DOI: 10.14529/jsfi140303

Early evaluation of direct large-scale InfiniBand networks with adaptive routing

***Alexander N. Daryin*[1,2], *Anton A. Korzh*[3]**

We assess the problem of choosing optimal direct topology for InfiniBand networks in terms of performance. Newest topologies like Dragonfly, Flattened butterfly and Slim Fly are considered, as well as standard Tori and Hypercubes. We consider some reasonable extensions to InfiniBand hardware which could be implemented by vendors easily and may allow reasonable routing algorithms for such topologies. A number of routing algorithms are proposed and compared for various traffic patterns. Mapping algorithms for Dragonfly and Flattened Butterfly are proposed. Based on this research it has been decided to use Flattened Butterfly topology for system #22 in November 2014 Top 500 list.

Keywords: adaptive routing, InfiniBand, high-radix topology, network simulation.

Introduction

InfiniBand is a *de facto* industry standard in supercomputing with almost 45% entries on the latest Top 500 list [1]. However, the largest systems on the list tend to have custom interconnection networks, with only 2 of Top 10 systems built using InfiniBand. A possible explanation of this fact is that large number of nodes drives the need for modern cost-effective topologies such as Dragonfly and Flattened Butterfly that are not readily supported in current InfiniBand infrastructure. Moreover, these topologies in turn require advanced routing algorithms, including adaptive routing, that cannot be implemented within InfiniBand specification.

The purpose of this paper is to investigate possible performance of large InfiniBand networks given that some extra features are added to the switches.

1. Topologies

We compare topologies and routing algorithm on the following reference configuration. Network contains $n = 2{,}048$ switches with concentration $C = 8$, thus 16,384 nodes. Each switch has $d = 36$ ports, 8 of those taken by nodes and 28 are available for inter-switch connections. Hardware is assembled in 32 twin racks of 64 switches (512 nodes) each. Hence it is desirable to exploit local connections in groups of 16, 32, or 64 switches to reduce cabling cost. Only direct topologies are considered. Number of switches is fixed to maintain roughly the same cost for different topologies.

Theoretical upper bound [16] on the relative bisection bandwidth β is $\frac{d-2}{2C} + o(1) \approx 162{,}5\%$. Practically constructible graphs with greatest known bisection are Ramanujan graphs [22] with lower bound on bisection $\frac{d-2\sqrt{d-1}}{2C}$ for $d = q + 1$ where q is a prime power. For $q = 25$ this yields a 100% relative bisection. However, such graphs are available in a very few sizes and have intractable structure for designing routing algorithm on them. Therefore we assume that a 50% relative bisection bandwidth obtained on several topologies is a reasonable value.

[1]T-Platforms, Moscow, Russia
[2]National Research University – Higher School of Economics, Moscow, Russia
[3]Micron Technology, Inc., Boise, USA

1.1. Tori

A *torus* is a Cartesian product of cycles. Up to 4D tori may be supported in InfiniBand. For higher dimensions it is impossible to provide deadlock freedom of minimal routing. Torus-2QoS, a routing engine for tori in the Open Subnet Manager (OpenSM), only supports 2D and 3D tori.

We shall use the following notation both for tori and for FlatFly later: N is the number of dimensions; K_n is the size of dimension n; L_n is the width of links within dimension n.

The relative bisection of a torus is then $\beta = \min\{4L_n/CK_n\}$ over $n = \overline{1,N}$, its degree is $d = C + 2\sum_n L_n$. A combinatorial search yields the following two options with the greatest possible bisection:

Dimension	Size	Link Widths	Degree	Diameter	Bisection
3D	$8 \times 16 \times 16$	3, 5, 5	34	20	15,6%
4D	$4 \times 8 \times 8 \times 8$	2, 4, 4, 4	36	14	25,0%

Cayley graphs of commutative groups are a well known alternative of tori, having a similar cabling structure and routing algorithms, but much smaller diameter and greater bisection bandwidth. We do not consider such topologies explicitly, but e.g. a FlatFly is a Cayley graph of a product of cyclic groups with appropriate generators; Dragonfly is a vertex-transitive graph and thus has a structure similar to that of a Cayley graph of a non-commutative group.

1.2. FlatFly

Consider a Cartesian product of N full graphs. Historically, several names were used for this topology:

- *Generalized Hypercube* [5] (this term implies single inter-switch links and one adapter per switch);
- *Flattened Butterfly* [14] (implying single inter-switch links, and equals orders of full graphs equal to concentration);
- *HyperX* [3] (subsumes both previous cases).

We shall refer to this topology as *FlatFly* (FF) and use the same notation as for tori: K_n for orders of full graphs and L_n for link width in each of them, $n = \overline{1,N}$. According to [3], the relative bisection bandwidth of the FlatFly network is $\beta = \min\{K_nL_n\}/2C$. Diameter of a FlatFly is equal to its dimension N, and degree is equal to $C + \sum_n L_n(K_n - 1)$.

Parameters of optimal FlatFly network are determined by a combinatorial search procedure outlined in [3]. For our setup this is a 4D FlatFly with dimensions $4 \times 8 \times 8 \times 8$ and link widths 2, 1, 1, 1. It's diameter is 4, radix is 35, and relative bisection is 50%.

Cabling of this network is relatively easy. A 8×8 part (3rd and 4th dimensions) occupies a twin rack. Then each rack is connected to 3 racks in the first dimension with bundles of 128 cables, and to 7 racks in the second dimension with bundles of 64 cables.

FlatFly may serve as a building block for other topologies. For example, groups in Dragonfly topology are usually connected as a full graph (which is a 1D FlatFly), or a 2D FlatFly [9].

1.3. Hypercube

A *hypercube* is a Cartesian product of 2-vertex complete graphs, and may be considered a particular case of both torus and FlatFly. Unlike tori, dimension of hypercube built with

InfiniBand is only limited by the number of nodes in subnet (which in practice means a maximum of 14D). However, as a FlatFly it usually is not optimal in terms of bisection bandwidth and diameter.

In our setup, hypercube is 11D with double links in each dimension. This gives diameter 11, relative bisection 25%, and radix 30. The remaining 6 ports may be utilized to increase bisection bandwidth within one rack to 37,5%.

1.4. Dragonfly

A *Dragonfly* [15] consists of groups connected as full graphs (with "local" links). These groups, regarded as vertices, are again connected as a full graph (with "global" links).

Denote by K the size of a group, and by G the number of global links per switch. Then there are KG groups containing K^2G switches[4]. A typical shortest path crosses one "global" and two "local" full graphs, hence the relative bisection is $\beta = \min\{K/4C, G/2C\}$. Graph diameter is 3, and degree is $C + G + K - 1$.

It is reasonable to choose K and G such that $K/4C = G/2C$, which gives $K = 16$ and $G = 8$ for our setup. The radix is 30. Unlike other considered topologies, cabling of global links in a Dragonfly is really messy.

1.5. Slim Fly

A *Slim Fly* [4] topology is built with McKay–Miller–Širáň (MMS) graphs [20] parameterized by a prime power q. Its $2q^2$ vertices are elements of linear space $\mathbb{Z}_2 \times \mathbb{F}_q \times \mathbb{F}_q$, where $\mathbb{F}_q$ is a field of order q. Coordinates of vertices are denoted as (t, x, y). There are two kinds of edges:

1. y-edges: (t, x, y_0) is connected with (t, x, y_1) iff $y_1 - y_0 \in X_t$. For $q = 2^p$, the sets X_t are $X_0 = \{1, \xi^2, \xi^4, \ldots, \xi^{q-2}\}$ and $X_1 = \{\xi, \xi^3, \ldots, \xi^{q-1}\}$, where ξ is a primitive element of $\mathbb{F}_q$.
2. t-edges: $(0, x_0, y_0)$ is connected with $(1, x_1, y_1)$ iff $y_0 = x_0x_1 + y_1$.

Diameter of a MMS graph is 2. For $q = 2^p$ its radix is $C + 3Lq/2$, and we hypothesize that its relative bisection is $\beta = Lq/2C$. Here L is link width.

Unfortunately, an MMS graph with 2,048 switches corresponds to $q = 32$ and requires switches of radix 56. Instead of that, we will consider Cartesian products of Slim Fly and FlatFly. There are two options for our setup:

	Slim Fly q	L	FlatFly K_n	L_n	Degree	Diameter	Bisection
SF×FF-1	4	2	8×8	1, 1	34	4	50%
SF×FF-2	8	1	16	1	35	3	50%

SF×FF-1 replaces 4×8 dimensions of our FlatFly setup with a Slim Fly. This further simplifies cabling: now each twin rack is connected to 6 other racks using bundles of 128 cables. Also, it saves 1,024 optical cables.

SF×FF-2 has the same radix as FlatFly, but a lower diameter of 3, and groups of 16 switches have relative bisection 100%.

[4]Actually there are $KG + 1$ groups and $(KG + 1)K$ switches, but we allow for a small irregularity in topology omitting one global link per group, for the total number of switches to be a power of 2.

2. Mapping algorithms

Some routing algorithms assume that the structure of a subnet is given in advance, and that all switches have their coordinates assigned. This may not be always the case in practice. For example, in OpenSM a routing engine is given only a list of nodes and links, and has to recover network structure on its own.

Here we present mapping algorithms for FlatFly and Dragonfly.

2.1. Mapping a FlatFly

To map a FlatFly, we construct an equivalence relation ε on the set of its links, using the following rules.

1. Two uni-directional links e_1 and e_2 constituting one bi-directional link are equivalent: $e_1 \varepsilon e_2$.
2. Two parallel links e_1 and e_2 connecting the same two switches are equivalent: $e_1 \varepsilon e_2$.
3. Three links e_1, e_2, e_3 forming a rectangle are equivalent: $e_1 \varepsilon e_2$, $e_2 \varepsilon e_3$.
4. If four links $e_1, \ldots, e_4$ form a rectangle, then the opposite sides of this rectangle are equivalent: $e_1 \varepsilon e_3$, $e_2 \varepsilon e_4$.

The total number of traversed edges here is at most nd^2.

The rules above do not define the entire relation ε; we need to compute their transitive closure. This may be done efficiently using the Union-Find algorithm from [25] with $O(n \ln n)$ operations.

Each equivalence class D_i of ε corresponds to a dimension of FlatFly, $i = \overline{1, N}$. Using ε, we then assign coordinates to switches. Additional N equivalence relations ε_i are constructed using the following rule. If edge e connects vertices v_1 and v_2, and $e \in D_i$, then $v_1 \varepsilon_j v_2$ for all $j \neq i$. After calculating the transitive closure, each equivalence class C_{ij} of ε_i corresponds to a single value of i-th coordinate, $j = \overline{1, K_i}$.

The described algorithm can be applied to incomplete topologies. Experiments on our setup have shown that FlatFly is still mapped correctly if 20% switches and 15% links are missing at random. Hypercube is mapped correctly with 15% missing switches and 10% missing links.

2.2. Mapping a Dragonfly

For Dragonfly mapping, we construct the same equivalence relation ε as for FlatFly. Each equivalence class of ε containing more than one edge represents local links of one Dragonfly group. All other links are global links.

In our setup, incomplete Dragonfly is mapped correctly with 40% switches and 10% links missing at random.

3. Routing

InfiniBand [2] fabric is a packet-switching network. Each network device is assigned a subnet address called *Local Identifier* (LID). Each link has several *Virtual Lanes* (VL) used by the routing engine to provide deadlock freedom and quality of service. The route of each packet from source to destination is determined by two header fields set at the source node: Destination LID (DLID) and Service Level (SL).

LID is a 16-bit integer, but the actual address capacity is 48K since 16K addresses are used for multicast groups. SL takes values from 0 to 15. There may be 1, 2, 4, 8, or 15 VLs per link; a common value for modern hardware is 8 VLs.

There are four mechanisms constituting an InfiniBand routing function:

1. Each switch has a *Linear Forwarding Table* (LFT) that maps DLIDs into output ports.
2. After selecting the output port, switch consults an *SL2VL* Table to select output VL depending on input port, output port and SL.
3. Before sending a packet, the source node requests a *Path Record* from the subnet manager, to find out which SL should be used for these source and destination nodes.
4. *LID Mask Control* (LMC) feature allows assigning more than one LID per node. This results in several paths between the same two nodes that may be used for load balancing or other purposes.

In theory [8], one may consider routing functions in a very general form

$$(\text{Source}, \text{Destination}, \text{Switch}, \text{In Port}, \text{In VC}) \to (\text{Out Port}, \text{Out VC}).$$

InfiniBand factors this dependence into

$$\begin{cases} (\text{Source}, \text{Destination}) \to (\text{DLID}, \text{SL}), \\ (\text{Switch}, \text{DLID}) \to \text{Out Port}, \\ (\text{Switch}, \text{SL}, \text{In Port}, \text{Out Port}) \to \text{Out VL}. \end{cases}$$

In particular, the VL chosen does not directly depend on the destination, which limits the choices in the design of deadlock-free routing algorithms.

3.1. Adaptive routing

InfiniBand specification [2] does not support *Adaptive Routing* (AR). Moreover, a number of measures has been taken to ensure that packets arrive in order. On the other hand, it is always possible to find a traffic pattern on which a particular static routing will achieve only a small portion of the available bisection bandwidth [11]. For example, in our setup if 8 nodes on one switch communicate with 8 nodes on a neighbor switch using any minimal routing, they will only get 12,5% bandwidth, even if relative bisection is 50,%.

A number of strategies have been proposed to implement AR in InfiniBand. *Multipath routing* uses LMC and congestion notification mechanism to select a least-congested path at the source [18]. A possible modification to switch hardware [19] would treat all LIDs assigned to the same node interchangeably and dynamically choose output port from LFT entries corresponding to them. Finally, Mellanox claims support of AR in its switches [21].

Here we assume that switch hardware is modified in such a way that for each destination it is possible to specify a *set* of output ports instead of a single port. This allows implementing a *minimal* adaptive routing. However, although it may perform better on some traffic patterns, many other patterns will not benefit from adaptivity if only one shortest path is available between a pair of nodes (which is exactly the case in our example with 8 collocated nodes talking to nodes on a neighbor switch). Furthermore, a fully adaptive minimal routing will in general have credit loops that cannot be eliminated using standard InfiniBand features.

The key problem in designing non-minimal adaptive routing algorithms for InfiniBand is that no information is accumulated in packet header as a packet traverses the fabric. The only

header field not covered by the Invariant CRC is VL. However, input VL does not influence neither output port nor output VL. Thus, it is hard, if possible, for a non-minimal routing to guarantee progress towards destination and to avoid livelocks.

Since VL is the only header field that can change from hop to hop, we assume the following modifications to the switch hardware:

1. Output VL is selected using input VL: either it is simply incremented (so called *VL hopping*), or input VL is used instead of SL in the SL2VL mechanism (so that it becomes VL2VL).
2. There is a separate forwarding table for each input VL[5].

With these modifications, the routing function is now described as

$$\begin{cases} (\text{Source}, \text{Destination}) \to (\text{DLID}, \text{VL}), \\ (\text{Switch}, \text{In VL}, \text{DLID}) \to \{\text{Out Port}_1, \ldots, \text{Out Port}_k\}, \\ (\text{Switch}, \text{In VL}, \text{In Port}, \text{Out Port}) \to \text{Out VL}. \end{cases}$$

To reduce latency, it is beneficial to prefer shorter routes when possible. This is not part of a routing, but of a *selection function* that chooses one port from a set based on congestion information. In our simulation we assume that two priorities are available: high priority for minimal paths and low priority for non-minimal. An exception is deflection routing requiring three levels of priority.

We finally note that using AR also requires significant changes to software (including MPI) to properly handle out-of- order packets. This is beyond the scope of this article.

3.2. Torus routing

A deadlock-free routing for tori usually uses two virtual channels to prevent cycles within a dimension[6]: one for packets crossing a dateline in that dimension, another for all other packets. To avoid cycles between dimensions, they are traversed in a fixed order, hence the name *Direction Order Routing* (DOR).

It turns out that dateline crossing at each dimension should be determined at the source node and stored in SL (because VL cannot be chosen per destination address). Since SL has 4 bits, the maximum dimension of InfiniBand tori is 4D.

Summing up, adaptive routing for tori can be only used to select one of the parallel output links.

3.3. FlatFly routing

DOR routing can be applied to FlatFly. Unlike tori, there is no need to use two VLs since a packet traverses at most one link in each dimension.

Adaptive DOR (ADOR) allows an additional hop in each dimension [9, 26]. This is implemented by the following rules:

1. VL 0 is used when input port belongs to a host or to a different dimension than output port.
2. VL 1 is used if input and output ports belong to the same dimension.

[5]Some routing algorithms require a separate forwarding table for each input VL and each input port.

[6]Limitations of InfiniBand routing preclude using other deadlock-avoidance schemes such as turn-based routing. VL hopping is also not an option since the diameter of torus network is larger than the number of available VLs.

3. Packet with input VL 0 is routed to all switches in current dimension, preferring the shortest route.
4. Packet with input VL 1 is routed in current dimension along the shortest route.

DOR and ADOR do not use a large portion of crossbar (all transitions from higher to lower dimensions). *Mixed DOR* uses the remaining two bits of VL to make possible several orderings of dimensions.

1. At the source node, SL is set of $d \mod 4$, where d is the index of the destination host within the destination switch.
2. Based on SL, one of the four dimension orders are used[7]: $(1,2,3,4)$, $(2,4,1,3)$, $(3,1,4,2)$, $(4,3,2,1)$.

Mixed ADOR combines Mixed DOR and ADOR and chooses VL as either $2SL$ (for hops in dimension order) or $2SL+1$ (for extra hops in the same dimension).

Another pitfall of (A)DOR is that all packets with the same source and destination switches will meet at the intermediate switches. *Twisted DOR* solves this problem by making an obligatory non-minimal hop at the beginning. Assume that we are at the switch with coordinates $(x_1, \ldots, x_n)$ and d is the index of the destination host.

1. If a packet comes from a host[8], route it to switch with coordinates $(x_1, \ldots, d \mod K_n)$ using VL 1 (if $d = x_n \mod K_n$, do nothing).
2. After that, proceed with DOR using VL 0.

Twisted ADOR is a combination of Twisted DOR and ADOR, with the exception that no additional hop is allowed in the last dimension (since it has been already made at the first step).

Finally, *Twisted Mixed DOR* combines Twisted and Mixed DORs, and the same goes for *Twisted Mixed ADOR*.

3.4. Hypercube routing

As hypercube is a particular case of a FlatFly, we can apply the latter's DOR and Mixed DOR routing algorithms. With only two switches per dimension, ADOR is the same as DOR here. Also, "Twisted" routings did not perform well on simulator, so we don't use them either.

There are special AR algorithms for hypercubes, e.g. described in [10]. We have simulate three of them:

1. *Negative First*: route adaptively in all negative dimensions (those where coordinate of source is 1 and that of destination is 0), then route adaptively in all other dimensions.
2. *All but one (ABO) Negative First*: route adaptively in all negative dimensions except dimension 1, then route adaptively in all other dimensions.
3. *All but one (ABO) Positive Last*: route adaptively in all negative dimension and dimension 1, then route adaptively in all other dimensions.

3.5. Dragonfly routing

Although Dragonfly features low diameter and high relative bisection, advanced routing techniques are required to achieve appropriate network throughput. Indeed, any two groups are connected with only one link, and if all hosts in the first group happen to communicate with all

[7] These permutations where found using integer linear programming minimizing the maximum load on each part of the crossbar.

[8] This algorithm requires selection of forwarding table depending on input port .

hosts in the second one sharing the same link, the resulting throughput will be $1/KC$, which is less than 1% for our setup.

A popular Dragonfly routing UGAL is based on a scheme proposed by Valiant [27]: most packets are first routed to a randomly chosen intermediate group, and only then to the destination. However, there is no hardware support for Valiant-type routings in InfiniBand, so it is impossible to implement UGAL together with RDMA.

The basic *Static* routing for Dragonfly is[9]:

1. in source group, make a local hop to the node that has a link to the destination group;
2. make a global hop;
3. in destination group, make a local hop to the destination switch.

Here we use a family of adaptive routing algorithms that use only local congestion information. They are coded by a combination of letters G, S, I, and D, each allowing extra non-minimal hops of different kinds:

G: extra global hop is allowed (leading to an intermediate group);
S: extra local hop is allowed in the source group;
I: extra local hop is allowed in the intermediate group;
D: extra local hop is allowed in the destination group.

At most 8 total hops are possible. Up to 6 VLs are required: VLs 0 and 1 are used in the source group, 2 and 3 in the intermediate group, 4 and 5 in the destination group. Higher priority is assigned to minimal routes.

3.6. SlimFly and SF×FF routing

Here we describe the structure of the minimal routing in Slim Fly. Consider the following cases when routing a packet from (t_s, x_s, y_s) to (t_d, x_d, y_d).

1. $t_s = t_d = t$, $x_s = x_d = x$. Then either $y_s - y_d \in X_t$, or there exists $y_i \in F_q$ such that $y_s - y_i \in X_t$ and $y_i - y_d \in X_t$ (by construction of X_t). Depending on that, the shortest route is either "y" or "yy" (encoded by the link types). Note that it cannot be one of "ty" or "yt" since $t_s = t_d$. Also it cannot be "tt" as will be seen below.
2. $t_s = t_d = t$, $x_s \neq x_d$. Since y-links do not change x, the only possible option is the "tt" route. Denote by (t_i, x_i, y_i) the intermediate vertex, $t_i = 1 - t$. Then[10] if $t = 0$,

$$\begin{cases} y_s = x_s x_i + y_i; \\ y_d = x_d x_i + y_i \end{cases} \quad \Longrightarrow \quad \begin{cases} x_i = (y_s - y_d)/(x_s - x_d), \\ y_i = y_s - x_s x_i. \end{cases} \tag{1}$$

If $t = 1$,

$$\begin{cases} y_i = x_s x_i + y_s; \\ y_i = x_d x_i + y_d \end{cases} \quad \Longrightarrow \quad \begin{cases} x_i = -(y_s - y_d)/(x_s - x_d), \\ y_i = y_s + x_s x_i. \end{cases} \tag{2}$$

Note that these systems are not solvable if $x_s = x_d$. All arithmetic is in the field F_q. Relations (1) and (2) may be unified as

$$\begin{cases} x_i = (-1)^t (y_s - y_d)/(x_s - x_d), \\ y_i = y_s + (-1)^t x_s x_i. \end{cases}$$

[9] Note that this routing is not minimal, as some shortest paths consist of two global hops.
[10] All arithmetic here is in $\mathbb{F}_q$.

3. $t_s \neq t_d$. If $(t_d - t_s)(y_s - y_d) = x_s x_d$, then the shortest route is the corresponding t-link. Otherwise it should be either "ty" or "yt".
   (a) "ty". The intermediate node is (t_d, x_d, y_i), and then

$$y_s - y_i = (t_d - t_s)x_s x_d \quad \Longrightarrow \quad y_i = y_s - (t_d - t_s)x_s x_d.$$

   given that $y_i - y_d \in X_{t_d}$.
   (b) "yt". The intermediate node is (t_s, x_s, y_i), and then

$$y_i - y_d = (t_d - t_s)x_s x_d \quad \Longrightarrow \quad y_i = y_d + (t_d - t_s)x_s x_d.$$

   given that $y_s - y_i \in X_{t_s}$.
   At least one of two cases should hold since the diameter of MMS graph is 2.

As seen from the minimal routing analysis, the number of shortest paths between (t_s, x_s, y_s) and (t_d, x_d, y_d) is

- 1 or 2 if $t_s \neq t_d$ (but the case of 1 is encountered more often);
- exactly one of $t_s = t_s$, $x_s \neq x_d$;
- usually multiple if $t_s = t_d$, $x_s = x_d$.

Experiments show that most source-destination pairs only have 1 shortest path between them.

To avoid deadlocks, on two-hop paths VL 0 is used on the first hop and VL 1 on the second. This may be implemented using standard SL2VL feature: if a packet comes from a host, select VL 0, otherwise select VL 1.

DOR, ADOR and their Mixed and Twisted variants may be used for SF×FF products, considering Slim Fly as a dimension of a Flat Fly.

3.7. Topology agnostic algorithms

There exists a number of static generic routing algorithms for InfiniBand: UpDn [17], DF-SSSP [7], LASH [23]. Achieving deadlock freedom for an arbitrary topology is challenging, and mentioned algorithms do not provide such a guarantee.

Under our assumptions, VL hopping may be used to provide deadlock freedom. Indeed, no credit loops are possible if VLs are used in strictly increasing order. This works when maximum path length does not exceed the number of available VLs. In our setup, we can use VL hopping for FlatFly, Dragonfly and SF×FF.

We use a family of algorithms that we call *Distance Based Routing*. Its parameters are E – the number of extra hops allowed, and F – the flag indicating whether deflection is permitted.

Suppose that shortest distance between source and destination switches is $D > 0$, and the remaining distance at current switch is $d > 0$.

1. Source host assigns VL $= D + E \mod 8$ to the packet.
2. On each hop, VL is decremented $\mod 8$.
3. For each output port p leading to a switch, calculate distance from that switch to the destination switch, d_p.
   (a) If $d > d_p$, route to p with high priority (shortest path).
   (b) If $d = d_p \leq \mathrm{VL}_{\mathrm{out}}$, route to p with medium priority (routing sideways).
   (c) If $d < d_p \leq \mathrm{VL}_{\mathrm{out}}$ and F is set, route to p with low priority (deflection).

In these terms, fully adaptive minimal routing corresponds to $E = 0$ and is called *Distance Based*. Non-minimal routing ($E > 0$) without deflection is denoted as *Distance Based + E*.

Finally, routing with deflection is denoted *Deflection + E*. In the latter case $E \geq 2$, otherwise enabling deflection makes no sense.

4. Simulation results

In this section, we compare performance of described topologies and routing algorithms. For large-scale networks, we have developed a parallel event-driven simulator following methodology described in [6]. It has been verified against BookSim and real-world data and produces consistent results.

The following table contains relative saturation bandwidth [6] for each topology, routing algorithm and traffic pattern. Since network size is a power of two, we use bit patterns for comparison: complement, reverse, rotation, shuffle, and transpose. There is also uniform (random all-to-all) pattern.

Routing	Uniform	Cmpl	Rvrs	Bit Rotn	Shuf	Trns
Torus						
DOR 3D	26,5%	15,6%	7,6%	7,0%	7,6%	7,1%
DOR 4D	48,2%	25,0%	2,9%	17,0%	12,2%	2,9%
FlatFly						
DOR	81,3%	11,7%	1,6%	12,5%	7,8%	2,3%
Mixed DOR	89,8%	11,7%	5,5%	7,8%	5,5%	2,3%
Twisted DOR	24,2%	11,7%	11,7%	11,7%	12,5%	5,5%
Twisted Mixed DOR	46,9%	12,5%	11,7%	9,0%	10,9%	2,3%
ADOR	52,3%	50,0%	3,1%	25,0%	24,2%	2,3%
Mixed ADOR	56,3%	50,0%	10,2%	24,2%	24,2%	5,5%
Twisted ADOR	24,2%	50,0%	21,1%	24,2%	24,2%	11,7%
Twisted Mixed ADOR	53,1%	28,1%	22,7%	24,2%	24,2%	5,5%
Distance Based	93,8%	11,7%	9,4%	11,7%	9,4%	10,2%
Distance Based +1	95,3%	11,7%	14,8%	24,2%	24,2%	24,2%
Distance Based +2	94,5%	24,2%	14,8%	24,2%	24,2%	24,2%
Distance Based +3	95,3%	24,2%	15,6%	39,8%	24,2%	24,2%
Distance Based +4	96,1%	50,8%	14,8%	39,8%	37,5%	37,5%
Deflection +2	96,1%	24,2%	24,2%	24,2%	24,2%	24,2%
Deflection +3	94,5%	24,2%	24,2%	50,0%	24,2%	24,2%
Deflection +4	94,5%	50,8%	24,2%	50,0%	24,2%	24,2%
Hypercube						
DOR	24,2%	24,2%	0,8%	24,2%	11,7%	0,8%
Mixed DOR	24,2%	24,2%	2,3%	11,7%	11,7%	0,8%
Negative First	2,3%	0,0%	0,8%	0,8%	0,8%	0,8%
ABO Negative First	5,5%	0,0%	0,8%	0,8%	0,8%	0,8%
ABO Positive Last	5,5%	0,0%	0,8%	0,8%	0,8%	0,8%
Dragonfly						
Static	24,2%	5,5%	5,5%	5,5%	5,5%	5,5%
S	64,8%	0,0%	11,7%	0,8%	0,8%	9,4%
D	68,8%	0,0%	11,7%	0,8%	0,8%	10,9%

Routing	Uniform	Bit Cmpl	Rvrs	Rotn	Shuf	Trns
SD	50,8%	0,0%	11,7%	0,8%	0,8%	11,7%
G	11,7%	5,5%	5,5%	5,5%	5,5%	5,5%
GS	11,7%	5,5%	11,7%	5,5%	5,5%	9,4%
GI	11,7%	11,7%	5,5%	10,9%	11,7%	5,5%
GD	11,7%	11,7%	5,5%	11,7%	21,9%	5,5%
GSI	22,7%	11,7%	5,5%	10,9%	11,7%	5,5%
GID	11,7%	46,9%	5,5%	24,2%	24,2%	5,5%
GSD	11,7%	11,7%	11,7%	11,7%	11,7%	10,9%
GSID	23,4%	24,2%	5,5%	24,2%	24,2%	5,5%
Distance Based	82,0%	0,0%	5,5%	0,8%	0,8%	5,5%
Distance Based +1	61,7%	0,8%	10,9%	0,8%	2,3%	11,7%
Distance Based +2	59,4%	0,8%	11,7%	2,3%	2,3%	9,4%
Distance Based +3	50,0%	0,8%	11,7%	2,3%	2,3%	10,9%
Distance Based +4	24,2%	2,3%	11,7%	2,3%	2,3%	10,9%
Deflection +2	57,0%	0,8%	5,5%	2,3%	2,3%	9,4%
Deflection +3	48,4%	2,3%	11,7%	2,3%	5,5%	11,7%
Deflection +4	24,2%	2,3%	11,7%	5,5%	5,5%	11,7%
SF×FF-1						
Distance Based	80,5%	11,7%	11,7%	10,9%	5,5%	9,4%
Distance Based +1	80,5%	11,7%	39,1%	11,7%	14,8%	11,7%
Distance Based +2	80,5%	19,5%	30,5%	11,7%	11,7%	11,7%
Distance Based +3	80,5%	19,5%	30,5%	11,7%	11,7%	11,7%
Distance Based +4	80,5%	19,5%	30,5%	11,7%	11,7%	11,7%
Deflection +2	82,0%	11,7%	24,2%	23,4%	21,9%	24,2%
Deflection +3	81,2%	19,5%	24,2%	24,2%	24,2%	32,8%
Deflection +4	80,5%	21,9%	24,2%	24,2%	24,2%	32,8%
SF×FF-2						
Distance Based	61,7%	5,5%	5,5%	5,5%	5,5%	5,5%
Distance Based +1	61,7%	11,7%	14,8%	7,0%	7,0%	7,0%
Distance Based +2	61,7%	11,7%	14,8%	7,0%	7,0%	8,6%
Distance Based +3	61,7%	11,7%	17,2%	7,0%	7,0%	8,6%
Distance Based +4	71,9%	11,7%	17,2%	7,0%	7,0%	7,0%
Deflection +2	61,7%	10,2%	11,7%	11,7%	10,9%	11,7%
Deflection +3	61,7%	11,7%	24,2%	11,7%	11,7%	23,4%
Deflection +4	74,2%	11,7%	18,8%	11,7%	10,9%	11,7%

As expected, tori and hypercube perform worse than other topologies on uniform traffic due to lower relative bandwidth.

For FlatFly, the best routing is "Distance Based +4". Comparing its results with "Distance Based +3" and ADOR family, we conclude that the optimal number of extra hop is equal to dimension of FlatFly, and it's critical that these hops are made without particular dimension order. Deflection helps to achieve full bisection bandwidth on some patterns, but reduces bandwidth for others.

Hypercube shows low performance on bit reverse and bit transpose patterns. Turn model routings (negative first/ positive last) perform really badly, as they create significant imbalance in packet distribution across the network.

Results for Dragonfly are mixed, with GID and GSD routings showing best results for bit traffic patterns and low performance on uniform traffic at the same time. Distance based and deflection routings perform worse than our specialized algorithms.

Results for SF×FF-1 are better than for SF×FF-2. Deflection helps to break the 12,5% barrier on three traffic patterns, with "Deflection +4" showing the best results for almost all traffic patterns.

Comparing different topologies, FlatFly and SF×FF-1 seem to be the winners, according to the following criteria

1. performance on uniform traffic;
2. performance of best routing on bit traffic patterns;
3. performance of "Distance Based" routing[11];

5. Related work

A performance comparison of various types of interconnects is given in [13]. In [28, 29] authors analyze and simulate adaptive routing in InbiniBand fat trees. Improving performance of large-scale InfiniBand networks through optimized task placement is subject of [24]. DFSSSP routing [7, 12] optimizes InfiniBand fabric performance by statically balancing network paths.

6. Conclusions

In this paper, we have analyzed possible topologies for large-scale InfiniBand systems (including tori, hypercube, Dragonfly, Flattened Butterfly, Slim Fly). In most cases, adaptive routing is required in order to achieve theoretical bandwidth limits. We analyze standard InfiniBand routing and list a minimum set of features that should be added in order to support adaptive routing. We describe specialized adaptive routing algorithms for each topology, and a family of topology-agnostic (distance-based) routings. Then we provide simulation results for considered topologies and routing algorithms. Best performance results are shown by Flattened Butterfly and a combination of Flattened Butterfly and Slim Fly.

This work has been partially supported by the Russian Ministry of Education and Science (project # 14.579.21.0074).

References

1. November 2014 TOP500 list (`http://top500.org/lists/2014/11/`).

2. *InfiniBand^{TM} Architecture Specification*, volume 1. IBTA, 2007.

[11] Results for "Distance Based" routing give an upper bound for any static minimal routing. Similarly, results for DOR and Mixed DOR give an upper bound for static DOR-based algorithms.

3. J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber. HyperX: topology, routing, and packaging of efficient large-scale networks. In *Proceedings of SC'09*. IEEE, Nov 2009.

4. M. Besta and T. Hoefler. Slim Fly: A cost effective low-diameter network topology. In *Proceedings of SC'14*, pages 348–359. IEEE, Nov 2014.

5. L. N. Bhuyan and D. P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Transactions on Computers*, 33(4):323–333, Apr 1984.

6. W. J. Dally and B. P. Towles. *Principles and Practices of Interconnection Networks*. Elsevier Science, 2003.

7. J. Domke, T. Hoefler, and W. E. Nagel. Deadlock-free oblivious routing for arbitrary topologies. In *Proceedings of IPDPS-11*, pages 616–627. IEEE, May 2011.

8. J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*. Elsevier Science, 2002.

9. G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard. Cray Cascade: a scalable HPC system based on a Dragonfly network. In *Proceedings of SC'12*, pages 1–9. IEEE, Nov 2012.

10. C. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of the ACM*, 41(5):874–902, Sep 1994.

11. T. Hoefler, T. Schneider, and A. Lumsdaine. Multistage switches are not crossbars: Effects of static routing in high-performance networks. In *Proceedings of CLUSTER 2008*, pages 116–125. IEEE, Sep 2008.

12. T. Hoefler, T. Schneider, and A. Lumsdaine. Optimized routing for large-scale infiniband networks. In *Proceedings of HOTI'09*, pages 103–111, Aug 2009.

13. D. J. Kerbyson, K. J. Barker, A. Vishnu, and A. Hoisie. A performance comparison of current HPC systems: Blue Gene/Q, Cray XE6 and InfiniBand systems. *Future Generation Computer Systems*, 30:291–304, Jan 2014.

14. J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of ISCA'07*, number 13, pages 126–137. ACM, May 2007.

15. J. Kim, W. J. Dally, S. L. Scott, and D. Abts. Cost-efficient Dragonfly topology for large-scale systems. *IEEE Micro*, 29(1):33–40, 2008.

16. A. V. Kostochka and L. S. Melnikov. On bounds of the bisection width of cubic graphs. In J. Nesetril and M. Fiedler, editors, *Proceedings of the Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity*, volume 51 of *Annals of Discrete Mathematics*, pages 151–154. Elsevier, 1992.

17. P. Lopez, J. Flich, and J. Duato. Deadlock-free routing in InfiniBand^{TM} through destination renaming. In *Proceedings of ICPP-01*, pages 427–434. IEEE, Sep 2001.

18. D. F. Lugones, D. Franco, and E. Luque. Dynamic routing balancing on infiniband network. *Journal of Computer Science & Technology*, 8(2):104–110, Jul 2008.

19. J. C. Martinez, J. Flich, A. Robles, P. Lopez, and J. Duato. Supporting fully adaptive routing in InfiniBand networks. In *Proceedings of IPDPS-03*. IEEE, Apr 2003.

20. B. D. McKay, M. Miller, and J. Širáň. A note on large graphs of diameter two and given maximum degree. *Journal of Combinatorial Theory, Series B*, 74(1):110–118, Sep 1998.

21. Mellanox Technologies. SwitchX product brief (`http://www.mellanox.com/related-docs/prod_silicon/SwitchX_VPI.pdf`).

22. M. Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, Sep 1994.

23. T. Skeie, O. Lysne, and I. Theiss. Layered shortest path (LASH) routing in irregular system area networks. In *Proceedings of IPDPS-02*, pages 162–169, Apr 2002.

24. H. Subramoni, S. Potluri, K. C. Kandalla, B. Barth, J. Vienne, J. Keasler, K. Tomko, K. Schulz, A. Moody, and D. K. Panda. Design of a scalable InfiniBand topology service to enable network-topology-aware placement of processes. In *Proceedings of SC'12*. IEEE, Nov 2012.

25. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, Apr 1975.

26. A. Thamarakuzhi and J. A. Chandy. 2-dilated flattened butterfly: A nonblocking switching topology for high-radix networks. *Computer Communications*, 34(15):1822–1835, Sep 2011.

27. L. G. Valiant. A scheme for fast parallel communication. *SIAM journal on computing*, 11(2):350–361, May 1982.

28. E. Zahavi, I. Keslassy, and A. Kolodny. Distributed adaptive routing for big-data applications running on data center networks. In *Proceedings of ANCS'12*, pages 99–110. ACM, 2012.

29. E. Zahavi, I. Keslassy, and A. Kolodny. *Distributed Adaptive Routing Convergence to Non-Blocking DCN Routing Assignments*, volume 32, pages 88–101. Jan 2014.