Extending Optimal Oblivious Reconfigurable Networks to all N

Tegan Wilson* Daniel Amir* Vishal Shrivastav† Hakim Weatherspoon*
Robert Kleinberg*

November 9, 2022

Abstract

Reconfigurable networks have traditionally suffered from long reconfiguration times due to hardware limitations. With the emergence of new technologies that can reconfigure on the order of nanoseconds, the Oblivious Reconfigurable Network (ORN) design paradigm has been proposed to take advantage of this new capability. Two of the most important performance metrics for network operators considering ORNs are latency and throughput, which are inherently opposed and thus lead to a tradeoff. Previously, we constructed two families of ORN designs, called EBS and VBS, and proved that together they achieve optimal maximum latency (up to a constant factor) for any given throughput value r.

Unfortunately, both families are defined only for very restrictive network sizes, N. This poses a challenge for implementing networks based on these designs in practice, as it is unlikely that a given system will have a suitable size. In this work, we extend both EBS and VBS to any network size while minimizing the impact on maximum latency and throughput. We prove that the extended versions achieve optimal maximum latency for all sufficiently large N, except when the throughput r exactly equals the reciprocal of an even integer.

1 Introduction

In a reconfigurable network, the connectivity pattern changes dynamically to enable efficient routing of messages. This idea arguably dates back to the early days of telephony when human operators manually reconfigured circuits to connect callers, but it is undergoing a renaissance due to recent technologies such as optical circuit switching [11, 30] and free-space optics [13, 16] that enable network reconfiguration within microseconds [20, 23] or even nanoseconds [8, 9]. At time scales such as these, networking becomes an intricately choreographed dance, with data traveling along paths whose links come into being while packets are in flight. The design space of network architectures that take advantage of this capability is now being actively explored, via prototype systems [22, 21, 26] and theoretical modeling and analysis [2].

There are various metrics one could use to evaluate the performance of a network design: throughput, goodput, average latency, tail latency, fault tolerance, and so on. In this work we focus on two canonical performance measures: maximum latency and guaranteed throughput. Maximum latency measures the worst-case time interval between when a packet arrives onto the network and when it is delivered to its destination. Guaranteed throughput measures the worst-case average transmission rate over a set of possible traffic demand matrices; the precise definition, which is deferred to Section 2, captures the intuition that an N-node network that guarantees throughput r should simulate a "big switch" with N input and output ports, each of line rate r, that allows an arbitrary matching between its input and output ports.

The objectives of minimizing latency and maximizing throughput in a reconfigurable network are in conflict with one another: achieving high throughput requires using routing paths composed of few physical links, which in turn necessitates waiting longer for those links to come into being as the network cycles through a sequence of interconnection topologies. In earlier work [2] we specified a mathematical model of oblivious reconfigurable networks — those in which the dynamic network topology and the distribution over routing paths are predetermined, before the traffic matrix is known — and we quantified the trade-off between latency and throughput for such networks: we identified a lower bound on maximum latency, denoted by $L^*(r, N)$, such that for every throughput rate[1] $r \in (0, \frac{1}{2}]$, the maximum latency must be at least $L^*(r, N)$. Moreover, we showed

*Cornell University
†Purdue University

[1] In our model of oblivious reconfigurable networks it is easy to prove that no network design can guarantee throughput greater than $\frac{1}{2}$; see [2] for a proof.

that this lower bound is tight up to a constant factor, in the following sense: for every $r \in (0, \frac{1}{2}]$ there exist infinitely many N admitting an oblivious reconfigurable network design that achieves latency r and throughput $O(L^*(r, N))$. To prove this result we exhibited two families of reconfigurable networks, the *Elementary Basis Scheme* (EBS) and the *Vandermonde Basis Scheme* (VBS), such that for each r, at least one of EBS and VBS is capable of attaining the stated latency bound for infinitely many N.

1.1 Our results and techniques A major limitation of the EBS and VBS designs is that they are not defined for all network sizes. The EBS design requires the number of nodes in the network, N, to be a perfect h^{th} power, where $h = \lfloor \frac{1}{2r} \rfloor$. The VBS design is even more restrictive: it requires N to be a perfect $(h+1)^{\text{th}}$ power of a prime number. In practice the number of nodes in a datacenter network would rarely satisfy these restrictions, so the EBS and VBS designs must be regarded mainly as a theoretical exercise unless this limitation can be removed. In this paper we succeed in removing the limitation on the network size. Our main result shows that there exist oblivious reconfigurable network designs achieving maximum latency $O(L^*(r, N))$ for all sufficiently large N, whenever $\frac{1}{2r}$ is not an integer.

To shed light on the challenge of proving such a result, it helps to reflect on the constrast between network design with N nodes and algorithm design with input size N. In algorithm design, it is common to define an algorithm first on special input sizes (e.g., when N is a power of two) and then to extend the algorithm to all N by "padding" the input. For example, Strassen's matrix multiplication algorithm works by reducing N-by-N matrix multiplication to seven instances of $\left(\frac{N}{2}\right)$-by-$\left(\frac{N}{2}\right)$ matrix multiplication and then solves those instances recursively. This requires N to be a power of 2; to use the same algorithm when N lies strictly between two powers of 2, one first pads the matrices with zeros until the number of rows and columns equals the next power of 2 greater than N. The padding scheme works because the extra zeros function as placeholders that have no effect on the relevant entries of the matrix product. In networking, there is no corresponding way to "pad a network" with fictional nodes that have no effect on the actual physical nodes comprising the network. Either one assigns physical nodes to play the role of the fictional ones — but then the load on those physical nodes is affected — or one allocates no physical resources to do the work of the fictional nodes, but this affects the physical nodes of the network when they try to communicate using a routing path that goes through one of the fictional nodes.

Our main technical contribution is to define a padding scheme that circumvents these difficulties and allows us to extend the EBS and VBS network designs to all sufficiently large values of N. To do so, we pad the network with "dummy nodes" until the total number of physical and dummy nodes matches one of the network sizes for which EBS or VBS is defined. Then, rather than allocating physical resources to do the work of the dummy nodes, we simply eliminate all of the flow on routing paths that pass through dummy nodes. In order to ensure that the resulting network design guarantees throughput r, in the padded network we use the EBS or VBS design with throughput guarantee $r/(1-\delta)$, where $\delta > 0$ is chosen to be small enough that the maximum latency increases by only a constant factor. To prove that the guaranteed throughput is indeed greater than or equal to r, we must show that in the worst case over all potential traffic demand matrices, the fraction of flow that EBS or VBS routes through dummy nodes (henceforth, "dummy flow") will not exceed δ. It turns out that this step of the analysis is sensitive to the placement of the dummy nodes, and one of the key technical challenges our work overcomes is to identify a way of placing dummy nodes such that for any individual source or destination node, the total amount of dummy flow originating at that source (or terminating at that destination) is not too large.

1.2 Related work The most important related work is [2], as this work extends the results described therein. Additional related work includes the following.

Oblivious routing in general networks: There exists extensive theory on oblivious routing on general networks. The most frequently analyzed performance metric is congestion, measured by competitive ratio. [24] proved that for general networks, there exist oblivious routing schemes that are polylog(N)-competitive, a bound which was subsequently improved by [17]. Further work developed polynomial-time algorithms for generating oblivious routing schemes that meet this bound [7, 17, 4], one of which was demonstrated on wide-area network topologies and traffic characteristics in [3]. [25] further improved this result by describing how to compute an $\mathcal{O}(\log N)$-competitive oblivious routing scheme using a simple, fast algorithm based on multiplicative weights and FRT's randomized approximation of general metric spaces by tree metrics [10]. This algorithm was demonstrated on wide-area traffic in [19]. Additionally, [14] found that routing schemes oblivious to both traffic and the cost functions associated with each edge could achieve a competitive ratio of polylog(N). These works differ from ours

in two ways. First, these works assume a fixed network topology which must be served by an oblivious routing algorithm, while our works co-design both the network topology and the routing algorithm together. In addition, these works evaluate congestion, which is similar but not analogous to our definition of throughput.

One prior work which evaluates throughput of oblivious routing schemes is [15], which proves a lower bound of $\Omega(\frac{\log N}{\log \log N})$ on the competitive ratio of throughput in general networks. However, their definition of throughput differs from ours as well, as it sums the flow rate of all sender-receiver pairs in the entire network. The competitive ratio of average latency in oblivious routing over general networks has been analyzed by [18]; however they assign resistance values to each edge to compute latency, and only provide a routing scheme achieving the $\mathcal{O}(\log N)$-competitive ratio when routing to a single target. Additionally, [12] analyzes the competitive ratio of hop-constrained oblivious routing, which seeks to bound maximum latency in some way, though again, they look at general networks.

Valiant load balancing Our ORN designs all make use of Valiant load balancing (VLB). Both VLB and oblivious routing in general were introduced by Valiant and Brebner in [28] and [27]. These works examined VLB over a direct-connect hypercube topology, and showed it to have optimal latency under queuing in this context. They focus on unit demand permutation traffic and do not consider throughput. While our context differs substantially, both in the use of a reconfigurable fabric instead of a direct connect topology, as well as our evaluation of different performance metrics, our work shows that VLB remains an optimal strategy in the ORN context.

Load-Balanced Switches: There are significant parallels between ORNs and the load-balanced switch architecture proposed by Chang [1], which uses static schedules and sends traffic obliviously via intermediate nodes. However, this architecture differs from ORNs both in its use of specialized intermediate nodes (rather than using end-hosts as intermediate nodes) and in its focus on monolithic switches.

Circuit-Switched Datacenter Network Architectures: Multiple proposals exist for hybrid datacenter networks, including c-Through [30] and Traffic Matrix Scheduling [23]. In these networks, a reconfigurable circuit-switched fabric is combined with a packet-switched network in order to serve both throughput- and latency-sensitive traffic, respectively. With recent advances in circuit switching technology, it is worth reconsidering if a separate packet-switched network is still necessary for supporting low-latency traffic.

Oblivious Circuit-Switched Networks: While [2] first described the ORN design paradigm, there previously existed several examples of designs that fit the paradigm. Rotornet [22] and Sirius [6] use optical circuit switches to build a datacenter-wide reconfigurable network fabric, while Shoal [26] uses electronic circuit switches in a disaggregated rack environment. These works demonstrate multiple ways in which the ORN paradigm can be implemented in practice. However, these designs all use similar schedules that achieve a single tradeoff point between throughput and latency that maximizes throughput. EBS can be seen as a generalization of these schedules which achieves additional tradeoff points with substantially improved latency scaling.

Opera [21] evolves on the ORN paradigm by combining greatly lengthened timeslots and high node degrees with non-oblivious routing. This enables the network to have an expander graph topology at every point in time, thus allowing low-latency traffic to be sent via multiple hops over a single network configuration. Throughput-sensitive traffic is instead held until the schedule advances to a topology with a direct link to the destination, when it can be sent via a single hop. This design makes strong assumptions about the workload, including that bandwidth-sensitive traffic demand is near all-to-all, limiting its flexibility.

2 Definitions

In this section we present definitions formalizing an Oblivious Reconfigurable Network (ORN). We use similar definitions as previously in [2], restated here for convenience.

We assume a network of N nodes which communicate in discrete synchronous timeslots. The nodes are joined by a communication fabric that allows for an arbitrary pattern of one-directional communication links in each timeslot, subject to a degree constraint d. We assume $d = 1$ throughout the rest of this paper. [2] discusses why this special case reduces to the general case for any constant d.

In practice, data is partitioned into fixed-size indivisible units called *frames* or *packets*. In this work we treat data as continuously divisible, and allow for sending fractional quantities of flow along multiple paths when routing from source to destination. This is a standard abstraction in theoretical work on oblivious routing, justified in the following way. One can interpret fractional flow as a probability distribution over routing paths; each discrete frame is sent along one path sampled from this distribution, and the total flow value over an edge represents the

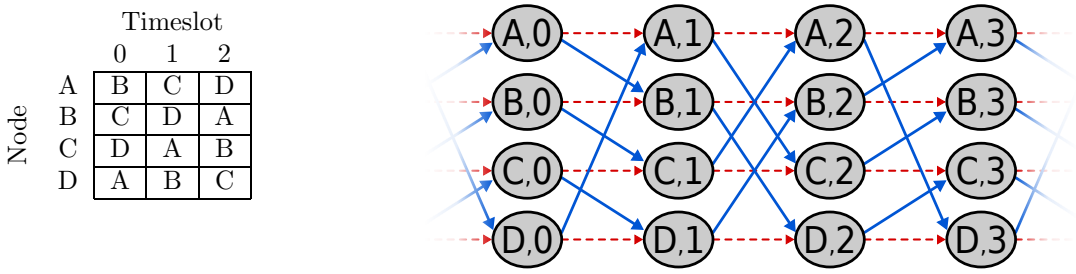

Figure 1: A connection schedule among four nodes, as well as part of its corresponding virtual topology. The full virtual topology represents a countably infinite number of timeslots.

expected flow crossing the edge.

DEFINITION 2.1. *A* connection schedule $\boldsymbol{\pi}$ *with size* N *and period length* T *is a sequence of permutations* $\pi_0, \pi_1, \ldots, \pi_{T-1}$*, each mapping* $[N]$ *to* $[N]$*. When* $\pi_k(i) = j$*, then node* i *is allowed to send one frame to node* j *during any timeslot* t *such that* $t \equiv k \pmod{T}$*.*

The virtual topology *of the connection schedule* $\boldsymbol{\pi}$ *is a graph* $G_{\boldsymbol{\pi}}$*, with vertex set* $[N] \times \mathbb{Z}$ *and edge set consisting of the union of* virtual edges E_{virt} *and* physical edges E_{phys}*. Edges* $e \in E_{virt}$ *are of the form* $(i,t) \to (i,t+1)$ *and represent the frame waiting at node* i *during the timeslot* t*. Edges* $e \in E_{phys}$ *are of the form* $(i,t) \to (\pi_t(i), t+1)$ *and represent the frame being transmitted from* i *to* $\pi_t(i)$ *at timeslot* t*.*

Let $\mathcal{P}(a,b,t)$ denote the set of all paths in $G_{\boldsymbol{\pi}}$ from node a to node b beginning at timeslot t (and ending at some timeslot, say t'). Let $\mathcal{P} = \bigcup_{a,b,t} \mathcal{P}(a,b,t)$ denote the set of all paths in $G_{\boldsymbol{\pi}}$.

DEFINITION 2.2. *A* flow *is a function* $f : \mathcal{P} \to [0, \infty)$*. For a given flow* f*, the amount of flow traversing an edge* e *is defined as:*

$$F(f,e) = \sum_{P \in \mathcal{P}} f(P) \cdot \mathbf{1}_{e \in P}$$

We say that f *is* feasible *if for every physical edge* $e \in E_{phys}$*,* $F(f,e) \leq 1$*.*

DEFINITION 2.3. *The* latency $L(P)$ *of a path* P *in* $G_{\boldsymbol{\pi}}$ *is the number of timeslots elapsed between when the path begins and when it ends. Note this is equal to the number of edges it contains (both virtual and physical). For a nonzero flow* f*, the* maximum latency *is the maximum over all paths in the flow*

$$L_{max}(f) = \max_{P \in \mathcal{P}} \{L(P) : f(P) > 0\}$$

We remark that our definitions of latency and of the virtual topology $G_{\boldsymbol{\pi}}$ incorporate the idealized assumption of zero propagation delay. In other words, we assume that a frame sent in one timeslot is received by the beginning of the following timeslot, thus the number of edges of a path in the virtual topology accurately reflects the number of timeslots between when the frame originates and when it reaches its destination.

DEFINITION 2.4. *An* oblivious routing scheme R *with period* T *is a function that associates to every* $(a,b,t) \in [N] \times [N] \times \mathbb{Z}$ *a flow* $R_{a,b,t}$ *such that:*

1. $\forall P \notin \mathcal{P}(a,b,t) \;\; R_{a,b,t}(P) = 0$.
2. $\sum_P R_{a,b,t}(P) = 1$.
3. $R_{a,b,t+T}$ *is equivalent to* $R_{a,b,t}$ *(except all paths have been transposed by* T *timeslots).*

DEFINITION 2.5. *A* demand matrix *is an $N \times N$ matrix which associates to each ordered pair (a, b) an amount of flow to be sent from a to b. A* demand function *D is a function that associates to every $t \in \mathbb{Z}$ a demand matrix $D(t)$ representing the amount of flow $D(t, a, b)$ to originate between each source-destination pair (a, b) at timeslot t.*

The throughput requested by demand function *D is the maximum, over all t, over the maximum row and column sums of $D(t)$.*

DEFINITION 2.6. *An oblivious routing scheme is said to* guarantee throughput *r if the* induced flow *defined by*

$$f(R, D) = \sum_{(a,b,t) \in [N] \times [N] \times \mathbb{Z}} D(t, a, b) R_{a,b,t}.$$

is feasible whenever the demand function D requests throughput at most r.

One can interpret Definition 2.6 in the following way; the network is able to simulate a "big switch" on N input and output ports with line rate r. As long as the amount of flow originating at any node a or destined for any node b does not exceed r at each timeslot, the network is able to route all data without violating capacity constraints.

3 Summary of Previous Results

3.1 Lower Bound [2] proves the following lower bound for the latency of an ORN design based on its guaranteed throughput.

THEOREM 3.1. *Given an ORN design $\mathcal{R}$ which guarantees throughput r, the maximum latency suffered by any routing path P with $R_{a,b,t}(P) > 0$ over all a, b, t is bounded by the following equation*

$$L_{max} \geq \Omega\left(L^*(r, N)\right) \tag{3.1}$$

where $h = \lfloor \frac{1}{2r} \rfloor$, $\varepsilon = h + 1 - \frac{1}{2r}$, and $L^*(r, N) = h\left((\varepsilon N)^{1/h} + N^{1/(h+1)}\right)$

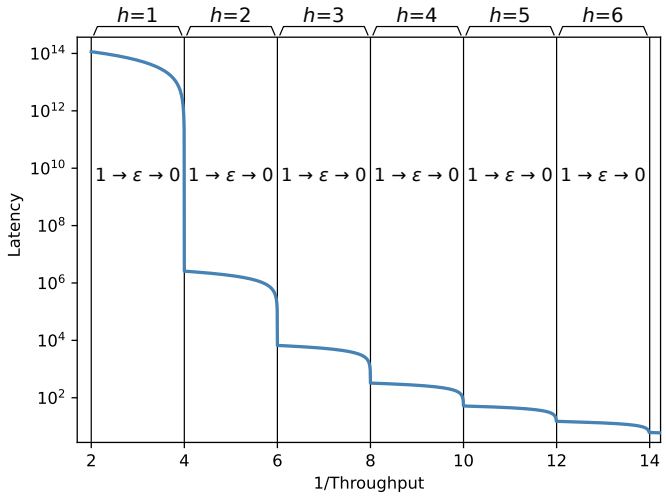

Figure 2: A plot of the lower bound for the latency of an ORN containing 10^9 nodes that can guarantee a given throughput. The plot also indicates how h and ε vary as the target throughput guarantee changes.

3.2 Valiant Load Balancing Both of the ORN designs that we present below rely on a technique called Valiant Load Balancing (VLB) [29]. VLB operates in two stages: first, traffic is routed from the source to a random intermediate node in the network. Then, traffic is routed from the intermediate node to its final destination. Here, we refer to these two components of the overall path as *semi-paths*. This two-stage design

	0	1	2	3
AA	BA	CA	AB	AC
BA	CA	AA	BB	BC
CA	AA	BA	CB	CC
·	·	·	·	·
·	·	·	·	·
BC	CC	AC	BA	BB
CC	AC	BC	CA	CB

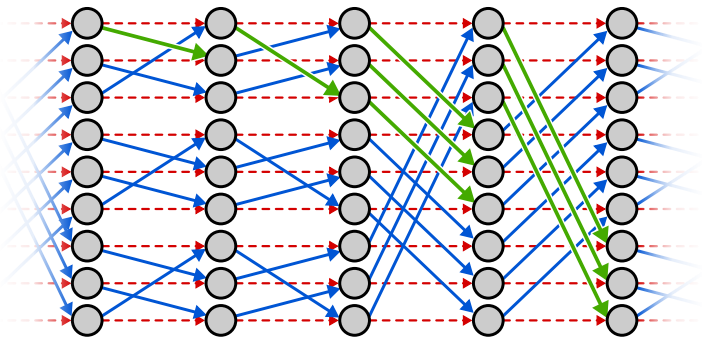

Figure 3: Connection schedule for 9 nodes in $\ell = 2$ EBS, as well as part of the corresponding virtual topology. Physical edges used on semi-paths from ((AA),0) to other nodes are highlighted in green. This schedule can be seen as a generalization of the one presented in Figure 1.

ensures the traffic is uniformly distributed throughout the network regardless of demand. The following lemma follows directly from this design.

Let D_{unif} denote the $N \times N$ demand matrix with every entry equal to $\frac{1}{N}$. If $\mathcal{R}$ is a routing scheme such that rD_{unif} is feasible and the maximum latency is L, then there exists an oblivious routing scheme $VLB(\mathcal{R})$ that guarantees throughput $r/2$ and has maximum latency $2L$.

3.3 Elementary Basis Scheme As can be seen in Figure 2, the lower bound curve is composed of curved stairsteps, with one stairstep for each value of h. The curve is nearly horizontal when ε is close to 1, but becomes much more vertical when ε is close to 0. In order to achieve a tight upper bound across both of these regions, [2] constructs two families of ORN designs which we summarize here. We begin with the Elementary Basis Scheme, or EBS, which achieves a tight bound on the horizontal regions where ε is close to 1.

3.3.1 Connection Schedule The EBS connection schedule has each node participate in a series of sub-schedules called round robins. A round robin for a set of n nodes S is a schedule of $n-1$ timeslots in which each element of S has the chance to send directly to each other element exactly once. A tuning parameter h, called the *order* of the connection schedule, governs the number of round robins in which nodes participate when using EBS.

To optimally guarantee throughput r, EBS requires $N = n^h$ nodes, for integer n and $h = \lfloor \frac{1}{2r} \rfloor$. Treat the node set $[N]$ as elements of the group $(\mathbb{Z}/(n))^h$ by assigning each node $a \in [N]$ a unique set of coordinates $(a_0, a_1, ..., a_{h-1}) \in (\mathbb{Z}/(n))^h$. Nodes participate in h round robins, each containing the n nodes that match in all but one of the h coordinates. We refer to these round robins as *phases* of the schedule. One full iteration of the EBS schedule consists of h phases, one for each coordinate of the nodes. Since each phase is composed of $n-1$ timeslots, the period of the schedule is $T = h(n-1)$.

To fully define the EBS connection schedule, we identify each permutation π_k of the connection schedule using a phase number p, $0 \le p < h$, and a scale factor s, $1 \le s < n$, such that $k = (n-1)p + s - 1$. Let $\mathbf{e}_p$ denote the standard basis vector whose p^{th} coordinate is 1 and all other coordinates are 0. The connection schedule is then $\pi_k(\boldsymbol{i}) = \boldsymbol{i} + s\mathbf{e}_p = \boldsymbol{j}$. Since $\mathbf{e}$ is the standard basis, $j_x = i_x$ for $x \neq p$, and $j_p = i_p + s \pmod n$.

3.3.2 Oblivious Routing Scheme As discussed in section Section 3.2, the EBS oblivious routing scheme is based around Valiant load balancing, with each path composed of two semi-paths.

To create a semi-path between node a and node b starting at timeslot t, the following greedy algorithm is used starting at node (a, t) in the virtual topology: if the outgoing physical edge leads to a node with a decreased Hamming distance to b (i.e. it matches b in the modified coordinate), traverse the physical edge. Otherwise, traverse the virtual edge. This algorithm terminates when it reaches node b. Note that because there are h coordinates, the largest Hamming distance possible is h, and the longest semi-paths use h physical links.

For later convenience of analysis, here we use a slightly modified version of EBS compared to [2], in which

we hold frames until the next phase boundary before sending them. Thus, to construct a full path from node a to node b starting at timeslot t, the first step is to traverse up to $n-2$ virtual edges until a phase boundary is reached at timeslot t'. Then, select an intermediate node c in the system uniformly at random and traverse the semi-path from node a to node c starting at timeslot t'. Let t'' be the timeslot at which we reach c. If $t'' < t' + T$, traverse virtual edges until node $(c, t' + T)$ is reached. Finally, traverse the semi-path from c to b starting at $t' + T$.

The EBS oblivious routing scheme is formed as follows: for $R_{a,b,t}$, for all intermediate nodes c, construct the path from (a, t) to b via c as described above, and assign it the value $\frac{1}{N}$. Assign all other paths the value 0. Because there are N possible intermediate nodes, each of which is used to define one path from (a, t) to b, this routing scheme defines one unit of flow.

3.3.3 EBS Guarantees Due to our modification of the oblivious routing scheme, the maximum latency achieved is increased by up to $n-2$ timeslots compared to [2], resulting in the following bound:

THEOREM 3.2. *The EBS design with level h on N nodes for any $N = n^h$ for positive integer n guarantees throughput $\frac{1}{2h}$ and achieves maximum latency $L \leq (2h+1)(n-1) - 1$.*

The following bound from [2] is unchanged.

THEOREM 3.3. *For $0 < r \leq \frac{1}{2}$ let $h = \lfloor \frac{1}{2r} \rfloor$ and $\varepsilon = h + 1 - \frac{1}{2r}$. The EBS design of order h attains maximum latency at most $\mathcal{O}(L^*(r, N))$, except when*

$$\varepsilon \leq 2\sqrt{\frac{2h}{\pi}} \left(\frac{2e}{C}\right)^h$$

for constant $C > 2e$.

COROLLARY 3.1. *When $\varepsilon \geq \frac{1}{2} \cdot \frac{1}{16(h+1)(1+\frac{1}{2h})^2}$, the EBS design of order h attains maximum latency of at most $\mathcal{O}(L^*(r, N))$.*

3.4 Vandermonde Basis Scheme Because EBS always uses semi-paths that use at most h physical hops, its maximum latency only depends on h and N. This is tight with the lower bound in the nearly horizontal sections where ε is away from 0. When ε is small, sending all flow along paths using at most $2h$ physical hops leaves a large amount of unused capacity in the network, making the throughput guarantee feasible even if paths use more than $2h$ physical hops on average. This increased flexibility afforded by the extra physical hops means that in theory, the same number of destinations can be reached in fewer timeslots, thus reducing maximum latency.

The Vandermonde Basis Scheme, or VBS, is designed to take advantage of this idea by allowing both h-hop and $(h+1)$-hop semi-paths. This is done by modifying order $h+1$ EBS so that more destinations are reachable using h-hops over multiple "periods", by constantly changing the bases underlying the connection schedule using Vandermonde vectors.

In VBS, every source-destination pair can be routed using $(h+1)$-hop semi-paths in the upcoming basis. However, by considering phases from future bases, it is possible to route some fraction of pairs using just h hops. As more future phases get considered, more semi-paths can be routed using h hops, and thus the higher the throughput guarantee. Considering fewer phases lowers throughput, but in return also lowers the maximum latency. This flexibility allows VBS to be tight with the lower bound over the entire small-ε region.

3.4.1 Connection Schedule As discussed, VBS uses a modified version of the EBS schedule for order $h+1$. Instead of elementary basis vectors, the phases are defined using Vandermonde vectors (to be defined in the next paragraph). Additionally, instead of using a single basis, the VBS connection schedule is formed from a longer sequence of phases, with any set of $h+1$ adjacent phases corresponding to a basis.

To optimally guarantee throughput r, VBS requires $N = q^{h+1}$ nodes, for some prime number q. Similarly to EBS, each node a is assigned a unique set of $h+1$ coordinates $(a_0, a_1, ..., a_h)$, each ranging from 0 to $q-1$. This maps each node to a unique element of $\mathbb{F}_q^{h+1}$. We identify each permutation π_k of the connection schedule using a scale factor s, $1 \leq s < q$, and a phase number p, $0 \leq p < q$, such that $k = (q-1)p + s - 1$. Each phase p is formed using the Vandermonde vector $\boldsymbol{v}(p) = (1, p, p^2, ..., p^h)$. This produces the connection schedule $\pi_k(\boldsymbol{i}) = \boldsymbol{i} + s\boldsymbol{v}(p)$.

3.4.2 Oblivious Routing Scheme Similarly to EBS, VBS's oblivious routing scheme is based around Valiant Load Balancing, with each path composed of one semi-path to a randomly chosen intermediate node, followed by another semi-path to the destination node. Semi-paths in VBS are only defined starting at phase boundaries, so the first step of a VBS path is to traverse up to $q-2$ virtual edges until a phase boundary is reached. Semi-paths are then defined for a given (p, a, b) triple, where p is the phase number in which the path begins. Following the initial virtual edges to reach a phase boundary, we concatenate the semi-path from the source to the intermediate node, followed by the semi-path from the intermediate node to the destination.

VBS uses two routing strategies for semi-paths: *single-basis* (SB) paths, which use $h+1$ physical hops over the next $h+1$ phases, and *hop-efficient* (HE) paths, which use h physical hops over the following Q phases, where Q is the lowest integer such that $\binom{Q}{h} \geq 4q\varepsilon$. Note that the dependence on ε means that Q is optimized depending on the throughput guarantee. If a given (p, a, b) triple has a HE path available, it is routed along that path; otherwise it is routed along the SB path (which always exists). We describe both path types below.

Single-basis paths The single-basis path, or SB path, for a given (p, a, b) is formed as follows: First, we define the displacement vector $\boldsymbol{d} = b - a$, as well as the basis $Y = (\boldsymbol{v}(p), \boldsymbol{v}(p+1), ..., \boldsymbol{v}(p+h)$. Note that the vectors in the basis Y are those used to form the $h+1$ phases beginning with phase p. Then, we find $\boldsymbol{s} = Y^{-1}\boldsymbol{d}$. Over the next $h+1$ phases, for every timeslot $t' \equiv (p', s')$, if $s' = \boldsymbol{s}_{p'}$, the physical edge is traversed. Otherwise, the virtual edge is traversed. This corresponds to traversing $\boldsymbol{d}$ through its decomposition in Y, beginning at a and ending at b.

Although this algorithm completes within $h+1$ phases, following this virtual edges are traversed for a further Q phases to ensure that both SB and HE paths take $h+1+Q$ phases to complete. While it is possible for an SB path to have fewer than $h+1$ hops, this becomes increasingly rare as N grows without bound.

Hop-efficient paths A hop-efficient path, or HE path, is formed as follows. First, for $h+1$ phases, only virtual edges are traversed. This ensures that SB and HE paths beginning in the same phase p use disjoint sets of vectors, which simplifies the analysis of VBS. Following this initial buffer period, h phases are selected out of the next Q phases, and one physical hop is taken in each selected phase. During all other timeslots within the Q phases, virtual hops are taken.

3.4.3 VBS Guarantees [2] proves the following guarantee about the VBS design.

THEOREM 3.4. *For a guaranteed throughput value r, let $h = \lfloor \frac{1}{2r} \rfloor$ and $\varepsilon = h + 1 - \frac{1}{2r}$. If $\varepsilon \leq \frac{1}{16(h+1)(1+\frac{1}{2h})^2}$, then for any $N = q^{h+1}$ where q is prime, there exists a VBS design on N nodes which guarantees throughput r and achieves maximum latency at most $\mathcal{O}(L^*(r, N))$.*

Note that the ranges in which EBS and VBS are tight with the lower bound together cover the entire range $r \in (0, 1/2]$.

4 Dummy Node Designs

In the following section, we describe the main theorem of this paper.

THEOREM 4.1. *Given a guaranteed throughput value $r \in (0, 1/2]$ for which $\frac{1}{2r} \notin \mathbb{Z}$, there exists an integer N_0 such that, for all integers $N \geq N_0$, there exists an ORN design on N nodes which guarantees throughput r and achieves maximum latency within $\mathcal{O}(L^*(r, N))$.*

To prove this theorem, we describe schemes for modifying both EBS and VBS to work with an arbitrary number of nodes, and then provide tight guarantees on the maximum latency of the modified designs.

4.1 EBS Dummy Node Design

4.1.1 Design description Let h, ε be defined given r as before: $h = \lfloor \frac{1}{2r} \rfloor$ and $\varepsilon = h + 1 - \frac{1}{2r}$. Let N be the number of nodes we would like to run Dummy EBS on. That is, N is an integer that is not an integer h-power.

Let $M \geq N$ be the smallest such M for which there exists an integer m and $M = m^h$, and consider the h-level EBS design on M nodes. By Theorem 3.2, this design can guarantee throughput $\frac{1}{2h}$ within max latency $(2h+1)(m-1)-1$ in a network of exactly M nodes.

We will designate a set of potential dummy nodes $\mathcal{D}$, and define the EBS dummy node design at level h on N nodes in the following way: choose a subset $D \subseteq \mathcal{D}$ such that $N + |D| = M$. All flow paths that do not travel through nodes in D remain untouched and continue to route flow as specified by EBS on M nodes. Flow paths that travel through D no longer send flow.

Note that this design will be able to guarantee a slightly smaller throughput value than previously, and some node pairs and starting timeslots may have more flow attributed to them by the routing scheme than others. To fix this second point, one can normalize the amount of flow attributed to each pair by the minimum over all pairs.

To prove what throughput value this design guarantees, we will show that if we eliminate all flow that would have been routed through nodes in $\mathcal{D}$ using EBS, no pair loses more than a δ fraction of flow for sufficiently small δ.

Designate the following set $\mathcal{D}$ as the potential dummy node set,

$$\mathcal{D} = \left\{ \left(i_0, i_1, ..., i_{h-2}, \ell + \sum_{j=0}^{h-2} i_j \right) : i_0, ..., i_{h-2} \in [m] \text{ and } \ell \in \{0, ..., h-1\} \right\}$$

Before we check that eliminating all flow on a, b, t triples that would have routed through $\mathcal{D}$ still guarantees a high enough throughput rate, we should first check to ensure that $|\mathcal{D}|$ is large enough. That is, it must be at least $M - N$. Note that N cannot be smaller than $(m-1)^h$, as otherwise there would be a smaller integer h-power M' with $M > M' \geq N$. Therefore, it is enough to show that $|\mathcal{D}|$ is at least $m^h - (m-1)^h$.

Using the fundamental theorem of calculus, one can see that $\int_{m-1}^{m} hx^{h-1} = m^h - (m-1)^h$. The integrand hx^{h-1} will always be no more than hm^{h-1}, since x takes values in the interval $[m-1, m]$. Therefore, $\int_{m-1}^{m} hx^{h-1} \leq hm^{h-1} = |\mathcal{D}|$.

4.1.2 Tightness Guarantees

LEMMA 4.1. *The EBS dummy node design on N nodes can guarantee throughput $\frac{1}{2h}\left(1 - \frac{2h^2}{m}\right)$ and maximum latency $(2h+1)(m-1)-1$ for $h = \left\lfloor \frac{1}{2r} \right\rfloor$ and M equal to the smallest integer h-power larger than N, for sufficiently large N.*

Proof. We can bound the guaranteed throughput rate by bounding the amount of flow between any worst case triplet that goes through the set $\mathcal{D}$. Since EBS uses Valiant Load Balancing, we can bound the fraction of flow that gets eliminated by dummy nodes for any triple a, b, t by using the total fraction of semi-paths that get eliminated when routing semi-paths of a worst-case node a starting at timeslot t to all intermediate nodes v_{int}.

Consider node $a = (a_0, ..., a_{h-1})$ and dummy node $d = (d_0, ..., d_{h-1}) \in \mathcal{D}$, and suppose t occurs in phase p. We would like to count the number of intermediate nodes v_{int} for which the semi-path from a to v_{int} starting at timeslot t goes through d.

Suppose d and a match in all but one coordinate. Then in order for d to be on the semi-path from a to v_{int} starting at timeslot t, it must be the case that d and a are mismatched in coordinate $p+1$, and that v_{int} matched d in the $(p+1)$-th coordinate. There are m^{h-1} such v_{int}, leaving a $\frac{1}{m}$ fraction with the property. If we choose all our dummy nodes from $\mathcal{D}$, then there are at most h dummy nodes that match a in all but the $(p+1)$-th coordinate.

Consider more generally if a d and a match in all but k consecutive coordinates, starting at phase $p+1$. If we choose all our dummy nodes from $\mathcal{D}$, then there are at most hm^{k-1} such dummy nodes. And given such a dummy node, the fraction of v_{int} that get eliminated is at most $\frac{1}{m^k}$.

Thus, if we pick all our dummy nodes from $\mathcal{D}$, we can bound the fraction of flow δ_{EBS} that gets eliminated at each node. Note that we gain a factor of 2 due to VLB, by applying this argument for both semi-paths $a \to v_{int}$ and $v_{int} \to b$.

$$\begin{aligned} \delta_{EBS} &\leq 2 \sum_{k=1}^{h} \frac{1}{m^k} hm^{k-1} \\ &\leq 2h \sum_{k=1}^{h} \frac{1}{m} = \frac{2h^2}{m} \end{aligned}$$

So, we can guarantee throughput

$$\begin{aligned} r &\geq \frac{1}{2h}(1-\delta_{EBS}) \\ &\geq \frac{1}{2h}\left(1-\frac{2h^2}{m}\right) \end{aligned}$$

□

Since the above goes toward $\frac{1}{2h}$ as $M \longrightarrow \infty$, if we wanted to guarantee a throughput value r for which $\varepsilon \neq 1$, then the EBS dummy node design on any number of nodes N can guarantee throughput r for sufficiently large N.

LEMMA 4.2. *Let $r \in (0, \frac{1}{2})$ be a throughput value and ε, h defined as usual; $h = \lfloor \frac{1}{2r} \rfloor$ and $\varepsilon = h + 1 - \frac{1}{2r}$. If $\varepsilon \in \left[\frac{1}{2} \cdot \frac{1}{16(h+1)(1+\frac{1}{2h})^2}, 1\right)$, then the EBS dummy node design on N nodes achieves maximum latency $\mathcal{O}\left(L^*(r,N)\right)$ for sufficiently large N.*

Proof. Let $r \in (0, \frac{1}{2})$ be given as above and let M be the smallest integer h-power larger than or equal to N.

Since $\varepsilon < 1$, then $r < \frac{1}{2h}$, meaning that for large enough N (and therefore M), $r \geq \frac{1}{2h}\left(1 - \frac{2h^2}{m}\right)$. Therefore for large enough N, the EBS dummy node scheme can guarantee throughput r and achieve maximum latency $(2h+1)(m-1)-1$.

To show that $(2h+1)(m-1)-1 \leq \mathcal{O}\left(L^*(r,N)\right)$, we use the fact that $N \geq (m-1)^h$. Therefore $2hN^{1/h}$ is at least $2hm\left(1-\frac{h}{m}\right)^{1/h}$ which is no more than a constant factor less than $(2h+1)(m-1)-1$ for sufficiently large N.

Finally, when $\varepsilon \geq \frac{1}{2} \cdot \frac{1}{16(h+1)(1+\frac{1}{2h})^2}$, by Corollary 3.1 the maximum latency $2hN^{1/h} \leq \mathcal{O}\left(L^*(r,N)\right)$, completing our proof. □

4.2 VBS Dummy Node Design

4.2.1 Design description The Vandermonde Basis Scheme, or VBS, is a bit trickier to work with. It requires the use of far more dummy nodes due to the primality requirement of $N^{\frac{1}{h+1}}$, and it requires some additional tweaking of the schedule to ensure that dummy nodes stay sufficiently well distributed throughout the network, due to the changing Vandermonde vector phases. Additionally, it also requires individual analysis of both single-basis and hop-efficient semi-path types.

Let h, ε be defined given r as before: $h = \lfloor \frac{1}{2r} \rfloor$ and $\varepsilon = h + 1 - \frac{1}{2r}$. Let N be an integer that is not a prime $(h+1)$-power. That is, there is no prime number q for which $N = q^{h+1}$. Let $M > N$ be the smallest such M for which there exists a prime q and $M = q^{h+1}$. First, we define the set of possible dummy nodes $\mathcal{D}$.

$$\mathcal{D} = \left\{ \left(i_0, i_1, ..., i_{h-1}, \ell + \sum_{j=0}^{h-1} i_j \right) : i_0, ..., i_{h-1} \in [q] \text{ and } \ell \in \{0, ..., (h+1)q^{0.525} - 1\} \right\}$$

We confirm that $\mathcal{D}$ is large enough by citing the following theorem about prime gaps.

THEOREM 4.2. *[Baker, Harman, Pintz 2001] [5] For all $x > x_0$, the interval $[x - x^\theta, x]$ contains at least one prime number for $\theta = 0.525$.*

Thus, it is sufficient to show that $|\mathcal{D}|$ is at least $M - N \leq q^{h+1} - (q - q^{0.525})^{h+1}$. Using the fundamental theorem of calculus, it is clear that

$$\begin{aligned} q^{h+1} - (q - q^{0.525})^{h+1} &= \int_{q-q^{0.525}}^{q} (h+1)x^h dx \\ &\leq (h+1)q^h(q^{0.525}) = |\mathcal{D}| \end{aligned}$$

Restricted Vandermonde Vectors In order to ensure that dummy nodes are well distributed throughout the network, we restrict the Vandermonde vectors of the VBS dummy node design so that lines parallel to these vectors intersect $\mathcal{D}$ in a limited number of points. In particular, we would like for each such line to contain no more than $(h+1)q^{0.525}$ elements of $\mathcal{D}$. In order to prove sufficiently many of these Vandermonde vectors exist, let us examine $\mathcal{D}$ more closely.

We can represent $\mathcal{D}$ as the following union

$$\begin{aligned}\mathcal{D} &= \bigcup_{k=1}^{(h+1)q^{0.525}} \mathcal{D}_k \\ &= \bigcup_{k=1}^{(h+1)q^{0.525}} \left\{\vec{i} : i_0 + i_1 + ... + i_{h-1} + i_h = k\right\}\end{aligned}$$

We will call a Vandermonde vector *bad* if it belongs to the set $\mathcal{D}_0 = \left\{\vec{i} : i_0 + i_1 + ... + i_{h-1} + i_h = 0\right\}$, and otherwise we call the Vandermonde vector *good*. We will restrict our VBS connection schedule to use only good Vandermonde vectors. Observe that the vector $\vec{v} = (1, \alpha, \alpha^2, \ldots, \alpha^h)$ is bad if and only if the equation $1 + \alpha + \alpha^2 + \ldots + \alpha^{h-1} + \alpha^h = 0$ is satisfied. This is a polynomial equation of degree h in α, so there are at most h bad Vandermonde vectors and at least $q - h$ good ones. Since the VBS connection schedule requires Q Vandermonde vectors where Q is the least integer satisfying $\binom{Q}{h} \geq 4\varepsilon q$, there are sufficiently many distinct good vectors so long as $q > h/(1 - 4\varepsilon)$.

If $\vec{v}$ is a good Vandermonde vector and L is any line parallel to $\vec{v}$, then the intersection $L \cap \mathcal{D}$ contains at most $(h+1)q^{0.525}$ elements. To see why, represent L as the set $\{\vec{a} + r \cdot \vec{v} \mid r \in \mathbb{F}_q\}$ for some $\vec{a} \in \mathbb{F}_q^{h+1}$ and recall that $\mathcal{D}$ is partitioned into the sets

$$\mathcal{D}_k = \left\{\vec{i} \mid i_0 + \cdots + i_h = k\right\}$$

with k ranging from 1 to $(h+1)q^{0.525}$. It suffices for us to prove that $L \cap \mathcal{D}_k$ has at most one element, for each k. The relation $\vec{a} + r \cdot \vec{v} \in \mathcal{D}_k$ holds if and only if $\sum_{j=0}^{h}(a_j + r\alpha^j) = k$. Rewrite this equation as $r\left(\sum_{j=0}^{h} \alpha^j\right) = k - \sum_{j=0}^{h} a_j$, and observe that $\sum_{j=0}^{h} \alpha^j \neq 0$ because $\vec{v} = (1, \alpha, \ldots, \alpha^h)$ is a good Vandermonde vector. Hence there is a unique r satisfying the equation, and consequently $L \cap \mathcal{D}_k$ has exactly one element.

We define the VBS dummy node design at level $(h+1)$ on N nodes in the following way: let M be the smallest prime $(h+1)$-power greater than or equal to N. Choose a subset $D \subseteq \mathcal{D}$ such that $N + |D| = M$. All flow paths that do not travel through nodes in D remain untouched and continue to route flow as specified by EBS on M nodes. Flow paths that travel through D no longer send flow.

Note that in this definition, like in the EBS dummy node design, some node-timeslot triples a, b, t may have more flow attributed to them by the routing scheme than others. To fix this, one can again normalize the amount of flow attributed to each pair by the minimum over all pairs.

4.2.2 Tightness Guarantees

LEMMA 4.3. *Let r be a throughput rate achievable by the original VBS design. Then the VBS dummy node design on N nodes can guarantee throughput at least $r\,(1 - \delta_{VBS})$ for $\delta_{VBS} = \frac{2(h+1)^2}{q^{0.475}}$ and M equal to q^{h+1}, the smallest prime $(h+1)$-power larger than or equal to N.*

Proof. Consider a node $a = (a_0, ..., a_h)$ and starting timeslot t. Like in the proof of Lemma 4.1 we will bound the total fraction of v_{int} for which the semi-path from a to v_{int} starting at timeslot t goes through some potential dummy node, $d \in \mathcal{D}$. We will do this separately for single basis and hop efficient semi-path types.

Bounding δ for single basis paths Consider the "1st worst case", when starting node a and dummy node d are exactly 1 hop apart, using one of the q hops that occur in the phase beginning next after timeslot t. Then if d is on the semi-path from (a, t) to v_{int}, it is reached on the first hop of the path. The fraction of v_{int} which may be reached on a semi-path after visiting d is $\frac{1}{q}$. The number of dummy nodes with this property with respect to a and t is at most $q^{0.525}$, by definition of the potential dummy node set $\mathcal{D}$.

More generally, we consider the "kth worst case", in which the dummy node d can be reached within the next k phases from a starting at timeslot t. In this case, the fraction of flow eliminated by d is at most $\frac{1}{q^k}$ and the amount of dummy nodes with this property is at most $(h+1)q^{0.525}q^{k-1}$.

Putting all cases together, the maximum fraction of flow that gets eliminated from single-basis semi-paths is at most

$$\begin{aligned}\delta_{SB} &\leq 2\sum_{k=1}^{h+1}\frac{1}{q^k}(h+1)q^{0.525}q^{k-1}\\ &= \frac{2(h+1)^2}{q^{0.475}} = \delta_{VBS}\end{aligned}$$

Bounding δ for hop-efficient paths Consider a node $a=(a_0,...,a_h)$ and timeslot t. The total number of hop-efficient semi-paths leaving (a,t) is $\binom{Q}{h}q^h$. Meanwhile, we can bound the number of semi-paths leaving (a,t) that pass through $\mathcal{D}$ in the following way: again choose the h out of Q phases that the semi-path will take hops in. If the semi-path passes through $\mathcal{D}$, then in one of the h chosen phases, the edge chosen leads to a dummy node $d\in\mathcal{D}$. Note that in each phase, there are at most $(h+1)q^{0.525}$ dummy nodes reachable by one hop. So, we can bound the number of semi-paths which leave (a,t) and pass through $\mathcal{D}$ by $\binom{Q}{h}h(h+1)q^{0.525}$. (Note that this estimation overcounts paths which pass through $\mathcal{D}$ multiple times.) So, including the factor 2 due to using 2 semi-paths per routing path, the fraction of hop-efficent path flow that gets eliminated is at most

$$\begin{aligned}\delta_{HE} &\leq 2\cdot\frac{h(h+1)q^{0.525}}{q^h}\\ &\leq 2\cdot\frac{(h+1)^2}{q^{0.475}}\cdot\frac{1}{q^{h-1}}\\ &\leq \frac{2(h+1)^2}{q^{0.475}} = \delta_{VBS}\end{aligned}$$

□

LEMMA 4.4. *Let $r\in(0,\frac{1}{2})$ be a throughput value and ε, h defined as usual; $h=\lfloor\frac{1}{2r}\rfloor$ and $\varepsilon = h+1-\frac{1}{2r}$. If $\varepsilon\leq\frac{1}{2}\cdot\frac{1}{16(h+1)(1+\frac{1}{2h})^2}$, then the VBS dummy node design can guarantee throughput r and achieve maximum latency $\mathcal{O}\left(L^*(r,N)\right)$ for all sufficiently large N.*

Proof. Suppose we would like to guarantee throughput r. Then we need to find an r' such that $r'(1-\delta_{VBS})=r$, and show that (1) VBS works on throughput r' for large enough M, and (2) the maximum latency achieved by the VBS design on M nodes guaranteeing throughput r' is not significantly higher than $L^*(r,N)$. That is, we'd like for $L^*(r',M)\leq\mathcal{O}(L^*(r,N))$. First, we'll examine the relationship between r' and r, and ε' and ε.

$$\begin{aligned}r'\left(1-\frac{2(h+1)^2}{q^{0.475}}\right) &= r\\ r' &= r*\frac{q^{0.475}}{q^{0.475}-2(h+1)^2}\end{aligned}$$

Recall that $h=\lfloor\frac{1}{2r}\rfloor$. r' is set to be slightly larger than r, but note that it is small enough for $h'=h$. That is, $h=\lfloor\frac{1}{2r'}\rfloor$. Since r' is larger than r, then ε' will be slightly larger than ε. We can write ε' as a function of ε, h, and M below.

$$\begin{aligned}
\varepsilon' &= h+1-\frac{1}{2r\left(\frac{q^{0.475}}{q^{0.475}-2(h+1)^2}\right)} \\
&= h+1-\frac{q^{0.475}-2(h+1)^2}{2rq^{0.475}} \\
&= h+1-\frac{1}{2r}+\frac{2(h+1)^2}{q^{0.475}} \\
&= \varepsilon+\frac{2(h+1)^2}{q^{0.475}} \\
&\leq \frac{1}{2}\cdot\frac{1}{16(h+1)(1+\frac{1}{2h})^2}+\frac{2(h+1)^2}{q^{0.475}} \\
&\leq \frac{1}{16(h+1)(1+\frac{1}{2h})^2}
\end{aligned}$$

The last two steps holds when we take M to be large enough. By Theorem 3.4, the VBS design can guarantee throughput r' when taken with large enough M.

To show that the VBS dummy design achieves maximum latency $\mathcal{O}\left(L^*(r,N)\right)$, recall that $N \geq M - \left(q-q^{0.525}\right)^{h+1} \geq M - q^h(h+1)q^{0.525}$. We compare $L^*(r',M)$ and $L^*(r,N)$ in the following way.

$$\begin{aligned}
&L^*(r',M) \leq \mathcal{O}(L^*(r,N)) \\
\Longleftarrow\ &M^{1/(h+1)}+M^{1/h}\left(\varepsilon+\frac{2(h+1)^2}{q^{0.475}}\right)^{1/h} \leq \mathcal{O}\left(\left(M\left(1-\frac{h+1}{q^{0.475}}\right)\right)^{1/(h+1)}+\varepsilon^{1/h}\left(M\left(1-\frac{h+1}{q^{0.475}}\right)\right)^{1/h}\right)
\end{aligned}$$

We will separately show that the ε-related terms are a constant apart from each other, and that the M terms are a constant apart from each other (for large enough M).

For the ε-related terms, we would like for

$$\left(\varepsilon+\frac{2(h+1)^2}{q^{0.475}}\right)^{1/h} \leq \mathcal{O}(\varepsilon^{1/h})$$

Note that

$$\begin{aligned}
&\varepsilon+\frac{2(h+1)^2}{q^{0.475}} \leq 2^h\varepsilon \\
\Longrightarrow\ &\left(\varepsilon+\frac{2(h+1)^2}{q^{0.475}}\right)^{1/h} \leq 2\varepsilon^{1/h} = \mathcal{O}(\varepsilon^{1/h})
\end{aligned}$$

As long as M is large enough, this will hold.

Now consider the M terms,

$$M^{1/h} \text{ and } \left(M\left(1-\frac{h+1}{q^{0.475}}\right)\right)^{1/h}.$$

To show these are a constant apart from one another, it is enough to show that for large enough M, $\left(1-\frac{h+1}{q^{0.475}}\right)^{1/h}$ is at least a constant. This is true, as this value tends toward 1 as M goes to infinity. The same holds true for $\left(1-\frac{h+1}{q^{0.475}}\right)^{1/(h+1)}$, found in the other relevant M term.

In total, this shows that using the VBS dummy node design, the tight throughput latency tradeoff points for any number of nodes N that is sufficiently large. ∎

Now we have the tools to prove our main theorem, restated below.

THEOREM 4.1. *Given a guaranteed throughput value $r \in (0, 1/2]$ for which $\frac{1}{2r} \notin \mathbb{Z}$, there exists an integer N_0 such that, for all integers $N \geq N_0$, there exists an ORN design on N nodes which guarantees throughput r and achieves maximum latency within $\mathcal{O}(L^*(r, N))$.*

Proof. Case 1: $\varepsilon \leq \frac{1}{2} \cdot \frac{1}{16(h+1)(1+\frac{1}{2h})^2}$, and we use the VBS dummy node design. (Lemma 4.4)

Case 2: $\varepsilon \in \left(\frac{1}{2} \cdot \frac{1}{16(h+1)(1+\frac{1}{2h})^2}, 1\right)$, and we use the EBS dummy node design. (Lemma 4.2) ∎

5 Conclusion and Open Questions

In this paper we introduced Oblivious Reconfigurable Network designs extending EBS and VBS, which together can guarantee throughput $r \in (0, 1/2)$ for any number of nodes N. We showed these designs achieve maximum latency $\mathcal{O}(L^*(r, N))$ for all sufficiently large N, whenever $\frac{1}{2r}$ is not an integer. In this section we sketch some of the most appealing opportunities for follow-up work.

5.1 Addressing when $\frac{1}{2r}$ is an Integer When $\frac{1}{2r}$ is an integer, $\varepsilon = 1$ and one cannot inflate the throughput r without increasing the integer $h = \lfloor \frac{1}{2r} \rfloor$ as well. In our analysis in Section 4, we relied on finding a suitable $r' > r$ with $h' = h$, which is infeasible in this case.

In [2], we simplified a step to get the clean $\frac{1}{2h}$ guaranteed throughput bound for EBS. However, the exact guaranteed throughput of EBS is slightly higher, and decreases toward $\frac{1}{2h}$ as the number of nodes increases; EBS on M nodes can guarantee throughput $r' = \left(\frac{m}{2h(m-1)}\right)$ for $m^h = M$. However, one can see that this slight increase will not be enough to make the EBS dummy node design feasible on throughput $\frac{1}{2r}$.

$$\begin{aligned} r'(1 - \delta_{EBS}) &\geq \frac{1}{2h} \\ \iff \frac{1}{2h}\left(\frac{m}{m-1}\right)\left(1 - \frac{2h^2}{m}\right) &\geq \frac{1}{2h} \\ \iff \frac{m - 2h^2}{m-1} &\geq 1 \end{aligned}$$

which is clearly untrue for all values M, since $h \geq 1.x$

Alternatively, one can attempt to increase $r \to r'$ and run the VBS dummy node design for throughput r' on N nodes. While this creates a feasible routing scheme, the maximum latency will no longer be $\mathcal{O}(L^*(r, N))$ according to our analysis. This is due to a combination of the differing exponents h and h', and the width of the current prime gap results.

5.2 Bridging the gap between theory and practice This work represents a big step forward in bridging the gap between theory and practice by extending optimal ORN designs to arbitrary system sizes. Eliminating the requirement for highly specific system sizes enables these designs to be usable in practice.

Still, there remain significant gaps between our model's assumptions and the reality of deployed networks. For example, our model uses the idealized assumption of fractional flow, completely ignoring congestion that could appear due to queueing of indivisible frames. Attempting to tackle this issue would introduce complexity, both in terms of the difficulty in analysis, as well as the necessity of deviating from purely oblivious routing in the event of queueing. A frame that was intended to be transmitted on a given link but finds that link blocked due to congestion must either be transmitted in a subsequent period or on a different link; in either case, the frame must deviate from its intended path. Given this, it may be appealing to explore intermediate solutions between fully centralized control and a fully oblivious model. One potential middle ground could be coupling a fully oblivious connection schedule with a partially adaptive routing scheme based on local information such as queue lengths at each node.

Our model also fails to account for propagation delay between nodes and processing delay at each node. In our model, it is possible for a frame to be sent on a physical link in one timeslot, and then be sent on another physical link in the very next timeslot. However in a physical network, nodes must receive and process frames before they can be forwarded, which adds delay. Our model could be extended to take this into account by

changing the physical links in the virtual topology as follows. Rather than connecting from timeslot t to $t+1$, they could instead connect to timeslot $t+d_p+1$, where d_p represents the total propagation and processing delay. This would likely necessitate changes to our ORN designs to maintain optimal performance.

Acknowledgements

This work was supported in part by NSF grants CSR-1704742, CHS-1955125, and DBI-2019674, as well as a Microsoft Investigator Fellowship.

References

[1] *Load balanced birkhoffvon neumann switches, part i: one-stage buffering*, Computer Communications, 25 (2002), pp. 611–622.

[2] D. Amir, T. Wilson, V. Shrivastav, H. Weatherspoon, R. Kleinberg, and R. Agarwal, *Optimal oblivious reconfigurable networks*, in Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022, New York, NY, USA, 2022, Association for Computing Machinery, pp. 1339–1352.

[3] D. L. Applegate and E. Cohen, *Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs*, in Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 25-29, 2003, Karlsruhe, Germany, A. Feldmann, M. Zitterbart, J. Crowcroft, and D. Wetherall, eds., ACM, 2003, pp. 313–324.

[4] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, *Optimal oblivious routing in polynomial time*, in Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC '03, New York, NY, USA, 2003, Association for Computing Machinery, p. 383388.

[5] R. C. Baker, G. Harman, and J. Pintz, *The difference between consecutive primes, ii*, Proceedings of the London Mathematical Society, 83 (2001), pp. 532–562.

[6] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, et al., *Sirius: A flat datacenter network with nanosecond optical switching*, in Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, 2020, pp. 782–797.

[7] M. Bienkowski, M. Korzeniowski, and H. Räcke, *A practical algorithm for constructing oblivious routing schemes*, in Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '03, New York, NY, USA, 2003, Association for Computing Machinery, p. 2433.

[8] Q. Cheng, A. Wonfor, J. L. Wei, R. V. Penty, and I. H. White, *Demonstration of the feasibility of large-port-count optical switching using a hybrid mach–zehnder interferometer–semiconductor optical amplifier switch module in a recirculating loop*, Opt. Lett., 39 (2014), pp. 5244–5247.

[9] M. Ding, A. Wonfor, Q. Cheng, R. V. Penty, and I. H. White, *Scalable, low-power-penalty nanosecond reconfigurable hybrid optical switches for data centre networks*, in 2017 Conference on Lasers and Electro-Optics (CLEO), May 2017, pp. 1–2.

[10] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. Syst. Sci., 69 (2004), pp. 485–497.

[11] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, *Helios: a hybrid electrical/optical switch architecture for modular data centers*, in Proceedings of ACM SIGCOMM, 2010.

[12] M. Ghaffari, B. Haeupler, and G. Zuzic, *Hop-constrained oblivious routing*, in Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021, New York, NY, USA, 2021, Association for Computing Machinery, p. 12081220.

[13] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, *Projector: Agile reconfigurable data center interconnect*, in Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM 16, New York, NY, USA, 2016, Association for Computing Machinery, p. 216229.

[14] A. Gupta, M. T. Hajiaghayi, and H. Räcke, *Oblivious network design*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006, ACM Press, 2006, pp. 970–979.

[15] M. T. Hajiaghayi, R. D. Kleinberg, F. T. Leighton, and H. Räcke, *New lower bounds for oblivious routing in undirected graphs*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006, ACM Press, 2006, pp. 918–927.

[16] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, *Firefly: A reconfigurable wireless data center fabric using free-space optics*, in Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM 14, New York, NY, USA, 2014, Association for Computing Machinery, p. 319330.

[17] C. Harrelson, K. Hildrum, and S. Rao, *A polynomial-time tree decomposition to minimize congestion*, in SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003), A. L. Rosenberg and F. M. auf der Heide, eds., ACM, 2003, pp. 34–43.

[18] P. Harsha, T. P. Hayes, H. Narayanan, H. Räcke, and J. Radhakrishnan, *Minimizing average latency in oblivious routing*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008, S. Teng, ed., SIAM, 2008, pp. 200–207.

[19] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. Lim, and R. Soulé, *Semi-oblivious traffic engineering: The road not taken*, in 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018, S. Banerjee and S. Seshan, eds., USENIX Association, 2018, pp. 157–170.

[20] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, *Circuit switching under the radar with reactor*, in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, WA, Apr. 2014, USENIX Association, pp. 1–15.

[21] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, *Expanding across time to deliver bandwidth efficiency and low latency*, in 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), Santa Clara, CA, Feb. 2020, USENIX Association, pp. 1–18.

[22] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, *Rotornet: A scalable, low-complexity, optical datacenter network*, in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017, pp. 267–280.

[23] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, *Integrating microsecond circuit switching into the data center*, in Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM 13, New York, NY, USA, 2013, Association for Computing Machinery, p. 447458.

[24] H. Räcke, *Minimizing congestion in general networks*, in The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings., 2002, pp. 43–52.

[25] H. Räcke, *Optimal hierarchical decompositions for congestion minimization in networks*, STOC '08, New York, NY, USA, 2008, Association for Computing Machinery.

[26] V. Shrivastav, A. Valadarsky, H. Ballani, P. Costa, K. S. Lee, H. Wang, R. Agarwal, and H. Weatherspoon, *Shoal: A network architecture for disaggregated racks*, in 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, 2019, USENIX Association.

[27] L. G. Valiant, *A scheme for fast parallel communication*, SIAM J. Comput., 11 (1982), pp. 350–361.

[28] L. G. Valiant and G. J. Brebner, *Universal schemes for parallel communication*, (1981), pp. 263–277.

[29] L. G. Valiant and G. J. Brebner, *Universal schemes for parallel communication*, in Proceedings of the thirteenth annual ACM symposium on Theory of computing, 1981, pp. 263–277.

[30] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, *C-through: Part-time optics in data centers*, in Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM 10, New York, NY, USA, 2010, Association for Computing Machinery, p. 327338.