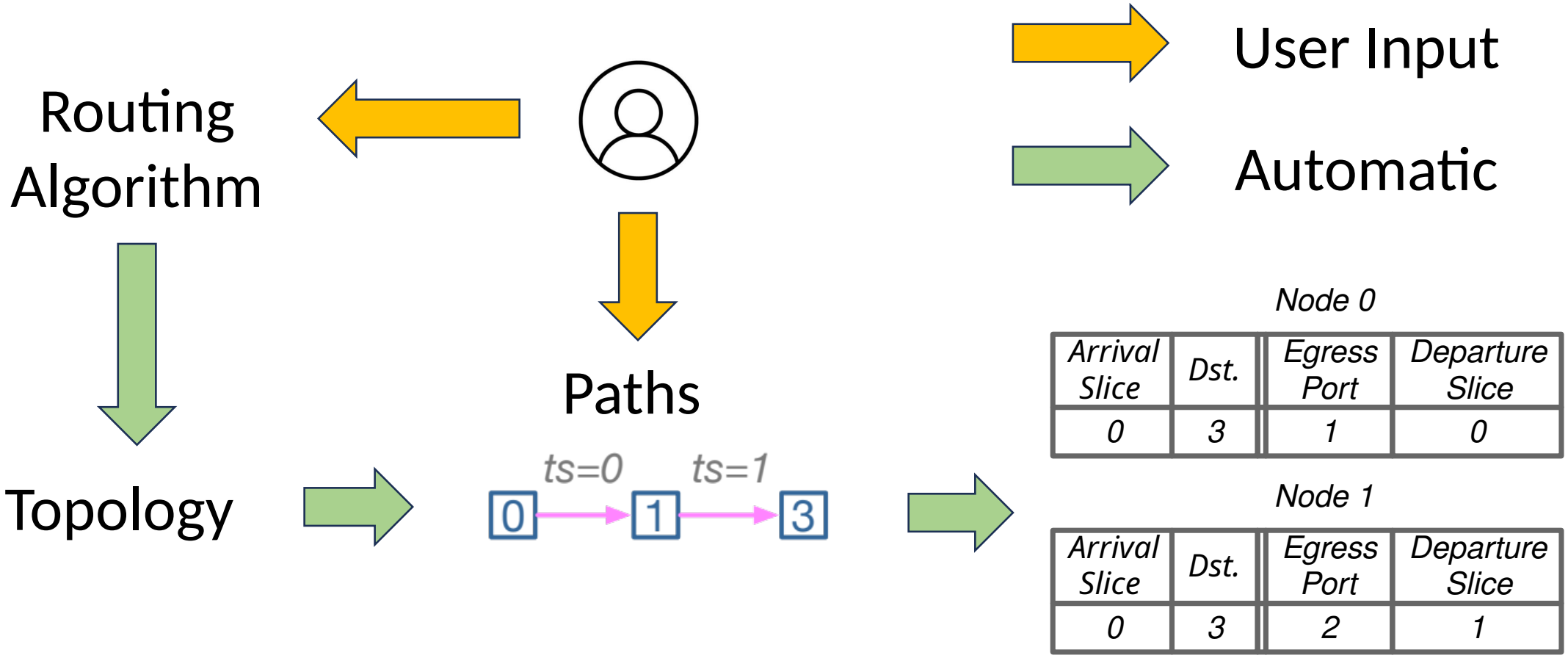
Under Time-Flow Table: Time-Based Packet Scheduling



Under Time-Flow Table: Time-Based Packet Scheduling



Above Time-Flow Table:
Automatic Time-Flow Table Generation



User APIs

Routing:

add_path(Path)

→ Registers a routing path between source and destination

User APIs

Routing:

add_path(Path)

→ Registers a routing path between source and destination

routing_algo(src, dst, time_slice, topo) -> Path [User-defined]

→ Custom routing logic implemented by user

User APIs

Routing:

add_path(Path)

routing_algo(src, dst, time_slice) -> Path [User-defined]

Topology:

connect(node1, node2, time_slice)

User APIs

Routing:

add_path(Path)

routing_algo(src, dst, time_slice) -> Path [User-defined]

Topology:

connect(node1, node2, time_slice)

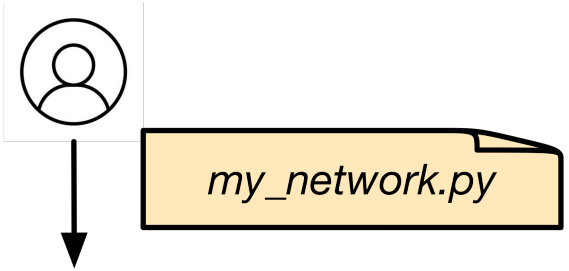
Monitor:

queue_depth(switch, port, queue)

packet_drop(switch)

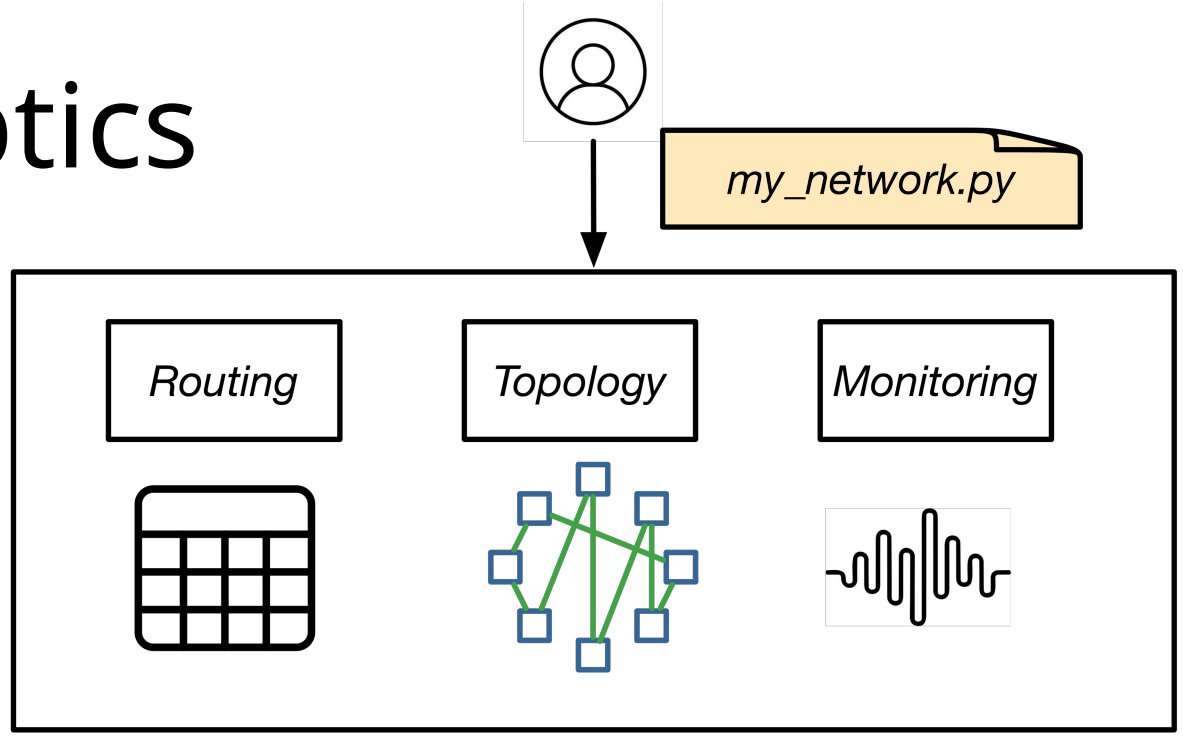
…

OpenOptics System



User Script (Python)

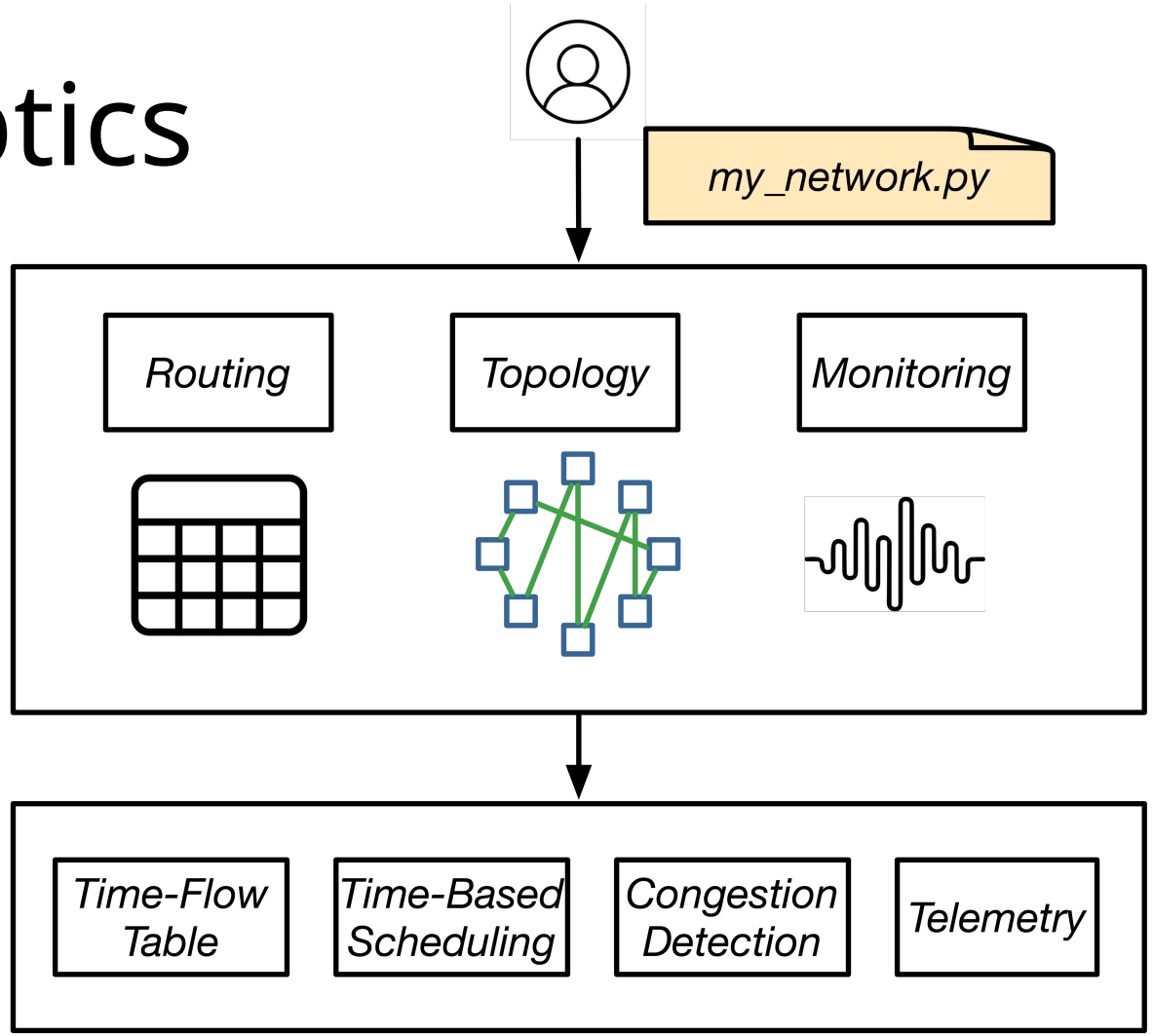
OpenOptics System



User Script (Python)

OpenOptics Control Plane (Python/C++)

OpenOptics System

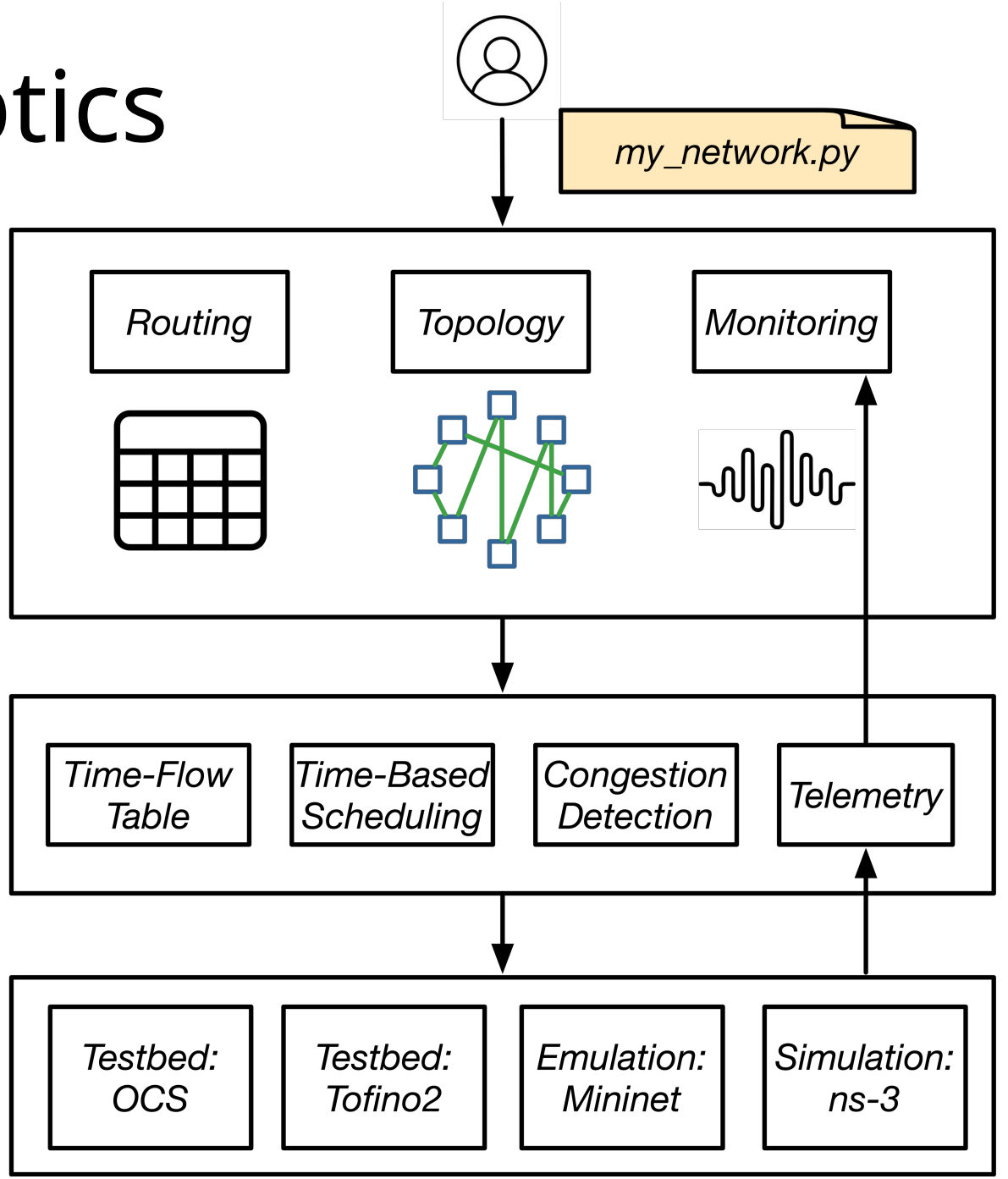


User Script (Python)

OpenOptics Control Plane (Python/C++)

OpenOptics Data Plane (P4)

OpenOptics System



User Script (Python)

OpenOptics Control Plane (Python/C++)

OpenOptics Data Plane (P4)

OpenOptics Backends

Infrastructure Services

- Congestion detection
- Traffic push-back
- Buffer offloading
- Flow pausing
- Traffic collection

- Please refer to our paper

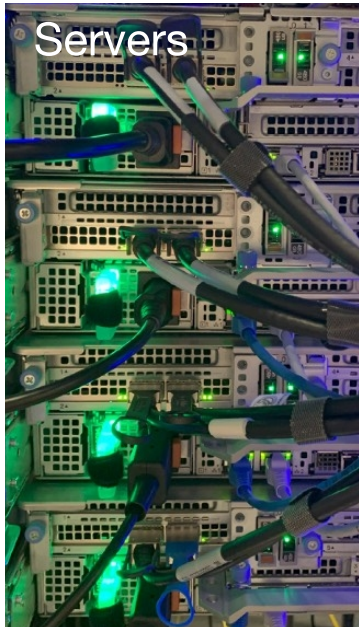
The Question:

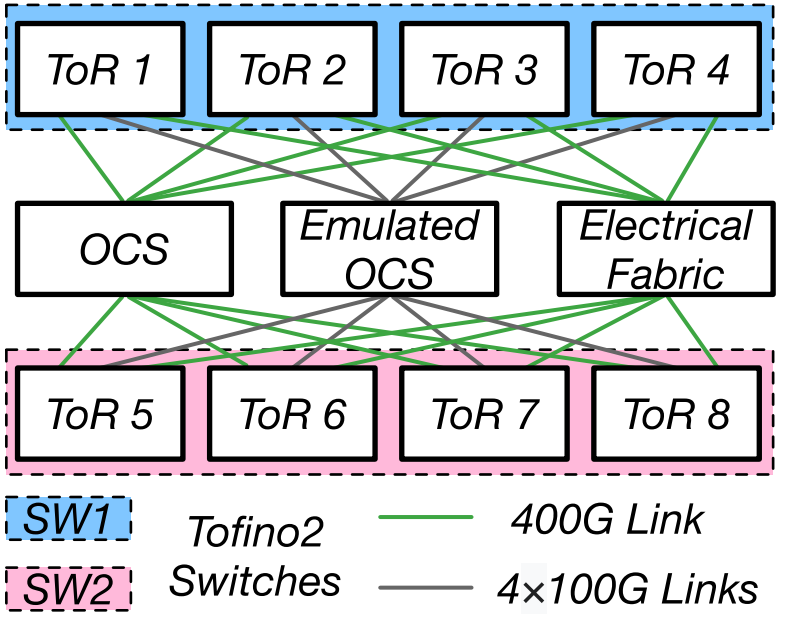
How Easy Can It Be to Deploy an Optical DCN in OpenOptics?

Answer: As Little as 10 Lines of Python

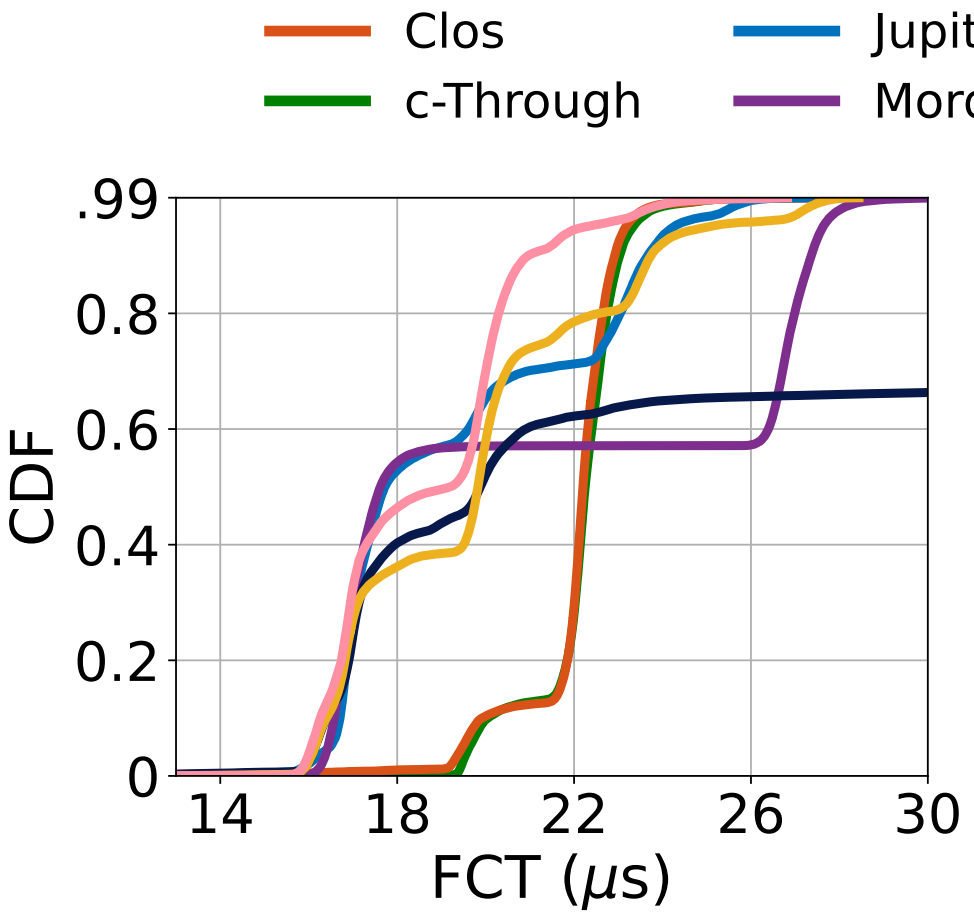
```python
from openoptics import Toolbox, OpticalTopo, OpticalRouting

net = Toolbox.BaseNetwork(nb_node=8, time_slice_duration_ms=128)
circuits = OpticalTopo.round_robin(nb_node=8)
net.deploy_topo(circuits)
paths = OpticalRouting.routing_hoho(net.get_topo())
net.deploy_routing(paths)
net.start()
```

Evaluation on the Testbed

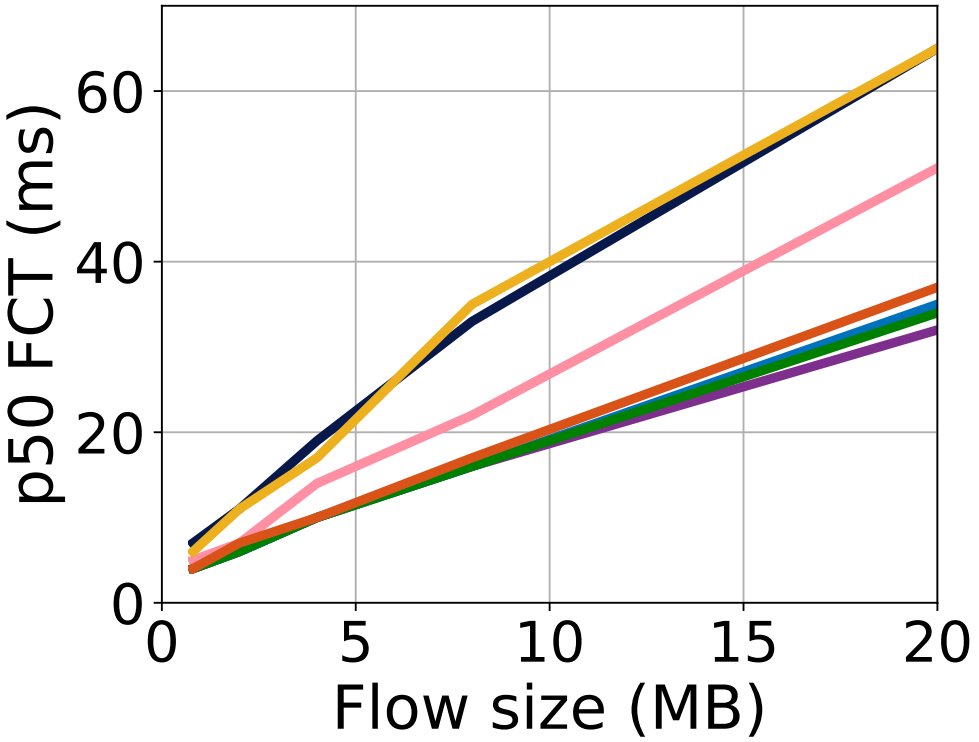




Cross Architecture Comparison

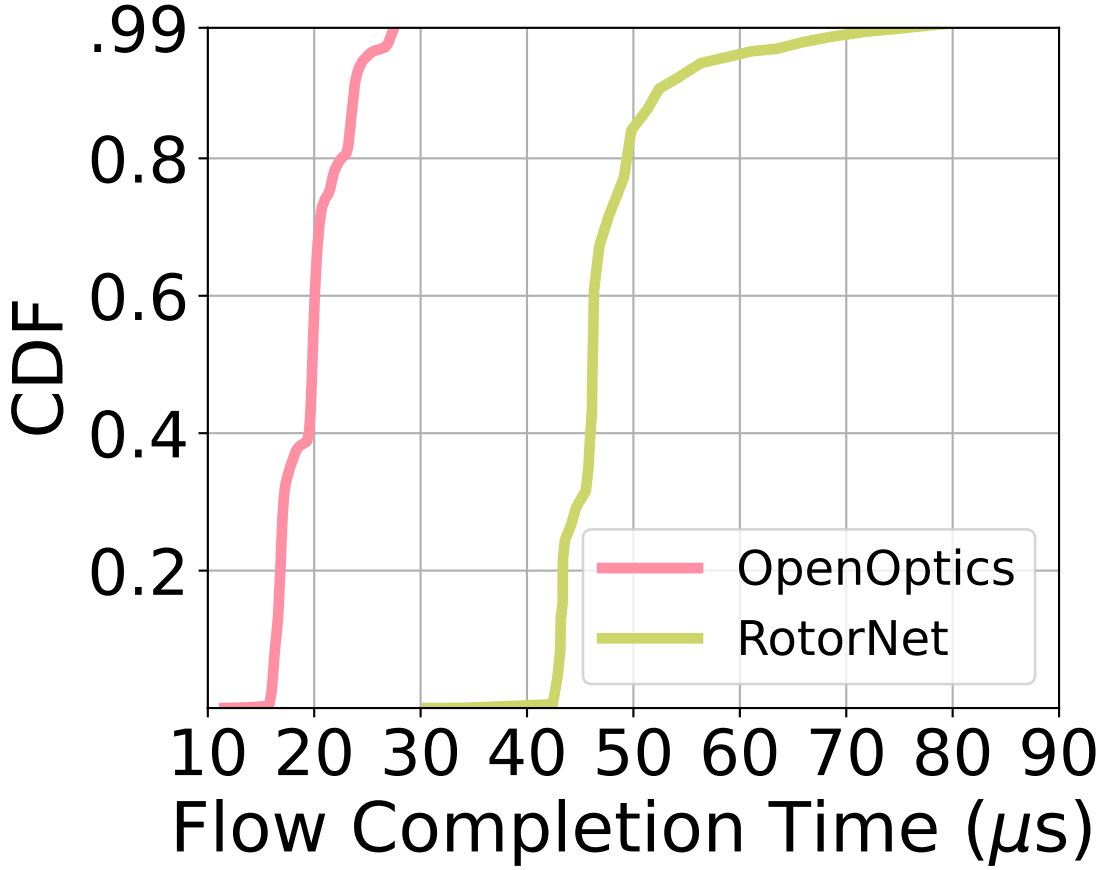


Memcached



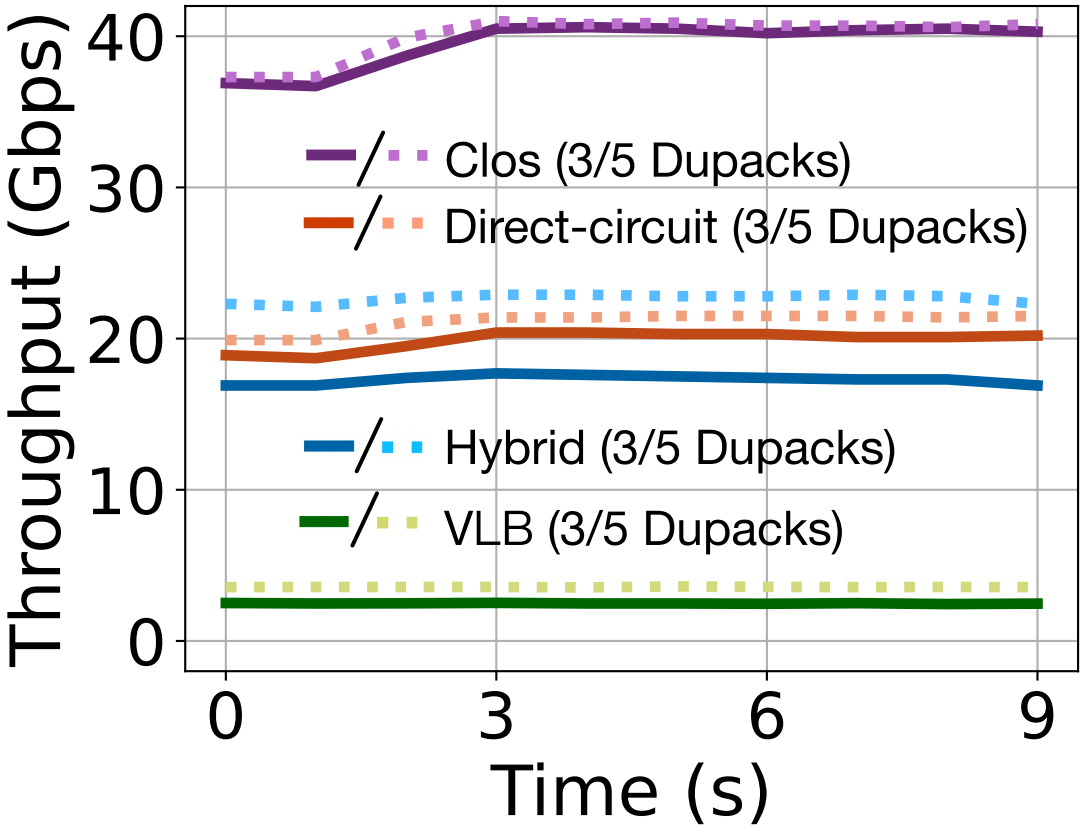
All-reduce

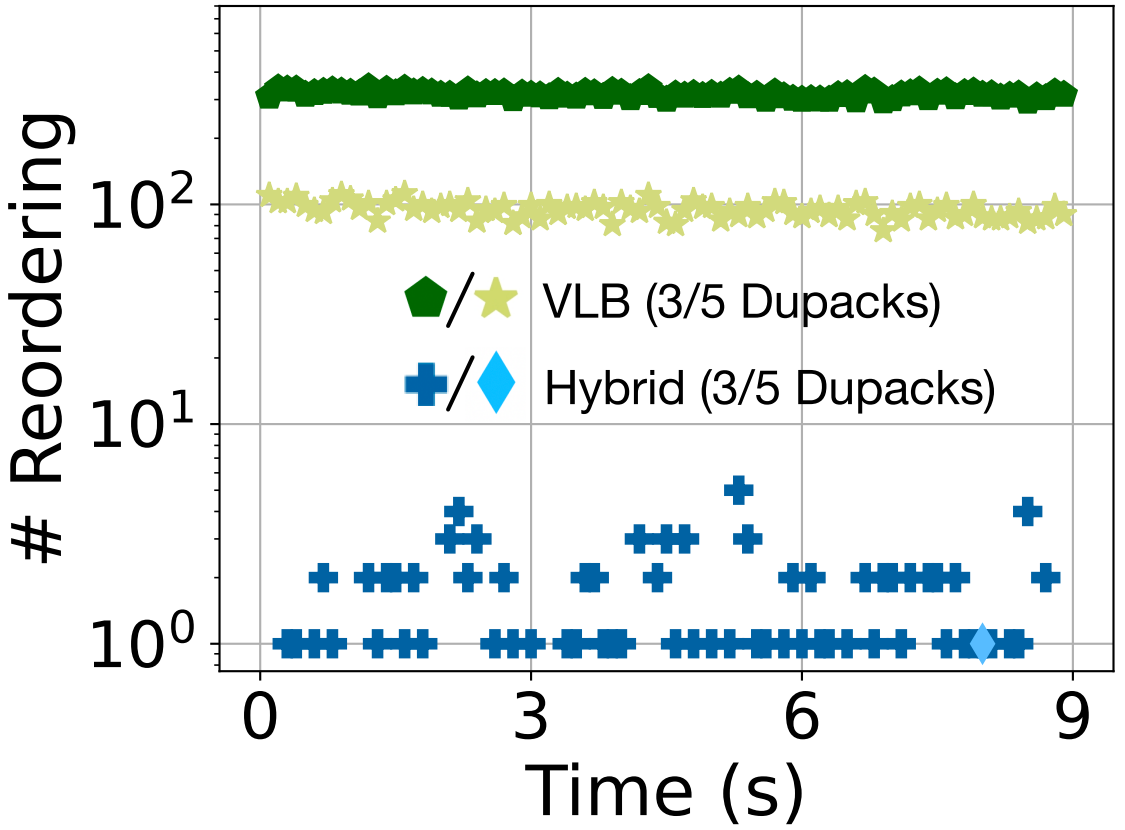
Compare with Original Results



Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking, SIGCOMM'24

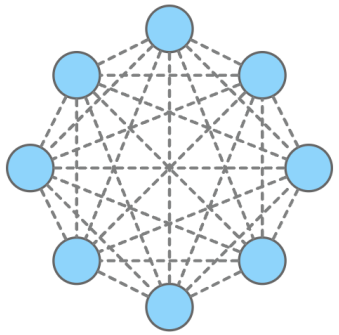
Case Study: Congestion Control





Observibility: Dashboard

Open Source



⚡ OpenOptics ⚡

Search ⌘ + K

- Quick Start
- Installation
- Tofino Backend
- Examples
  - Topology Primitive
  - Routing Primitive
  - Direct Routing
  - Fastest Path Routing
  - Source Routing

OpenOptics: Democratizing Optical DCNs

Easy design, testing, and deployment of optical DCNs for everyone.

Note

- The OpenOptics paper has been accepted by NSDI'26!

OpenOptics is a general framework for realizing different optical data center network architectures in a plug-and-play manner. With OpenOptics, users can deploy customized optical data center networks on the testbed, emulation, or simulation with ~10 lines of Python code. Under the hood, user configurations are translated into table entries and control plane programs, which are then deployed to the underlying optical and P4-programmable switches.

```
from openoptics import Toolbox, OpticalTopo, OpticalRouting

net = Toolbox.BaseNetwork(name="demo", backend="Mininet",
                          nb_node=8, time_slice_duration_ms=512)
circuits = OpticalTopo.round_robin(nb_node=nb_node)
net.deploy_topo(circuits)
paths = OpticalRouting.routing_direct(net.get_topo())
net.deploy_routing(paths, routing_mode="Per-hop")
```

GitHub

⚡ OpenOptics ⚡

Website

Easy design, testing, and deployment of optical DCNs for everyone.

Yiming Lei
ylei@mpi-inf.mpg.de