New link arrangements for Dragonfly networks

Madison Belka*, Myra Doubet*, Sofia Meyers*, Rosemary Momoh*, David Rincon-Cruz†§, David P. Bunde*
* *Knox College*
Galesburg, IL 61401
{*mebelka,mrdoubet,semeyers,romomoh,dbunde*}*@knox.edu*
† *Columbia University*
New York, NY 10027
dr2884@columbia.edu

***Abstract*—Dragonfly networks have been proposed to exploit high-radix routers and optical links for high performance computing (HPC) systems. Such networks divide the switches into groups, with a local link between each pair of switches in a group and a global link between each group. Which specific switch serves as the endpoint of each global link is determined by the network's global link arrangement. We propose two new global link arrangements, each designed using intuition of how to optimize bisection bandwidth when global links have high bandwidth relative to local links. Despite this, the new arrangements generally outperform previously-known arrangements for all bandwidth relationships.**

1. Introduction

As the high performance computing (HPC) community first contemplated exascale systems, it was clear that existing network interconnects could not simply be scaled up to add nodes [1]. Such an approach would consume too much power and separate nodes by prohibitively large numbers of hops. A variety of new interconnect topologies were proposed to meet this demand (e.g. [2], [3], [4], [5], [6]), many of them taking advantage of new technologies such as high-radix routers [7], which spread increasing bandwidth over an increasing number of ports, and optical links, which allow messages to travel using fewer hops and less energy.

Of these proposed topologies, Dragonfly [5] has gained the most traction. In a Dragonfly network, computational nodes are attached to network switches, which are organized into groups. Each pair of switches in a group is connected with an electrical *local link*. Each switch also has optical *global links*, which are cheaper and more power efficient for long connections. These are arranged so that every group has a global link to each other group. Figure 1 shows a Dragonfly network.

The size of a Dragonfly network is determined by the number of nodes per switch (p), the number of switches per group (a), and the number of global links per switch (h). The number of groups is $g = ah + 1$. A Dragonfly network for a specific set of parameters is called a (p, a, h)-Dragonfly.

§*Work performed while a student at Knox College.*

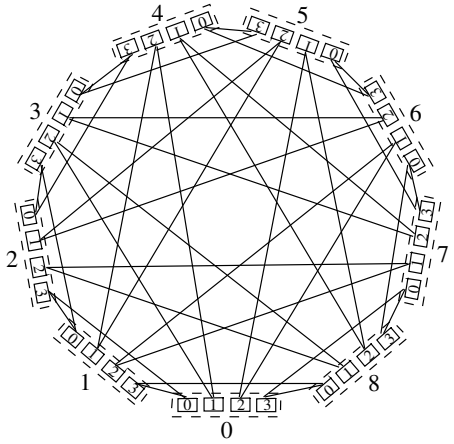

Figure 1. $(p, 4, 2)$-Dragonfly with the relative global link arrangement. The solid boxes are switches and dashed boxes contain the switches of a group; the nodes themselves are not depicted.

The Dragonfly topology is the focus of significant research (e.g. [8], [9], [10]). It has also been adapted for the Cray XC [11], [12] and PERCS [13].

The original description of Dragonfly includes some ambiguity; although it specifies that each pair of groups is connected by a global link, the switches involved are not specified. Camarero et al. [14] noticed this ambiguity, coined the term *global link arrangement* for a way to resolve it, and identified three distinct arrangments.

Hastings et al. [15] evaluated these global link arrangements with bisection bandwidth, a widely-used metric for network topologies that attempts to quantify the performance of a large communication pattern on a system-wide job. It is based on the idea of a *cut*, a partition of the network. The *bandwidth* of a cut is the number of links (more generally, the sum of their weights) crossing the cut (i.e. with an endpoint in each part). The *bisection bandwidth* of a network is the minimum bandwidth of a cut having parts of equal size (within one for odd-sized networks). To allow local and global links to have different bandwidths, these were assigned weights 1 and α respectively.

Hastings et al. [15] computed the bisection band-

width for a $(p, 4, 2)$-Dragonfly. Surprisingly, they found that changing the global link arrangement can improve this metric by as much as 50% without any hardware cost (all arrangements have one optical link between each pair of groups). For arbitrarily-sized networks, they focused on large α and distinguished performance by whether the bisection bandwidth included an α term. This is a potentially-important distinction because Dragonfly implementations have used values of α up to 3.58 [11]. To determine the dependence on α, they identified the network's *global connected components* (GCCs), the connected components formed by only global links if local links are ignored. Then they characterized situations when the network's GCCs are structured so that global links cross any bisecting cut, such as when there are an odd number of equally-sized GCCs.

In this paper, we propose and evaluate two additional global link arrangements. Our aim is to optimize for large α by creating arrangements that form a single GCC, i.e. every pair of switches is connected by a path of global links. This property guarantees that the bandwidth of **every** cut grows with α. By including every cut rather than just bisections, this goal addresses a criticism of the bisection bandwidth metric, that it privileges a specific type of cut. Intutitively, the notation of forming a single GCC is appealing as a way of fully utilizing both types of links. This intuition is somewhat borne out by our results, which show that the new arrangements also provide high bisection bandwidth for small α, where the global links are relatively less important.

Our two new arrangements are significantly different, reflecting the different approaches taken in their discovery. Our nautilus arrangement is the generalization of a small single-GCC network created by trial and error. The helix arrangement was created from scratch to be more symmetric and thus easier to reason about.

We evaluate the new arrangements in two ways. First of all, we computationally determine their bisection bandwidth as a function of α on four specific Dragonfly networks. We find that the new arrangements outperform the previously-known ones at large α. Despite our focus on global edges, however, the new arrangements also match the performance of older ones at small α. The only lagging occurs at intermediate values of α and only on two of the sample networks.

For our second evaluation of the new arrangements, we identify when they are guaranteed to form a single GCC. (Despite being a design goal of the arrangements, this does not occur for all values of the Dragonfly parameters.) Specifically, we show that the nautilus arrangement forms a single GCC when $h > 2$ and either $a \leq h$ or $a = 2h$. We also develop techniques that seem promising to show this property for other $a > h$. For helix, we show that it forms a single GCC when $h \geq 4$.

The rest of this paper is organized as follows. In Section 2, we define the nautilus and helix arrangements. In Section 3, we present the exact bisection bandwidth of our arrangements on the sample Dragonfly networks. In Section 4, we sketch the proofs that our arrangements form a single GCC with appropriate parameter choices. In Section 5, we discuss related work. Finally, in Section 6, we conclude and discuss possible future work.

2. Global link arrangements

To formally specify global link arrangements, we first need some definitions. We name each switch (i, j) according to its group i and its position j within that group. For simplicity, we consider group numbers modulo the number of groups g so that (i, j) and $(i + g, j)$ are the same.

It is also useful to view the switches of a group as forming a single *virtual switch*, connecting that group's nodes to the global links attached to the virtual switches of other groups. In this view, the top-level network is also fully connected, with each virtual switch directly connected to the others. The ports of this virtual switch are then associated with the global links, with the first switch's global links on the first h ports, the second switch's global links on the next h ports, and so on. The local edges become internal structures associated with the virtual switch.

The three previously-known global link arrangements are absolute (aka consecutive), relative (aka palmtree), and circulant [14], [15]. The *absolute arrangement* connects the ports of a virtual switch to each group in order. Thus, port 0 goes to group 0, port 1 goes to group 1, etc. The only exception is to skip the link that would have its source and destination in the same group. Formally, the k^{th} link of switch (i, j) connects to group $jh + k$ if $jh + k < i$ and group $jh + k + 1$ otherwise.

The *relative arrangement* is similar to absolute, but more symmetric because each group behaves like group 0. Specifically, port 0 of group i goes to group $i+1$, port 1 goes to group $i+2$, port 2 goes to group $i+3$, etc. Formally, the k^{th} link of switch (i, j) connects to group $i+jh+k+1 \pmod g$. This is the arrangement used in Figure 1.

The *circulant arrangement* is similar to the relative arrangement except that its links alternate between going to higher-numbered groups and lower-numbered groups. Specifically, port 0 of group i goes to group $i + 1$, port 1 goes to group $i - 1$, port 2 goes to group $i + 2$, port 3 goes to group $i - 2$, etc. Formally, the k^{th} link of switch (i, j) connects to group $i + (\lfloor (jh + k)/2 \rfloor + 1)(-1)^{jh+k}$. This arrangement assumes h is even so that the same number of links go in each direction.

2.1. Nautilus arrangement

The first of our new arrangements is the *nautilus arrangement*. For this arrangement, we divide each group's switches into two categories. The even-numbered switches (i.e. switches (i, j) where j is even) are $+$ switches while the odd-numbered ones are $-$ switches. Links are established from the switches one at a time in the order given by their group and switch number; links are made from switch $(0, 0)$, switch $(0, 1)$, and so on through group 0, followed by the switches in group 1, etc. The number of links made when a switch is visited during this process depends on how many links have already been established to that switch. For a switch (i, j) these links are made to the next groups in either

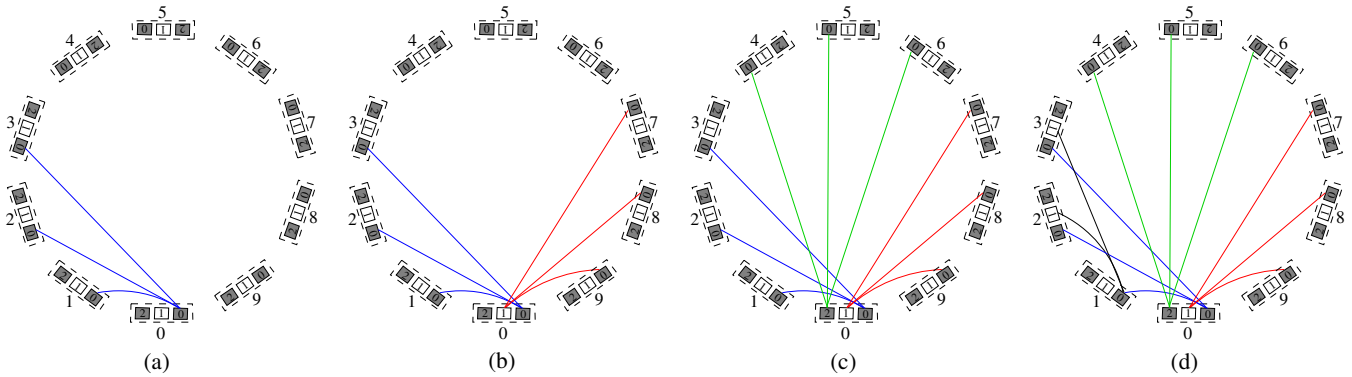

Figure 2. First four steps of nautilus global link arrangement construction for a $(p, 3, 3)$-Dragonfly. Shading distinguishes between $+$ (shaded) and $-$ (unshaded) switches. Parts show the arrangement after the links are made from (a) switch $(0, 0)$, (b) switch $(0, 1)$, (c) switch $(0, 2)$, and (d) switch $(1, 0)$.

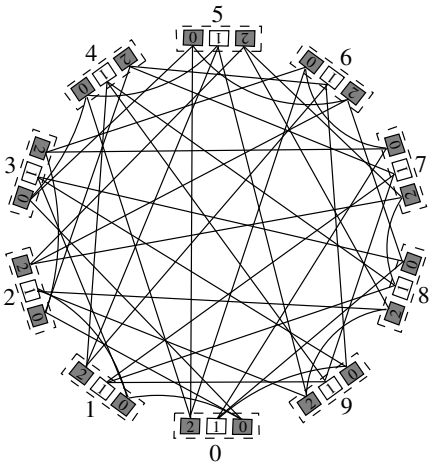

Figure 3. $(p, 3, 3)$-Dragonfly with the nautilus global link arrangement.

the $+$ or $-$ direction which do not already have an edge to group i. Regardless of switch type, all links made from group i go to switch $i \bmod a$.

The first several steps of this construction for a $(p, 3, 3)$-Dragonfly are shown in Figure 2. Parts (a) through (c) of the figure show the links made from group 0 to switch 0s in other groups. First, switch $(0, 0)$ makes links to the next three groups (part (a)) since it is a $+$ switch. Then switch $(0, 1)$, a $-$ switch, makes links to three groups in the $-$ direction (part (b)). Next, switch $(0, 2)$ makes links in the $+$ direction, skipping groups 1, 2, and 3 since these groups are already connected to group 0 (part (c)). Part (d) shows the first links made from group 1 to switch $1 \bmod a = 1$ in the next two groups; only two links are made because switch $(1, 0)$ already has a link from group 0. The completed nautilus arrangement is shown in Figure 3.

When discussing the nautilus arrangement, it is useful to note which switch made each link. A link is called *outgoing* with respect to an incident switch if that switch made the link. Otherwise, it is called *incoming*. The name "nautilus" comes from the number of incoming links in each group, which starts at 0 and grows, suggesting a spiral similar to the shell cross section of the marine cephalopod nautilus.

The main challenge reasoning about the nautilus arrangement is its definition through a sequential construction, with the actions at a given stage depending on previous stages. It is not immediately obvious that the construction even creates a valid global link arrangement, though it does:

Lemma 1. The nautilus construction creates exactly one link between each pair of groups and exactly h links incident on each switch.

Proof: That there is exactly one link between each pair of groups will be implied by there being the correct number of links for each switch since the construction always checks if two groups are connected before joining them with a link.

Each switch will have at least h links because it makes this many when it is visited during the construction. No switch acquires extra links while being visited (by construction) or afterwards (since its group is linked to all others as a result of being visited). Thus, it suffices to show that a switch will not get extra links as incoming links made before it is visited. This cannot happen since the groups visited before the switch are numbered consecutively from 0, there are at most $g - 1 = ah$ of them, and only every a^{th} of them falls into the switch's congruence class. □

The presence of both incoming and outgoing links also complicates determining the groups to which each switch is adjacent without stepping through the construction. We omit the details, but each switch's outgoing links are incident on a contiguous interval of groups, the endpoints of which are given by piecewise functions.

One way to describe the nautilus arrangement is as a reordered version of relative. The first $-$ switch in nautilus is analogous to the group's last switch in relative since going in the $-$ direction is the same as going nearly all the way around in the $+$ direction. The added complexity compared to the relative arrangement comes from the way nautilus uses group numbers to determine the switch number reached by outgoing links. The simpler approach of relative means that it only connects 0^{th} switches with $(a-1)^{\text{st}}$ switches, 1^{st} switches with $(a-2)^{\text{nd}}$ switches, and so on.

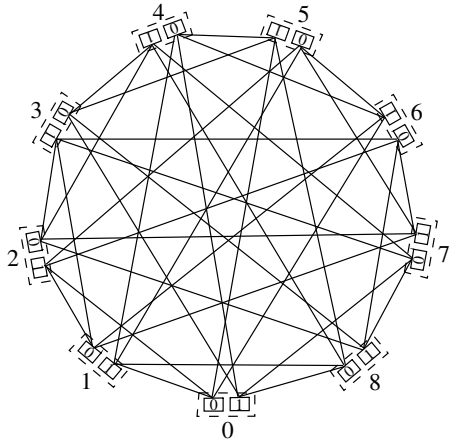

Figure 4. $(p, 2, 4)$-Dragonfly with the helix global link arrangement.

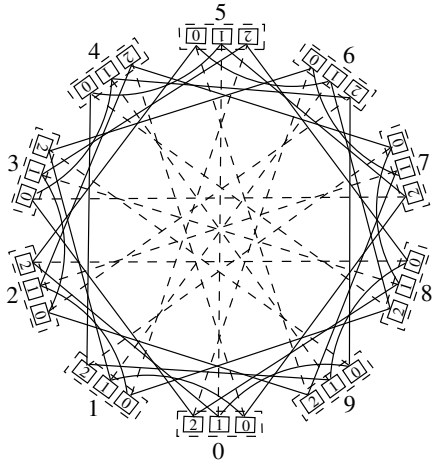

Figure 5. $(p, 3, 3)$-Dragonfly with the helix global link arrangement. The dashed links are the mutual links.

2.2. Helix arrangement

Our second new global link arrangement is the *helix arrangement*. Initially assume that the number of global links per switch h is even; we return to the odd case below. Each switch makes $h/2$ outgoing links similar to the relative arrangement, with the k^{th} such link going to the group numbered $k+1$ higher. Similarly, it receives $h/2$ *incoming* links from $h/2$ lower-numbered groups. The other endpoint of each outgoing link is a switch numbered one higher and of each incoming link is a switch numbered one lower (both $\mod a$). (This sense that each switch is receiving from smaller numbers and sending to larger numbers is the reason for the name helix.) Formally, the k^{th} outgoing link from switch (i, j) goes to switch

$$(i + j \lfloor h/2 \rfloor + k + 1, (j + 1) \bmod a) \tag{1}$$

and its k^{th} incoming link comes from switch

$$(i - 1 - k - j \lfloor h/2 \rfloor, (j - 1) \bmod a). \tag{2}$$

Figure 4 shows the helix arrangement of a $(p, 2, 4)$-Dragonfly.

Although the description above references relative, the helix arrangement also has a strong resemblence to circulant since each switch has $h/2$ links to higher-numbered groups and $h/2$ to lower-numbered groups. The rotation of switch numbers greatly changes which groups the incoming links are from, however; for example, the outgoing links of switch 0 go to the next $h/2$ groups, but its incoming links are from switches numbered $a - 1$ and thus come from groups approximately $(a - 1) \lfloor h/2 \rfloor$ positions back.

For Dragonfly networks with odd h, switches in the helix arrangement make $\lfloor h/2 \rfloor$ outgoing links and receive $\lfloor h/2 \rfloor$ incoming links. We call the remaining link of each switch a *mutual link*. These links go to the a groups between those incident on the group's longest outgoing links (i.e. those of switch $a - 1$) and its longest incoming links (i.e. those of switch 0). Formally, the other endpoint of the mutual link of switch (i, j) is switch

$$(i + a \lfloor h/2 \rfloor + j + 1, a - j - 1). \tag{3}$$

Figure 5 shows the helix arrangement for a $(p, 3, 3)$-Dragonfly, with the mutual links drawn using dashed lines.

Because it has a more symmetric structure than the nautilus arrangement, it is easier to see that the helix construction gives a global link arrangement. Each group has one link to each of the $a \lfloor h/2 \rfloor$ groups immediately following it and the $a \lfloor h/2 \rfloor$ groups immediately preceeding it, provided by the $2 \lfloor h/2 \rfloor$ links at each switch. This suffices when h is even.

When h is odd, we also need to look at the mutual links. The mutual links for group i come from the group reached by neither the outgoing nor the incoming links; note that Equation 3 specifies the same groups as Equations 1 or 2 if the group number j is replaced by a and that each switch gets a single mutual link.

3. Exact bisection bandwidth

We begin evaluating the global link arrangements by looking at their effect on the exact bisection bandwidth for small Dragonfly networks. We are restricted to small networks because determining a graph's bisection bandwidth is NP-complete, with the best known poly-time approximation algorithm solving it to within a factor of $O(\log^2 n)$ [17].

Any cut of the network has bandwidth which is a linear function of α. The bisection bandwidth is then the lower envelope of all the functions corresponding to bisecting cuts. To solve for it, we use a brute force solver to find a min-bandwidth cut for a specific value of α. We first solve for $\alpha = 0$ and a large value of α. This gives two functions that meet at some value of α. We solve for the min-bisection cut at that value of α. If the solution is one of the functions, then those functions fully determine the bisection bandwidth. Otherwise, we have found a new function and can recursively solve for the curve on each side of the intersection point.

Figures 6–9 show the bisection bandwidth as a function of α for four specific Dragonfly networks. The specific

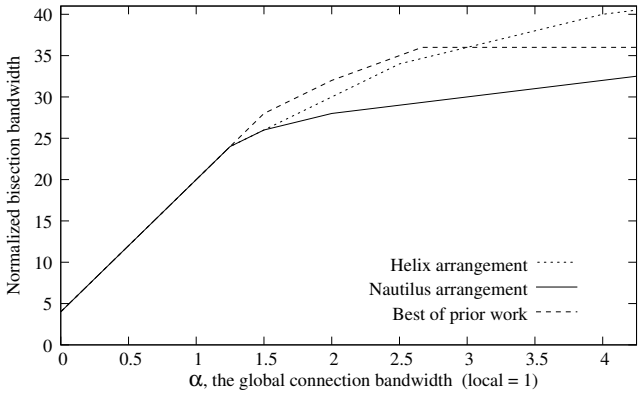

Figure 6. Bisection bandwidth as a function of α for each link arrangement on $(p, 4, 2)$-Dragonfly.

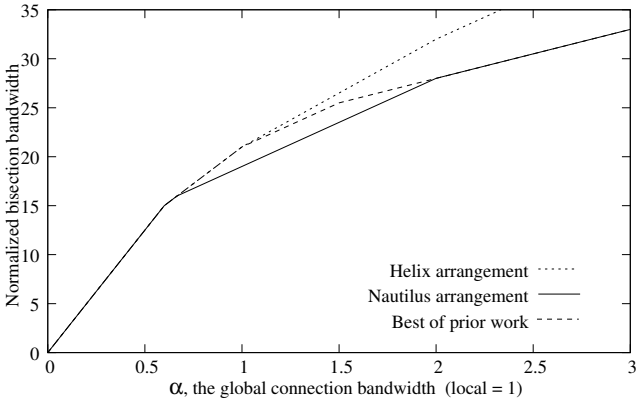

Figure 7. Bisection bandwidth as a function of α for each link arrangement on $(p, 3, 3)$-Dragonfly.

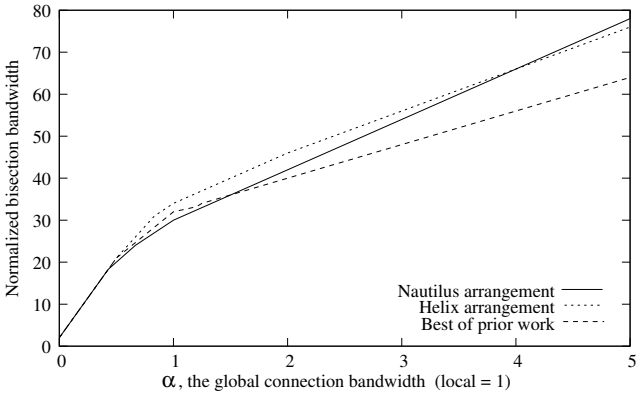

Figure 8. Bisection bandwidth as a function of α for each link arrangement on $(p, 3, 4)$-Dragonfly.

Dragonfly configurations were chosen to provide a variety of networks at the limit of the size for which we could compute the bisection bandwidth in a reasonable time (up to $\approx$ 2 days for a single value of α). Each figure gives the bisection bandwidths for the helix and nautilus arrangements, plus another line showing the pointwise maximum of the absolute, relative, and circulant arrangements. (Figure 7 does not include circulant, which cannot be used since h is odd.)

Figure 6 plots the bisection bandwidth of the $(p, 4, 2)$-Dragonfly. This is a *balanced* network, obeying the suggested relationship $a = 2h$ that spreads the load between the different types of links [5]. In this figure, the "best of" line is always the circulant arrangement, which dominates relative and absolute. Which line gives the highest bisection bandwidth varies as a function of α. The helix, nautilus, and circulant arrangements are in a three-way tie until $\alpha = 1.25$, at which point circulant actually beats the other two until $\alpha = 3$. From that point on, helix dominates. The nautilus arrangement is never the strictly best, though it does dominate circulant for large (> 6) values of α; this happens because the circulant arrangement never gives bisection bandwidth greater than 36 while nautilus continues to improve.

Figure 7 plots the bisection bandwidth of the $(p, 3, 3)$-Dragonfly. For this size, the "best of" line corresponds to the relative arrangement, whose bisection bandwidth is always at least as high as the absolute arrangement; recall that circulant cannot be used because h is odd. Looking at the figure, we see a three-way tie until $\alpha = 2/3$, with the tie between the helix and relative arrangements continuing until $\alpha = 1$. After this, the helix arrangement is always strictly better than the others. The tie between the relative and nautilus arrangements resumes at $\alpha = 2$.

Figure 8 plots the bisection bandwidth of the $(p, 3, 4)$-Dragonfly. The "best of" line depicts the circulant arrangement until $\alpha = 1.2$, after which the relative arrangement is better. (The two tie until $\alpha = 3/7$.) The relationshiop between the three lines is complicated. They are tied until $\alpha = 0.5$, after which the helix arrangement always gives a higher bisection bandwidth than the "best of". The nautilus arrangement temporarily falls into third place at $\alpha = 0.5$, but gradually comes from behind, passing the mixed "best of" arrangement at $\alpha = 1.5$ and giving the highest overall bisection bandwidth for $\alpha \geq 4$.

Figure 9 plots the bisection bandwidth of the $(p, 2, 8)$-Dragonfly. For this size network, the helix, nautilus, and relative arrangements all give the same bisection bandwidth despite their different graphs. This makes helix and nautilus tied with the best of prior work for $\alpha \leq 1/7$ and $\alpha \geq 0.3$ since the relative arrangement is the best prior algorithm for these ranges. Between $\alpha = 1/7$ and $\alpha = 0.3$, however, the circulant arrangement is slightly better.

Looking across all four of the specific sizes for which we solved the entire bisection bandwidth function, our new arrangements perform generally well. Either the helix or nautilus arrangement provides the highest bisection bandwidth except for intermediate values of α on the $(p, 4, 2)$-Dragonfly and the $(p, 2, 8)$-Dragonfly. The helix/nautilus value is strictly higher for large values of α one all except the $(p, 2, 8)$-Dragonfly, where it achieves a tie.

When comparing our two new arrangements, the helix arrangement is generally better except for large α values on the $(p, 3, 4)$-Dragonfly. Even the nautilus arrangement does

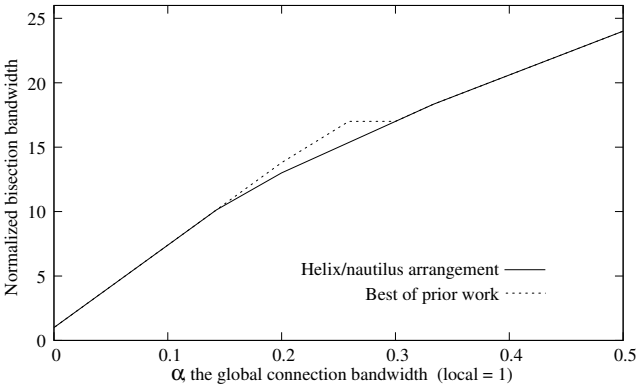

Figure 9. Bisection bandwidth as a function of α for each link arrangement on $(p, 2, 8)$-Dragonfly. Helix and nautilus differ, but tie for all α.

well against the previously-known arrangements however; its relative performance to them can also be summarized as a tie at low values of α, giving higher bisection bandwidth at high values of α, and sometimes giving a slightly lower bisection bandwidth at intermediate values of α. We take these results as a validation of our approach; despite focusing on high α by creating arrangements that form a single GCC, the resulting arrangements are competive at lower values of α as well.

As a final comment on the results in this section, we observe that the nautilus arrangement does not create a single GCC on the $(p, 4, 2)$-Dragonfly. The arrangement on this network has multiple GCCs, including one with only 3 switches; note that the theorems in the next section concerning the nautilus arrangement require $h > 2$. The arrangement on this network does provide increasing bisection bandwidth as α increases, but that is because the GCCs cannot be split evenly rather than there being only one of them.

The theorem in the next section concerning the helix arrangement requires $h \geq 4$ so it also does not show that the arrangement forms a single GCC for the $(p, 4, 2)$-Dragonfly or the $(p, 3, 3)$-Dragonfly. In these cases, however, the arrangement does form a single GCC anyway.

4. Proofs of 1 GCC

Now we evaluate the new arrangments based on our original design goal of creating a single GCC.

4.1. Nautilus arrangement

We begin with a couple of tools to help manage the complexity of the nautilus arrangement. We call a switch with only outgoing links *self-determined* and one with only incoming links *pre-determined*.

Because of how the construction procedure adds links, there is a simple characterization for the location of self-determined switches.

Lemma 2. In the nautilus arrangement, a switch (i, j) is self-determined iff $j \geq i$.

Proof: When a group i is visited, an edge is added to switch $i \bmod a$ in every higher-numbered group, meaning those switches are not self-determined. Thus, all switches in group 0 are self-determined because this group is visited first, but no other switch 0s are self-determined. Similarly, switches 1 through $a - 1$ of group 1 are self-determined, but switch 1 is not self-determined in any high-numbered group. Continuing this process shows that only the first a groups have self-determined and the only such switches are those whose number is $\geq$ their group number. □

One reason the concept of self-determined switches is useful is that the other switches are connected to them.

Lemma 3. If the nautilus arrangement connects all self-determined switches, then all switches form a single GCC.

Proof: The self-determined switches being connected is enough because all are reachable from them. Any other switch has an incoming link, which goes to a switch first touched earlier in network construction. Such a sequence of links can not continue indefinitely since the visiting order is a total order on a finite set. □

Analogous arguments give equivalent results for pre-determined switches.

Lemma 4. In the nautilus arrangement, a switch (i, j) is pre-determined iff $j < a - (ah - i)$.

Lemma 5. If the nautilus arrangement connects all pre-determined switches, then all switches form a single GCC.

With these tools, we are ready to identify cases when the nautilus arrangement creates a single GCC. We give three different proofs depending on the relationship between a (the number of switches in a group) and h (the number of global links per switch). All of these proofs use the assumption $h > 2$. The nautilus arrangement does not form a single GCC for many sizes with $h = 2$; doing so would require that all the switches form a single ring.

We sketch our arguments in order of a, beginning with the case $a < h$.

Theorem 6. The nautilus global link arrangement connects all switches into a single GCC when $a < h$ and $h > 2$.

Sketch: The main case concerns groups $G = \{a, \ldots, 2a - 1\}$. Switch $(0, 0)$ can reach the 0 switches of G in two hops via switch $(a, 0)$. It can also reach each $j \neq 0$ switch of G in either two hops via $(j, 0)$ or three hops via $(a, 0)$ and $(a + j, 0)$.

From G, we can reach all switches in higher-numbered groups; switch (i, j) with $i \geq 2a$ is reached by an outgoing link from group $a + j \in G$ since every pair of groups is connected and group $a + j$'s only incoming links are from lower-numbered groups.

For lower-numbered groups, any switch which is not self-determined (i.e. (i, j) with $j < i < a$) can be reached from $(0, 0)$ in two hops via switch $(j, 0)$. Self-determined switches are then also reachable because their links go to non-self-determined switches, all of which are reachable. □

Now we present the case $a = h$.

Theorem 7. The nautilus global link arrangement connects all switches into a single GCC when $a = h$ and $h > 2$.

Sketch: The main idea is to show that all the self-determined switches are connected and then apply Lemma 3.

Observe that all of group 0 is connected since switches $(0, j)$ and $(0, j+1)$ are connected via $((j+1)a, 0)$ and $((j+1)a + 1, 0)$. Since every zero switch (i.e. $(i, 0)$) is adjacent to a switch in group 0, all zero switches are connected.

Each self-determined switch has outgoing links to $a = h$ contiguous groups, one of which has a group number congruent to 0. If this group is not ah, reaching that a switch in that group suffices since it has an outgoing link to a zero switch. To show that all of group ah is connected to a zero switch, observe that (ah, j) with $j \neq 0$ is connected to $(i, 1)$ for some $a + 1 \leq i \leq 2a - 1$, all of which are connected to $(0, 0)$ via $(1, 2)$ and $(2a, 1)$. □

The remaining case is $a > h$. We were not able to show that the nautilus arrangement always creates a single GCC in this case, but did show that it does so when $a = 2h$. This is a particularly important subcase since it represents a balanced network.

Theorem 8. The nautilus global link arrangement connects all switches into a single GCC when $a = 2h$ and $h > 2$.

Sketch: The argument is more complicated, but in many ways it is the mirror image of the proof of Theorem 7. We show that all pre-determined switches are connected and apply Lemma 5. The 1 switches (i.e. $(i, 1)$) are connected because switches of group 1 connect intervals of them, which are joined by the switches $(ah, 0)$, (ah, h), and $(a + 1, 2)$. Since every switch of group ah is adjacent to a 1 switch, this means group ah is connected.

Our final step is to show that all pre-determined switches of each switch number are connected. The pre-determined j switches are connected via $(j, 1)$, $(j, 3)$, and $(a + j, 3)$. □

We conjecture that the nautilus arrangement forms a single GCC whenever $a > h$; this was computationally verified for $3 \leq h \leq 9$ and $h + 1 \leq a \leq 500$. The hard part of generalizing the proof of Theorem 8 seems to be showing that all 1 switches are connected.

A recurring theme in all the proofs in this section is finding the right switches to avoid the complexity of the nautilus construction. Certain switches seem generally convenient for this purpose. Groups 0 and ah get around the mixture of incoming and outgoing edges by having only one type. Switch 0s and 1s are convenient for working with groups 0 and ah respectively since they are connected. The groups a through $2a-1$ were useful in the proof of Theorem 6 since each of their switches has exactly one incoming link from outside this range; other intervals of a groups could potentially be used in a similar way.

4.2. Helix arrangement

The symmetry of the helix arrangement makes it much easier to reason about. We are able to show that (when $h \geq 4$) it forms a single GCC using a short proof with one case.

Theorem 9. The helix global link arrangement connects all switches into a single GCC when $h \geq 4$.

Proof: Any switch $(i, 0)$ can reach switch $(i + 1, 0)$ via a sequence of two hops, taking the second outgoing link (hence the need for $h \geq 4$) to switch $(i+2, 1)$ and then that switches incoming link to switch $(i+1, 0)$. Thus, all switches with switch index 0 are connected. Since any switch (i, j) can reach a switch with index 0 via a sequence of j arbitrary incoming links, all switches are connected. □

We also conjecture that the helix arrangement also forms a single GCC when $h = 2$ and $h = 3$. We programmatically verified that it does for a up to 200 in both cases.

5. Related work

To our knowledge, the only prior work directly looking at different global link arrangements for Dragonfly networks is by Camarero et al. [14] and Hastings et al. [15]. Camarero et al. defined the three previously-known link arrangements as part of exploration of Dragonfly variations that use trunking (multiple links between groups). Despite defining the global link arrangements, they did not assign any significance to this difference. The focus of the paper is on the relationship between Dragonfly and Hamming graphs, which allows the importation of routing algorithms designed for Hamming graphs to Dragonfly networks. Hastings et al. [15] showed that the three arrangements gave up to a 50% different bisection bandwidth on a $(p, 4, 2)$-Dragonfly using the same technique we used in Section 3. They also characterized the GCC structure of each arrangement.

Alverson et al. [11] previously computed the bisection bandwidth of Dragonfly networks using the graph where each vertex is a virtual router that cannot be split. This implicitly assumes that local links have unbounded bandwidth, which leads to the conclusion that all global link arrangements are equivalent.

Beyond Camarero et al. [14] mentioned above, there is quite a body of work on routing for Dragonfly networks (e.g. [8], [18], [19]). These papers implicitly choose a global link arrangement (often relative), but the algorithms seem to be independent of the arrangement selected.

Others have attempted to improve the performance of Dragonfly systems with block-based mapping schemes to distribute tasks to nodes (e.g. [20], [21], [22]) or scheduling intra-job communication [23].

6. Conclusion and Discussion

Our results reinforce the surprising point that such a tiny detail as the global link arrangement actually matters for network performance. It is also interesting that nautilus and helix generally outperform the previously-known arrangements for all values of α despite being designed specifically for large α. Forming a single GCC seems to be a reasonable proxy for the intuitive notion of "well-connected".

The helix arrangement is particularly interesting since it has the dual benefits of being easier to describe and

use in proofs as well as generally offering higher bisection bandwidth. Its ease of use comes from the symmetry that we made an explicit design goal after running into the complexity of the nautilus arrangement. We wonder if this symmetry also leads to its superior bisection bandwidth. This metric can be thought of as a game against an adversary trying to find a weakness in the network. Network asymmetries present the adversary with alternative lines of attack whereas symmetric networks limit their choices. In support of this idea is that the best performing global link arrangements are all the symmetric ones (helix, relative, and circulant). Of course the outperformance of nautilus on the $(p, 3, 4)$-Dragonfly shows that reality is more complicated than this simple idea, but symmetry does seem to generally help.

There remain plentiful opportunities for future research. Theorems 8 and 9 leave open the questions of whether a single GCC is always formed by the nautilus arrangement when $a > h$ and the helix arrangement when h is 2 or 3.

A larger project would be to determine the bisection bandwidth of nautilus or helix on a general Dragonfly network. Forming a single GCC means that bisection bandwidth is grows linearly with α, but gives no indication of the coefficients. Hastings et al. [15] solved the bisection bandwidth for large α more precisely when the network had multiple GCCs by counting the number of local edges that must be cut to separate them. Greater understanding of the structure of the single GCC may make similar results possible, particularly for the helix arrangement. It would also be interesting to compute bisection bandwidth for specific values of α (i.e. not just "large α").

Acknowledgments

We thank the anonymous reviewers for their comments and suggestions. This work was partially supported by the National Science Foundation under grant CNS-1423413 and the Paul K. and Evalyn Elizabeth Richter Memorial Funds.

References

[1] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. Williams, and K. Yelick, "Exascale computing study: Technology challenges in achieving exascale systems," tech. rep., 2008.

[2] S. Scott, D. Abts, J. Kim, and W. Dally, "The BlackWidow high-radix Clos network," in *Proc. 33rd Ann. Intern. Symp. Comput. Arch. (ISCA)*, pp. 16–28, 2006.

[3] D. Abts, A. Bataineh, S. Scott, G. Faanes, J. Schwarzmeier, E. Lundberg, T. Johnson, M. Bye, and G. Schwoerer, "The Cray BlackWidow: A highly scalable vector multiprocessor," in *Proc. ACM/IEEE Conf. High Performance Networking and Computing (SC)*, 2007.

[4] J. Kim, W. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proc. 34th Ann. Intern. Symp. Comput. Arch. (ISCA)*, pp. 126–137, 2007.

[5] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proc. 35th Ann. Intern. Symp. Comput. Arch. (ISCA)*, pp. 77–78, 2008.

[6] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[7] J. Kim, W. Dally, B. Towles, and A. Gupta, "Microarchitecture of a high-radix router," in *Proc. 32nd Ann. Intern. Symp. Comput. Arch. (ISCA)*, pp. 420–431, 2005.

[8] P. Yébenes, J. Escudero-Sahuquillo, P. García, and F. Quiles, "Efficient queuing schemes for hol-blocking reduction in dragonfly topologies with minimal-path routing," in *Proc. 1st Intern. Workshop High-Performance Interconnection Networks Towards the Exascale and Big-Data Era*, 2015.

[9] B. Prisacari, G. Rodrguez, C. Minkenberg, M. Garcia, E. Vallejo, and R. Beivide, "Performance optimization of load imbalanced workloads in large scale dragonfly systems," in *Proc. IEEE 16th Intern. Conf. High Performance Switching and Routing*, 2015.

[10] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in Dragonfly-based supercomputers," in *Proc. 30th IEEE Intern. Parallel and Distributed Processing Symp. (IPDPS)*, 2016.

[11] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC series network," white paper, Cray, Inc., 2012.

[12] 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 *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, p. 103, 2012.

[13] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony, "The PERCS high-performance interconnect," in *Proc. 18th Symp. High-Performance Interconnects (Hot Interconnects)*, 2010.

[14] C. Camarero, E. Vallejo, and R. Beivide, "Topological characterization of Hamming and Dragonfly networks and its implications on routing," *ACM Trans. Architec. Code Optim.*, vol. 11, no. 4, p. 39, 2014.

[15] E. Hastings, D. Rincon-Cruz, M. Spehlmann, S. Meyers, A. Xu, D. Bunde, and V. Leung, "Comparing global link arrangements for Dragonfly networks," in *Proc. IEEE Cluster*, pp. 361–370, 2015.

[16] S. A. Jyothi, A. Singla, P. Godfrey, and A. Kolla, "Measuring throughput of data center network topologies," in *Proc. 2014 ACM Intern. Conf. Measurement and modeling of computer systems (SIGMETRICS)*, pp. 597–598, 2014.

[17] U. Feige and R. Krauthgamer, "A polylogarithmic approximation of the minimum bisection," *SIAM J. Comput.*, vol. 31, no. 4, pp. 1090–1118, 2002.

[18] N. Jiang, J. Kim, and W. Dally, "Indirect adaptive routing on large scale interconnection networks," in *Proc. 36th Ann. Intern. Symp. Comput. Arch. (ISCA)*, pp. 220–231, 2009.

[19] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodriguez, J. Labarta, and C. Minkenberg, "On-the-fly adaptive routing in high-radix hierarchical networks," in *Proc. 41st Intern. Conf. Parallel Processing (ICPP)*, pp. 279–288, 2012.

[20] A. Bhatele, N. Jain, W. Gropp, and L. Kale, "Avoiding hot-spots on two-level direct networks," in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.

[21] V. Chakaravarthy, M. Kedia, Y. Sabharwal, N. Katta, R. Rajamony, and A. Ramanan, "Mapping strategies for the PERCS architecture," in *Proc. 19th Intern. Conf. High Performance Computing (HiPC)*, 2012.

[22] N. Jain, A. Bhatele, X. Ni, N. Wright, and L. Kale, "Maximizing throughput on a dragonfly network," in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[23] B. Prisacari, G. Rodriguez, and C. Minkenberg, "Generalized hierarchical all-to-all exchange patterns," in *Proc. 27th IEEE Intern. Parallel and Distributed Processing Symp. (IPDPS)*, 2013.