NegotiaToR: Towards A Simple Yet Effective On-demand Reconfigurable Datacenter Network

Cong Liang[1], Xiangli Song[1], Jing Cheng[1], Mowei Wang[2], Yashe Liu[2], Zhenhua Liu[2], Shizhen Zhao[3], Yong Cui[1]

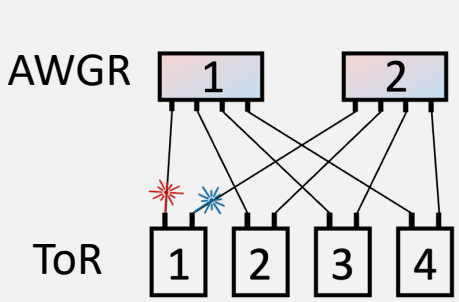
[1]Tsinghua University, [2]Huawei Technologies Co., Ltd, [3]Shanghai Jiao Tong University

Background: Optical switching in DCN

- **Electrical packet switching:** Capacity limitation in the post Moore's Law era
- **Optical switching:** High capacity & Low cost

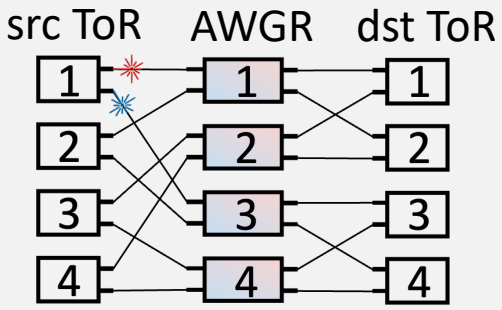
[1] Ballani et al., SIGCOMM '20

  - Recent advance of AWGR-based switching hardware
    - ✓ **Low end-to-end reconfiguration delay:** Within 10 ns[1] *Sufficient for fast reaction to dynamic traffic patterns among ToRs*
    - ✓ **DCN scalability:** Using flat topologies above ToRs



Parallel network topology

(High port-count AWGRs)



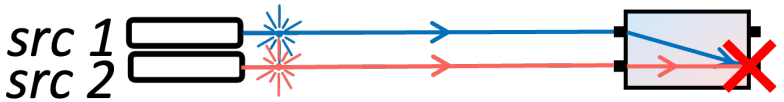
Thin-clos topology

(Low port-count AWGRs)

But how to schedule the fast-reconfigurable optical links?

Scheduling the fast-reconfigurable optical links among ToRs

- **Goal:** Rapidly getting non-conflicting paths



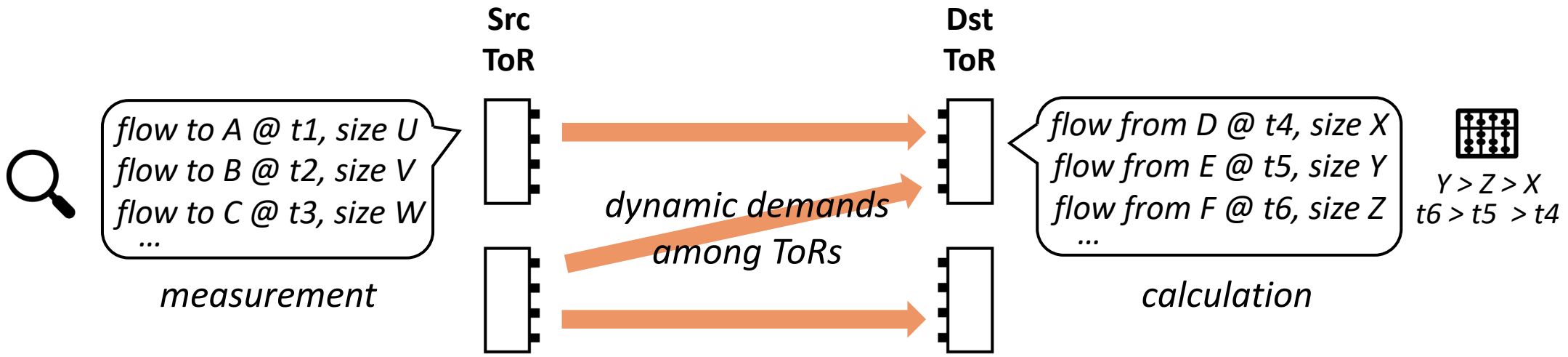
- **Traffic-oblivious:** Predefined uniform round-robin & Data relay (e.g., Sirius[1])
  - **Practical** scheduler-less design
  - Topology & traffic mismatch: Sacrificed latency and goodput

[1] Ballani et al., SIGCOMM '20

- **On-demand:** Adapting topology to real-time traffic demands
  - Potentially better performance
  - Practicality concerns: Fast on-demand *distributed* scheduling?

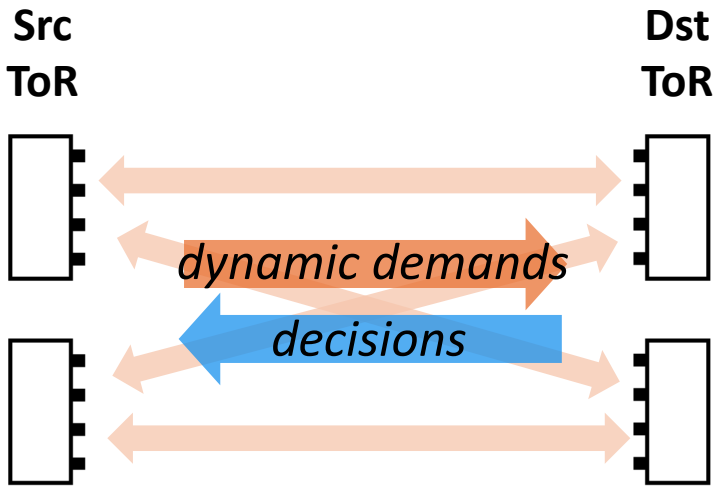
Practical yet high performance on-demand distributed scheduling?

Challenges: On-demand distributed scheduling of fast-reconfigurable optical links among ToRs



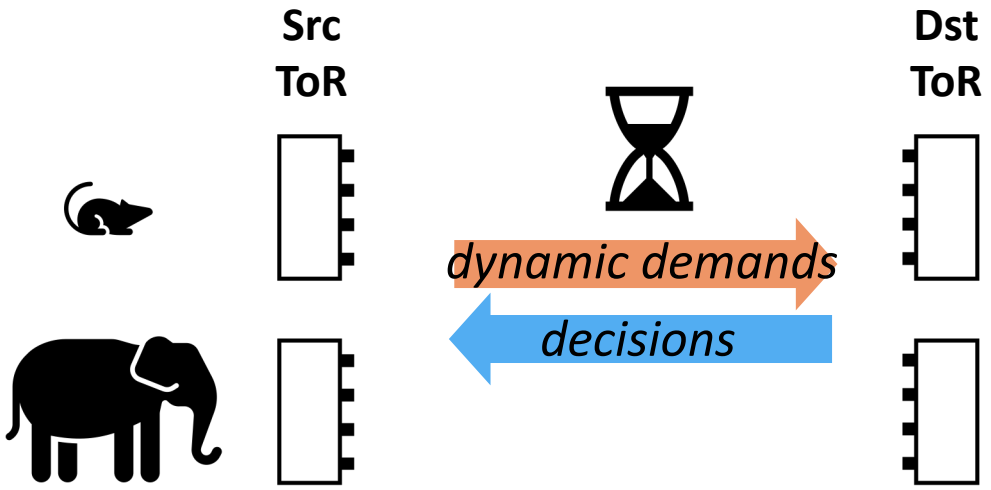
- Demand measurement + Fast calculation at DCN scale?
  - *Challenge 1:* High implementation complexity

Challenges: On-demand distributed scheduling of fast-reconfigurable optical links among ToRs



- Demand measurement + Fast calculation at DCN scale?
  - *Challenge 1:* High implementation complexity
- Separate network for scheduling messages?
  - *Challenge 2:* Expensive to build and maintain a separate control plane

Challenges: On-demand distributed scheduling of fast-reconfigurable optical links among ToRs



- Demand measurement + Fast calculation at DCN scale?
  - *Challenge 1:* High implementation complexity
- Separate network for scheduling messages?
  - *Challenge 2:* Expensive to build and maintain a separate control plane
- Scheduling delay vs. Mice flows / Incasts?
  - *Challenge 3:* Downgraded latency due to scheduling delay

NegotiaToR overview

An on-demand reconfigurable DCN architecture
towards a simple yet effective design

Challenge 1
High implementation complexity

NegotiaToR Matching algorithm
Towards minimalist on-demand scheduling on flat topologies

Challenge 2
Expensive to build and maintain a separate control plane

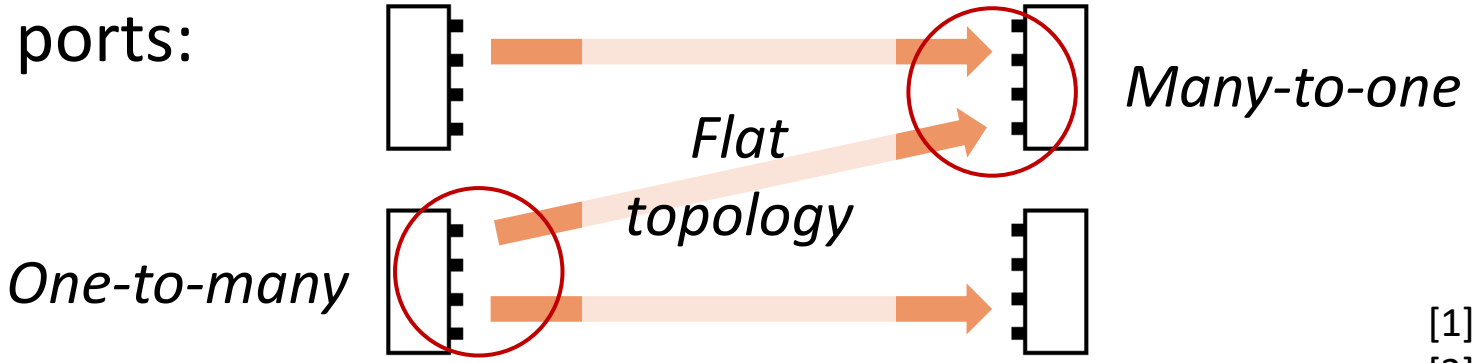
Two-phase epoch
In-band pipelined scheduling & One-hop direct transmission

Challenge 3
Downgraded latency due to scheduling delay

Scheduling delay bypassing
FCT optimization for latency-sensitive flows

Scheduling optical links among ToRs on flat topologies

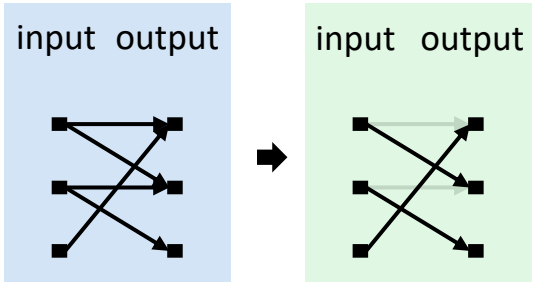
- Conflicts at ports:



A matching problem!

[1] Anderson et al., ToCS ’93
[2] McKeown, PhD Thesis, UC Berkeley ’95
[3] McKeown et al., ToN ’99

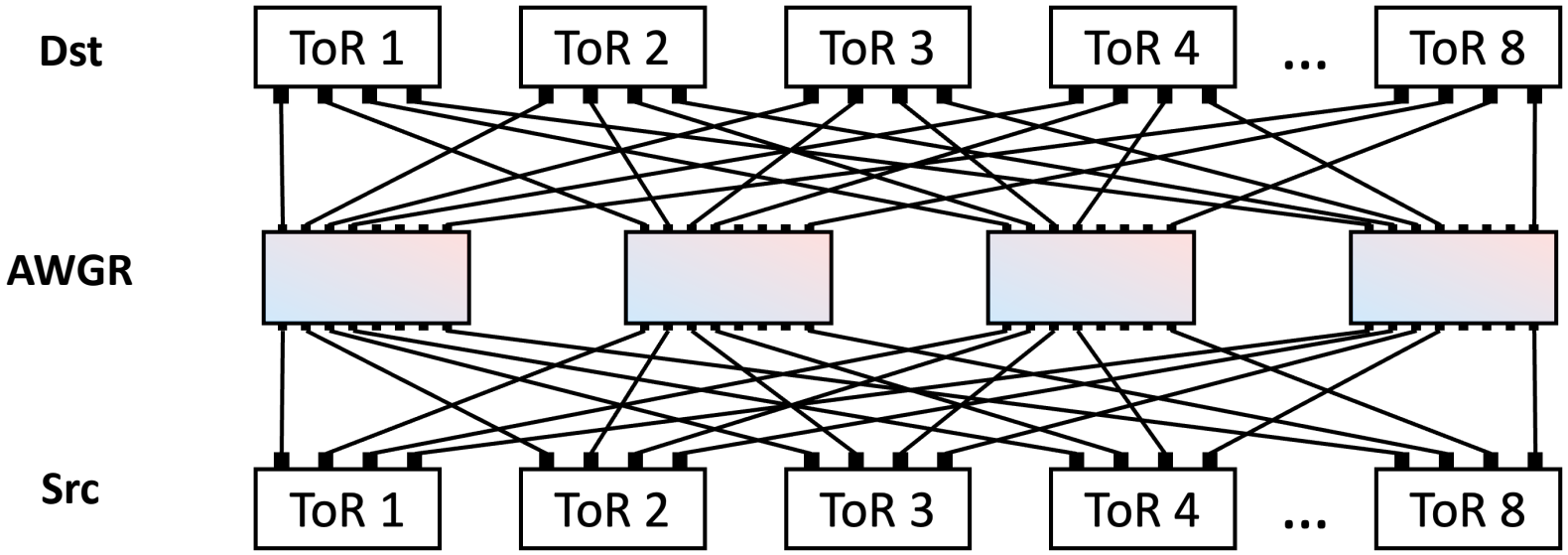
- Similar problems exist in **crossbar packet switches**
  - Non-conflicting input-output matches: PIM[1], RRM[2], iSLIP[3]…
    - ✓ Fast and practical
    - ✓ Extensive implementation experience



NegotiaToR Matching (inspired by RRM[2])

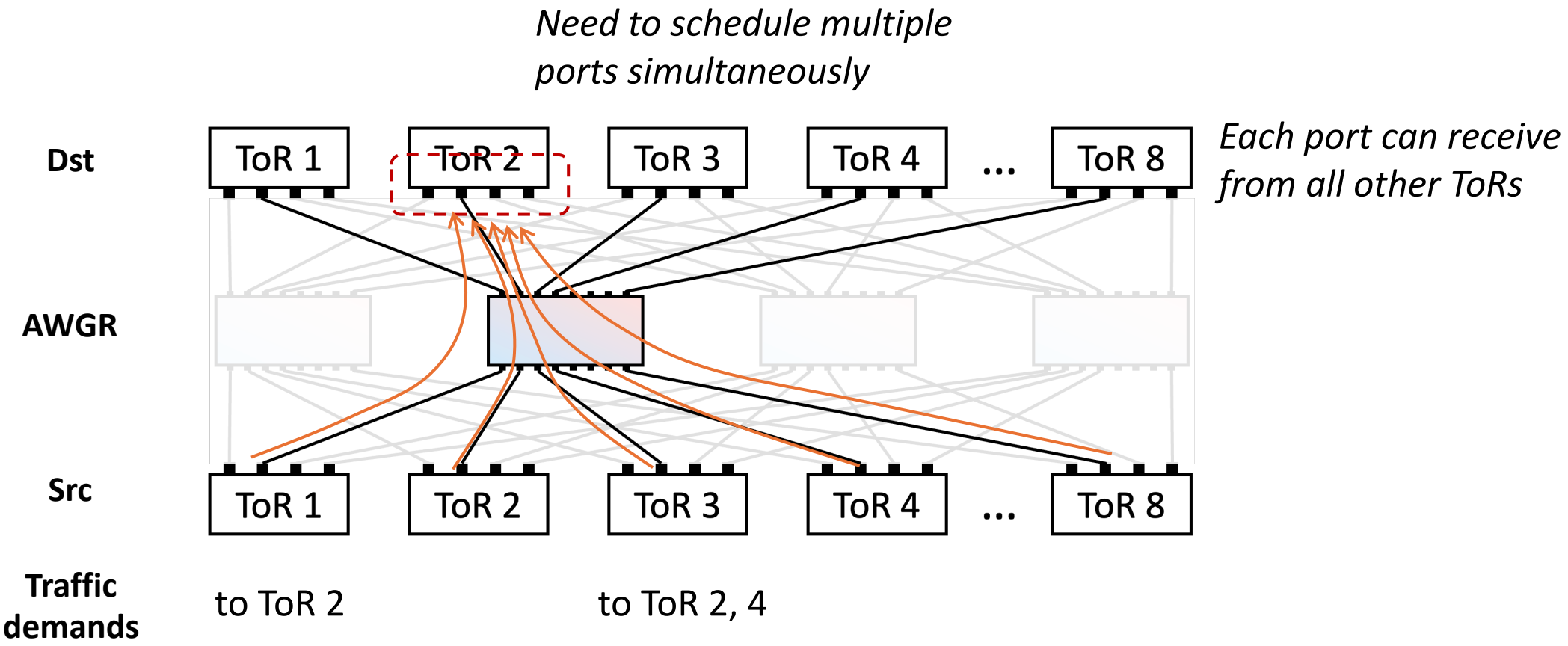
NegotiaToR Matching on flat topologies

- Example on the parallel network topology:



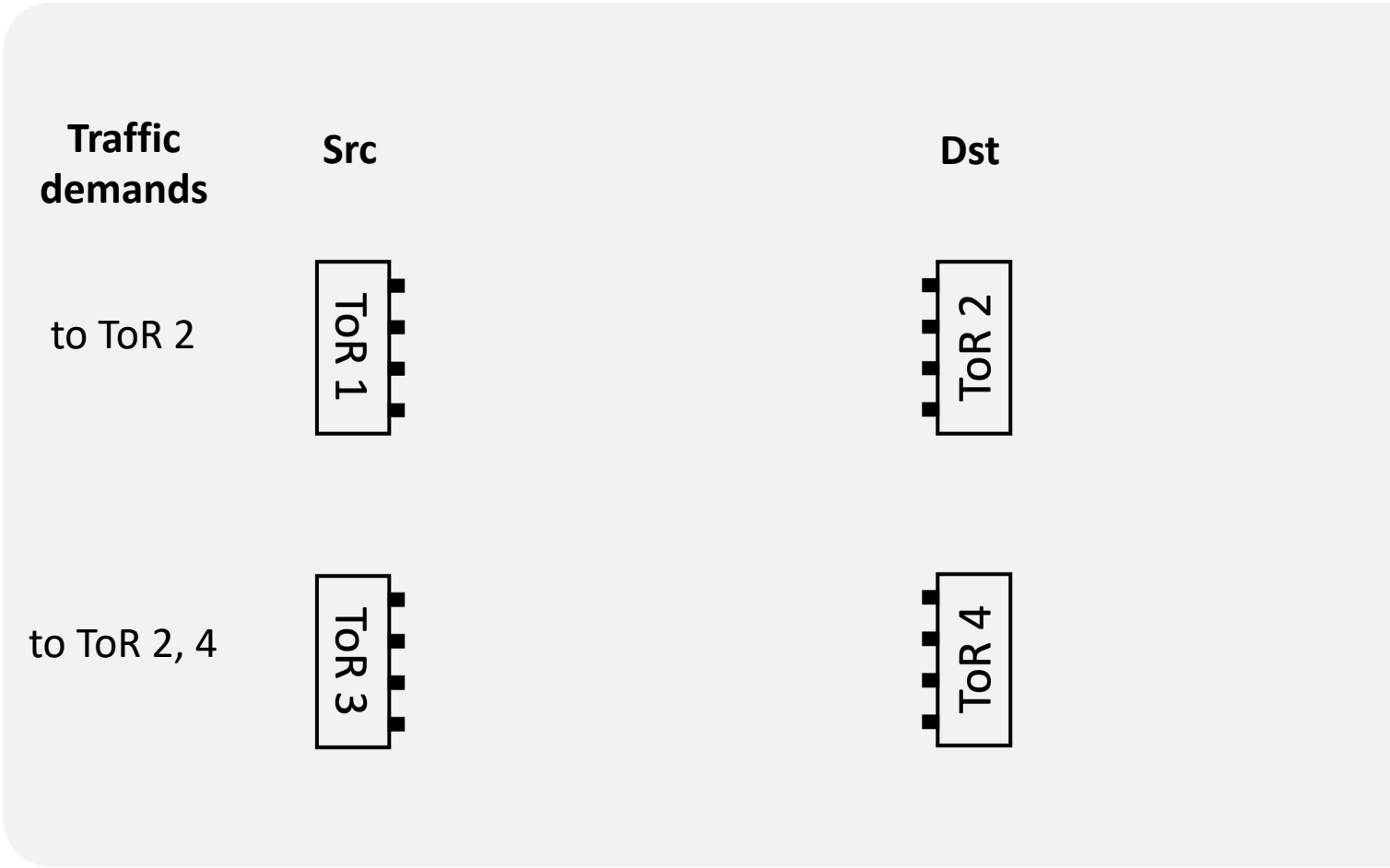
NegotiaToR Matching on flat topologies

- Example on the parallel network topology:



NegotiaToR Matching on flat topologies

- Example on the parallel network topology:

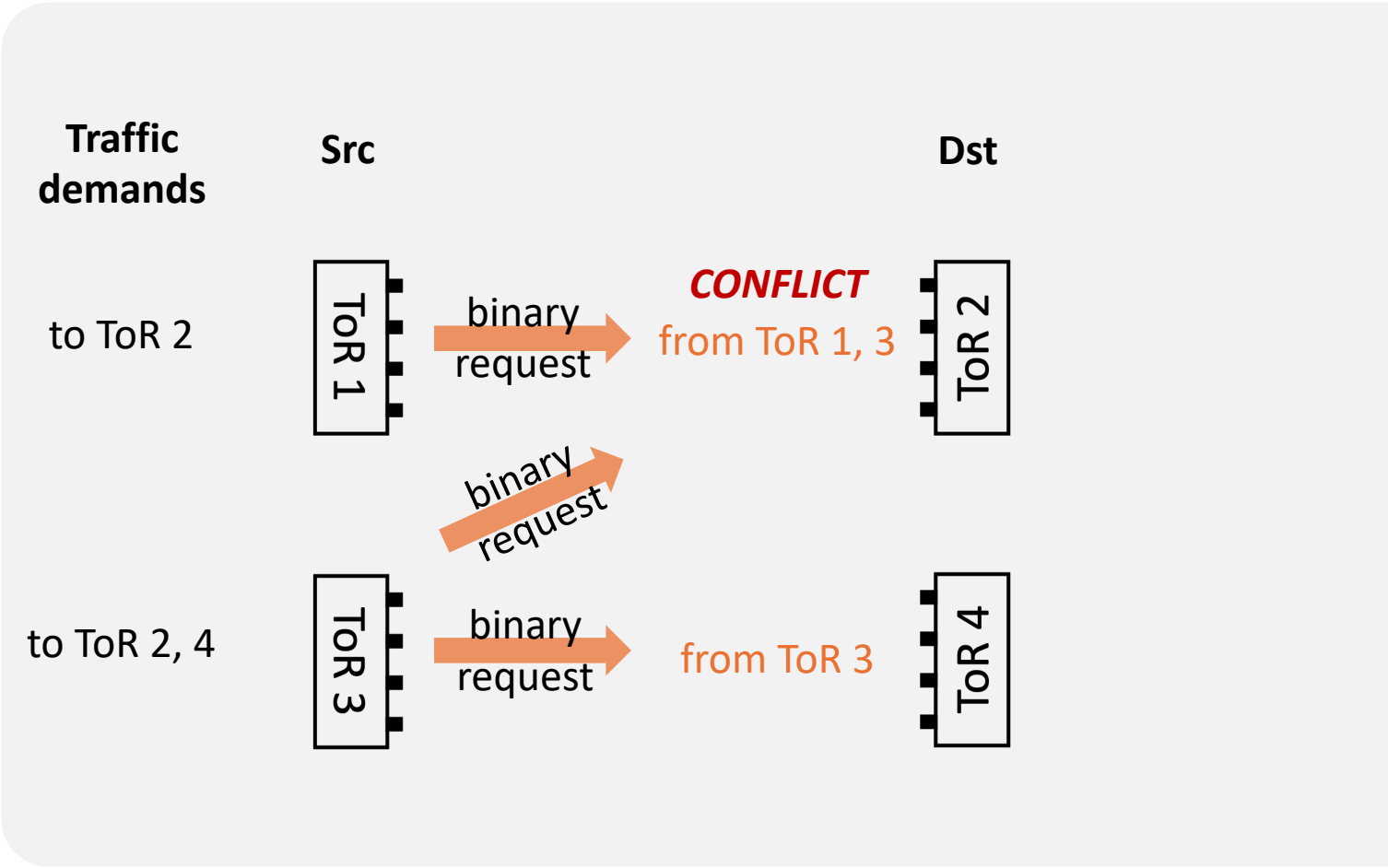


NegotiaToR Matching on flat topologies

- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations



NegotiaToR Matching on flat topologies

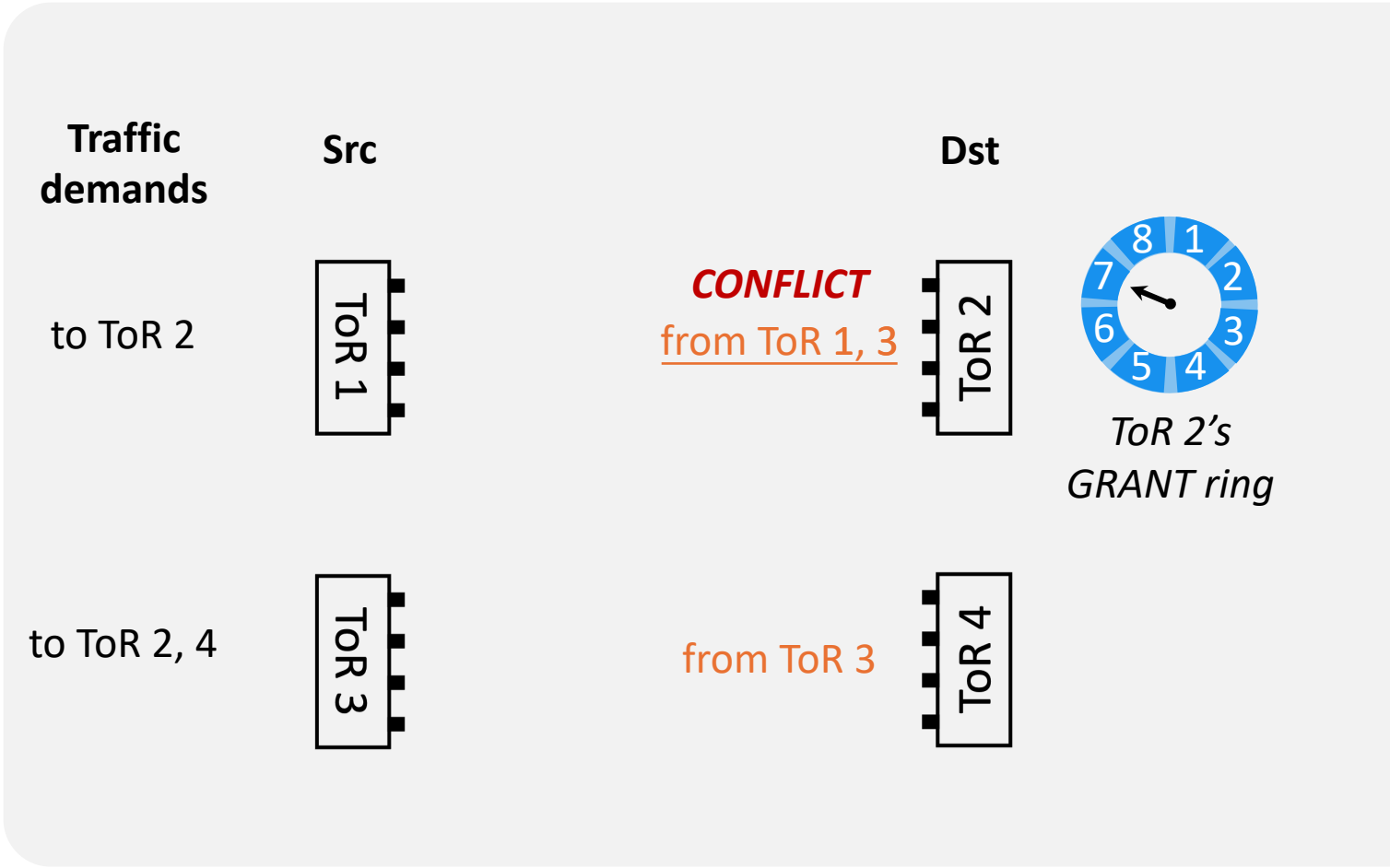
- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM



NegotiaToR Matching on flat topologies

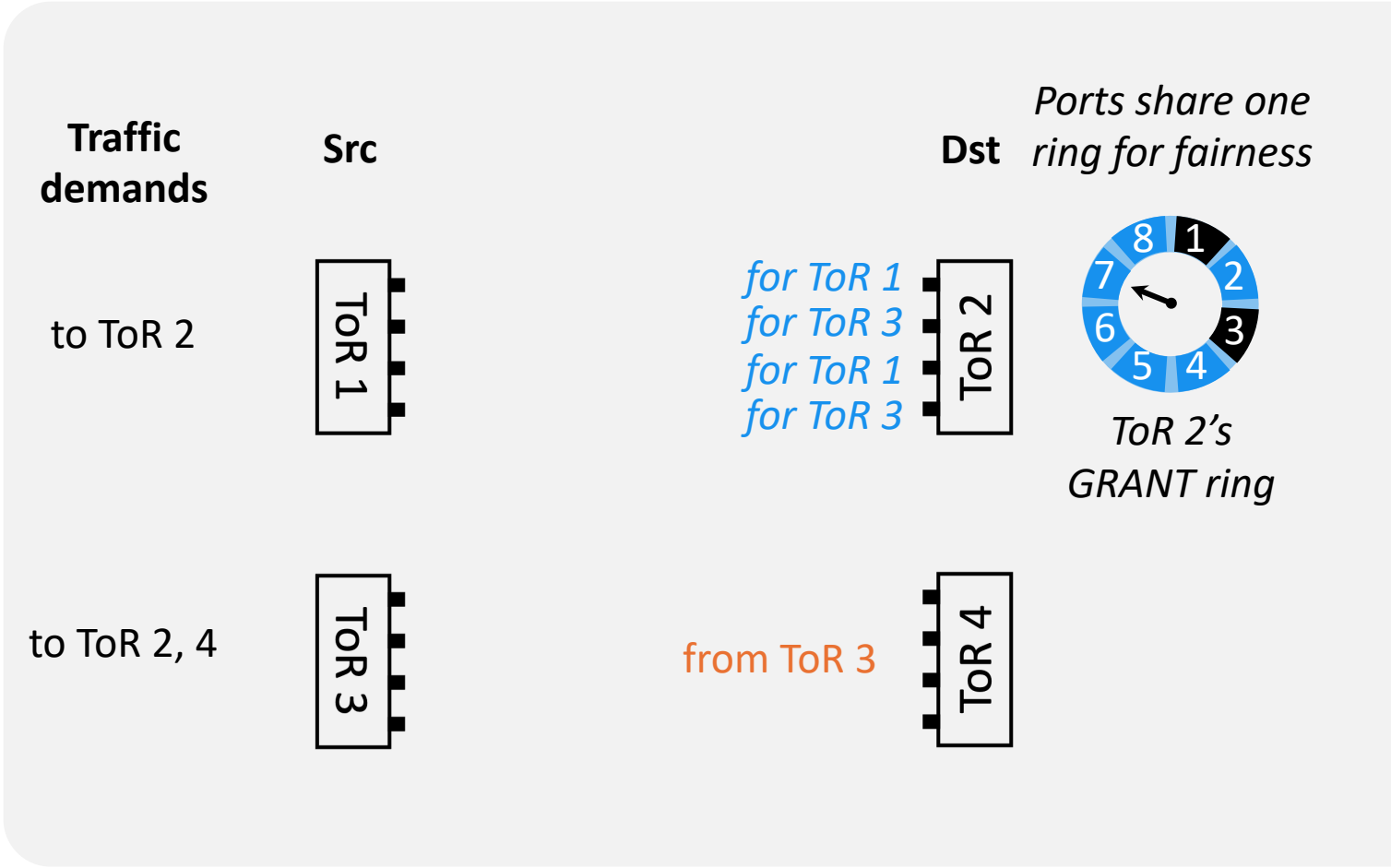
- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM



NegotiaToR Matching on flat topologies

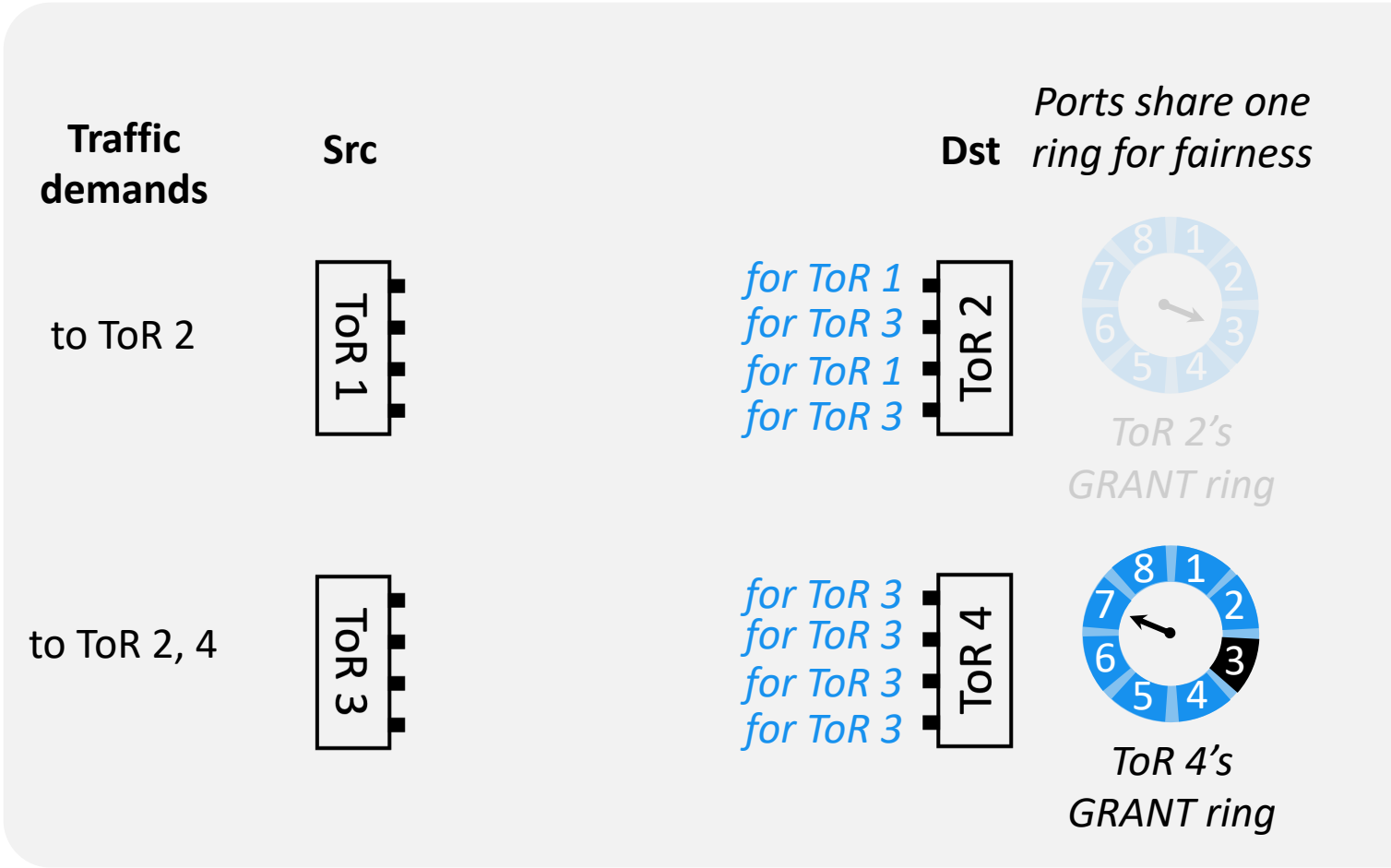
- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM



NegotiaToR Matching on flat topologies

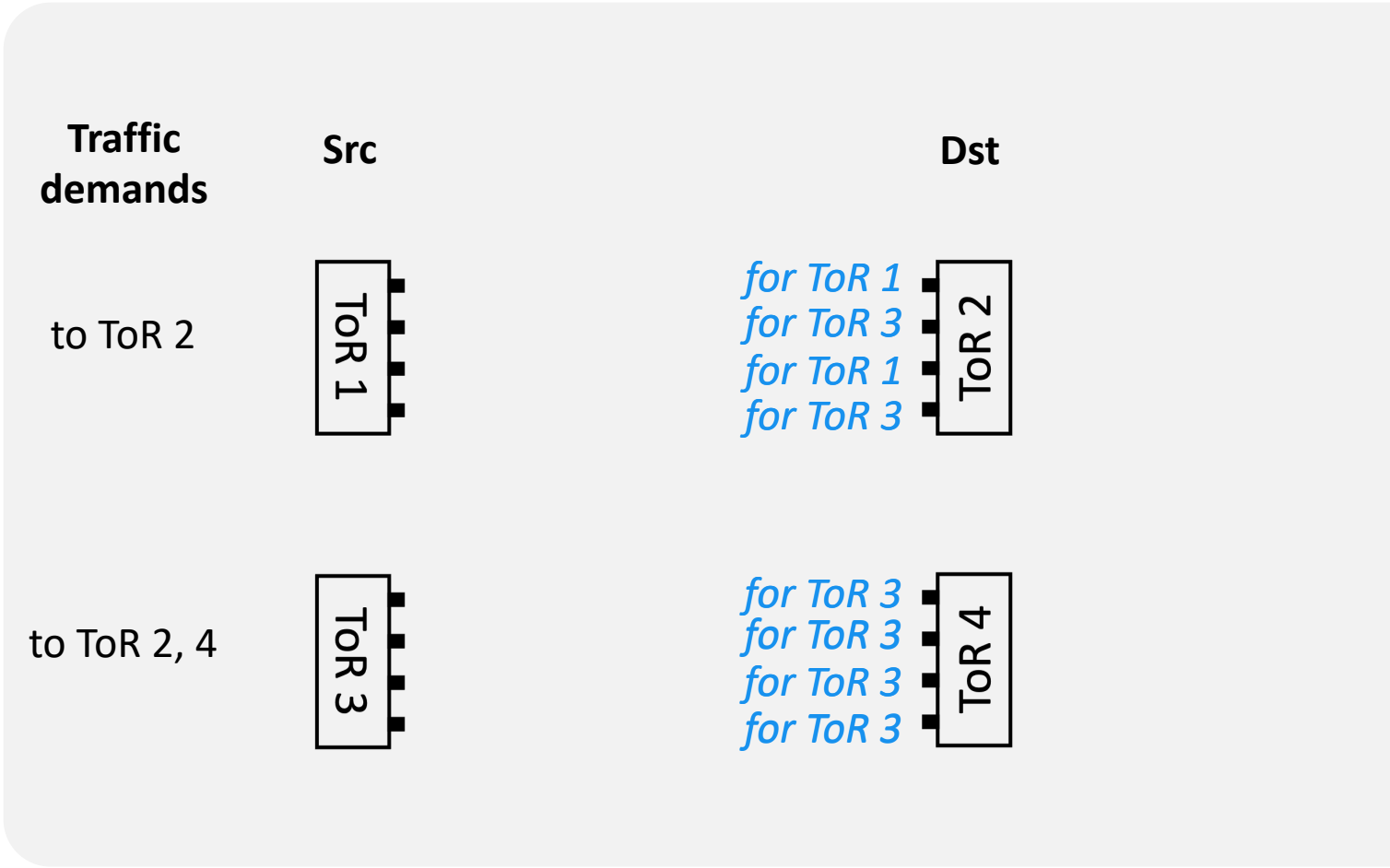
- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM



NegotiaToR Matching on flat topologies

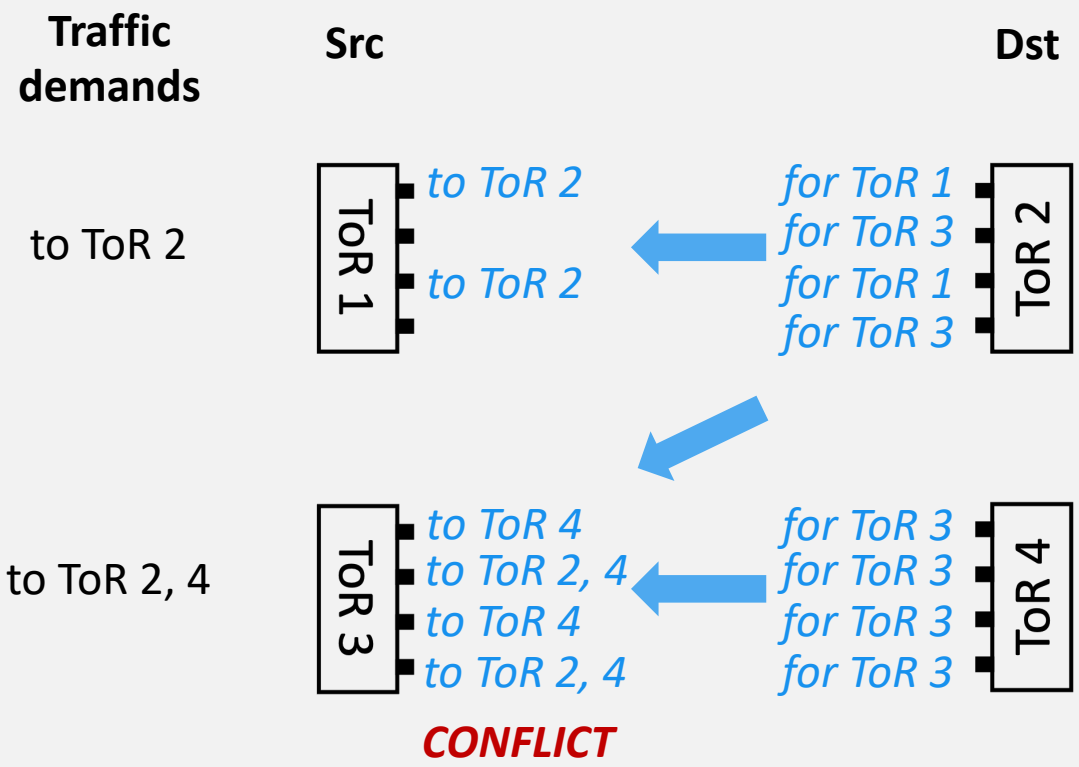
- Example on the parallel network topology:

ToR-level REQUEST

Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM



NegotiaToR Matching on flat topologies

- Example on the parallel network topology:

ToR-level REQUEST

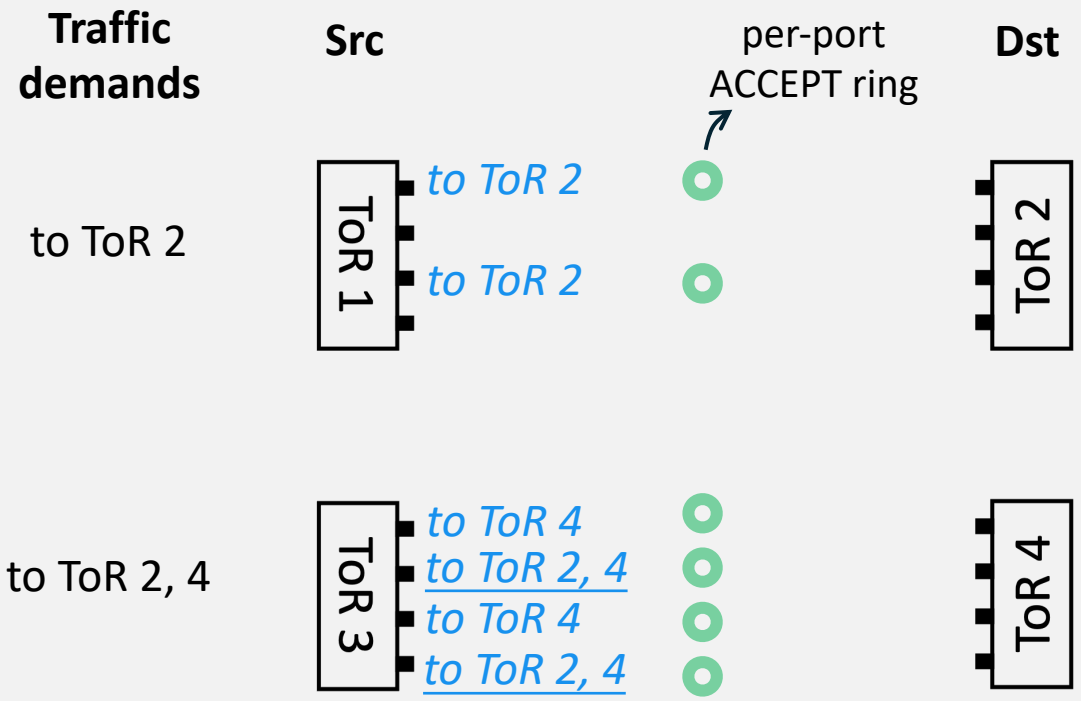
Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM

Port-level ACCEPT

Eliminate one-to-many conflicts at sources with RRM



NegotiaToR Matching on flat topologies

- Example on the parallel network topology:

ToR-level REQUEST

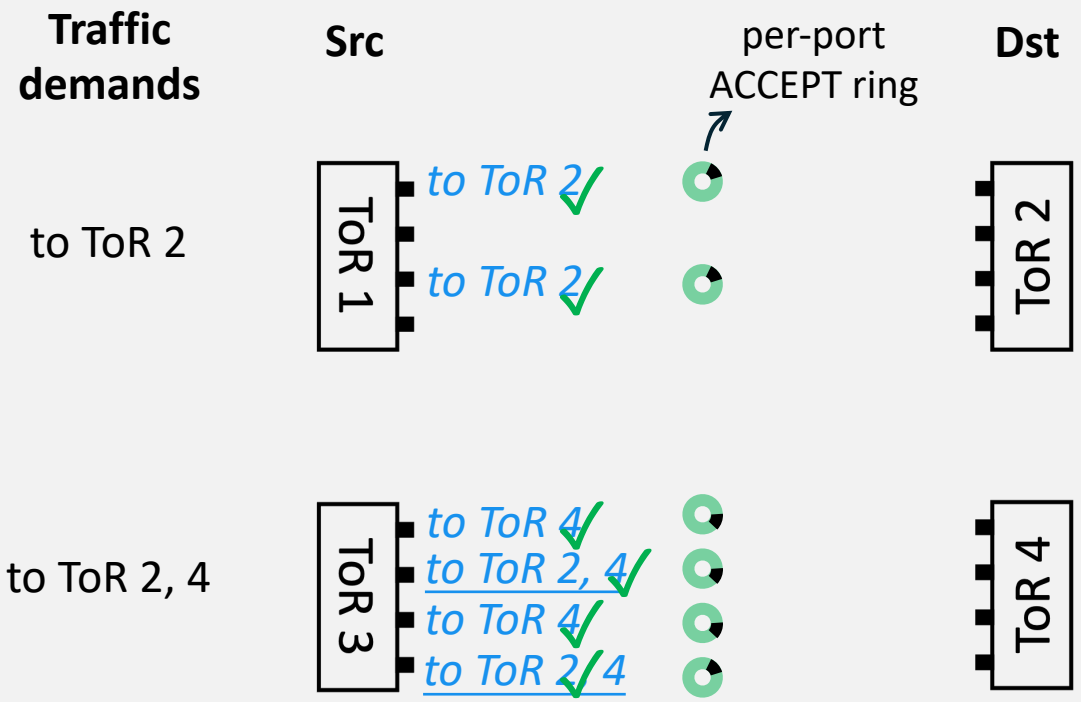
Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM

Port-level ACCEPT

Eliminate one-to-many conflicts at sources with RRM



NegotiaToR Matching on flat topologies

- Example on the parallel network topology:

ToR-level REQUEST

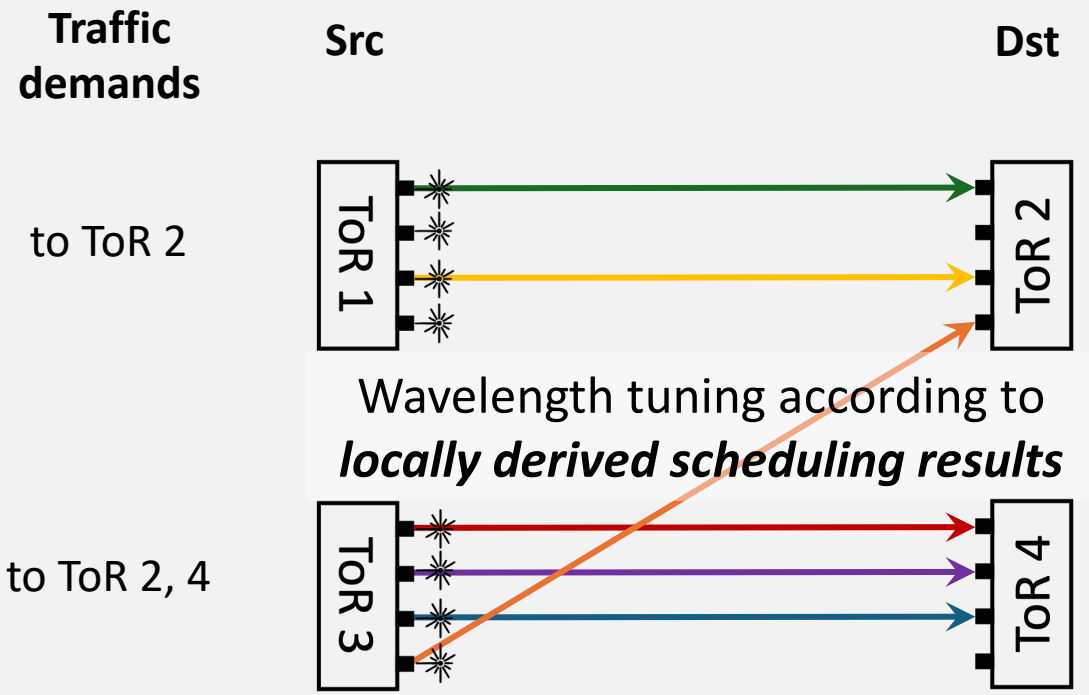
Send binary requests to destinations

Port-level GRANT

Eliminate many-to-one conflicts at destinations with RRM

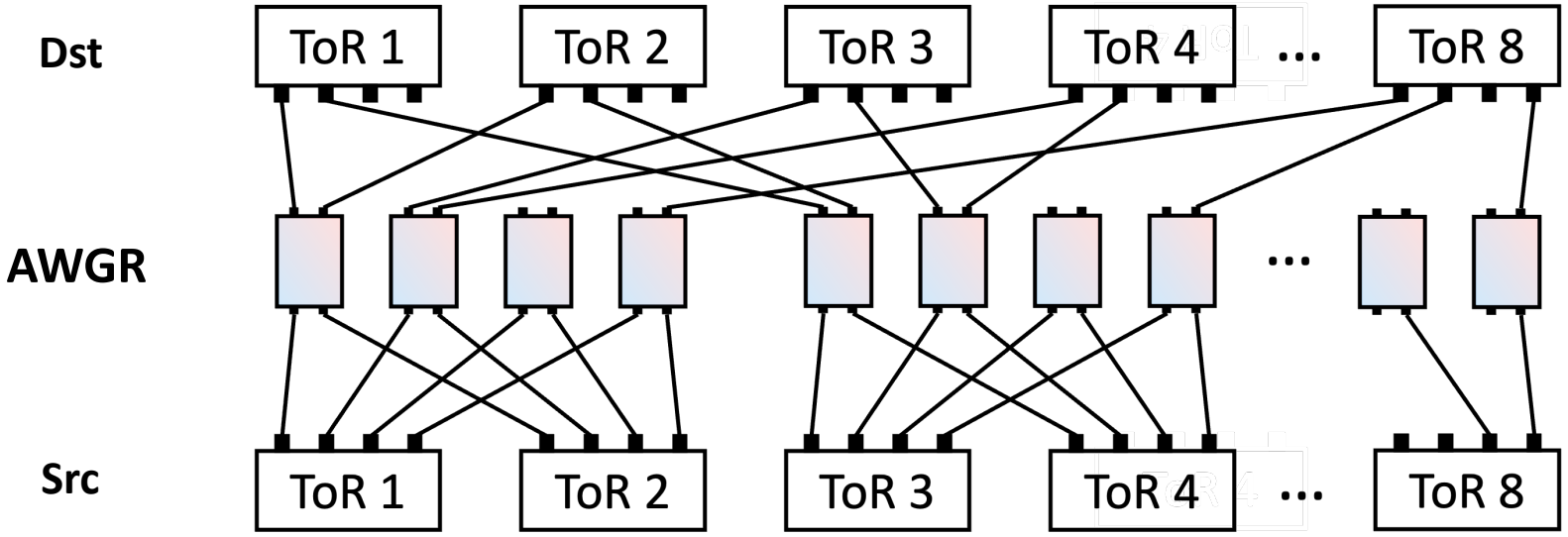
Port-level ACCEPT

Eliminate one-to-many conflicts at sources with RRM



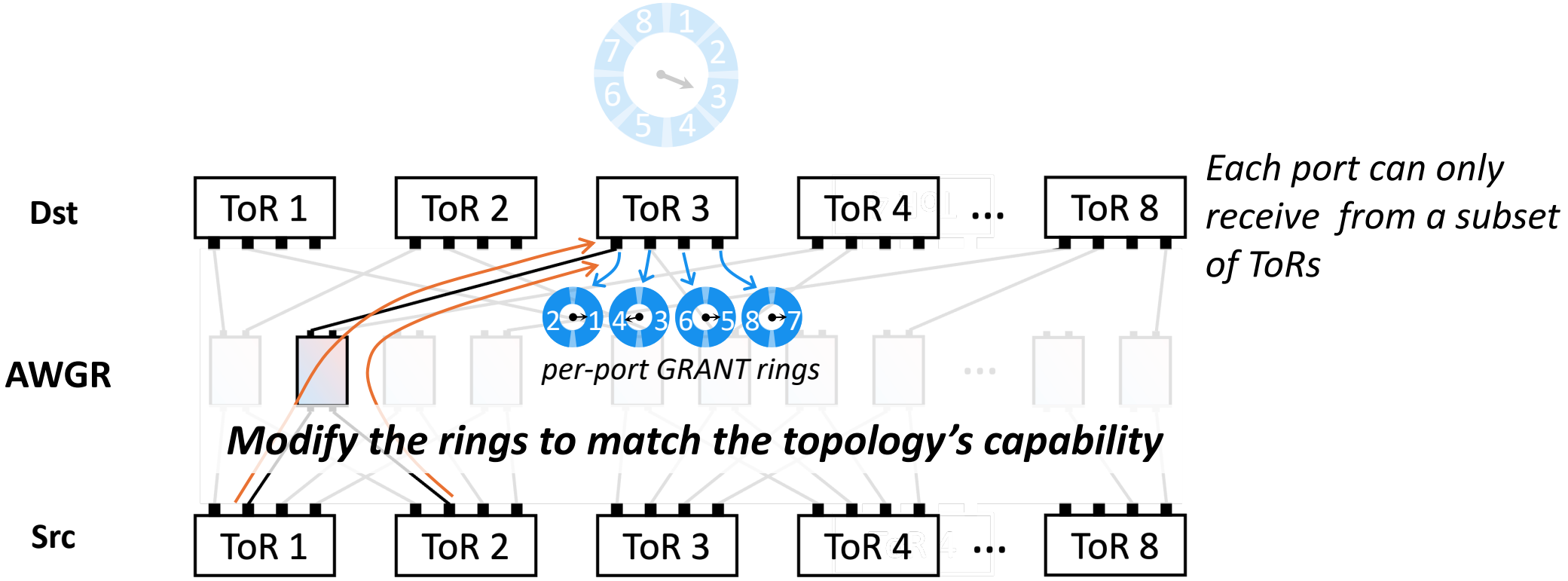
Adapting NegotiaToR Matching to topology capability

- For topologies with limited connectivity…
  - Example on the thin-clos topology with lower port-count AWGRs



Adapting NegotiaToR Matching to topology capability

- For topologies with limited connectivity…
  - Example on the thin-clos topology with lower port-count AWGRs

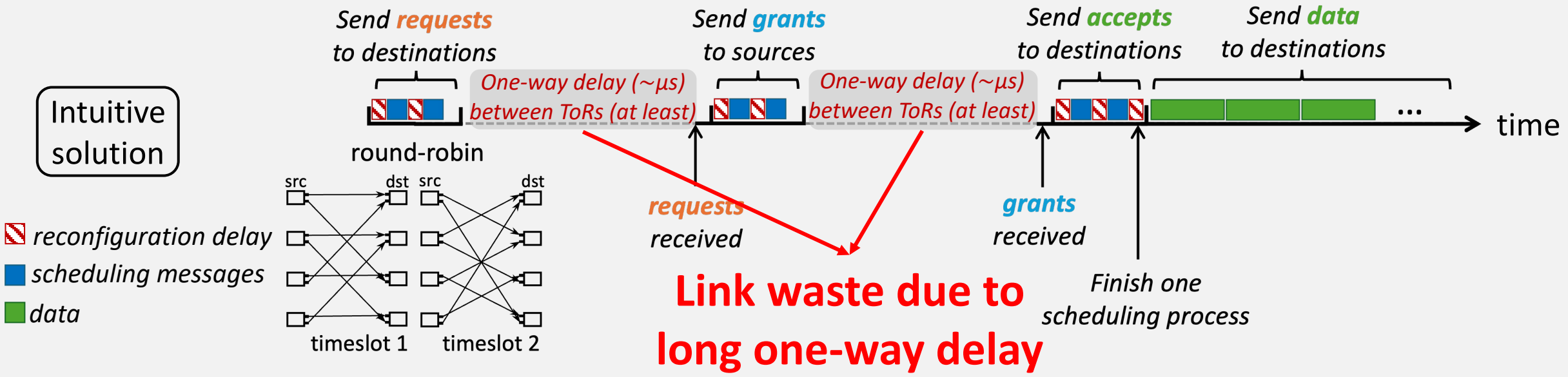


Supporting frequent exchanges of scheduling messages

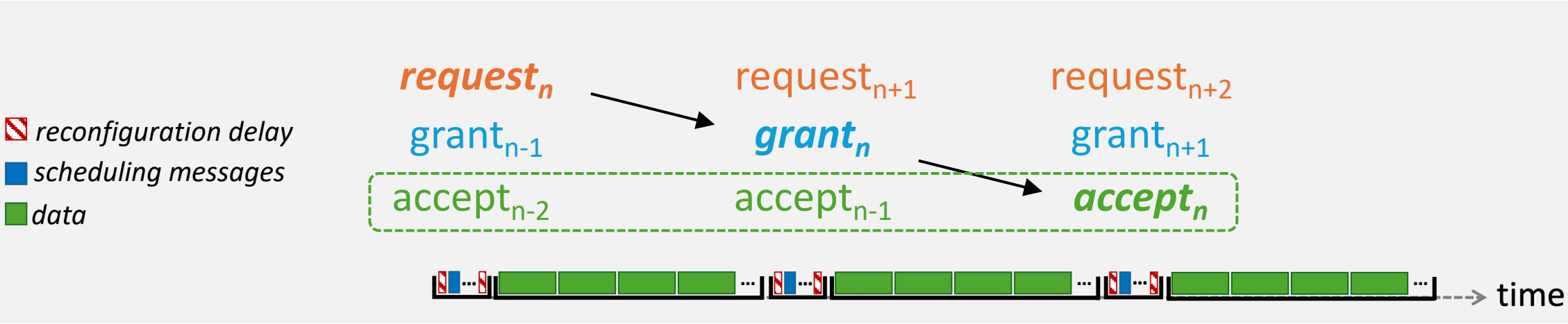
ToR-level REQUEST → Port-level GRANT → Port-level ACCEPT

- Requires full-mesh connectivity among all ToRs
- Following previous practices[1][2][3]
  - Round-robin reconfiguration to work as full-mesh

[1] Mellette et al., SIGCOMM '17
[2] Shrivastav et al., NSDI '19
[3] Ballani et al., SIGCOMM '20



Pipelined scheduling through two-phase epochs



- **Pipelined scheduling**: Run three scheduling processes simultaneously
  - Get a set of accepts after each round-robin
  - Reconfiguration and data transmission after accepts

Pipelined scheduling through two-phase epochs

reconfiguration delay
scheduling messages
data

One NegotiaToR Epoch *(fixed time interval)*

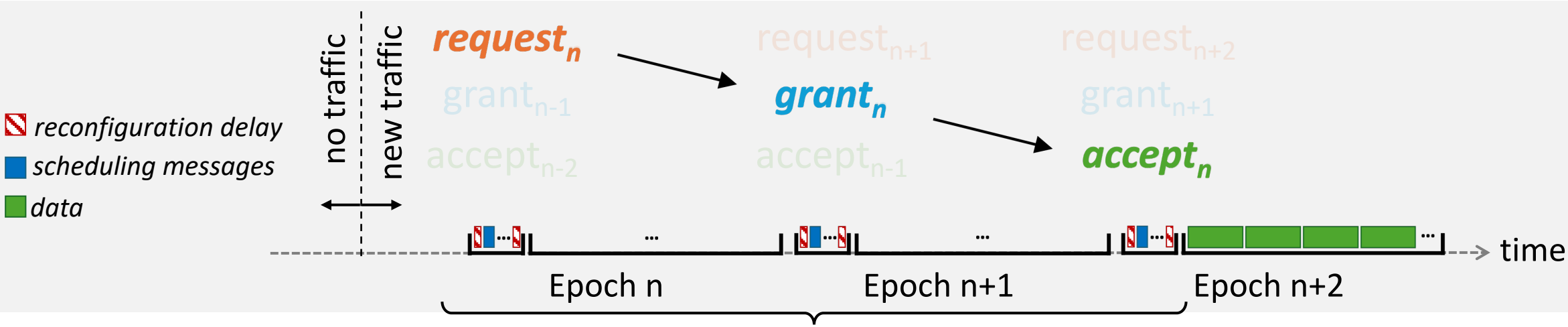
Predefined phase

- *Frequent reconfigurations*
- *Predefined connections*
- *Pipelined distributed scheduling*

Scheduled phase

- *No reconfiguration during it*
- *On-demand connections*
- *Send data through direct paths*

Latency vs. Scheduling delay



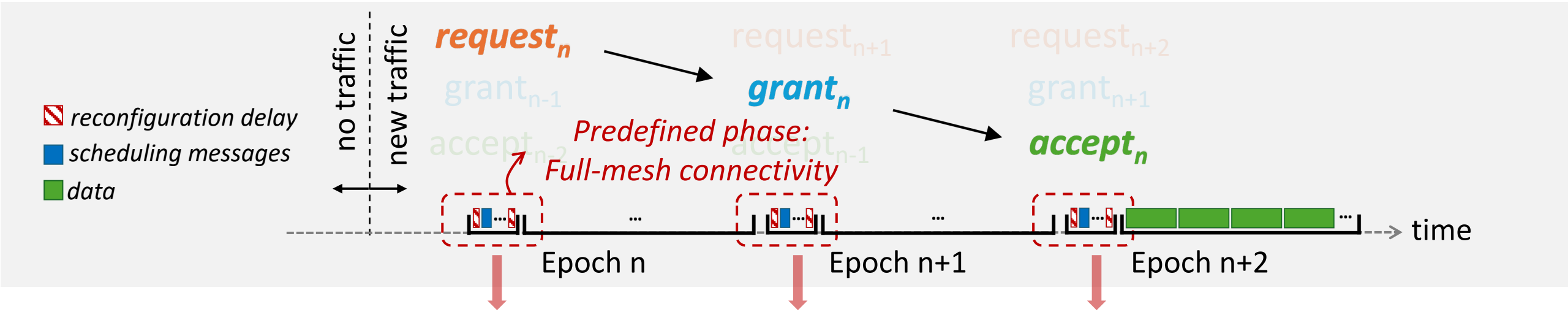
Long scheduling delay (even in lucky cases) ~2 epochs

- Due to ~µs one-way delay between ToRs
- The scheduling delay may be longer when conflicts occur

Mice flows? Incasts?

Bypassing scheduling delay to optimize latency?

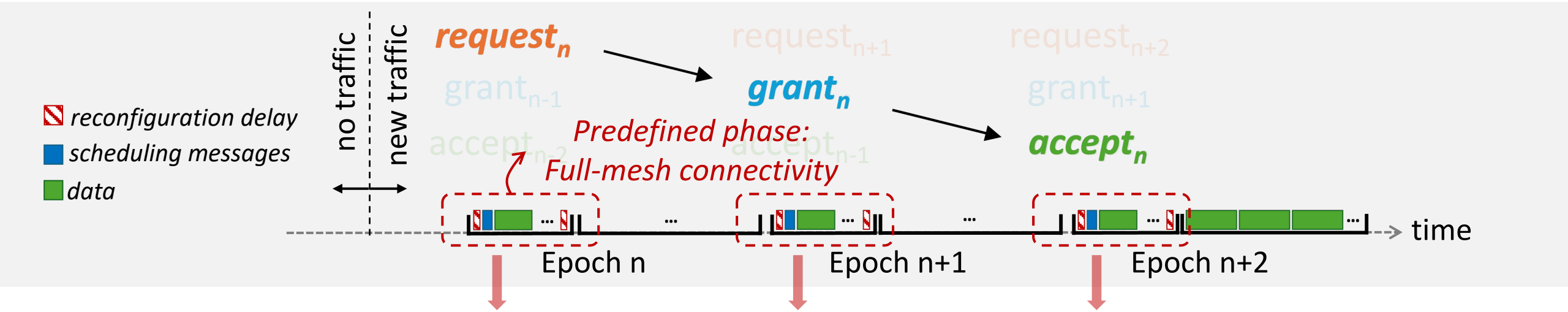
Scheduling delay bypassing for latency optimization



Role of the predefined phase

- Control plane: Exchanges of scheduling messages

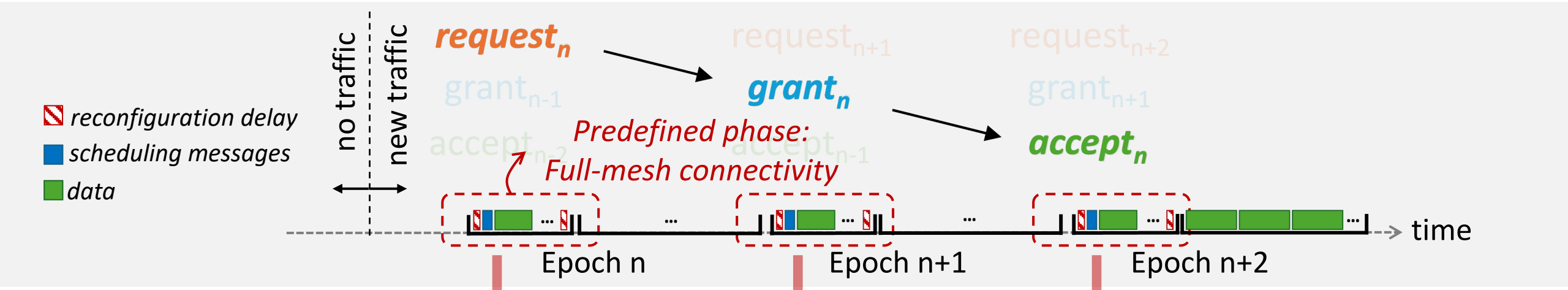
Scheduling delay bypassing for latency optimization



Role of the predefined phase

- Control plane: Exchanges of scheduling messages
- Data plane: Also piggybacking small data packets
  - Guaranteed unscheduled transmission
  - Sufficient for mice flows (small in size), even under incasts

Scheduling delay bypassing for latency optimization



Role of the predefined phase

- Control plane: Exchanges of scheduling messages
- Data plane: Also piggybacking small data packets
  - Guaranteed unscheduled transmission
  - Sufficient for mice flows (small in size), even under incasts
  - Maximize opportunities: Prioritize mice flows

HoL blocking by elephant flows?

✓ Information-agnostic mice flow prioritization like PIAS[1]

[1] Bai et al., NSDI '15

NegotiaToR's design efforts towards practicality

[1] McKeown, PhD Thesis, UC Berkeley '95
[2] Gupta et al., IEEE Micro '99

- **Proven implementation strategies**
  - Inspired by RRM[1]: Programmable priority encoders to implement rings[2]
- **Simple designs**
  - No iteration
  - No data relay
  - Binary requests
  - Stateless scheduling

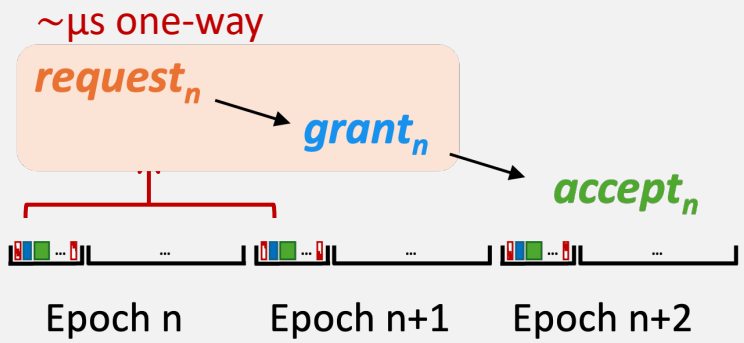
Could a more complex design significantly improve NegotiaToR's performance?

The rationality of NegotiaToR's design choices

- **Example:** Iterative scheduling?
  - Following practices in crossbar switch port matching
  - ! Difference: NegotiaToR has a longer delay between consecutive steps

No iteration

***Scheduling delay*:**
2 epochs +

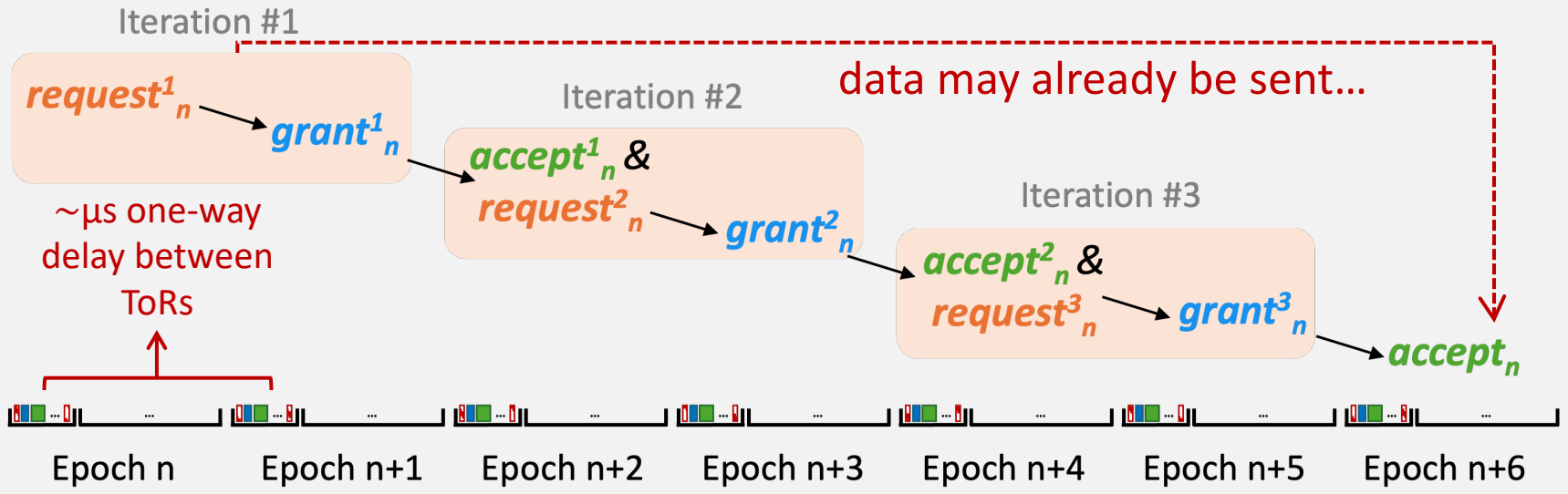


The rationality of NegotiaToR's design choices

- **Example:** Iterative scheduling?
  - Following practices in crossbar switch port matching
  - ! Difference: NegotiaToR has a longer delay between consecutive steps

3 iterations

***Scheduling delay*:**
6 epochs +



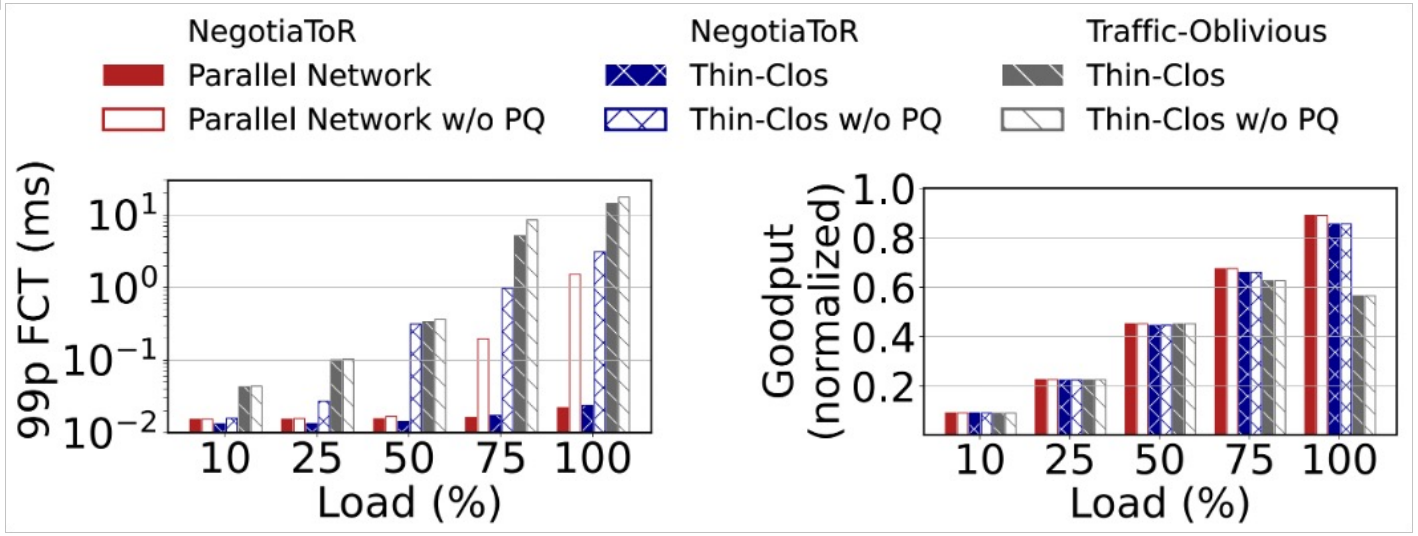
Possibly downgraded performance **(Confirmed by experimental results)** ← Link waste ← Outdated requests

Please refer to our paper for more explorations

Evaluation

[1] Roy et al., SIGCOMM '15
[2] Ballani et al., SIGCOMM '20

- Simulation setup: 128 ToRs, Hadoop trace[1]
  - Connected by 2 flat topologies
  - 10 ns reconfiguration delay
  - 2x speedup for both NegotiaToR and the traffic-oblivious scheme[2]
  - Consider ToRs as endpoints
  - Results without mice flow prioritization are also shown



99th-percentile FCT of mice flows (<10KB)

Goodput

NegotiaToR achieves both small mice flow FCT and high goodput on two representative flat topologies

NegotiaToR summary

An on-demand reconfigurable DCN architecture
towards a simple yet effective design

NegotiaToR Matching algorithm
Towards minimalist on-demand scheduling
on flat topologies

Two-phase epoch
In-band pipelined scheduling &
One-hop direct transmission

Scheduling delay bypassing
FCT optimization for
latency-sensitive flows

Thank you!

Speaker: Cong Liang
liangxcong@outlook.com