Harvard John A. Paulson
School of Engineering
and Applied Sciences

MAX PLANCK INSTITUTE
FOR INFORMATICS

Red Hat AI

SyncWise: Error-Aware Time Synchronization for Reconfigurable Data Center Networks

Yiming Lei, Jialong Li[^], Zhengqing Liu[*], **Raj Joshi**[†#], Yiting Xia

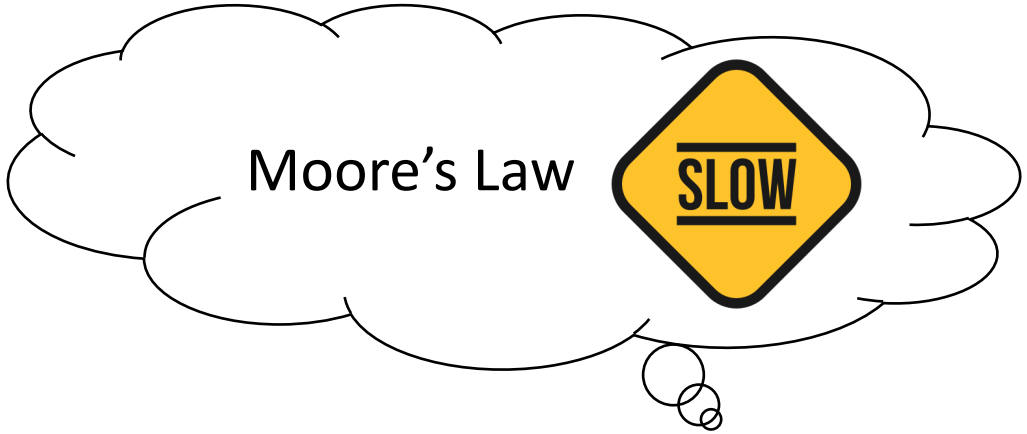
Max Planck Institute for Informatics

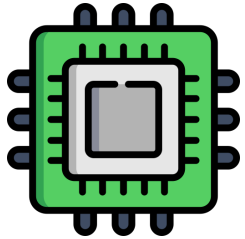
[^]Shenzhen University of Advanced Technology

[*]Imperial Colledge of London

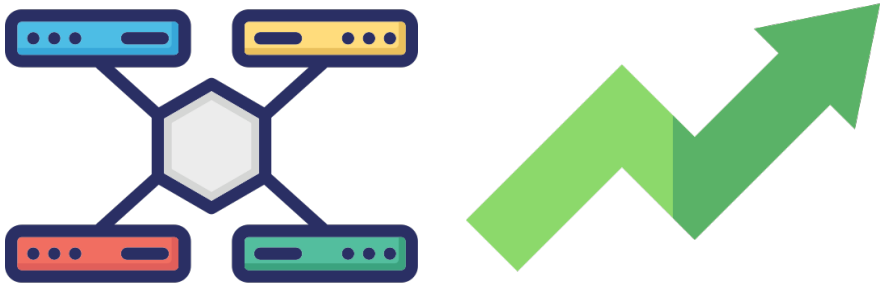
[†]Red Hat [#]Harvard University

Scaling Network Bandwidth Beyond Silicon





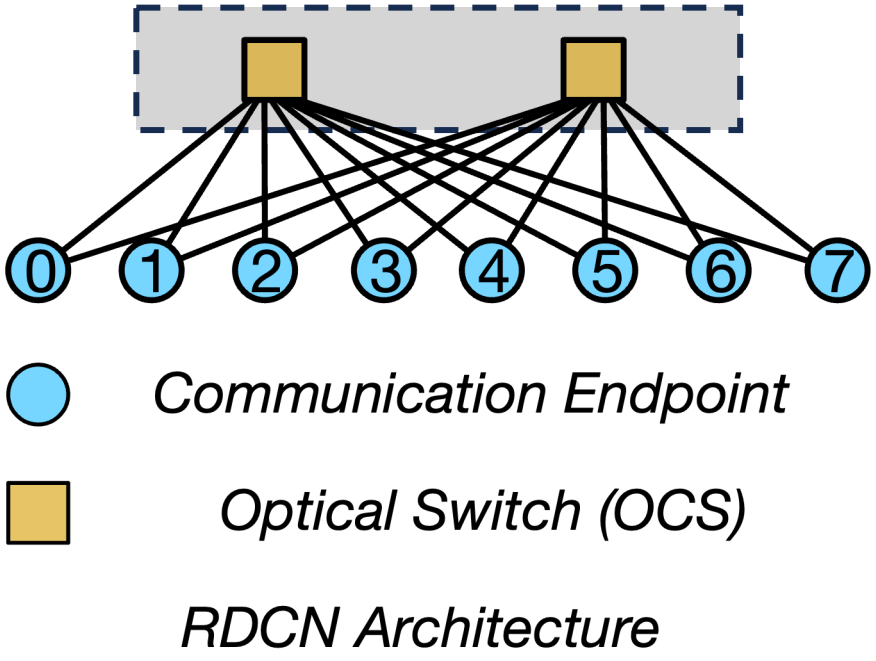
Merchant Silicon



Need new ways to scale DC network bandwidth

Reconfigurable Data Center Networks (RDCNs)

Electrical pkt-switching core → Optical Circuit Switches



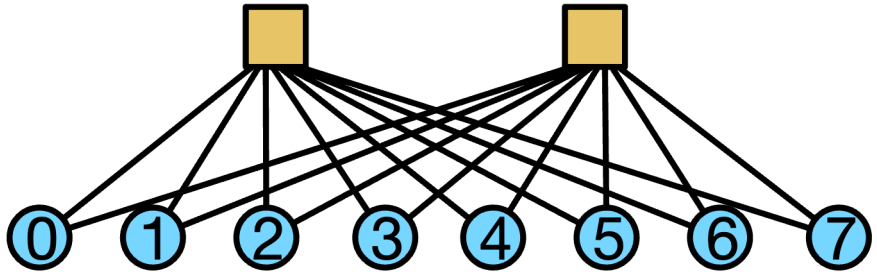
Optical circuit switches (OCS)

- Energy-efficient and cost-effective
- High-bandwidth alternative to power-hungry electrical switching ASICs

Reconfigurable Data Center Networks

Trade-off with Optical Circuit Switches

- 1 connection between a port-pair at a given time

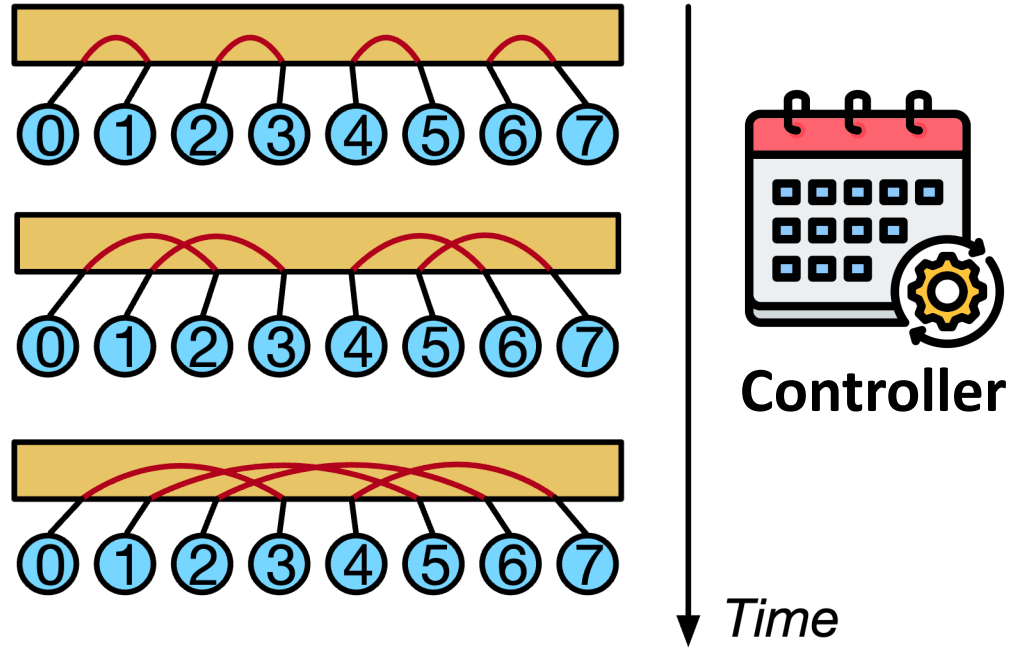


Communication Endpoint

Optical Switch (OCS)

RDCN Architecture

Emulate all-to-all connections (like pkt-switched networks)



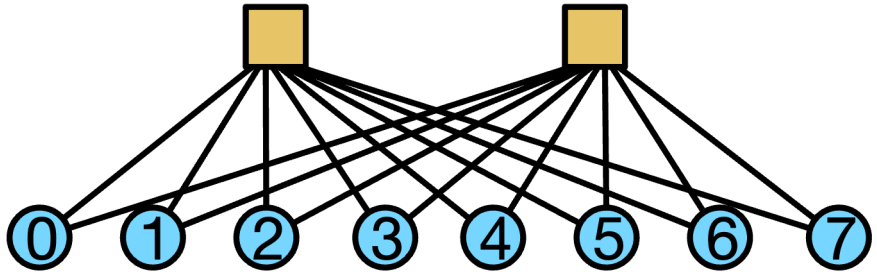
OCS Internal Connection

OCS Configurations

Reconfigurable Data Center Networks

Trade-off with Optical Circuit Switches

- 1 connection between a port-pair at a given time

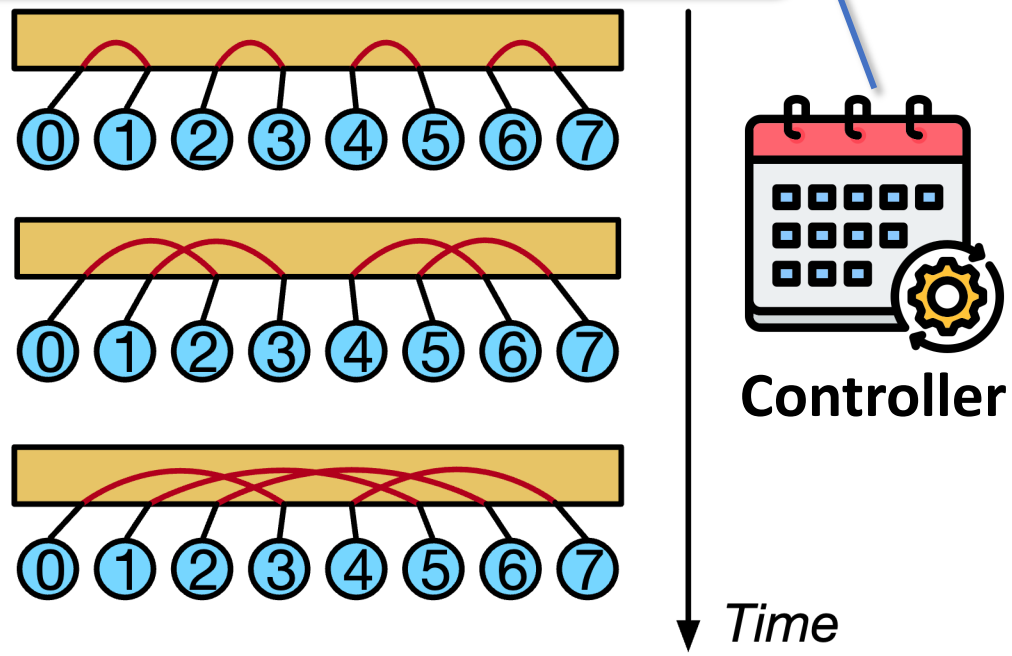


Communication Endpoint

Optical Switch (OCS)

RDCN Architecture

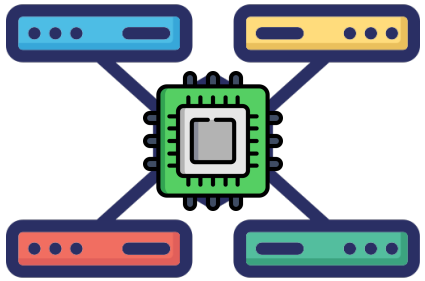
Modern OCS: reconfigure circuits on µs to ns timescales



OCS Internal Connection

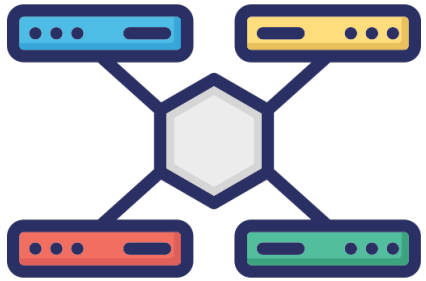
OCS Configurations

Time Sync is Essential for RDCN



Electrical pkt-switched DCN

Time Sync is an
application requirement

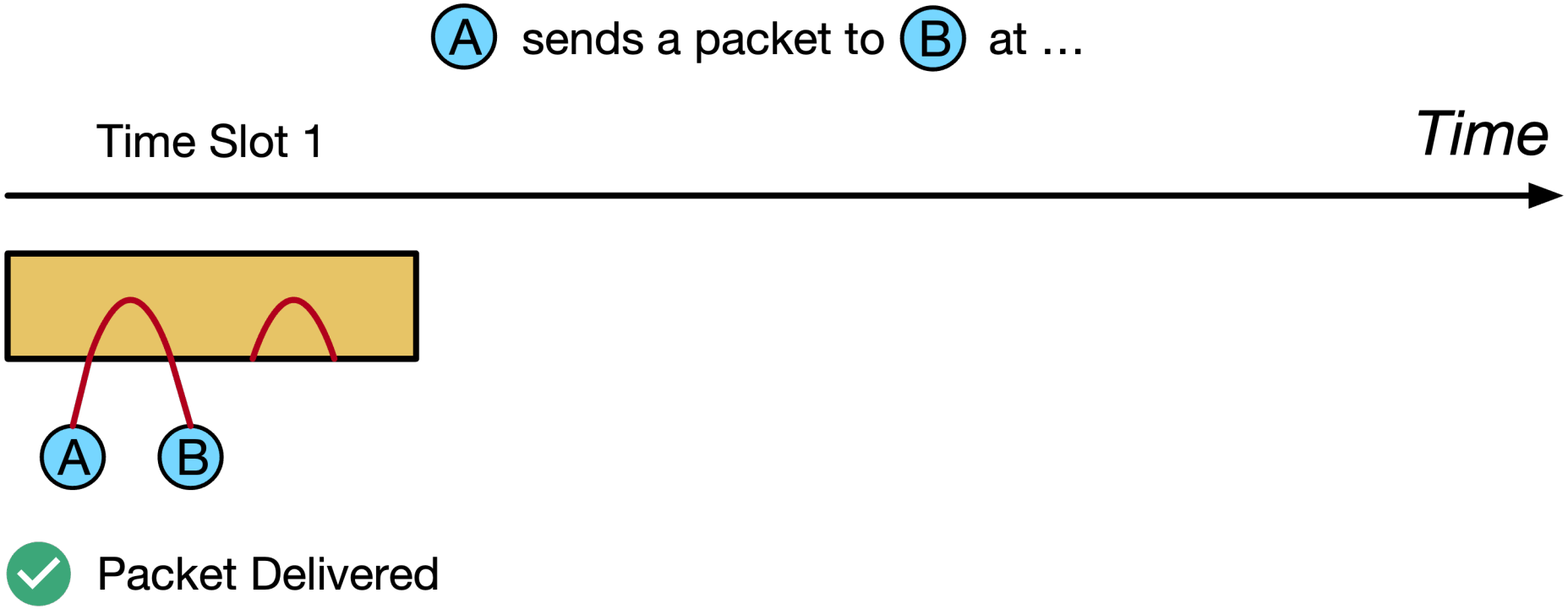


RDCN

Time Sync is a
routing requirement

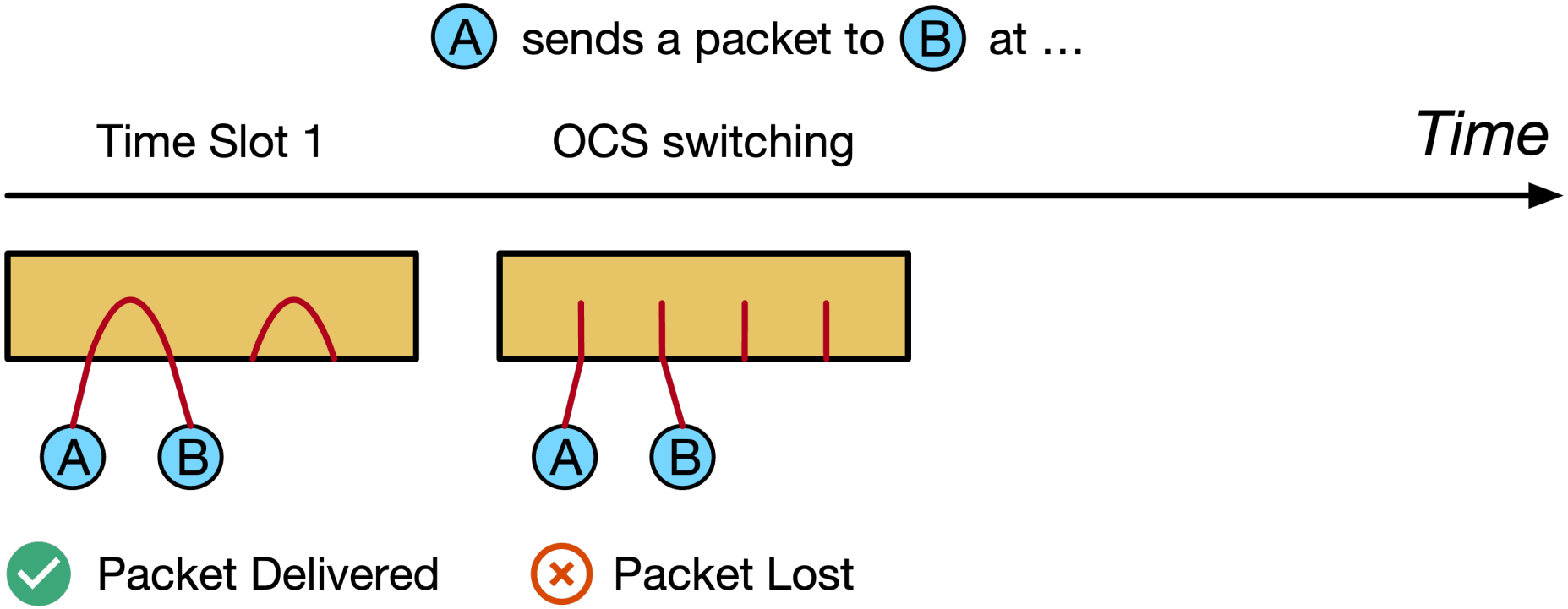
Time Sync is Essential for RDCN

- Circuits appear and vanish; so data MUST land within a circuit window



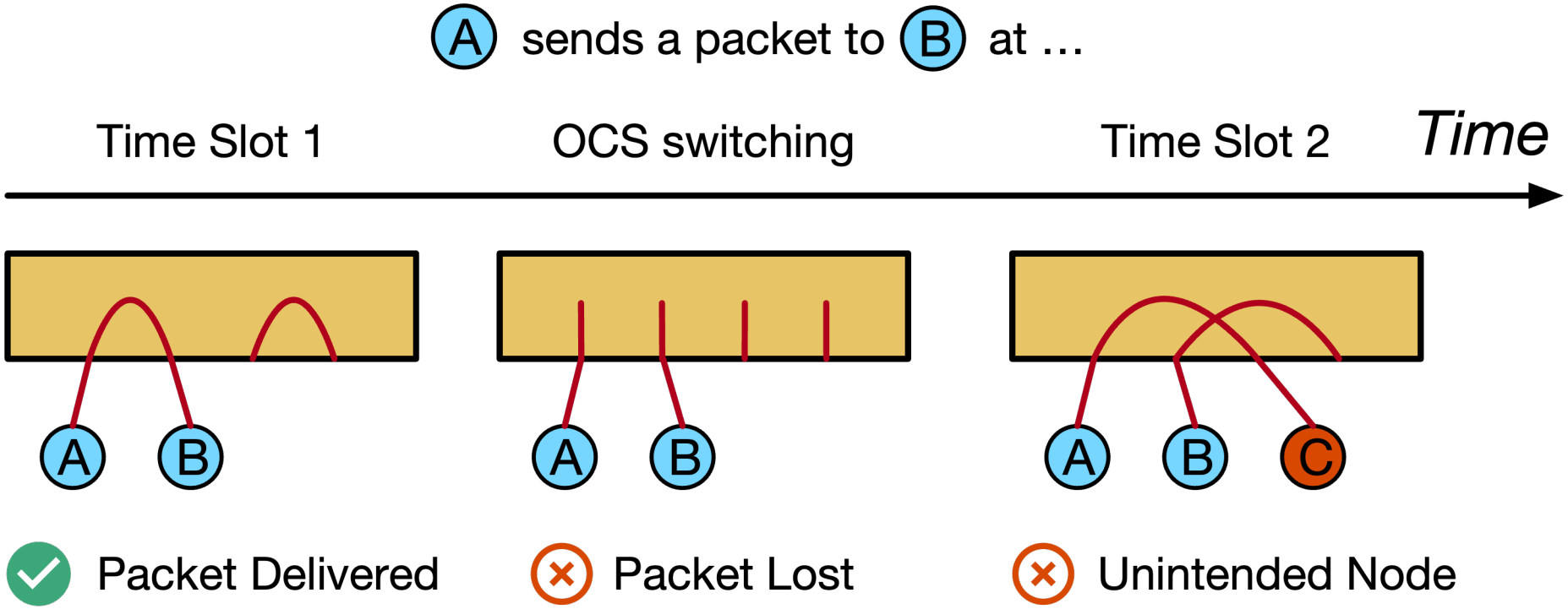
Time Sync is Essential for RDCN

- Circuits appear and vanish; data must land inside the circuit window
- Miss the window → packet dropped



Time Sync is Essential for RDCN

- Circuits appear and vanish; data must land inside the circuit window
- Miss the window → packet dropped or sent to an unintended node

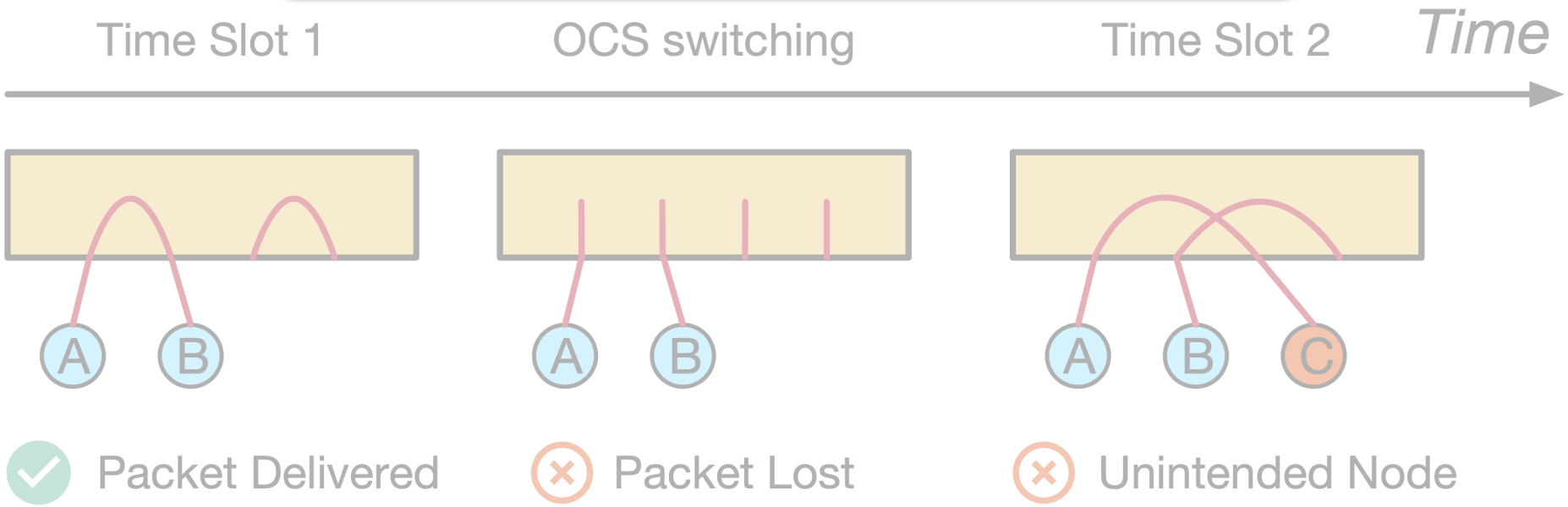


Time Sync is Essential for RDCN

- Circuits app… circuit window
- Miss the wi… ntended node

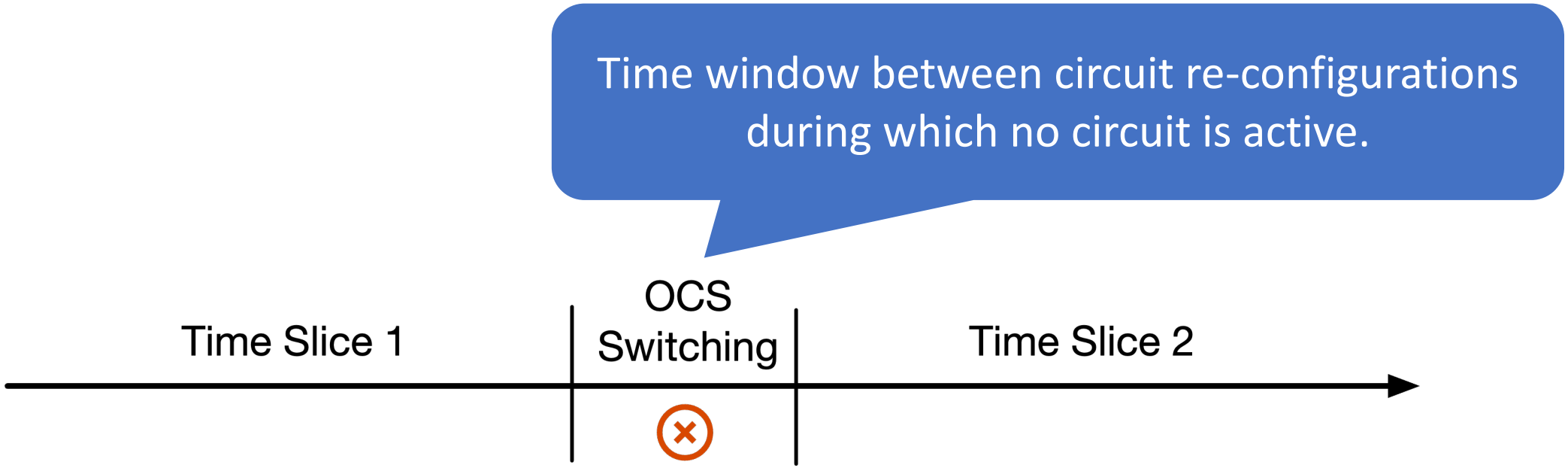
Clock sync is needed!

For nodes to know *when* to send packets for a desired destination.



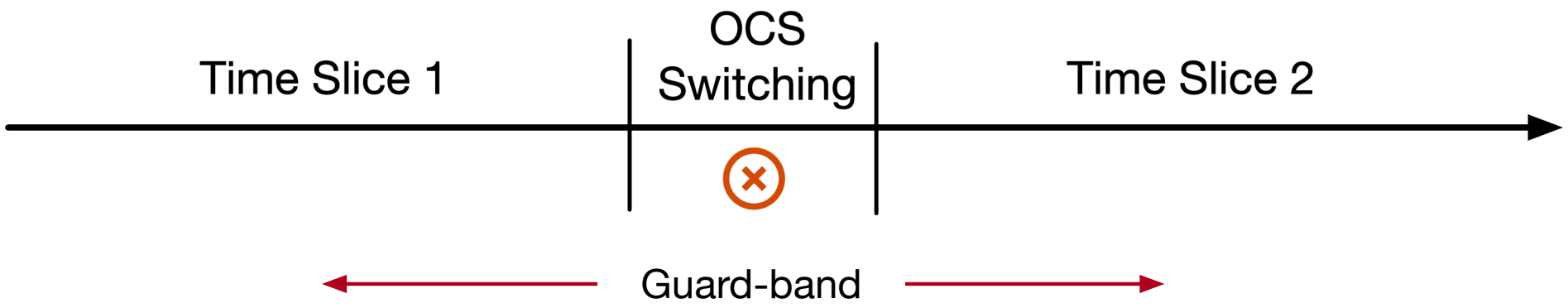
RDCN's Requirement for Time Sync

- Key requirement: Packets must not be sent during OCS switching



RDCN's Requirement for Time Sync

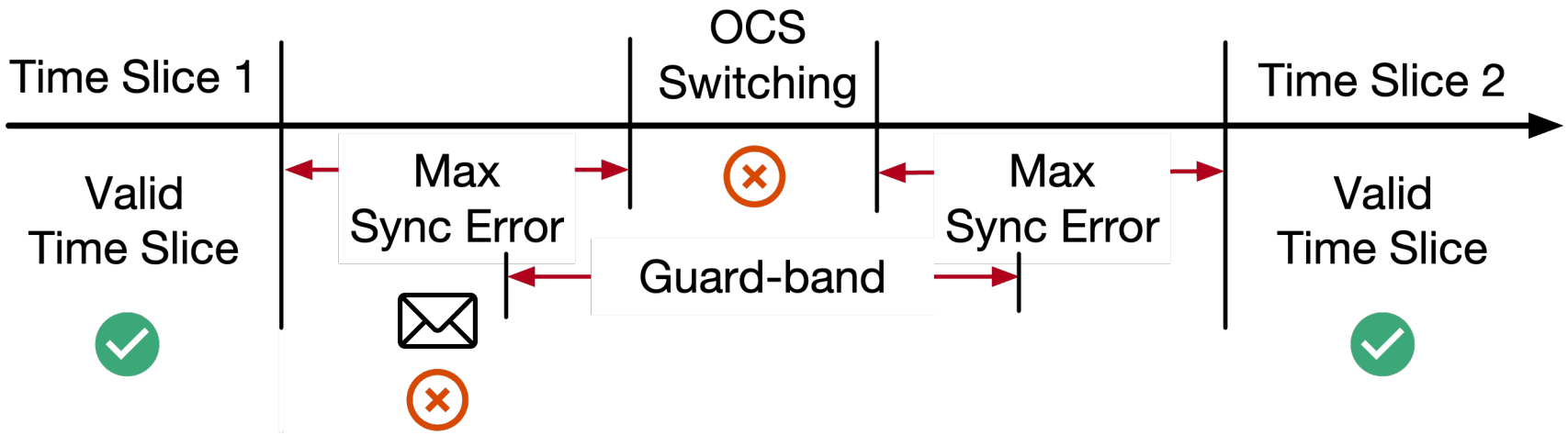
- RDCNs have a "guard-band": prevent pkts being sent at wrong time



RDCN's Requirement for Time Sync

If the guard-band is set aggressively small:

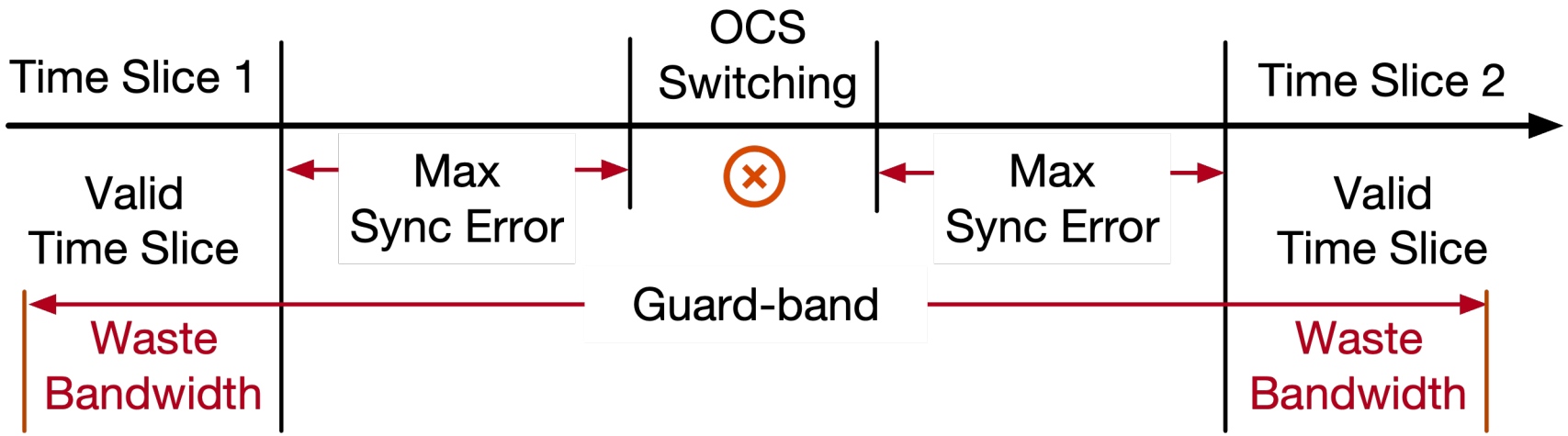
- Case: due to time sync error, pkts sent during OCS Switching. Dropped!



RDCN's Requirement for Time Sync

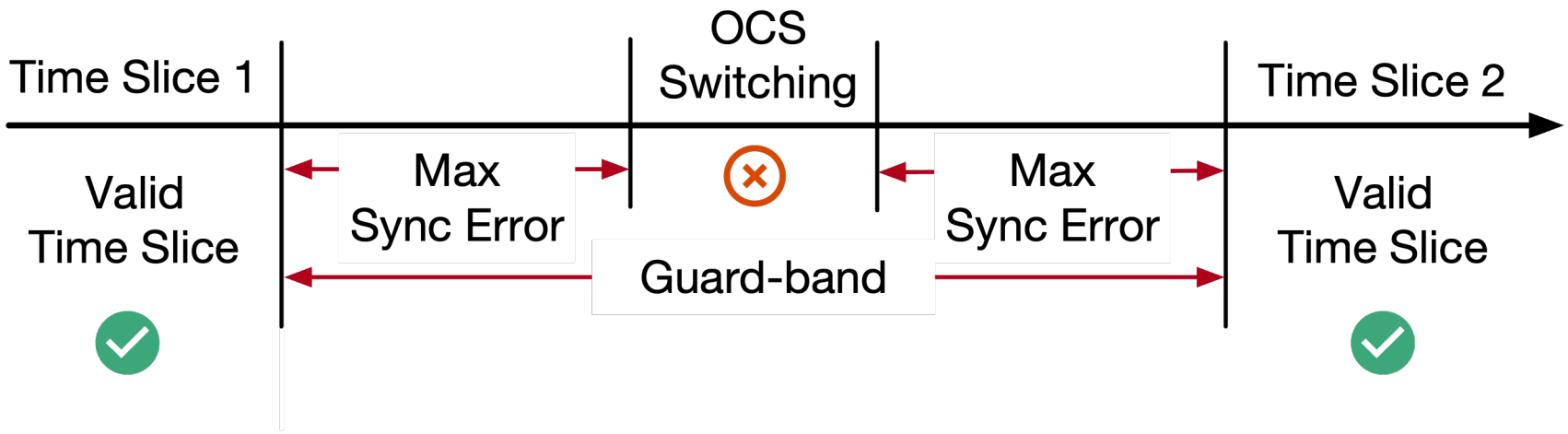
If the guard-band is set conservatively large, leads to bandwidth loss

- e.g. 10% larger than OCS Switching → 10% loss in bandwidth



RDCN's Requirement for Time Sync

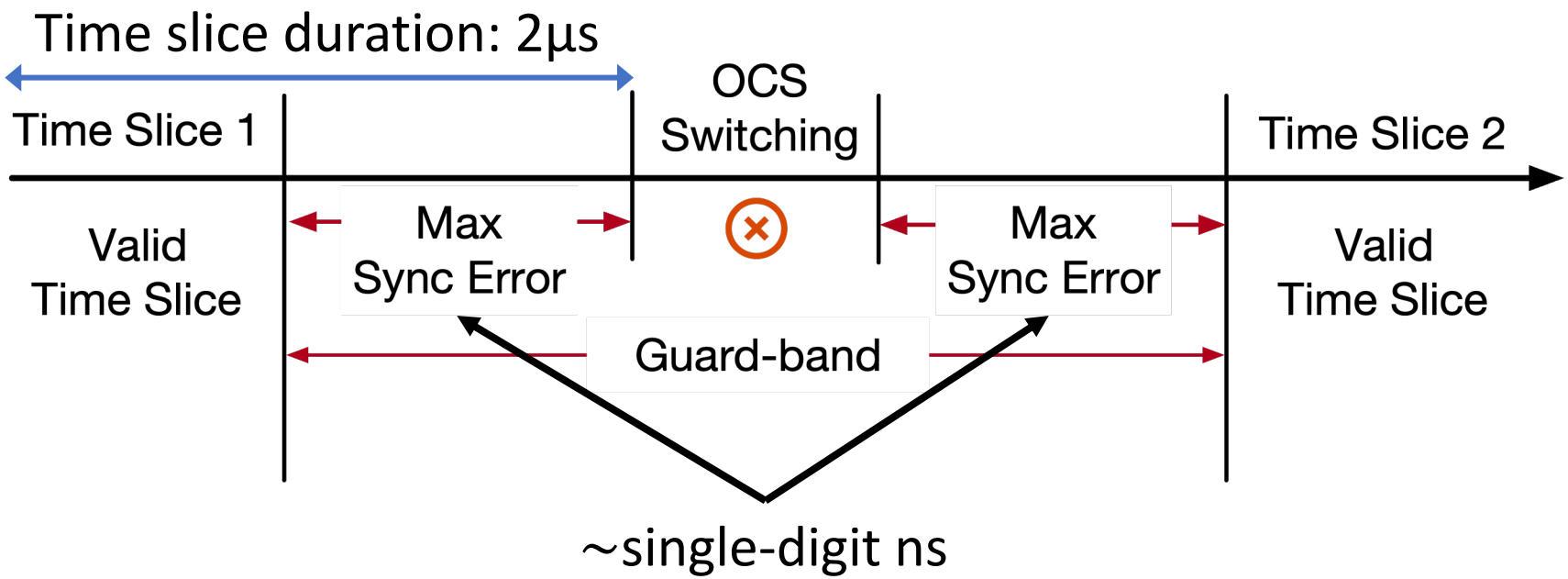
- Right guard-band size = 2 x **worst-case sync error** + OCS switching time
  - Ensures no packet loss during OCS switching
  - Bandwidth loss: min possible; proportional to max sync error



RDCN's Requirement for Time Sync

RDCN design: time slice duration of 2µs.

- 99% network utilization → Max Sync error = ~**single-digit ns range**



RDCN's Requirement for Time Sync

RD

•

ge

Time Sync Requirement for RDCNs

Worst-case sync error must be **bounded**

AND

error bound **within ~single-digit ns range**

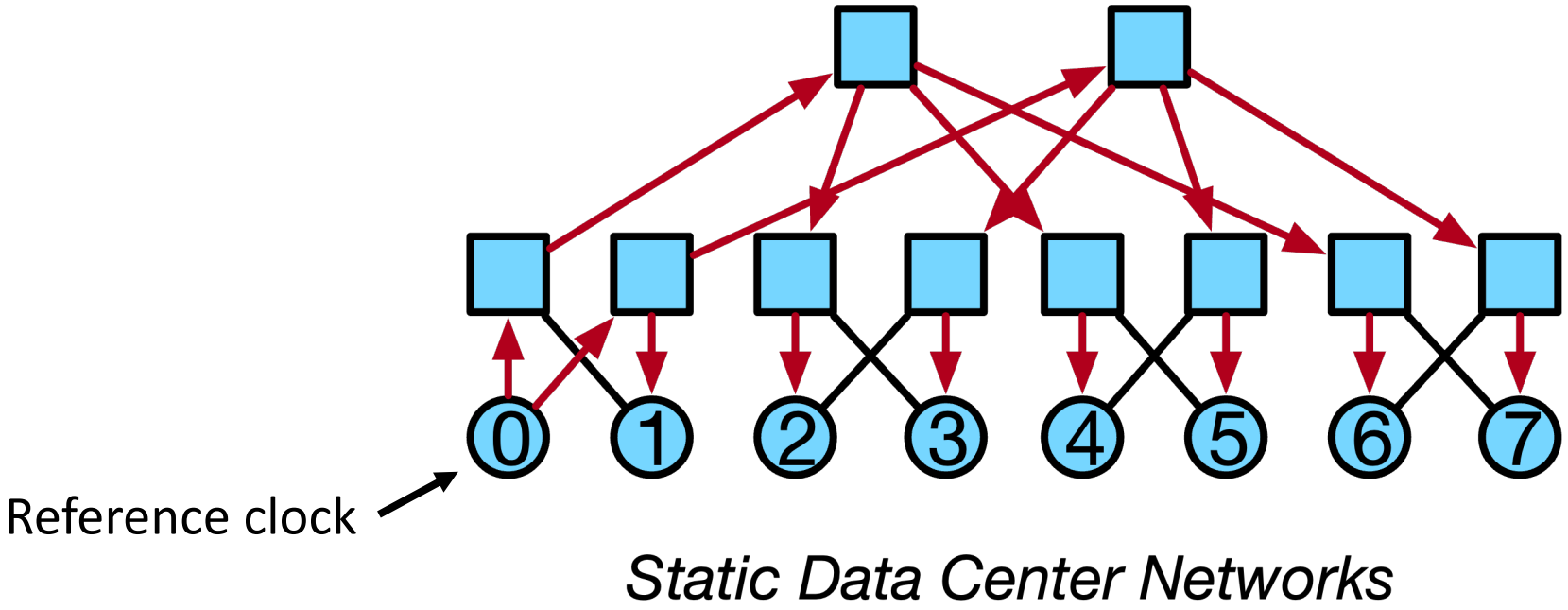
for modern µs-scale circuit switching!

~single-digit ns

Why can’t existing time sync solutions meet these requirements?

Existing Time Sync designed for Static DCNs

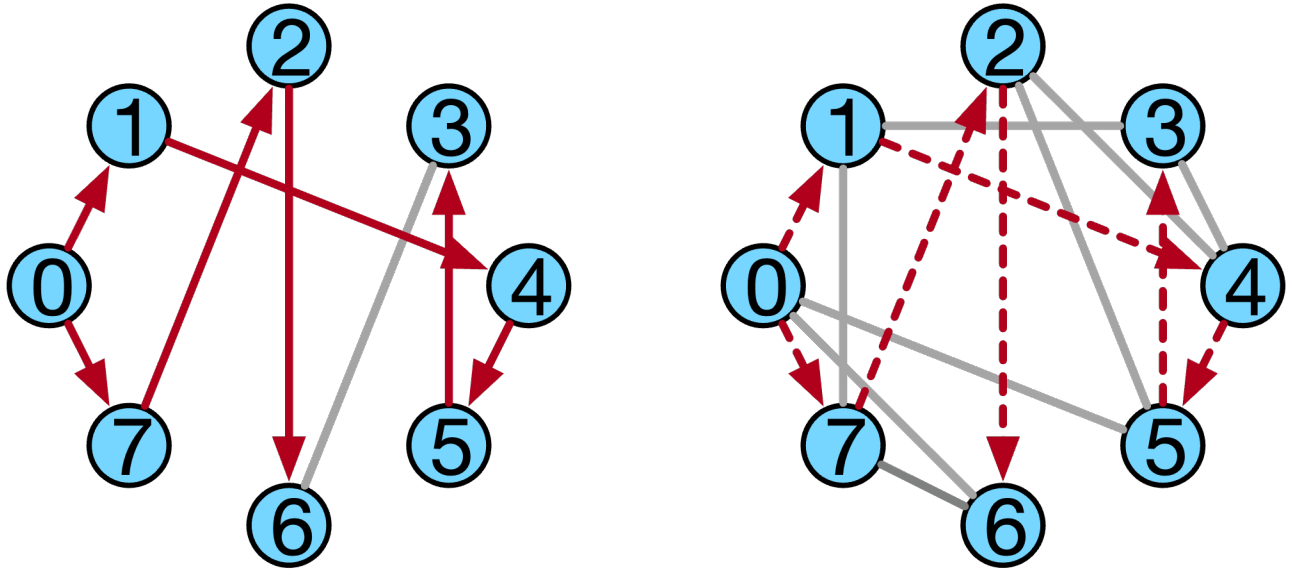
- Rely on a static spanning tree rooted at a reference clock [1, 2]



[1] Precision Time Protocol (PTP), IEEE 1588-2008
[2] Sundial: Fault-tolerant Clock Synchronization for Datacenters, OSDI' 20

Existing Time Sync mainly target Static DCNs

- RDCNs: topology changes rapidly → disrupts clock propagation trees!

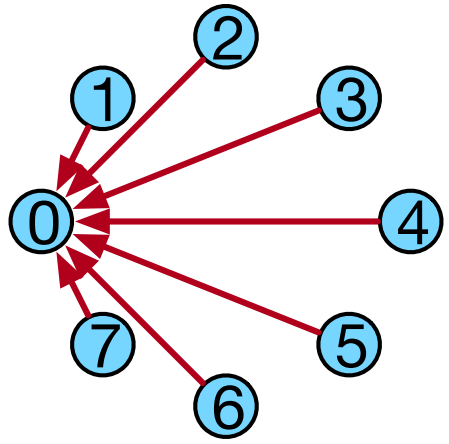


Time Slice 1 *Time Slice 2*

Reconfigurable Data Center Networks

Existing Time Sync mainly target Static DCNs

- Firefly (SIGCOMM ‘25): somewhat robust to spanning tree disruption
  → gossip avgs of peer clocks; not rely on clock propagation trees
  → averaging doesn’t give a tight, worst-case bound on the sync error



Average Peers

[1] Firefly: Scalable, ultra-accurate clock synchronization for datacenters. SIGCOMM 2025

Research Question

How to design a time sync protocol for RDCNs with a ns-scale sync error bound?

Our Answer: Sync Wisely!

Key Insight

Each node can **predict its *current* sync error bound** based pre-profiled drift and hop errors bounds.

Key Idea

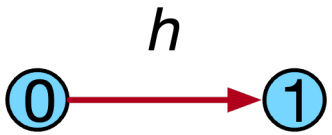
Each node only synchronizes with neighbors whose **predicted error bound is lower** than its own.

Why Sync Error Bound can be predicted?

Sync error analysis

- 2 sources of error: node synchronizes its clock with another node

$$h$$



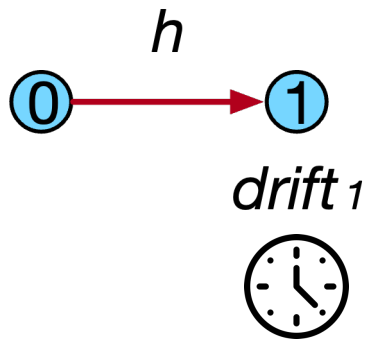
Hop error: noise in each sync operation
(inaccurate timestamping, delay asymmetry)

Why Sync Error Bound can be predicted?

Sync error analysis

- 2 sources of error: node synchronizes its clock with another node

$drift_i$ h



Drift error: caused by freq difference between two clocks' crystals. After each sync op, two clocks drift away as time goes on.

Why Sync Error Bound can be predicted?

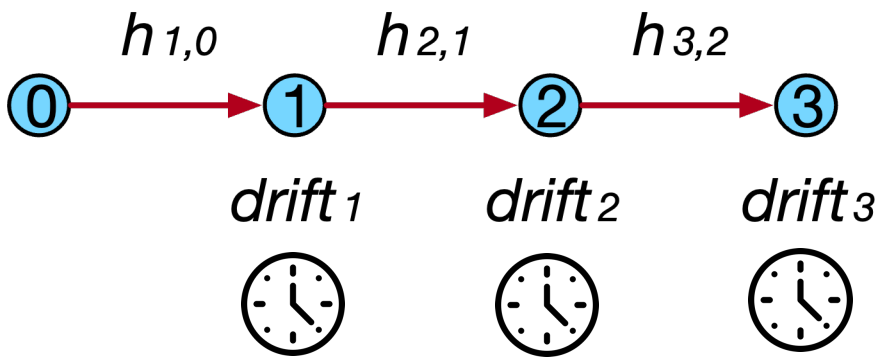
Sync error analysis

- Clock propagated from ref node → formalize error along sync path

$$\text{Error} = \sum_{i=0}^{l} \int_{0}^{\tau_i} drift_i(t)\, dt + \sum_{i=1}^{l} h_{i,i-1}$$

Accumulated effect of clock drift

Sum of multiple hop errors



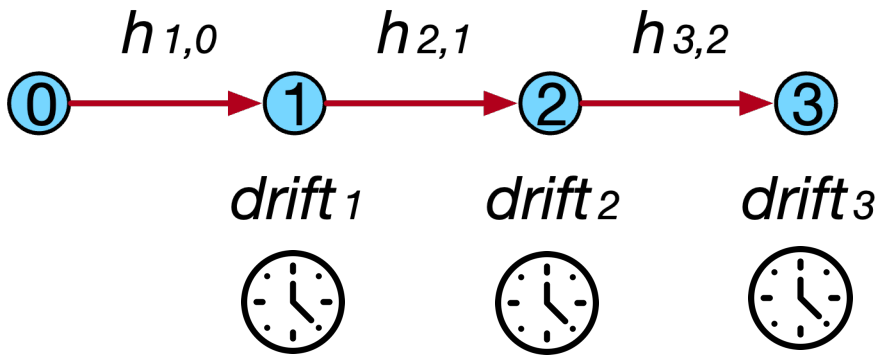
Note: *l* is clock propagation path

Why Sync Error Bound can be predicted?

Sync error analysis

- Eqn involves multiple runtime variables → value is unknown at runtime!

$$\text{Error} = \sum_{i=0}^{l} \int_{0}^{\tau_i} drift_i(t)\, dt + \sum_{i=1}^{l} h_{i,i-1} \leftarrow \textbf{unknown}$$



Note: / is clock propagation path

Why Sync Error Bound can be predicted?

Sync error analysis

$$\text{Error} = \sum_{i=0}^{l} \int_{0}^{\tau_i} drift_i(t)\, dt + \sum_{i=1}^{l} h_{i,i-1} \leftarrow \textbf{unknown}$$

Maximal

$$\text{Error Bound} = \sum_{i=0}^{l} V_i\, \tau_i + H \times l \leftarrow \textbf{Predictable!}$$

Note: *l* is clock propagation path

Why Sync Error Bound can be predicted?

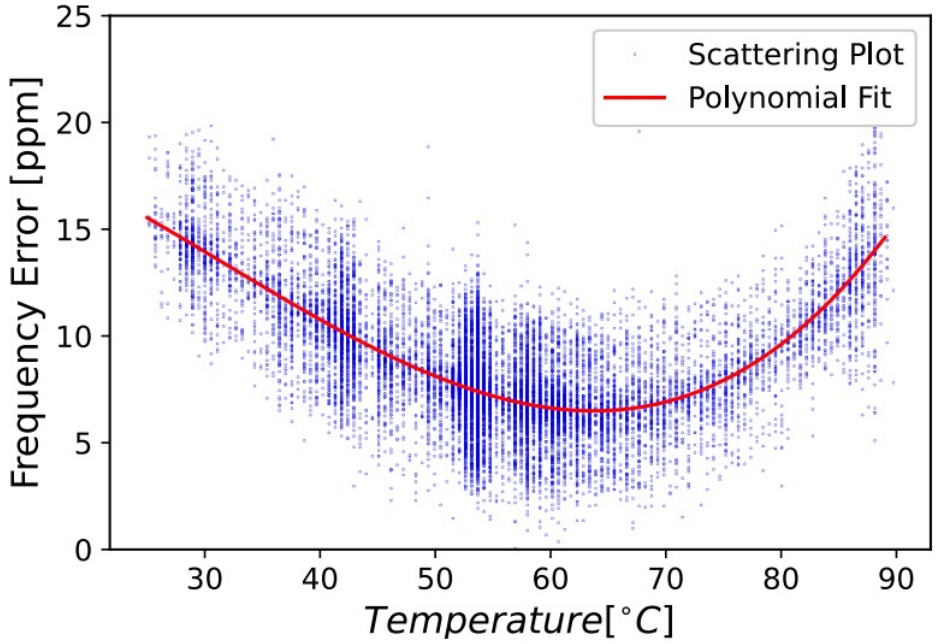


Figure from ”Graham: Synchronizing Clocks by Leveraging Local Clock Properties”, NSDI’22

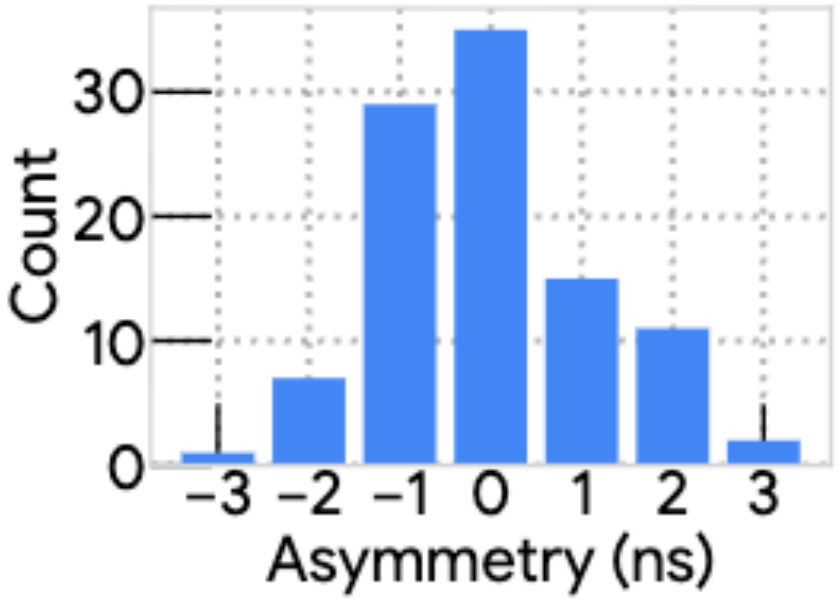


Figure from ”Firefly: Synchronizing Clocks by Leveraging Local Clock Properties”, SIGCOMM 25

Drift: not completely unpredictable.

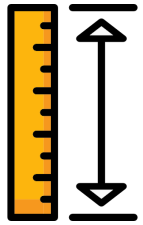
→ influenced by temp, small runtime variance

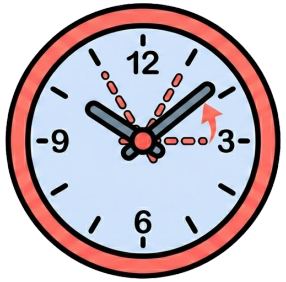
Hop error: follows a distribution

but has a clear range!

Nodes can track their own error bound

Before the RDCN starts operating, we can **Pre-Profile**





Each node's **maximum drift variance (*V*)**

Network-wide **bound on hop error (H)**

Nodes can track their own error bound

Then, during runtime, each node can track the sync error bound:

- As time passes, sync error bound increases with the **drift variance**

$$\text{Error Bound} = \sum_{i=0}^{l} V_i \, \tau_i + H \times l$$

Nodes can track their own error bound

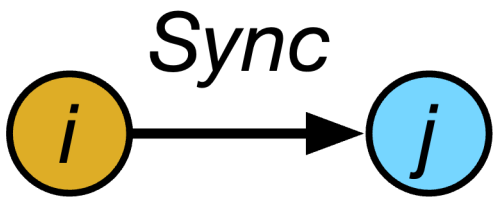
Then, during runtime, each node can track the sync error bound:

- In each sync op, error bound increases by 1 more maximal hop error.

$$\text{Error Bound} = \sum_{i=0}^{l} V_i \, \tau_i + H \times l$$

Error-Aware Synchronization

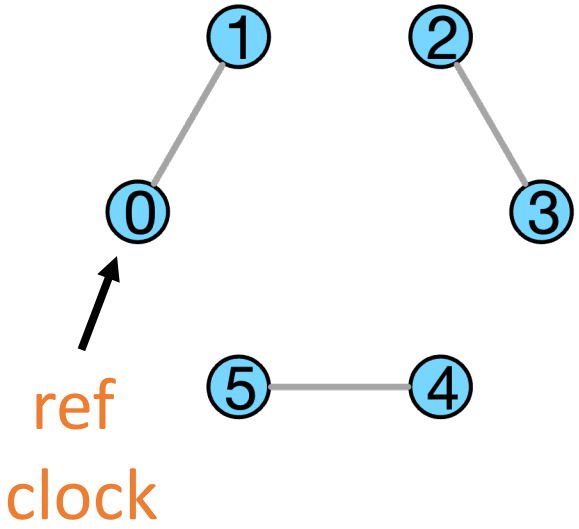
- Nodes keep track of their error bounds
- Only sync with nodes which can reduce its error bound – **Sync Wisely**



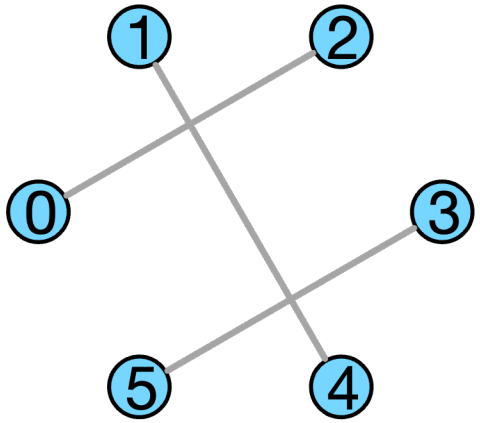
$$\text{if } Bound_i + H < Bound_j :$$
$$clock_j \leftarrow clock_i$$
$$Bound_j \leftarrow Bound_i + H$$

Error-Aware Synchronization

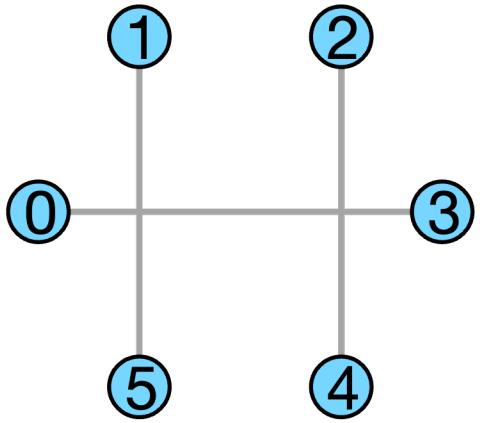
RDCN: 6 nodes



Time Slice 1



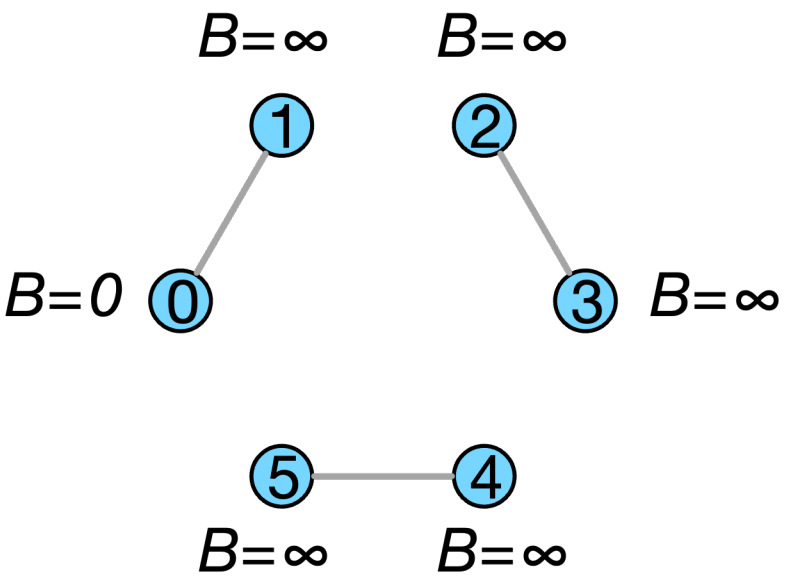
Time Slice 2



Time Slice 3

Hop Error Bound = 5 — *Optical Links* → *Clock Propagation*

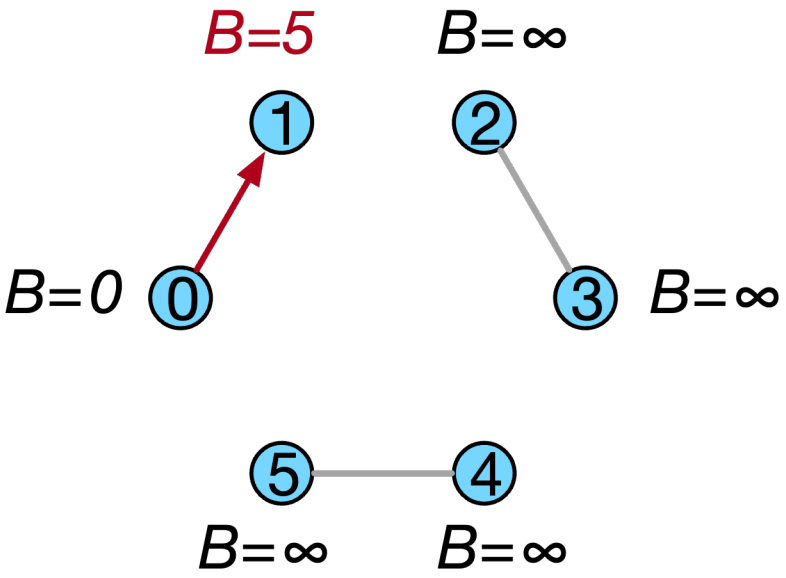
Error-Aware Synchronization



Time Slice 1

Hop Error Bound = 5 — *Optical Links* → *Clock Propagation*

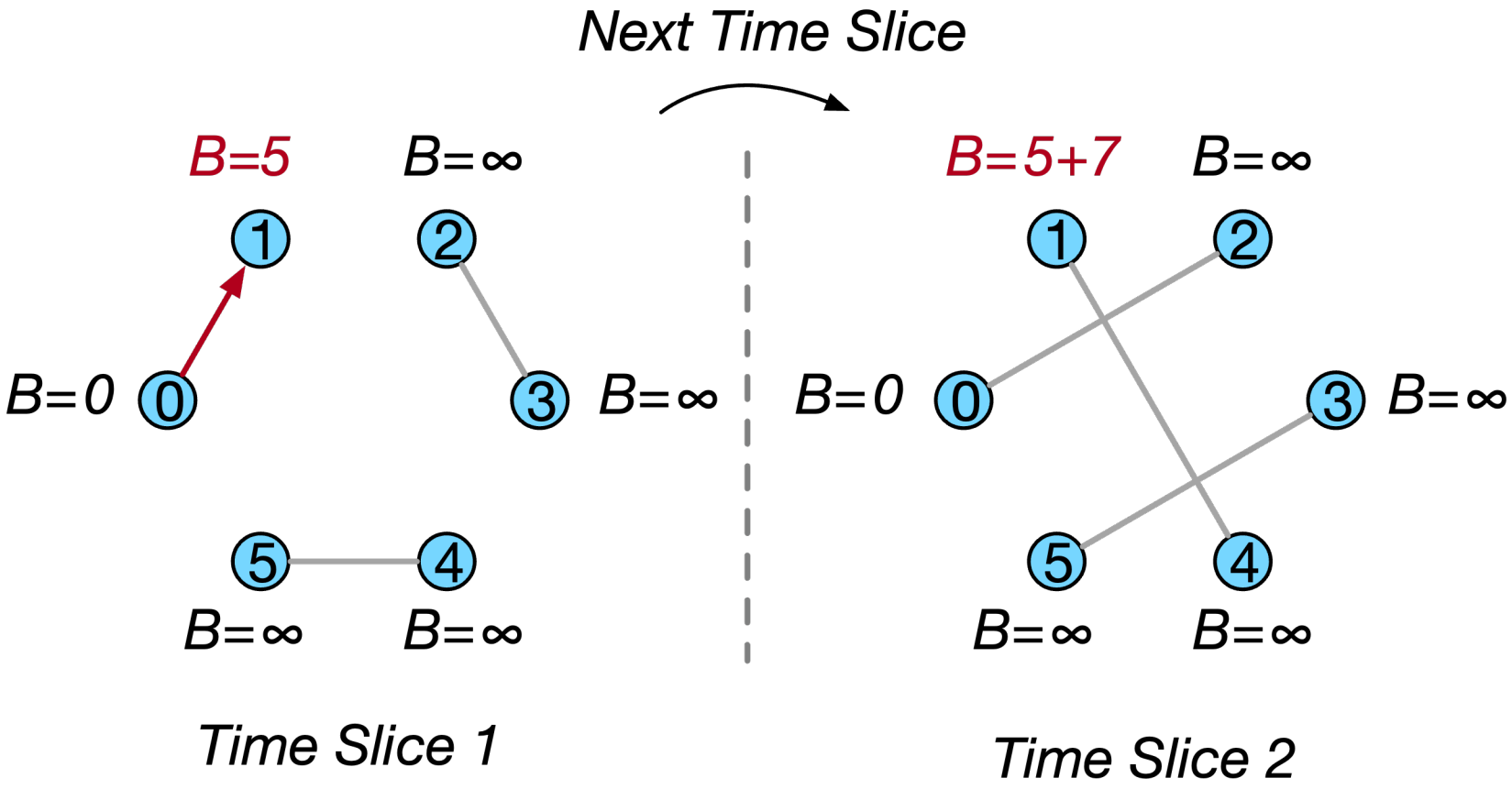
Error-Aware Synchronization



Time Slice 1

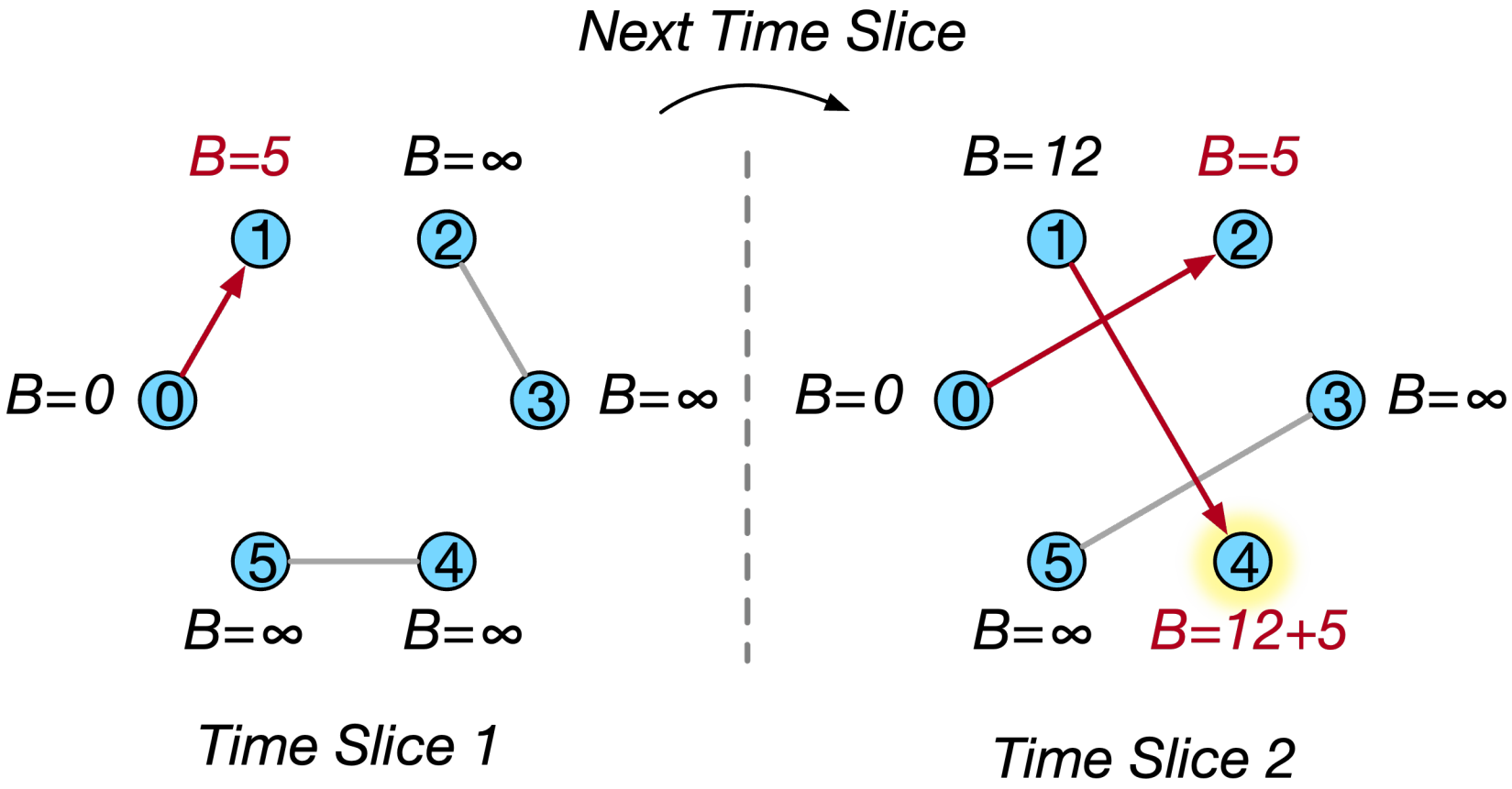
Hop Error Bound = 5 — *Optical Links* → *Clock Propagation*

Error-Aware Synchronization



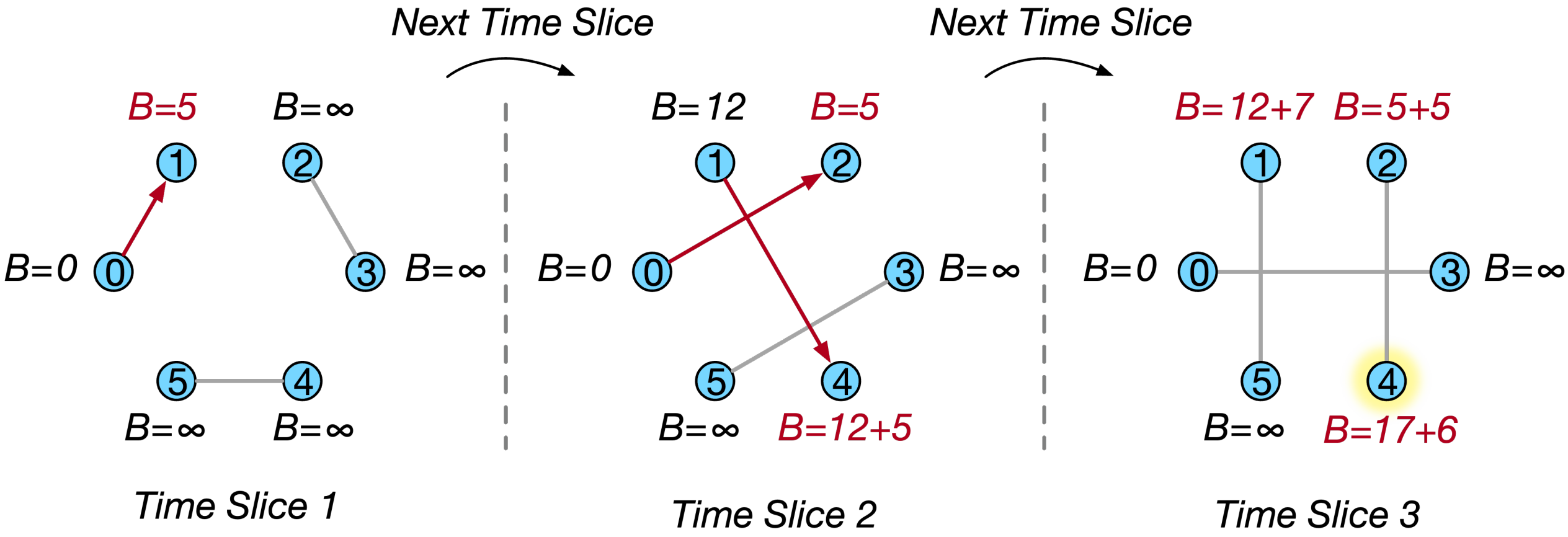
Node ID	0	1	2	3	4	5
Drift Variance (ns/slice)	0	7	5	12	6	10

Error-Aware Synchronization



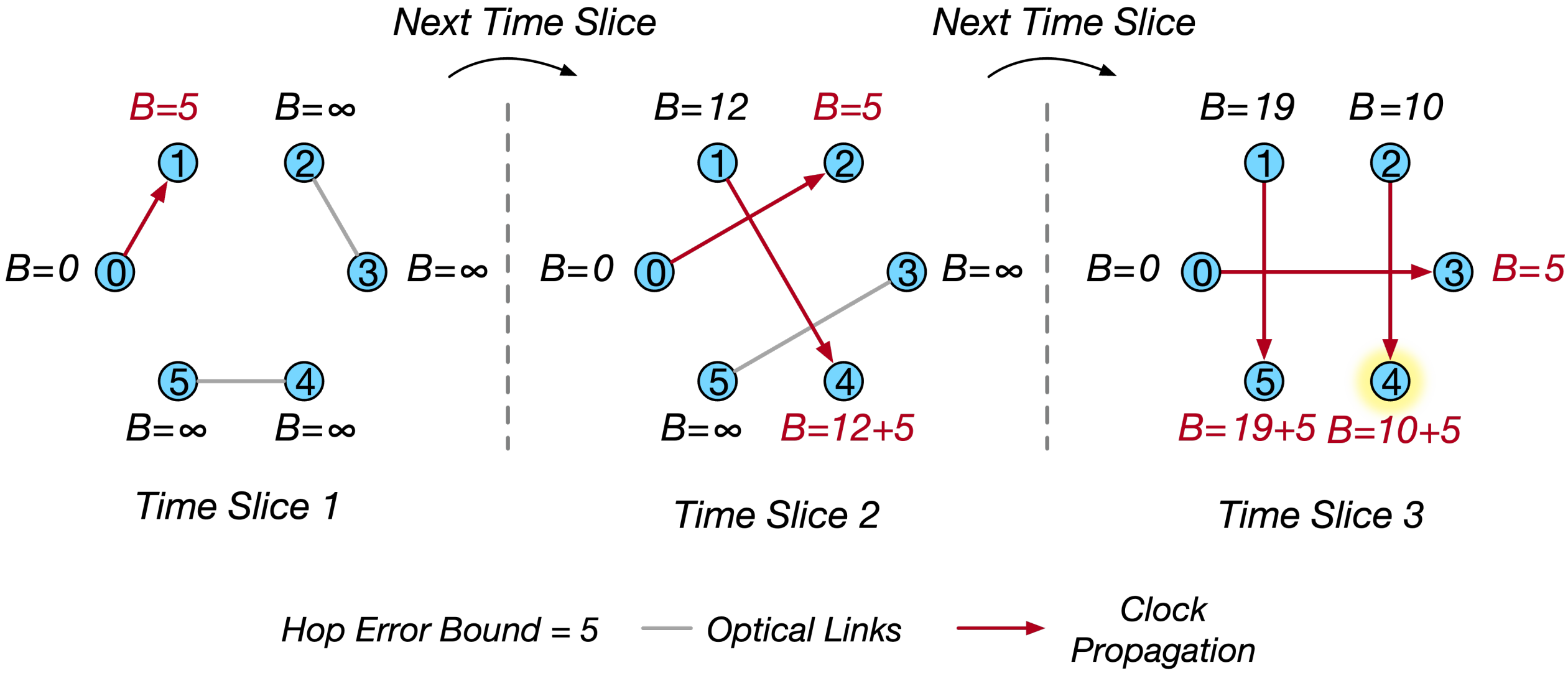
Hop Error Bound = 5 — Optical Links → Clock Propagation

Error-Aware Synchronization



Node ID	0	1	2	3	4	5
Drift Variance (ns/slice)	0	7	5	12	6	10

Error-Aware Synchronization



Error-Aware Synchronization

Two Key Properties

For a given circuit schedule → Achieves the minimal possible sync error bound over all parent-selection choices.

With a periodic RDCN topology, this error bound is itself periodic and fully pre-computable offline.

→ Guard band computable a priori

(See paper for full proof)

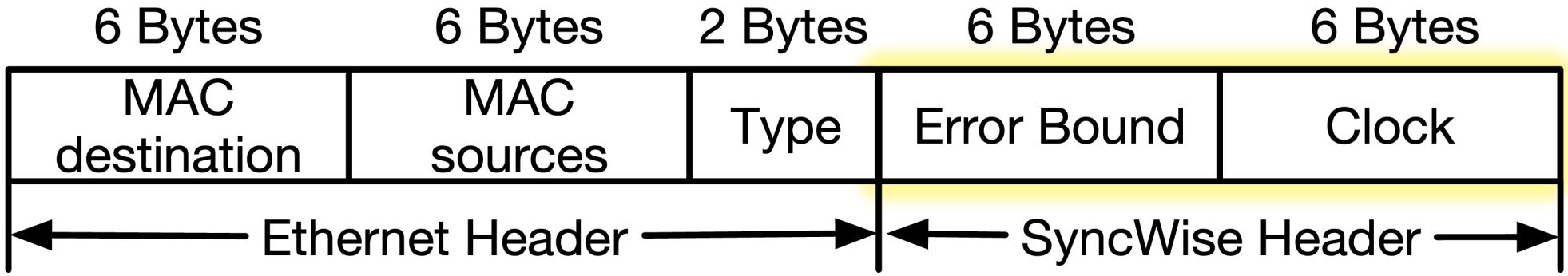
Error-Aware Synchronization

Implication of these 2 properties

RDCN designers can determine the tight synchronization floor for their fabric *before* choosing guard-bands.

Implementation

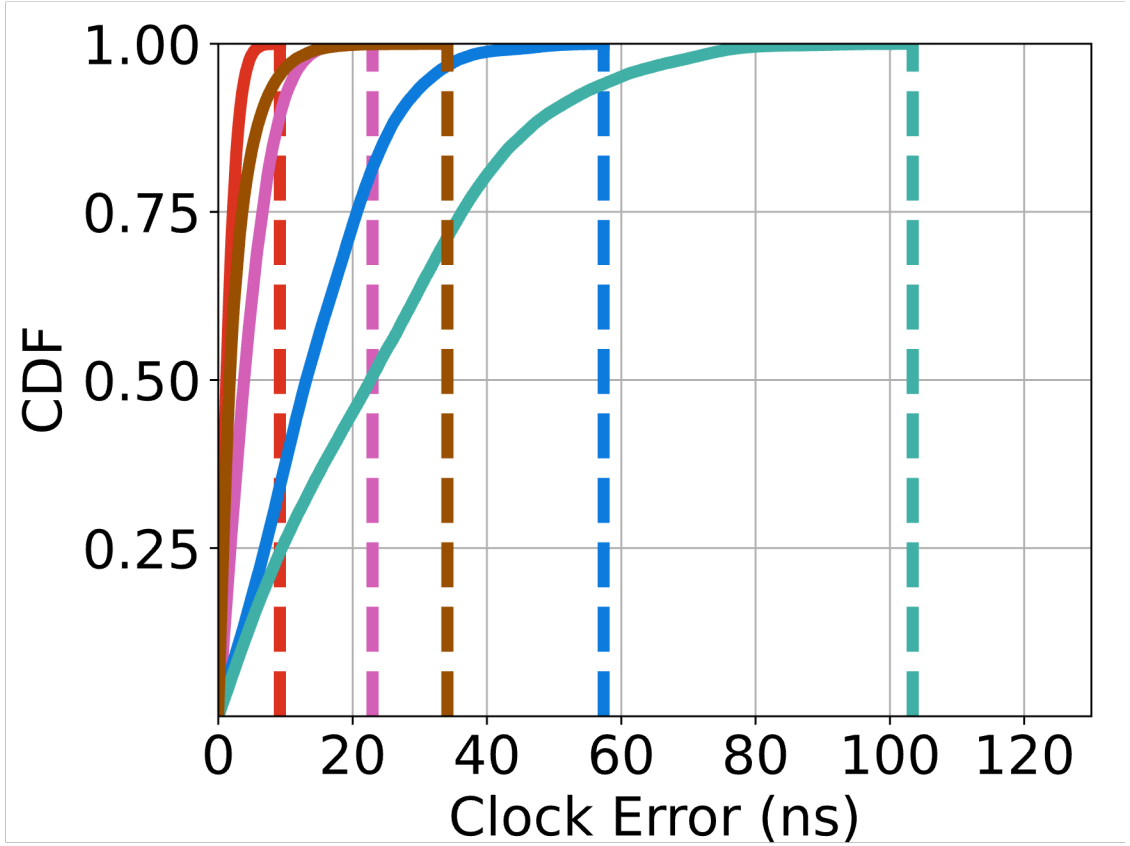
- Error-aware sync: distributed network protocol at any layer.
- Prototype: Ethernet protocol using Tofino2 programmable switches



- Event-driven simulator: extensive evaluation and comparison
  - Open sourced: https://github.com/mpi-ncs/SyncWise

Key results from simulation

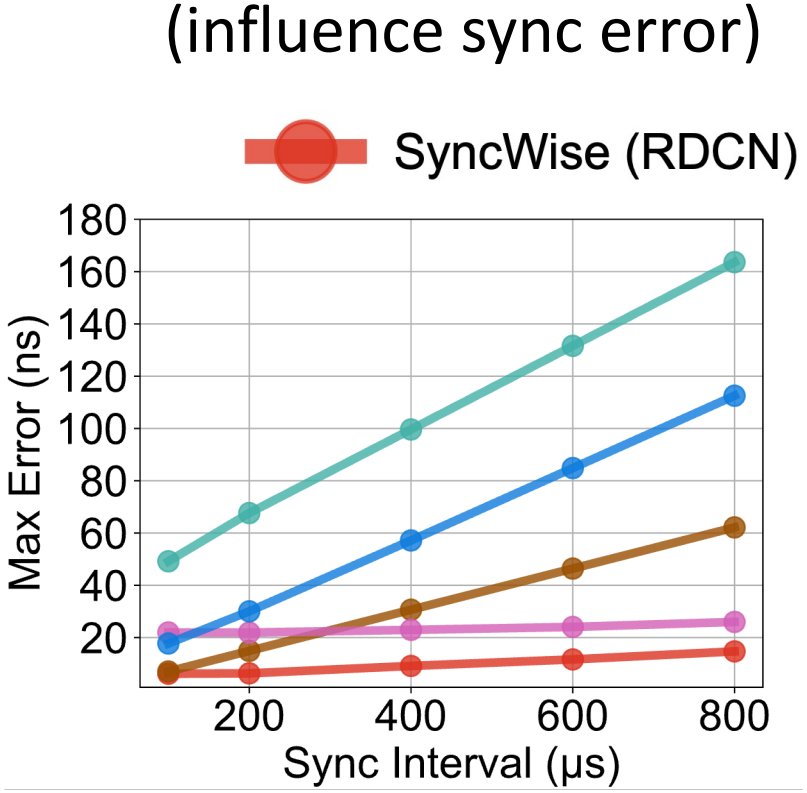
- Simulation: 108 ToR switches, 6 optical uplinks, round-robin circuit
- Baselines: Sync protocols for **static** DCNs (adapt to RDCN)

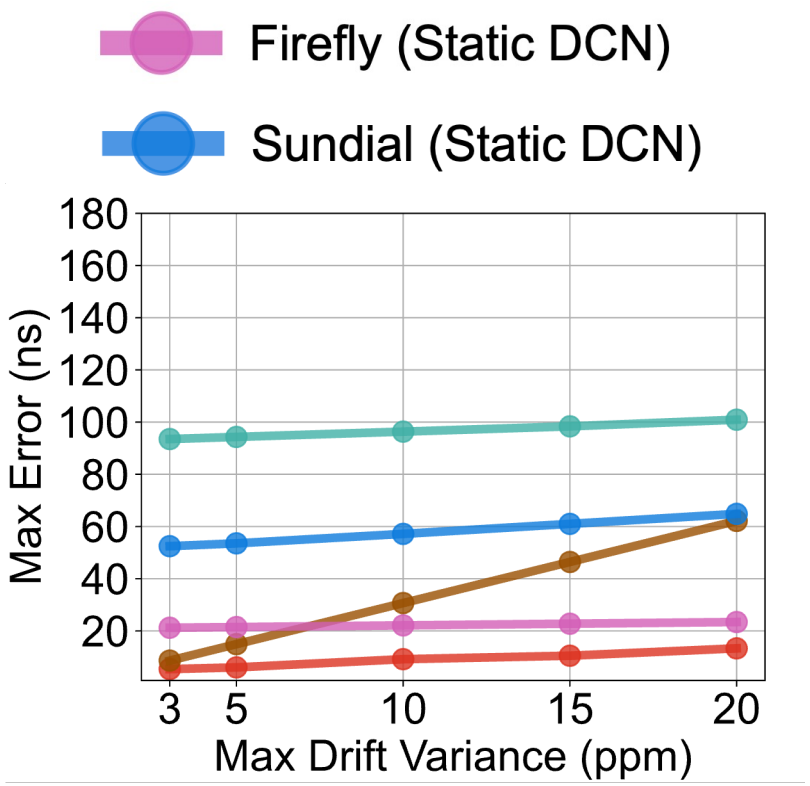


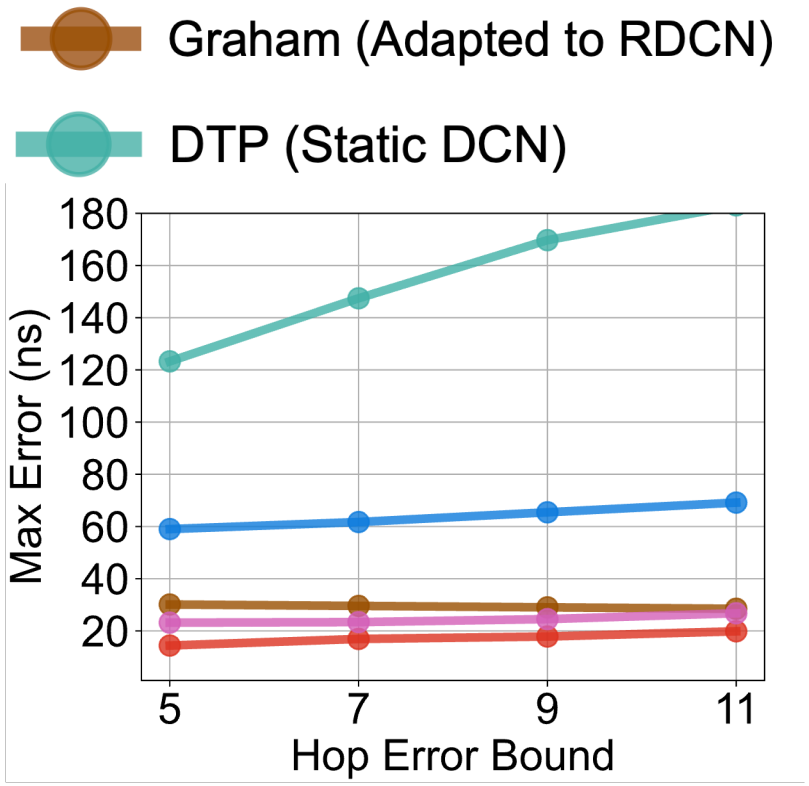
	Max Error	
SyncWise (RDCN)	9ns	
Firefly (Static DCN)	23ns	2.5X
Graham (Adapted to RDCN)	34ns	3.8X
Sundial (Static DCN)	57ns	6.3X
DTP (Static DCN)	103ns	11.4X

Key results from simulation

- Stress test SyncWise under diverse setups
- Varying 3 parameters: Sync interval, Max Drift Variance, Hop Error Bound (influence sync error)







Summary

Bounded time sync is an essential building block for RDCNs

Nodes can predict their sync error bounds

→ error-aware sync to achieve provably tight bounds.

Error-aware sync, a general principle applicable beyond RDCNs

Thank you!

Credits: This presentation template was adapted from Slidesgo, including icons by Flaticon.

Flaticon icons by: Candy Design, Creaticca Creative Agency, Freepik, IconBaandar, manshagraphics, PerfectPixel, pojok d, popcornarts, VectorPortal, and zero_wing.