Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 124245, 15 pages
doi:10.1155/2012/124245

Research Article

Distributed and Fault-Tolerant Routing for Borel Cayley Graphs

Junghun Ryu,[1] Eric Noel,[2] and K. Wendy Tang[1]

[1] *Department of Electrical & Computer Engineering, Stony Brook University, SUNY, Stony Brook, NY 11794-2350, USA*
[2] *AT&T Labs, USA*

Correspondence should be addressed to K. Wendy Tang, wendy.tang@stonybrook.edu

Received 3 April 2012; Revised 25 July 2012; Accepted 8 August 2012

Academic Editor: Yanmin Zhu

We explore the use of a pseudorandom graph family, Borel Cayley graph family, as the network topology with thousands of nodes operating in a packet switching environment. BCGs are known to be an efficient topology in interconnection networks because of their small diameters, short average path lengths, and low-degree connections. However, the application of BCGs is hindered by a lack of size flexibility and fault-tolerant routing. We propose a fault-tolerant routing algorithm for BCGs. Our algorithm exploits the vertex-transitivity property of Borel Cayley graphs and relies on extra information to reflect topology change. Our results show that the proposed method supports good reachability and a small End-to-End delay under various link failures scenarios.

1. Introduction

Various graph-based interconnection networks have been applied to wavelength division multiplexed optical networks [1, 2], distributed parallel computation [3], distributed control [4], satellite constellations [5], chip design [6–9], and wireless sensor networks [10, 11]. In peer-to-peer overlay network schemes, various structure graphs are investigated compared to unstructured P2P overlay network [12]. For the example of structured P2P, k ring lattice is used in Chord [13] and de Bruijn graph is used in Koorde and Distance Halving [14, 15]. Also there are theoretic analyses to apply de Bruin and Cayley graphs to P2P [16, 17].

Deterministic characteristics for connections between nodes in structured graphs allow theoretical analysis and guarantee global properties such as a diameter and average path length [18]. Also graph-based networks can have symmetry, hierarchy, connectivity, and hamiltonicity, which are desired properties comparing random graph-based networks [17, 19].

Borel Cayley graphs (BCGs) have been shown to be efficient candidates for interconnection networks [20]. BCGs are known to have small diameters, average path lengths, and low-degree constant connections. The degree-diameter problem has been investigated in the contexts of interconnection networks [21, 22], wavelength division multiplexed optical networks [23], and VLSI layout design [24]. Also, BCGs are symmetric graphs, a property that enables distributed routing [25]. With consensus protocol [26], distributed node to node message exchange rule to drive nodes to an agreement for a quantity of interest, BCG showed better performance than mesh, torus, and small world networks [27]. Even though BCGs have such favorable properties, there are practical limitations in applying BCGs to networks. One of them is the lack of fault-tolerant routing algorithms: existing BCG routing algorithms do not account for node or communication link failures. Researchers have studied fault-tolerant routing on mesh, toroidal mesh, and de Bruijn graphs [28–30].

In this paper, we present a fault-tolerant routing algorithm for BCG, which accounts for communication link failures. For fault-tolerant routing, the routing tables of nodes are updated distributively in response to link failures. We quantify the performance of the routing algorithm by considering packet reachability and average hop count for different levels of communication link failures. Our simulation results show our proposed method to improve delivery performance by 20% to 350%. We also show packet congestion by proposed algorithms according to packet generation rate. We assume that contention is solved by

the MAC layer. Thus, we abstract this case as a graph with point-to-point links and transform the problem into a graph.

This paper is organized as follows. Section 2 reviews basic concepts and definitions for BCGs and related terminology. Section 3 presents our network model and compares BCGs with other known graph topologies. Section 4 presents the data structures used by our proposed routing algorithm for BCGs. Section 5 describes behaviors of the proposed routing algorithm. Section 6 presents simulation results to estimate reachability and the average hop count of our proposed routing algorithm. Conclusions are presented in Section 7.

2. Preliminaries

In the following, we provide a definition of Cayley graphs, Borel subgroup, and Borel Cayley graphs.

Definition 1 (Cayley graph [20]). A graph $C = \mathbb{C}(V, G)$ is a Cayley graph with vertex set V such that if two nodes $v_1, v_2 \in V$ are adjacent then $v_1 = v_2 * g$ for some $g \in G$, where $(V, *)$ is a finite group and $G \subset V \setminus \{I\}$. G is called the generator set of the graph and I is the identity element of the finite group $(V, *)$.

The definition of a Cayley Graph requires vertices to be elements of a group but does not specify a particular group.

Definition 2 (Borel subgroup). If V is a Borel subgroup of general linear 2×2 matrices set, then

$$V = \left\{ \begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix} : \; x = a^t (\text{mod } p), y \in Z_p, t \in Z_k \right\}, \tag{1}$$

where a is a fixed parameter $a \in Z_p \setminus \{0, 1\}$, p is prime, and k is the order of a. That is, k is the smallest positive integer such that $a^k = 1 \pmod p$.

Definition 3 (Borel Cayley graph (BCG) [20]). Let V be a Borel subgroup and let G be a generator set such that $G \subseteq V \setminus \{I\}$, then $B = \mathcal{B}(V, G)$ is a Borel Cayley graph with vertices 2×2 matrix elements of V. There exists a directed edge from v to u if only only if $u = v * g$, where $u \neq v \in V$, $g \in G$ and $*$ is the modulo-p multiplication chosen as a group operation.

Definition 4 (GCR [20]). A graph R is a generalized chordal ring (GCR) if nodes of R can be labeled with integers modulo number of nodes N, and there exists a divisor q of N such that node i is connected to node j if and only if node $i + q \pmod N$ is connected to node $j + q \pmod N$.

The connection rules of elements are defined by connection constants. Based on Definition 4, connection constants for i and $i + q$ are identical. When the graph is four regular, there are four connection constants. For example, Figure 1 shows a degree 4 GCR with 21 nodes and $q = 3$ classes. For

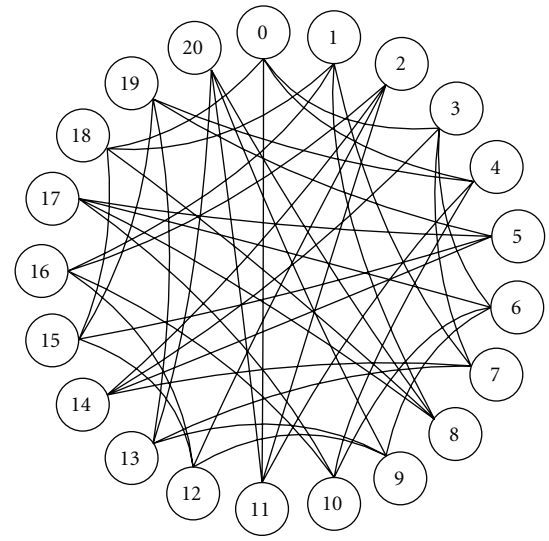


Figure 1: A Borel Cayley graph in the GCR representation with $p = 7$, $a = 2$, $k = 3$, $t_1 = 0$, $t_2 = 1$, $y_1 = 1$, and $y_2 = 1$.

$V = 0, 1, 2, \ldots, 20$ and any $i \in V$, the connection rules can be define as

$$\text{if } i(\text{mod}3) = \begin{cases} \text{“0” : } i \text{ is connected to } i+3, i-3, i+4, i-10 \pmod{21} \\ \text{“1” : } i \text{ is connected to } i+6, i-6, i+7, i-4 \pmod{21} \\ \text{“2” : } i \text{ is connected to } i+9, i-9, i-7, i+10 \pmod{21}. \end{cases} \tag{2}$$

Proposition 5. *For any finite Cayley graph with vertex set V and any $T \in V$ such that $T^m = I$, there exists a GCR representation of C with divisor $q = N/m$, where I is the identity element.*

The proof of these propositions is given in [20] and not repeated here. T is referred to as the transform element. a_i are class representing elements.

For simplicity of the GCR representation, we chose T and a_i as follows [20]:

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad a_i = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}^i. \tag{3}$$

Any vertex $v \in V$ is represented with T and a_i as follows [20]:

$$v = T^j * a_i = \begin{pmatrix} 1 & j \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a^i & j \\ 0 & 1 \end{pmatrix}. \tag{4}$$

BCGs are defined over a group of matrices. The systematic representation of BCGs from the group domain to the integer domain is useful for routing because nodes are defined in the integer domain and the integer domain provides a systematic

For a degree-4 Borel Cayley graph in the GCR representation with $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $a_i = \begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix}$, we have $q = k$ classes, where $a^k = 1 \pmod p$. Assume the generators to be g_1, g_2, g_1^{-1} and g_2^{-1}, where

$$g_1 = \begin{pmatrix} a^{t_1} & y_1 \\ 0 & 1 \end{pmatrix}, \qquad g_2 = \begin{pmatrix} a^{t_2} & y_2 \\ 0 & 1 \end{pmatrix}.$$

Given the source $i = m_1 q + c_1$ and the destination $j = m_2 q + c_2$.
While $(i \neq j)$
Step 1: Identify new destination,

$$j' = \langle a^{q-c_1}(m_2 - m_1)\rangle_p q + \langle c_2 - c_1\rangle_q,$$

where $\langle\ \rangle_p$ signifies the operation within the bracket $\langle\ \rangle$ is modulo p.
Step 2: From row j' of a precalculated routing table, determine which link to take.
Step 3: Identify new source, $i' = mq + c$ and
$m = y_1$, $c = t_1$, if link g_1 was chosen
$m = y_2$, $c = t_2$, if link g_2 was chosen
$m = p - \langle a^{q-t_1} y_1\rangle$, $c = q - t_1$, if link g_1^{-1} was chosen
$m = p - \langle a^{q-t_2} y_2\rangle$, $c = q - t_2$, if link g_2^{-1} was chosen
Step 4: $i = i'$ and $j = j'$

Algorithm 1: Vertex-transitive routing algorithm for Borel Cayley graphs.

description of connections. The node ID representation in GCR ($ID_g(v)$) is denoted as follows [20]:

$$ID_g(v) = q * j + i, \tag{5}$$

where q is the parameter k in Definition 2.

Symmetry or vertex transitivity is a preferable attribute for an efficient interconnection network topology. Informally, a symmetric or vertex-transitive graph looks the same from any node. This property allows to use an identical routing table at every node. Mathematically, this implies that for any two nodes a and b in the graph there exists an automorphism of the graph that maps a to b. This property is very useful for practical implementation of interconnection networks. Most of the well-known interconnection graphs, such as the toroidal mesh, hypercube, and cube-connected cycle, exhibit this property.

Proposition 6. *All Cayley graphs are vertex transitive [20].*

Every Cayley graphs can be represented with integer node labels through a transformation into a generalized chordal ring topology. However, generally speaking, GCR graphs are not fully symmetric. In [20], the authors provide a framework for the formulation of the complete symmetry (or vertex transitivity) of Cayley graphs in the integer domain of GCR representations.

Proposition 7. *Assume a Borel Cayley graph in the GCR representation with transform element* $T = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ *and representing elements of each class* i *as* $a_i = \left(\begin{smallmatrix} a^i & 0 \\ 0 & 1 \end{smallmatrix}\right)$. *Let* $i = m_1 q + c_1$, $j = m_2 q + c_2$, *and* $i' = m' q + c'$. *If* i *is connected to* j *with a sequence of generators, then* i' *is connected to* j' *with the same sequence of generators, where* $j' = \langle m' + a^{\langle c' - c_1\rangle_q}(m_2 - m_1)\rangle_p q + \langle c' - c_1 + c + 2\rangle_q$ *[25].*

Algorithm 1 shows Vertex-transitive routing (VT routing) algorithm that exploits the inherent symmetry of Cayley graphs and uses the identical routing table at any node [25].

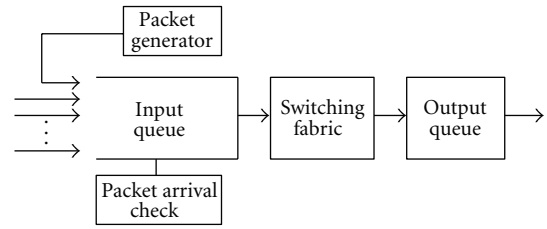


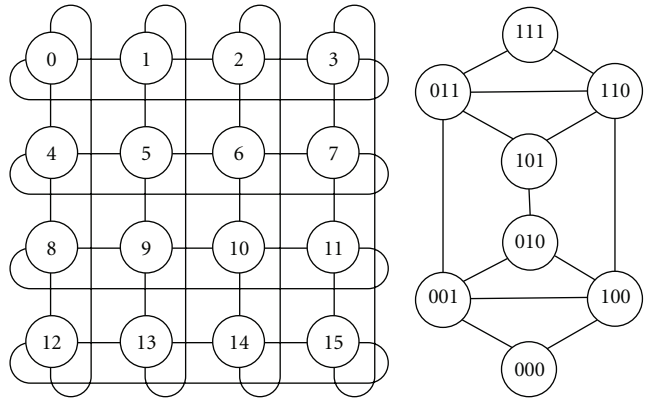
Figure 2: Node model.

3. Topology Comparison

3.1. Network Model. A network that consists of a set of nodes is connected by full duplex point-to-point links. The node model is depicted in Figure 2. Each node consists of an input queue for transit messages (Rx), a packet generator, a switching fabric, and an output queue. Modules inside a node are connected by zero delay links. It takes a single time unit for a packet to move from an output queue to an input queue. Time is slotted and synchronized so that all nodes receive and transmit packets simultaneously.

An input queue is FIFO served. In each time slot, the input queue accepts up to the number of packets, the degree of a node in the same time slot. Depending on the model, the input buffer size ranges from one to infinity. The output buffer size is one. The packet arrival module removes packets from the input queue if the current node is the destination node. The switching fabric determines the next node of the packet taken from the input queue by a routing algorithm. Every node in the network can be a source, a destination, or a relay. We assume that nodes generate information at a constant average rate of R packets per time slot (Packet generator).

Three traffic patterns are considered in this paper: All-to-All traffic pattern (Pattern 0), All-to-one traffic pattern (Pattern 1), and All-to-M traffic pattern (Pattern 2). In Pattern 0, all nodes are the source and destination nodes, which



(a) 4 x 4 toroidal mesh network

(b) 2-dimensional de Bruijn graph of 3 symbols

Figure 3: Toroidal mesh network and de Bruijn graph.

```
(1) procedure Routing Algorithm For UDB (cur, dst)      ▷cur: the current node, dst: the destination node
(2)   i ⟵ match_fwd(cur, dst)
(3)   j ⟵ match_bwd(cur, dst)
(4)   if i < j then
(5)       using FP sent to the particular left neighbor
(6)   else if i = j then
(7)       randomly using FP or BP
(8)   else
(9)       using BP sent to the particular right neighbor
(10)  end if
(11) end procedure
```

Algorithm 2: Undirected de Bruijn graph routing algorithm.

generate packets to uniformly randomly selected destination node. The probability of any source node communicating to any destination node in the network is constant and equal to $1/n$ where n is the number of destination nodes. In Pattern 1, nodes send packets to node 0, the only destination node. That is, the one node (sink) gathers the information generated by all other nodes in the network [31]. In Pattern 2, nodes send packets to a group of nodes, 5% of the total nodes in this case. This network model depicts a situation arising in integrated devices. Nodes located on the borders can be connected to high-capacity transmission lines.

3.2. Topologies. Toroidal mesh networks and de Bruijn graphs are popular topologies for interconnection networks [1, 5, 16]. In the following, we provide a definition for toroidal mesh networks and de Bruijn graphs. Then we show simulation results comparing Borel Cayley graphs with the aforementioned traffic patterns.

3.2.1. Toroidal Mesh Network. Figure 3(a) shows a toroidal mesh network (torus) which consists of R rows by C columns. When a node is represented by $rC + c$, neighboring nodes are defined as follows: $((r-1) \bmod R)C + c$, $rC + (c-1) \bmod C$, $((r+1) \bmod R)C + c$, and $rC + (c+1) \bmod C$. We use a Greedy row-first routing algorithm on the torus mesh network. If a packet is not in the destination column, then the packet is routed along the row towards the destination column. Otherwise the packet is routed along the column toward the destination node [32].

3.2.2. De Bruijn. The undirected de Bruijn graph (UDB) has $N = d^k$ nodes of degree $2d$ [1]. We use binary de Bruijn graphs $DG(2, k)$ of $N = 2^k$ nodes. A node of the network with binary address $a_{k-1}a_{k-2}\cdots a_1a_0$ has neighbors: $a_{k-2}a_{k-3}\cdots a_0a_{k-1}$, $a_{k-2}a_{k-3}\cdots a_0\overline{a}_{k-1}$, $a_0a_{k-1}\cdots a_2a_1$, and $\overline{a}_0a_{k-1}\cdots a_2a_1$. Figure 3(b) shows an undirected de Bruijn graph for $k = 3$ and $d = 2$ where self-loops are removed.

Algorithm 2 corresponds to the routing algorithm for UDB. The routing algorithm consists of transmitting a packet to either its left or right neighbors [33]. Algorithm 2 defines the Forward Path (FP) as the path taken by a packet when a left neighbor is chosen as the next node and the Backward Path (BP) when a right neighbor is chosen. From de Bruijn graph's properties, it is easy to calculate the number of hops that it needs to reach the destination using FP or BP. This is done by matching the postfix portion of the source address with the prefix portion of the destination address.

The more digits are matched; the shorter is the path between source and destination. For example, in Figure 3(b), assume node 011 needs to transmit a packet to node 110. Since 11 is the postfix of the source and the prefix of the destination, node 011 will reach node 110 in one hop. In [33], it defined $match\_fwd(cur, dst)$ to be an operation which returns the number of hops required to reach the destination along a FP. Similarly, $match\_bwd(cur, dst)$ returns the number of hops along a BP.

3.2.3. Performance Comparison. The parameters for the mesh networks we used for performance comparison are described in Table 1. BCG and Torus are degree 4 graphs. Most nodes of UDB have degree 4 except the few nodes, with self-loops. Table 2 shows our benchmark mesh networks topological properties such as the diameter and the average path length. The diameter is the greatest distance between any two nodes. The average path length is the average number of edges between all possible node pairs. Constrained by degree 4, BCG has the smallest diameter and the shortest average path length.

We also considered the performance of our network models. We used two metrics for comparison: (a) End-to-End delay and (b) reachability. We define End-to-End delay as the time required by packets to travel from a source to a destination and the reachability as the number of packets reaching destination over the number of generated packets. When running our simulations, we used the three types of traffic patterns presented in 3.1. We set the input buffer length to infinite, 5, and 10. Our simulation running time is 100000 ticks.

Figure 4 shows End-to-End delay (ETE delay) as a function of packet generation rate for our three traffic patterns. BCG exhibits the smallest End-to-End delay across all traffic patterns. Each network shows that End-to-End delay increases rapidly above a certain traffic generation rate called *traffic congestion point*. An efficient network topology should consider both End-to-End delay and network saturation. BCG shows a small End-to-End delay and a more robust traffic congestion point than UDB and Torus.

When a buffer at each node is finite, packets can be overflowed for large packet generation rates and thus reachability is not achieved as 100%. Figures 5 and 6 show reachabilities with buffer length 10 and 5, respectively. Reachability with finite buffer decreases above certain packet generation rate. Decreasing reachability packet generation rate of BCG is larger than others even though they have almost the same number of nodes and edges. UDB and Torus show similar reachabilities with traffic pattern 1 because End-to-End delays of traffic pattern 1 increase rapidly together at the similar packet generation rate.

Table 1: Mesh networks parameters.

	N	Parameters
BCG	1081	$P = 47, k = 23, a = 2, t_1 = 1, t_2 = 7, y_1 = 1, y_2 = 1$
Torus	1088	$R = 32, C = 34$
UDB	1024	$d = 2, k = 10$

Table 2: Static Property.

	AVG. path length	Diameter
BCG	5.54	7
Torus	16.52	33
UDB	6.77	10

4. Data Structure of Exhaustive Routing

A conventional routing algorithm for Borel Cayley graphs, Vertex-transitive routing, exploits the graphs vertex-transitive property. That property allows to use an identical routing table in every node. The routing table is created for node 0 only; and for other nodes to use that table, a simple node ID translation is applied to the destination node ID. The Vertex-transitive routing algorithm guarantees the shortest path between any source-destination pair. However, Vertex-transitive routing only applies to a static network and cannot account for node/link failures. The goal of our proposed routing algorithm (Exhaustive routing) is to route messages in the presence of link failures.

Exhaustive routing has two phases. In Phase 1, packets are routed through the shortest path according to the Static Routing Table. If there is a link failure making the shortest path unavailable, a Dynamic Routing Table for Type 1 packets is updated and other shortest paths (following the Static Routing Table) will be used. However, when all shortest paths from the BCG are disconnected, there can still be a path between the source and destination. In that case, Phase 2 of Exhaustive routing is used.

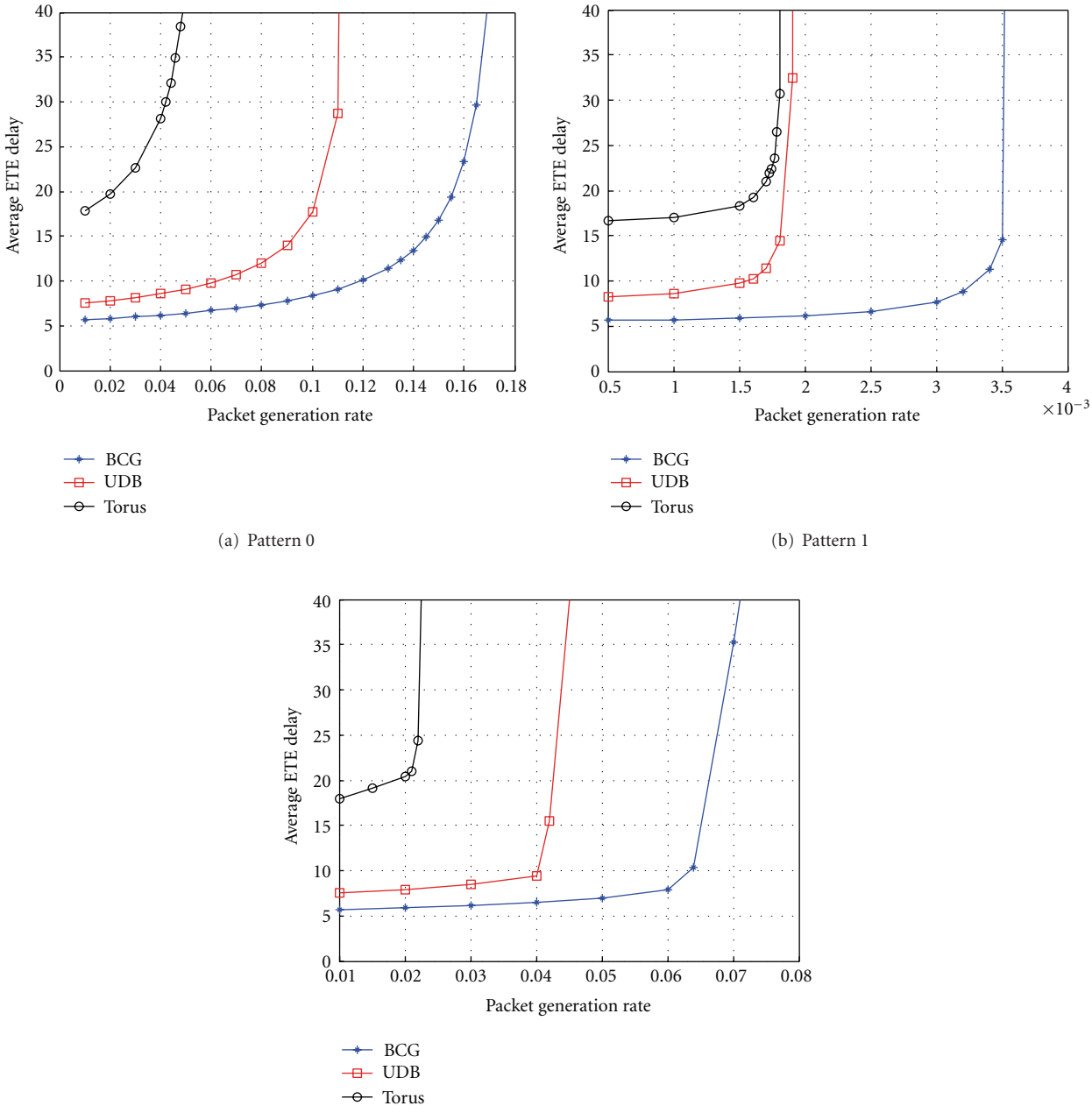
Phase 2 exploits the path length information in the Static Routing Table to search for possible routes besides the shortest paths from BCG. A Dynamic Routing Table for Type 2 packets is created to indicate "next best" paths as well as any unusable link due to failures. Basically, in Phase 2, packets are routed according to the path length information and update unusable links in the Dynamic Routing Table.

We define two types of routing tables according to whether or not the table changes in response to link failures: (a) a Static Routing Table and (b) a Dynamic Routing Table. The following provides a more detailed description of the two types of routing tables.

4.1. Static Routing Table. The Static Routing Table is precalculated and identical across all nodes. The Static Routing Table is defined for a reference source node, node 0. Each time a message needs to be routed from a node different than node 0. So the destination ID is mapped from absolute ID (Global ID of the destination in a network) to relative ID (ID of the destination in the view of the current node regarded as a reference node) as follows, referring to Algorithm 1:

$$j' = \left\langle a^{q-c_1}(m_2 - m_1)\right\rangle_p q + \langle c_2 - c_1\rangle_q, \tag{6}$$

where the current node's absolute ID is $m_1 q + c_1$, the destination node's absolute ID is $m_2 q + c_2$, and j' is the relative destination ID. We denote the absolute ID of node



(a) Pattern 0

(b) Pattern 1

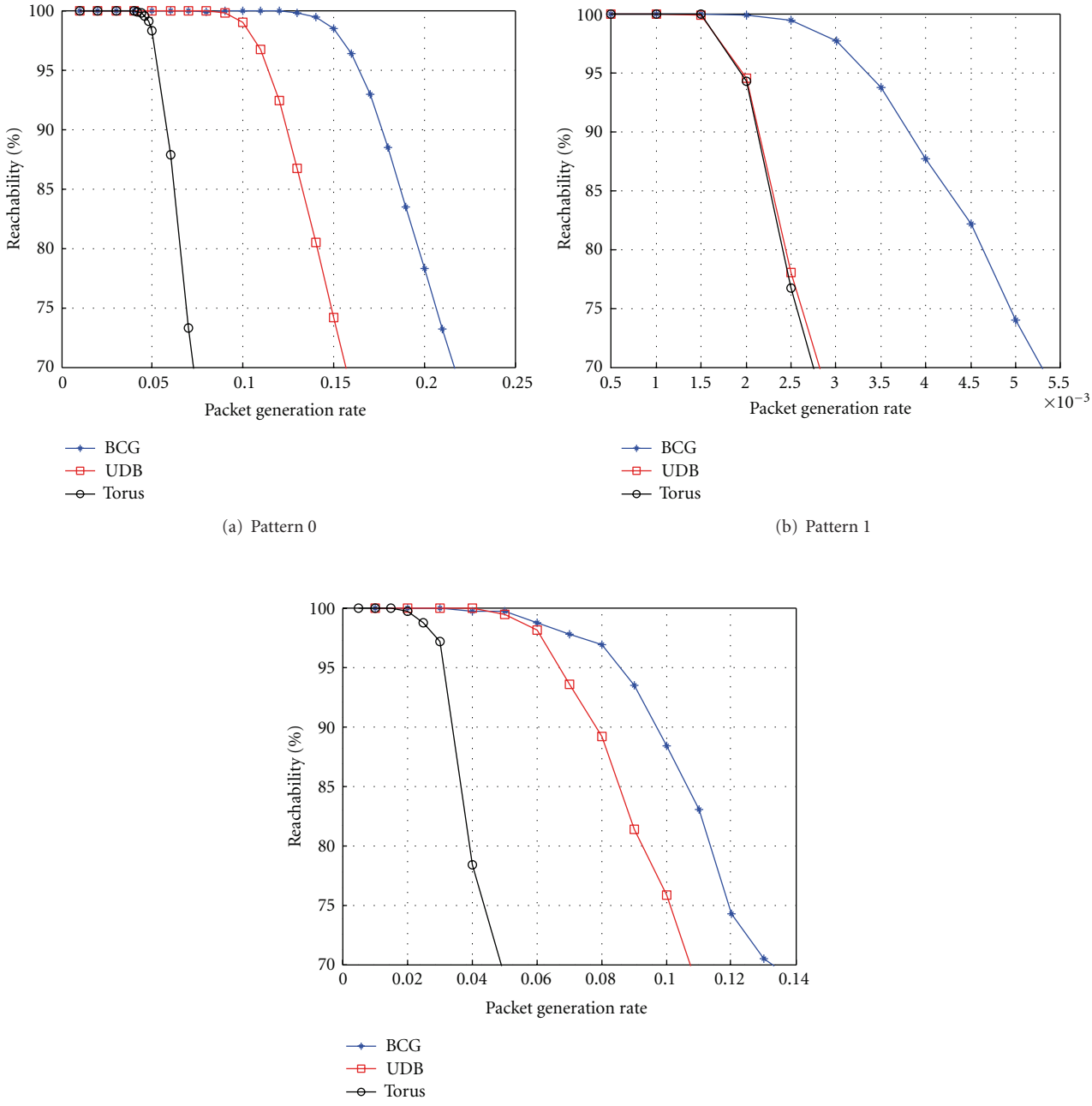
(c) Pattern 2

Figure 4: End-to-End delay with infinite buffer.

u by $a\mathrm{ID}(u)$ and the relative ID of node u at node v by $r\mathrm{ID}(u, v)$. Row indexes of Static Routing Table are relative IDs.

The number of rows in the Static Routing Table is the number of nodes minus one and the number of columns is the number of generators (nodal degree). In Figure 7(a), a number 1 in the routing table for Vertex-transitive routing indicates the shortest path through that generator link. For instance, for the relative destination ID 4, the shortest path at node u is through generator g_2.

The Static Routing Table for Exhaustive routing (SRTBL) includes the shortest path lengths to destination through an indicated generator. The shortest path lengths are calculated from Dijkstra's algorithm [34]. The first row of the Vertex-transitive routing table in Figure 7(a) shows an entry of 1 in the generator g_1^{-1} cell. On the other hand, in the first row of SRTBL in Figure 7(b), the shortest path is though generator g_1^{-1} and is two hops away from destination. If we choose the generator g_2 not g_1^{-1}, it would take four hops to reach destination. We denote the routing data to node v from node u by $\mathrm{SRTBL}(r\mathrm{ID}(v, u))$ and the hop count through generator g from node u to node v by $\mathrm{SRTBL}(r\mathrm{ID}(v, u), g)$. For example, $\mathrm{SRTBL}(r\mathrm{ID}(3, 0)) = (1, 3, 3, 3)$ and $\mathrm{SRTBL}(r\mathrm{ID}(3, 0), g_2) = 3$.



(a) Pattern 0

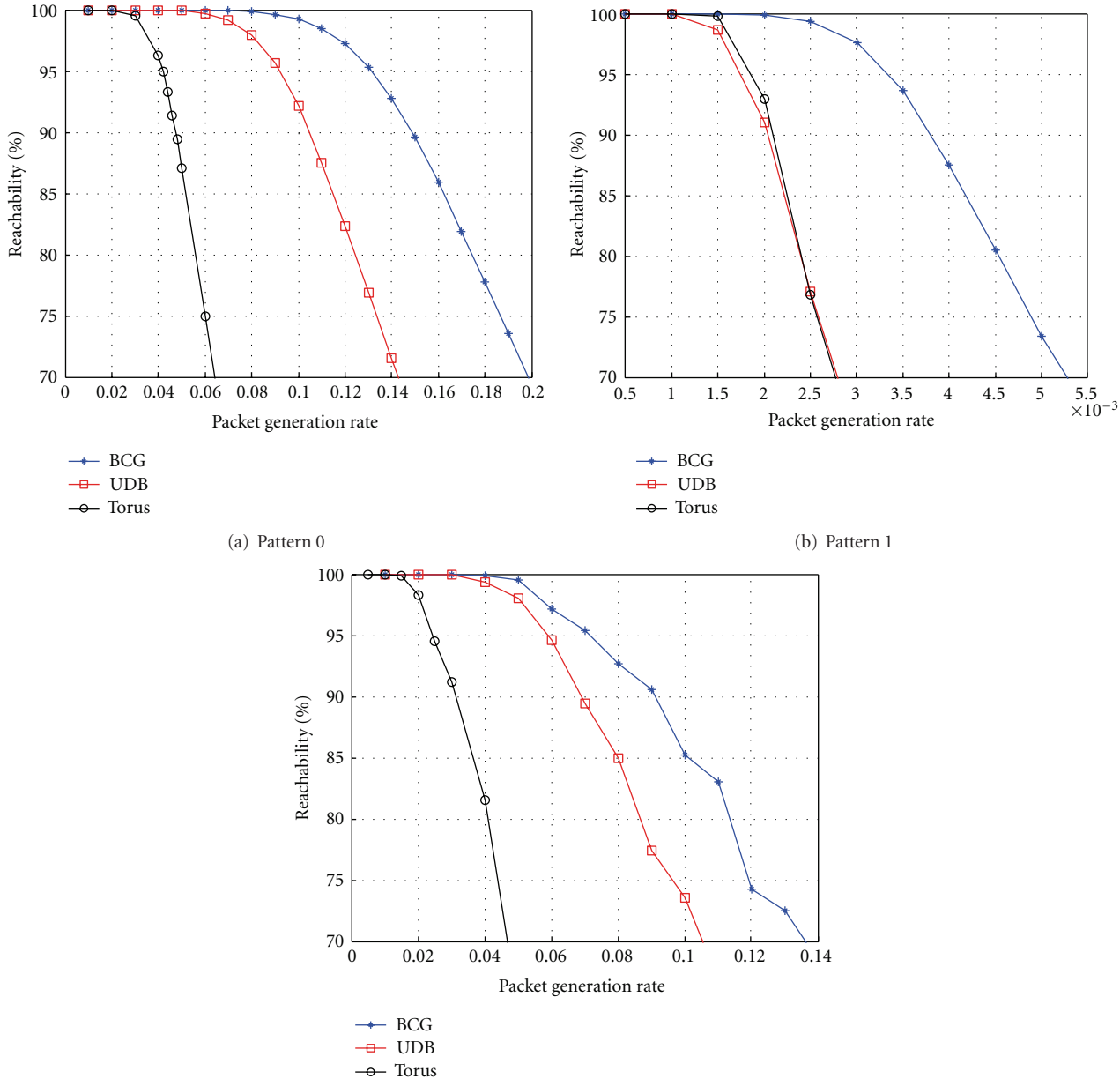
(b) Pattern 1

(c) Pattern 2

Figure 5: Reachability with buffer length 10.

4.2. Dynamic Routing Table. The Dynamic Routing Table (DRTBL) is generated or updated based on route availability. From the contents of a received packet, a node can determine whether or not certain shortest path links are no longer available. We will explain how to determine when links are no longer available in Section 5. DRTBL is generated for each destination node, hence the size of DRTBL will vary. For example, node u detects that the g_2 link for node v is no longer available. If there is no DRTBL for node v, node u generates a new DRTBL for node v. Otherwise, it sets the g_2 link to zero in existing DRTBL for node v. DRTBLs are unique at each node. So the relative ID is no longer needed. The index of DRTBL is the absolute ID. We denote routing data for node v at node u by DRTBL(aID(v), aID(u)) and data indicated by generator g for node v at node u by DRTBL(aID(v), aID(u), g).

The Exhaustive routing algorithm has two phases during which Type 1 and Type 2 packets are forwarded in the first phase and the second phase, respectively. We denote DRTBL for Type 1 packets by D1RTBL and DRTBL for Type 2 packets by D2RTBL. Table 3 summarizes routing tables for Exhaustive routing.



(a) Pattern 0

(b) Pattern 1

(c) Pattern 2

Figure 6: Reachability with buffer length 5.

5. Phases of Exhaustive Routing

5.1. Phase 1. Phase 1 of Exhaustive routing exploits all the shortest paths extracted from BCG with SRTLB and D1RTBL.

5.1.1. Forwarding Rule of Phase 1. In Phase 1 of Exhaustive routing, the goal of forwarding a packet is to identify links corresponding to the shortest paths for the destination in SRTBL. Then packets are forwarded to a randomly selected link among the identified shortest path links. When a node can no longer forward a packet using the SRTBL, the shortest path links are disconnected, and the node forwards the packet to the previous node using the packet's Path history. Path history is an ordered list of nodes traversed by the packet being routed. The Path history is included in the Packet frame as follows:

$$\text{Packet frame} = \{\text{Source, Destination, Packet Type, Path history}\}. \quad (7)$$

5.1.2. Generating Dynamic Routing Table of Phase 1. Once a node receives a packet, it uses the Path history of the packet to determine whether or not the packet is a returned packet. If the node ID is found at the position

	g_1	g_1^{-1}	g_2	g_2^{-1}
1	0	1	0	0
2	0	0	0	1
3	1	0	0	0
4	0	0	1	0
5	1	1	1	0

(a) Routing table for Vertextransitive routing

	g_1	g_1^{-1}	g_2	g_2^{-1}
1	3	2	4	4
2	3	4	3	2
3	1	3	3	3
4	3	3	1	2
5	3	3	3	4

(b) Static Routing Table for Exhaustive routing

Figure 7: Original vertex-transitive routing table and Static Routing Table of BCG in Figure 1. Note that some parts of routing tables are omitted for brevity.

Table 3: Summary of routing tables for Exhaustive routing.

SRTBL	Precalculated
	Data are not changed
	Data are path lengths via generator
	For Phase 1 and Phase 2
D1RTBL	Generated dynamically
D2RTBL	Data are bits indicating the available links
	Data are updated when node/link failures occur
D1RTBL	For Phase 1
	Initialized with data indicating shortest path links
D2RTBL	For Phase 2
	Initialized with data indicating all links are to be explored

before the last in the Path history, the received packet is a returned packet. When a node receives a returned packet, its D1RTBL is generated or updated. The SRTBL consists of path lengths via each generator. The links in SRTBL with smallest cell for a given rID are used. For example, in Figure 7(b), when node a determines the packet whose destination is node b ($r\mathrm{ID}(b,a) = 5$) is a returned packet via generator g_1; it generates the D1RTBL for node b. we get $\mathrm{D1RTBL}(a\mathrm{ID}(b)) = (1,1,1,0)$ since the smallest number is 3 in the $\mathrm{SRTBL}(r\mathrm{ID}(b,a) = 5) = (3,3,3,4)$. It sets the entry indicated by generator g_1 to zero. Finally, node a has $\mathrm{D1RTBL}(a\mathrm{ID}(b)) = (0,1,1,0)$.

The generator link used by a returned packet is set to zero in the D1RTBL. Upon receiving a returned packet, if another shortest path exits (D1RTBL entry for the destination is 1), the node forwards the packet to a node indicated by the generator link. Otherwise the node removes the last node ID from the Path history and forwards the packet to the previous node. If the packet goes back all the way to the source node and the source node does not have any shortest path to the destination from D1RTBL, the node changes the packet type from Type 1 to Type 2. Phase 1 of Exhaustive routing supports routing delivery as long as there is at least one shortest path extracted from BCG. Algorithm 3 shows Phase 1 of our Exhaustive routing algorithm.

5.2. Phase 2. Table generation rules and packet forwarding rules for Type 1 and Type 2 packets are different. A Type 1 packet returns to source when there is no shortest path within D1RTBL. Then the source node changes the packet type from Type 1 to Type 2. The Type 2 packet is forwarded via a communication link having the smallest value in the SRTBL when a node does not have a D2RTBL of the destination.

A packet gets stuck at a node having no available shortest path in Phase 1. In Phase 2, the node receiving Type 2 packet updates or generates a D2RTBL of the destination node. The D2RTBL is initialized to 1 at all edges when generated. Type 2 packets directly refer to all path lengths information from the SRTBL. The node chooses the link having the smallest path length in SRTBL and an entry of one in the corresponding D2RTBL. When no outgoing link can be identified, a node forwards the packet back to the previous node in the Path history. From this, Exhaustive routing exploits more routes to destination. This mechanism improves the reachability exploiting more available paths to destination. Figure 8 shows our Exhaustive routing algorithm flow.

The phase 2 of Exhaustive routing can have loops as shown in Figure 9. Assume node s sends a packet to node d via a. The packet reaches to node e via c but the communication link is disconnected. Then except the incoming link, the packet is forwarded to node f. The packet follows in a circle like $a \to b \to c \to e \to f \to h \to a \to b$. To prevent loops, Phase 2 uses a different method to check whether a packet is a returned packet. In the case of Phase 1, a node checks whether the previous node of the last node in the Path history is itself. However, in the case of Phase 2 of Exhaustive routing, the node checks all the Path history node IDs. Then node a is a returned packet referring to the Path history (s,a,b,c,e,f,h). Node a sets to zero at the generator

```
(1) procedure Packet forwarding (pkt)
(2)   if pkt.dst = curID then                    ▷ pkt.dst: destination of packet
(3)     Packet delivery is successful            ▷ curID: node ID of current node
(4)     Return
(5)   end if
(6)   if pkt.dst is a returned packet Table then
(7)     Update Dynamic Routing Table             ▷ Case: node in path history
(8)   end if
(9)   if pkt.dst Dynamic Routing Table then
(10)    if There is an available link in row of destination in Dynamic Routing Table then
(11)      Forward the packet to randomly selected available generator link in Dynamic Routing Table
(12)      Return
(13)    else if pkt.src = curID then
(14)      Change a type of pkt from Type 1 to Type 2   ▷ There is no available shortest path from BCG
(15)      Go to Phase 2 of Exhaustive routing
(16)    else
(17)      Update Path history and forward the packet to the previous node
(18)    end if
(19)  else if Row of destination in Static Routing Table then
(20)    Forward the packet to randomly selected available generator link, which has the smallest path length within the
same row, in Static Routing Table
(21)    Return
(22)  end if
(23) end procedure
```

Algorithm 3: Phase 1 of Exhaustive routing.

link to node b in D2RTBL(d, a). The generator to h also sets to zero through the same method. Finally, node a delivers the packet via node m with Path history (s, a).

6. Simulation

We have designed simulators and performed experiments to evaluate our proposed fault-tolerant routing algorithm. We simulated BCG networks with $N = 1081$ (N is the number of nodes). We list the parameter values for BCGs used in Table 1. Parameters p and a determine N and BCG parameter k. Parameters t_1 and y_1 were used to construct the first generator. Parameters t_2 and y_2 were used to construct the second generator. Using two different generators and their inverse generators, we construct undirected degree 4 BCGs. We arbitrary chose parameters ts and ys for generators.

6.1. Static Property Performance. First, we simulated network disconnection by edge eliminations on BCG. We randomly select edges to be eliminated. For each simulated case, we generated 100 networks. The BCG is originally a connected graph. When we eliminate some edges, the network can consist of multiple network components (components are not connected each other). We measured packet delivery performance to the largest component only if the largest component has over 95% of the total nodes. Figure 10 shows the percentage of connected graphs among the 100 network samples for each edge elimination rate. From those results, we simulated BCG ranging from 5% to 35% elimination of edges.

We showed two metrics for comparison of proposed routing algorithms: (a) routability and (b) the average hop count. We define the routability as the number of reachable source and destination pairs among all pairs of nodes in the largest component of a network and the average hop count as the average number of nodes traversed by a packet between its source and destination. In this subsection, the simulator does not generate a packet before the previous sent packet is dropped by a routing algorithm or reached to the destination (only one packet exists in the network), which helps to measure the routing algorithm performance regardless of packet congestion and buffer length.

We compared Exhaustive routing with Exhaustive routing with only Phase 1 routing (Phase 1 routing) and vertex-transitive routing (VT routing). Results of the Exhaustive routing are acquired after dynamic routing tables are stabilized. For comparison purpose, in our implementation of the original VT routing, a random optimal link is chosen in cases where multiple optimal links exist. When there is no available link, the packet is discarded.

6.1.1. Routability. Figure 11 shows routability of BCG with 1081 nodes after eliminating edges. Phase 1 routing shows larger routability than Vertex-transitive routing because Phase 1 routing exploits all shortest paths between source and destination. Exhaustive routing shows the largest routability.

6.1.2. Average Hop Count. Figure 12 shows the average hop counts of BCG with 1081 nodes with 35% edge elimination. Comparing the average hop counts of our proposed

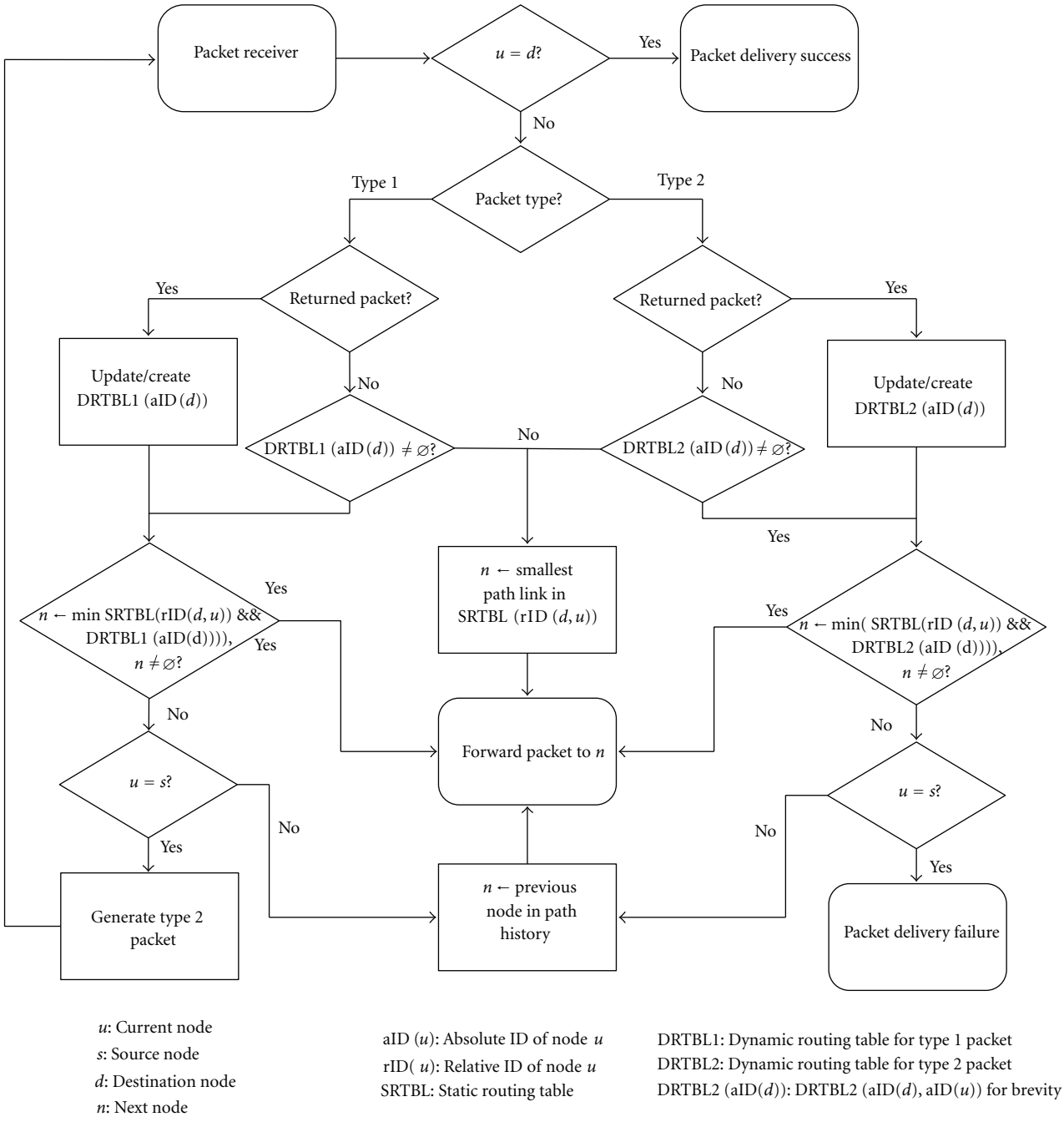


Figure 8: Exhaustive routing algorithm flow.

algorithms is unfair when routability is not the same. For example, one routing algorithm supports packet delivery to only nodes within a short distance from their sources and another routing algorithm supports packet delivery to all the nodes. In this case, average hop count of the second algorithm is larger than the former. So we compared only average hop count of Exhaustive routing with the results of Optimal routing in which the shortest path from the current network between the source and destination node is used. When a packet is not delivered to the destination, we exclude it from the average hop count.

6.1.3. Distribution of Hop Counts. We investigated frequency of hop counts when nodes are eliminated. Figure 13 shows hop counts distribution. The histogram of hop counts exhibits a right-skewed distribution with a high frequency of short hop counts.

6.2. Dynamic Property Performance. We show dynamic properties of our routing algorithms, which means measuring performance when multiple packets are flowing simultaneously in the network, as opposed to the case of static properties. In this case, packet generation rate and buffer

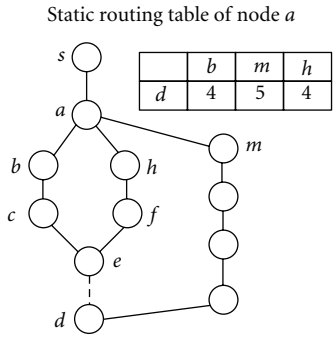


Figure 9: Network to illustrate a loop of Exhaustive routing. The dot line between nodes indicates that the communication is disconnected.

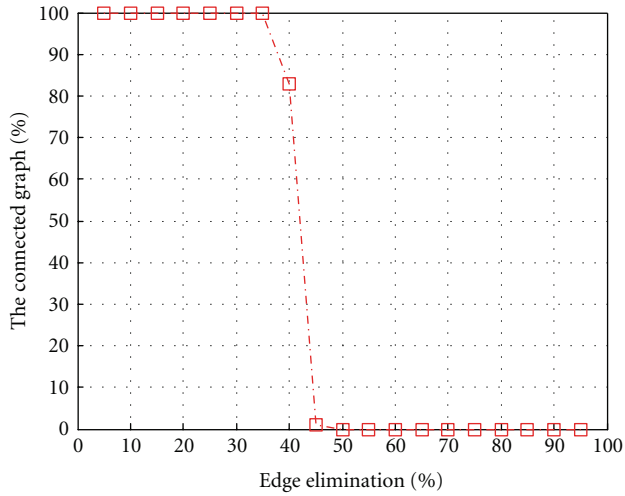


Figure 10: Connected graph in case with edge elimination.

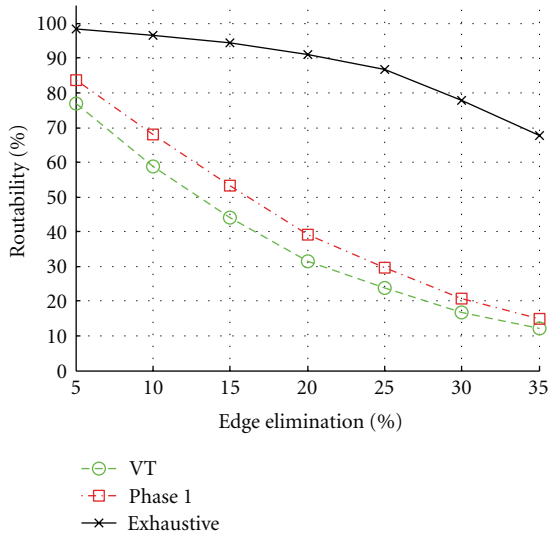


Figure 11: Routability with 1081 BCG after edges are eliminated.

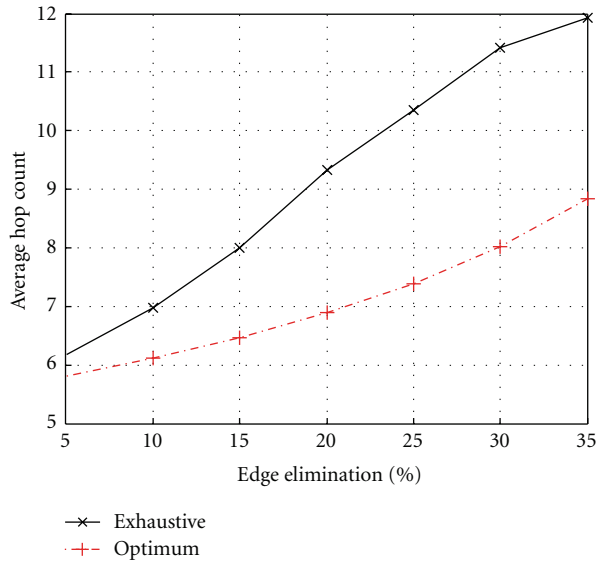


Figure 12: Average hop count with 1081 BCG after edges are eliminated.

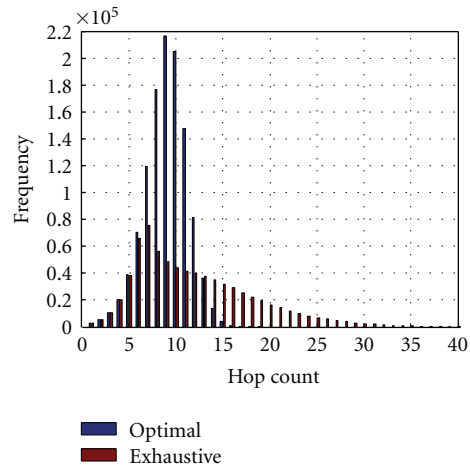


Figure 13: Hop count distribution of BCG with 1081 nodes after 35% edge elimination. Note that hop counts exceeding 40 are not shown for brevity.

length are important parameters. We used the network model described in Section 3 and removed a randomly chosen edge each time unit according to the edge failure rate. Simulation running time is 1000000 ticks that is ten times longer than previous one because more time is needed to observe the effects of link failures. The evaluation was done in terms of the reachability according to packet generation rate, edge failure rate, and buffer length.

Table 4 shows reachability, the average ETE delay, and average occupied buffer length with infinite buffer and 0.0005 edge failure rate. Exhaustive routing produced the highest reachability but also the longest End-to-End delay. End-to-End delay does not consider nonreached packets.

Figure 14 shows reachability as a function of packet generation rate from 0.05 to 0.25. With 0.05 packet generation

Table 4: Routing comparison with infinite buffer, 0.05 packet generation rate, and 0.0005 edge failure rate.

	Reachability	ETE delay	Buffer length
VT	56.97%	6.14	0.23
Phase 1	60.7175%	6.23	0.19
Exhaustive	92.75%	9.19	0.43

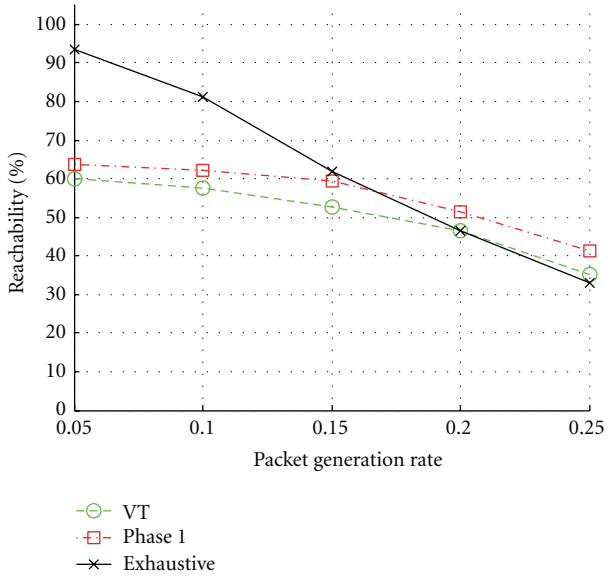


Figure 14: Comparison of fault-tolerant routing reachability with buffer length 5 and 0.0005 edge failure rate according to packet generation rate.

rate, Exhaustive routing produced 56% and 47% better reachability than VT routing and Phase 1, respectively. However, as packet generation rate increases, the performance becomes similar because of network capacity and overflowed packets. With 0.25 packet generation rate, Phase 1 routing shows 26% better reachability than Exhaustive routing because Exhaustive routing has a longer average hop count to support higher reachability in static analysis. It increases occupied buffer length and packets are dropped. Phase 1 routing shows better reachability than VT routing even at high packet generation rate. Phase 1 routing does not try to send packets by Dynamic Routing Table when there is no shortest path extracted from BCG. So packets guaranteed shortest path length existing in the network. Also, the occupied buffer length of Phase 1 is smaller than one of VT.

We investigated reachability in response to edge failure rate with 0.05 packet generation rate as shown in Figure 15. We changed edge failure rate, which means that frequent of edge failure changes and the number of fault edges is changed at the end of simulation. The total edge number of a degree-4 BCG with 1081 nodes is 2162 and at the edge failure rate of 0.0005, 0.00075, 0.001, 0.00125, and 0.0015, the expected number of eliminated edges at the end of simulation are 500, 750, 1000, 1250, and 1500, respectively. Regardless of edge

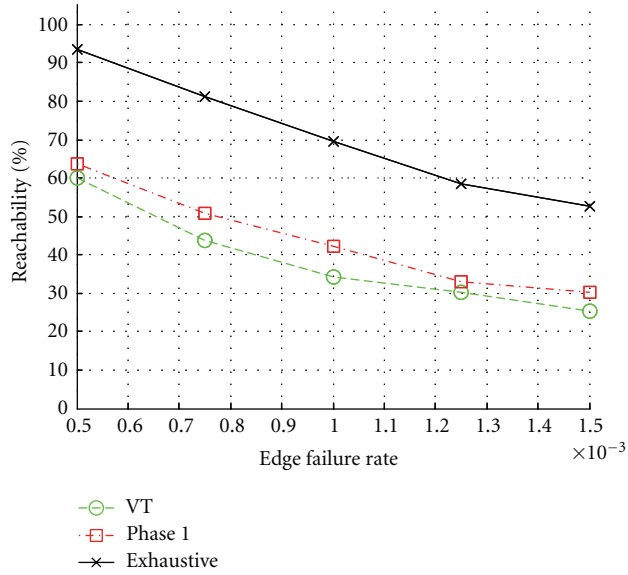


Figure 15: Comparison of fault-tolerant routing reachability with buffer length 5 and 0.05 packet generation rate according to edge failure rate.

failure rate, Exhaustive routing shows 50% to 100% better reachability. Reachabilities of VT routing, Phase 1 routing, and Exhaustive routing are decreased by 42%, 47%, and 56%, respectively, as edge failure rate increases from 0.0005 to 0.0015.

7. Conclusions

In networks, communication failures are possible scenarios. The existing vertex-transitive routing for Borel Cayley graphs, described in Section 2, cannot efficiently tolerate node/link failures. We proposed a fault-tolerant routing algorithm, the "Exhaustive routing," that uses an identical routing table at each node and exploits multiple shortest paths.

Exhaustive routing has two phases. In Phase 1, packets are routed through the shortest path existing in the Borel Cayley graph. When all shortest paths from the BCG are disconnected, Phase 2 of Exhaustive routing is used to exploit possible paths besides the shortest paths in the Borel Cayley graph. Through simulation, we found that the proposed Exhaustive routing showed 20% to 350% better routability than that of the vertex-transitive routing with various amount of link failures. Regarding the average hop count, Exhaustive routing showed just 30% longer than the Optimal routing. We also showed good network topology properties of Borel Cayley graph comparing with torus and de Bruijn graphs with various traffic patterns.

When we consider simultaneous multiple packets flowing, Exhaustive routing showed over 50% better reachability with certain packet generation rate. However, with high packet generation rate, Phase 1 routing showed better reachability because it sends packets along the guaranteed shortest paths of the original Borel Cayley graph. In summary,

Exhaustive routing has good reachability and small average hop counts. Our proposed fault-tolerant routing algorithm makes it possible for Borel Cayley graphs to be deployed in realistic network scenarios.

Acknowledgments

The authors are partially supported by the National Science Foundation under Grants nos. CNS 0829656 and IIP 0917956. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] K. N. Sivarajan and R. Ramaswami, "Lightwave networks based on de Bruijn graphs," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 70–79, 1994.

[2] K. Zhu and B. Mukherjee, "Traffic grooming in an optical WDM mesh network," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 1, pp. 122–133, 2002.

[3] R. Duncan, "Survey of parallel computer architectures," *Computer*, vol. 23, no. 2, pp. 5–16, 1990.

[4] R. D'Andrea and G. E. Dullerud, "Distributed control design for spatially interconnected systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 9, pp. 1478–1495, 2003.

[5] J. Sun and E. Modiano, "Capacity provisioning and failure recovery for Low Earth Orbit satellite constellation," *International Journal of Satellite Communications and Networking*, vol. 21, no. 3, pp. 259–284, 2003.

[6] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 71–121, 2006.

[7] H. Moussa, A. Baghdadi, and M. Jézéquel, "Binary de Bruijn interconnection network for a flexible LDPC/turbo decoder," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 97–100, May 2008.

[8] A. El Gamal, "Trends in CMOS image sensor technology and design," in *Proceedings of the IEEE International Devices Meeting (IEDM '02)*, pp. 805–808, December 2002.

[9] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, "An empirical investigation of mesh and torus NoC topologies under different routing algorithms and traffic models," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD '07)*, pp. 19–26, IEEE Computer Society, Washington, DC, USA, August 2007.

[10] E. Noel and W. Tang, "Novel sensor MAC protocol applied to cayley and manhattan street networks with cross bow MICA2," in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON '06)*, vol. 2, pp. 626–631, September 2006.

[11] A. A. Taleb, J. Mathew, and D. K. Pradhan, "Fault diagnosis in multi layered De Bruijn based architectures for sensor networks," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM '10)*, pp. 456–461, April 2010.

[12] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys &Tutorials*, vol. 7, no. 2, pp. 72–793, 2005.

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computers Communications (ACM SIGCOMM '01)*, pp. 149–160, ACM, New York, NY, USA, August 2001.

[14] M. F. Kaashoek and D. R. Karger, "Koorde: a simple degree-optimal distributed hash table," in *Proceedings of the International Peer-to-Peer Symposium (IPTPS '03)*, 2003.

[15] M. Naor and U. Wieder, "Novel architectures for P2P applications: the continuous-discrete approach," *ACM Transactions on Algorithms*, vol. 3, no. 3, article 34, 2007.

[16] D. Loguinov, J. Casas, and X. Wang, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1107–1120, 2005.

[17] C. Qu, W. Nejdl, and M. Kriesell, "Cayley dhts—a group-theoretic framework for analyzing dhts based on cayley graphs," in *Proceedings of the International Symposium on Parallel and Distributed Processing and Applications (ISPA '04)*, Springer, 2004.

[18] G. Barrenetxea, B. Berefull-Lozano, and M. Vetterli, "Lattice networks: capacity limits, optimal routing, and queueing behavior," *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 492–505, 2006.

[19] S. B. Akers and B. Krishnamurthy, "Group-theoretic model for symmetric interconnection networks," *IEEE Transactions on Computers*, vol. 38, pp. 555–566, 1992.

[20] K. W. Tang and B. W. Arden, "Representations of borel cayley graphs," *SIAM Journal on Discrete Mathematics*, vol. 6, pp. 655–676, 1993.

[21] J. C. Bermond, C. Delorme, and J. J. Quisquater, "Strategies for interconnection networks: some methods from graph theory," *Journal of Parallel and Distributed Computing*, vol. 3, no. 4, pp. 433–449, 1986.

[22] M. Miller and J. Siran, "Moore graphs and beyond: a survey of the degree/diameter problem," *Electronic Journal of Combinatorics*, vol. 14, 2009.

[23] G. Panchapakesan and A. Sengupta, "On a lightwave network topology using Kautz digraphs," *IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1131–1137, 1999.

[24] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 313–356, 2002.

[25] K. W. Tang and B. W. Arden, "Vertex-transitivity and routing for Cayley graphs in GCR representations," in *Proceedings of the ACM/SIGAPP Symposium on Applied Computing (SAC '92)*, pp. 1180–1187, ACM, New York, NY, USA, March 1992.

[26] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[27] J. Yu, E. Noel, and K. W. Tang, "A graph theoretic approach to ultrafast information distribution: borel Cayley graph resizing algorithm," *Computer Communications*, vol. 33, no. 17, pp. 2093–2104, 2010.

[28] J. W. Mao and C. B. Yang, "Shortest path routing and fault-tolerant routing on de Bruijn networks," *Networks*, vol. 35, no. 3, pp. 207–215, 2000.

[29] C. T. Ho and L. Stockmeyer, "A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers," *IEEE Transactions on Computers*, vol. 53, no. 4, pp. 427–438, 2004.

[30] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 848–864, 1995.

[31] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[32] B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*, Kluwer Academic, Norwell, Mass, USA, 1999.

[33] Z. Feng and O. W. Yang, "Routing algorithms in the bidirectional de Bruijn graph metropolitan area networks," in *Proceedings of the Military Communications Conference (MILCOM '94)*, vol. 3, pp. 957–961, IEEE, October 1994.

[34] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.