Near-Optimal Throughput with Shallow Buffers in Reconfigurable Datacenter Networks

Vamsi Addanki, Chen Avin, Stefan Schmid

Technische Universität Berlin

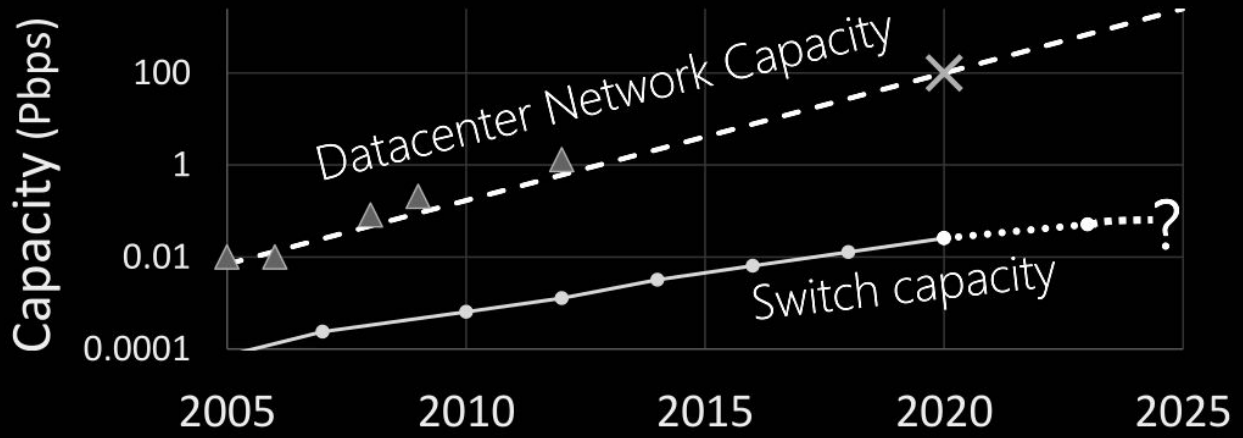
אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Technische Universität Berlin

SIGMETRICS 2023

European Research Council
Established by the European Commission

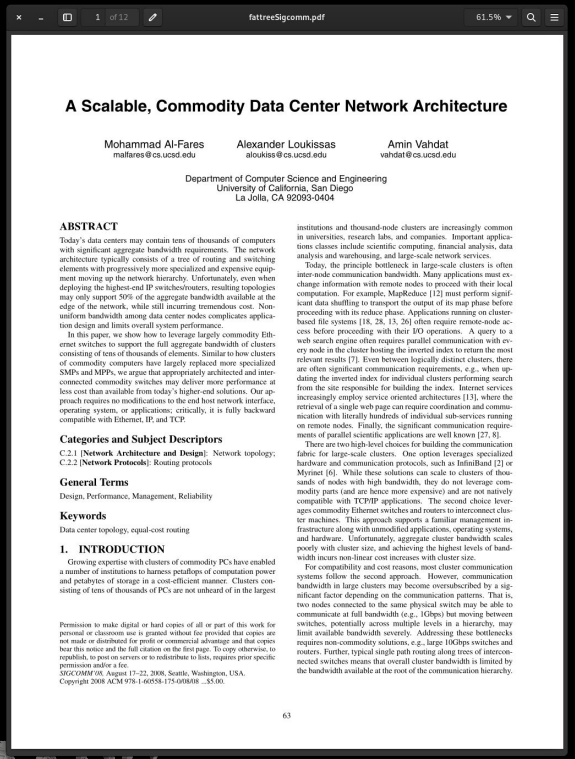
Network Demand vs Capacity Mismatch



MARS

[1] A flat datacenter network with nanosecond optical switching (SIGCOMM 2020)

Traditional Approach: Static Datacenter Topologies



fattreeSigcomm.pdf

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares
malfares@cs.ucsd.edu

Alexander Loukissas
aloukiss@cs.ucsd.edu

Amin Vahdat
vahdat@cs.ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404

ABSTRACT

Today's data centers may contain tens of thousands of computers with significant aggregate bandwidth requirements. The network architecture typically consists of a tree of routing and switching elements with progressively more specialized and expensive equipment moving up the network hierarchy. Unfortunately, even when deploying the highest-end IP switches/routers, resulting topologies may only support 50% of the aggregate bandwidth available at the edge of the network, while still incurring tremendous cost. Non-uniform bandwidth among data center nodes complicates application design and limits overall system performance.

In this paper, we show how to leverage largely commodity Ethernet switches to support the full aggregate bandwidth of clusters consisting of tens of thousands of elements. Similar to how clusters of commodity computers have largely replaced more specialized SMPs and MPPs, we argue that appropriately architected and interconnected commodity switches may deliver more performance at less cost than available from today's higher-end solutions. Our approach requires no modifications to the end host network interface, operating system, or applications; critically, it is fully backward compatible with Ethernet, IP, and TCP.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology; C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Design, Performance, Management, Reliability

Keywords

Data center topology, equal-cost routing

1. INTRODUCTION

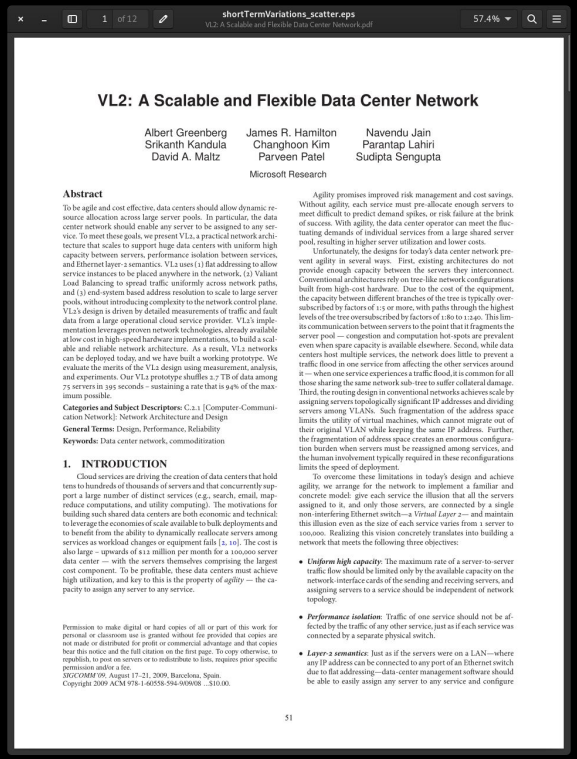
Growing expertise with clusters of commodity PCs have enabled a number of institutions to harness petaflops of computation power and petabytes of storage in a cost-efficient manner. Clusters consisting of tens of thousands of PCs are not unheard of in the largest institutions and thousand-node clusters are increasingly common in universities, research labs, and companies. Important applications classes include scientific computing, financial analysis, data analysis and warehousing, and large-scale network services.

Today, the principle bottleneck in large-scale clusters is often inter-node communication bandwidth. Many applications must exchange information with remote nodes to proceed with their local computation. For example, MapReduce [12] must perform significant data shuffling to transport the output of its map phase before proceeding with its reduce phase. Applications running on cluster-based file systems [18, 28, 13, 26] often require remote-node access before proceeding with their I/O operations. A query to a web search engine often requires parallel communication with every node in the cluster hosting the inverted index to return the most relevant results [7]. Even between logically distinct clusters, there are often significant communication requirements, *e.g.*, when updating the inverted index for individual clusters performing search from the site responsible for building the index. Internet services increasingly employ service oriented architectures [13], where the retrieval of a single web page can require coordination and communication with literally hundreds of individual sub-services running on remote nodes. Finally, the significant communication requirements of parallel scientific applications are well known [27, 8].

There are two high-level choices for building the communication fabric for large-scale clusters. One option leverages specialized hardware and communication protocols, such as InfiniBand [2] or Myrinet [6]. While these solutions can scale to clusters of thousands of nodes with high bandwidth, they do not leverage commodity parts (and are hence more expensive) and are not natively compatible with TCP/IP applications. The second choice leverages commodity Ethernet switches and routers to interconnect cluster machines. This approach supports a familiar management infrastructure along with unmodified applications, operating systems, and hardware. Unfortunately, aggregate cluster bandwidth scales poorly with cluster size, and achieving the highest levels of bandwidth incurs non-linear cost increases with cluster size.

For compatibility and cost reasons, most cluster communication systems follow the second approach. However, communication bandwidth in large clusters may become oversubscribed by a significant factor depending on the communication patterns. That is, two nodes connected to the same physical switch may be able to communicate at full bandwidth (e.g., 1Gbps) but moving between switches, potentially across multiple levels in a hierarchy, may limit available bandwidth severely. Addressing these bottlenecks requires non-commodity solutions, e.g., large 10Gbps switches and routers. Further, typical single path routing along trees of interconnected switches means that overall cluster bandwidth is limited by the bandwidth available at the root of the communication hierarchy.

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.



shortTermVariations_scatter.eps
VL2: A Scalable and Flexible Data Center Network.pdf

VL2: A Scalable and Flexible Data Center Network

Albert Greenberg
Srikanth Kandula
David A. Maltz

James R. Hamilton
Changhoon Kim
Parveen Patel

Navendu Jain
Parantap Lahiri
Sudipta Sengupta

Microsoft Research

Abstract

To be agile and cost effective, data centers should allow dynamic resource allocation across large server pools. In particular, the data center network should enable any server to be assigned to any service. To meet these goals, we present VL2, a practical network architecture that scales to support huge data centers with uniform high capacity between servers, performance isolation between services, and Ethernet layer-2 semantics. VL2 uses (1) flat addressing to allow service instances to be placed anywhere in the network, (2) Valiant Load Balancing to spread traffic uniformly across network paths, and (3) end-system based address resolution to scale to large server pools, without introducing complexity to the network control plane. VL2's design is driven by detailed measurements of traffic and fault data from a large operational cloud service provider. VL2's implementation leverages proven network technologies, already available at low cost in high-speed hardware implementations, to build a scalable and reliable network architecture. As a result, VL2 networks can be deployed today, and we have built a working prototype. We evaluate the merits of the VL2 design using measurement, analysis, and experiments. Our VL2 prototype shuffles 2.7 TB of data among 75 servers in 395 seconds – sustaining a rate that is 94% of the maximum possible.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Network]: Network Architecture and Design

General Terms: Design, Performance, Reliability

Keywords: Data center network, commoditization

1. INTRODUCTION

Cloud services are driving the creation of data centers that hold tens to hundreds of thousands of servers and that concurrently support a large number of distinct services (e.g., search, email, mapreduce computations, and utility computing). The motivations for building such shared data centers are both economic and technical: to leverage the economies of scale available to bulk deployments and to benefit from the ability to dynamically reallocate servers among services as workload changes or equipment fails [2, 10]. The cost is also large – upwards of $12 million per month for a 100,000 server data center — with the servers themselves comprising the largest cost component. To be profitable, these data centers must achieve high utilization, and key to this is the property of *agility* — the capacity to assign any server to any service.

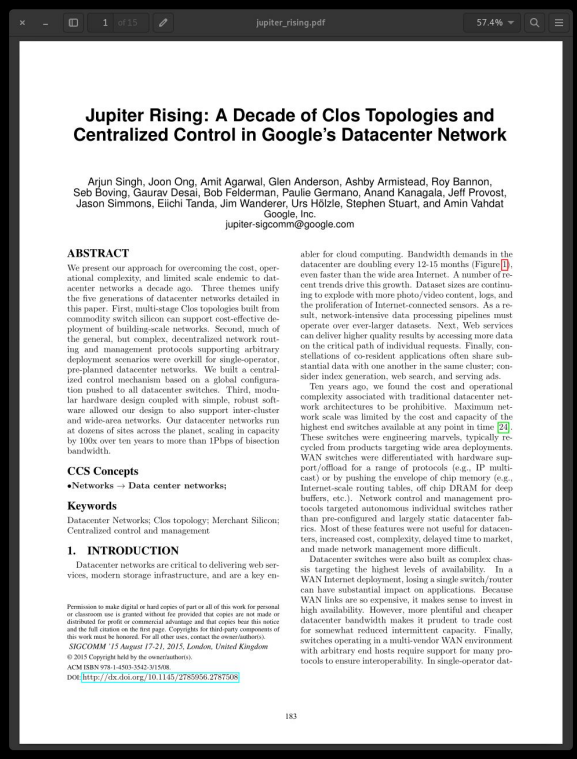
Agility promises improved risk management and cost savings. Without agility, each service must pre-allocate enough servers to meet difficult to predict demand spikes, or risk failure at the brink of success. With agility, the data center operator can meet the fluctuating demands of individual services from a large shared server pool, resulting in higher server utilization and lower costs.

Unfortunately, the designs for today's data center network prevent agility in several ways. First, existing architectures do not provide enough capacity between the servers they interconnect. Conventional architectures rely on tree-like network configurations built from high-cost hardware. Due to the cost of the equipment, the capacity between different branches of the tree is typically oversubscribed by factors of 1:5 or more, with paths through the highest levels of the tree oversubscribed by factors of 1:80 to 1:240. This limits communication between servers to the point that it fragments the server pool — congestion and computation hot-spots are prevalent even when spare capacity is available elsewhere. Second, while data centers host multiple services, the network does little to prevent a traffic flood in one service from affecting the other services around it — when one service experiences a traffic flood, it is common for all those sharing the same network sub-tree to suffer collateral damage. Third, the routing design in conventional networks achieves scale by assigning servers topologically significant IP addresses and dividing servers among VLANs. Such fragmentation of the address space limits the utility of virtual machines, which cannot migrate out of their original VLAN while keeping the same IP address. Further, the fragmentation of address space creates an enormous configuration burden when servers must be assigned among services, and the human involvement typically required in these reconfigurations limits the speed of deployment.

To overcome these limitations in today's design and achieve agility, we arrange for the network to implement a familiar and concrete model: give each service the illusion that all the servers assigned to it, and only those servers, are connected by a single non-interfering Ethernet switch—a *Virtual Layer 2*— and maintain this illusion even as the size of each service varies from 1 server to 100,000. Realizing this vision concretely translates into building a network that meets the following three objectives:

- ***Uniform high capacity***: The maximum rate of a server-to-server traffic flow should be limited only by the available capacity on the network-interface cards of the sending and receiving servers, and assigning servers to a service should be independent of network topology.
- ***Performance isolation***: Traffic of one service should not be affected by the traffic of any other service, just as if each service was connected by a separate physical switch.
- ***Layer-2 semantics***: Just as if the servers were on a LAN—where any IP address can be connected to any port of an Ethernet switch due to flat addressing—data-center management software should be able to easily assign any server to any service and configure

SIGCOMM'09, August 17–21, 2009, Barcelona, Spain.



jupiter_rising.pdf

Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network

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
Google, Inc.
jupiter-sigcomm@google.com

ABSTRACT

We present our approach for overcoming the cost, operational complexity, and limited scale endemic to datacenter networks a decade ago. Three themes unify the five generations of datacenter networks detailed in this paper. First, multi-stage Clos topologies built from commodity switch silicon can support cost-effective deployment of building-scale networks. Second, much of the general, but complex, decentralized network routing and management protocols supporting arbitrary deployment scenarios were overkill for single-operator, pre-planned datacenter networks. We built a centralized control mechanism based on a global configuration pushed to all datacenter switches. Third, modular hardware design coupled with simple, robust software allowed our design to also support inter-cluster and wide-area networks. Our datacenter networks run at dozens of sites across the planet, scaling in capacity by 100x over ten years to more than 1Pbps of bisection bandwidth.

CCS Concepts

•**Networks** → **Data center networks;**

Keywords

Datacenter Networks; Clos topology; Merchant Silicon; Centralized control and management

1. INTRODUCTION

Datacenter networks are critical to delivering web services, modern storage infrastructure, and are a key enabler for cloud computing. Bandwidth demands in the datacenter are doubling every 12-15 months (Figure 1), even faster than the wide area Internet. A number of recent trends drive this growth. Dataset sizes are continuing to explode with more photo/video content, logs, and the proliferation of Internet-connected sensors. As a result, network-intensive data processing pipelines must operate over ever-larger datasets. Next, Web services can deliver higher quality results by accessing more data on the critical path of individual requests. Finally, consolidations of co-resident applications often share substantial data with one another in the same cluster; consider index generation, web search, and serving ads.

Ten years ago, we found the cost and operational complexity associated with traditional datacenter network architectures to be prohibitive. Maximum network scale was limited by the cost and capacity of the highest end switches available at any point in time [24]. These switches were engineering marvels, typically recycled from products targeting wide area deployments. WAN switches were differentiated with hardware support/offload for a range of protocols (e.g., IP multicast) or by pushing the envelope of chip memory (e.g., Internet-scale routing tables, off chip DRAM for deep buffers, etc.). Network control and management protocols targeted autonomous individual switches rather than pre-configured and largely static datacenter fabrics. Most of these features were not useful for datacenters, increased cost, complexity, delayed time to market, and made network management more difficult.

Datacenter switches were also built as complex chassis targeting the highest levels of availability. In a WAN Internet deployment, losing a single switch/router can have substantial impact on applications. Because WAN links are so expensive, it makes sense to invest in high availability. However, more plentiful and cheaper datacenter bandwidth makes it prudent to trade cost for somewhat reduced intermittent capacity. Finally, switches operating in a multi-vendor WAN environment with arbitrary end hosts require support for many protocols to ensure interoperability. In single-operator dat-

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.
DOI: http://dx.doi.org/10.1145/2785956.2787508

MARS

Traditional Approach: Static Datacenter Topologies

fattreeSigcomm.pdf

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares
malfares@cs.ucsd.edu

Alexander Loukissas
aloukiss@cs.ucsd.edu

Amin Vahdat
vahdat@cs.ucsd.edu

jellyfish.pdf

Jellyfish: Networking Data Centers, Randomly

Ankit Singla[†]*; Chi-Yao Hong*[†]*, Lucian Popa*[‡]*, P. Brighten Godfrey*[†]
[†] *University of Illinois at Urbana–Champaign*
[‡] *University of California, Berkeley*

1 Introduction

Data centers today form the backbone of cloud operations. A well provisioned data center network is important to ensure that servers do not face bandwidth bottlenecks to utilization; to isolate services from each other; and to gain more freedom in workload placement, rather than having to tailor placement of workloads to where bandwidth is available [16]. As a result, a significant body of work has tackled the problem of building high network capacity interconnects [1, 10–13, 26, 28, 29].

One crucial problem that has been ignored in these designs is that of incremental expansion of the network, *i.e.*, adding servers and network capacity incrementally to the data center. This may be motivated by growth of the user base, which requires more servers, or by the deployment of more bandwidth-intensive applications. Such expansion can be made feasible by either planned overprovisioning of space and power, or by an upgrade of the server base. The latter enables some of the old servers to be replaced by a larger number of more powerful and, at the same time, less power-consuming new servers.

Industry experience indicates that incremental expansion is an important problem. Consider the growth of Facebook's data center server population from ∼ 30,000 in November 2009 to more than 60,000 by June 2010 [24]. While Facebook has added new data center facilities too, much of this growth is more incremental in existing facilities ("adding capacity on a daily basis" [23]). For instance, Facebook announced that it will double the size of its facility at Prineville, Oregon by early 2012 [9]. Industry experts have also identified incremental build-out as a useful strategy to reduce cap-ex up-front [20].

Do current high-bandwidth data center network proposals allow incremental growth? Consider the fat-tree proposal [1] as an illustrative example. The entire structure is completely determined by the port-count k of the switches available. This is limiting in at least two ways. First, it makes the design space very coarse: full bisection bandwidth fat-trees can only be built at sizes 3456, 8192, 27648, and 65536 corresponding to the commonly available port counts of 24, 32, 48, and 64. Second, even if (for example) 50-port switches were available, the smallest incremental upgrade from the 48-port switch fat-tree would be 3,602 servers. Moreover, this "incremental" growth would require replacing all the 48-port switches by 50-port switches. There is, of course, the possibility of making localized changes like replacing a switch with one with larger port count – this necessarily requires either the servers in the vicinity to share capacity unevenly compared to the rest of the server pool. Thus, without compromise on its structure, bandwidth or cost, a fat-tree is not amenable to incremental growth. Other topologies have similar problems: a hypercube [3] allows power-of-2 sizes, a de Bruijn-like construction [27] allows power-of-3 sizes, etc.

Since it seems that *structure* hinders incremental expansion, we propose the opposite: a random network interconnect. The approach, which we call **Jellyfish**, constructs a *random graph* topology at the switch layer. The inherently sloppy nature of this design has the potential of being significantly more flexible than past designs. Additional components – racks of servers or switches – to improve capacity – can be incorporated with a few random edge swaps. The design naturally supports heterogeneity, allowing the addition of newer network elements with higher port-counts as they become available, unlike past proposals which depend on certain regular port-counts [1, 11–13, 26, 28]. Jellyfish also allows construction of arbitrary-size networks, unlike past proposals discussed above which limit the network to very coarse design points dictated by their structure.

Are we sacrificing high bandwidth for incremental growth? No. To the contrary, we show that Jellyfish supports *more* servers at full bisection bandwidth than an equal-cost fat-tree [1] and with lower mean path length. Jellyfish also provides high connectivity and path diversity. In addition, as we discuss later, Jellyfish is resilient to failures and miswirings during construction.

On the other hand, a random data center network presents several interesting and important challenges that must be addressed for it to be viable. Among these are a suitable routing mechanism (schemes depending on a structured topology are not applicable), physical construction, and cabling layout. We discuss these challenges, which we address in this paper, in §3–§6. We then present a comprehensive evaluation of the performance of Jellyfish in §7. Finally, we present results from our implementation of a Jellyfish network in §8.

[*] Fittingly, a coin-toss decided the first among the first two authors.

shortTermVariations_scatter.eps
VL2: A Scalable and Flexible Data Center Network.pdf

VL2: A Scalable and Flexible Data Center Network

Albert Greenberg
Srikanth Kandula
David A. Maltz

James R. Hamilton
Changhoon Kim
Parveen Patel

Navendu Jain
Parantap Lahiri
Sudipta Sengupta

Microsoft Research

slimfly.pdf

Slim Fly: A Cost Effective Low-Diameter Network Topology

Maciej Besta
ETH Zurich
maciej.besta@inf.ethz.ch

Torsten Hoefler
ETH Zurich
htor@inf.ethz.ch

***Abstract*—We introduce a high-performance cost-effective network topology called Slim Fly that approaches the theoretically optimal network diameter. Slim Fly is based on graphs that approximate the solution to the degree-diameter problem. We analyze Slim Fly and compare it to both traditional and state-of-the-art networks. Our analysis shows that Slim Fly has significant advantages over other topologies in latency, bandwidth, resiliency, cost, and power consumption. Finally, we propose deadlock-free routing schemes and physical layouts for large computing centers as well as a detailed cost and power model. Slim Fly enables constructing cost effective and highly resilient datacenter and HPC networks that offer low latency and high bandwidth under different HPC workloads such as stencil or graph computations.**

I. Introduction

Interconnection networks play an important role in today's large-scale computing systems. The importance of the network grows with ever increasing per-node (multi-core) performance and memory bandwidth. Large networks with tens of thousands of nodes are deployed in warehouse-sized HPC and data centers [8]. Key properties of such networks are determined by their *topologies*: the arrangement of nodes and cables.

Several metrics have to be taken into account while designing an efficient topology. First, high bandwidth is indispensable as many applications perform all-to-all communication [38]. Second, networks can account for as much as 33% of the total system cost [27] and 50% of the overall system energy consumption [2] and thus they should be cost and power efficient. Third, low endpoint-to-endpoint latency is important for many applications, e.g., in high frequency trading. Finally, topologies should be resilient to link failures.

In this paper we show that *lowering network diameter* not only reduces the latency but also the cost of a network and the amount of energy it consumes while maintaining high bisection bandwidth. Lowering the diameter of a network has two effects. First, it reduces energy consumption as each packet traverses a smaller number of SerDes. Another consequence is that packets visit fewer sinks and router buffers and will thus be less likely to contend with other packets flowing through the network. This enables us to reduce the number of costly routers and connections while maintaining high bisection bandwidth.

The well-known fat tree topology [30] is an example of a network that provides high bisection bandwidth. Still, every packet has to traverse many connections as it first has to move up the tree to reach a core router and only then go down to its destination. Other topologies, such as Dragonfly [28], reduce the diameter to three, but their structure also limits bandwidth and, as we will show, has a negative effect on resiliency.

In this work, we propose a new topology, called Slim Fly, which further reduces the diameter and thus costs, energy consumption, and the latency of the network while maintaining high bandwidth and resiliency. Slim Fly is based on graphs with lowest diameter for a given router radix and is, in this sense, approaching the optimal diameter for a given router technology. Figure 1 motivates Slim Fly by comparing the average number of network hops for random uniform traffic using minimal path routing on different network topologies.

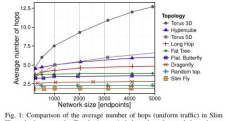


Fig. 1: Comparison of the *average* number of hops (uniform traffic) in Slim Fly and other networks. Topologies are in balanced or close to balanced configurations (explained in Section III), allowing for highest global bandwidth.[1]

Slim Fly enables us to construct cost-efficient full-bandwidth networks with over 100K endpoints with diameter two using readily available high radix routers (e.g., 64-port Black Widow [35] or Mellanox 108-port Director [5]). Larger networks with up to tens of millions of endpoints can be constructed with diameter three as discussed in Section II-A.

The main contributions of this work are:

- We design and analyze a new class of cost effective low-diameter network topologies called Slim Flies.
- We discuss and evaluate different deadlock-free minimal and adaptive routing strategies and we compare them to existing topologies and approaches.
- We show that, in contrast to the first intuition, Slim Fly, using fewer cables and routers, is more tolerant towards link failures than comparable Dragonflies.
- We show a physical layout for a datacenter or an HPC center network and a detailed cost and energy model.

MB is supported by the 2013 Google European Doctoral Fellowship in Parallel Computing.

[1] Numbers for random topologies are updated from values obtained using the Booksim simulator to the lower ones calculated with analytical formulas.

jupiter_rising.pdf

Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network

xpander-full.pdf

Xpander: Towards Optimal-Performance Datacenters

Asaf Valadarsky*
asaf.valadarsky@mail.huji.ac.il

Gal Shahaf†
gal.shahaf@mail.huji.ac.il

Michael Dinitz‡
mdinitz@cs.jhu.edu

Michael Schapira*
schapiram@huji.ac.il

Abstract

Despite extensive efforts to meet ever-growing demands, today's datacenters often exhibit far-from-optimal performance in terms of network utilization, resiliency to failures, cost efficiency, incremental expandability, and more. Consequently, many novel architectures for high performance datacenters have been proposed. We show that the benefits of state-of-the-art proposals are, in fact, derived from the fact that they are (implicitly) utilizing "expander graphs" (aka expanders) as their network topologies, thus unveiling a unifying theme of these proposals. We observe, however, that these proposals are not optimal with respect to performance, do not scale, or suffer from seemingly insurmountable deployment challenges. We leverage these insights to present Xpander, a novel datacenter architecture that achieves near-optimal performance and provides a tangible alternative to existing datacenter designs. Xpander's design turns ideas from the rich graph-theoretic literature on constructing optimal expanders into an operational reality. We evaluate Xpander via theoretical analyses, extensive simulations, experiments with a network emulator, and an implementation on an SDN-capable network testbed. Our results demonstrate that Xpander significantly outperforms both traditional and proposed datacenter designs. We discuss challenges to real-world deployment and explain how these can be resolved.

1 Introduction

The rapid growth of Internet services is placing tremendous demands on datacenters. Yet, as evidenced by the extensive research on improving datacenter performance [24, 25, 6, 48, 52, 23, 45], today's datacenters often exhibit far-from-optimal performance in terms of network utilization, resiliency to failures, cost efficiency, amenability to incremental growth, and beyond.

1.1 The Secret to High Performance

We show that state-of-the-art proposals for next-generation datacenters, e.g., low-diameter networks such as Slim Fly [9], or random networks like Jellyfish [48], have an implicit scalable structure: they employ network topologies that are "expander graphs" (aka expanders). We leverage this insight to present Xpander, a novel datacenter architecture that achieves near-optimal performance and provides a tangible alternative to existing datacenter designs. Xpander's design turns ideas from the rich graph-theoretic literature on constructing optimal expanders into an operational reality. We evaluate Xpander via theoretical analyses, extensive simulations, experiments with a network emulator, and an implementation on an SDN-capable network testbed. Our results demonstrate that Xpander significantly outperforms both traditional and proposed datacenter designs. We discuss challenges to real-world deployment and explain how these can be resolved.

1.2 Why Expanders?

Intuitively, in an expander graph the total capacity from any set of nodes S to the rest of the network is large with respect to the size of S. We present a formal definition of expanders in Section 2. Since this implies that in an expander network, even under worst-case traffic patterns, a sizable fraction of the network capacity can be used, expanders are appealing as datacenter topologies. Indeed, as we show, expanders are the basis of existing proposals for high performance datacenters. Yet, as we show, existing proposals are not optimal with respect to performance, do not scale, or suffer from seemingly insurmountable deployment challenges. We leverage these insights to present Xpander, a novel datacenter architecture that achieves near-optimal performance and provides a tangible alternative to existing datacenter designs. Xpander's design turns ideas from the rich graph-theoretic literature on constructing optimal expanders into an operational reality. We evaluate Xpander via theoretical analyses, extensive simulations, experiments with a network emulator, and an implementation on an SDN-capable network testbed. Our results demonstrate that Xpander significantly outperforms both traditional and proposed datacenter designs. We discuss challenges to real-world deployment and explain how these can be resolved.

Traditional Approach: Static Datacenter Topologies

fattreeSigcomm.pdf

A Scalable, Comm

Mohammad Al-Fares
malfares@cs.ucsd.edu

1 Introduction

Data centers today form the
tions. A well provisioned dat
tant to ensure that servers do
necks to utilization; to isolate
and to gain more freedom in
than having to tailor placem
bandwidth is available [16].
body of work has tackled the
network capacity interconnec

One crucial problem that ha
signs is that of incremental ex
adding servers and network c
data center. This may be moti
base, which requires more set
of more bandwidth-intensive
sion can be made feasible by
sioning of space and power,
server base. The latter enabl
to be replaced by a larger nun
at the same time, less power-

Industry experience indicat
sion is an important proble
of Facebook's data center
30,000 in November 2009 to
2010 [24]. While Facebook
ter facilities too, much of this
tal in existing facilities ("add
sis" [23]). For instance, Face
double the size of its facili
early 2012 [9]. Industry exp
cremental build-out as a useft
up-front [20].

Do current high-bandwidth
posals allow incremental gro
proposal [1] as an illustrative
ture is completely determine
switches available. This is lit
First, it makes the design spa
tion bandwidth fat-trees can
8192, 27648, and 65536 corresponding to the commonly

*Fittingly, a coin-toss decided the first among the first two authors.

f10-nsdi13.pdf

F10: A Fault-Tolerant Engineered Network

Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, Thomas Anderson
University of Washington

Abstract

The data center network is increasingly a cost, reliability and performance bottleneck for cloud computing. Although multi-tree topologies can provide scalable bandwidth and traditional routing algorithms can provide eventual fault tolerance, we argue that recovery speed can be dramatically improved through the co-design of the network topology, routing algorithm and failure detector. We create an engineered network and routing protocol that directly address the failure characteristics observed in data centers. At the core of our proposal is a novel network topology that has many of the same desirable properties as FatTrees, but with much better fault recovery properties. We then create a series of failover protocols that benefit from this topology and are designed to cascade and complement each other. The resulting system, F10, can almost instantaneously reestablish connectivity and load balance, even in the presence of multiple failures. Our results show that following network link and switch failures, F10 has less than 1/7th the packet loss of current schemes. A trace-driven evaluation of MapReduce performance shows that F10's lower packet loss yields a median application-level 30% speedup.

1 Introduction

Data center networks are an increasingly important component to the cost, reliability and performance of cloud services. This has led to recent efforts by the network research community to explore new topologies [11, 12, 13], new routing protocols [11] and new network management layers [3, 4, 20], with a goal of improving network cost-effectiveness, fault tolerance and scalability.

A state of the art approach is taken by Al-Fares et al. [3] and its followup project PortLand [20]. In these systems, the data center network is constructed in a multi-rooted tree structure called a FatTree (inspired by fat-trees [17]) of inexpensive, commodity switches. These proposals provide scalability, both in terms of port count and the overall bisection bandwidth of the network. They also deliver better performance at low costs, primarily due to their use of commodity switches.

The use of a large number of commodity switches, however, opens up questions regarding what happens when links and switches fail. A FatTree has redundant paths between any pair of hosts. If end host operating system changes are possible between these end hosts, the network can be set up to provide multiple paths. The end host manages packet loss and congestion across the paths using MPTCP [22]. In many cases, the data center operator is not in control of the OS, requiring a network-level solution to fault tolerance. A consequence of our work is to show that entirely network-level failure recovery can be practical and nearly instantaneous in a data center setting.

Addressing this need for network-layer recovery, FatTree architectures have proposed using a centralized manager that collects topology and failure information from the switches. It then periodically generates and disseminates back to the switches and end-hosts alternate sets of routes to avoid failures. Centralized route management is both simple and flexible—a reasonable design choice provided that failures do not occur very often.

Recent measurements of network-layer failures in data centers, however, have shown that failures are frequent and disruptive [10]. Network-layer failures can reduce the volume of traffic delivered by more than 40% even when the underlying network is designed for failure resilience. As data centers grow, the probability of network failures and the consequent disruptions on the system as a whole will likely increase, further exacerbating the problem.

Our goal is to co-design a topology and set of protocols that admit near-instantaneous, fine-grained, localized, network-level recovery and rebalancing for common-case network failures. Because the network is already a significant part of the cost of the data center, we limit ourselves to not introducing any additional hardware relative to PortLand. Other work has shown that local repair is possible at the cost of significant added hardware relative to a standard FatTree [9, 12, 13], so our work can be seen as either improving the speed of repair in FatTree and other multi-tree networks or in reducing the hardware cost of fast repair in more general networks. A limitation of our work is that we assume that we can change both the network topology and the protocols used between network switches.

Our system is called F10 (the Fault-Tolerant Engineered Network), a network topology and a set of protocols that can recover rapidly from almost all data center network failures. We design a novel topology to make it easier to do localized repair and rebalancing after failures. This topology is applicable to the FatTree and other multi-tree networks. We then redesign the routing protocols to take advantage of the modified topology. To satisfy the need for extremely fast failover, we use a local recovery mechanism that reacts almost instantaneously at the cost of additional latency and increased congestion. Some failures are not short-term, so local rerouting eventually triggers a slightly slower pushback mechanism that redirects traffic flows before they reach the faulty components.

USENIX Association 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13) 399

shortTermVariations_scatter.eps
VL2: A Scalable and Flexible Data Center Network.pdf

Scalable and F

bert Greenberg
lkanth Kandula
David A. Maltz

A Cost Effec
To

Maciej Besta
ETH Zurich
maciej.besta@inf.ethz.ch

fatclique.pdf

Understanding Lifecycle Management Complexity of Datacenter Topologies

Mingyang Zhang *USC*
Radhika Niranjan Mysore *VMWare*
Sucha Supittayapornpong *USC*
Ramesh Govindan *USC*

Abstract

Most recent datacenter topology designs have focused on performance properties such as latency and throughput. In this paper, we explore a new dimension, life cycle management complexity, which attempts to understand the complexity of deploying a topology and expanding it. By analyzing current practice in lifecycle management, we devise complexity metrics for lifecycle management, and show that existing topology classes have low lifecycle management complexity by some measures, but not by others. Motivated by this, we design a new class of topologies, FatClique, that, while being performance-equivalent to existing topologies, is comparable to, or better than them by *all* our lifecycle management complexity metrics.

1 Introduction

Over the past decade, there has been a long line of work on designing datacenter topologies [2, 35, 31, 32, 3, 4, 20, 1]. While most have focused on performance properties such as latency and throughput, and on resilience to link and switch failures, datacenter *lifecycle management* [30, 38] has largely been overlooked. Lifecycle management is the process of building a network, physically deploying it on a data-center floor, and expanding it over several years so that it is available for use by a constantly increasing set of services.

With datacenters living on for years, sometimes up to a decade [31, 12], their lifecycle costs can be high. A data center design that is hard to deploy can stall the rollout of services for months; this can be expensive considering the rate at which network demands have historically increased [31, 23]. A design that is hard to expand can leave the network functioning with degraded capacity impacting the large array of services that depend on it.

It is therefore desirable to commit to a data-center network design only after getting a sense of its lifecycle management cost and complexity over time. Unfortunately, the costs of the large array of components needed for deployment such as switches, transceivers, cables, racks, patch panels[1], and cable trays, are proprietary and change over time, and so are hard to quantify. An alternative approach is to develop *complexity measures* (as opposed to dollar costs) for lifecycle management, but as far as we know, no prior work has addressed this. In part, this is due to the fact that intuitions about lifecycle management are developed over time and with operations experience, and these lessons are not made available universally.

[1]A patch panel or a wiring aggregator is a device that simplifies cable re-wiring.

Unfortunately, in our experience, this lack of a clear understanding of lifecycle management complexity often results in costly mistakes in the design of datacenters that are discovered during deployment and therefore cannot be rectified. Our paper is a first step towards useful characterizations of lifecycle management complexity.

Contributions. To this end, our paper makes three contributions. First, we design several *complexity metrics* (§3 and §4) that can be indicative of lifecycle management costs (*i.e.*, capital expenditure, time and manpower required). These metrics include the number of: switches, patch panels, *bundle-types*, expansion steps, and links to be re-wired at a patch panel rack during an expansion step.

We design these metrics by identifying structural elements of network deployments that make their deployment and expansion challenging. For instance, the number of switches in the topology determines how complex the network is in terms of *packaging* – laying out switches into homogeneous racks in a space efficient manner. Wiring complexity can be assessed by the number of cable *bundles* and the patch panels a design requires. As these increase, the complexity of manufacturing and packaging all the different cable bundles efficiently into cable trays, and then routing them from one patch panel to the next can be expected to increase. Finally, because expansion is carried out in steps [38], where the network operates at degraded capacity at each step, the number of expansion steps is a measure of the reduced availability in the network induced by lifecycle management. Wiring patterns also determine the number of links that need to be rewired at a patch panel during each step of expansion, a measure of step complexity [38].

Our second contribution is to use these metrics to compare the lifecycle management costs of two main classes of datacenter topologies recently explored in the research literature (§2), Clos [2] and expander graphs [32, 35]. We find that neither class dominates the other: Clos has relatively lower wiring complexity; its symmetric design leads to more uniform bundling (and fewer cable bundle types); but expander graphs at certain scales can have simpler packaging requirements due to their edge expansion property [32]; they end up using much fewer switches than Clos to achieve the same network capacity. Expander graphs also demonstrate better expansion properties because they have *fat edges* (§4) which permit more links to be rewired in each step.

Finally we design and synthesize a novel and practical class of topologies called FatClique (§5), that has lower overall lifecycle management complexity compared to Clos and expander graphs. We do this by combining favorable design

USENIX Association 16th USENIX Symposium on Networked Systems Design and Implementation 235

jupiter_rising.pdf

ecade of Clos Topologies and
n Google's Datacenter Network

xpander-full.pdf

ptimal-Performance Datacenters

Gal Shahaf[†]
gal.shahaf@mail.huji.ac.il

Michael Dinitz[‡]
mdinitz@cs.jhu.edu

ichael Schapira*
apiram@huji.ac.il

the network topology and exploiting the diversity of short paths afforded by expanders for efficient delivery of data traffic. Thus, our first contribution is shedding light on the underlying reason for the empirically good performance of previously proposed datacenter architectures, by showing that these proposals are specific points in a much larger design space of "expander datacenters". We observe, however, that these points are either not sufficiently close to optimal performance-wise, are inherently not scalable, or face significant deployment and maintenance challenges (e.g., in terms of unpredictability and wiring complexity).

We argue that the quest for high-performance datacenter designs is inextricably intertwined with the rich body of research in mathematics and computer science on building good expanders. We seek a point in this design space that offers *near-optimal performance* guarantees while providing a *practical* alternative for today's datacenters (in terms of cabling, physical layout, backwards compatibility with today's protocols, and more). We present Xpander, a novel expander-datacenter architecture carefully engineered to achi-eve both these desiderata.

Importantly, utilizing expanders as network topologies has been proposed in a large variety of contexts, ranging from parallel computing and high-performance computing [49,15,14,39] to optical networks [41] and peer-to-peer networks [40,59]. Our main contributions are examining the performance and operational implications of utilizing expanders in the *datacenter networking context*, and seeking optimal design points in this specific domain (namely, Xpander). Indeed, despite the large body of research on expanders, many aspects of using expanders as datacenter networks (e.g., throughput-related performance measures, specific routing and congestion control protocols, deployment costs, incremental growth, etc.) remain little understood. We next elaborate on expanders, expander datacenters, and Xpander.

1.2 Why Expanders?

Intuitively, in an expander graph the total capacity from any set of nodes S to the rest of the network is large with respect to the size of S. We present a formal definition of expanders in Section 2. Since this implies that in an ex-

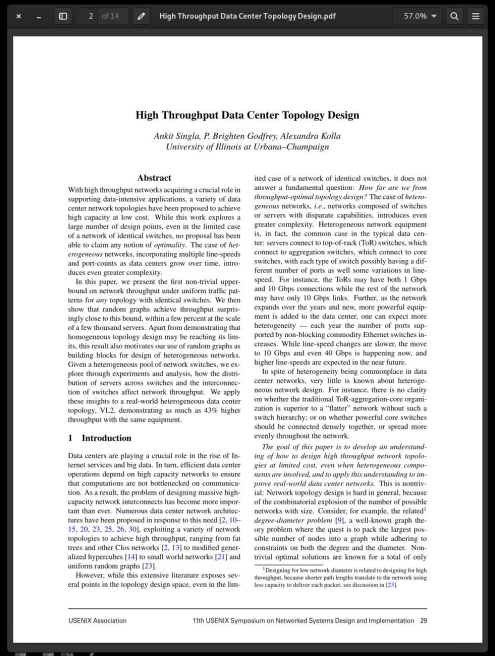
Traditional Approach: Static Datacenter Topologies

Which topology has better throughput?

MARS

Traditional Approach: Static Datacenter Topologies

Which topology has better throughput?



High Throughput Data Center Topology Design

Ankit Singla, P. Brighten Godfrey, Alexandra Kolla
University of Illinois at Urbana–Champaign

Abstract

With high throughput networks acquiring a crucial role in supporting data-intensive applications, a variety of data center network topologies have been proposed to achieve high capacity at low cost. While this work explores a large number of design points, even in the limited case of a network of identical switches, no proposal has been able to claim any notion of *optimality*. The case of *heterogeneous* networks, incorporating multiple line-speeds and port-counts as data centers grow over time, introduces even greater complexity.

In this paper, we present the first non-trivial upper-bound on network throughput under uniform traffic patterns for *any* topology with identical switches. We then show that random graphs achieve throughput surprisingly close to this bound, within a few percent at the scale of a few thousand servers. Apart from demonstrating that homogeneous topology design may be reaching its limits, this result also motivates our use of random graphs as building blocks for design of heterogeneous networks. Given a heterogeneous pool of network switches, we explore through experiments and analysis, how the distribution of servers across switches and the interconnection of switches affect network throughput. We apply these insights to a real-world heterogeneous data center topology, VL2, demonstrating as much as 43% higher throughput with the same equipment.

1 Introduction

Data centers are playing a crucial role in the rise of Internet services and big data. In turn, efficient data center operations depend on high capacity networks to ensure that computations are not bottlenecked on communication. As a result, the problem of designing massive high-capacity network interconnects has become more important than ever. Numerous data center network architectures have been proposed in response to this need [2, 10–15, 20, 23, 25, 26, 30], exploiting a variety of network topologies to achieve high throughput, ranging from fat trees and other Clos networks [2, 13] to modified generalized hypercubes [14] to small world networks [21] and uniform random graphs [23].

However, while this extensive literature exposes several points in the topology design space, even in the limited case of a network of identical switches, it does not answer a fundamental question: *How far are we from throughput-optimal topology design?* The case of *heterogeneous* networks, *i.e.*, networks composed of switches or servers with disparate capabilities, introduces even greater complexity. Heterogeneous network equipment is, in fact, the common case in the typical data center: servers connect to top-of-rack (ToR) switches, which connect to aggregation switches, which connect to core switches, with each type of switch possibly having a different number of ports as well some variations in line-speed. For instance, the ToRs may have both 1 Gbps and 10 Gbps connections while the rest of the network may have only 10 Gbps links. Further, as the network expands over the years and new, more powerful equipment is added to the data center, one can expect more heterogeneity — each year the number of ports supported by non-blocking commodity Ethernet switches increases. While line-speed changes are slower, the move to 10 Gbps and even 40 Gbps is happening now, and higher line-speeds are expected in the near future.

In spite of heterogeneity being commonplace in data center networks, very little is known about heterogeneous network design. For instance, there is no clarity on whether the traditional ToR-aggregation-core organization is superior to a "flatter" network without such a switch hierarchy; or on whether powerful core switches should be connected densely together, or spread more evenly throughout the network.

The goal of this paper is to develop an understanding of how to design high throughput network topologies at limited cost, even when heterogeneous components are involved, and to apply this understanding to improve real-world data center networks. This is nontrivial: Network topology design is hard in general, because of the combinatorial explosion of the number of possible networks with size. Consider, for example, the related[1] *degree-diameter problem* [9], a well-known graph theory problem where the quest is to pack the largest possible number of nodes into a graph while adhering to constraints on both the degree and the diameter. Non-trivial optimal solutions are known for a total of only

[1]Designing for low network diameter is related to designing for high throughput, because shorter path lengths translate to the network using less capacity to deliver each packet; see discussion in [23].

Measuring and Understanding Throughput of Network Topologies

Sangeetha Abdu Jyothi*, Ankit Singla†, P. Brighten Godfrey*, Alexandra Kolla*
*University of Illinois at Urbana–Champaign †ETH Zurich

Abstract—**High throughput is of particular interest in data center and HPC networks. Although myriad network topologies have been proposed, a broad head-to-head comparison across topologies and across traffic patterns is absent, and the right way to compare worst-case throughput performance is a subtle problem.**

In this paper, we develop a framework to benchmark the throughput of network topologies, using a two-pronged approach. First, we study performance on a variety of synthetic and experimentally-measured traffic matrices (TMs). Second, we show how to measure worst-case throughput by generating a near-worst-case TM for any given topology. We apply the framework to study the performance of these TMs in a wide range of network topologies, revealing insights into the performance of topologies with scaling, robustness of performance across TMs, and the effect of scattered workload placement. Our evaluation code is freely available.

I. Introduction

Throughput is a fundamental property of communication networks: at what rate can data be carried across the network between desired end-points? Particularly for data centers and high performance computing, an increase in throughput demand among compute elements has reinvigorated research on network topology, and a large number of network topologies have been proposed in the past few years to achieve high capacity at low cost [3, 10], [1], [13], [16]–[18], [11], [19], [40], [24], [45].

However, there is little order to this large and ever-growing set of network topologies. We lack a broad comparison of topologies, and there is no open, public framework available for testing and comparing topology designs. The absence of a well-specified benchmark complicates research on network design, making it difficult to evaluate a new design against the numerous past proposals, and difficult for industry to know which threads of research are most promising to adopt.

Our goal is to build a framework for accurate and consistent measurement of the throughput of network topologies, and use this framework to benchmark proposed data center and HPC topologies.

To accomplish this, we need metrics for comparison of throughput, and this turns out to be a subtle problem. Throughput can be measured by testing particular workloads, or traffic matrices (TMs), but the immediate question is what TMs to test. One approach is to test a variety of common TMs, which can provide insight into the effect of topological structure for particular use cases reflected by those specific TMs. However, we argue it is useful to go beyond this. In HPC and data center networks, TMs may vary widely depending on the use-case and across time as applications spin up and down or migrate [16], [20]–[23]. Although some applications map well onto certain topologies and known worst-case traffic patterns [34], [43] can be avoided in such cases, a mix of multiple applications could still produce an unintended difficult TM. In fact, [21] observes that network contention among applications sharing an HPC system will worsen in the near future (even after accounting for the expected increase in network speeds) as HPC applications grow in size and variety. Hence, it is useful to understand the *worst-case* performance of a topology, for any TM. However, currently, there does not exist a systematic way to evaluate the worst-case throughput achievable in a network and to identify the traffic pattern responsible for it.

Our key contributions, then, are to (1) develop a heuristic to measure worst-case throughput, and (2) provide an expansive benchmarking of a variety of topologies using a variety of TMs – TMs generated from real-world measurements, synthetic TMs, and finally our new near-worst-case TMs. We discuss each of these in more detail.

(1a) **We evaluate whether cut-based metrics**, such as bisection bandwidth and sparsest cut, solve the problem of estimating worst-case throughput. A number of studies (e.g. [2], [10], [41], [42], [46]) employ cut metrics. It has been noted [47] that bisection bandwidth does not always predict average-case throughput (in a limited setting; [N]). But do cuts measure *worst-case* throughput? We show that it does not, by proving the existence of two families of networks, A and B, where A has a higher cut-metric even though B supports *asymptotically* higher worst-case throughput. Further, we show that the mismatch between cuts and worst-case throughput exists even for highly-structured networks of small size – a 5-ary 3-stage butterfly with only 25 nodes – where the sparsest-cut found through brute force computation is strictly greater than the worst-case throughput.

(1b) Since cut metrics don't achieve our goal, **we develop a heuristic to measure worst-case throughput.** We propose an efficient algorithm to generate a near-worst-case TM for any given topology. We show empirically that these near-worst-case TMs approach a theoretical lower bound of throughput in practice. Note that Kodialam et. al. [26] previously developed a TM for a somewhat different reason[1] which could be repurposed as a near-worst-case TM. Compared with [26], our methodology finds TMs that are just as close to the worst

[1]Upper-bounding throughput performance of a routing scheme using an LP formulation. In addition, [26] did not use their TM to benchmark topologies.

A Throughput-Centric View of the Performance of Datacenter Topologies

Pooria Namyar
University of Southern California

Sucha Supittayaporn pong
Vidyasirimedhi Institute of Science and Technology

Minlan Yu
Harvard University

Mingyang Zhang
University of Southern California

Ramesh Govindan
University of Southern California

ABSTRACT

While prior work has explored many proposed datacenter designs, only two designs, Clos-based and expander-based, are generally considered practical because they can scale using commodity switching chips. Prior work has used two different metrics, bisection bandwidth and throughput, for evaluating these topologies at scale. Little is known, theoretically or practically, how these metrics relate to each other. Exploiting characteristics of these topologies, we prove an upper bound on their throughput, then show that this upper bound better estimates worst-case throughput than all previously proposed throughput estimators and scales better than most of them. Using this upper bound, we show that for expander-based topologies, unlike Clos, beyond a certain size of the network, no topology can have full throughput, even if it has full bisection bandwidth; in fact, even relatively small expander-based topologies fail to achieve full throughput. We conclude by showing that using throughput to evaluate datacenter performance instead of bisection bandwidth can alter conclusions in prior work about datacenter cost, manageability, and reliability.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Network performance modeling**; **Network manageability**; **Topology analysis and generation**. • **General and reference** → **Metrics**.

KEYWORDS

Data centers, Throughput, Clos topologies, Network management

ACM Reference Format:

Pooria Namyar, Sucha Supittayaporn pong, Mingyang Zhang, Minlan Yu, and Ramesh Govindan. 2021. A Throughput-Centric View of the Performance of Datacenter Topologies. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3452296.3472913

[1]Other topology designs, such as Dragonfly [38], and Slimfly [6], do not scale to the sizes of modern data centers, so we do not consider them in this paper; see §7.

Reconfigurable Datacenter Networks

- Generalization of the design space: *Topology can change over time*
- Static networks are a special case

MARS

Reconfigurable Datacenter Networks

- Generalization of the design space: *Topology can change over time*
- Static networks are a special case

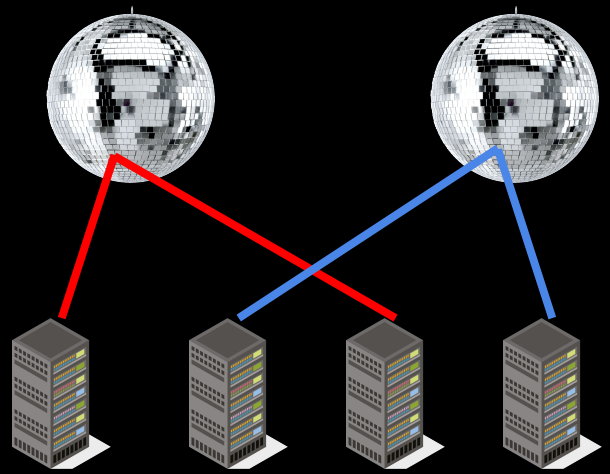
Sirius [Sigcomm 2020]

MARS

Reconfigurable Datacenter Networks

- Generalization of the design space: *Topology can change over time*
- Static networks are a special case

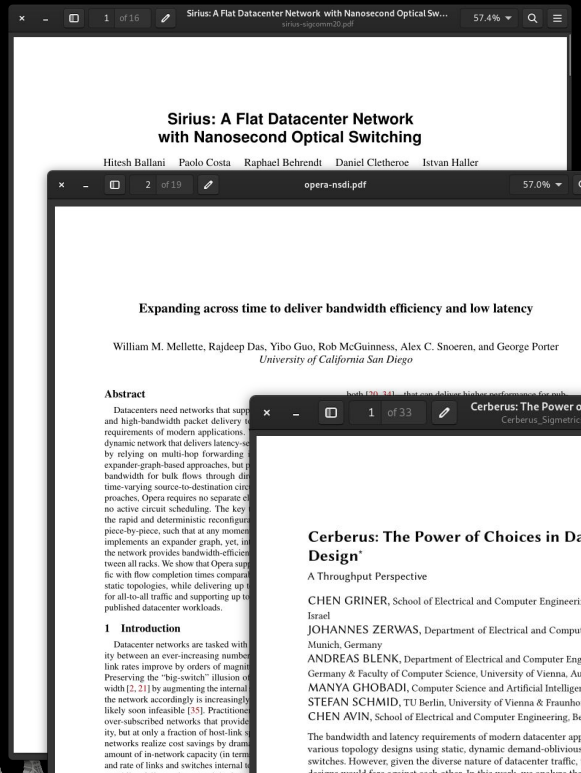
Sirius [Sigcomm 2020]



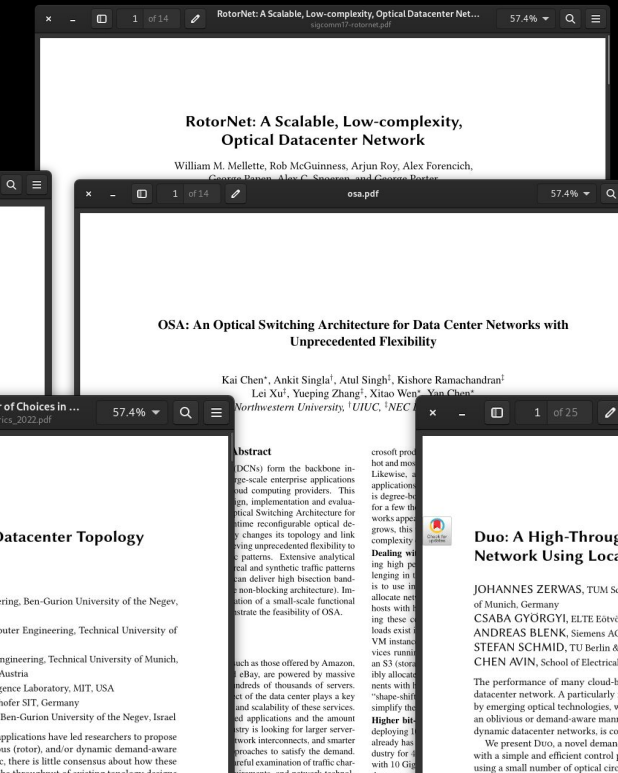
ProjecToR [Sigcomm 2016]

MARS

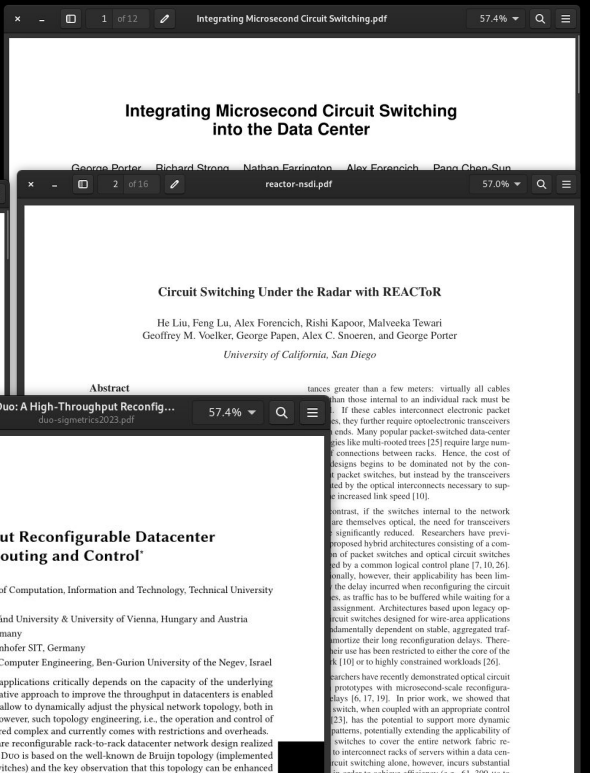
Reconfigurable Datacenter Networks


Sirius: A Flat Datacenter Network with Nanosecond Optical Switching

Hitesh Ballani Paolo Costa Raphael Behrendt Daniel Cletheroe Istvan Haller


RotorNet: A Scalable, Low-complexity, Optical Datacenter Network

William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich,


Integrating Microsecond Circuit Switching into the Data Center

Expanding across time to deliver bandwidth efficiency and low latency

William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter

University of California San Diego

Abstract

Datacenters need networks that supp
and high-bandwidth packet delivery t
requirements of modern applications.
dynamic network that delivers latency-s
by relying on multi-hop forwarding
expander-graph-based approaches, but
bandwidth for bulk flows through dir
time-varying source-to-destination circ
proaches, Opera requires no separate e
no active circuit scheduling. The key
the rapid and deterministic reconfigura
piece-by-piece, such that at any momen
implements an expander graph, yet, in
the network provides bandwidth-efficien
tween all racks. We show that Opera sup
fic with flow completion times compara
static topologies, while delivering up t
for all-to-all traffic and supporting up t
published datacenter workloads.

1 Introduction

Datacenter networks are tasked with
ity between an ever-increasing numbe
link rates improve by orders of magnit
Preserving the "big-switch" illusion o
width [2, 21] by augmenting the internal
the network accordingly is increasingly
likely soon infeasible [35]. Practitione
over-subscribed networks that provide
ity, but at only a fraction of host-link s
networks realize cost savings by dram
amount of in-network capacity (in term
and rate of links and switches internal t
providing full-speed connectivity betwe
hosts, and more limited capacity betwe

The catch, of course, is that any under
inherently biases the network toward cer
tional over-subscribed Clos topologies o
traffic at full line rate; researchers hav
ways of deploying a limited amount of
either through disparate link and swi

OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility

Kai Chen*, Ankit Singla[1], Atul Singh[2], Kishore Ramachandran[2]

Lei Xu[2], Yueping Zhang[2], Xitao Wen[2], Yan Chen*

*Northwestern University, [1]UIUC, [2]NEC

Circuit Switching Under the Radar with REACToR

He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari

Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter

University of California, San Diego

Abstract

Cerberus: The Power of Choices in Datacenter Topology Design*

A Throughput Perspective

CHEN GRINER, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel

JOHANNES ZERWAS, Department of Electrical and Computer Engineering, Technical University of Munich, Germany

ANDREAS BLENK, Department of Electrical and Computer Engineering, Technical University of Munich, Germany & Faculty of Computer Science, University of Vienna, Austria

MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA

STEFAN SCHMID, TU Berlin, University of Vienna & Fraunhofer SIT, Germany

CHEN AVIN, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel

The bandwidth and latency requirements of modern datacenter applications have led researchers to propose various topology designs using static, dynamic demand-oblivious (rotor), and/or dynamic demand-aware switches. However, given the diverse nature of datacenter traffic, there is little consensus about how these designs would fare against each other. In this work, we analyze the throughput of existing topology designs under different traffic patterns and study their unique advantages and potential costs in terms of bandwidth and latency "tax". To overcome the identified inefficiencies, we propose CERBERUS, a unified, two-layer leaf-spine optical datacenter design with three topology types. CERBERUS systematically matches different traffic patterns with their most suitable topology type: e.g., latency-sensitive flows are transmitted via a static topology, all-to-all traffic via a rotor topology, and elephant flows via a demand-aware topology. We show analytically and in simulations that CERBERUS can improve throughput significantly compared to alternative approaches

Duo: A High-Throughput Reconfigurable Datacenter Network Using Local Routing and Control*

JOHANNES ZERWAS, TUM School of Computation, Information and Technology, Technical University of Munich, Germany

CSABA GYÖRGYI, ELTE Eötvös Loránd University & University of Vienna, Hungary and Austria

ANDREAS BLENK, Siemens AG, Germany

STEFAN SCHMID, TU Berlin & Fraunhofer SIT, Germany

CHEN AVIN, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel

The performance of many cloud-based applications critically depends on the capacity of the underlying datacenter network. A particularly innovative approach to improve the throughput in datacenters is enabled by emerging optical technologies, which allow to dynamically adjust the physical network topology, both in an oblivious or demand-aware manner. However, such topology engineering, i.e., the operation and control of dynamic datacenter networks, is considered complex and currently comes with restrictions and overheads.

We present DUO, a novel demand-aware reconfigurable rack-to-rack datacenter network design realized with a simple and efficient control plane. DUO is based on the well-known de Bruijn topology (implemented using a small number of optical circuit switches) and the key observation that this topology can be enhanced using dynamic ("opportunistic") links between its nodes.

In contrast to previous systems, DUO has several desired features: i) It makes effective use of the network capacity by supporting integrated and multi-hop routing (paths that combine both static and dynamic links). ii) It uses a work-conserving queue scheduling which enables out-of-the-box TCP support. iii) DUO employs greedy routing that is implemented using standard IP longest prefix match with small forwarding tables. And iv) during topological reconfigurations, routing tables require only local updates, making this approach ideal for dynamic networks.

Reconfigurable Datacenter Networks

Which topology has better throughput?

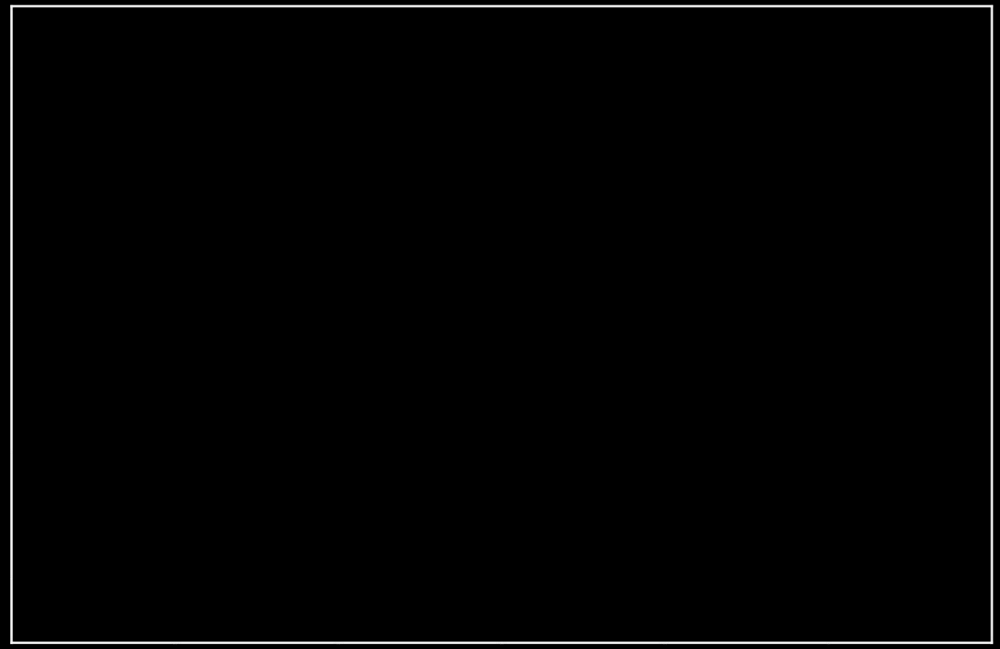
MARS

Reconfigurable Datacenter Networks

Which topology has better throughput?

Main focus of this paper.
Stay tuned!

MARS



Static DCNs

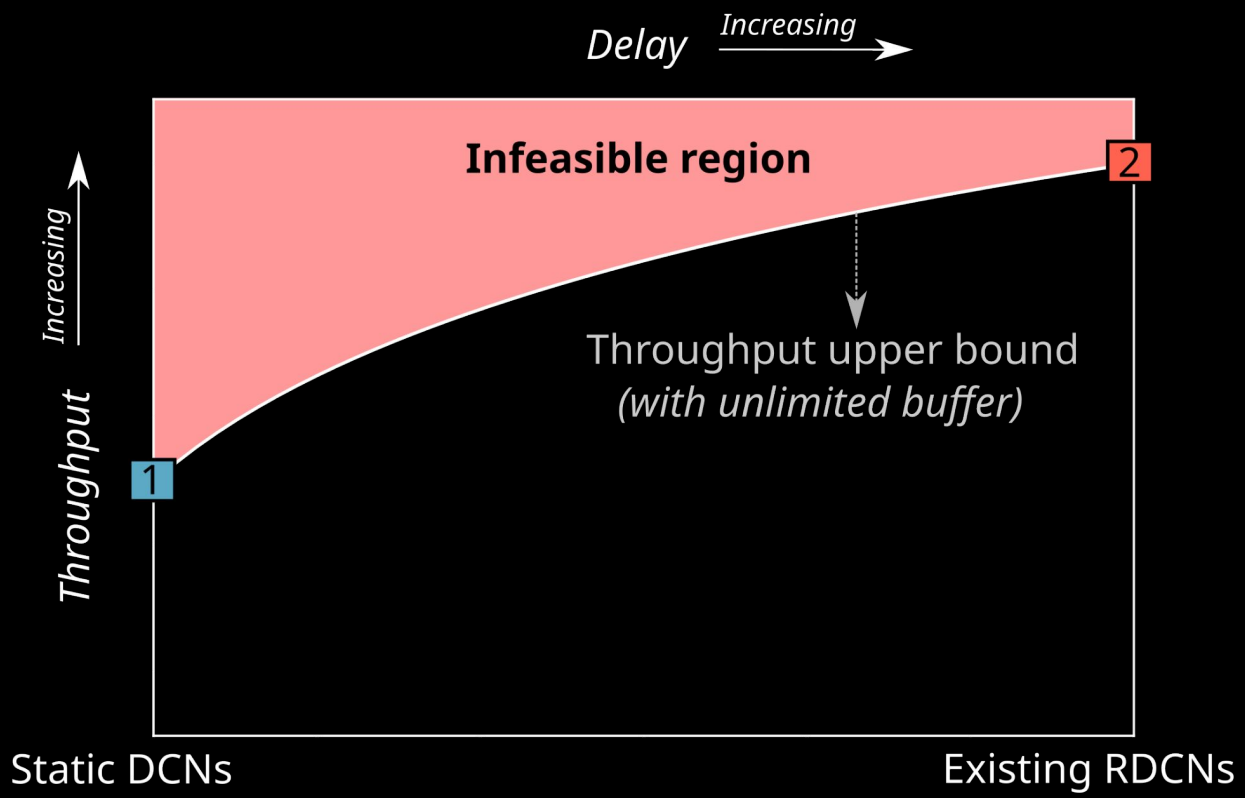
Existing RDCNs

MARS

Topology Over Time

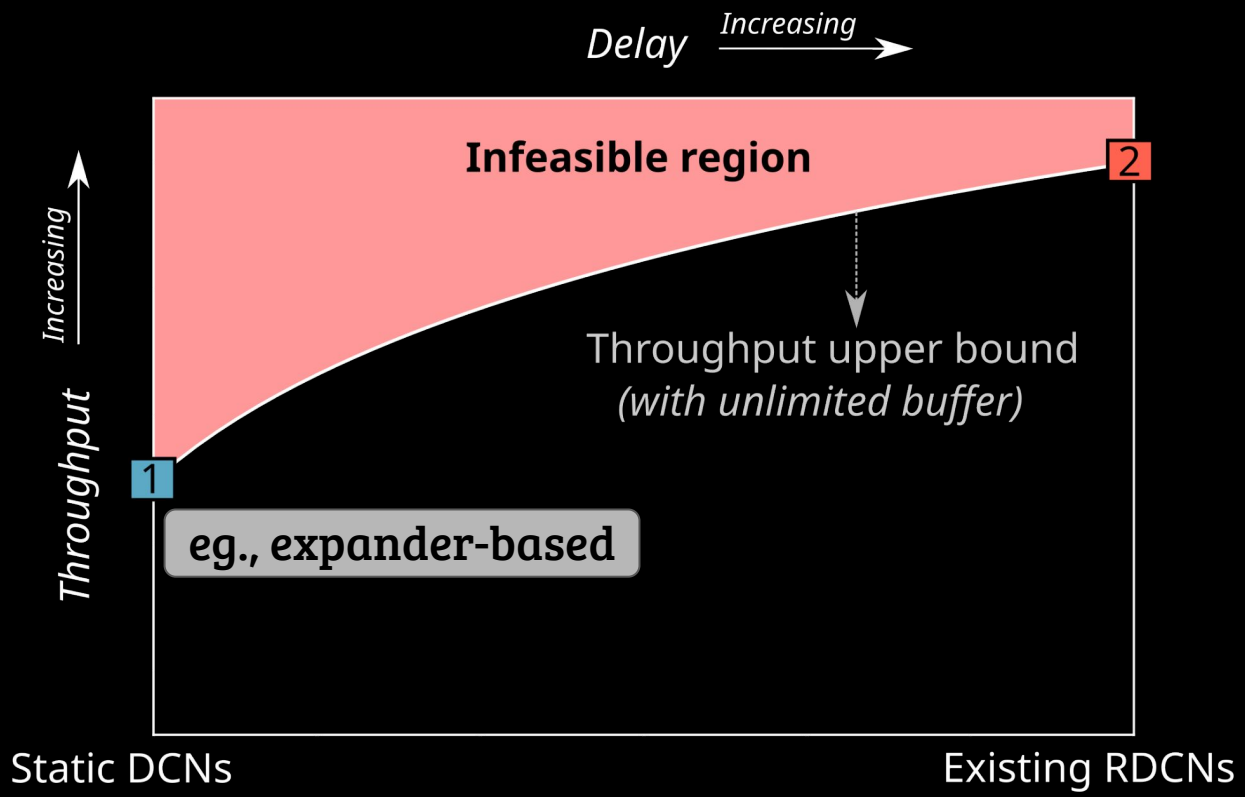
MARS

Topology Over Time



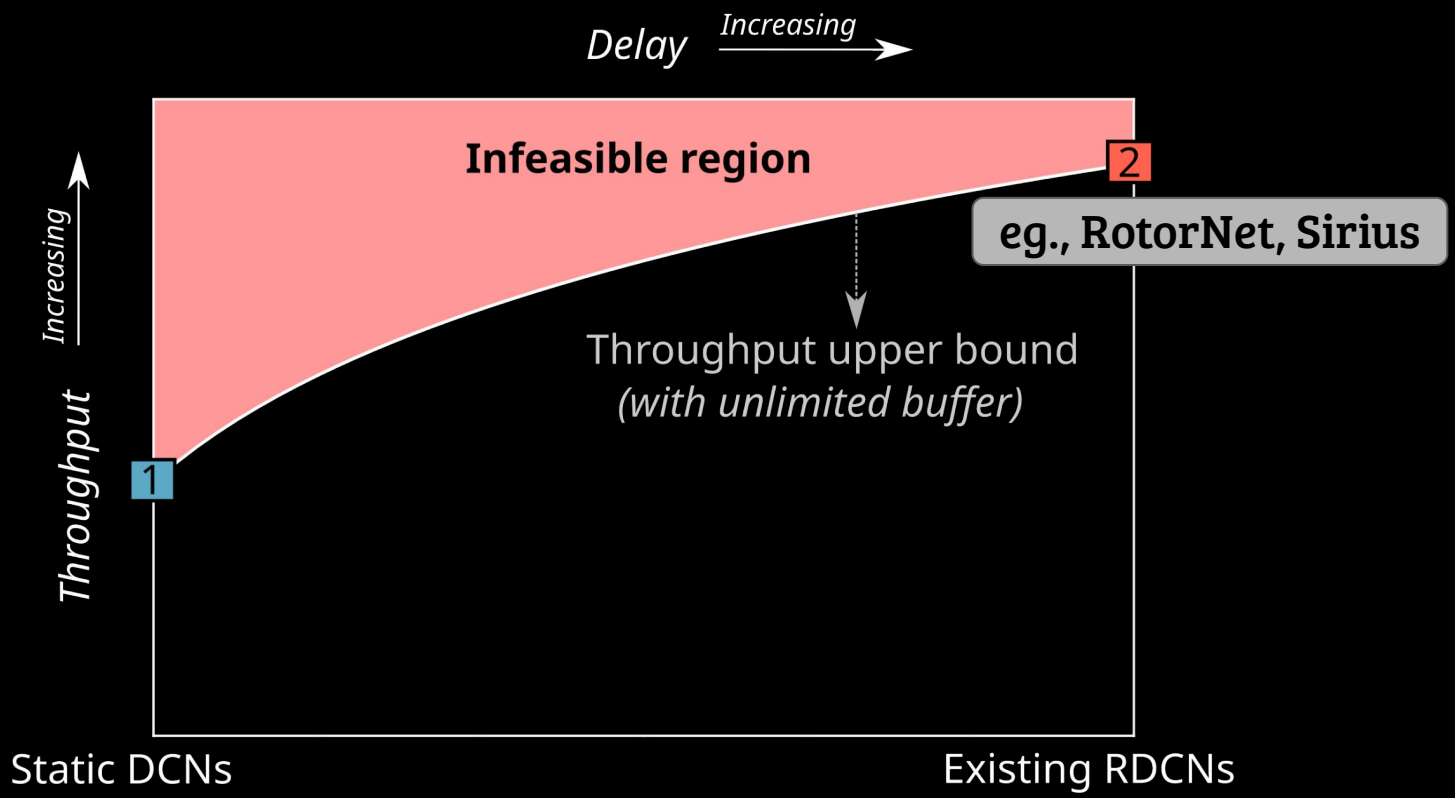
MARS

Topology Over Time



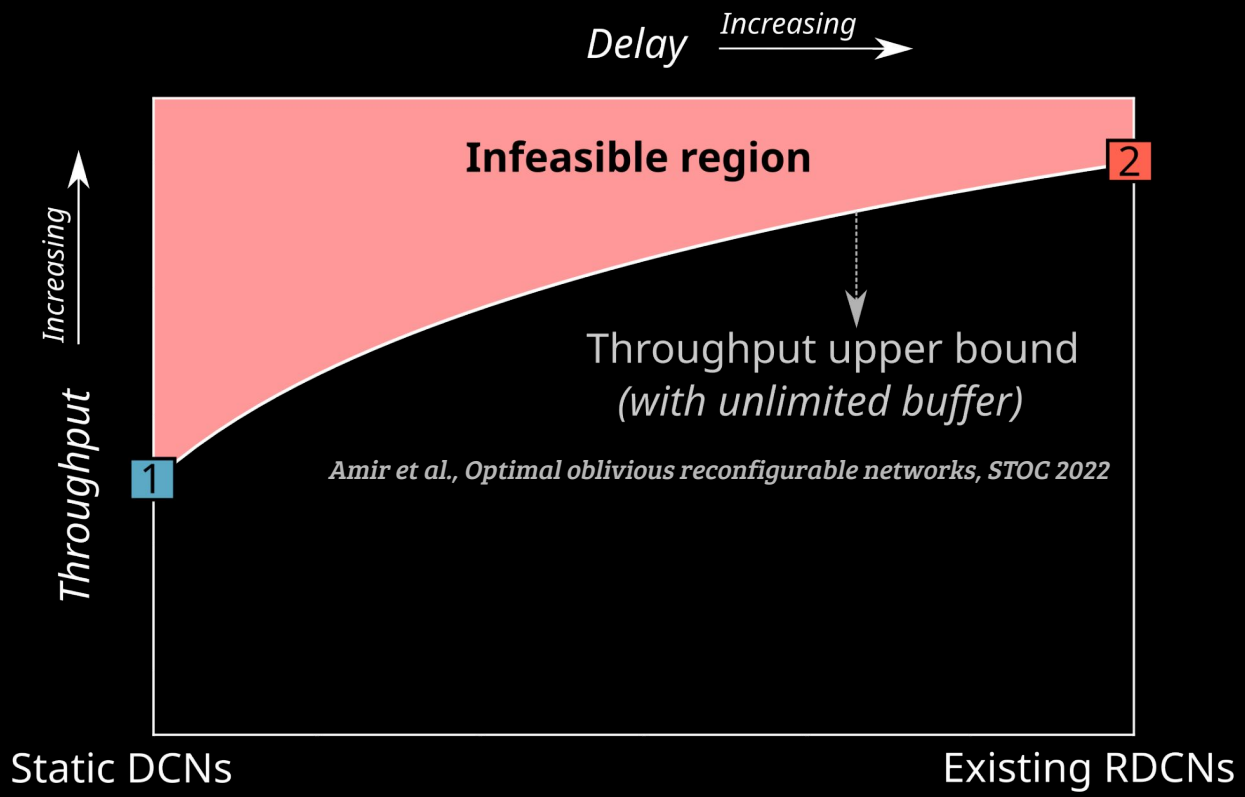
MARS

Topology Over Time



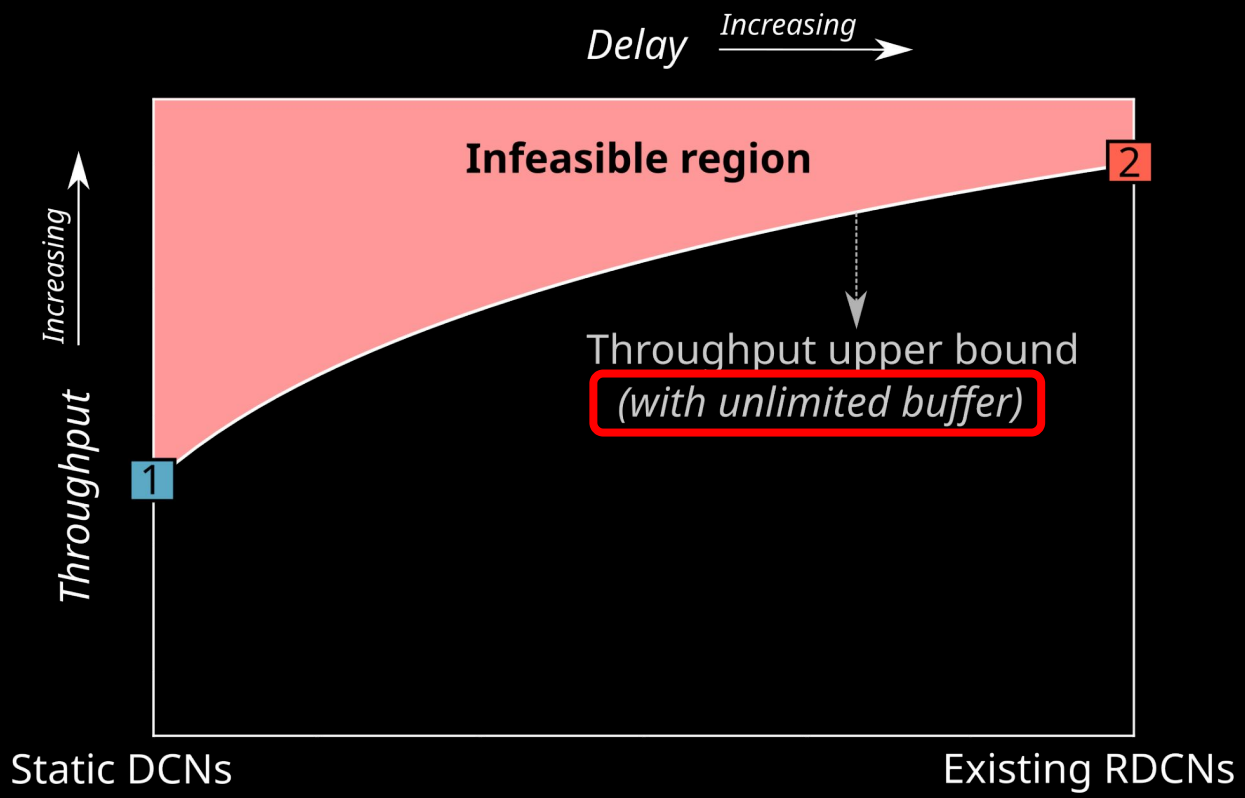
MARS

Topology Over Time



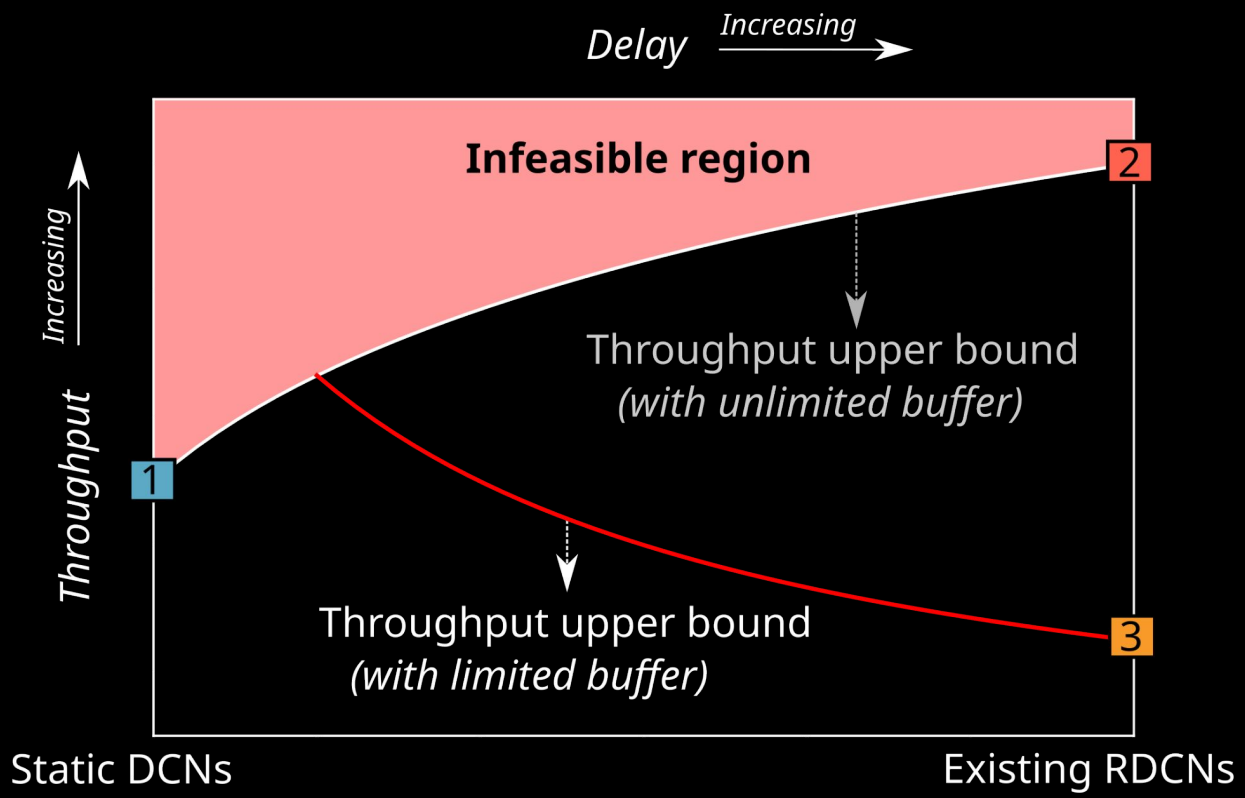
MARS

Topology Over Time



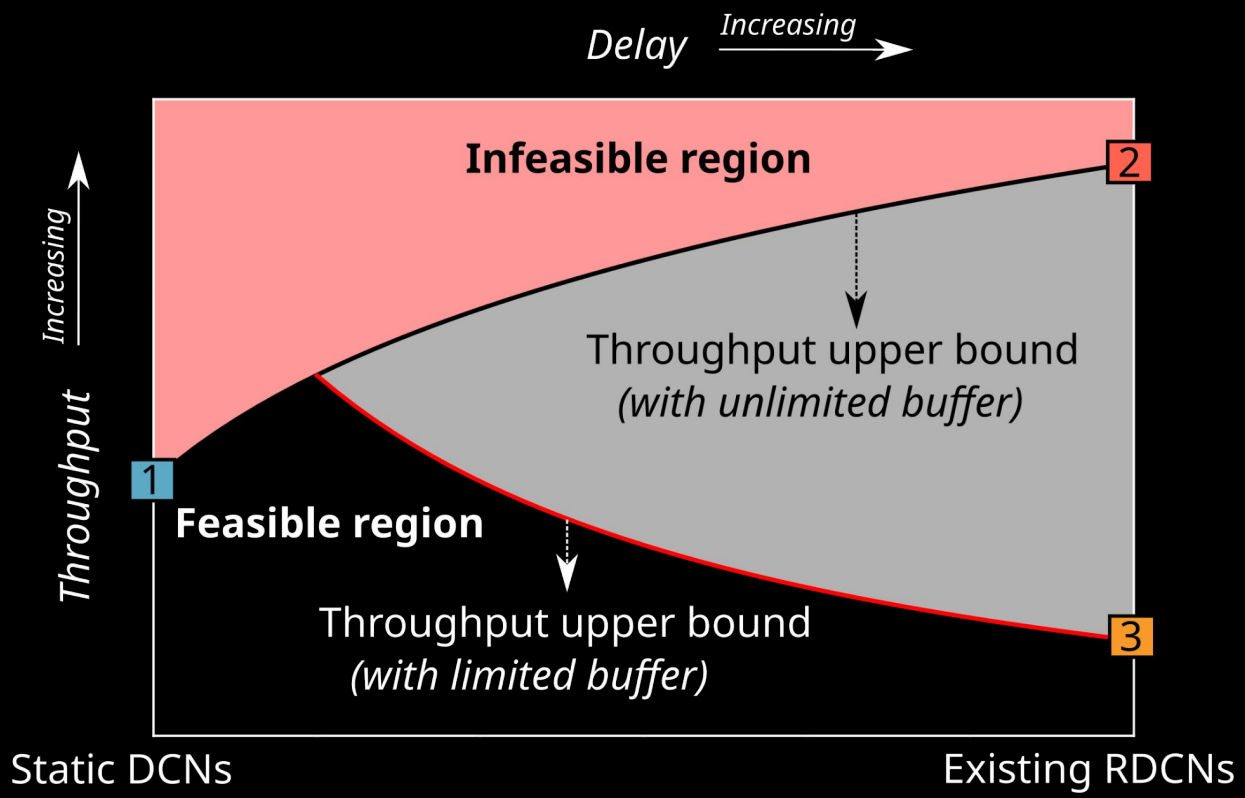
MARS

Topology Over Time



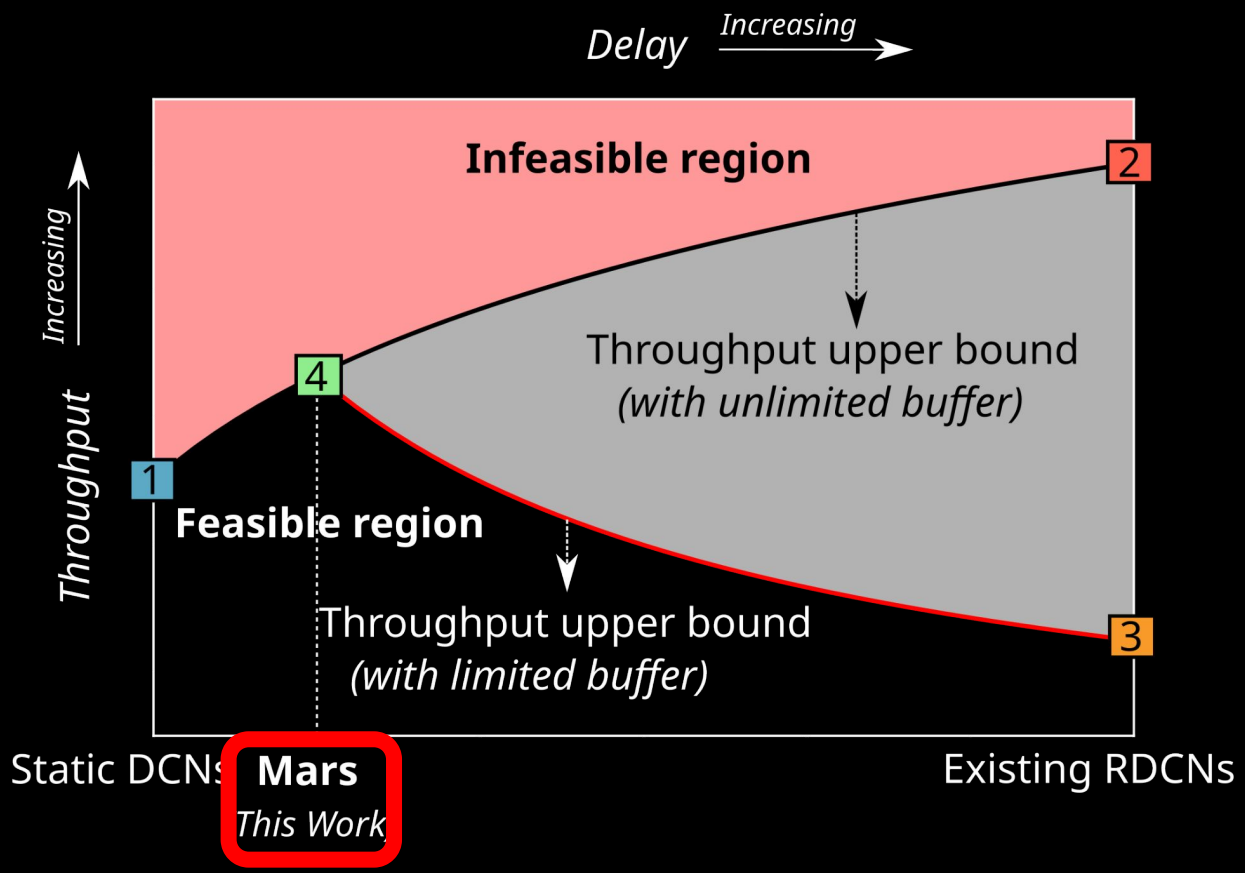
MARS

Topology Over Time

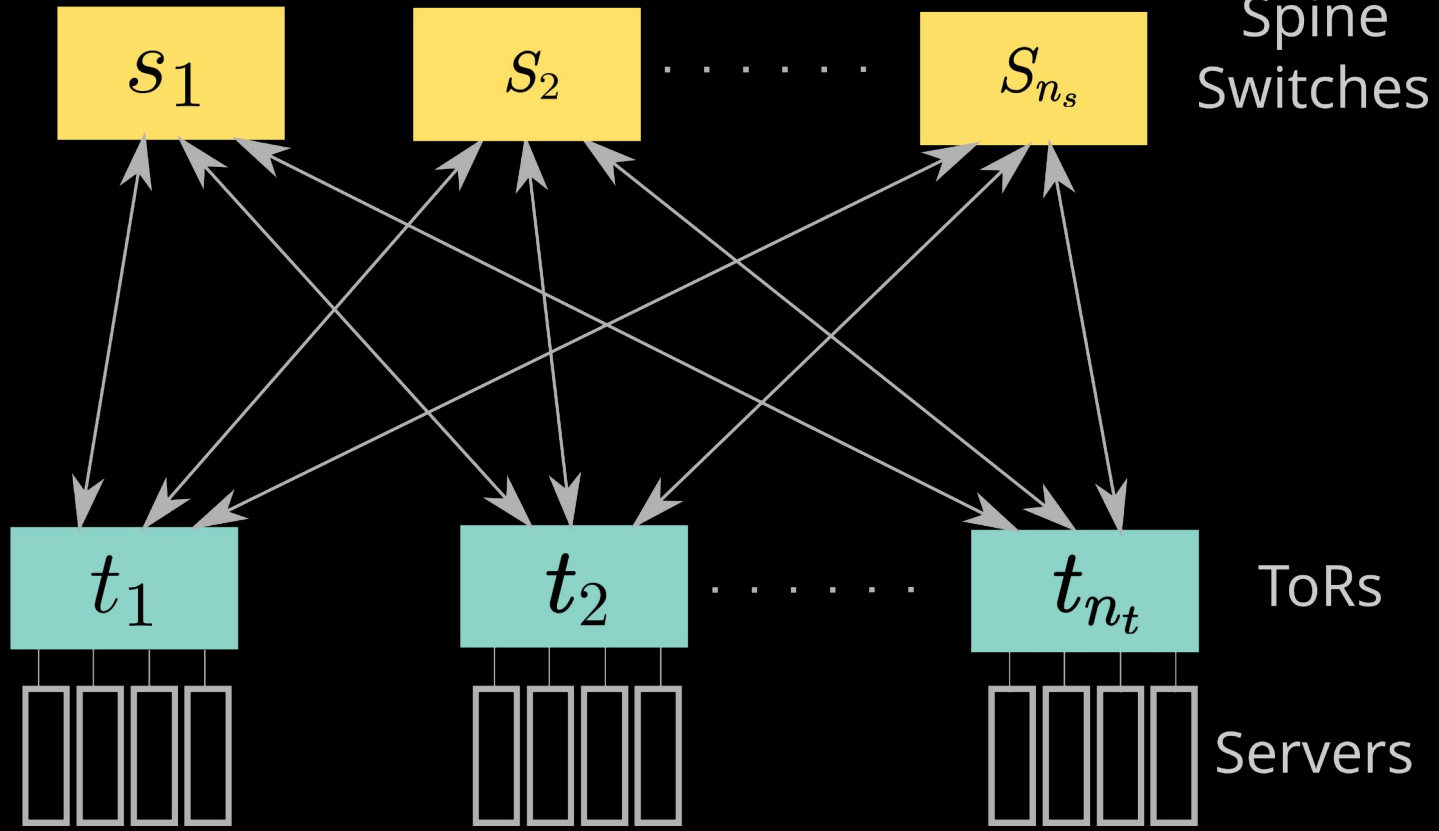


MARS

Topology Over Time

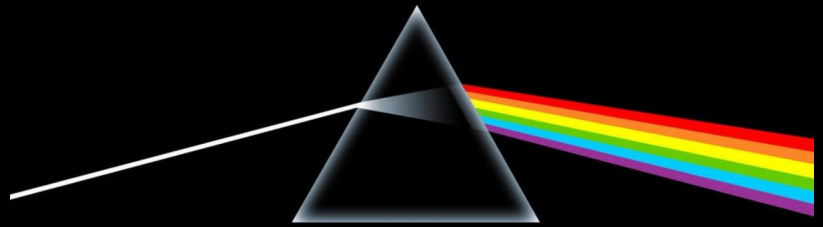

Delay
Increasing
Infeasible region
2
Increasing
Throughput
4
Throughput upper bound
(with unlimited buffer)
1
Feasible region
Throughput upper bound
(with limited buffer)
3
Static DCNs
Mars
This Work
Existing RDCNs
Topology Over Time

MARS



MARS

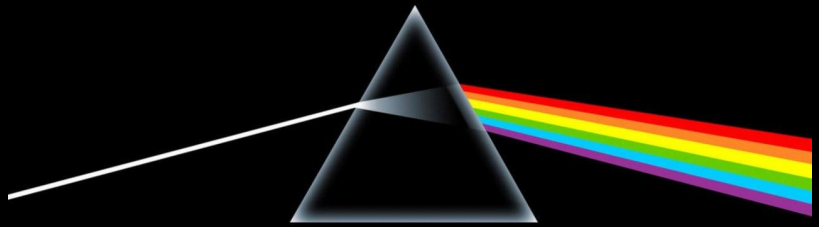
Emerging Technologies - Optical Networks on the Rise



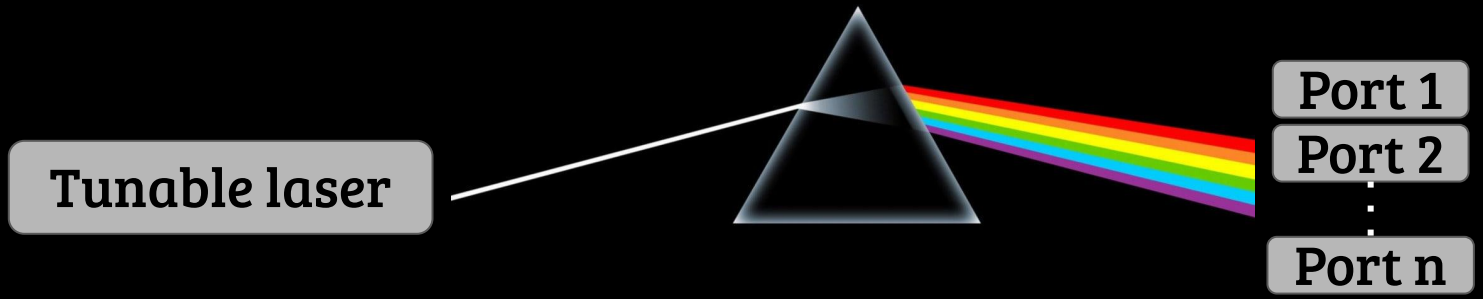
MARS

Emerging Technologies - Optical Networks on the Rise

Tunable laser

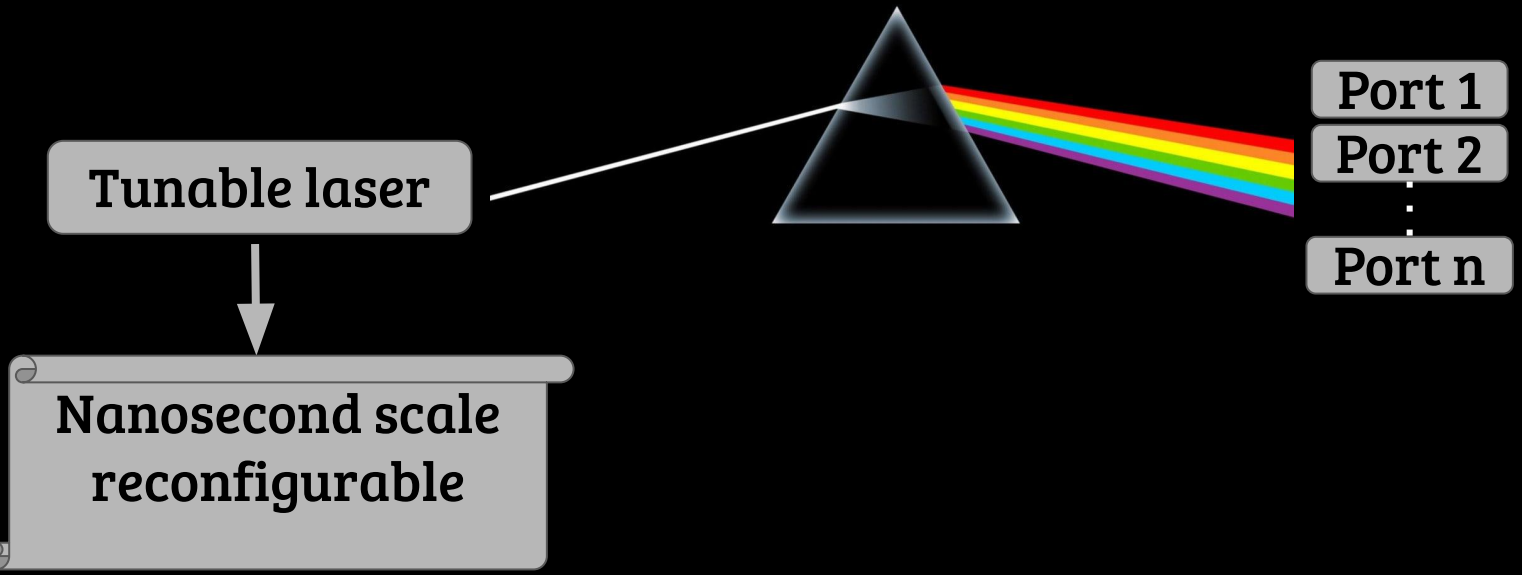


Emerging Technologies - Optical Networks on the Rise



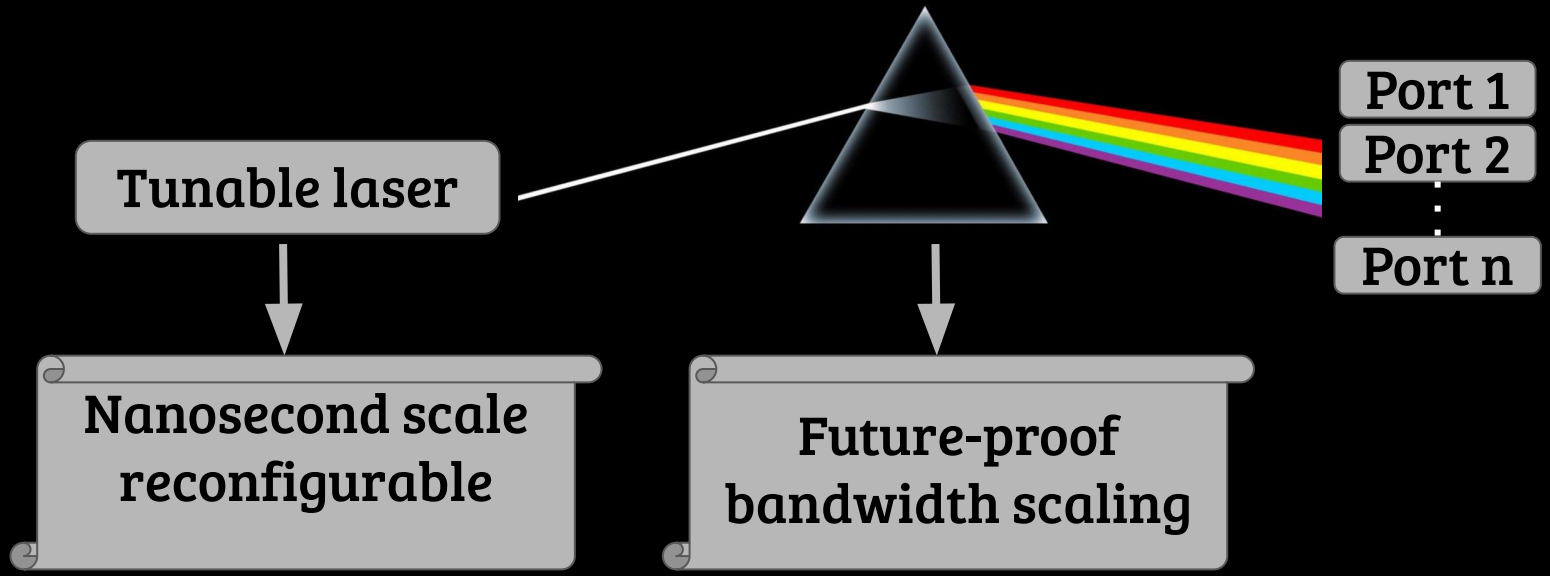
MARS

Emerging Technologies - Optical Networks on the Rise



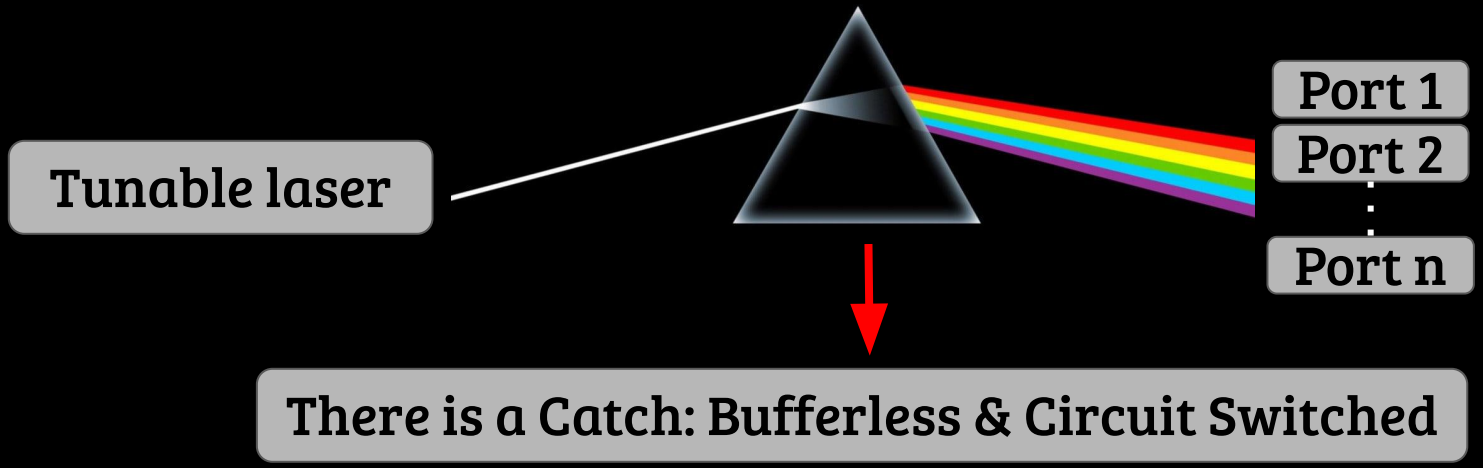
MARS

Emerging Technologies - Optical Networks on the Rise

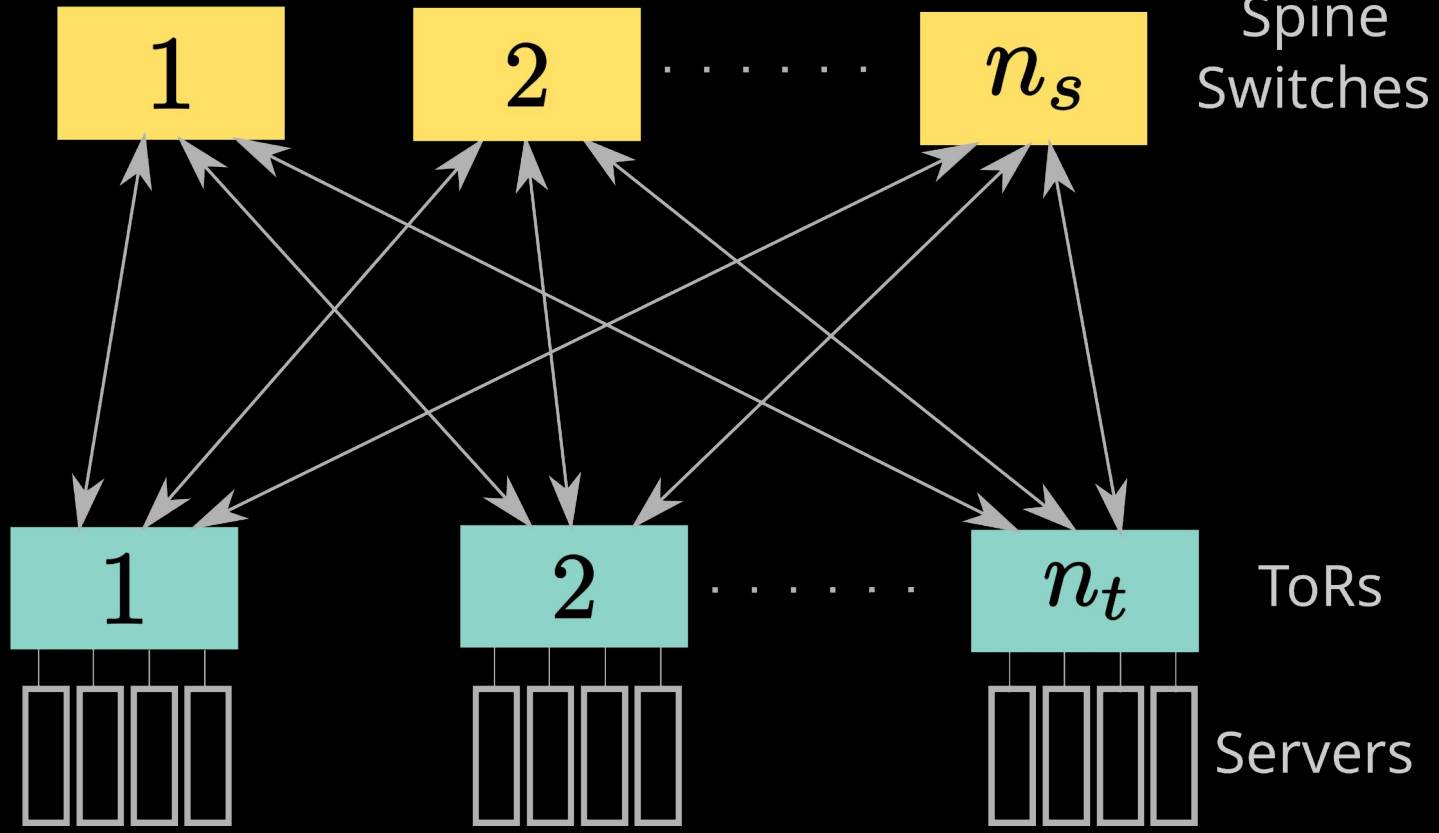


MARS

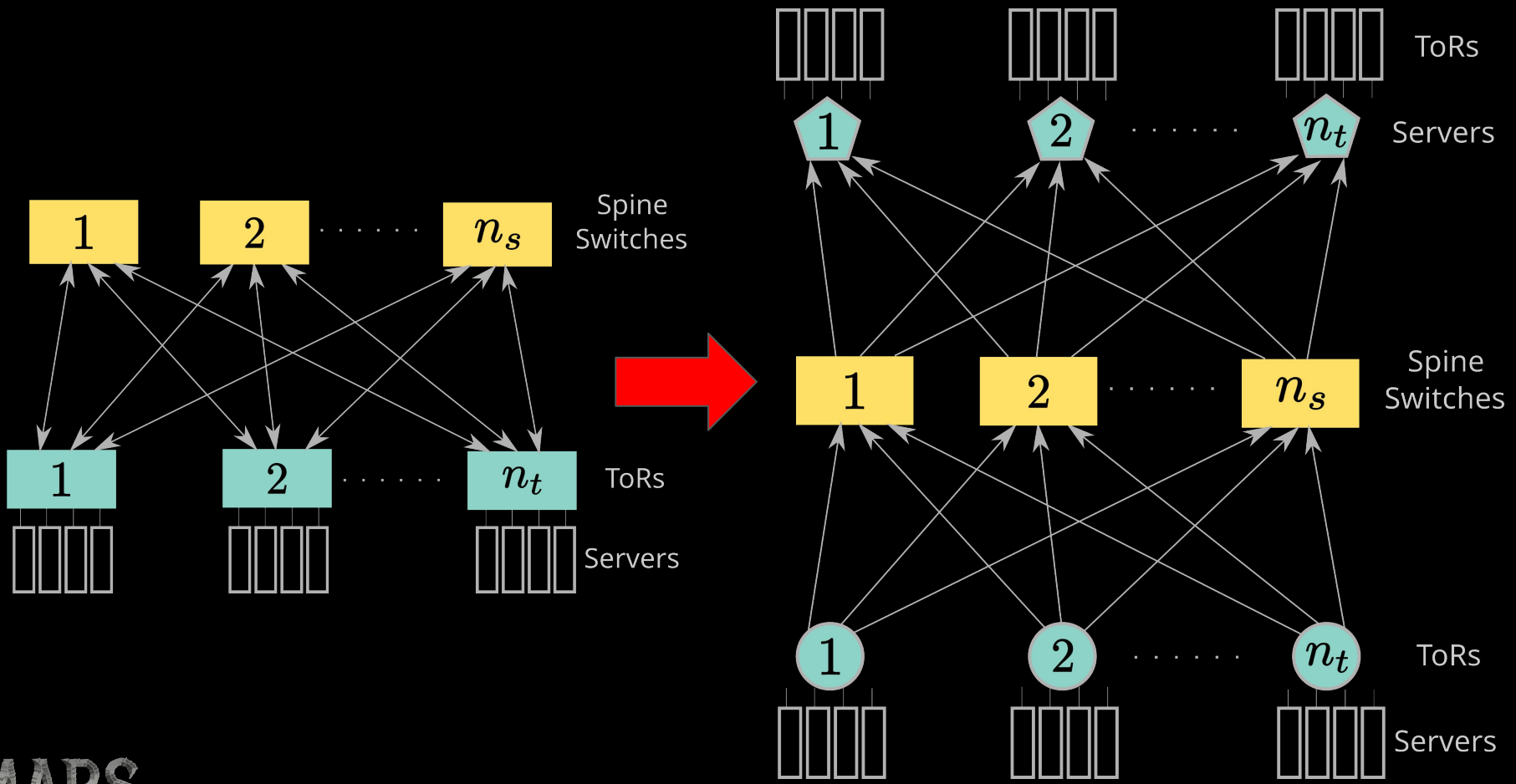
Emerging Technologies - Optical Networks on the Rise



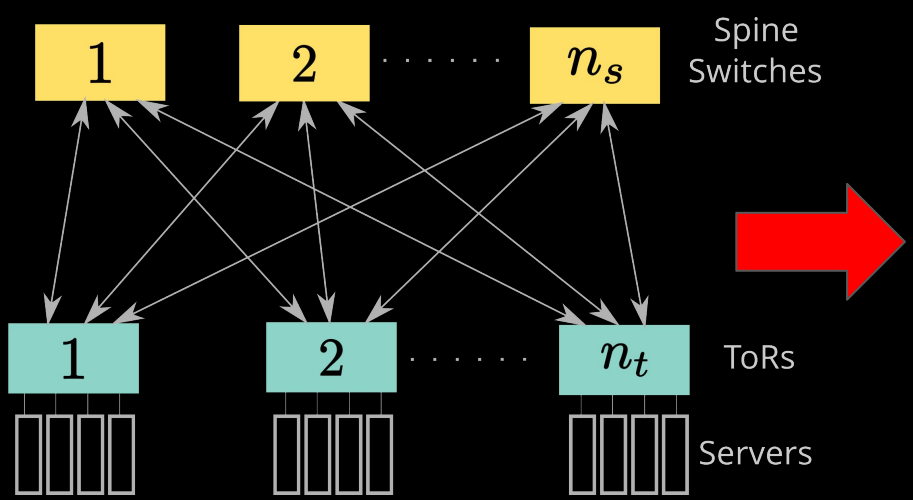
MARS

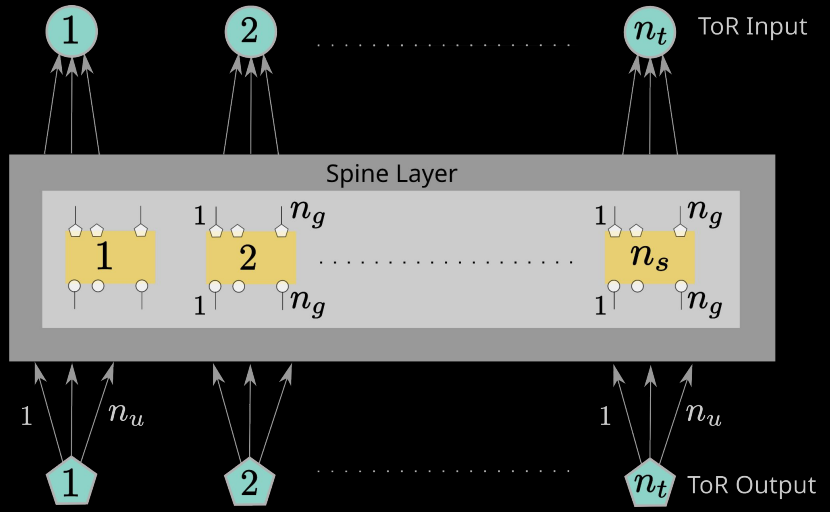


MARS

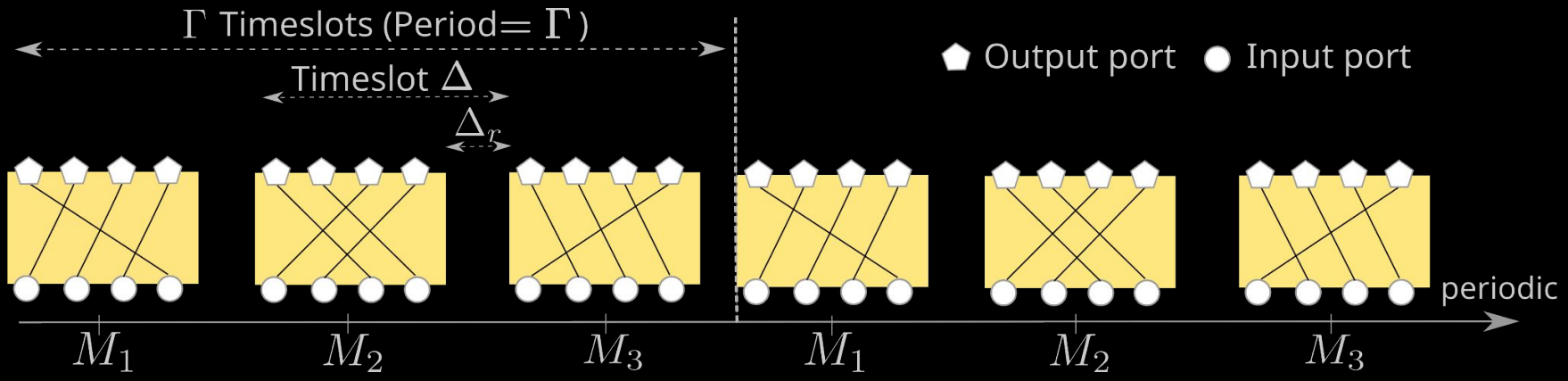

1
2
n_s
Spine Switches
1
2
n_t
ToRs
Servers
1
2
n_t
ToRs
1
2
n_s
Spine Switches
1
2
n_t
ToRs
Servers

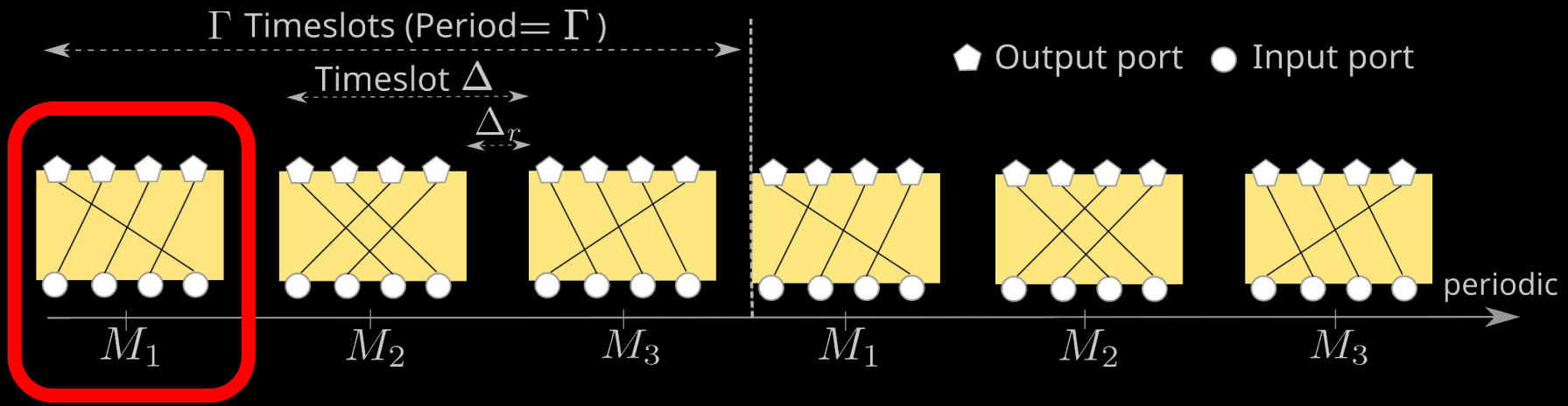
MARS


1
2
n_s
Spine Switches
1
2
n_t
ToRs
Servers

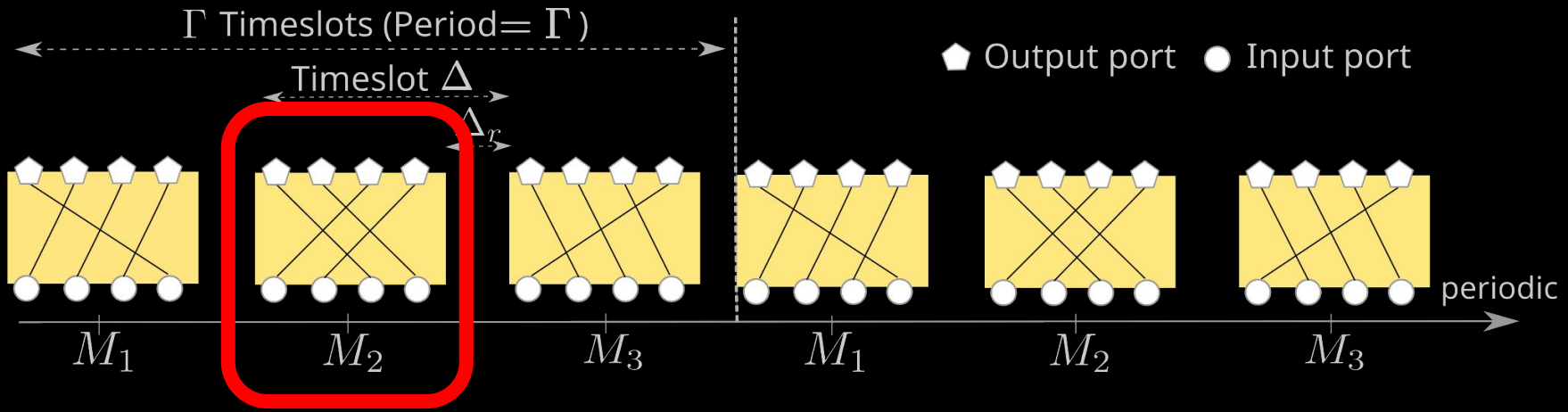

1
2
n_t
ToR Input
Spine Layer
1
2
n_s
n_g
1
n_u
ToR Output

MARS

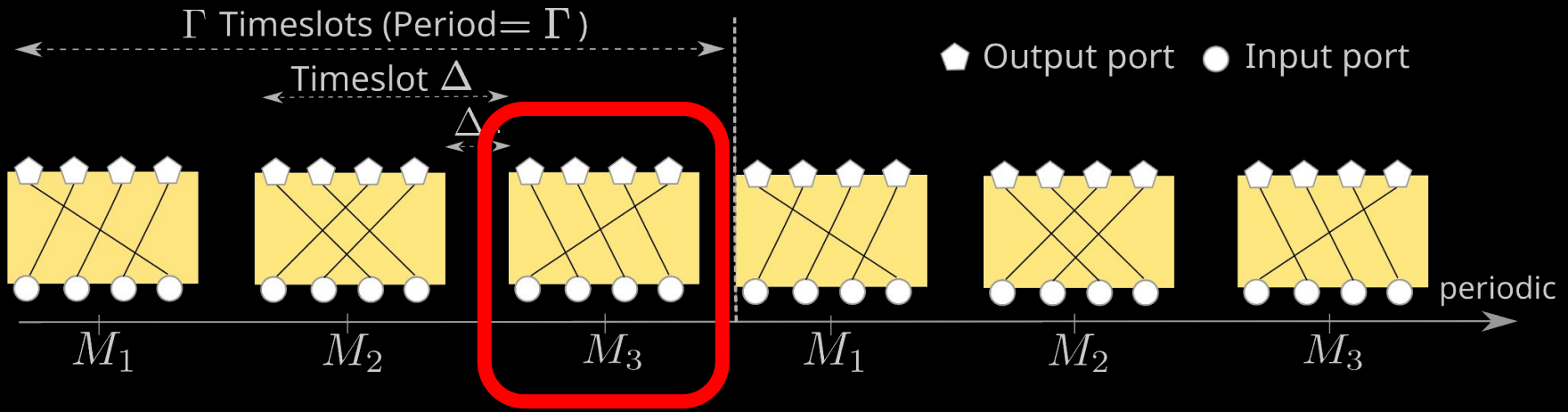

Γ Timeslots (Period$=\Gamma$)
Timeslot Δ
Δ_r
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3


Γ Timeslots (Period= Γ)
Timeslot Δ
Δ_r
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3

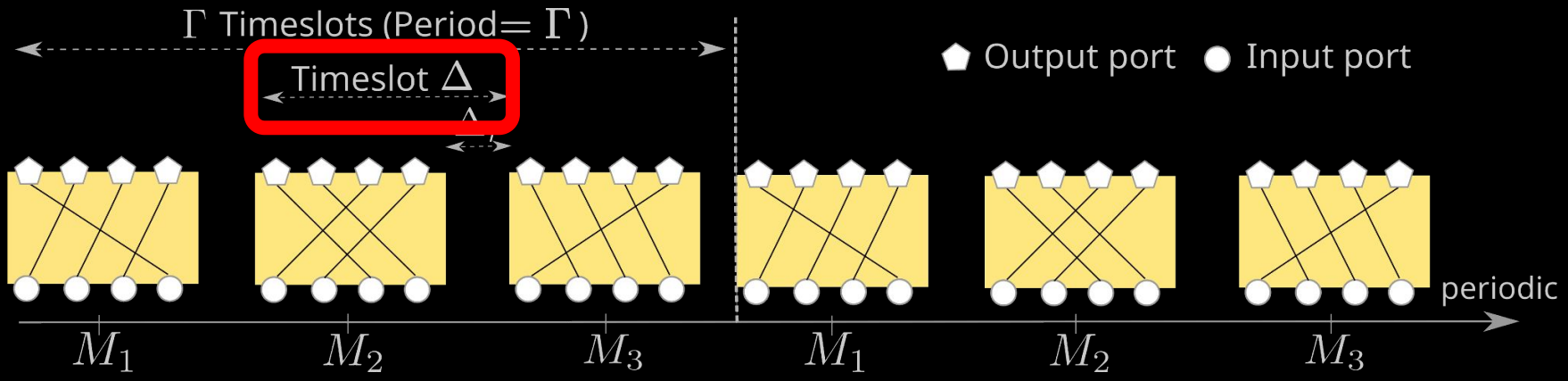
MARS


Γ Timeslots (Period= Γ)
Timeslot Δ
Δ_r
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3

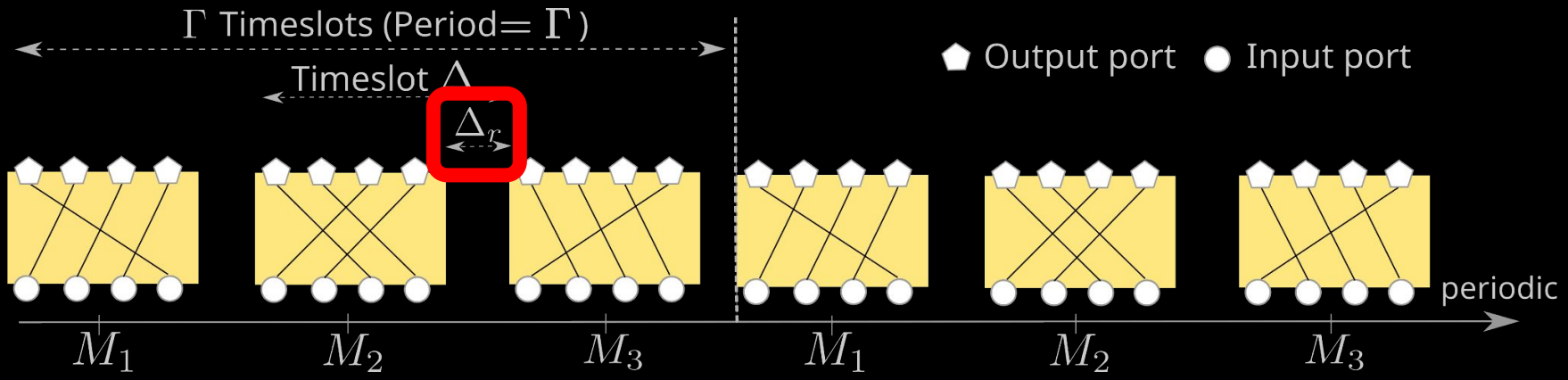
MARS

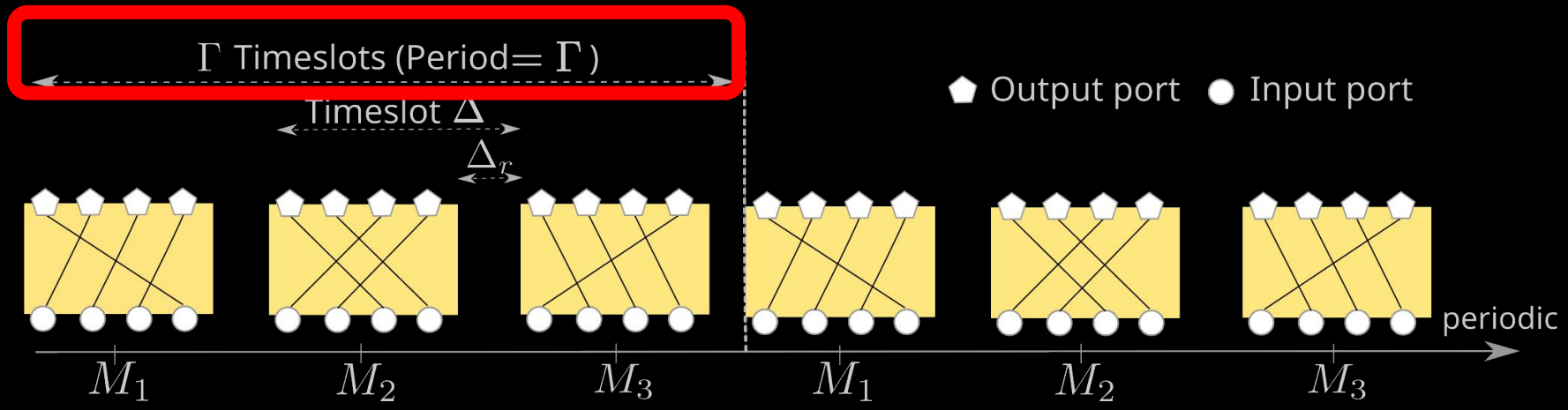

Γ Timeslots (Period= Γ)
Timeslot Δ
Δ
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3

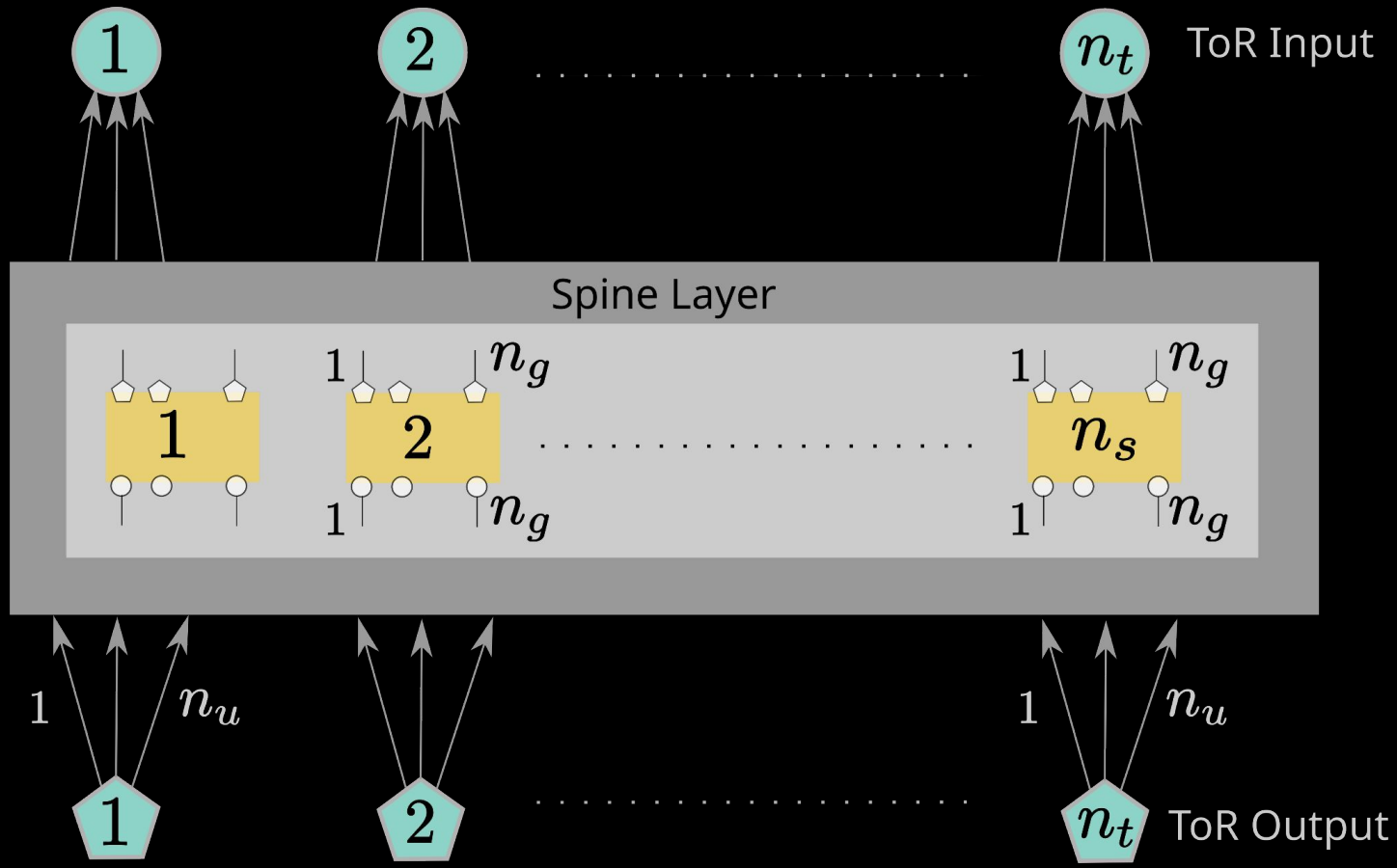
MARS


Γ Timeslots (Period$=\Gamma$)
Timeslot Δ
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3

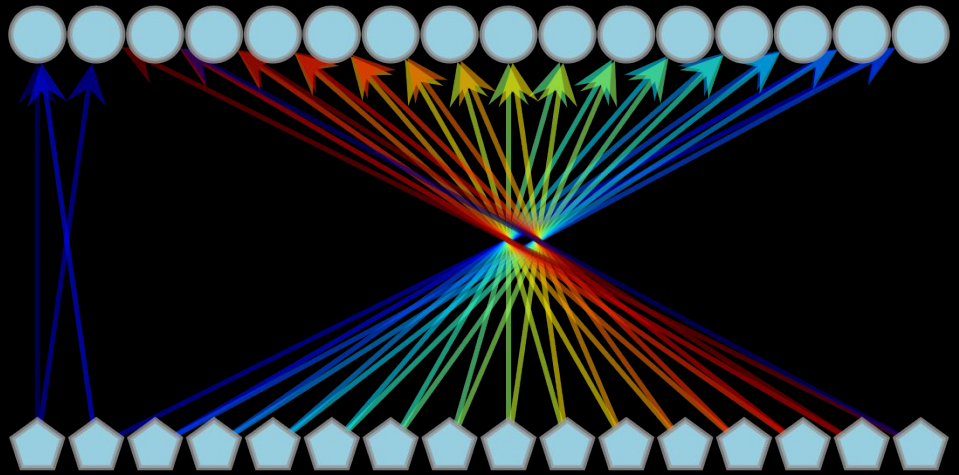
MARS


Γ Timeslots (Period$=\Gamma$)
Timeslot Δ
Δ_r
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3


Γ Timeslots (Period$=\Gamma$)
Timeslot Δ
Δ_r
Output port
Input port
periodic
M_1
M_2
M_3
M_1
M_2
M_3

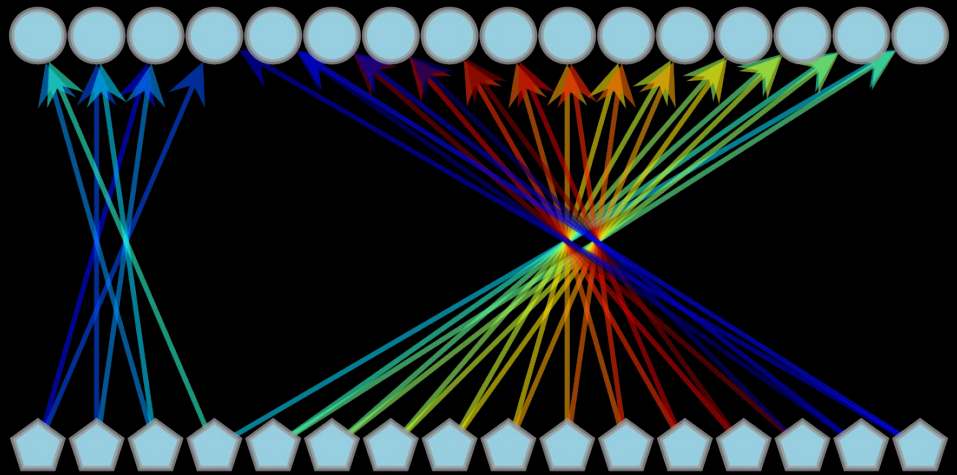

1
2
n_t
ToR Input
Spine Layer
1
1
n_g
2
1
n_g
1
n_g
n_s
1
n_g
1
n_u
1
n_u
1
2
n_t
ToR Output

Timeslot 1



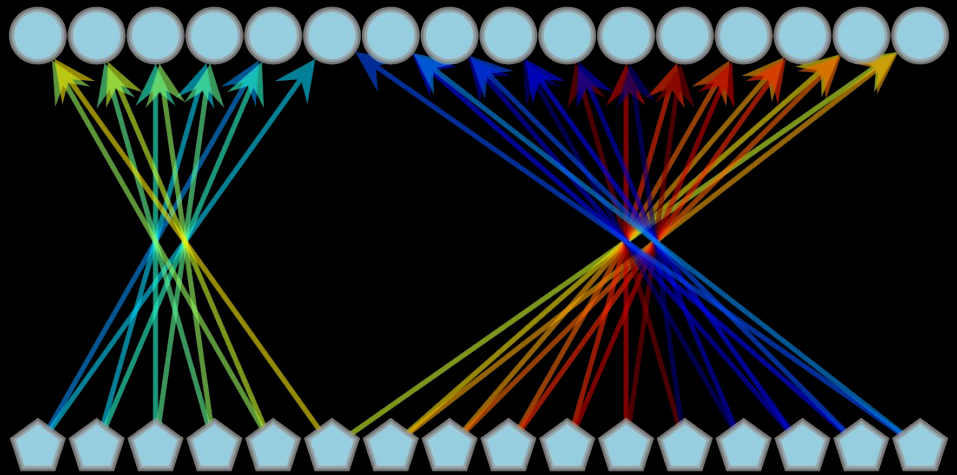
MARS

Timeslot 2



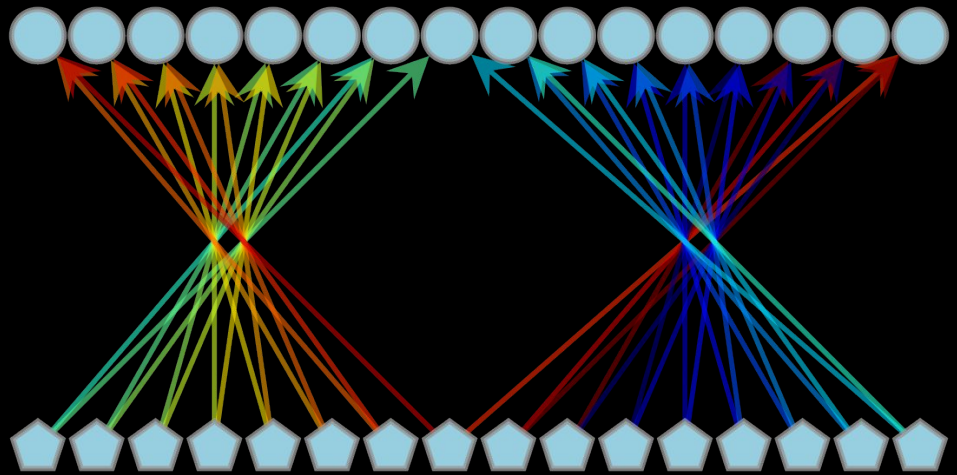
MARS

Timeslot 3



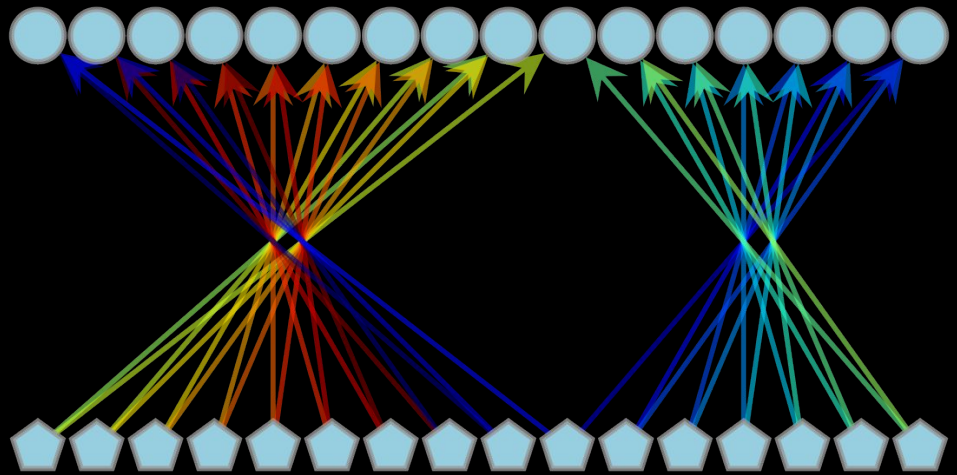
MARS

Timeslot 4



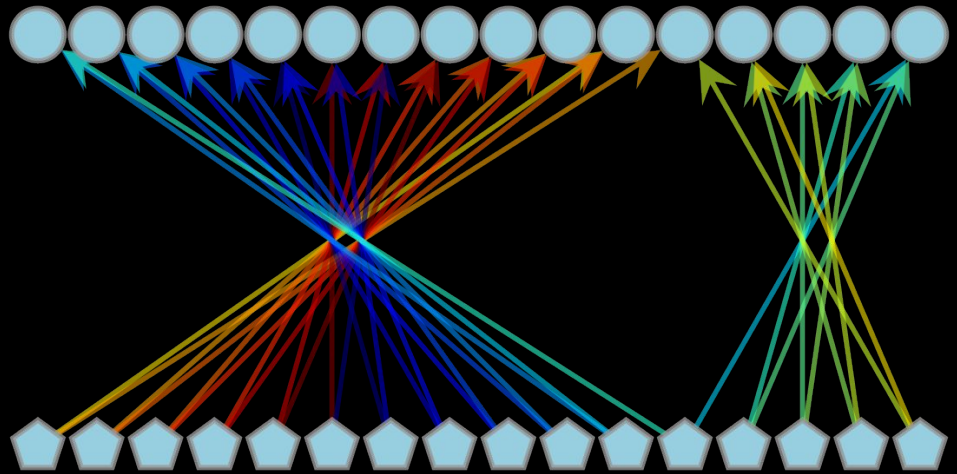
MARS

Timeslot 5



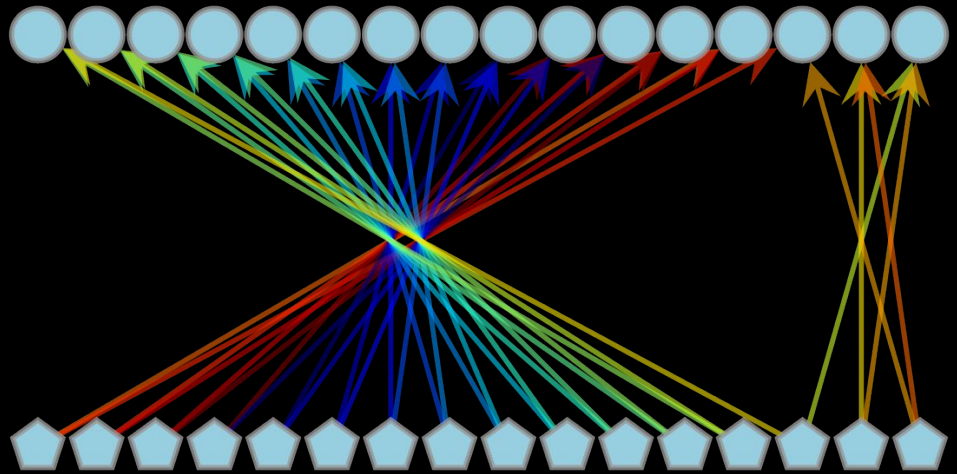
MARS

Timeslot 6



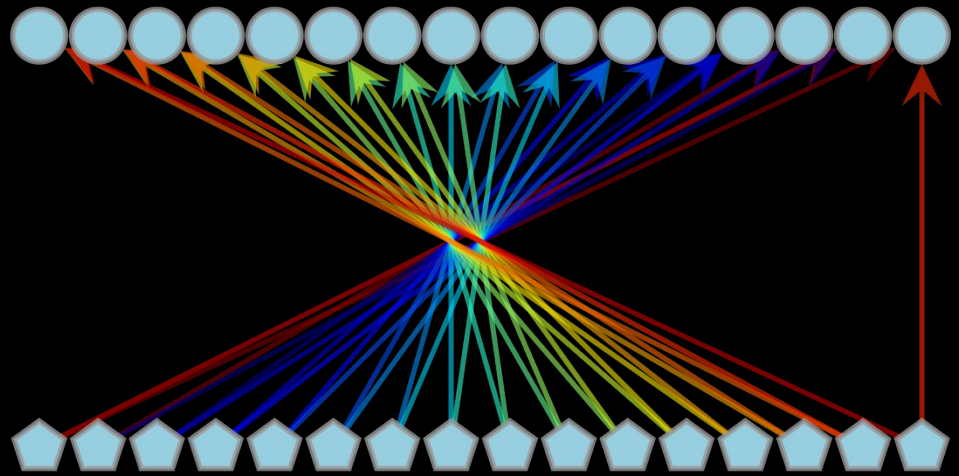
MARS

Timeslot 7



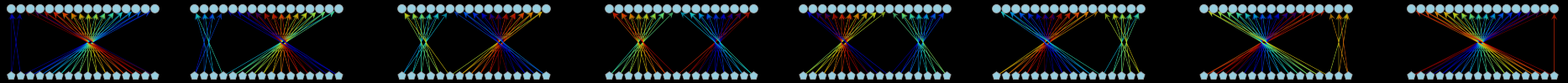
MARS

Timeslot 8



MARS

Periodic Graph



Optimal Topology

Input: Demand Matrix $\mathcal{M}$

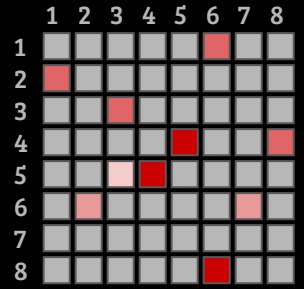
Available buffer size

Output: Periodic graph

Maximize: Throughput

MARS

Throughput



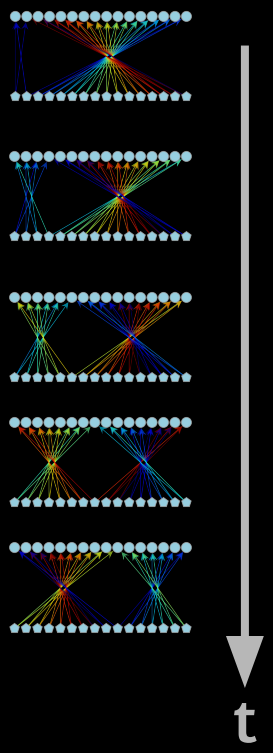
$\mathcal{M}$ X $\theta(\mathcal{M})$

Demand Matrix

Throughput

Highest scaling factor such that the scaled demand $\theta(\mathcal{M})$ is feasible in the periodic graph

Periodic Graph



MARS

Throughput of the Periodic Graph

Input: Periodic Graph $\mathcal{G}$

Demand Matrix $\mathcal{M}$

Objective: Maximize $\theta(\mathcal{M})$

Output: $\theta(\mathcal{M})$ and a feasible *flow**

**subject to conservation, demand and capacity constraints*

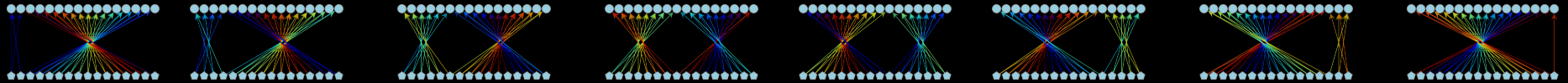
MARS

Theorem 1: Periodic Graph ⟷ Static Graph

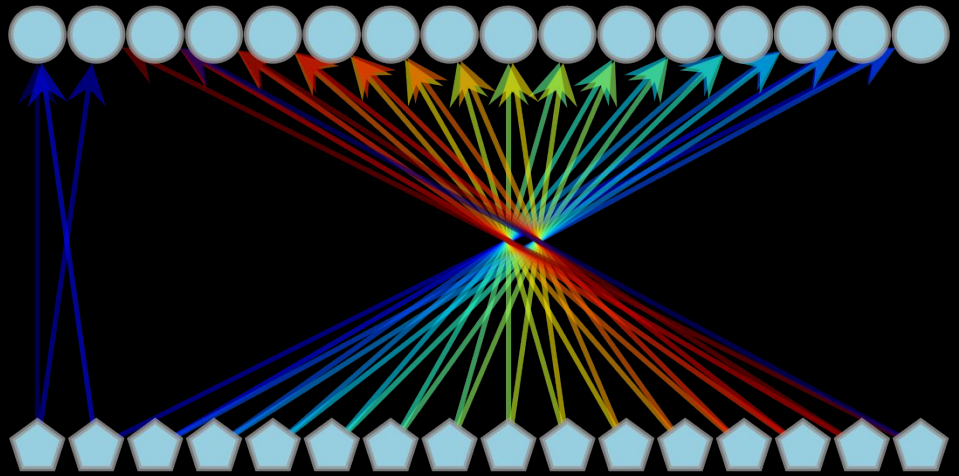
- The periodic graph has the same throughput as that of a static graph it emulates - *Static Emulated Graph*

MARS

Periodic Graph

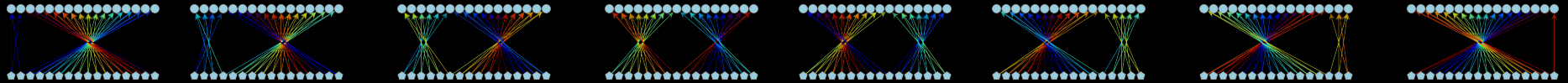


Static Emulated Graph

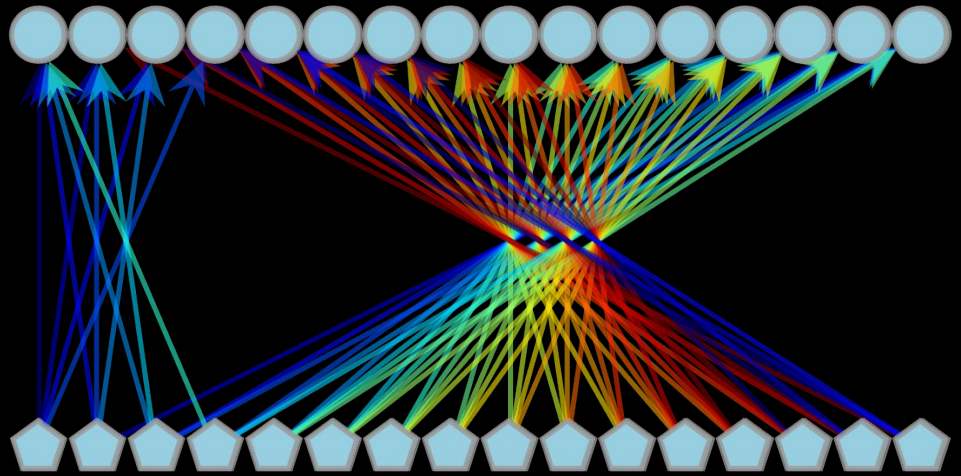


MARS

Periodic Graph

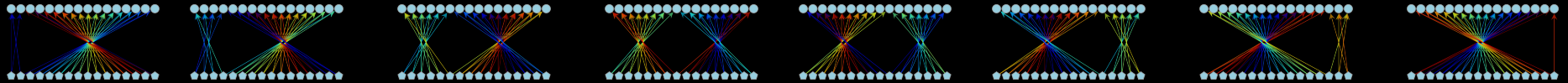


Static Emulated Graph

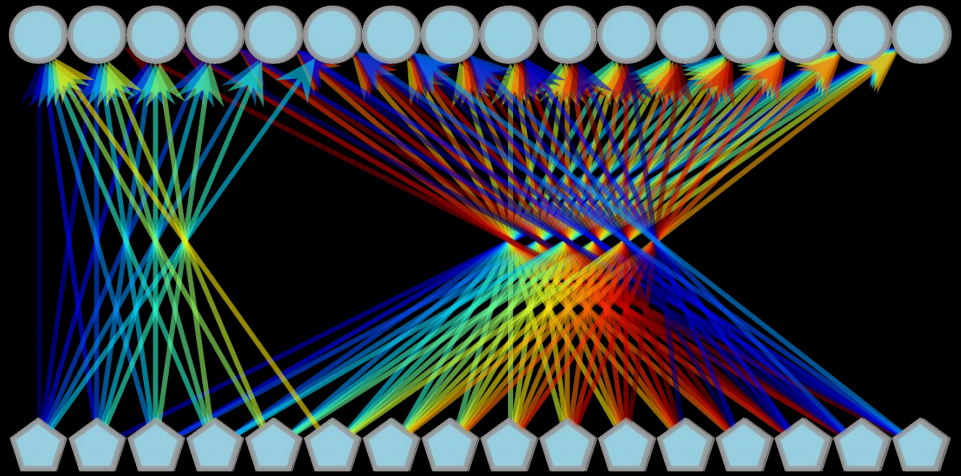


MARS

Periodic Graph

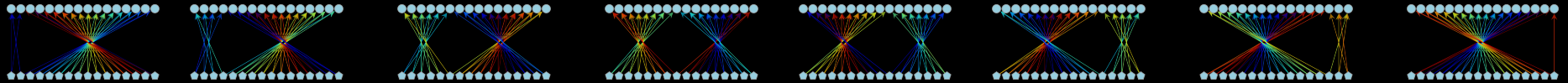


Static Emulated Graph

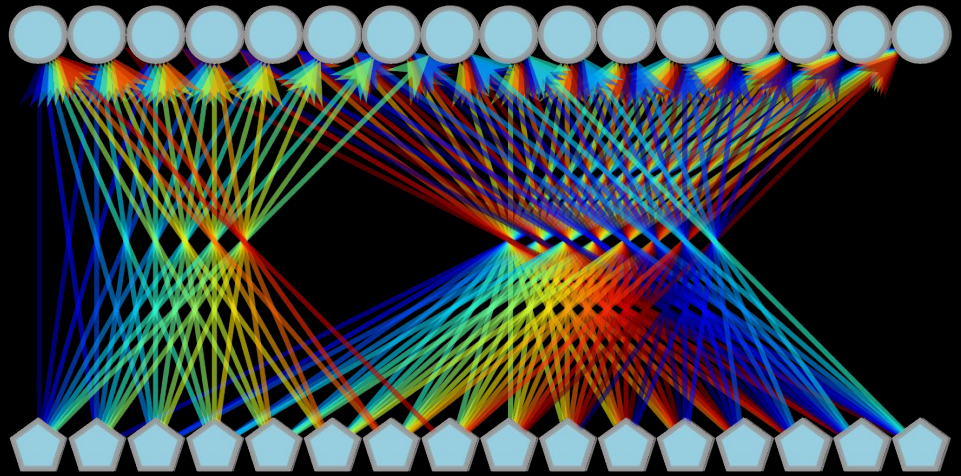


MARS

Periodic Graph

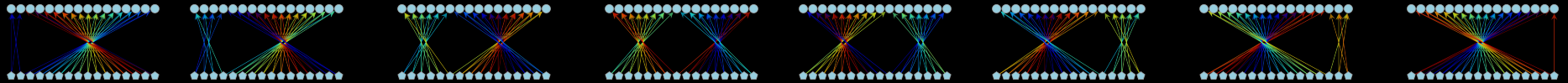


Static Emulated Graph

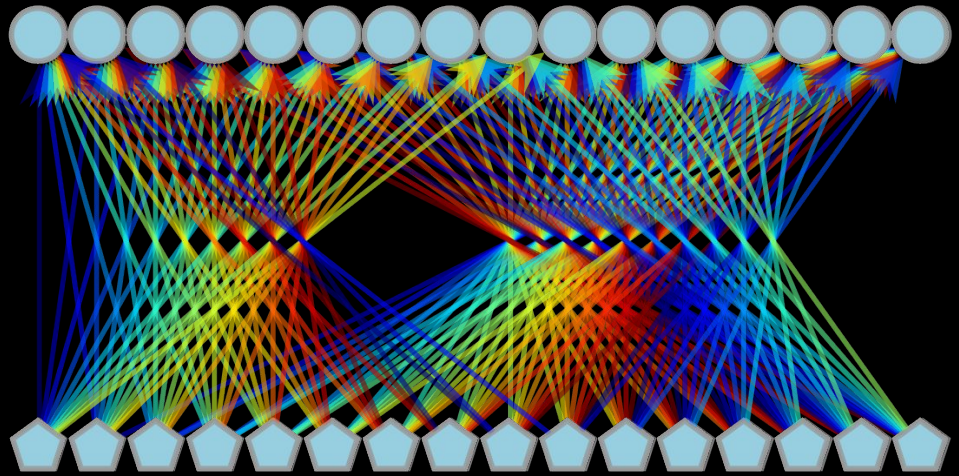


MARS

Periodic Graph

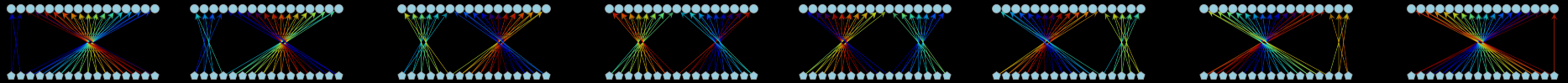


Static Emulated Graph

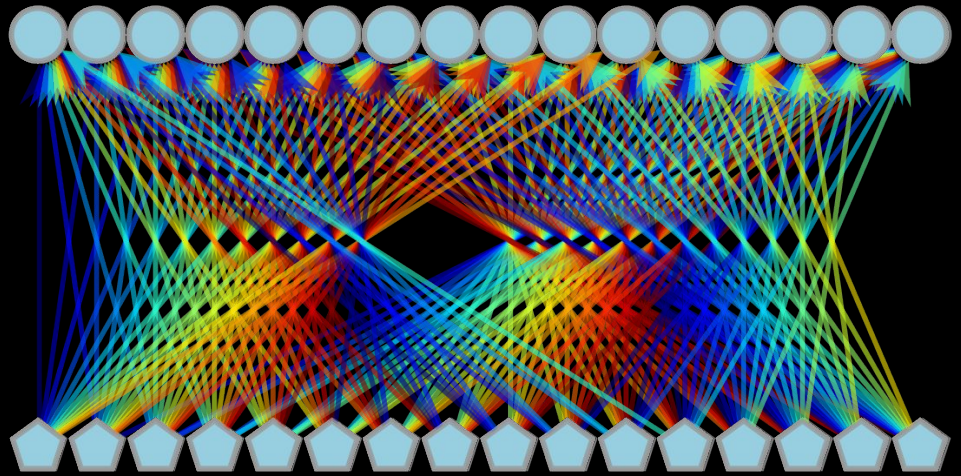


MARS

Periodic Graph

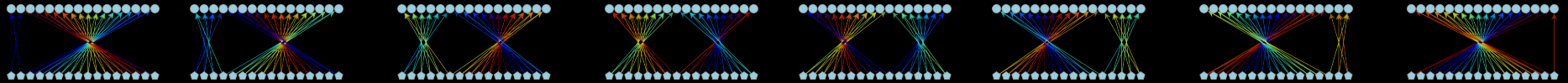


Static Emulated Graph

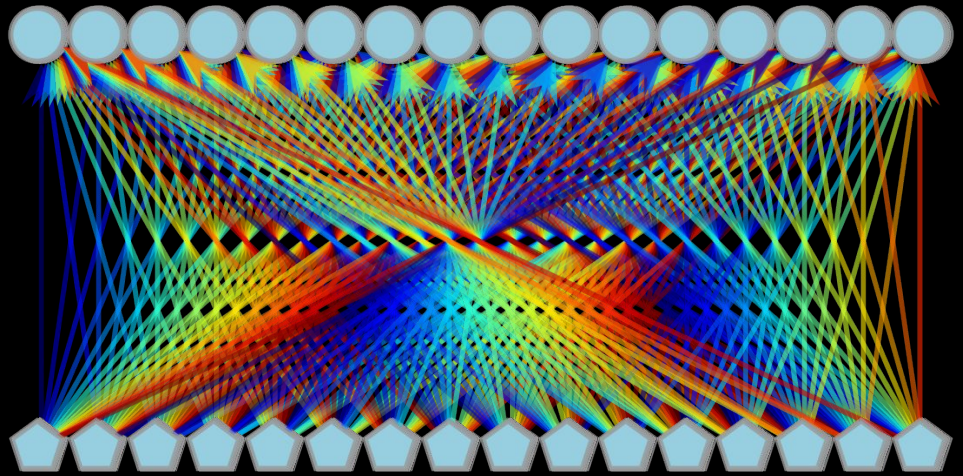


MARS

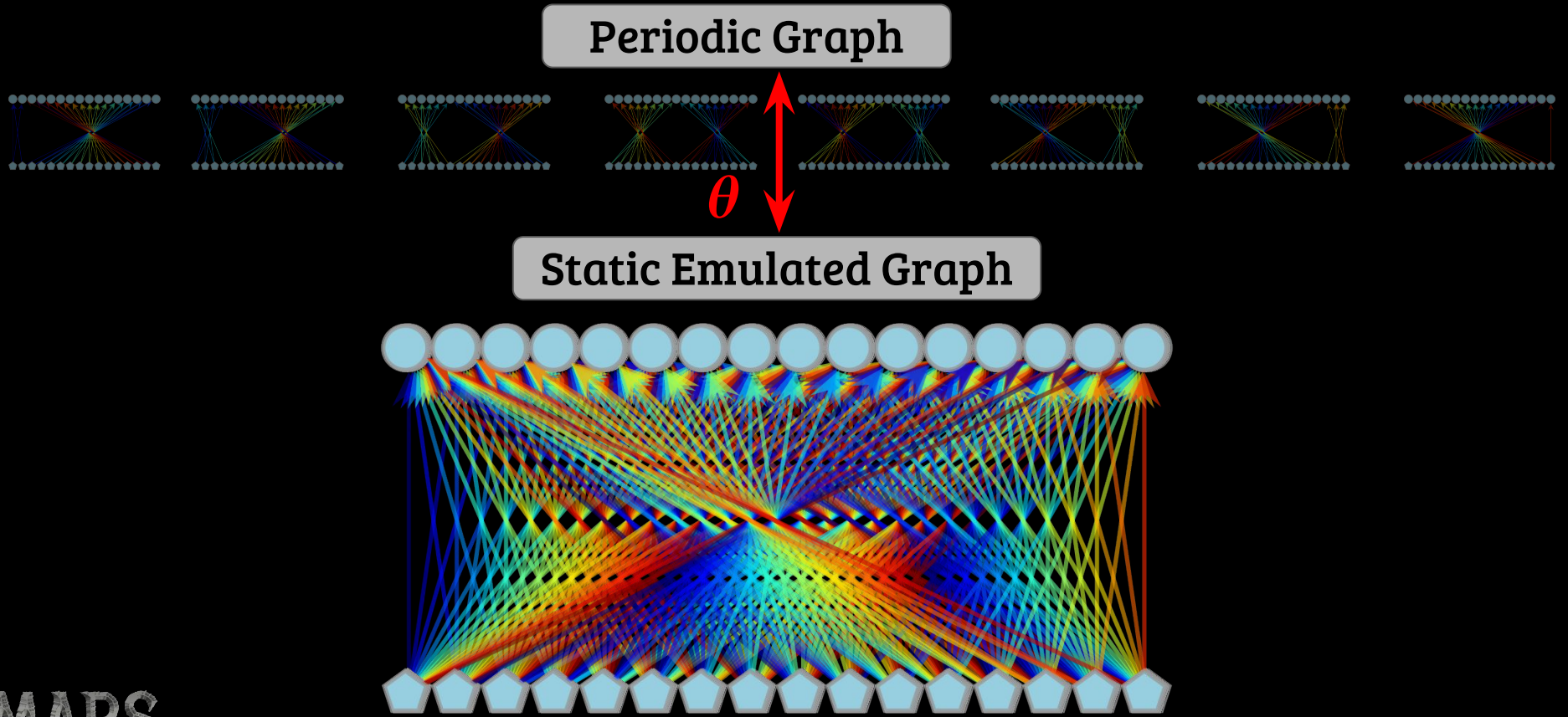
Periodic Graph



Static Emulated Graph



MARS



MARS

Throughput of the Periodic Graph

Input: Periodic Graph $\mathcal{G}$

Demand Matrix $\mathcal{M}$

Objective: Maximize $\theta(\mathcal{M})$

Output: $\theta(\mathcal{M})$ and a feasible *flow**

**subject to conservation, demand and capacity constraints*

MARS

Throughput of the Periodic Graph

Input: ~~Periodic Graph $\mathcal{G}$~~ Static Emulated Graph G

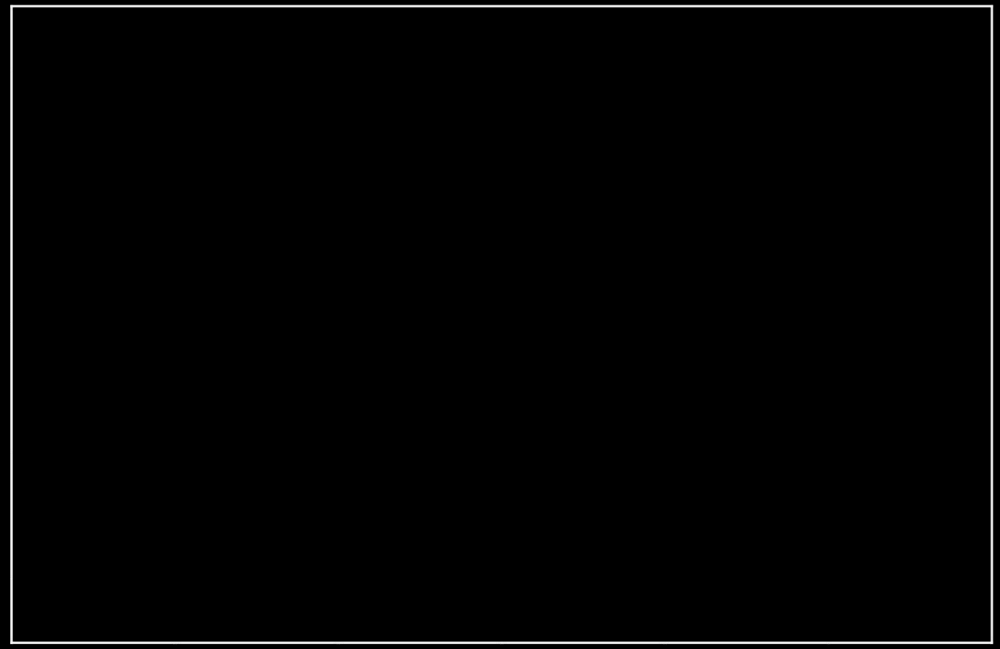
Demand Matrix $\mathcal{M}$

Objective: Maximize $\theta(\mathcal{M})$

Output: $\theta(\mathcal{M})$ and a feasible *flow**

**subject to conservation, demand and capacity constraints*

MARS



Uni-regular
(Static DCNs)

Complete Graph
(Existing RDCNs)

Degree *Increasing* →

Emulated Graph (Topology Over Time)

MARS

Theorem 2: Throughput Upper Bound

- Throughput is a function of ***Average Route Length***

$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \cdot \mathrm{ARL}(\mathcal{M}, F)}$$

MARS

Theorem 2: Throughput Upper Bound

- Throughput is a function of ***Average Route Length***

$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \cdot \mathrm{ARL}(\mathcal{M}, F)}$$

MARS

Theorem 2: Throughput Upper Bound

- Throughput is a function of ***Average Route Length***

$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \cdot \mathrm{ARL}(\mathcal{M}, F)}$$

MARS

Theorem 2: Throughput Upper Bound

- Throughput is a function of ***Average Route Length***

$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \, \mathrm{ARL}(\mathcal{M}, F)}$$

MARS

Theorem 2: Throughput Upper Bound

- Throughput is a function of ***Average Route Length***

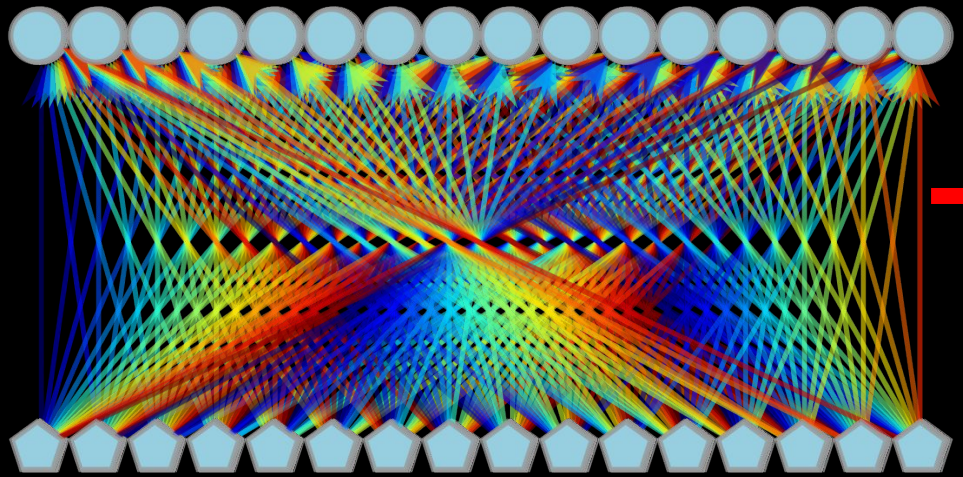
$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \cdot \mathrm{ARL}(\mathcal{M}, F)}$$

MARS

Periodic Graph

θ

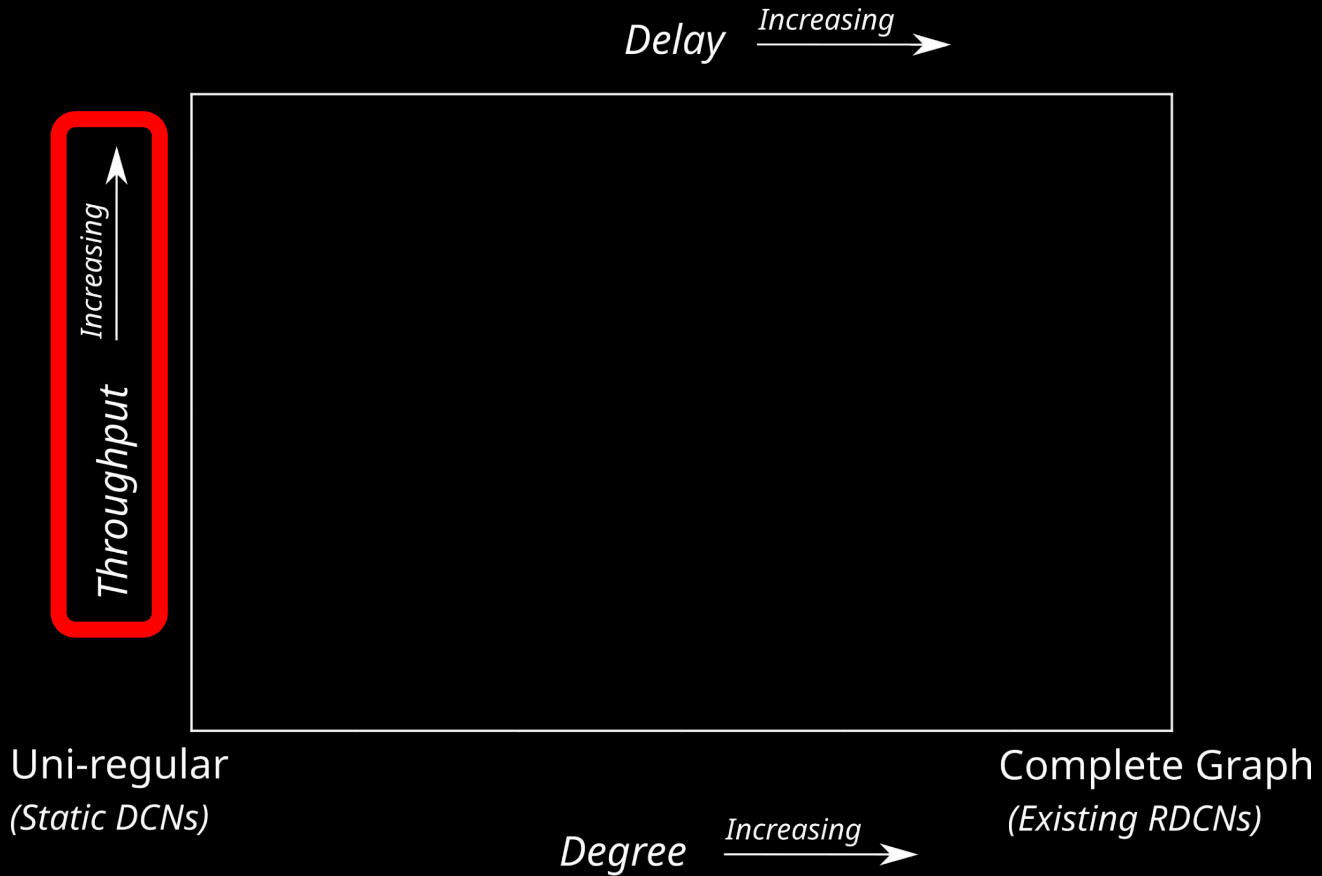
Static Emulated Graph



$$\theta(\mathcal{M}, F) \leq \frac{\hat{C}}{M \cdot \mathrm{ARL}(\mathcal{M}, F)}$$

Capacity / Demand x ARL

MARS


Delay
Increasing
Throughput
Increasing
Uni-regular
(Static DCNs)
Complete Graph
(Existing RDCNs)
Degree
Increasing
Emulated Graph (Topology Over Time)

MARS

Delay

Input: Periodic Graph $\mathcal{G}$

Demand Matrix $\mathcal{M}$

Throughput θ and a feasible flow $\mathcal{F}$

Output: Minimum worst-case delay

MARS

Theorem 3: Delay

- Delay bound is a function of:
  - Degree **d** of the emulated graph
  - Duration of the period $\mathbf{\Gamma \cdot \Delta}$
  - Throughput θ

$$L_{max} \geq \mathrm{ARD}(\mathcal{M}, \mathcal{F}) = \mathrm{ARL}(\mathcal{M}, \mathcal{F}) \cdot \Gamma \cdot \Delta$$
$$\geq \Omega\left(\frac{d \cdot \Delta}{n_u \cdot \theta(\mathcal{M}, \mathcal{F})}\right)$$

MARS

Theorem 3: Delay

- Delay bound is a function of:
  - Degree **d** of the emulated graph
  - Duration of the period $\Gamma \cdot \Delta$
  - Throughput θ

$$L_{max} \geq \mathrm{ARD}(\mathcal{M}, \mathcal{F}) = \mathrm{ARL}(\mathcal{M}, \mathcal{F}) \cdot \Gamma \cdot \Delta$$

$$\geq \Omega\left(\frac{d \cdot \Delta}{n_u \cdot \theta(\mathcal{M}, \mathcal{F})}\right)$$

MARS

Theorem 3: Delay

- Delay bound is a function of:
  - Degree **d** of the emulated graph
  - Duration of the period $\Gamma \cdot \Delta$
  - Throughput θ

$$L_{max} \geq \mathrm{ARD}(\mathcal{M}, \mathcal{F}) = \mathrm{ARL}(\mathcal{M}, \mathcal{F}) \cdot \Gamma \cdot \Delta$$

$$\geq \Omega\left(\frac{d \cdot \Delta}{n_u \cdot \theta(\mathcal{M}, \mathcal{F})}\right)$$

MARS

Theorem 3: Delay

- Delay bound is a function of:
  - Degree **d** of the emulated graph
  - Duration of the period $\Gamma \cdot \Delta$
  - Throughput θ

$$L_{max} \geq \mathrm{ARD}(\mathcal{M}, \mathcal{F}) = \mathrm{ARL}(\mathcal{M}, \mathcal{F}) \cdot \Gamma \cdot \Delta$$

$$\geq \Omega\left(\frac{d \cdot \Delta}{n_u \cdot \theta(\mathcal{M}, \mathcal{F})}\right)$$

MARS

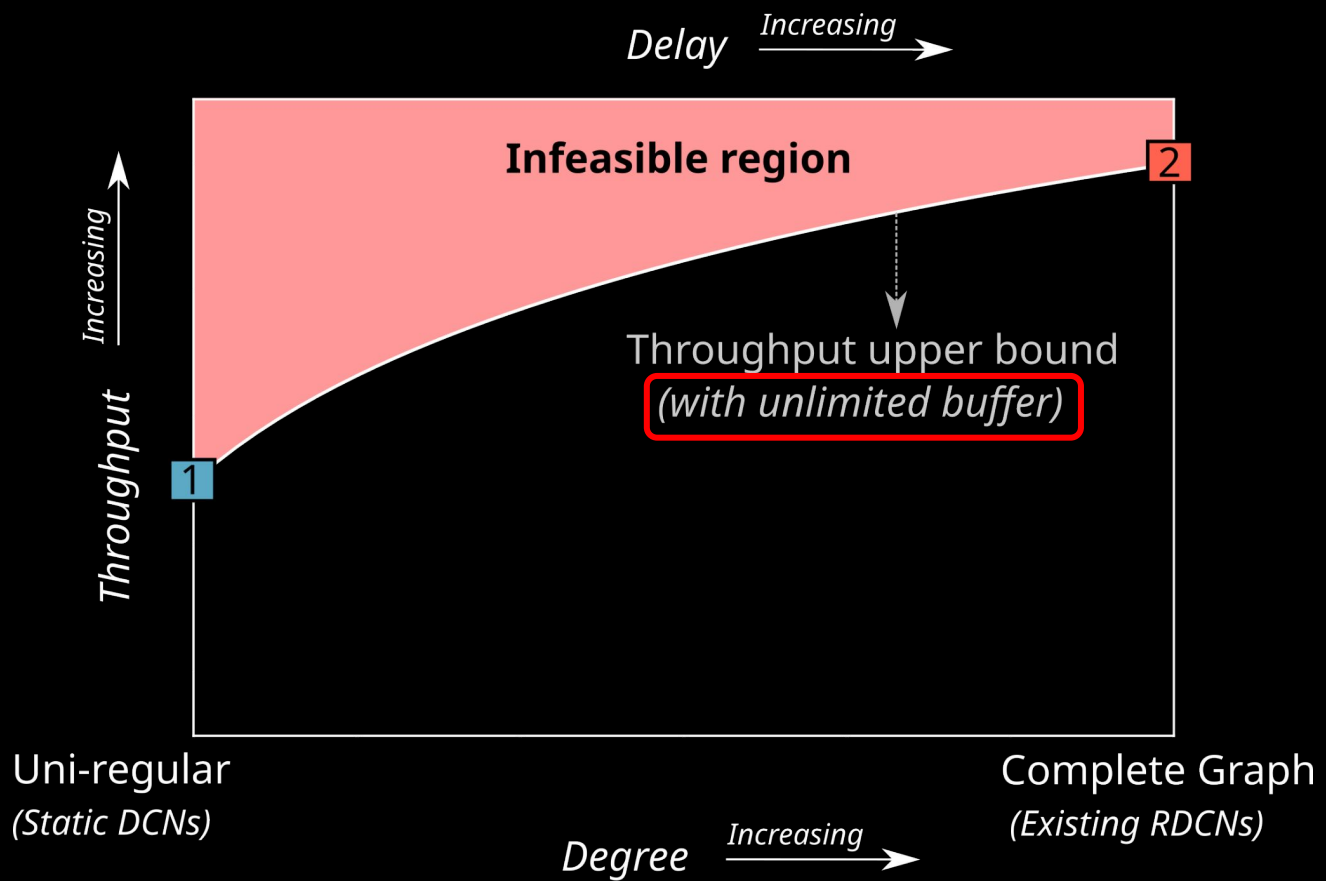
Theorem 3: Delay

- Delay bound is a function of:
  - Degree **d** of the emulated graph
  - Duration of the period $\Gamma \cdot \Delta$
  - Throughput θ

$$L_{max} \geq \mathrm{ARD}(\mathcal{M}, \mathcal{F}) = \mathrm{ARL}(\mathcal{M}, \mathcal{F}) \cdot \Gamma \cdot \Delta$$

$$\geq \Omega\left(\frac{d \cdot \Delta}{n_u \cdot \theta(\mathcal{M}, \mathcal{F})}\right)$$

MARS


Delay
Increasing
Infeasible region
2
Increasing
Throughput
Throughput upper bound
(with unlimited buffer)
1
Uni-regular
(Static DCNs)
Complete Graph
(Existing RDCNs)
Degree
Increasing
Emulated Graph (Topology Over Time)

MARS

Buffer Requirements

Input: Periodic Graph $\mathscr{G}$

Demand Matrix $\mathscr{M}$

Throughput θ and a feasible flow $\mathscr{F}$

Output: Minimum required buffer to achieve throughput θ

MARS

Theorem 4: Buffer Requirements

- The required buffer is at least the ***throughput delay product***

$$\hat{B} \geq (\theta(\mathcal{M}, \mathcal{F}) \cdot M) \cdot \mathrm{ARD}(\mathcal{M}, \mathcal{F})$$

MARS

Theorem 4: Buffer Requirements

- The required buffer is at least the ***throughput delay product***

$$\hat{B} \geq (\theta(\mathcal{M}, \mathcal{F}) \cdot M) \cdot \mathrm{ARD}(\mathcal{M}, \mathcal{F})$$

MARS

Theorem 4: Buffer Requirements

- The required buffer is at least the ***throughput delay product***

$$\hat{B} \geq (\theta(\mathcal{M}, \mathcal{F}) \cdot M) \cdot \mathrm{ARD}(\mathcal{M}, \mathcal{F})$$

MARS

Theorem 4: Buffer Requirements

- The required buffer is at least the *throughput delay product*

$$\hat{B} \geq (\theta(\mathcal{M}, \mathcal{F}) \cdot M) \cdot \mathrm{ARD}(\mathcal{M}, \mathcal{F})$$

MARS

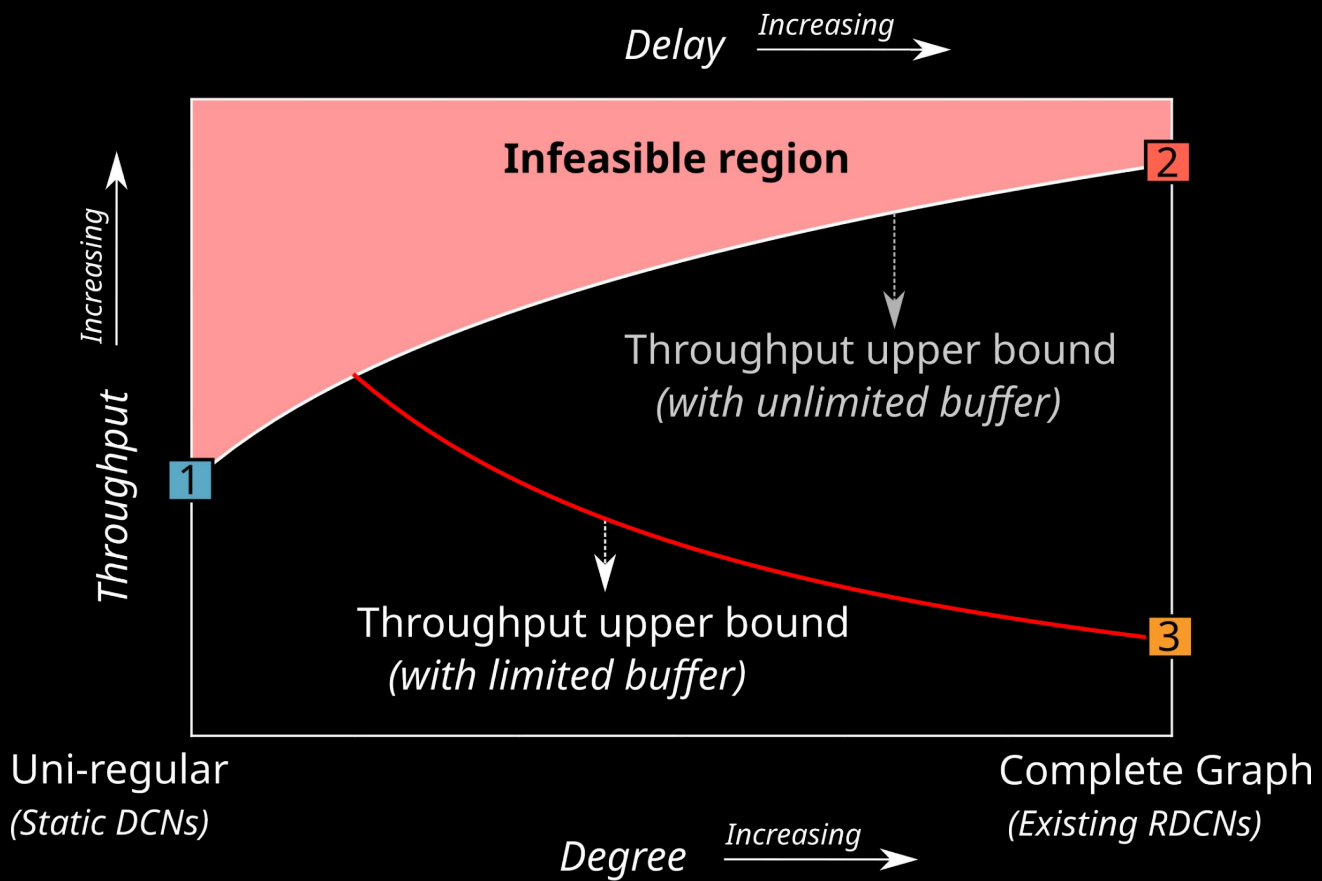
Theorem 4: Buffer Requirements

- The required buffer is at least the *throughput delay product*

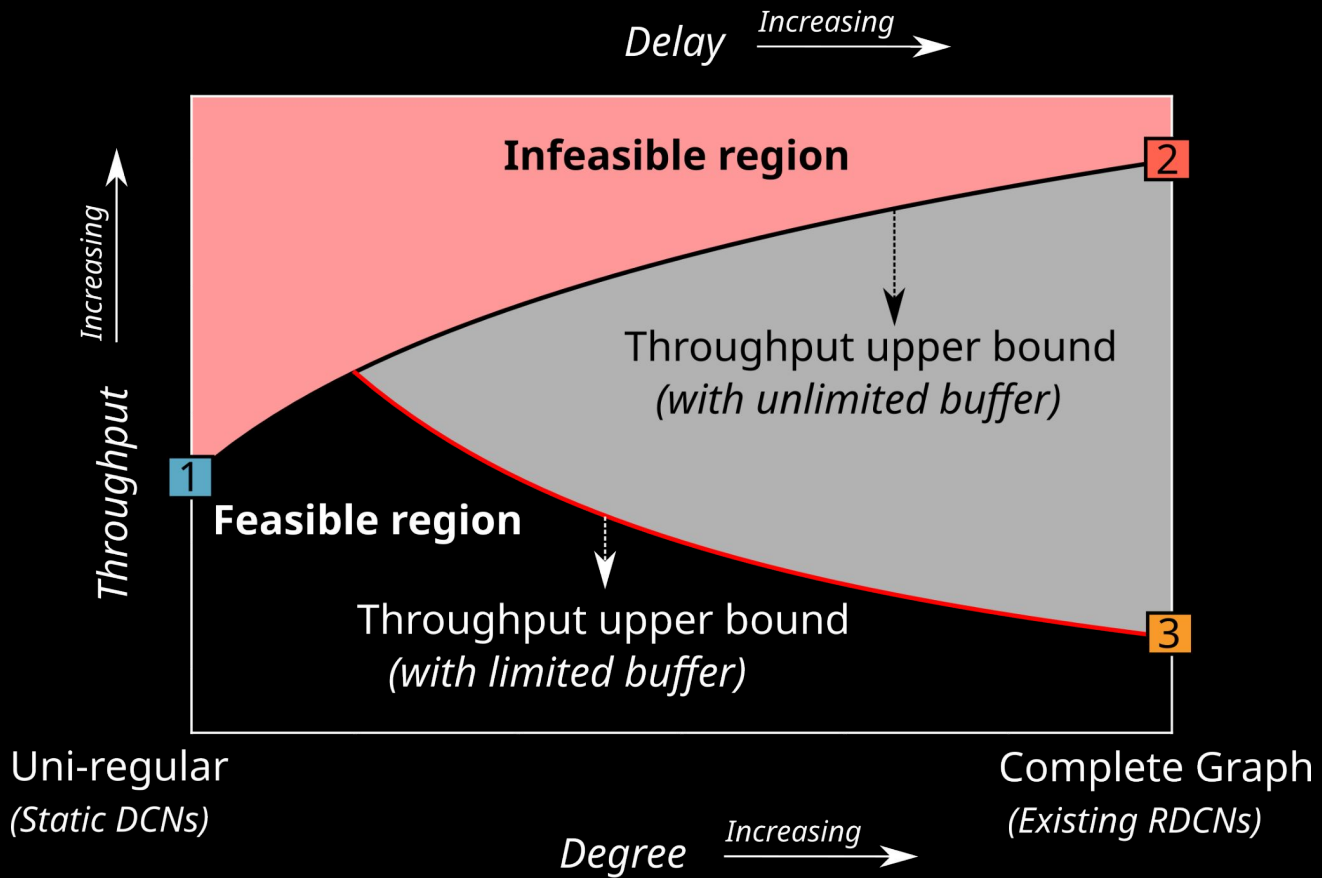
$$\hat{B} \geq (\theta(\mathcal{M}, \mathcal{F}) \cdot M) \cdot \mathrm{ARD}(\mathcal{M}, \mathcal{F})$$

Buffer > Bandwidth x Delay

MARS


Delay
Increasing
Infeasible region
2
Increasing
Throughput
Throughput upper bound
(with unlimited buffer)
1
Throughput upper bound
(with limited buffer)
3
Uni-regular
(Static DCNs)
Complete Graph
(Existing RDCNs)
Degree
Increasing
Emulated Graph (Topology Over Time)

MARS


Delay
Increasing
Infeasible region
2
Increasing
Throughput
Throughput upper bound
(with unlimited buffer)
1
Feasible region
Throughput upper bound
(with limited buffer)
3
Uni-regular
(Static DCNs)
Complete Graph
(Existing RDCNs)
Degree
Increasing
Emulated Graph (Topology Over Time)

Goals

- Maximize Throughput
- Minimize Latency
- Minimize Buffer Requirements

Optimal Topology with Buffer Constraints

Input: ~~Periodic Graph $\mathcal{G}$~~

Demand Matrix $\mathcal{M}$

Available buffer size B at each node

Output: Degree d of the emulated graph → Periodic graph

Maximize: Throughput θ

MARS

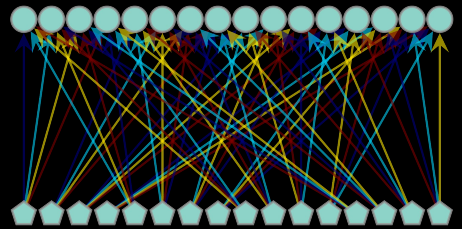
Optimal Topology with Buffer Constraints

Output: $d = B/(c \cdot \Delta)$

Optimal Topology with Buffer Constraints

Output: $d = B/(c \cdot \Delta)$

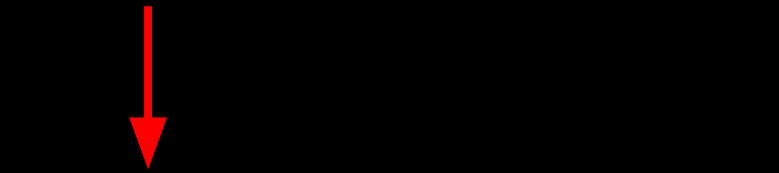
d-regular directed deBruijn graph



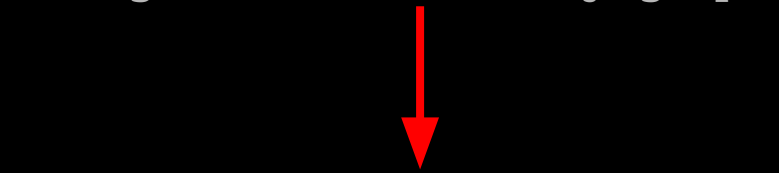
MARS

Optimal Topology with Buffer Constraints

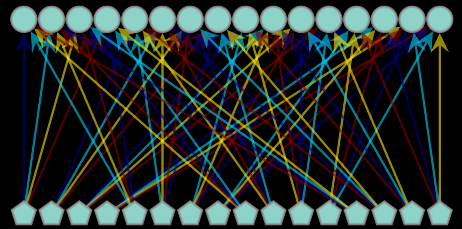
Output: $d = B/(c \cdot \Delta)$



d-regular directed deBruijn graph



Decomposition to d matchings



MARS

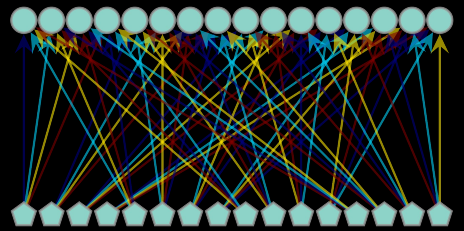
Optimal Topology with Buffer Constraints

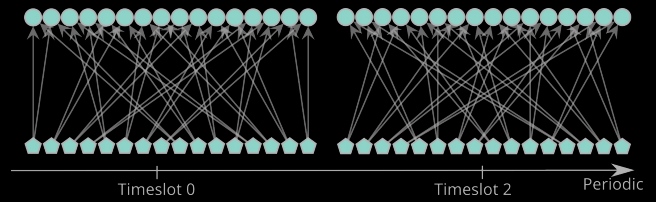
Output: $d = B/(c \cdot \Delta)$

d-regular directed deBruijn graph

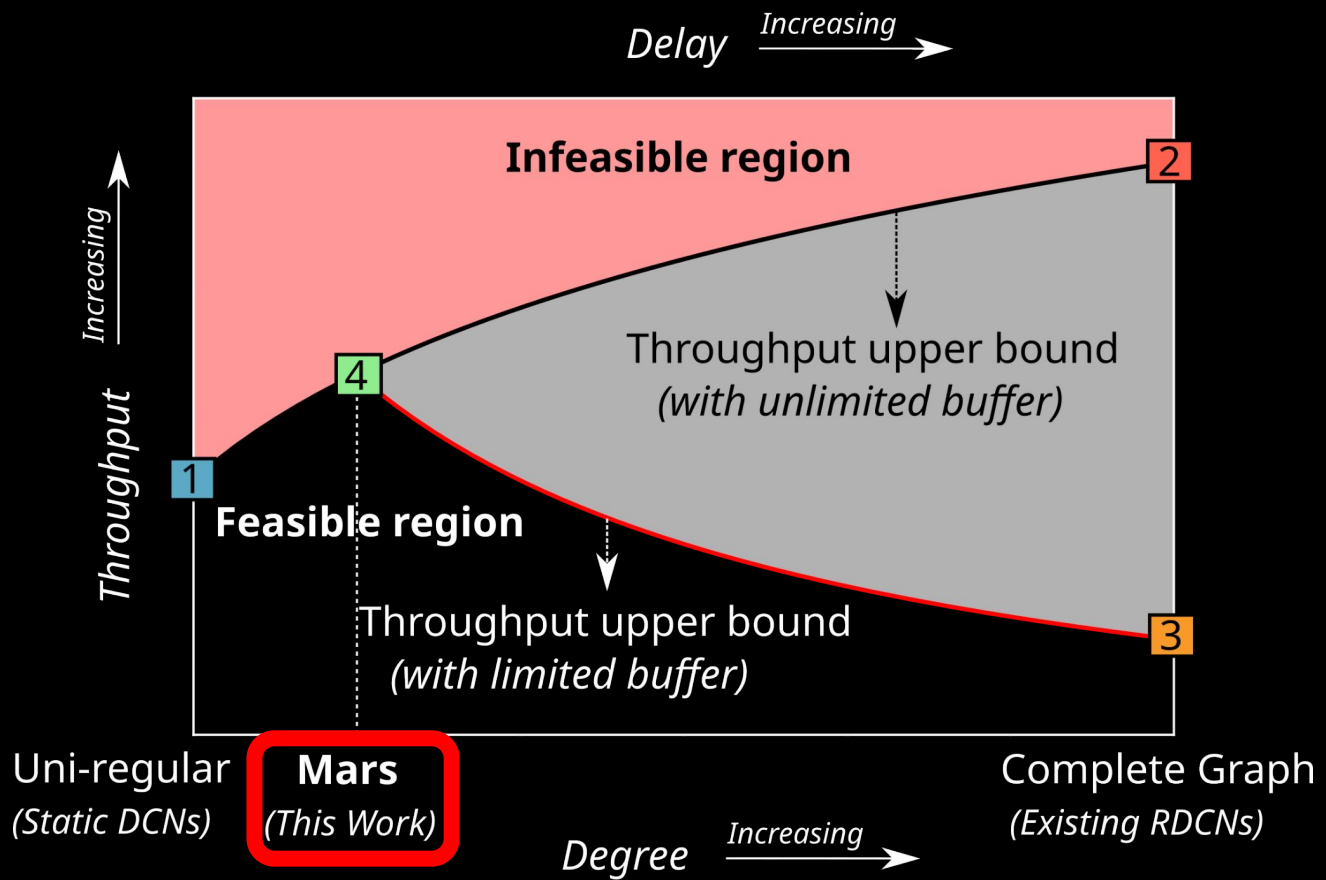
Decomposition to d matchings

Periodic graph





MARS


Delay
Increasing
Infeasible region
2
Increasing
Throughput
4
Throughput upper bound
(with unlimited buffer)
1
Feasible region
Throughput upper bound
(with limited buffer)
3
Uni-regular
(Static DCNs)
Mars
(This Work)
Complete Graph
(Existing RDCNs)
Degree
Increasing
Emulated Graph (Topology Over Time)

MARS

Optimal Topology Implications and Future Outlook

Output: $d = B/(c \cdot \Delta)$

If the reconfiguration technology (Δ) remains same:

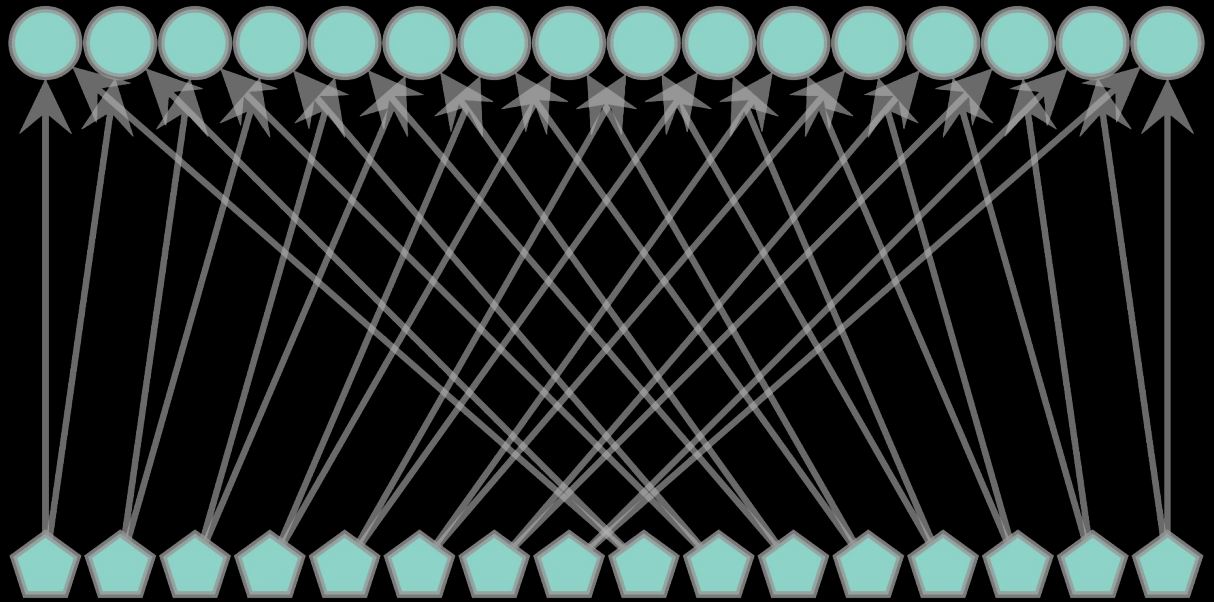
- Buffer sizes (B) must keep up with the increase in capacity (c)

If Buffer sizes (B) do not keep up:

- Increase in capacity (c) must be accompanied by decrease in reconfiguration times (Δ)
- If not, reducing the degree (d) of the emulated graph is inevitable to optimize throughput → eventually reaching the case of static topologies.

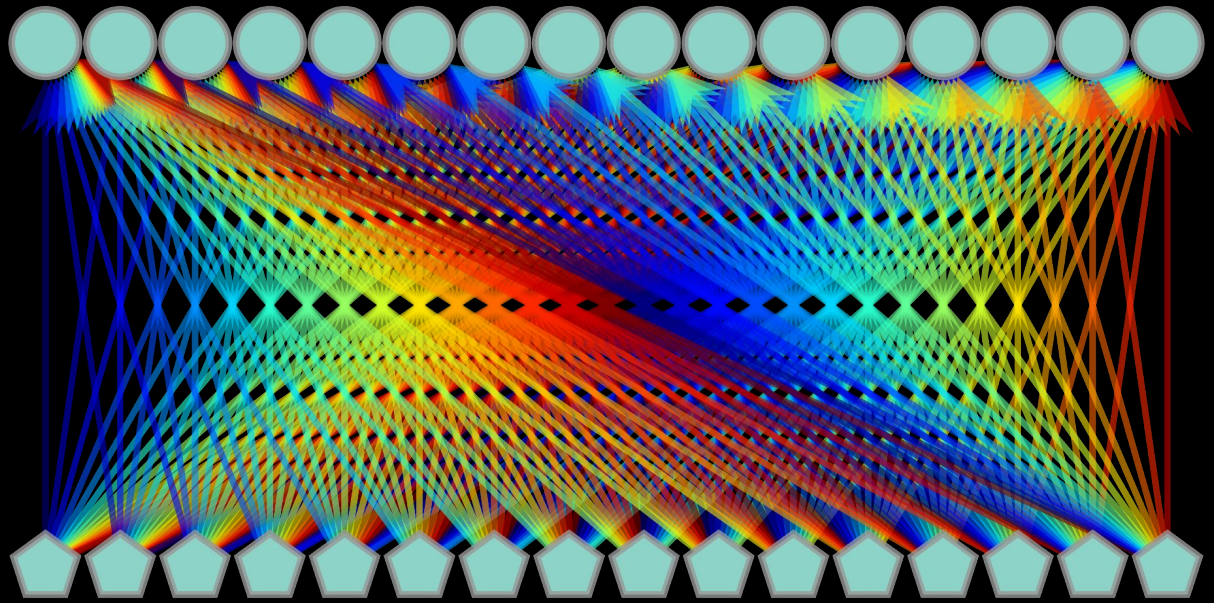
MARS

Static DCNs *(uni-regular)*



Low throughput but low delay and buffer requirements

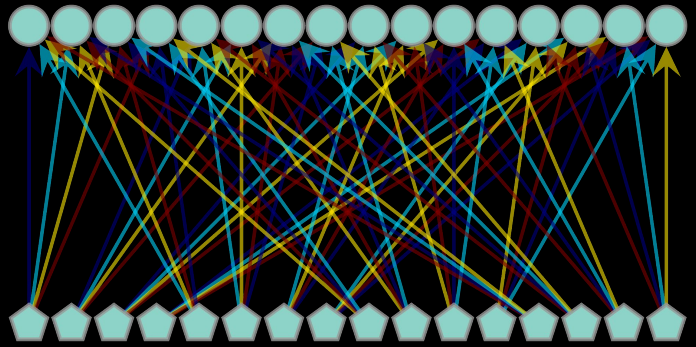
Existing RDCN designs *(Emulating a complete graph)*



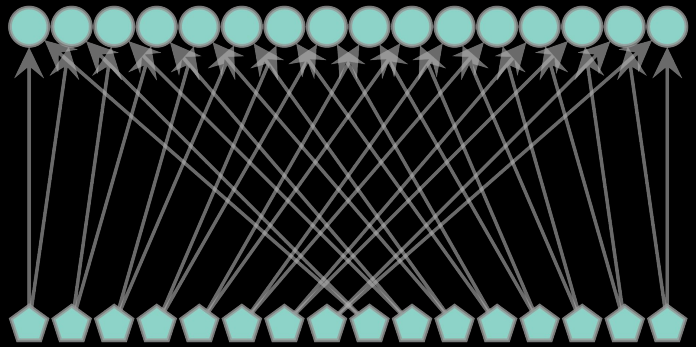
High throughput but high delay and buffer requirements

MARS

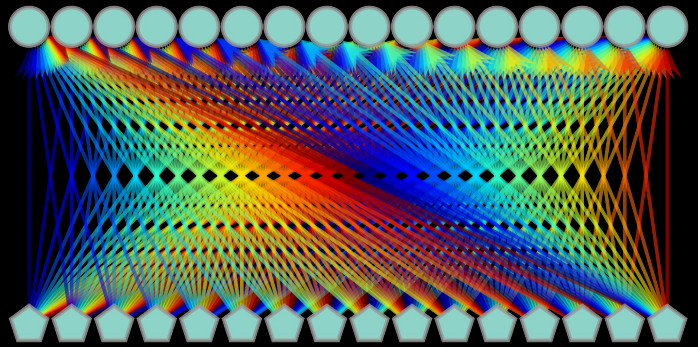
MARS



Near-optimal throughput within the available buffer



Static DCN: Low Throughput



Existing RDCN: High Delay and buffer

Thank you

Vamsi Addanki
vamsi@inet.tu-berlin.de
@Vamsi_DT

MARS