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

ABSTRACT

Recent advances in fast optical switching technology show promise in meeting the high goodput and low latency requirements of datacenter networks (DCN). We present NegotiaToR, a simple network architecture for optical reconfigurable DCNs that utilizes on-demand scheduling to handle dynamic traffic. In NegotiaToR, racks exchange scheduling messages through an in-band control plane and distributedly calculate non-conflicting paths from binary traffic demand information. Optimized for incasts, it also provides opportunities to bypass scheduling delays. NegotiaToR is compatible with prevalent flat topologies, and is tailored towards a minimalist design for on-demand reconfigurable DCNs, enhancing practicality. Through large-scale simulations, we show that NegotiaToR achieves both small mice flow completion time (FCT) and high goodput on two representative flat topologies, especially under heavy loads. Particularly, the FCT of mice flows is one to two orders of magnitude better than the state-of-the-art traffic-oblivious reconfigurable DCN design.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Network architectures**.

KEYWORDS

Datacenter network; Optical switching

ACM Reference Format:
Cong Liang, Xiangli Song, Jing Cheng, Mowei Wang, Yashe Liu, Zhenhua Liu, Shizhen Zhao, Yong Cui. 2024. NegotiaToR: Towards A Simple Yet Effective On-demand Reconfigurable Datacenter Network. In *ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3651890.3672222

1 INTRODUCTION

With more and more applications running in the cloud, traffic demands in DCNs increase continuously [45]. Among the applications, tasks like high performance computing (HPC) are greatly affected by both goodput and latency, putting stringent requirements on the network [27]. However, existing packet-switched DCNs struggle to meet these requirements due to inadequate capacity of switching chips [4] which will worsen with the slowdown of Moore's Law [35, 36]. To address this, the development of optical switching technology [11, 13, 14, 19, 24, 37, 42, 43] especially fast optical switching [13, 14, 19, 43] has led researchers to turn to reconfigurable optical DCNs [4, 5, 8, 10, 15, 21, 23, 28, 29, 32, 33, 38, 39, 47, 50, 51, 54], which provide higher capacity as well as lower cost compared with packet switching.

Unlike packet-switched networks that rely on buffers to absorb conflicts, bufferless optical switching requires synchronous scheduling and reconfiguration to accommodate the dynamic traffic demands including incasts [1, 6, 20, 53]. Recently, traffic-oblivious reconfigurable DCNs [4, 33, 44] have drawn researchers' attention because of their simplicity. They schedule the network according to predefined rules and use data relay to cope with dynamic traffic patterns. However, despite their simplicity, the data-relay design can lead to compromised performance in goodput and latency. Consequently, there is a growing need for a new design that can effectively handle dynamic traffic demands, ensuring both high goodput and low latency while still maintaining feasibility.

We present NegotiaToR, an on-demand reconfigurable DCN architecture with a simple design. With in-band distributed scheduling, it dynamically adapts the optical links among top-of-rack (ToR) switches to traffic demands, and sends data directly to destinations through one-hop paths. Beyond the scheduled connections, NegotiaToR also provides unscheduled connections, mitigating the impact of scheduling delays even under incasts. Utilizing existing arrayed waveguide grating routers (AWGR) and fast-tunable lasers, NegotiaToR is compatible with prevalent flat topologies, achieving a better performance than the state-of-the-art traffic-oblivious scheme on the same hardware with similar complexity.

On-demand is an intuitive solution to handle the dynamic traffic in DCNs, especially for unpredictable ToR-ToR traffic where demand forecasting based on historical data is difficult [6]. When put into the context of fast optical switching, previous on-demand solutions [5, 10, 15, 23, 29, 38, 50] introduce excessive scheduling complexity and raise practicality concerns. The key challenge of NegotiaToR thus lies in realizing scalable on-demand scheduling with high practicality.

First of all, scheduling often leads to reduced practicality due to the scale of DCNs [41] and the dynamics of traffic demands [1, 6, 20, 53]. Measuring detailed traffic demands of all senders and then using them as input to calculate non-conflicting paths often introduce high complexity. In contrast, NegotiaToR's demand information is binary. No data size information is needed by the scheduling algorithm. Meanwhile, NegotiaToR distributes the scheduling to ToRs, where each ToR only accounts for the scheduling of ingoing and outgoing traffic, achieving a comparable low complexity with traffic-oblivious designs.

ACM SIGCOMM '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0614-1/24/08.
https://doi.org/10.1145/3651890.3672222

Then, it is usually expensive to build and maintain a robust control plane for distributed scheduling in on-demand optical reconfigurable DCNs. Supporting a distributed scheduling algorithm requires ToRs to frequently transmit scheduling messages while avoiding conflicts with data transmission. An independent control network can solve the problem, but will introduce additional deployment and maintenance costs. NegotiaToR implements an in-band control plane, where all ToRs periodically establish all-to-all connectivity with fast optical switching. The scheduling algorithm runs on it in a pipelined manner. Senders can access the scheduling results locally and tune lasers to the derived wavelength without the need to deliver scheduling results.

Finally, ensuring low latency of mice flows, particularly when incast happens, is challenging in scheduling. Mice flows that occupy a large number of flows in DCNs [34] are latency-sensitive and usually arrive simultaneously as incasts due to the partition/aggregate design pattern [1] of DCN applications. Other than lowering the scheduling delay, NegotiaToR utilizes the periodical all-to-all connectivity to piggyback a small volume of mice flow data along with scheduling messages, providing opportunities to bypass scheduling delays. This effectively manages incasts, promising application performance even under concurrent traffic demands.

The design of NegotiaToR is guided by the principle of Occam's Razor. For deployment practicality, we hope to find a minimalist design of on-demand reconfigurable DCNs. Possibilities are also explored to trade off simplicity for better performance. We evaluate NegotiaToR through simulations on two representative flat topologies. Results show that NegotiaToR outperforms the state-of-the-art traffic-oblivious reconfigurable DCN design under comparable complexity, both in FCT and goodput. Particularly, NegotiaToR's mice flow FCT is one to two orders of magnitude better.

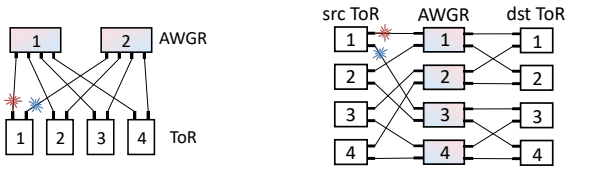
To summarize, we make the following contributions:

- We present NegotiaToR, a simple reconfigurable DCN architecture with scalable on-demand scheduling for flat topologies with fast optical switching enabled **(§3)**. It comprises the distributed NegtotiaToR Matching scheduling algorithm **(§3.2)**, the two-phase epoch serving as in-band control plane and data plane **(§3.3)**, and a mechanism to bypass scheduling delays even under incasts **(§3.4)**.
- We explore the possibilities to trade off simplicity for better performance, and show that extra complexity does not necessarily translate into proportionate performance gains **(§3.5)**.
- We evaluate NegotiaToR through simulations (**§4**). Results show that with similar complexity, NegotiaToR outperforms the state-of-the-art traffic-oblivious design in both FCT and goodput.

This work does not raise any ethical issues.

2 BACKGROUND & MOTIVATION

Optical switching technology has developed rapidly, and there has been a great interest in using them in DCNs in recent years [4, 8, 10, 15, 21, 23, 28, 29, 32, 33, 38, 39, 42, 47, 50, 51, 54]. To fully utilize the high capacity and power efficiency of optical switching, one trend is to connect racks with optical switches directly [4, 10, 21, 32, 33, 51]. This poses two main considerations for optical switching hardware:



(a) Parallel network topology built with high port-count AWGRs

(b) Thin-clos topology built with low port-count AWGRs

Figure 1: Flat topologies in AWGR-based optical DCNs.

supporting the large number of ToRs, and accommodating the highly dynamic traffic demands among them.

Fast optical switching technology is ready for DCNs. Recent advances in AWGR-based switching, mainly faster reconfiguration (e.g., nanoseconds end-to-end reconfiguration delay [4]) and higher port count (e.g., 270 [37], 400 [24], and 512 [11] ports), have enabled it to meet the dynamic traffic demands of DCN. AWGR is a fully-passive optical switch. By tuning the wavelength of the tunable laser at the source, data can be forwarded to different destinations. This makes it a fit for distributed scheduling, where the source can tune the wavelength according to the scheduling results derived locally. Flat topologies with one layer of AWGRs shown in Figure 1 are commonly used, due to the insertion loss and wavelength limitation of AWGR cascading [40, 52].

Both high and low port-count AWGRs are capable of connecting all ToRs in the DCN scale. Notably, for high port-count AWGRs, a single AWGR can interconnect all ToRs, facilitating the implementation of the parallel network topology as shown in Figure 1(a). For more accessible low port count AWGRs that are insufficient to connect all ToRs alone, the thin-clos topology [40, 52] illustrated in Figure 1(b) becomes a practical alternative. In this topology, unlike the former one, each port of a ToR can only connect to a subset of other ToRs, and all ports together can reach the whole network. Meanwhile, recent advancements in fast-tunable lasers [4, 13, 14, 16, 19, 43, 49] and time synchronization [4, 18] have reduced the end-to-end reconfiguration delay to as low as 10 nanoseconds [4], which is sufficient to cope with the dynamic traffic demands among ToRs.

Scheduling optical interconnections among ToRs with dynamic and unpredictable traffic demands is challenging. For optical interconnections at higher switch levels, like the spine layer, since the traffic pattern is typically predictable due to job placement strategies, infrequent reconfiguration based on historical trace and traffic engineering techniques are sufficient [8, 39] for performance.

However, regarding inter-ToR connections, they face the challenge of highly dynamic and unpredictable traffic which includes bursty scenarios like incasts [1, 6, 20, 53]. The development of fast switching technology has made it possible to reconfigure the network on-demand to adapt to the dynamic traffic. A centralized scheduler can do the job, but faces practicality concerns because of the scheduler's limited scalability. Recently, researchers have turned to traffic-oblivious designs [4, 44]. The network reconfigures itself regularly in a round-robin manner to provide all-to-all

connectivity, regardless of actual traffic patterns. To mitigate the resulting mismatch of network connectivity and traffic demands, they use Valiant's Load Balancing (VLB) [9, 48] to adapt the traffic to the network, uniforming the traffic pattern to all-to-all by spreading data across the network before routing it to the final destination, and thus utilizing all links. Such approaches are practical but come at the expense of goodput and latency. Data relay doubles the traffic volume, competing for receivers' bandwidth, potentially causing congestion and damaging goodput where worst-case goodput can downgrade to 50%. Meanwhile, the detouring also damages mice flow FCT, particularly when elephant flows are spread across the network and block the mice ones at intermediate nodes. The performance downgrade worsens under heavier loads, which is a critical concern for HPC tasks like large-scale ML training where large amounts of flows are synchronously released to the network [25, 27, 47]. NegotiaToR is designed to meet these needs, offering a practical solution that can accommodate the high-performance requirements of modern DCNs where fast optical switching technology is ready.

3 NEGOTIATOR DESIGN

NegotiaToR is a simple network architecture for reconfigurable DCNs where bufferless optical links connect buffered ToRs. With minimal traffic demand information, it schedules traffic distributedly on the ToRs via the in-band control plane in an on-demand while scalable manner.

3.1 Design overview

NegotiaToR is compatible with prevalent flat topologies. As depicted in Figure 1, we choose two representative flat topologies to demonstrate our design, the parallel network topology that necessitates high port-count AWGRs, and the thin-clos topology that only needs readily available low port-count AWGRs. In both topologies, ToRs' uplink ports are equipped with fast-tunable lasers and attached to AWGR-based optical switches. One ToR maintains a FIFO queue for each of the other ToRs in the network. Data are first sent to this per-destination queue before being put into per-port queues and heading for their destinations. For data transmission, ToRs are time-synchronized[1], and simultaneously send bits according to non-conflicting port-level matches that the topology can provide.

To this end, NegotiaToR employs a distributed scheduling mechanism on a per-epoch basis, where an epoch is a fixed-length time interval. Each epoch comprises two phases, working as the in-band control plane and data plane **(§3.3)**. All packets are transmitted directly to their destinations through one-hop paths. We present the design of one epoch in Figure 2. In the predefined phase, ToRs exchange scheduling messages through the all-to-all connectivity provided by round-robin via fast wavelength switching, enabling distributed scheduling. Note that network reconfiguration only happens in this phase, lowering reconfiguration overhead. In the subsequent scheduled phase, ToRs set the wavelengths to the locally derived scheduling results and send data packets. To mitigate the impact of scheduling delays, especially under incasts, NegotiaToR also piggybacks a small volume of data with scheduling messages

[1]There is no need for AWGRs to be time synchronized since they are entirely passive.

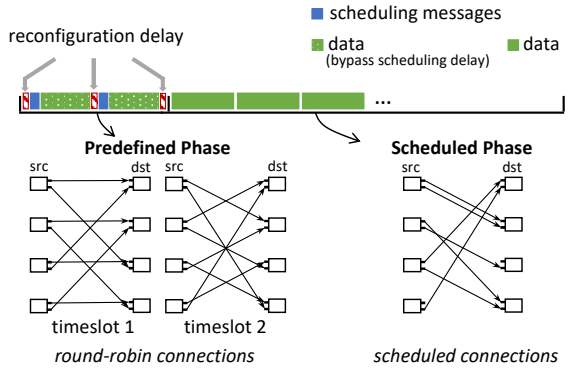


Figure 2: Each epoch in NegotiaToR comprises two phases, and reconfiguration only happens in the predefined phase.

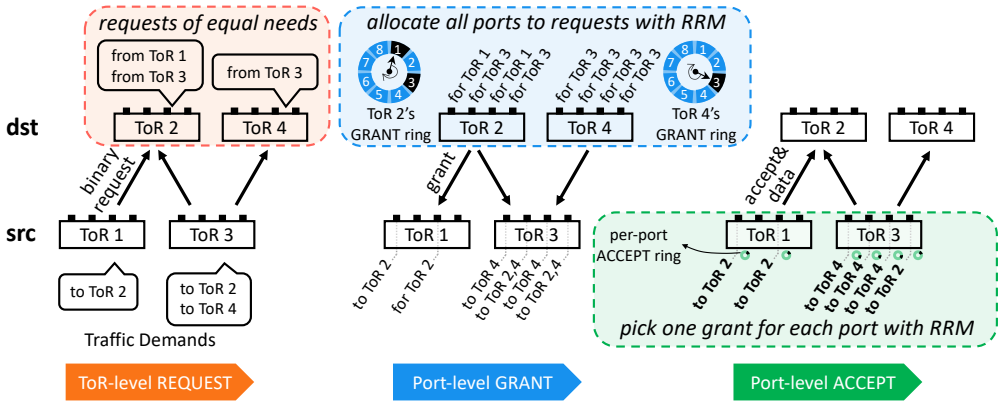
utilizing the unscheduled connections in the predefined phase, bypassing the scheduling delay especially for latency-sensitive mice flows **(§3.4)**.

Utilizing the in-band control plane, ToRs distributedly run a simple on-demand scheduling algorithm, NegotiaToR Matching **(§3.2)**, which accounts for both network performance and scalability. The algorithm does not require traffic forecasting and utilizes a non-iterative approach, making it suitable for the DCN scenario where ToR-ToR traffic is highly dynamic and the round-trip time (RTT) between ToRs is often long (e.g., several microseconds) compared with nanoseconds reconfiguration delay. Consequently, iterative distributed scheduling would result in long scheduling delays, adversely affecting mice flow performance that could otherwise benefit from fast optical switching. Each ToR functions like a negotiator. ToRs get simplified binary local traffic demands from their per-destination queues, exchange scheduling messages regularly with each other, and make scheduling decisions locally according to their incoming and outgoing traffic needs. A new scheduling process starts at the beginning of each epoch. One process lasts for three epochs, and the scheduling for consecutive epochs is performed in a pipelined manner. This way, NegotiaToR distributes on-demand scheduling computation to each ToR and adapts the network to real-time traffic, realizing feasibility and high performance.

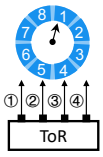
Guided by the principle of Occam's Razor, we carefully orchestrate NegotiaToR's design to be simple yet effective, especially the scheduling algorithm. We believe this is essential when it comes to industrial deployment. One may wonder whether the minimalist algorithm is sufficient and whether a slightly more complexity can significantly improve NegotiaToR's performance. To demonstrate the rationality of our design choices, we investigate several potential variants of NegotiaToR **(§3.5)**, which are meant to trade off complexity for possible performance gains. This exploration reveals that our minimalist design is sufficient, and additional complexity may not proportionately enhance performance as expected.

3.2 On-demand distributed scheduling

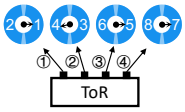
We designed a simple matching algorithm, NegotiaToR Matching, for scalable on-demand scheduling, which requires only three steps and no iteration. We first show the design of the algorithm itself



(a) An example on the parallel network topology. REQUEST: sources send binary requests representing traffic demands to their destinations. GRANT: conflict elimination at the destinations' side (i.e., many-to-one). ACCEPT: conflict elimination at the sources' side (i.e., one-to-many). The accepted grants indicate a set of non-conflicting matches



(b) Per-ToR GRANT ring on the parallel network topology



(c) Per-port GRANT rings on the thin-clos topology

Figure 3: NegotiaToR Matching's workflow. (a) shows the workflow on the parallel network topology. (b) and (c) illustrate the variance in the GRANT step across two topologies, attributable to differing connection capabilities. On the parallel network topology, port ①-④'s grant priority is determined by a shared ring, whereas on thin-clos it's determined by port-specific rings. Once granted, the pointer is updated to prioritize the next source.

before going into the big picture of how the algorithm runs on NegotiaToR's fabric and how NegotiaToR sends traffic according to locally derived scheduling results.

3.2.1 NegotiaToR Matching algorithm. The algorithm runs on ToRs. From real-time binary traffic demand information, ToRs distributedly generate non-conflicting matches for all uplink ports[2] so that ToR pairs with traffic demands will get connected. No traffic forecasting is needed. Each ToR is only responsible for traffic flows in and out of it, thus distributing the computational complexity. The scheduling process for one epoch is composed of three steps: REQUEST, GRANT, and ACCEPT. These steps distributedly assign links to source-destination port pairs, resolving conflicts at the destinations' and sources' ports, thereby ensuring collision-free data transmission for all ports.

We present the pseudo-code of NegotiaToR Matching in Algorithm 1. An illustrative example of its workflow is given in Figure 3. The example network consists of eight 4-port ToRs, where the parallel network topology connects them with four 8-port AWGRs, and the thin-clos topology with eight 2-port AWGRs. Only four ToRs are shown for simplicity. Note that the algorithm can be used in various flat topologies, including both the parallel network and the thin-clos topology presented in Figure 1 once the GRANT step is modified according to the topology's connection capabilities.

ToR-level REQUEST. Each ToR maintains per-destination FIFOs for all ToRs, and data are pushed into the corresponding queue first before heading for their destinations. By checking if the queue has pending data or not, ToRs thus know local traffic demands and notify the destinations by sending requests to them. The requests are ToR-level and are not bound to any specific port. The requests are binary and contain no size or flow-level information, and thus all requests indicate an equal need for link resources. This simplification facilitates subsequent rapid distributed calculation while being sufficient for DCN performance, as we will see later.

Port-level GRANT. Now, each destination ToR is aware of the requests from multiple ToRs. To avoid collisions at the destinations' side, the ToR will pick one request for each port in turn. To this end, inspired by RRM [31] used in the scope of crossbar packet switch port scheduling, the destination ToR employs round-robin rings to allocate ports to these requests. The position of the ring's pointer denotes the highest priority source, with priority diminishing in a clockwise direction. After granting one source, the pointer is incremented to the next source of the round-robin schedule. This way, we prioritize the source ToR that's least recently granted, effectively ensuring fairness and avoiding starvation in port allocation.

Depending on the connection capabilities the topology provides, the implementation of this ring differs. As shown in Figure 3(b), there is only one GRANT ring per ToR in the parallel network topology since one port can receive data from any other ToRs. Sharing scheduling states among ports of the ToR thus helps improve scheduling fairness further. In contrast, in the thin-clos topology, each destination ToR has multiple smaller GRANT rings like in Figure 3(c), one for each port. This is because one port in thin-clos can only receive data from a subset of source ToRs [40, 52]. As a result, grants for a specific port can only be chosen from a subset of requests indicated by the ring.

Port-level ACCEPT. After destinations send the grants back, there are possibilities that one port gets multiple grants from different destinations. To resolve this conflict and ensure that only one destination is assigned to one port, the source ToR accepts one grant for each port from received port-level grants, using a per-port ACCEPT round-robin ring for fairness.

[2]Unless specified, we refer to ToR's *uplink port* as *port* in this paper.

Algorithm 1: NegotiaToR Matching Algorithm

```
function REQUEST(per-destination queues)
    foreach queue_i in per-destination queues do
        // queue_i's destination is ToR_i
        if queue_i.has_pending_data then
            send a request to ToR_i;
function GRANT(received requests)
    // allocate ports with round-robin rings
    randomly initialize rings;
    foreach port_i do
        // get the highest-priority request with per-ToR or
         per-port GRANT ring
        request_x ← ring_grant.get_request;
        port_i.grant ← request_x;
        ring_grant.pointer_update;
        send port_i.grant back to ToR_x;
function ACCEPT(received grants)
    // accept grants with round-robin rings
    randomly initialize rings;
    foreach port_i do
        // get the highest-priority grant with per-port ACCEPT
         ring
        grant_x ← ring_accept.get_grant;
        port_i.accept ← grant_x;
        ring_accept.pointer_update;
        send traffic to ToR_x through port_i;
```

After these three steps, all ToRs are aware of a set of non-conflicting matches indicated by ACCEPT that's derived locally. NegotiaToR Matching enables scalable on-demand coordination among ToRs in a distributed manner. It has a comparable scheduling complexity to traffic-oblivious solutions [4, 33, 44], which require congestion control to avoid buffer overflow at intermediate ToRs, similar to NegotiaToR Matching's minimalist request-grant mechanism. Following the derived scheduling, NegotiaToR can achieve collision-free data transmission through bufferless links.

3.2.2 Matching efficiency. We demonstrate NegotiaToR Matching's efficiency in theory with a simplified model, showing that the algorithm has a matching efficiency of 63% even under high loads where conflicts are frequent.

Consider the following scenario, where we assume n ToRs ($n > 1$) are sending traffic to each other, each having m uplink ports. ToRs are connected by the **parallel network topology** in Figure 1(a). In this model, grants and accepts are randomly and uniformly given. In the GRANT step, on average, a source ToR receives grants from n ToRs, and a destination ToR allocates $\frac{m}{n}$ ports per request. The chance that one port is granted by one destination ToR is thus $\frac{m}{n}/m = \frac{1}{n}$. Focusing on a specific port ($port_0$) at the source side, the probability of $port_0$ being included in a specific grant ($grant_0$) is thus $\frac{1}{n}$. Let X be the number of competing grants for $port_0$, excluding $grant_0$. X thus follows a binomial distribution $X \sim B(n-1, \frac{1}{n})$. The acceptance probability of $grant_0$ by $port_0$ is $Y = \frac{1}{X+1}$. The expected value of Y is thus $E(\frac{1}{X+1}) = \sum_{k=0}^{n-1} \frac{1}{k+1} P(X = k)$, which is $1 - (1 - \frac{1}{n})^n$.

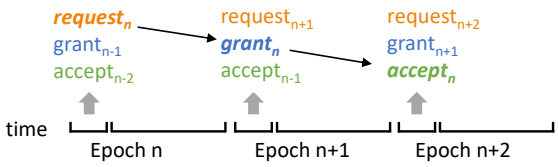


Figure 4: Scheduling for different epochs is performed in a pipelined manner.

As n increases, $E[Y]$ monotonically decreases and approaches $1-\frac{1}{e}$. This indicates that when the competition is intense, $grant_0$ still has a 63% chance to be accepted, and thus $port_0$ at the destination side will get matched. Otherwise, $port_0$ at the destination side will be wasted due to the destination's port being reserved but not accepted by the source ToR. When the same set of ToRs are connected by the **thin-clos topology**, the matching efficiency itself is typically higher. This is because the number of competing grants for $port_0$ will be smaller due to limited connectivity offered by the topology, leading to a larger value of $E[Y]$.

This conclusion applies symmetrically to all ports, presenting the matching efficiency of NegotiaToR Matching, including at scale. We compare the theoretical results with the simulation results in Appendix A.1, validating their consistency.

NegotiaToR Matching is carefully orchestrated to keep simple while fitting in the DCN scenario, like long RTT and high traffic dynamics. Later in §3.5, we delve into the rationality of our design choices of NegotiaToR Matching, i.e., the minimalist approach like binary requests and no iteration. In §4, we give comprehensive evaluation results conducted on both the parallel network and thin-clos topology, showing that NegotiaToR Matching achieves high performance in both goodput and FCT across these topologies.

3.3 Network stack design of control and data planes

There are two expectations for NegotiaToR's network stack. First, it should provide high-frequency connection opportunities among all ToRs to ensure timely exchanges of NegotiaToR Matching scheduling messages. Second, it should provide most of the connection opportunities for the source-destination pairs with traffic demand to ensure high network goodput. To this end, as we presented in Figure 2, each NegotiaToR epoch is split into two phases, the predefined phase and the scheduled phase.

3.3.1 In-band pipelined scheduling. The first phase of each epoch frequently reconfigures its connections based on predefined matches and provides all-to-all connectivity. Due to the reconfiguration delay introduced by hardware and time synchronization error, each reconfiguration will take a while to finish, during which no bits can be sent. For this reason, NegotiaToR inserts guardbands between two timeslots in the predefined phase. Note that there is no reconfiguration in the scheduled phase, limiting the impact of guardbands on goodput.

The first phase comprises several short timeslots of equal length, and all source-destination pairs will get connected once. At the beginning of each timeslot, ToRs reconfigure the wavelengths of

their tunable lasers simultaneously according to predefined round-robin matches, establishing all-to-all connections for scheduling message exchange. To provide one round of all-to-all connection among N S-port ToRs, when forming a parallel network topology, it takes $\lceil \frac{N-1}{S} \rceil$ timeslots with the use of S AWGRs, each having N ports. When forming a thin-clos topology, it takes W timeslots using $\lceil \frac{NS}{W} \rceil$ AWGRs, each having W ($W \geq \lceil \frac{N}{S} \rceil$) ports.

Utilizing the all-to-all connectivity, ToRs get to exchange scheduling messages periodically. To this end, one approach is to finish one scheduling process and get the final scheduling results inside the same epoch. However, the strict front-back dependencies of each scheduling's REQUEST, GRANT, and ACCEPT steps require three rounds of round-robins to finish one scheduling process. Meanwhile, a step can start only after the scheduling messages from the previous step are received, thus introducing one-way delay (half RTT) between two consecutive steps. These factors together lead to an extremely long predefined phase, making the network degenerate into a traffic-oblivious design, thus damaging goodput.

Instead, NegotiaToR distributes one scheduling process's *request*, *grant*, and *accept* into three epochs. Figure 4 illustrates this pipelined workflow. This way, only one round of all-to-all connections is needed in each epoch, shortening the predefined phase. Since we set the scheduled phase to be long to provide enough on-demand sending opportunities, scheduling messages can reach destinations and get processed before the next epoch[3]. Each scheduling thus takes a minimum of around two epochs, facilitating a low scheduling delay. After the predefined phase, NegotiaToR can send data according to its non-conflicting scheduling results.

3.3.2 One-hop data transmission. The second phase establishes connections based on calculated matches, adapting the network to actual traffic demands. ToRs set the wavelengths of tunable lasers according to NegotiaToR Matching's scheduling results calculated in the first phase for data transmission.

In this phase, data in the corresponding per-destination queue is sent until the epoch ends or the queue empties, satisfying the traffic demands and contributing to a high goodput. Regarding latency, because of NegotiaToR's conflict-free scheduling, even though we aimed to design a low-latency algorithm and employ no iteration, a scheduling delay of at least two epochs is inevitable for a previously empty source-destination pair that has newly arrived data. Moreover, extra waiting delay will be introduced if this request is not granted. For elephant flows, this is acceptable. However, for mice flows that require low latency, this will essentially damage their FCT. FCT-oriented optimizations are needed.

3.4 Incast-optimized scheduling delay bypass

In DCNs, other than bandwidth-sensitive elephant flows, there are also latency-sensitive mice flows, which occupy a small portion of the total traffic volume but represent a large portion when measured by the number of flows [34]. These mice flows often appear as incasts where multiple mice flows arrive simultaneously in burst and compete for the limited scheduling opportunities. They require small FCT for good application performance, which is hard to achieve considering scheduling delays. The performance degradation worsens when incast happens.

[3]Note that the one-way delay between ToRs and the processing delay together may be longer than one epoch. In this case, the pipelined scheduling still works, only to expand to more epochs.

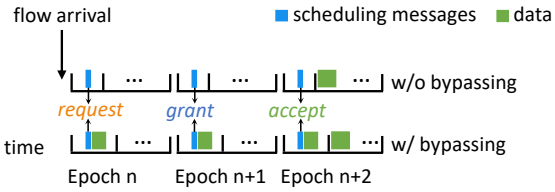


Figure 5: Bypassing scheduling delay with unscheduled transmission in the predefined phase.

3.4.1 Data piggybacking in the predefined phase. To optimize mice flows' FCT especially under incast, NegotiaToR offers unscheduled transmission opportunities that bypass the scheduling delay. In the first phase, NegotiaToR piggybacks one small data packet along with scheduling messages utilizing the predefined all-to-all connectivity, as we showed in Figure 5. Therefore, each ToR pair is guaranteed the transmission of at least one packet in every epoch, regardless of scheduling results.

We limit the size of piggybacked data to be small so that the predefined phase can still be short, ensuring that the majority of connections in one epoch are established in the scheduled phase based on actual traffic demands for high goodput. This is sufficient for mice flows considering their small size. Mice flows thus can be sent promptly via one-hop paths without suffering from scheduling delays, even under incasts, contributing to their small FCTs.

After deploying such a piggybacking mechanism, we apply a slight adjustment to NegotiaToR Matching—requests can only be sent when the queued data in per-destination queues exceeds three piggybacked packets instead of zero. This is because, during the scheduling delay, three packets are guaranteed to be sent in predefined phases of three consecutive epochs, regardless of the scheduling result. Raising the request threshold thus can avoid providing connections to source-destination pairs with no traffic demand, reducing bandwidth waste.

3.4.2 Priority queue for mice flow prioritization. Note that elephant flows in source ToRs, whose FCT is not our concern, may block mice flow data and occupy precious unscheduled transmission opportunities. In order to maximize the benefits of the piggybacked packet and mitigate such head-of-line blocking, NegotiaToR deploys mice flow priority mechanism in ToRs. We adopted an existing solution—the information-agnostic mice flow priority mechanism, PIAS [3]. By utilizing multi-level feedback queues, it can send mice flow data first without requiring prior knowledge of flow sizes, maintaining the practicality of NegotiaToR's design. This facilitates mice flows' timely transmission in both the predefined and scheduled phases.

With data piggybacking in the predefined phase and mice flow priority queues in ToRs, mice flow FCT is reduced, most of which is even below the scheduling delay. We evaluate the effectiveness of these two design elements through microbenchmarks, detailed in §4.2. So far, we have presented the design of NegotiaToR's network stack. We summarize the role of the two phases in Table 1.

	Predefined Phase	Scheduled Phase
In-band Control Plane	on-demand distributed scheduling	/
One-hop Data Transmission	unscheduled transmission (bypass scheduling delay)	scheduled transmission

Table 1: Two phases' role as the control plane and data plane in one NegotiaToR epoch.

3.5 The rationality of NegotiaToR's design choices

When designing the on-demand scheduling algorithm, we are committed to ensuring performance while reducing complexity for deployment practicality. This leads to our design choices towards a minimalist design, like binary requests and no iteration. This subsection delves into a common question: *could a more complex design significantly improve NegotiaToR's performance?* We examine this possibility, analyzing the reasoning behind our minimalist design, and demonstrate that extra complexity does not necessarily translate into proportionate performance gains. Note that we do not claim that our design is the ultimate solution. It remains an open problem to optimally balance complexity with performance, inviting further investigation and potential enhancements.

No iteration. NegotiaToR adopts a non-iterative design, distinguishing it from traditional iterative matching algorithms like PIM [2], RRM [31] and iSLIP [30]. Iteration can potentially improve matching efficiency and goodput, but also increases complexity and scheduling delays, especially in DCNs where the RTT between ToRs is long when put into the context of nanoseconds reconfiguration delay. With longer scheduling delays, the scheduling result is more likely to be outdated, since previous epochs may have already sent all the data during the long scheduling process, leaving the scheduled links empty, wasting bandwidth and thus adversely affecting performance.

We design an iterative version of NegotiaToR Matching, and investigate the impact of iteration through simulations. Details can be found in **Appendix A.2.1**. Simulation results confirm that using iteration is not a good idea to improve goodput for NegotiaToR. The iterative approach yields minimal or even negative improvements in goodput due to the outdated scheduling results, and consistently exhibits worse FCT due to longer scheduling delays. In contrast, simply using a 2× link rate speedup (i.e., the bandwidth ratio between uplinks and downlinks for each ToR is 2:1) upon the original non-iterative scheduling, which is a common practice in optical switching, achieves high goodput and superior FCT, making the iteration process unnecessary. As a result, we use 2× speedup in the following exploration.

No data relay. NegotiaToR chooses direct one-hop paths for all data, instead of traffic-oblivious relay [4, 33, 44]. There exists a third design choice, traffic-aware selective relay, which only enables data relay for elephant flows under light loads [8]. This avoids goodput damages of the traffic-oblivious one under heavy loads as well as mice flow FCT damages caused by more hops. By fully utilizing the empty links, it may potentially improve goodput for the thin-clos topology (Figure 1(b)) where the connectivity is limited.

To figure out if the performance gain is worth the added complexity, like intermediate ToR selection and congestion control for the relayed traffic, we carefully design a traffic-aware selective relay algorithm and conduct simulations on the thin-clos topology. Details can be found in **Appendix A.2.2**. Results show minor or no goodput gain, because NegotiaToR's goodput is already good at light loads, while the data relay barely helps at heavy loads. Considering the complexity of implementing data relay, we conclude that relay is not a good fit for NegotiaToR.

Binary requests. Another potential improvement is to include more information in the requests indicating priorities, instead of using binary demand information and simply prioritizing the least recently allocated pair with round-robin rings. We explore two approaches: one goodput-oriented approach that prioritizes the pairs with more pending data to improve link utilization, and one FCT-oriented approach that prioritizes the pairs with longer head-of-line packet waiting delays to reduce tail FCT.

To study if the benefits outweigh the additional complexities of size measurement, delay logging, and sorting, we evaluate the two approaches through simulations. We attach the details in **Appendix A.2.3**. Results indicate that for both approaches, the performance improvements are relatively modest compared with the added complexity. We thus conclude that binary requests are simple and effective enough for NegotiaToR.

Stateless scheduling. In our design, ToRs send requests solely according to real-time demands, without tracking ongoing requests from previous epochs. This can lead to over-scheduling the same source-destination pair, potentially resulting in under-utilized links and hurting goodput when all data has already been sent at the time of acceptance. Incorporating a stateful traffic matrix can mitigate this, but will introduce the complexity of maintaining the states, which reduces the robustness of the system especially under failures. Moreover, for lightly loaded scenarios, the impact of duplicate requests is negligible since the wasted links are not needed otherwise. For heavily loaded scenarios, since the data arrive at the sources continuously, whenever scheduled, the sources can always fully utilize the links, which can even decrease FCTs. The impact of stateless scheduling is thus minor.

To validate this, we designed and simulated a stateful version of NegotiaToR Matching, as detailed in **Appendix A.2.4**. The results confirm our analysis, showing a negligible difference between stateful and stateless scheduling. This justifies our design choice.

For now, we have found that the possibilities discussed above do not yield significant performance gains, and the added complexity is not worth the cost. Therefore, NegotiaToR converges to its present form, which is orchestrated towards a minimalist design. We acknowledge that our exploration has not covered the entire design spectrum. As DCN applications continue to develop, we believe there are opportunities for further optimizing NegotiaToR. We leave this as future work.

3.6 Practical considerations

When deploying NegotiaToR in real-world DCNs, it's crucial to address practical concerns such as link failure handling. This section delves into how NegotiaToR manages these practical problems.

3.6.1 Fault tolerance. NegotiaToR incorporates a mechanism for handling link failures. In the predefined phase, even when there is no scheduling message to send, we intentionally let each ToR send a dummy message to distinguish between link failures and lack of traffic. The loss of messages in the first phase indicates possible link failures. ToRs also employ this message to give feedback on whether they have received any bits in the reverse direction, along with port information if applicable.

This way, a ToR can effectively determine whether one port's egress or ingress link has failed[4]. In the predefined phase, each port in one ToR is expected to receive bits from multiple source ToRs. An ingress link failure may have happened if a ToR consistently fails to receive bits through a particular port. Similarly, on the sender side, repeated feedbacks of undelivered data originating from a specific port indicate an egress link fault. Upon detecting a link failure, ToRs will alert the maintainers and broadcast the detected failures to update their scheduling rules. This involves excluding the affected egress links and ingress links from data transmission. Once the faulty links are repaired, the transmission of scheduling messages can resume, and the corresponding links can be included in the scheduling again. ToRs can resume data transmission through them. As for the recovery of lost packets during handling the failure, NegotiaToR relies on upper-layer protocols, like TCP, for retransmission.

To further reduce link waste, for the parallel network topology, we also periodically change the round-robin rule in the predefined phase. This approach allows a pair of ToRs to exchange scheduling messages through multiple port-to-port links, instead of a specific one. Consequently, if one link experiences a failure, the exchange of scheduling messages with the corresponding subset of ToRs can still proceed through other links in subsequent epochs, guaranteeing all-to-all connectivity in the predefined phase. This ensures that the affected ToR pairs can still transmit data through the remaining functional links in the scheduled phase. For the connection-limited thin-clos topology where one source-destination pair can only communicate through identical ports, the same goal can be achieved by relaying scheduling messages and data to an intermediate ToR through ports that function normally. We test this fault detection and recovery mechanism on the parallel network topology in our evaluation (§4.3).

3.6.2 Scalable scheduling logic implementation. Since NegotiaToR Matching draws inspiration from RRM [31] that focuses on port matching inside crossbar packet switches, which is a well-mined area, it can benefit from applying proven implementation strategies in this field regarding hardware implementation, especially for the GRANT and ACCEPT rings. Switch port matching algorithms [30, 31] use programmable priority encoders [22] to construct round-robin arbiters to function as priority rings. One iteration of matching can be done in several clock cycles for a switch of hundreds of ports [26]. For the GRANT and ACCEPT rings in NegotiaToR Matching, the implementation is similar, but with a reduced complexity as the transition from the individual-switch scale to the DCN scale—this transition decentralizes the scheduling logic of ports from one single chip to multiple ToRs, and also extends the scheduling time budget compared to that inside switches. These factors collectively underscore the high practicality of implementing NegotiaToR Matching logic at scale.

3.6.3 End-to-end reconfiguration delay. NegotiaToR employs frequent network reconfigurations to serve the dynamic traffic demands among ToRs, thus preferring a shorter end-to-end reconfiguration delay to reduce guardband overhead. This calls for fast wavelength tuning hardware, fast clock and data recovery (CDR), and precise time synchronization so that a smaller guardband is sufficient to absorb the reconfiguration delay. Recent advancements in tunable lasers and CDR mechanisms have lowered the tuning delay to 10s of nanoseconds [4, 16, 49]. In particular, [4] employs disaggregated tunable lasers [43], along with amplitude caching [19] and clock phase caching [13, 14][5], and can limit tuning and CDR delays to under 10 nanoseconds.

As for time synchronization, recent studies have lowered synchronization errors to a few 10s of nanoseconds for a conventional packet-switched DCN [18]. For architectures like NegotiaToR where the AWGR is passive and no retiming or queueing latency jitter is involved, synchronization error can be further reduced. For example, [4] leverages round-robin all-to-all connections for synchronization, achieving errors within picoseconds. A primary clock is chosen, and ToRs periodically synchronize their local clocks and times with this primary. Similar mechanisms can be applied to NegotiaToR using the round-robin connections in the predefined phase. After synchronization, the clocks will slowly drift [18], and a guardband of several nanoseconds is adequate to absorb the drift till the next synchronization in the next predefined phase.

Together, current technologies can facilitate an end-to-end reconfiguration delay of 10 nanoseconds as demonstrated in [4], leading to our main reconfiguration delay setting in §4. Note that even if the guardband is enlarged to 100 nanoseconds, NegotiaToR remains effective with appropriate parameter modifications. We evaluate the influence of different guardbands in §4.2.

3.6.4 Epoch length setting considerations. We discuss the length settings of the two phases of one NegotiaToR epoch. First of all, we do not want the predefined phase to dominate, as the connections are irrelevant to current traffic demands and will downgrade to pure round-robin, resulting in a huge performance drop. Conversely, an extremely short predefined phase limits opportunities for scheduling delay bypassing, adversely affecting mice flows. Meanwhile, a scheduled phase shorter than one-way delay between ToRs is also undesirable due to increased scheduling delay. Furthermore, for a low scheduling delay and high scheduling frequency, the epoch length should not be excessively long, which leads to increased FCT and possible link waste due to outdated scheduling results.

To minimize overhead, guardbands before wavelength tunings are desired to account for 10% or less of all time. For a low wavelength tuning delay of 10 nanoseconds [4], it's rather easy to reach this goal. For longer guardbands, extending the scheduled phase proportionally, which does not involve reconfigurations, also meets this target. This leads to our default epoch length settings in evaluation (§4.1). We conduct simulations to show the influence of

[4]We detect faults of egress and ingress separately to prevent overreaction and simplify maintenance tasks.

[5]For NegotiaToR, ToRs read the *accept* packets to access the connections to be established, so that they can directly use the cached parameters for fast CDR.

different guardbands in §4.2. In §4.4, parameter sensitivity experiments are also conducted to further understand the impact of epoch length settings on performance.

3.6.5 Traffic management below ToRs. Our design focused on the interconnections above ToRs. Here, we discuss possible methods to do traffic control below ToRs. Traffic from hosts to ToRs is buffered at the ToR before transmission through the optical fabric. Backpressure-based or credit-based flow control can help to avoid packet drops, which stop corresponding data transmission when the buffer is full. On the receiver side, data arriving at destination ToRs are buffered before transmission to hosts. Since we introduce speedup in the optical fabric, and data destined for the same host may arrive synchronously at the ToR through multiple ports, the corresponding queue in the receiver side ToR may accumulate and cause packet drops. To cope with this, ToRs should monitor the length of this queue and only allow data transmission when buffer space is enough. This slight modification thus reduces packet drops.

To ensure reliability, we can run TCP-like protocols over NegotiaToR, which typically involve reordering at the receiver side. NegotiaToR's design inherently results in fewer out-of-order packets. For one source-destination pair, as long as the packets are moved from the per-destination queue to per-port queues in order at the source ToR (e.g., moving to lower-index port first if there are multiple available ports), and then consumed in the same order at the destination ToR, the order of packets will be preserved.

4 EVALUATION

We investigate the effectiveness of NegotiaToR's design elements, and evaluate its performance through large-scale simulations using a packet-level simulator, YAPS [17]. Simulations on the parallel network topology and the thin-clos topology are done, validating the generality of NegotiaToR on flat topologies.

4.1 Evaluation setup

Network setup. The network consists of 128 8-port ToRs, meaning each ToR connects to the optical fabric through 8 ports. We connect the same set of ToRs with the parallel network topology (Figure 1(a)) with 8 128-port AWGRs, and the thin-clos topology (Figure 1(b)) with 64 16-port AWGRs, respectively. One-way propagation delay between ToRs is $2\mu s$. The hosts under the same ToR have an aggregated bandwidth of 400 Gbps, and we provide a 2× link rate speedup to ToR uplink ports (i.e., 100 Gbps per port, and 8×100 Gbps total), as discussed in §3.5. Focusing on ToR interconnections, we consider ToRs as endpoints. FCT and goodput measurements are taken from the ToRs' perspective, marking the start and end of flows at the ToRs. Unless specified, data piggybacking (PB) in the predefined phase for scheduling delay bypass and priority queues (PQ) at sources for mice flow prioritization (§3.4) are both enabled. For PQ, we set three priorities: the first $1KB$ of flow data will be sent first, then the following $9KB$, and then the rest of the bits.

Baseline. We compare the performance of NegotiaToR with traffic-oblivious proposals on the same scale network (i.e., 128 8-port ToRs), following Sirius [4] to implement the state-of-the-art benchmark on the same simulator. Note that while NegotiaToR can be customized for different flat topologies, Sirius is specifically designed for the thin-clos topology. Its relay-enabled round-robin scheduling cannot utilize the sufficient connectivity of the parallel networks, resulting in identical performance on both topologies. **Therefore, for brevity, we only show the results on the thin-clos topology for the traffic-oblivious scheme**. 2× speedup and priority queue for mice flow prioritization are also enabled. Since the multi-level-feedback-queue based prioritization [3] does not apply to data at intermediate nodes, we only enable it at sources.

Epoch settings. Based on [4], the guardband for reconfiguration in NegotiaToR is also set to $10ns$ unless specified. By default, in the predefined phase, each timeslot takes $60ns$, comprising a $10ns$ guardband and $50ns$ for transmitting NegotiaToR Matching scheduling messages along with data packet header ($30B$ each, including $request$, $grant$, and $accept$) plus a data payload ($595B$). During the scheduled phase, no reconfiguration or guardband is required, with each timeslot lasting $90ns$ for sending one data packet (including a $10B$ header). We set the length of the scheduled phase to 30 timeslots to balance goodput and FCT unless specified. Consequently, for both topologies, the predefined phase takes $16 * 60ns = 0.96\mu s$, and the scheduled phase takes $30 * 90ns = 2.7\mu s$. This leads to an epoch size of $3.66\mu s$, where the guardbands account for 4.37%. We assume that the one-way delay between ToRs ($2\mu s$) and the calculation delay together is lower than one epoch, leading to a scheduling delay of around two epochs.

Workload characteristics. Unless specified, we generate the workload after published DCN traces collected from Meta's Hadoop clusters [41]. The trace is highly tailed, where 60% of the flows are less than $1KB$, while more than 80% of the bits are from elephant flows larger than $100KB$. All the flows arrive based on a Poisson process, with sources and destinations chosen uniformly at random. We define the network load as $L = \frac{F}{R \cdot N \cdot \tau}$. F is the mean flow size, R is the per-ToR bandwidth, N is the number of ToRs, and τ is flows' inter-arrival time. Since we view the network as starting from ToRs instead of hosts, R is the aggregated bandwidth of the hosts under one ToR, which is 400 Gbps. We test the network's goodput and FCT under various loads, ranging from 10% to 100%. Other than the Hadoop workload, NegotiaToR's performance under various workloads is also evaluated later in §4.4.

Evaluation metric. Unless noted, we simulate a real-world duration of $30ms$, and focus on 99th-percentile mice flow FCT, as well as the average goodput of all ToRs. Flows less than $10KB$ are regarded as mice flows. If not stated otherwise, goodput is normalized to the aggregated bandwidth of the hosts under one ToR (400 Gbps).

4.2 Microbenchmarks

We first understand the advantages of NegotiaToR's design elements through microbenchmarks.

The effectiveness of NegotiaToR's scheduling delay bypassing designs. We conduct ablation studies on our designs for bypassing scheduling delays, i.e., data piggybacking (§3.4.1) and mice flow prioritization (§3.4.2). The results are presented in Table 2. When data piggybacking is disabled, the timeslot in the predefined phase is shortened with only reconfiguration delays and scheduling messages left, and the scheduled phase is enlarged to keep

	Mice Flow FCT in Epochs (99p/Average)	
	Parallel Network	Thin-Clos
-	732.4/42.1	1216.4/75.0
PB	418.5/19.9	847.9/45.3
PQ	21.0/5.7	26.4/5.7
PB and PQ	6.0/1.6	6.5/1.6

Table 2: NegotiaToR's mice flow FCT at 100% load, with data piggyback (PB) in the predefined phase and priority queues (PQ) separately enabled and disabled.

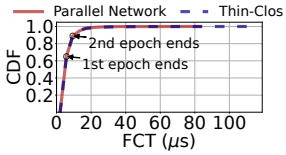
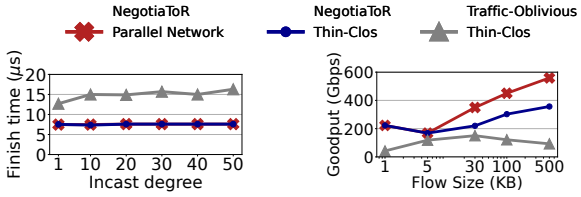


Figure 6: CDF of NegotiaToR's mice flow FCT at 100% load.



(a) Incast finish time

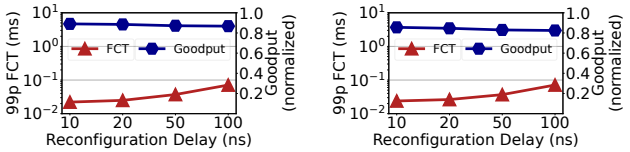
(b) Average goodput under all-to-all

Figure 7: Performance under incast and all-to-all loads.

the epoch length the same, accordingly maintaining the reconfiguration overhead ratio the same. At 100% load, enabling only data piggybacking itself largely reduces the 99th percentile FCT of mice flows compared to no optimization. When combined with priority queues, the head-of-line blocking of elephant flows is mitigated, allowing unscheduled piggybacking opportunities to be better utilized by mice flows. For both topologies, mice flow is thus further optimized, where the average FCT drops to 1.6 epochs, which is even less than the roughly 2-epoch scheduling delay.

To delve deeper into NegotiaToR's mice flow performance, we show the CDF of mice flow FCT at 100% load with both PB and PQ enabled in Figure 6. Both topologies provide identical connectivity in the predefined phase, leading to the overlapping of two lines for smaller FCTs. Across two topologies, over 80% of mice flows successfully bypass the scheduling delay, finishing within 2 epochs (the second turning point). This validates the effectiveness of our designs of bypassing scheduling delays. Mice flow performance thus can be guaranteed even under heavy loads.

Incast finish time. To further verify NegotiaToR's incast-optimized scheduling delay bypass design, we test incast workloads with varying incast degrees on NegotiaToR and the traffic-oblivious scheme, where a set of ToRs synchronously send one $1KB$ flow to the same ToR, and the number of source ToRs is the degree. Results are shown in Figure 7(a). Compared with the traffic-oblivious scheme, NegotiaToR consistently finishes the incast earlier at roughly the same time by piggybacking data in the predefined phase without scheduling, regardless of the incast degree. Note that NegotiaToR achieves almost the same incast finish time on two topologies. Again, this



(a) Goodput and mice flow FCT on parallel network

(b) Goodput and mice flow FCT on thin-clos

Figure 8: NegotiaToR under various reconfiguration delays at 100% load.

is because both topologies provide identical connectivity in the predefined phase. To gain further insights, we observe the goodput at the receivers' side, with results presented in Appendix A.3.

All-to-all goodput. NegotiaToR Matching can provide sufficient connectivity for traffic in an on-demand manner, even at heavy loads. We test all-to-all workloads with varying flow sizes to verify this, where each ToR synchronously sends equal-sized flows to all other ToRs. Average goodput during the transmission is shown in Figure 7(b). For heavier loads, NegotiaToR fully employs the 2× speedup, achieving much higher goodput. Its goodput on the parallel network is better than that on the thin-clos topology because of the latter's limited connectivity: with flows completing, the thin-clos topology experiences underutilization of links, whereas links in the parallel network maintain higher utilization. In contrast, even though with a 2× speedup, the goodput of the traffic-oblivious scheme is limited since the network is flooded by relayed traffic, which competes for receivers' bandwidth, thus damaging goodput, especially at heavy loads. We also make micro-observations, with detailed results presented in Appendix A.3.

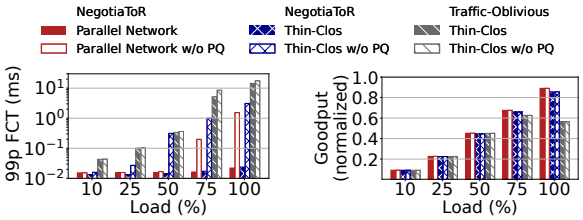
Adaptability to various reconfiguration delays. Despite $10ns$ end-to-end reconfiguration delays already being realized [4], we also evaluate NegotiaToR's performance under longer reconfiguration delays at 100% load. The length of the scheduled phase is accordingly adjusted to control the reconfiguration overhead. The findings, illustrated in Figure 8, reveal that with longer reconfiguration delays, NegotiaToR can still achieve good performance, showing the good generality of its design.

4.3 Main results

Now that we have investigated NegotiaToR's design elements, we evaluate its overall performance.

FCT and goodput on both topologies. We compare NegotiaToR's mice flow FCT and goodput with the traffic-oblivious scheme on both topologies. Considering real-world deployment practicality, we also show the results when the priority queue for mice flow prioritization is disabled.

NegotiaToR achieves better mice flow FCT at all loads regardless of topologies, as in Figure 9(a). With priority queues enabled, NegotiaToR's FCT is consistently one to two orders of magnitude better. Even without priority queues, such significant advantages still exist at lighter loads. This is because, for the traffic-oblivious scheme, the detouring of relayed traffic damages mice flow FCT, especially when elephant flows are spread across the network and block mice flows at intermediate nodes. While for NegotiaToR, the on-demand



(a) Mice flow FCT (b) Goodput

Figure 9: FCT and goodput at various loads. Results with priority queues (PQ) for mice flow prioritization enabled and disabled are both shown.

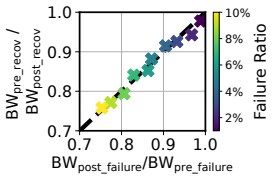


Figure 10: NegotiaToR's bandwidth usage changes during link failure and recovery.

scheduling together with the scheduling delay bypassing enables prompt transmission of mice flows.

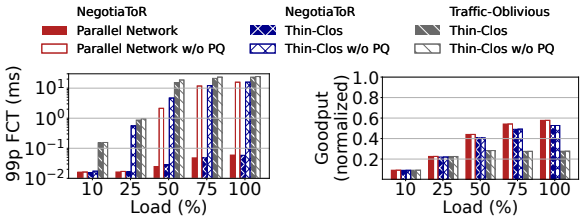
Regarding goodput, NegotiaToR remarkably outperforms the traffic-oblivious scheme on both topologies at high loads, as in Figure 9(b). Under low loads, the traffic-oblivious scheme achieves high goodput by utilizing empty links to relay traffic. As the load increases, relayed traffic saturates the network, competing for bandwidth and becoming a bottleneck for goodput. In contrast, NegotiaToR's on-demand scheduling can better utilize the network's bandwidth and reduce bandwidth waste.

Through Figure 9, we also see that no matter which topology is used, NegotiaToR maintains comparable performance under identical parameter settings, where the performance on the thin-clos topology is marginally lower than on the parallel network due to its limited connectivity. These results underscore that NegotiaToR can adapt well to various flat topologies.

Fault tolerance. We test the effectiveness of NegotiaToR's fault tolerance mechanism. In the 30*ms* real-world duration, we simulate different levels of simultaneous link failures on the parallel network topology, recover them, and show the bandwidth usage changes in Figure 10. Since a single egress or ingress link failure will affect all traffic passing through it, this will lead to disproportional bandwidth reduction. With 1% of links failing, NegotiaToR's bandwidth usage drops to 98.9%, while a 10% failure rate leads to 75.3%. Upon link recovery, the bandwidth usage returns to its pre-failure level. This validates NegotiaToR's robust fault tolerance capability. We also make micro-observations to better understand its behavior under link failures, with results shown in Appendix A.4.

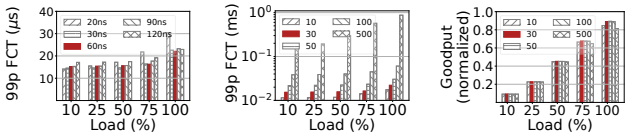
4.4 Deep dive results

We further evaluate NegotiaToR under various scenarios to better understand its performance.



(a) Mice flow FCT (b) Goodput

Figure 11: FCT and goodput at various loads with no speedup.



(a) Sensitivity of predefined phase timeslot duration: mice flow FCT

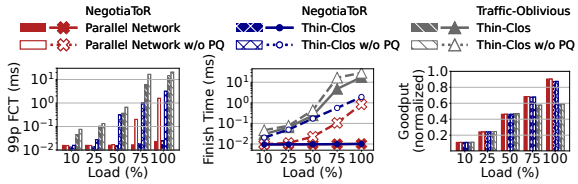
(b) Sensitivity of scheduled phase length (in timeslots): mice flow FCT and goodput

Figure 12: FCT and goodput under various parameter settings on the parallel network topology. Bars marked in red indicate the setting the evaluation uses by default.

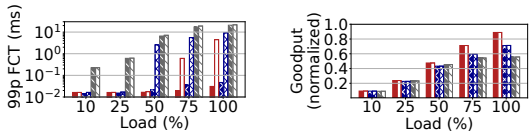
Performance under constrained bandwidth. Previous evaluations are conducted with a 2× speedup. Here we remove this speedup, provide identical bandwidth to ToR uplinks and downlinks, and test NegotiaToR and the traffic-oblivious scheme's performance under the same workload with §4.3. Simulations are done on both the parallel network and thin-clos topologies. Results illustrated in Figure 11 align with previous findings. NegotiaToR's on-demand scheduling can better exploit the constrained bandwidth and achieve good performance, highlighting its practicality.

Parameter sensitivity experiment. We investigate the impact of the length of two phases in NegotiaToR by adjusting one at a time. We show experiments conducted on the parallel network under the workload used in §4.3 as an example. We first adjust the duration of each timeslot in the predefined phase, which affects the amount of data that can be piggybacked without scheduling, ranging from 20ns to 120ns including a 10ns guardband. We show the mice flow FCT in Figure 12(a), and the goodput is omitted because the difference with Figure 9(b) is minor. Subsequently, we adjust the length of the scheduled phase from 10 to 500 timeslots, and present the performance in Figure 12(b). The results match our analysis in §3.6.4 well. They also demonstrate that even when the parameters are approximately set around their optimal values, the impact on NegotiaToR's performance is minor, showing its robustness.

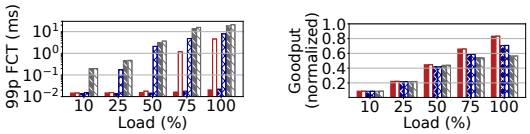
Performance under various workloads. We further evaluate NegotiaToR under various workloads while maintaining the same epoch length setting as before. We first randomly mix incasts on top of the workload used in §4.3 to mimic bursty traffic, where each incast has a degree of 20 and a flow size of 1*KB*, and all incasts take 2% of ToR's aggregated downlink bandwidth. Performance of background traffic and incasts are shown in Figure 13(a). Results



(a) Performance under Hadoop [41] mixed with incast. From left to right is mice flow FCT of the background traffic, average incast finish time, and overall goodput



(b) Mice flow FCT and goodput under web search workload [1]



(c) Mice flow FCT and goodput under the aggregated traffic from Google datacenter [34, 46]

Figure 13: FCT and goodput under more workloads.

indicate that with the incast-optimized scheduling delay bypassing mechanism, NegotiaToR can serve incasts well with minor impact on the background traffic.

Additionally, we test NegotiaToR under two more workloads. The heavier web search [1] workload where more than 80% flows exceed $10KB$, and the lighter workload from Google datacenter [34, 46] where more than 80% flows are less than $1KB$. Even though without parameter fine-tuning, results in Figure 13(b) and Figure 13(c) show consistent FCT and goodput advantages of NegotiaToR as before, emphasizing the efficiency of its on-demand scheduling.

5 RELATED WORK

Matching algorithms. On-demand scheduling on reconfigurable flat topologies can be viewed as a matching problem. Fast matching algorithms such as PIM [2], RRM [31] and iSLIP [30] employ iterations of request, grant, and accept for input-output port connection scheduling in crossbar packet switches. Unlike the mesh connections of the scheduling logic for ports inside a switch, ToRs in reconfigurable DCNs are multi-ported and interconnected through various topologies, demanding topology-adaptive matching. NegotiaToR draws inspiration from them and is uniquely tailored for ToR matching. While maintaining feasibility, it can adapt to various flat topologies. Pipelined scheduling and scheduling delay bypassing are designed to accommodate the long RTT between ToRs.

There are also other practices for using request-grant based or receiver-driven algorithms in DCNs. Packet-switched schemes, like dcPIM [7], ExpressPass [12] and Homa [34] focus on single-port host-level matching, aiming to control the congestion in the network. They utilize over-commitment to optimize network utilization, contrasting the conflict-free requirement of reconfigurable DCNs. On-demand scheduling for reconfigurable DCNs has also seen similar approaches. For example, ProjecToR [21] employs a request-grant based algorithm to schedule optical links among ToRs. However, it requests at a per-port granularity and leads to minor scalability, and also adds complexity by measuring the waiting delays of bundles of packets at sources for priority decision, whereas NegotiaToR pursues a minimalist approach with binary per-ToR requests and no need for delay measurement. Through experimental explorations including ProjecToR (Appendix A.2.5), we highlight the effectiveness of NegotiaToR's design.

Connect ToRs with reconfigurable networks. The networks need to accommodate both short FCT and high goodput with good deployment practicality, serving the dynamic traffic demands between ToRs. When the reconfiguration delay of the switching hardware is long (like milliseconds to microseconds) compared with RTT, FCT is a main concern. Proposals use a hybrid packet-switched network [15, 23, 29, 33, 50] or multi-hop routing [10, 21, 23, 32] to optimize mice flow FCT. For instance, Opera [32] reconfigures its topology to a set of pre-calculated expander graphs, enabling mice flows to be immediately sent through multi-hop paths. This approach allows Opera to deliver low mice flow FCT in the order of tens of microseconds with the optical network alone. However, due to hardware reconfiguration delay limitations, its direct connections still mismatch the real-time traffic demands, posing challenges in accommodating the dynamic traffic.

With recent advancements in fast switching hardware featuring nanoseconds reconfiguration delay, opportunities have been seen to serve the dynamic traffic. Previous on-demand scheduling proposals like PULSE [5] ensure performance, but raise scalability concerns due to complex scheduling logic. Sirius [4] utilizes the traffic-oblivious method combined with data relay, which, although simple, encounters difficulties in maintaining high goodput under heavy loads due to bandwidth competition caused by relayed traffic, and in providing short FCT due to the relay latency. In contrast, NegotiaToR aims to achieve on-demand scheduling over fast switching hardware through a minimalist design. With distributed matching, pipelined scheduling and incast-optimized scheduling delay bypassing schemes, NegotiaToR offers short FCT and high goodput through one-hop transmission while maintaining low complexity, further exploiting the fast reconfiguration capability.

6 CONCLUSION

We presented NegotiaToR, an on-demand reconfigurable DCN architecture with a simple design. With the two-phase epoch, it runs NegotiaToR Matching distributedly in-band to reconfigure the network according to dynamic real-time traffic demands. It also provides an incast-optimized scheduling delay bypassing scheme to mitigate the impact of scheduling delays. NegotiaToR is compatible with prevalent flat topologies, and is tailored towards a minimalist design for on-demand reconfigurable DCNs, enhancing practicality. By exploiting the fast optical switching technology to provide high performance with low complexity, we hope that NegotiaToR can facilitate the development of next-generation reconfigurable DCNs.

ACKNOWLEDGMENTS

We thank our shepherd, Alex Snoeren, and the anonymous SIGCOMM reviewers for their useful feedback on this paper. Yong Cui (cuiyong@tsinghua.edu.cn) is the corresponding author. This work was supported by the National Natural Science Foundation of China under Grants 62132009, 62221003 and 62272292.

REFERENCES

[1] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). In *Proceedings of SIGCOMM*.

[2] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. 1993. High Speed Switch Scheduling for Local Area Networks. *ACM Trans. Comput. Syst.* 11, 4 (1993).

[3] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. 2015. Information-Agnostic Flow Scheduling for Commodity Data Centers. In *Proceedings of NSDI*.

[4] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, István Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. In *Proceedings of SIGCOMM*.

[5] Joshua L. Benjamin, Thomas Gerard, Domaniç Lavery, Polina Bayvel, and Georgios Zervas. 2020. PULSE: Optical Circuit Switched Data Center Architecture Operating at Nanosecond Timescales. *Journal of Lightwave Technology* 38, 18 (2020), 4906–4921.

[6] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. 2011. MicroTE: fine grained traffic engineering for data centers. In *Proceedings of CoNEXT*.

[7] Qizhe Cai, Mina Tahmasbi Arashloo, and Rachit Agarwal. 2022. dcPIM: near-optimal proactive datacenter transport. In *In Proceedings of SIGCOMM*.

[8] Peirui Cao, Shizhen Zhao, Min Yee Teh, Yunzhuo Liu, and Xinbing Wang. 2021. TROD: Evolving From Electrical Data Center to Optical Data Center. In *Proceedings of INFOCOM*.

[9] Cheng-Shang Chang, Duan-Shin Lee, and Yi-Shean Jou. 2002. Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering. *Computer Communications* 25, 6 (2002).

[10] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2012. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In *Proceedings of NSDI*.

[11] Stanley Cheung, Tiehui Su, Katsunari Okamoto, and S. J. B. Yoo. 2014. Ultra-Compact Silicon Photonic 512×512 25 GHz Arrayed Waveguide Grating Router. *IEEE Journal of Selected Topics in Quantum Electronics* 20, 4 (2014).

[12] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In *Proceedings of SIGCOMM*.

[13] Kari Clark, Hitesh Ballani, Polina Bayvel, Daniel Cletheroe, Thomas Gerard, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, Philip Watts, Hugh Williams, Georgios Zervas, Paolo Costa, and Zhixin Liu. 2018. Sub-Nanosecond Clock and Data Recovery in an Optically-Switched Data Centre Network. In *Proceedings of ECOC*.

[14] Kari Clark, Daniel Cletheroe, Thomas Gerard, Istvan Haller, Krzysztof Jozwik, Kai Shi, Benn Thomsen, Hugh Williams, Georgios Zervas, Hitesh Ballani, Polina Bayvel, Paolo Costa, and Zhixin Liu. 2020. Synchronous subnanosecond clock and data recovery for optically switched data centres using clock phase caching. *Nature Electronics* (June 2020).

[15] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of SIGCOMM*.

[16] Alex Forencich, Valerija Kamchevska, Nicolas Dupuis, Benjamin G. Lee, Christian W. Baks, George Papen, and Laurent Schares. 2020. A Dynamically-Reconfigurable Burst-Mode Link Using a Nanosecond Photonic Switch. *Journal of Lightwave Technology* 38, 6 (2020), 1330–1340.

[17] Peter X Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, and Scott Shenker. 2015. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric. In *Proceedings of CoNEXT*.

[18] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. 2018. Exploiting a Natural Network Effect for Scalable, Fine-grained Clock Synchronization. In *Proceedings of NSDI*.

[19] Thomas Gerard, Kari Clark, Adam Funnell, Kai Shi, Benn Thomsen, Philip Watts, Krzysztof Jozwik, Istvan Haller, Hugh Williams, Paolo Costa, and Hitesh Ballani. 2021. Fast and Uniform Optically-Switched Data Centre Networks Enabled by Amplitude Caching. In *In Proceedings of OFC*.

[20] Ehab Ghabashneh, Yimeng Zhao, Cristian Lumezanu, Neil Spring, Srikanth Sundaresan, and Sanjay Rao. 2022. A Microscopic View of Bursts, Buffer Contention, and Loss in Data Centers. In *In Proceddings of IMC*.

[21] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *Proceedings of SIGCOMM*.

[22] P. Gupta and N. McKeown. 1999. Designing and implementing a fast crossbar scheduler. *IEEE Micro* 19, 1 (1999), 20–28.

[23] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-Space Optics. In *Proceedings of SIGCOMM*.

[24] Y. Hida, Y. Hibino, T. Kitoh, Y. Inoue, M. Itoh, T. Shibata, A. Sugita, and A. Himeno. 2001. 400-channel 25-GHz spacing arrayed-waveguide grating covering a full range of C- and L-bands. In *Proceedings of Optical Fiber Communication Conference and International Conference on Quantum Information*.

[25] Torsten Hoefler, Duncan Roweth, Keith Underwood, Bob Alverson, Mark Griswold, Vahid Tabatabaee, Mohan Kalkunte, Surendra Anubolu, Siyuan Shen, Abdul Kabbani, Moray McLaren, and Steve Scott. 2023. Datacenter Ethernet and RDMA: Issues at Hyperscale. arXiv:2302.03337

[26] Chun Kit Hung, M. Hamdi, and Chi-Ying Tsui. 2003. Design and implementation of high-speed arbiter for large scale VOQ crossbar switches. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*.

[27] Saurabh Jha, Archit Patke, Jim Brandt, Ann Gentile, Benjamin Lim, Mike Showerman, Greg Bauer, Larry Kaplan, Zbigniew Kalbarczyk, William Kramer, and Ravi Iyer. 2020. Measuring Congestion in High-Performance Datacenter Interconnects. In *Proceedings of NSDI*.

[28] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. In *Proceedings of ISCA*.

[29] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. 2014. Circuit Switching under the Radar with REACToR. In *Proceedings of NSDI*.

[30] Nick McKeown. 1999. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking (ToN)* 7, 2 (1999).

[31] Nicholas William McKeown. 1995. *Scheduling algorithms for input-queued cell switches*. University of California, Berkeley.

[32] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *Proceedings of NSDI*.

[33] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C Snoeren, and George Porter. 2017. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of SIGCOMM*.

[34] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities. In *Proceedings SIGCOMM*.

[35] G.E. Moore. 1998. Cramming More Components Onto Integrated Circuits. *Proc. IEEE* 11, 3 (1998).

[36] Samuel K. Moore. 2019. Another step toward the end of Moore's law: Samsung and TSMC move to 5-nanometer manufacturing - [News]. *IEEE Spectrum* (2019).

[37] Tomonobu Niwa, Hiroshi Hasegawa, Ken-Ichi Sato, Toshio Watanabe, and Hiroshi Takahashi. 2012. Large Port Count Wavelength Routing Optical Switch Consisting of Cascaded Small-Size Cyclic Arrayed Waveguide Gratings. *IEEE Photonics Technology Letters* 24, 22 (2012).

[38] George Porter, Richard D. Strong, Nathan Farrington, Alex Forencich, Pang-Chen Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. Integrating microsecond circuit switching into the data center. In *Proceedings of SIGCOMM*.

[39] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Muhammad Mukarram Bin Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve D. Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of SIGCOMM*.

[40] Roberto Proietti, Yawei Yin, Runxiang Yu, Christopher J. Nitta, Venkatesh Akella, Christopher Mineo, and S. J. Ben Yoo. 2013. Scalable Optical Interconnect Architecture Using AWGR-Based TONAK LION Switch With Limited Number of Wavelengths. *Journal of Lightwave Technology* 31, 24 (2013).

[41] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proceedings of SIGCOMM*.

[42] Ken-Ichi Sato. 2023. Optical Switching will Innovate Intra Data Center Networks. In *In Proceedings of OFC*.

[43] Kai Shi, Sophie Lange, Istvan Haller, Daniel Cletheroe, Raphael Behrendt, Benn Thomsen, Fotini Karinou, Krzysztof Jozwik, Paolo Costa, and Hitesh Ballani. 2019. System demonstration of nanosecond wavelength switching with burst-mode PAM4 transceiver. In *Proceedings of ECOC*.

[44] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. 2019. Shoal: A Network Architecture for Disaggregated Racks. In *Proceedings of NSDI*.

[45] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proceedings of SIGCOMM*.

[46] R. Sivaram. 2008. *Some Measured Google Flow Sizes.* Google internal memo, available on request.
[47] Min Yee Teh, Zhenguo Wu, Madeleine Glick, Sebastien Rumley, Manya Ghobadi, and Keren Bergman. 2022. Performance trade-offs in reconfigurable networks for HPC. *Journal of Optical Communications and Networking* 14, 6 (2022).
[48] Leslie G. Valiant and Gordon J. Brebner. 1981. Universal Schemes for Parallel Communication. In *Proceedings of STOC.*
[49] Théo Verolet, Sylvain Almonacil, Mijail Szczerban, José Manuel Estarán, Rajiv Boddeda, Fabien Boitier, Jean-Guy Provost, Haik Mardoyan, Fabrice Blache, Jean Decobert, Ségolène Olivier, Alexandre Shen, and Sébastien Bigo. 2020. Ultra-Fast Tunable Laser Enabling 4 ns Coherent Slot Switching Beyond 100 Gbit/s. In *Proceedings of ECOC.*
[50] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. 2010. C-Through: Part-Time Optics in Data Centers. In *Proceedings of SIGCOMM.*
[51] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. 2023. TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs. In *Proceedings of NSDI.*
[52] Yawei Yin, Roberto Proietti, Christopher J. Nitta, Venkatesh Akella, Christopher Mineo, S.J.B. Yoo, and Ke Wen. 2013. AWGR-based all-to-all optical interconnects using limited number of wavelengths. In *Proceedings of Optical Interconnects Conference.*
[53] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2017. High-resolution measurement of data center microbursts. In *Proceedings of IMC.*
[54] Shizhen Zhao, Peirui Cao, and Xinbing Wang. 2022. Understanding the Performance Guarantee of Physical Topology Design for Optical Circuit Switched Data Centers. In *Proceedings of SIGMETRICS.*

A APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A.1 Efficiency analysis validation of NegotiaToR Matching

To validate the theoretical results of NegotiaToR Matching's efficiency in §3.2.2, we examine the scheduling efficiency in our large-scale simulations at 100% load in §4.3, since heavy-load scenarios are closer to the intense-competition scenario we assumed in the theoretical model. For each epoch, we record the ratio of source-destination level accepts and grants (we call it match ratio) of all ToRs, and depict it in Figure 14. Results show that the actual match ratio is consistent with our expectations, with thin-clos topology ($n = 16, E[Y] = 0.644$) achieving a slightly higher match ratio than the parallel network topology ($n = 128, E[Y] = 0.634$).

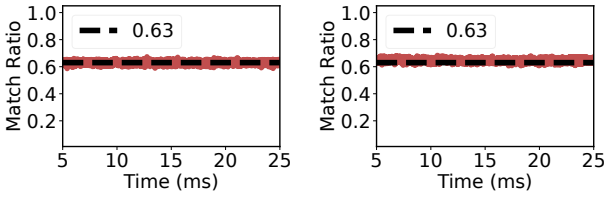
A.2 Details of the experimental exploration of other design choices

NegotiaToR's design is tailored towards simplicity for practicality. To find out whether a little more complexity can bring a significant performance gain, we explored some other design choices. In §3.5, we found that iterative scheduling, traffic-aware selective relay, informative requests, and stateful scheduling may not provide proportionate performance gains. We give the details of our exploration in this section. For simplicity, we only show the results of the parallel network topology, except for traffic-aware selective relay, which is designed for the thin-clos topology. Unless specified, simulation follows the default settings in §4.1.

A.2.1 Iterative NegotiaToR Matching. The impact of introducing iteration to NegotiaToR Matching is twofold. On one hand, iteration could enhance matching efficiency by fully utilizing unmatched ports, potentially improving goodput. However, in optical switching, lower goodput can often be effectively compensated by speedup (i.e., providing more aggregated uplink bandwidth than aggregated downlink bandwidth). On the other hand, introducing iteration significantly increases the scheduling delay due to the long RTT between ToRs, and also adds complexity. Traffic demand information is more likely to be outdated by the time the scheduling decision is received, since data may have already been sent by previous epochs during the long scheduling process, leading to link waste and thus performance degradation. Therefore, although the scheduling result may be closer to a maximal match, FCT and goodput could be adversely affected.

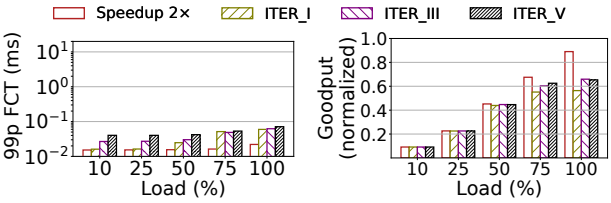
For investigation, we design an iterative version of NegotiaToR Matching. After deriving the *accept*, instead of directly sending data, new *request* is sent to destinations again along with indices of unmatched ports for further iterations. Multiple rounds of iteration also enlarge the scheduling delay. For one more iteration, the scheduling delay is enlarged by three epochs.

Simulation results. We conduct simulations of both an iterative and a non-iterative original version of NegotiaToR Matching on the parallel network topology. For the non-iterative one, we set the speedup factor to 2, as depicted in §4.1. For the iterative version, we try 3 and 5 rounds of iterations with no speedup. We set the size of scheduling messages (indicated by packet header size in



(a) Parallel network (b) Thin-clos

Figure 14: NegotiaToR's match ratio for epochs in both topologies, recorded in the simulation at 100% load in §4.3.



(a) Mice flow FCT (b) Goodput

Figure 15: FCT and goodput on the parallel network topology at various loads, with 2× speedup or iteration enabled.

simulation) to be the same for fair comparison, even though the iterative algorithm actually needs extra bits to transmit port-level occupation information, which leads to extra overhead.

Results in Figure 15 show that, as we expected, iterative scheduling adversely affects FCT at all loads because of the long scheduling delay. Meanwhile, with more rounds of iteration, goodput starts to decrease due to the outdated traffic demand information. Notably, at all loads, goodput is consistently equal to or lower than the non-iterative version with 2× speedup. This implies that instead of introducing iteration, it is more effective to increase the speedup factor to compensate for the possibly low matching efficiency, which also improves the FCT performance. Therefore, we conclude that iteration is not an optimal choice for NegotiaToR Matching.

A.2.2 Traffic-aware selective relay. In NegotiaToR, all data is sent through one-hop paths. Avoiding traffic-oblivious relay helps NegotiaToR avoid goodput damages under heavy loads caused by the doubling of data volume, as well as mice flow FCT damages caused by more hops. Meanwhile, data relay requires congestion control mechanisms to avoid buffer overflow at intermediate nodes, adding complexity, which NegotiaToR does not need.

A third design choice that is potentially beneficial for goodput also exists, where data relay is enabled only for elephant flows under light loads [8]. In the parallel network topology (Figure 1(a)), each ToR pair can transmit data through all ports, making relaying unnecessary. However, consider the thin-clos topology (Figure 1(b)), where each pair of ToRs is connected by a single port-to-port path. By enabling data relay, the number of available paths between ToRs is increased, and data transmission between ToRs could be done

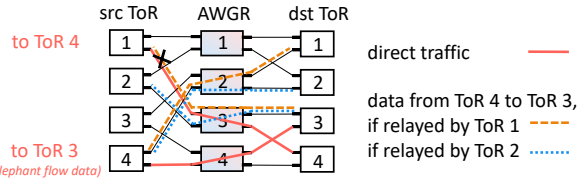


Figure 16: Traffic-aware selective relay on the thin-clos topology. Data relay is enabled for the large volume of data from ToR 4 to ToR 3. When selecting the intermediate ToR, ToR 1 is excluded due to the high volume of direct traffic to ToR 4, which would result in bandwidth competition at ToR 1 → AWGR 3 link. In contrast, ToR 2 can be selected to relay the data, avoiding potential goodput damage.

through multiple ports simultaneously, which could potentially improve goodput.

To observe the impact of such data relay on NegotiaToR, we extend NegotiaToR to support traffic-aware selective relay, aiming to improve the link utilization of the thin-clos topology when empty links are available, as we discussed in §3.5. For elephant flows, in addition to direct data transmission, it also considers utilizing lightly loaded ToRs to relay the data. The limit on flow size is to protect mice flows from increased FCT, while the constraints of the intermediate ToR are to avoid goodput reduction caused by competition with directly transmitted data.

We integrate traffic-aware selective relay into NegotiaToR Matching, including selecting the data to be relayed, filtering out loaded links using local direct traffic information, and preventing buffer overflow at the intermediate ToR with congestion control. We give an illustrative example in Figure 16.

① Before sending requests, source ToRs examine the data volume in the per-destination queue, and determine potential intermediate candidates.

- Co-designed with the mice flow prioritization mechanism (§3.4.2), we disable relay for data in the highest-priority queue (i.e., mice flow data), and only enable it for data in the lowest-priority queue (i.e., elephant flow data) if the data volume exceeds a certain threshold. This ensures that mice flows are always sent directly without FCT concerns, while elephant flows are relayed only when they have enough data to fill extra links so that they can finish sooner than if they were sent directly.
- Source ToRs then use local traffic information and find intermediate ToR candidates, excluding any that would cause bandwidth competition due to shared links with high-volume direct traffic. Requests for data relay are sent to the available candidates.

② In the GRANT step, the candidates evaluate their intermediate queue lengths and high-volume direct traffic conflicts on shared links. If the queue still has spare capacity, and there exists no high-volume concurrent direct traffic, then the candidates grant the request, indicating the maximal data volume it can relay.

③ Finally, in the ACCEPT step, the transmission of direct traffic is prioritized over relayed traffic. After sending direct data first,

	99p Mice Flow FCT (μs) / Normalized Goodput				
	10%	25%	50%	75%	100%
Base	13.2/9.1%	13.4/22.5%	14.2/44.6%	17.3/66.0%	23.8/85.6%
Two-Hop	13.4/9.1%	14.0/22.6%	16.8/45.1%	19.2/66.9%	24.2/86.8%

Table 3: FCT and goodput on the thin-clos topology at various loads, when the traffic-aware selective relay is enabled.

	99p Mice Flow FCT (μs) / Normalized Goodput				
	10%	25%	50%	75%	100%
Base	15.3/9.1%	15.4/22.6%	15.6/45.2%	16.3/67.5%	22.0/89.0%
Data-Size	15.6/9.1%	15.9/22.6%	16.4/45.2%	23.0/67.6%	44.2/89.8%
HoL-Delay	15.2/9.1%	15.2/22.6%	15.3/45.2%	15.3/67.6%	15.5/89.2%

Table 4: FCT and goodput on the parallel network topology at various loads, when NegotiaToR is equipped with informative requests.

	99p Mice Flow FCT (μs) / Normalized Goodput				
	10%	25%	50%	75%	100%
Base	15.3/9.1%	15.4/22.6%	15.6/45.2%	16.3/67.5%	22.0/89.0%
Stateful	13.5/9.1%	13.7/22.6%	13.9/45.2%	16.3/67.5%	23.2/88.8%

Table 5: FCT and goodput on the parallel network topology at various loads, when NegotiaToR maintains traffic matrices for stateful scheduling.

	99p Mice Flow FCT (μs) / Normalized Goodput				
	10%	25%	50%	75%	100%
Base	15.3/9.1%	15.4/22.6%	15.6/45.2%	16.3/67.5%	22.0/89.0%
ProjecToR	16.3/9.1%	21.6/22.6%	40.8/45.0%	52.2/66.1%	54.4/84.7%

Table 6: FCT and goodput on the parallel network topology at various loads, when we utilize ProjecToR's [21] scheduling algorithm.

the source ToR then sends the data to be relayed from the lowest-priority queue to the intermediate ToR.

Simulation results. We conduct simulations of NegotiaToR with traffic-aware selective relay enabled on the thin-clos topology. We show the results in Table 3 under the optimal relay setting we found. When data relay is enabled, FCT is barely affected due to we only selectively intermediating elephant flows. However, goodput is also merely improved. This is because, at light loads, NegotiaToR, with a 2× speedup, already achieves *near-optimal* goodput considering the overhead introduced by guardbands and headers, leaving little room for improvements. While at heavy loads where links are already saturated, data relay does not help, and is turned off by our traffic-aware relay scheme for most of the time. Therefore, we do not employ data relay in NegotiaToR.

A.2.3 Informative requests. To further improve goodput and FCT performance, we explore two approaches that utilize informative requests. One goodput-oriented approach is to include aggregated data size of the per-destination queue in requests, potentially improving link utilization and goodput by first scheduling the pairs with more demands. Another FCT-oriented approach involves including the waiting time of the head-of-line packets from the per-destination queues into requests, prioritizing the pairs with longer waiting delays, potentially avoiding starvation, and reducing tail FCT. Naturally, these improvements come at the cost of additional complexities, such as accessing per-destination queue sizes, logging per-packet delays, and implementing sorting mechanisms.

We implement the two strategies through simulations. For the goodput-oriented data-size approach, we directly include the aggregated data size in requests. While for the FCT-oriented HoL waiting time approach, in the context of priority queues, we need to avoid the long waiting time of elephant flows masking that of mice flows. Therefore, we give the HoL waiting time of the highest-priority queue a greater weight. For instance, based on the three-priority queue case used in our evaluation (§4.1), here we set the weighted HoL waiting time to be $HoL = (1-\alpha)\frac{HoL_{queue_0}+HoL_{queue_1}}{2} + \alpha \cdot HoL_{queue_2}$, where $queue_2$ is the lowest-priority queue and stores the data of elephant flows. Our simulations find that the best performance is attained when setting α to a small, non-zero value such as 0.001. This configuration allows for the prompt scheduling of source-destination pairs with mice flows, while also indicating the transmission needs of elephant flows.

Simulation results. We do simulations on the parallel network topology, and show the results in Table 4. The data-size approach attains extremely minor improvements in goodput, while adversely affecting FCT. The HoL-delay approach reduces FCT by 29.9% at 100% load, but offers minor to zero improvements at other loads. We thus conclude that the benefits of informative requests are not significant enough to justify the additional complexity. Binary requests, along with the scheduling delay bypassing mechanism, are sufficient to achieve high performance.

A.2.4 Stateful scheduling. To explore the benefits of stateful scheduling, we maintain a stateful traffic matrix at each destination ToR to track incoming traffic demands from all sources, thereby preventing over-scheduling and reducing bandwidth waste. After data arrives at the source, the source ToR sends requests to the destination ToR with the size of newly arrived data to update the matrix. Grants are given only if the matrix indicates that the source still has pending data to send, preventing unnecessary scheduling. After giving grants, the size of the remaining data to send in the matrix is decreased temporarily. After the source ToR gives accepts, the accept result will be piggybacked to the destination, allowing for matrix updates. If the source rejects the grant, the destination will revert the temporary adjustment of the matrix to accurately represent pending data for further scheduling.

Simulation results. We conduct simulations of NegotiaToR with stateful scheduling enabled on the parallel network topology, and present the results in Table 5. The results verify our expectations. With stateful scheduling, NegotiaToR's overall performance roughly remains the same. Therefore, we keep the design that the sources will send requests to the destinations, as long as currently there is pending data to send.

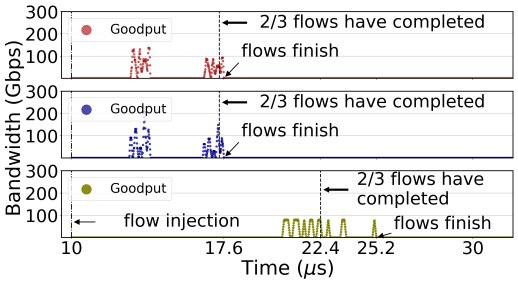


Figure 17: Incast workload: receiver bandwidth observation of incast degree 15. From upper to lower is the bandwidth of NegotiaToR on parallel network, NegotiaToR on thin-clos, and traffic-oblivious scheme on thin-clos.

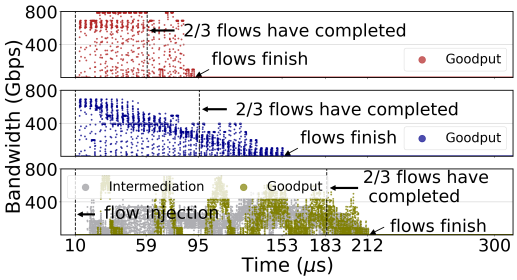


Figure 18: All-to-all workload: receiver bandwidth observation of all-to-all flow size $30KB$. From upper to lower is the bandwidth of NegotiaToR on parallel network, NegotiaToR on thin-clos, and traffic-oblivious scheme on thin-clos.

A.2.5 Other design choices. We also explore other design choices, including the distributed scheduling algorithm used in ProjecToR [21]. ProjecToR also aims to schedule the optical links among multi-ported ToRs in an on-demand manner. Even though it works in a free-space optical switching setting, it is still instructive to transfer its scheduling algorithm to our scenario and compare it with NegotiaToR Matching.

ProjecToR utilizes iterative request-grant based scheduling. It requires measuring and logging data's waiting delays at sources to determine the scheduling priority. Data is grouped into bundles, where each bundle is the unit of scheduling [6]. Bundles with higher waiting delays are scheduled first. Meanwhile, requests are per-port level, which means, unlike NegotiaToR, when sending requests, the sending port of corresponding data is already chosen. We transfer it to NegotiaToR, replacing NegotiaToR Matching, and evaluate its performance through simulations. For comparison, we set the bundle size to be the same with the amount of data sent in one epoch, and only run one round of iteration. Other designs, like NegotiaToR's incast-optimized scheduling delay bypassing scheme, including the priority queue for mice flow prioritization, remain the same.

Simulation results. We show the results in Table 6. Even though introducing more complexity, like measuring and logging data's waiting delays, ProjecToR's performance is still inferior to NegotiaToR, both in FCT and goodput. Adding iterations back to ProjecToR can potentially improve goodput, but will also increase the scheduling delay and thus further enlarge mice flow FCT. This highlights the effectiveness of NegotiaToR's minimalist design.

A.3 Micro-observation of NegotiaToR's performance under incast and all-to-all workloads

Micro-observation of the incast workload. In the microbenchmark in §4.2, we tested the performance of incast workloads. To gain further insights, here we sample the bandwidth usage of incast degree 15 at the destination side, as shown in Figure 17. Flows are injected at 10 μs. For traffic-oblivious designs, a long interval exists before the destination receives data, because the data needs to be relayed to a third ToR first. Whereas for NegotiaToR, due to the incast-optimized scheduling delay bypassing scheme, data can be promptly sent in the predefined phase, so that the destination receives data shortly after the injection, contributing to a short incast finish time. NegotiaToR performs identically across the parallel network and the thin-clos topology, because both topologies provide the same predefined phases.

[6]In our context, this is the amount of data that can be sent in one epoch.

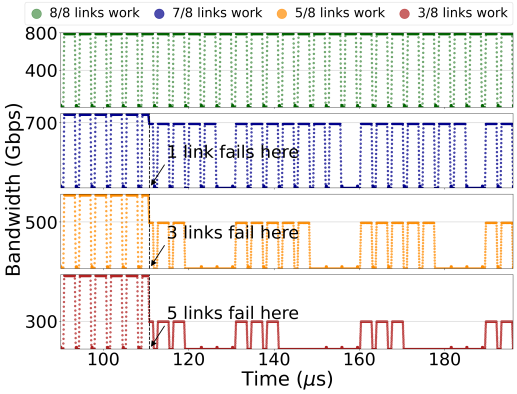


Figure 19: Fault tolerance micro-observation: bandwidth occupation after link failures.

Micro-observation of the all-to-all workload. For the all-to-all workload in §4.2, at a randomly chosen destination, we sample the bandwidth usage when the flow size 30 KB, as shown in Figure 18. Flows are also injected at 10 μs. For traffic-oblivious solutions, different from the case of incast workloads, other than the data destined to it, it also receives intermediate traffic (plotted as light-grey dots) that needs to be forwarded, which competes for the receiver bandwidth and does not contribute to goodput of this receiver. Meanwhile, for NegotiaToR, all received traffic is desired by this destination. As a result, with NegotiaToR Matching's on-demand scheduling, the destination in NegotiaToR continuously receives data at a high bandwidth, leading to high overall goodput.

A.4 Micro-observation of NegotiaToR's behavior under link failures

To better understand NegotiaToR's behavior under link failures, we let a source-destination pair continuously transmit data, and observe the changes of the receiver side bandwidth occupation after link failure happens on the parallel network topology, as shown in Figure 19.

When all links work, the bandwidth occupation shows an on-off shape, indicating different epochs. With links failing, the bandwidth occupation drops to the level of the remaining links. Meanwhile, note that there are epochs with zero bandwidth occupation. This is because of the loss of scheduling messages caused by link failures.

When the source-destination pair happens to rely on one of the failed links to send scheduling messages, the source will not receive any grants, and thus data transmission will be suspended (i.e., all epoch's bandwidth occupation is zero), leading to severe goodput reduction. However, since we regularly change the round-robin rule of the predefined phase, each source-destination pair will send scheduling messages through different links consecutively, mitigating the risk of over-reliance on a single link. As a result, this source-destination pair can still transmit data through other links, and the bandwidth occupation is not continuously zero.