mathematics

MDPI

Article

All-to-All Broadcast Algorithm in Galaxyfly Networks †

Hongbin Zhuang [1], Jou-Ming Chang [2], Xiao-Yan Li [1,*], Fangying Song [3] and Qinying Lin [1]

[1] College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China
[2] Institute of Information and Decision Sciences, National Taipei University of Business, Taipei 10051, Taiwan
[3] School of Mathematics and Statistics, Fuzhou University, Fuzhou 350108, China
* Correspondence: xyli@fzu.edu.cn

† This paper is an extended version of our paper published in Lin, Q.; Zhuang, H.; Li, X.-Y.; Su, L. All-to-all routing algorithm for Galaxyfly networks. In Proceedings of the 2022 IEEE 24th International Conference on High Performance Computing and Communications, Chengdu, China, 18–21 December 2022; pp. 888–893.

Abstract: The design of interconnection networks is a fundamental aspect of high-performance computing (HPC) systems. Among the available topologies, the Galaxyfly network stands out as a low-diameter and flexible-radix network for HPC applications. Given the paramount importance of collective communication in HPC performance, in this paper, we present two different all-to-all broadcast algorithms for the Galaxyfly network, which adhere to the supernode-first rule and the router-first rule, respectively. Our performance evaluation validates their effectiveness and shows that the first algorithm has a higher degree of utilization of network channels, and that the second algorithm can significantly reduce the average time for routers to collect packets from the supernode.

Keywords: Galaxyfly network; all-to-all broadcast; interconnection network; algorithm

MSC: 68M10; 68R10

Citation: Zhuang, H.; Chang, J.-M.; Li, X.-Y.; Song, F.; Lin, Q. All-to-All Broadcast Algorithm in Galaxyfly Networks. *Mathematics* **2023**, *11*, 2459. https://doi.org/10.3390/math11112459

Academic Editor: Ke Li

Received: 25 March 2023
Revised: 14 May 2023
Accepted: 23 May 2023
Published: 26 May 2023

1. Introduction

Large-scale supercomputers are essential for tackling complex scientific and industrial challenges [1,2]. To meet the growing demand for computing power, exascale systems are being developed, incorporating an increasing number of interconnected processors [3]. The current fastest supercomputer, Frontier [4], comprises 8,730,112 cores and delivers a peak performance of 1102.00 Pflop/s. The design of interconnection networks is crucial for achieving scalability in high-performance computing (HPC) systems. Topology is a critical design factor in interconnection networks, determining performance bounds [5] such as end-to-end latency and bisection bandwidth, as well as network costs.

In order to fabricate large-scale supercomputers, the development of efficient network topologies is imperative and should take into account a range of metrics. Primarily, high bandwidth and low latency are considered as the fundamental objectives for optimizing network topologies as they are pivotal in facilitating faster and smoother data communication [6]. Second, the total power consumption of supercomputers must be kept within practical limitations. In the case of exascale systems, for instance, a power envelope of 20–30 MW must be strictly adhered to in order to ensure low power consumption [7]. Third, the excellent network design should endow HPC systems with high flexibility so that HPC systems possess the ability to vary on different scales [8].

Considerable research efforts have been devoted to investigating a range of efficient network topologies for HPC systems, including Flattened Butterfly [9–11], Dragonfly [12–14], HyperX [15], Skywalk [16], and SlimFly [17]. These structures are capable of delivering low diameters for HPC systems while also ensuring scalability through the port numbers (radixes) of the construction blocks (routers) [18]. However, the radixes of routers in these topologies must unavoidably increase to meet the demands of existing high-radix topologies in exascale systems and beyond [19]. This poses a significant challenge for

high-performance routers, particularly commercial-off-the-shelf (COTS) routers, to expand their radix, resulting in major hurdles for existing topologies to scale flexibly. To address this issue, Lei et al. [20] introduced the Galaxyfly network, a flexible-radix low-diameter topology that achieves network flexibility under different configuration structures by reducing high-radix routers. The Galaxyfly network is composed of lots of supernodes, with routers within each supernode being fully connected. Notably, Dragonfly networks are a special case of Galaxyfly networks with only one supernode.

Collective communication is quite vital for interconnection networks [21]. Designing efficient routing algorithms for collective communication poses significant challenges due to high network bandwidth demands. Collective communication is typically achieved through unicast-based [22,23], tree-based [24,25], and path-based [26,27] approaches. McKinley et al. [28] proposed a multicast routing algorithm for wormhole-routed meshes and hypercubes without the technical support of hardware switching. Using the message combination, Suh and Yalamanchili [29] proposed an algorithm for minimizing message startups at the cost of larger message sizes. Juurlink et al. [30] optimized the trade-off between contributions owing to startups and those owing to the bounded capacity of the connections. At present, all-to-all broadcast algorithms for the Galaxyfly network have only been studied in the context of Dragonfly networks, highlighting the need for a dependable and efficient algorithm specifically designed for the Galaxyfly network.

In this paper, we propose two all-to-all broadcast algorithms to mitigate high bandwidth problems and routing inefficiencies when routers communicate collectively. Our approach involves the proposal of four fundamental algorithms for packet collection and distribution across distinct scenarios, followed by the development of two all-to-all broadcast algorithms. Our first algorithm prioritizes data delivery to the supernodes during the routing process, such that the network resources can be efficiently utilized. The second algorithm ensures the timely delivery of packets to other routers within the same supernode. In summary, the main contributions of this paper are listed as follows:

- We design four basic routing algorithms by utilizing recursion and breadth-first traversal.
- Based on basic routing algorithms, we propose two different all-to-all broadcast algorithms. The supernode-first all-to-all broadcast algorithm can achieve a higher utilization of network resources used in the routing process, while the router-first all-to-all broadcast algorithm delivers packets to all routers within the same supernode in priority.
- We implement the proposed algorithms to show their effectiveness and performance.

The rest of this paper is organized as follows. Section 2 describes the definition of the Galaxy graph and introduces the basic concepts of the Galaxyfly network. Section 3 provides four basic algorithms for two all-to-all broadcast algorithms. Section 4 demonstrates two all-to-all broadcast algorithms by utilizing four basic algorithms. Section 5 gives the experimental results. Section 6 concludes this paper.

2. Preliminaries

For ease of reading, please refer to Table 1 below, which contains most of the important notations used in this paper.

Table 1. Notations.

Symbol	Meaning
n	Number of clusters in Galaxyfly network.
q	Number of supernodes per cluster.
a	Number of routers per supernode.
p	Number of links per router to terminals.
h	Number of links per router to other supernodes.
S_i	The supernode S_i.
R_i	The router R_i.
U	The set of all supernodes.
$\|X\|$	Number of elements in the set X.

2.1. Galaxy Graph

Lei et al. [20] proposed Galaxy graphs, which have a diameter of at most, 2. Galaxy graphs are designed to build flexible sized networks for a given number of routers and radii. A Galaxy graph is a graph $G(V, E)$ of size $n \times q$ and is divided equally into n clusters, $V_0, V_1, V_2, \ldots, V_{n-1}$. Each cluster has q supernodes; q is a prime number with $q = 4\ell + \delta$, where $\ell \in \mathbb{N}$ and $\delta \in \{-1, 0, 1\}$.

There are two types of edges in the Galaxy graph:

- **Intra-cluster edges:** The construction of intra-cluster edges requires some basic knowledge on algebraic graphs over finite fields. We present only the basic results necessary to understand the construction process. Intra-cluster edges connecting nodes in the same cluster are constructed as in Definition 1.
- **Inter-cluster edges:** The construction of inter-cluster edges is determined by an isomorphic function H. First, we introduce a definition on the generator graphs (i.e., Theorem 1), then we introduce the definition of the isomorphic function H and a theorem on it (i.e., Definition 2 and Theorem 2). Inter-cluster edges connect nodes in two different clusters, constructed as in Theorem 1, Definition 2, and Theorem 2.

There are q nodes in a cluster. q generates a finite field, $\mathbb{F}_q = \{x_0, x_1, \ldots, x_{q-1}\}$. There exists a primitive element $\xi \in \mathbb{F}_q$, which generates $\mathbb{F}_q$ in the form of $\mathbb{F}_q = \{\xi^t \mod q | t \in \mathbb{N}\} \cup \{0\}$. Two generator sets, X and X', are constructed as Equations (1) and (2). All arithmetic operations are performed in a modulo q manner. Note that $|X| = |X'| = (q - \delta)/2$ and $X \cup X' = \mathbb{F}_q - \{0\}$.

$$X = \begin{cases} \{1, \xi^2, \ldots, \xi^{q-3}\} & q = 4\ell + 1 \\ \{1, \xi^2, \xi^4, \ldots, \xi^{2\ell-2}, \\ \xi^{2\ell-1}, \xi^{2\ell+1}, \ldots, \xi^{4\ell-3}\} & q = 4\ell - 1 \\ \{1, \xi^2, \ldots, \xi^{4\ell-2}\} & q = 4\ell. \end{cases} \tag{1}$$

$$X' = \begin{cases} \{\xi, \xi^3, \ldots, \xi^{q-2}\} & q = 4\ell + 1 \\ \{\xi, \xi^3, \xi^5, \ldots, \xi^{2\ell-1}, \\ \xi^{2\ell}, \ldots, \xi^{4\ell-4}, \xi^{4\ell-2}\} & q = 4\ell - 1 \\ \{\xi, \xi^3, \ldots, \xi^{4\ell-1}\} & q = 4\ell. \end{cases} \tag{2}$$

Definition 1 (see [20]). *A generator graph of $\hat{X} \in \{X, X'\}$ is a q node graph with nodes labeled $\{x_0, x_1, \ldots, x_{q-1}\}$, and there exists an intra-cluster edge between node x_i and x_j if $x_i - x_j \in \hat{X}$.*

Theorem 1 (see [31]). *If G, G' are generator graphs of X and X', respectively, then G and G' are isomorphic graphs.*

Definition 2 (see [20]). *An isomorphic function H is a one-to-one function defined on $\mathbb{F}_q = \{x_0, x_1, \ldots, x_{q-1}\}$ such that for any $\forall x_i \in \mathbb{F}_q$, x_i in G and $H(x_i)$ in G' are corresponding nodes in terms of isomorphism.*

Theorem 2 (see [20]). *For $x_i, x_j \in \mathbb{F}_q$, $x_i - x_j \in X$ if and only if $H(x_i) - H(x_j) \in X'$. The inter-cluster edges between any two clusters V_s and V_t $(0 \leq s < t < n)$ are constructed as follows: for each node x_i $(i = 0, 1, \ldots, q-1)$ in V_t, connect it to node $H(x_i)$ in V_s.*

Figure 1 shows the global connection between each pair of supernodes of a Galaxy graph with $n = 3$ clusters, each of which contains $q = 5$ supernodes. In Figure 1, we can observe that $q = 5$, $\delta = \ell = 1$, $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$, and $\xi = 2$. That is, each cluster contains the nodes 0–4. In order to distinguish the nodes between different clusters, we replace the identification of each node with S_i with $i \in \{1, 2, \ldots, 15\}$.

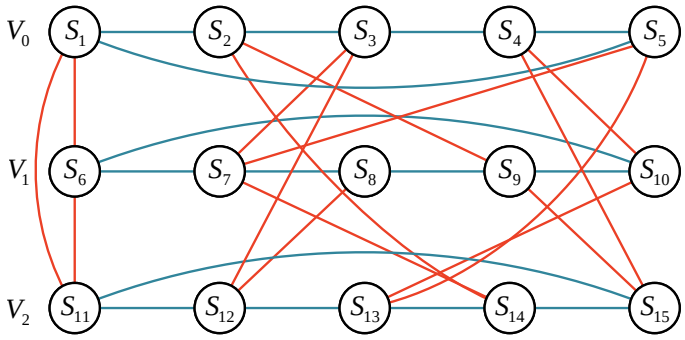

Figure 1. Global links of the Galaxy graph with $n = 3, q = 5$.

2.2. Galaxyfly Network

The Galaxyfly network is constructed by replacing each node of the Galaxy graph with a supernode. The way in which the routers are connected affects the performance of the network. For routers within a supernode, a full connection is used, and for global links, the same scheme as in [32] is used to connect the Galaxyfly network global links. Galaxyfly is defined by five parameters (n, q, a, p, h). The Galaxyfly network contains n clusters; each cluster contains q supernodes and each supernode contains a routers. Thus, there are a total of $n \times q \times a$ routers. Each router is connected to p terminals, so that the network size is $n \times q \times a \times p$.

The concepts of supernodes and clusters in Galaxyfly networks are similar to those of supernodes in Dragonfly and subgroups in SlimFly. This hierarchical structure allows Galaxyfly to match well for the traffic characteristics of various HPC applications. The adjustable parameters n, q and a enable the Galaxyfly network to customize the network size and bifurcation bandwidth.

Figure 2 shows a Galaxyfly network with three clusters, each of which has five supernodes. Each supernode is a fully connected graph of four routers.

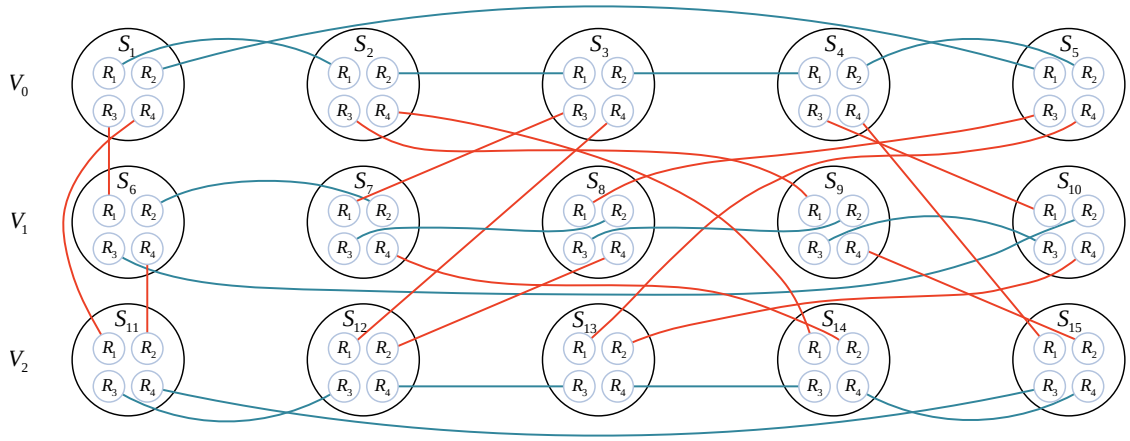

Figure 2. Global links in Galaxyfly networks with $n = 3, q = 5, a = 4, h = 1$.

3. Basic Algorithm

For the Galaxyfly networks, we propose four algorithms to implement the collection and distribution of packets in four different scenarios. All-to-all broadcast algorithms proposed in Section 4 will frequently call these four basic algorithms.

The basic algorithms include Router-Packet-Collection, Router-Packet-Distribution, Supernode-Packet-Collection, and Supernode-Packet-Distribution.

3.1. Algorithms: Router-Packet-Collection (RPC) and Router-Packet-Distribution (RPD)

The algorithms Router-Packet-Collection (i.e., Algorithm 1) and Router-Packet-Distribution implement the core algorithm of our proposed broadcast algorithm to collect and distribute packets within the supernode via recursion. Next, we first introduce the algorithm Router-Packet-Collection.

Algorithm 1: Router-Packet-Collection (RPC).

Input: A: the set of routers or a supernode;
m: a router in A;
Output: Packets from all routers within A are collected into the router m;
if A is a supernode **then** Let A be the set of routers in the supernode A;
Let $B = A \setminus \{m\}$;
Divide B into two subsets B_1 and B_2 such that $|B_1| = \lceil |B|/2 \rceil$, $|B_2| = \lfloor |B|/2 \rfloor$;
Let b_1 be an arbitrary router in B_1;
Let b_2 be an arbitrary router in B_2;
if $|B_1| = 1$ **then** Send packets of router b_1 in B_1 to the target router m;
else
  RPC(B_1, b_1);
  Send packets of router b_1 in B_1 to the target router m;
if $|B_2| = 1$ **then** Send packets of router b_2 in B_2 to the target router m;
else
  RPC(B_2, b_2);
  Send packets of router b_2 in B_2 to the target router m;
return;

Theorem 3. *In the RPC algorithm, the packets are accurately sent to the target router.*

In PRC, all routers in a supernode are divided into two sets, B_1 and B_2, for executing RPC(B_1, b_1) and RPC(B_2, b_2), respectively. If the number of routers in B_1 or B_2 is greater than 1, then the algorithm will recursively execute RPC to collect all packets in set B_1 into router b_1, send all packets in set B_2 into router b_2, and send the packets of b_1 and b_2 into router m. Thus, the router m will receive all the packets in the supernode A. The algorithm

solves the problem by decomposing the original problem into two subproblems, and thus, the time complexity of the algorithm is $O(\log_2 a)$ (see the definition of a in Table 1).

Figure 3 shows the execution process of the RPC algorithm in the Galaxyfly network. R_1–R_8 in the figure indicates the routers within the supernode. The specific detailed process is as follows:

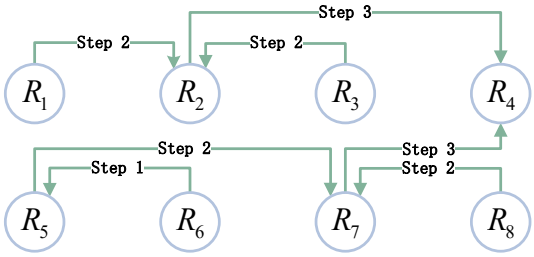

Figure 3. Collect packets from each router inside a supernode using the RPC algorithm in Galaxyfly networks.

The executing RPC$(\{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8\}, R_4)$ will send the packets from R_2 and R_7 to R_4. This process will call RPC$(\{R_1, R_2, R_3\}, R_2)$ and RPC$(\{R_5, R_6, R_7, R_8\}, R_7)$ to collect packets to R_2 and R_7. The result of executing RPC$(\{R_1, R_2, R_3\}, R_2)$ will send the packets from R_1 and R_3 to R_2. The result of executing RPC$(\{R_5, R_6, R_7, R_8\}, R_7)$ will send the packets from R_5 and R_8 to R_7. In addition, RPC$(\{R_5, R_6, R_7, R_8\}, R_7)$ will call RPC$(\{R_5, R_6\}, R_5)$, such that R_6 sends packets to R_5.

Since the links in the Galaxyfly networks are bidirectional, the RPD algorithm can be designed as the inverse process of RPC with the same time complexity. Therefore, we omit the details of RPD and use RPD(A, m) to denote the distribution of packets from router m to all other routers in the supernode A.

3.2. Algorithms: Supernode-Packet-Collection (SPC) and Supernode-Packet-Distribution (SPD)

The algorithm Supernode-Packet-Collection (i.e., Algorithm 2) collects packets from other supernodes into one supernode via breadth-first traversal. In contrast, the algorithm Supernode-Packet-Distribution distributes packets from one supernode to other supernodes. We first propose the algorithm Supernode-Packet-Collection as follows.

Algorithm 2: Supernode-Packet-Collection (SPC).

Input: U: the set of all supernodes;
A: a supernode within the set U;
Output: Collect packets from all other supernodes in U to supernode A;
Let $U = U \setminus \{A\}$;
for $x \in U$ **do**
  if the supernode x is adjacent to supernode A **then**
    $U = U \setminus \{x\}$;
    for $y \in U$ **do**
      if the supernode y is adjacent to supernode x **then**
        Let e be the router in y connected to supernode x;
        RPC(y, e);
        Pass packets from supernode y to supernode x through router e;
        $U = U \setminus \{y\}$;
    Let m be the router in A connected to the router g in supernode x;
    RPC(x, g);
    Pass packets from supernode x to supernode A through router g;
    RPD(A, m);
return;

Theorem 4. *In the SPC algorithm, the packets are accurately sent to the target supernode.*

The algorithm SPC collects packets from all supernodes in U to a supernode A. First, the algorithm SPC will find the supernode x that is adjacent to supernode A from the set U. Then it will find y that is adjacent to supernode x from the set U, and transmit packets from supernode y to supernode x. Supernode x collects all packets to the router inside it that is connected to A, and then sends the packets to supernode A. The algorithm sends packets via breadth-first traversal, and its time complexity is $O(n \times q \times \log_2 a)$ (see the definition of n, q, a in Table 1).

Figure 4 shows the execution process of the SPC algorithm in a Galaxyfly network with $n = 3$ and $q = 5$. The specific detailed process is as follows.

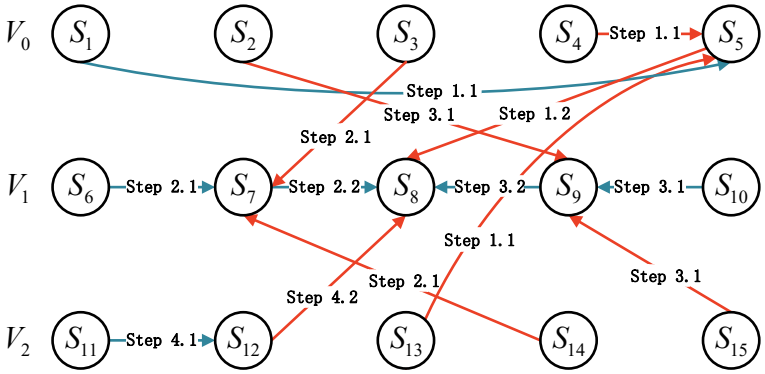

Figure 4. Collect packets from each supernode using the SPC algorithm in Galaxyfly networks.

Note that U is the set of all supernodes. We assume that the target supernode A is S_8. In the set U, the algorithm SPC will find the supernodes that are adjacent to supernode S_8, i.e., S_5, S_7, S_9, S_{12}. For S_5, the supernodes adjacent to it in U are S_1, S_4, and S_{13}. Pass the packets of S_1, S_4, and S_{13} to S_5 and collect them into a router inside S_5 connected to S_8. Then, pass the packets to S_8 and distribute the packets to all routers inside S_8. For S_7, the supernodes adjacent to it in U are S_3, S_6, and S_{14}. Pass the packets of S_3, S_6, and S_{14} to S_7 and collect them into a router inside S_7 connected to S_8. Then, pass the packets to S_8 and distribute the packets to all routers inside S_8. For S_9, the supernodes adjacent to it in U are S_2, S_{10}, and S_{15}. Pass the packets of S_2, S_{10}, and S_{15} to S_9 and collect them into a router inside S_9 connected to S_8. Then, pass the packets to S_8 and distribute the packets to all routers inside S_8. For S_{12}, the supernode adjacent to it is S_{11}. Pass the packet from S_{11} to S_{12} and collect them into a router inside S_{12} connected to S_8. Then, pass the packets to S_8 and distribute the packets to all routers inside S_8.

Since the links in Galaxyfly networks are bidirectional, the SPD algorithm can be designed as the inverse process of SPC with the same time complexity. Therefore, we omit the details of SPD and use SPD(U, A) to denote the distribution of packets from supernode A to all other supernodes in U.

4. All-to-All Broadcast Algorithm

In the Galaxyfly network, when each supernode needs to send a packet to all other supernodes, it is necessary to design all-to-all broadcast algorithms and to analyze its performance. In this process, each supernode contains packets that need to be sent to all other supernodes. That is, each supernode has to complete a broadcast operation. Obviously, this can be realized by utilizing the unicast operation for each supernode. However, such a simple strategy will result in many routers receiving a large number of redundant packets, taking up a large amount of bandwidth and resulting in the performance degradation of the network. This section proposes two all-to-all broadcast algorithms that can effectively solve these problems based on the four basic algorithms proposed in Section 3.

4.1. Supernode-First All-to-All (SFATA) Broadcast Algorithm

In Algorithm 3, named SFATA, once one supernode receives the packets, this supernode will distribute them to the next supernode immediately. Thus, the SFATA obeys the rule of supernode-first. The overview of the SFATA algorithm is shown as follows: The packets from the routers within the same supernode are collected into one router. Then, these packets are forwarded to a supernode, which receives all the packets and forwards them to all of the other supernodes. Finally, each supernode forwards the packets to all the routers inside it. The algorithm has the following features:

- All routers accept and send packets through the shortest feasible path.
- Each router does not receive redundant packets.

Algorithm 3: Supernode-first all-to-all broadcast algorithm (SFATA).

Input: U: the set of all supernodes;
A: a supernode;
Output: All routers within all supernodes receive all packets under the rule of supernode-first;
SPC(U, A);
SPD(U, A);
return;

Theorem 5. *In the SFATA algorithm, the packets are accurately sent to the target router and the target supernode.*

The SFATA algorithm executes the SPC algorithm to call the RPC algorithm exactly once for each supernode, which has a distance of 2 with the target supernode such that all the packets of these supernodes are collected to one router inside them. Next, the algorithm SPC sends the packets to the target supernode. Each time the packets are collected to a supernode, it is necessary to execute RPC to collect the packets to a router again, collecting $a \times n \times q - 1$ packets in total. The algorithm SPD is executed to distribute the collected packets to all the other supernodes. Each time a supernode receives the packets, the SPD algorithm will call the RPD algorithm to distribute the packets to all routers within the supernode. Each router receives a total of $a \times n \times q - 1$ packets. In the process, each router does not receive redundant packets. The time complexity of the algorithm is $O(n \times q \times \log_2 a)$.

Here, we assume that S_8 is the target supernode. First, the supernodes that have a distance of 2 with S_8 are S_1, S_2, S_3, S_4, S_6, S_9, S_{10}, S_{11}, S_{13}, S_{14}, and S_{15}, and thus, we execute RPC to collect the packets in these supernodes, respectively. Execute SPC such that the packets of S_1, S_4, and S_{13} are sent to S_5, packets of S_3, S_6, and S_{14} are sent to S_7, packets of S_2, S_{10}, and S_{15} are sent to S_9, and packets of S_{11} are sent to S_{12}. Execute RPC such that the packets inside S_5, S_7, S_9, and S_{12} are concentrated on the router connected with S_8, through which we send the packets to S_8. Execute RPD to send the packets from the routers connected with S_5, S_7, S_9, and S_{12} to other routers in S_8. Finally, execute SPD to send packets to all routers of other supernodes.

4.2. Router-First All-to-All Broadcast Algorithm (RFATA)

In Algorithm 4, named RFATA, once a packet is transmitted into one supernode, the supernode will distribute it to all routers inside it in priority, instead of distributing it to other supernodes immediately. Thus, the RFATA obeys the rule of router-first. The overview of the RFATA algorithm is shown as follows: The packets from routers within the same supernode are first collected into one router, and then this router will distribute these packets to all routers in the same supernode. Next, the packets are forwarded to all routers in other supernodes and this process obeys the rule of router-first. Once one supernode receives all the packets, this supernode will distribute the packets to all other supernodes. The algorithm has the following features:

- All routers receive and send packets through the shortest feasible path.
- The routers in the supernode will receive packets in priority.

Algorithm 4: Router-First All-to-All Broadcast Algorithm (RFATA).

Input: U: the set of all supernodes;
A: a supernode;
Output: All routers within all supernodes receive all packets under the rule of router-first;
for $x \in U$ **do**
  Let d be an arbitrary router in x;
  RPC(x, d);
  RPD(x, d);
Let $U = U \setminus \{A\}$;
for $x \in U$ **do**
  if the supernode x is adjacent to supernode A **then**
    $U = U \setminus \{x\}$;
    for $y \in U$ **do**
      if the supernode y is adjacent to supernode x **then**
        Let e be the router in y connected to the router l in supernode x;
        Pass the packet from supernode y to supernode x through router e;
        $U = U \setminus \{y\}$;
    RPD(x, l);
    Let m be the router in A connected to the router g in supernode x;
    Pass packets from supernode x to supernode A through router g;
    RPD(A, m);
SPD(U, A);
return;

Theorem 6. *In the RFATA algorithm, the packets are accurately sent to the target router and the target supernode.*

During the packet collection phase of the router within the supernode, the RFATA algorithm will execute RPC for each router to collect recursively all packets from the supernode. RPD is executed to distribute the collected packets to each router. Then, the packets are collected to a supernode using the breadth-first traversal algorithm. During the collection process, each supernode will pass the packets to all routers inside it immediately after receiving them. Finally, the SPD algorithm is executed to distribute the packets. The time complexity of the algorithm is $O(n \times q \times \log_2 a)$.

Here, we suppose that S_8 is the target supernode. First, RPC and RPD are executed on all supernodes, such that all the packets in the supernode are sent to all the routers inside it. The supernodes have a distance of 2 with S_8 are S_1, S_2, S_3, S_4, S_6, S_9, S_{10}, S_{11}, S_{13}, S_{14}, and S_{15}. The algorithm will send the packets of S_1, S_4, and S_{13} to S_5, send the packets of S_3, S_6, and S_{14} to S_7, send the packets of S_2, S_{10}, and S_{15} to S_9, send the packets of S_{11} to S_{12}, and send the packets of S_3, S_7, S_9, and S_{12} to S_8. Finally, SPD is executed to send the packets to all other supernodes' routers. During the entire process, as long as the supernode receives a packet, it will immediately send it to all routers inside it.

5. Simulation Results

In this section, for the SFATA and RFATA algorithms, we first validate their effectiveness and then evaluate their performance. The simulation experiments are written using NEDC and C++ language. The running environment is Intel Core i7-13700H, CPU 3.40 GHz, 32 GB RAM, Windows 10 64-bit OS under omnetpp 5.6. The proposed all-to-all broadcast algorithms only transmit the packets among the routers, which implies their

effectiveness and performance are independent of the parameters p and h. Therefore, we use Galaxyfly(n, q, a) to denote the Galaxyfly network in this section for brevity.

The workflow of experiments are shown as follows.

Step 1: Generate a Galaxyfly network by setting different numbers of clusters n, number of supernodes q, and number of routers a.

Step 2: Validate the effectiveness of proposed algorithms for Galaxyfly(n, q, a) under different packet sizes.

Step 3: Evaluate the performance of proposed algorithms for Galaxyfly(n, q, a) with a packet size of 160B.

Step 4: Record the above results and conduct Steps 1–3 several times to reduce accidental errors.

Throughout our simulation, we assume that the bandwidth of the network is 16 Gbps [33].

5.1. Validation

This subsection validates the effectiveness of our proposed algorithms. We generate five network configurations, i.e., Galaxyfly$(3, 5, 4)$, Galaxyfly$(3, 5, 8)$, Galaxyfly$(4, 5, 5)$, Galaxyfly$(4, 7, 4)$, and Galaxyfly$(4, 7, 5)$. The packet size, denoted by PS, varies among 160B, 320B, 640B, and 1280B. In order to explore the effectiveness of the SFATA and RFATA algorithms under these experimental configurations, we define the following three notations.

- N_r: the total number of routers in the network.
- N_c: the number of routers that successfully received packets from all routers.
- N_f: the number of routers that failed to receive packets from all routers.

Then, we further define two effectiveness metrics as follows.

- SuccessRate: the ratio of the number of routers receiving all packets successfully to the total number of routers, calculated by

$$\text{SuccessRate} = \frac{N_c}{N_r}, \tag{3}$$

- FailureRate: the ratio of the number of routers failing to receive all packets to the total number of routers, calculated by

$$\text{FailureRate} = \frac{N_f}{N_r}. \tag{4}$$

Table 2 presents the SuccessRate and FailureRate of the SFATA and RFATA algorithms for various PS values in different Galaxyfly network configurations: Galaxyfly(3, 5, 4), Galaxyfly(3, 5, 8), Galaxyfly(4, 5, 5), Galaxyfly(4, 7, 4), and Galaxyfly(4, 7, 5). Since the experimental results are identical for these four Galaxyfly configurations, we have consolidated them into a single table for convenience. In addition, to enhance readability, we use SFATA.SuccessRate and SFATA.FailureRate (respectively, RFATA.SuccessRate and RFATA.FailureRate) to denote the SuccessRate and FailureRate of the SFATA algorithm (respectively, RFATA algorithm), respectively.

Table 2. The SuccessRate and FailureRate of SFATA and RFATA algorithms for Galaxyfly$(3, 5, 4)$, Galaxyfly$(3, 5, 8)$, Galaxyfly$(4, 5, 5)$, Galaxyfly$(4, 7, 4)$, and Galaxyfly$(4, 7, 5)$ under different PS values.

PS	160B	320B	640B	1280B
SFATA.SuccessRate	100%	100%	100%	100%
SFATA.FailureRate	0%	0%	0%	0%
RFATA.SuccessRate	100%	100%	100%	100%
RFATA.FailureRate	0%	0%	0%	0%

We can observe that SFATA.SuccessRate and RFATA.SuccessRate always maintain 100%, while SFATA.FailureRate and RFATA.FailureRate are always equal to 0%. It indicates

that the two broadcast algorithms proposed are able to transmit all packets to each router in the Galaxyfly network. In addition, their SuccessRate and FailureRate are not affected by the PS values. Therefore, in subsequent experiments, we set the value of PS to 160B for convenience.

5.2. Performance Analysis

In this subsection, we will evaluate the performances of the SFATA and RFATA algorithms and compare them with the A2A algorithm, which is proposed for broadcast routing in Dragonfly networks [32]. Our evaluation is conducted from the following six metrics.

- AvgTime: the average time for all packets to be received by all routers in the network.
- MaxTime: the maximum time for all routers in the network to receive all packets, i.e., the running time of the whole all-to-all broadcast algorithm.
- MinTime: the minimum time for all packets to be received by the routers in the network.
- AvgPacket: the average redundant packets, which is the ratio of the total number of redundant packets to the total number of routers in the network.
- RouterTime: the average time for the router to collect packets from the supernode.
- AvgChannel: the average channel utilization, which is the ratio of the average time spent on all channels in the network to the total routing time (i.e., MaxTime).

We still implement the simulation under the aforementioned five network configurations, i.e., Galaxyfly$(3,5,4)$, Galaxyfly$(4,5,4)$, Galaxyfly$(4,5,5)$, Galaxyfly$(4,7,4)$, and Galaxyfly$(3,5,8)$. Tables 3–5 show the AvgTime, MaxTime, MinTime, AvgPacket, RouterTime, and AvgChannel of the SFATA, RFATA, and A2A algorithms, where the time units are µs.

Table 3. The AvgTime, MaxTime, MinTime, AvgPacket, RouterTime, and AvgChannel of SFATA algorithm.

(n,q,a)	$(3,5,4)$	$(4,5,4)$	$(4,5,5)$	$(4,7,4)$	$(3,5,8)$
AvgTime	34,100	43,700	62,400	57,300	91,600
MaxTime	56,500	73,700	92,900	101,300	132,900
MinTime	8500	9700	12,900	117,000	24,900
AvgPacket	0.000	0.000	0.000	0.000	0.000
RouterTime	27,366.7	35,057.5	52,587	46,030.4	82,947.5
AvgChannel	5.69%	5.72%	5.65%	5.73%	4.73%

Table 4. The AvgTime, MaxTime, MinTime, AvgPacket, RouterTime, and AvgChannel of RFATA algorithm.

(n,q,a)	$(3,5,4)$	$(4,5,4)$	$(4,5,5)$	$(4,7,4)$	$(3,5,8)$
AvgTime	44,920	56,925	82,000	71,735.7	137,213
MaxTime	67,400	87,000	112,600	115,800	178,700
MinTime	14,600	17,000	22,600	19,000	48,300
AvgPacket	1.000	2.333	1.200	0.643	1.000
RouterTime	2440	2310	3160	2096.43	8006.67
AvgChannel	4.02%	3.97%	3.80%	4.01%	2.95%

Table 5. The AvgTime, MaxTime, MinTime, AvgPacket, and AvgChannel of A2A algorithm [32].

(n,a)	$(3,20)$	$(4,20)$	$(4,25)$	$(4,28)$	$(3,40)$
AvgTime	14,295	18,080	24,696	28,896.4	38,997.5
MaxTime	21,900	28,200	34,800	39,000	55,900
MinTime	1800	1800	2200	2500	3800
AvgPacket	0.000	0.000	0.000	0.000	0.000
AvgChannel	12.99%	13.21%	13.52%	13.62%	10.53%

It can be observed that the AvgTime, MaxTime, and MinTime of the SFATA algorithm are always smaller than those of the RFATA algorithm. Particularly, the MaxTime of the RFATA algorithm is over 1.05 times that of the SFATA algorithm, implying the SFATA algorithm can save more than 5% of the time to complete the whole broadcast operation, compared to the RFATA algorithm. Moreover, there are no redundant packets in the SFATA algorithm. The AvgChannel of the SFATA algorithm is larger than that of the RFATA algorithm; thus, the SFATA algorithm has a higher network resource utilization than the RFATA algorithm. However, the RFATA algorithm has a lower RouterTime and this verifies the advantage of the RFATA algorithm in prioritizing packet delivery to all routers. For both the SFATA and RFATA algorithms, the AvgTime, MaxTime, MinTime, RouterTime, and AvgChannel are all positively correlated with the values of parameters n, q, a. From Table 5, it can be observed that with an equal number of routers, the A2A algorithm always outperforms both the SFATA algorithm and the RFATA algorithm in terms of AvgTime, MaxTime, MinTime, AvgPacket, and AvgChannel. This disparity arises due to the inherent differences between the Dragonfly and Galaxyfly networks with $q > 1$, characterized by the network diameters of 3 and 5, respectively. Notably, the Galaxyfly network offers the flexibility to adjust the number of supernodes (i.e., parameter q), thereby facilitating the generalizability of our findings in diverse scenarios, which is the unique advantage of our proposed algorithms.

From the above simulation results, we can conclude the features of our proposed algorithms, as shown in Table 6.

Table 6. The features of SFATA algorithm and RFATA algorithm.

	SFATA	**RFATA**
Rule	Supernode-first	Router-first
Advantages	Lower execution time AvgPacket equals to zero Higher AvgChannel	Lower RouterTime
Disadvantages	Higher RouterTime	Higher execution time Higher AvgPacket Lower AvgChannel
Application Scopes	Applications that require completion of the whole broadcast as soon as possible	Applications that require prioritization of packet delivery to all routers in the same supernode

6. Conclusions

In this paper, two all-to-all broadcast algorithms for the Galaxyfly network, complying with the rules of supernode-first and router-first, respectively, are not only proposed, but also actually implemented for delivering the packets from each router to all other routers. In order to quickly complete the broadcast, the SFATA algorithm collects packets from all supernodes to one supernode and then sends them to the other supernodes immediately. In contrast, the RFATA algorithm forwards incoming packets immediately to all routers within a supernode. The SFATA algorithm emphasizes the efficient utilization of network resources, whereas the RFATA algorithm prioritizes delivering packets to all routers within the same supernode.

Our future work aims to investigate fault-tolerant routing techniques that do not utilize virtual channels in Galaxyfly networks. This is currently an open and intriguing research question that requires further exploration. Additionally, we plan to explore the development of multicast routing algorithms that cater to the unique properties of the Galaxyfly network.

Author Contributions: Conceptualization, H.Z.; methodology, H.Z. and J.-M.C.; software, F.S. and Q.L.; validation, H.Z. and J.-M.C.; formal analysis, J.-M.C. and Q.L.; investigation, H.Z. and J.-M.C.; resources, X.-Y.L.; data curation, F.S. and Q.L.; writing—original draft preparation, H.Z.; writing—review and editing, X.-Y.L.; visualization, F.S.; supervision, J.-M.C. and X.-Y.L.; funding acquisition, J.-M.C. and X.-Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant 62002062, the Ministry of Science and Technology of Taiwan under Grant MOST-111-2221-E-141-006, and the Natural Science Foundation of Fujian Province under Grant 2022J05029.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in the study is available with the authors and can be shared upon reasonable request.

Acknowledgments: The authors would like to thank the anonymous reviewers and the editor for their careful reviews and constructive suggestions to help us improve the quality of this paper.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Fan, W.; Xiao, F.; Fan, J.; Han, Z.; Sun, L.; Wang, R. Fault-tolerant routing with load balancing in LeTQ networks. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 68–82. [CrossRef]
2. Liao, X.; Shen, Y.; Li, S.; Lu, Y.; Du, Y.; Chen, Z. Optimizing data query performance of Bi-cluster for large-scale scientific data in supercomputers. *J. Supercomput.* **2022**, *78*, 2417–2441. [CrossRef]
3. Mavroidis, I.; Papaefstathiou, I.; Lavagno, L.; Nikolopoulos, D.S.; Koch, D.; Goodacre, J.; Sourdis, I.; Papaefstathiou, P.; Coppola, M.; Palomino, M. Ecoscale: Reconfigurable computing and runtime system for future exascale systems. In Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 14–18 March 2016; pp. 696–701.
4. Sedova, A.; Davidson, R.; Taillefumier, M.; Elwasif, W. *HPC Molecular Simulation Tries Out a New GPU: Experiences on Early AMD Test Systems for the Frontier Supercomputer*; Oak Ridge National Lab: Oak Ridge, TN, USA, 2022.
5. Bharadwaj, S.; Yin, J.; Beckmann, B.; Krishna, T. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling. In Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 20–24 July 2020; pp. 1–6.
6. Cicconetti, C.; Conti, M.; Passarella, A. Low-latency distributed computation offloading for pervasive environments. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom), Kyoto, Japan, 11–15 March 2019; pp. 1–10.
7. ORNL's Exaflop Machine Frontier Keeps Top Spot, New Competitor Leonardo Breaks the Top10. 2022. Available online: https://www.top500.org/news/ornls-exaflop-machine-frontier-keeps-top-spot-new-competitor-leonardo-breaks-the-top10 (accessed on 23 March 2023).
8. Zahid, F.; Taherkordi, A.; Gran, E.G.; Skeie, T.; Johnsen, B.D. A self-adaptive network for HPC clouds: Architecture, framework, and implementation. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 2658–2671. [CrossRef]
9. Sensi, D.D.; Girolamo, S.D.; McMahon, K.H.; Roweth, D.; Hoefler, T. An in-depth analysis of the slingshot interconnect. In Proceedings of the SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 9–19 November 2020; pp. 1–14.
10. Kim, J.; Dally, W.J.; Abts, D. Flattened butterfly: A cost-efficient topology for high-radix networks. *SIGARCH Comput. Archit. News* **2007**, *35*, 126–137. [CrossRef]
11. Faanes, G.; Bataineh, A.; Roweth, D.; Court, T.; Froese, E.; Alverson, B.; Johnson, T.; Kopnick, J.; Higgins, M.; Reinhard, J. Cray cascade: A scalable HPC system based on a Dragonfly network. In Proceedings of the SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 10–16 November 2012; pp. 1–9.
12. Maglione-Mathey, G.; Yebenes, P.; Escudero-Sahuquillo, J.; Garcia, P.J.; Quiles, F.J.; Zahavi, E. Scalable deadlock-free deterministic minimal-path routing engine for infiniband-based dragonfly networks. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 183–197. [CrossRef]
13. Jiang, N.; Dennison, L.; Dally, W.J. Network endpoint congestion control for fine-grained communication. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA, 15–20 November 2015; pp. 1–12.
14. Xiang, D.; Li, B.; Fu, Y. Fault-tolerant adaptive routing in dragonfly networks. *IEEE Trans. Dependable Secur. Comput.* **2019**, *16*, 259–271. [CrossRef]

15. Ahn, J.H.; Binkert, N.; Davis, A.; McLaren, M.; Schreiber, R.S. HyperX: Topology, routing, and packaging of efficient large-scale networks. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, OR, USA, 14–20 November 2009; pp. 1–11.
16. Fujiwara, I.; Koibuchi, M.; Matsutani, H.; Casanova, H. Skywalk: A topology for HPC networks with low-delay switches. In Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, 19–23 May 2014; pp. 263–272.
17. Besta, M.; Hoefler, T. Slim fly: A cost effective low-diameter network topology. In Proceedings of the SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, USA, 16–21 November 2014; pp. 348–359.
18. Dai, Y.; Lu, K.; Xiao, L.; Su, J. A cost-efficient router architecture for HPC inter-connection networks: Design and implementation. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *30*, 738–753. [CrossRef]
19. Cao, J.; Lai, M.; Luo, Z.; Pang, Z. Efficient management and intelligent fault tolerance for HPC interconnect networks. In Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 4–6 December 2019; pp. 343–351.
20. Lei, F.; Dong, D.; Liao, X. Exploring the galaxyfly family to build flexible-scale interconnection networks. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 1054–1068. [CrossRef]
21. Joardar, B.K.; Duraisamy, K.; Pande, P.P. High performance collective communication-aware 3D network-on-chip architectures. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1351–1356.
22. Xiang, D.; Liu, X. Deadlock-free broadcast routing in dragonfly networks without virtual channels. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 2520–2532. [CrossRef]
23. Xiang, D.; Chakrabarty, K.; Fujiwara, H. Multicast-based testing and thermal-aware test scheduling for 3D ICs with a stacked network-on-chip. *IEEE Trans. Comput.* **2016**, *65*, 2767–2779. [CrossRef]
24. Lin, X.; Ni, L.M. Multicast communication in multicomputer networks. *IEEE Trans. Parallel Distrib. Syst.* **1993**, *4*, 1105–1117. [CrossRef]
25. Lin, X.; McKinley, P.K.; Ni, L.M. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. *IEEE Trans. Parallel Distrib. Syst.* **1994**, *5*, 793–804.
26. Panda, D.K.; Singal, S.; Kesavan, R. Multidestination message passing in wormhole *k*-ary *n*-cube networks with base routing conformed paths. *IEEE Trans. Parallel Distrib. Syst.* **1999**, *10*, 76–96. [CrossRef]
27. Boppana, R.V.; Chalasani, S.; Raghavendra, C.S. Resource deadlocks and performance of wormhole multicast routing algorithms. *IEEE Trans. Parallel Distrib. Syst.* **1998**, *9*, 535–549. [CrossRef]
28. McKinley, P.K.; Xu, H.; Esfahanian, A.-H.; Ni, L.M. Unicast-based multicast communication in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.* **1994**, *5*, 1252–1265. [CrossRef]
29. Suh, Y.; Valamanchili, S. All to-all communication with minimum start-up costs in 2D/3D tori and meshes. *IEEE Trans. Parallel Distrib. Syst.* **1998**, *9*, 442–458.
30. Jiang, N.; Kim, J.; Dally, W.J. Gossiping on meshes and tori. *IEEE Trans. Parallel Distrib. Syst.* **1998**, *9*, 513–525.
31. Hafner, P.R. Geometric realisation of the graphs of McKay–Miller–Širáň. *J. Comb. Theory Ser. B* **2004**, *90*, 223–232. [CrossRef]
32. Xiang, D.; Ju, Y. All-to-All Broadcast in Dragonfly Networks. In *Computing and Combinatorics. COCOON 2021*; Springer: Cham, Switzerland, 2021; pp. 13–24.
33. Fu, H.; Liao, J.; Yang, J.; Wang, L.; Song, Z.; Huang, X.; Yang, C.; Xue, W.; Liu, F.; Qiao, F.; et al. The Sunway TaihuLight supercomputer: System and applications. *Sci. China Inf. Sci.* **2016**, *59*, 072001. [CrossRef]