Design and Analysis of Structured Graph based Wireless Sensor Networks

A Dissertation Presented

by

Jung Hun Ryu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2012

Stony Brook University

The Graduate School

Jung Hun Ryu

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree,

hereby recommend acceptance of this dissertation.

K. Wendy Tang, Advisor of Dissertation
Associate Professor, Department of Electrical and Computer Engineering

Thomas Robertazzi, Chairperson of Defense
Professor, Department of Electrical and Computer Engineering

Carlos Gamboa, Adjunct Professor
Department of Electrical and Computer Engineering

Yanni Ellen Liu, Assistant Professor
Department of Computer Science

Eric C. Noel, Co-advisor of Dissertation
AT&T Laboratories

This dissertation is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Design and Analysis of Structured Graph based Wireless Sensor Networks

by

Jung Hun Ryu

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2012

Wendy Tang, Dissertation Advisor

Eric Noel, Dissertation Co-Advisor

In dense wireless sensor networks, a sensor node may have several neighbor nodes within radio range. In this case, efficient network design is more challenging than with sparse networks. Researches on efficient network design for dense networks are often based on random networks and include comprehensive network performance analysis. Random networks have many advantages for wireless sensor networks, such as short-distance radio range, flexibility for nodes to join or leave the network, and self-organization. However, there are also disadvantages inherent to random networks such as difficulty to guarantee global properties such as diameter and connectivity. On the other hand, there have not been a lot of research on the use of structured graphs for wireless sensor networks. This observation motivated us to investigate the benefits and limitation of applying structured graphs on wireless sensor networks and to take advantage of such graphs' guaranteed global properties.

We focus on Borel Cayley graphs (BCG) as a family of structured graph, which have been shown to be an efficient candidate as a logical topology for sensor networks. BCGs are known to have small diameters and average path lengths. Also, BCGs are symmetric graphs, a property that enables distributed routing. Even though BCGs have these favorable properties, there are practical limitations in applying BCGs to wireless sensor networks. One of them is the possible existence of long-distance communication edges between logically connected sensors since BCG connections are not based on nodes' geographical positions. Connections of BCGs look like pseudo-random connections. It is this pseudo-randomness that enables BCGs to have a small diameter and a short average path length between nodes. However, when overlaying a BCG on a sensor network, such pseudo-randomness makes it possible to produce long-distance connections. Another practical limitation is the lack of fault-tolerant routing algorithms: existing BCG routing algorithms do not account for node or communication link failures.

In this dissertation, we present results on the practical realization of BCGs as structured graphs for wireless sensor networks. In particular, we present node ID assignment algorithms that allow topology construction of BCGs with consideration of the nodes' geographical distribution; and also present a fault tolerant routing algorithm, accounting for node or communication link failures.

Our node ID assignment algorithms assign node IDs to a wireless sensor network based on a pre-defined BCG connection rule. The goal of our algorithms is either to (1) reduce the average physical communication distance based on the geographical positions of the network nodes, assuming all nodes are within communication distance of each other; or (2) to assign IDs that enables the network topology to represent most but not necessarily a complete representation of the BCG due to out of range nodes. With our fault tolerant routing algorithm, the routing tables of nodes are updated distributively in response to node/link failures. We quantify the performance of the routing algorithm by considering packet reachability and End-to-End delay for different levels of communication link failures and packet generation.

From our research, we show that with our proposed node ID assignment and fault tolerant routing algorithm, BCGs are good candidates as structured graph topologies for wireless sensor networks.

Contents

1 Introduction **1**

1.1 Wireless Sensor Networks 1

1.2 Routing Algorithms in Wireless Sensor Networks 2

1.3 Topology Control for Wireless Sensor Networks 4

1.4 Structured Graph based Networks 6

1.5 Our Contributions 6

1.6 Organization of Thesis 7

2 Preliminary **8**

2.1 Borel Cayley Graph 8

2.2 Routing for Borel Cayley Graph 12

3 Node ID Assignment for Borel Cayley Graph **16**

3.1 Introduction 16

3.2 Problem Statement 17

3.2.1 Average communication distance minimization when the radio range covers the deployment area 17

3.2.2 Completing communication edges with finite radio range . 19

3.3 Simulated Annealing based Node ID Assignment 20

3.4 Chordal Ring Based Node ID Assignment 23

3.4.1 CR representation and node ID conversion between GCR and CR 23

3.4.2 Algorithm 25

3.4.3 Average communication distance 27

3.5 Distributed Node ID swapping Assignment 30

3.5.1 Terminologies 30

3.5.2 Assumption 31

3.5.3 Algorithm 31

3.5.4 Example 32
3.5.5 Completeness of resultant graph 34
3.6 Data Dissemination 36
3.6.1 Consensus protocol 36
3.6.2 Simulation setup 37
3.6.3 Comparison between BCG-0 and BCG-1 39
3.6.4 Comparison between our proposed topology control and existing topology controls 39
3.6.5 Consensus protocol analysis with considering interferences 44

4 Distributed and Fault-tolerant Routing for Borel Cayley Graph 50
4.1 Introduction 50
4.2 Topology Comparison 51
4.2.1 Network model 51
4.2.2 Topologies 52
4.3 Data Structure of Exhaustive Routing 58
4.3.1 Static Routing Table 59
4.3.2 Dynamic Routing Table 60
4.4 Phases of Exhaustive Routing 61
4.4.1 Phase 1 61
4.4.2 Phase 2 62
4.5 Simulation 65
4.5.1 Static performance analysis 66
4.5.2 Dynamic performance analysis 69
4.5.3 Comparison Exhaustive routing with AODV 72

5 Conclusion and Future Research 77
5.1 Contribution 77
5.2 Future Research 79

List of Figures

2.1 Examples of generalized chordal ring and chordal ring. 9
2.2 Example of a routing table for Borel Cayley Graph. 13
2.3 Example of Vertex-transitive routing from node 0 to node 16. . . . 14

3.1 Graph representation with radio range enough to cover whole deployment area. 18
3.2 Graph representation with finite radio range. 19
3.3 Illustration of matrix M when node ID 0 and 4 are swapped. . . . 22
3.4 Examples of generalized chordal ring and chordal ring. 24
3.5 Illustration of CR assignment. Note that each alphabet represents a physical sensor node and a dashed line represents distance. . . . 26
3.6 Average communication distance of resultant graphs by node ID assignments in the case of sufficient radio range to cover whole sensor deployment area. 28
3.7 Histograms of communication distance by the proposed algorithms with $N = 272$ and a solid line shows polynomial fit line. 29
3.8 Host graph used as example of Dist-swap assignment. 33
3.9 Edge construction percentage with distributed node ID assignment. 34
3.10 Connected network percentage with node ID assignments. 35
3.11 Example of status agreement with consensus protocol on BCG. . . 36
3.12 Histograms of communication distance by the proposed algorithms with $N = 1081$. 40
3.13 Power consumption comparison between BCG-0 and BCG-1. . . 41
3.14 BCG-0, BCG-3 with radio range 40m, Max with 25m, E-MST, K-Neigh with $k = 10$ and 25m, Gabriel Graph. $N = 272$ 42
3.15 Comparison of diameters. 43
3.16 Comparison of average path lengths. 43
3.17 Comparison of consensus steps. 44
3.18 Comparison of power model 0 for consensus protocol. 44

3.19 Comparison of power model 1 for consensus protocol. 45
3.20 Collision percentage as a function of packet generation rate. . . . 47
3.21 Comparison of consensus step for STDs of 0.25 and 0.1 with 0.005 packet generation rate. 48
3.22 Comparison of consensus step, packet collision and power consumption with various packet generation rates and 30m radio range. 49

4.1 Node model. 51
4.2 Toroidal mesh network and de Bruijn graph. 52
4.3 End-to-End Delay with infinite buffer. 55
4.4 Reachability with buffer length 10. 56
4.5 Reachability with buffer length 5. 57
4.6 Original Vertex-transitive routing table and Static Routing Table of BCG in Figure 2.1(a). Note that some part of routing tables are omitted for brevity. 60
4.7 Exhaustive routing algorithm flow. 64
4.8 Network to illustrate a loop of Exhaustive routing. The dot line between nodes indicates that the communication is disconnected . 65
4.9 Connected BCG percentage as a function of edge elimination. . . 66
4.10 Routability with 1081 BCG after edges are eliminated. 67
4.11 Average hop count with 1081 BCG after edges are eliminated. . . 68
4.12 Hop count distribution of BCG with 1081 nodes after 35% edge elimination. Note that hop counts exceeding 40 are not shown for brevity. 68
4.13 Reachabilities of routings with a packet generation rate of 0.05 and an Edge failure rate of 0.001 by simulation ticks. 70
4.14 Comparison of fault-tolerant routing reachability with buffer length 5 and 0.0005 edge failure rate according to packet generation rate. Note that the result is accumulated during the whole simulation time. 70
4.15 Comparison of fault-tolerant routing reachability and power consumption with buffer length 5 and 0.05 packet generation rate according to edge failure rate. Note that the result is accumulated during the whole simulation time. 71
4.16 Reachabilities of AODV and Exhaustive routing according to packet generation rate. 73
4.17 End-to-End Delay of AODV and Exhaustive routing according to packet generation rate. 74

4.18 Reachabilities of AODV and Exhaustive routing according to edge failure rate with 0.001 packet generation rate. 74
4.19 Reachabilities of AODV and Exhaustive routings as a function of packet generation rate with interferences. 75
4.20 Reachabilities of Exhaustive routing function of packet generation rate and with interferences. 75
4.21 Reachabilities of Exhaustive routing function of edge failure rate and considering interference. 76

List of Tables

2.1 Parameters of Borel Cayley Graphs used in this thesis. 12
2.2 Vertex-transitive routing algorithm for Borel Cayley Graphs. . . . 14
2.3 A summary of Vertex-transitive routing example. 15

3.1 Dist-swap assignment operation after a node receives the Token packet. 32
3.2 Dist-swap assignment operation after a node receives the Info packet. 33
3.3 Dist-swap assignment operation after a node receives the Swap packet. 33
3.4 Summary of our proposed algorithms. 35
3.5 Radio model parameters. 38
3.6 Consensus protocol performance after 10000 ticks with 30m initial radio range and a packet generation rate of 5×10^{-3}. 46

4.1 Mesh networks parameters. 54
4.2 Static property. 54
4.3 Summary of routing tables for Exhaustive routing. 61
4.4 Routing comparison with infinite buffer, 0.05 packet generation rate, and 0.0005 edge failure rate. Note that the result is accumulated during the whole simulation time. 69

Chapter 1

Introduction

1.1 Wireless Sensor Networks

An ad-hoc wireless sensor network (WSN) is a self-organized and distributed network consisting of a large number of small and light sensor nodes [1,2]. A sensor node includes a processor, a wireless radio, and various sensors to monitor and sense environmental parameters such as temperature, moisture, pressure, etc. In a WSN, sensor nodes interchange information and collaborate with each other to achieve a common mission.

The flexibility, fault tolerance, high sensing fidelity, low cost, and rapid deployment characteristics of sensor networks create many new and exciting application areas for remote sensing [3]. Ad-hoc wireless sensor networks applications include building monitoring [4], environmental sensing [5–7], traffic monitoring [8], and surveillance [9]. This has been enabled by the availability, particularly in recent years, of sensors that are small, cheap, and intelligent.

Any realization of sensor networks must satisfy the constraints introduced by factors such as fault tolerance, scalability, cost, hardware, topology change, environment, and power consumption. Lack of any centralized control and possible node mobility give rise to many issues at the network, medium access, and physical layers, which have no counterparts in wired networks such as Internet or in cellular networks.

Selecting efficient algorithms such as routing, topology control, and information dissemination for WSNs depends on the application goal and its operating environment. Hardware specification of sensor nodes varies [10]. Each sensor has limited battery or relatively long-term power supplies. Also transmission power, modulation schemes, and data transmission rates vary from vendor to vendor. A wireless sensor node needs various storage such as Random Access Memory (RAM) to store application related data and Read Only Memory (ROM) to store program code, which needs to be stored permanently. Usable routing table size in sensors affects the design of routing algorithms. Depending on the applications, the number of nodes in WSNs ranges from hundreds to thousands, and nodes can be static or mobile. The moving speed of nodes also affects routing algorithm performance. Clearly, efficient algorithms should be different for each application.

1.2 Routing Algorithms in Wireless Sensor Networks

In wired networks, link state and distance vector protocols are mostly used. While in link state routing protocols nodes flood the network with local information, distance vector routing protocols exchange network-wide information only with local nodes. These protocols, although scalable for a large number of nodes, are computationally expensive for wireless sensor nodes and introduce a large amount of overhead. In wireless sensor networks, their applications are limited to small networks.

Different approach routing algorithms exist to support wireless sensor networks [11, 12]. Flooding is a common technique that can be used for routing in sensor networks. In flooding, each packet received by a node is broadcasted to its neighbors; unless the receiving node is the destination or the packet has reached its maximum hop limit. Flooding is a reactive technique, and it does not require costly topology maintenance and complex route discovery algorithms. However, it has several deficiencies such as duplicated messages, redundant messages generation, and resource blindness without energy awareness [13].

A derivation of flooding algorithm is gossiping in which nodes do not broadcast but send the incoming packets to a randomly selected neighbor [14]. A sensor node randomly selects one of its neighbors to send the data. Once the neighbor node receives the data, it randomly selects another sensor node. Rumor routing is another strategy to reduce the overhead of flooding mechanisms. Rumor routing is based on the idea that two random lines in a square have a relatively high probability to intersect. As such, the protocol sends service agents to set routing and event information in all the nodes that they visit. When one of the nodes in the network detects an important sensing data, another service agent is generated that goes through the network until it intersects with one of the nodes that has been previously embedded with routing and event information.

Data centric routing is based on the following idea: Routing, storage, and querying mechanisms can be performed more efficiently if communication is based on application data instead of the traditional IP global identifiers [13, 15]. This data centric approach to routing provides additional energy savings derived from the communication overhead of binding identifiers that are no longer needed.

Besides the above routing algorithms, there are classic routing algorithms that exploit wireless sensor network characteristics: Dynamic Source Routing in Ad Hoc Wireless Networks (DSR) [16] and Ad-hoc On-Demand Distance Vector Routing (AODV) [17]. These routing protocols create routes only when required by the source node. When a node requires a route to a destination, it initiates a route discovery process within a network. Each node receiving the information caches the route information and uses it for future packets toward the destination.

Some wireless sensor networks have the unique geographical information for each node from the global positioning system (GPS). Geographical routing uses nodes' geographical locations to determine how packets are routed. As a result, nodes do not need to make complex computations to find the next hop since routing decisions are made based on local information. Moreover, because of locality, nodes do not need to maintain large data structures. Geographical routing reduces the communication overhead substantially because routing table advertisements are not needed. Greedy routing, a geographical routing, was originally proposed in [18]. In greedy routing, a message is forwarded to the neighboring node that is closest to the destination. It assumes that the network is sufficiently dense, nodes know their own location and their neighbors' locations, and multi-hop forwarding is reliable.

The performance of geographical routing relies on topology construction algorithms. Face routing requires a planar graph as the topology of a network [19][1]. A message is routed along the interior of a plane (face) of the communication graph. When a message crosses the source-destination line, it moves to an adjacent plane. In this case, the efficient construction of planar graphs is a critical issue. Several planar graphs have been proposed: the Relative Neighborhood Graph [20], the Gabriel Graph [21], and the Delaunay Graph [22]. Face routing and Greedy routings have limitations. When there is no node close to the destination, Greedy routing gets stuck although an alternative path exits through other nodes. Face routing can also get stuck when there are errors in measuring the sensor's position.

In [23], the authors introduce and analyze the concept of virtual coordinates

[1]A planar graph is a graph that can be embedded in the plane without graph edges crossing

for routing. They convert physical positions to virtual positions to circumvent stuck packets. More recently, Matthew Caesar et al. propose to use virtual position not related to the sensor's physical position [24]. They use Virtual Ring Routing (VRR), inspired by overlay routing algorithms in Distributed Hash Tables (DHTs). It is implemented directly on top of the link layer. VRR provides both traditional point-to-point network routing and DHT routing to the node responsible for a hash table key. The nodes are given arbitrary identifiers and organize themselves into a virtual ring in the increasing order of their IDs cyclically.

Landmark concept is used for routing protocols, similar to how human uses landmarks for routing [25, 26]. They divide the map into several parts with landmarks. Routing takes place between divided parts and then routing takes place inside the part that contains the destination.

Another approach is cluster-based routing to reduce routing complexity and therefore make routing more scalable and reduce energy consumption. In this category, one of routing protocols is Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol [27]. LEACH is a distributed algorithm by which nodes group together in clusters and select themselves as cluster heads. Since cluster heads perform more energy consuming functions than the rest of the nodes, LEACH not only changes them periodically but it also makes sure that cluster heads are selected in a fair and uniform manner, so energy is consumed in an even manner. Once a node is selected as a cluster head, the algorithm makes sure that it has a low probability of becoming a cluster head again. Nodes only communicate with their respective cluster head.

1.3 Topology Control for Wireless Sensor Networks

Some WSN applications require very dense networks. Hundreds to several thousands of nodes can be deployed throughout a sensor field. For example, some machine diagnosis applications use up to 3000 nodes in a 100m by 100m area [28], or sensors can be deployed within tens of feet of each other for object tracking [15]. The topology of a large and dense sensor network has direct impact on its performance. For example, topology control algorithms are essential in reducing energy consumption and radio interference [29]; thus expanding the network's lifetime.

Our goal is to control the topology of the graph representing the communication links between network nodes in order to maintain some of the graph's properties such as network robustness and small diameter. For sparse wireless sensor networks, nodes within communication range are viewed as connected neighbors.

On the other hand, for densely populated wireless sensor networks, the number of neighbors (or degree) of sensor nodes should be small to save battery power. However, when the nodal degree is limited, a network can become disconnected if proper neighbors are not chosen.

To achieve a fully connected ad hoc network, there must be at least one wireless multi-hop path between any node pairs. Therefore. the connectivity depends on the number of nodes per unit area (node density) and their radio transmission range [30]. Each single node contributes to the connectivity of the entire network. In ad hoc network, the nodes' transmission power determines the network topology. When the transmission power of a node is increased, a longer transmission range results, which in turn increases the possible number of nodes to be directly connected. Conversely, when a node's transmission power is too low, it can become isolated in the network [31, 32].

According to [33], topology control algorithms can be first classified as homogeneous and nonhomogeneous approaches. In the former case, nodes are assumed to use the same transmission range, and the topology control problem reduces to one of determining the minimum radio range such that a certain network-wide property is satisfied. In the latter case, nodes are allowed to choose different transmission ranges.

Nonhomogeneous topology control algorithms can be classified as location-based, direction-based, and neighbor-based approaches. In all these approaches, a network topology is formed in an ad-hoc manner based on sensor nodes' location, direction or some sort of neighbor ordering. Examples of neighbor-based approaches are K-Neigh and XTC [34, 35]. The goal of K-Neigh is to keep the number of neighbors of a node equal to, or slightly below a given value k. A protocol that shares many similarities with K-Neigh is the XTC protocol presented in Wattenhofer and Zollinger [35]: the neighbors of a node u are ordered according to some metric (e.g., distance or link quality), and based on a simple rule, u decides which nodes are kept as immediate neighbors in the final network topology. However, there is no guarantee on the topology's overall properties such as a bounded diameter and average path length.

Our approach is to impose a logical topology on a physical network. Then the resulting network is connected and has a small constant degree and a short diameter. The advantage of imposing a logical topology is that computing optimal routes is independent of the physical topology and generates little message overhead. Moreover, a sparse communication graph requires little maintenance in the presence of node mobility.

1.4 Structured Graph based Networks

Structured graphs have been studied for a long time and are good candidates for interconnection networks [36, 37]. Various graph based interconnection networks have been applied to wavelength division multiplexed optical networks [37, 38], distributed parallel computation [39], distributed control [40], satellite constellations [41], and chip design [42–44]. In peer-to-peer (P2P) overlay network schemes, structured graphs are investigated and compared against unstructured P2P overlay networks [45]. As an example of structured P2P overlay networks, k ring lattices are used in Chord [46] and de Bruijn graphs are used in Koorde and Distance Halving [47, 48]. Also, there are theoretical results on de Bruin and Cayley graphs applied to P2P [49, 50].

Graph based wireless sensor networks have also been explored by other researchers [24, 51–55]. In general, a predefined graph topology with a deterministic connection rule facilitates performance analysis. In addition, some graph topologies have advantageous properties for WSNs such as: symmetry, hierarchy, hamiltonicity, constant low-degree, and distributed routing protocols.

1.5 Our Contributions

We focus on Borel Cayley graphs (BCG), a member of the Cayley graph family [56]. In [57, 58], Borel Cayley graphs have been shown to be efficient logical topologies for dense WSNs. BCGs are known to have small diameters, short average path lengths, and low constant degree connections. Moreover, BCGs are symmetric graphs, a property that enables distributed routing [59].

Consensus protocol has been used to quantify how fast a network is capable of distributing information [60, 61]. It is a distributed node-to-node message exchange rule to drive nodes to an agreement for a quantity of interest. BCGs showed good performance for consensus protocol when compared to mesh, torus, and small world networks [62]. Also, because of its dense property (small degree and diameter), BCG can be used in interconnection networks [63, 64], wavelength division multiplexed optical networks [65], and VLSI layout design [66]. More recently, efforts to circumvent inherent limitations of Borel Cayley graphs, such as choosing a good generation set and rendering size more flexible have been published in [67, 68]. Through these efforts and our following contributions, BCGs are becoming a feasible solution for wireless sensor networks.

Topology Control for Borel Cayley Graphs We propose topology control for Borel

Cayley structured graphs applied to wireless sensor networks. A structured graph is constructed by connection rules with identifiers. Connection rules are not related directly to geographical location. Without a proper algorithm to assign IDs to sensors, there could be unnecessary long-distance communication edges. Therefore, we propose reducing average communication distance of BCGs overlaid on WSNs. We also consider how to maximize a partial BCG topology in the limited radio range scenario.

Distributed and Fault-tolerant Routing for Borel Cayley Graphs We propose a fault-tolerant routing algorithm for BCG. Routing performance depends on traffic patterns (many-to-one data collection, one-to-many data dissemination, point-to-point routing), traffic load (frequency of message delivery), expected network operational time, and data packet size and format. Sensors can be disabled due to energy depletion, and communication links can be interrupted by obstacles. In case of sensor failure or communication fault, we quantify routing performance as a function of the fault rate in terms of reachability and End-to-End delay.

1.6 Organization of Thesis

The rest of this thesis is organized as follows. In Chapter 2, we review the basic concept and definitions for BCGs and related terminology. In Chapter 3, we present our node ID assignment algorithms (Topology Control) for BCGs to reduce communication distance or improve completeness of connections. In Chapter 4, we show our fault tolerant routing algorithm when edges fail. Conclusions and future work are presented in Chapter 5.

Chapter 2

Preliminary

2.1 Borel Cayley Graph

In this chapter, we review the definitions of the generalized chordal ring, Cayley graphs in general, and Borel Cayley graphs in particular. Much of the material in this chapter was taken from Dr. Tang's papers [56, 59, 69].

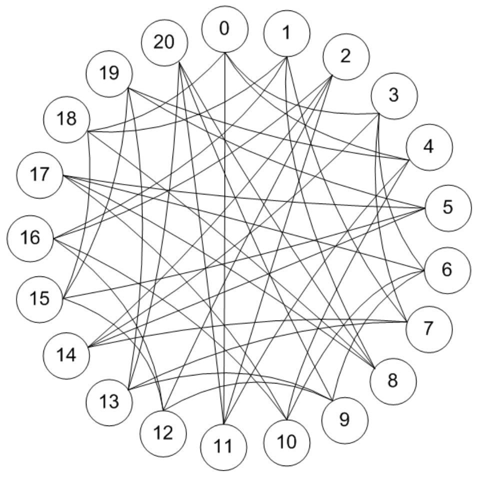
Definition 1 (GCR [69])**.** A graph R is a generalized chordal ring (GCR) if nodes of R can be labeled with integers modulo the number of nodes N, and there exists a divisor q of N such that node i is connected to node j if and only if node $i + q$ (mod N) is connected to node $j + q$ (mod N).

Figure 2.1(a) shows a degree 4 GCR with 21 nodes $V = \{0, 1, 0, \ldots, 20\}$, $q = 3$ classes and the connection rules for node $i \in V$ given by:

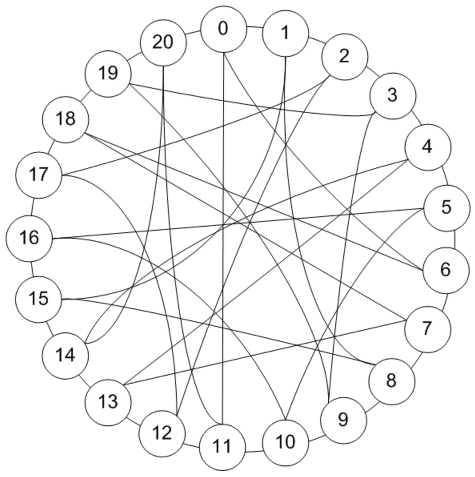
$$\text{if } i \bmod 3 = \begin{cases} \text{“0”: } i \text{ is connected to } i+3, i-3, i+4, i-10 & \pmod{21} \\ \text{“1”: } i \text{ is connected to } i+6, i-6, i+7, i-4 & \pmod{21} \\ \text{“2”: } i \text{ is connected to } i+9, i-9, i-7, i+10 & \pmod{21} \end{cases}$$

The connection rules of elements are defined by connection constants. Connection constants of i and $i + q$ are the same according to the definition. There are four connection constants when the graph is four regular. For example, the 21-node graph used connection constants $+3$, -3, $+4$, and -10 for class 0. We call those constants GCR constants.

A chordal ring (CR) is a special case of GCR, in which every node i has connections to node $i+1$ modulo N and $i-1$ modulo N. In other words, all nodes



(a) Generalized chordal ring.



(b) Chordal ring.

Figure 2.1: Examples of generalized chordal ring and chordal ring.

on the circumference of the ring are connected to form a Hamiltonian cycle. [1]

Figure 2.1(b) shows a degree 4 CR with 21 nodes $V = \{0, 1, 0, \ldots, 20\}$, $q = 7$ classes, and the connection rules for $i \in V$ given by:

$$\text{if } i \bmod 7 = \begin{cases} \text{“0”: } i \text{ is connected to } i+1, i-1, i-10, i+6 & \pmod{21} \\ \text{“1”: } i \text{ is connected to } i+1, i-1, i+7, i-7 & \pmod{21} \\ \text{“2”: } i \text{ is connected to } i+1, i-1, i+10, i-6 & \pmod{21} \\ \text{“3”: } i \text{ is connected to } i+1, i-1, i+6, i-5 & \pmod{21} \\ \text{“4”: } i \text{ is connected to } i+1, i-1, i+9, i+10 & \pmod{21} \\ \text{“5”: } i \text{ is connected to } i+1, i-1, i+5, i-10 & \pmod{21} \\ \text{“6”: } i \text{ is connected to } i+1, i-1, i-6, i-9 & \pmod{21} \end{cases}$$

There, connection constants, or CR constants, for class 0 are $+1$, -1, -10, and $+6$.

Definition 2 (Cayley Graph [56])**.** A graph $C = \mathbb{C}(V, G)$ is a Cayley graph with vertex set V if two nodes $v_1, v_2 \in V$ are adjacent, there exists $g \in G$ such that $v_1 = v_2 * g$, where $(V, *)$ is a finite group and $G \subset V \backslash \{I\}$. 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

[1]Hamiltonian cycle is a graph cycle through a graph that visits each node exactly once.

but does not specify a particular group.

Definition 3 (Borel Subgroup)**.** If V is a Borel subgroup of the general linear 2×2 matrices set, then

$$V = \left\{ \begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix} : x = a^t \,\mathrm{mod}(p),\ y \in Z_p,\ t \in Z_k \right\}, \tag{2.1}$$

where $a \in Z_p \backslash \{0, 1\}$, p is prime, and k is the order of a. That is, k is the smallest positive integer such that $a^k = 1$ (mod p).

Definition 4 (Borel Cayley Graph (BCG) [57])**.** 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 the 2×2 matrix elements of V and with directed edge from v to u if $u = v * g$ where $u \neq v \in V$, $g \in G$ and $*$ is a modulo-p multiplication chosen as a group operation.

As an example, consider a Borel Cayley graph with $p = 7$, $a = 2$, $k = 3$, $t_1 = 0$, $t_2 = 1$, $y_1 = 1$, and $y_2 = 1$ and generator set $G = g_1, g_1^{-1}, g_2, g_2^{-1}$. We assume

$$g_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, g_2 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}. \tag{2.2}$$

Then two directed edges generated by one generator and its inverse are given by :

$$\begin{aligned} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} * g_2 &= \begin{pmatrix} 2 & 2 \\ 0 & 1 \end{pmatrix}, \\ \begin{pmatrix} 2 & 2 \\ 0 & 1 \end{pmatrix} * g_2^{-1} &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

When two nodes are linked by one directed edge, the opposite directed edge is constructed with the inverse generator. So, we only need to consider undirected edges.

Proposition 1. 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 and T is referred to as the transform element.

Proposition 2. All degree-4 Borel Cayley graphs have CR representations.

The proofs of these propositions are given in [69] [56] and not repeated here.

Let a_i be a class representing element. Then to simplify GCR representation we use T and a_i as follows [69]:

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, a_i = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}^i. \tag{2.3}$$

So any vertex $v \in V$ can be represented with T and a_i as follows [56]:

$$v = T^j * a_i = \begin{pmatrix} 1 & j \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a^i & j \\ 0 & 1 \end{pmatrix}. \tag{2.4}$$

Since BCGs are defined over a group of matrices, we define a, a mapping of BCG representation from the group domain to the integer domain. This allows us to take advantage of BCGs' systematic representation of connections to the integer domain, which includes simple routing tables. The node v representation in GCR ($ID_g(v)$) is denoted as follows [69]:

$$ID_g(v) = q * j + i, \tag{2.5}$$

where q is k. As an example, Figure 2.1(a) shows a GCR representation of a Borel Cayley graph in the integer domain.

A Borel Cayley is represented as a GCR which has connection rules expressed in terms of GCR constants. When a Borel Cayley graph is represented in the integer domain, its GCR constants can be calculated by a systematic method. In [69], for any Borel Cayley graph with $n = p \times k$, and $a^k = 1$ (mod p), we assume the generators g_1, g_2, and their inverses to be:

$$\begin{aligned} g_1 &= \begin{pmatrix} a^{t_1} & y_1 \\ 0 & 1 \end{pmatrix}, & g_2 &= \begin{pmatrix} a^{t_2} & y_2 \\ 0 & 1 \end{pmatrix}, \\ g_1^{-1} &= \begin{pmatrix} a^{k-t_1} & -a^{k-t_1}y_1 \\ 0 & 1 \end{pmatrix}, & g_2^{-1} &= \begin{pmatrix} a^{k-t_2} & -a^{k-t_2}y_2 \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{2.6}$$

Using the transform element $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and the representing element of class i, $\begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix}$, we can create a GCR representation of the graph with divisor $q = k$. To understand BCG representation in the GCR domain, lets consider vertex j in class i. Then j is connected to $j + \alpha_i$, $j + \alpha_i^{-1}$, $j + \beta_i$, $j + \beta_i^{-1}$ (mod

Table 2.1: Parameters of Borel Cayley Graphs used in this thesis.

N	p	k	a	t_1	t_2	y_1	y_2
272	17	16	3	1	2	1	1
506	23	22	7	1	7	1	1
1081	47	23	2	1	7	1	1
1474	67	22	3	1	7	1	1
2265	151	15	2	1	7	1	1

n) that corresponds to generators g_1, g_1^{-1}, g_2, and g_2^{-1}, where

$$\begin{aligned}
\alpha_i &= (\langle i+t_1\rangle_q + \langle a^i \times y_1\rangle_p) \times q - i, \\
\alpha_i^{-1} &= (\langle i-t_1\rangle_q + \langle -a^{i-t_1} \times y_1\rangle_p) \times q - i, \\
\beta_i &= (\langle i+t_2\rangle_q + \langle a^i \times y_2\rangle_p) \times q - i, \\
\beta_i^{-1} &= (\langle i-t_2\rangle_q + \langle -a^{i-t_2} \times y_2\rangle_p) \times q - i,
\end{aligned}$$

and $\langle i+t\rangle_q$ denotes $(i+t)(\text{mod } q)$.

From these equations, we can construct a BCG topology in the integer domain with low computation cost and not in the matrix domain.

In Table 2.1, we list the parameter values for all the BCGs used in this thesis. 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, we construct undirected BCGs. We arbitrarily chose parameters t_s and y_s for generators.

2.2 Routing for Borel Cayley Graph

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 makes it possible 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 such property [70–72].

	g_1	$g^{-1}{}_1$	g_2	$g^{-1}{}_2$
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
6	1	0	0	0
7	1	0	0	0
8	0	1	0	0
9	1	0	0	1
10	0	0	1	0
11	0	0	0	1
12	0	1	0	1
13	1	0	1	1
14	1	0	0	0
15	0	1	0	0
16	0	1	1	1
17	1	1	1	0
18	0	1	0	0
19	0	0	1	0
20	0	0	0	1

Figure 2.2: Example of a routing table for Borel Cayley Graph.

Proposition 3. All Cayley graphs are vertex-transitive [59].

As shown in the previous section, every Cayley graph 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 [59], the authors provide a framework for the formulation of the complete symmetry (or vertex-transitivity) of Cayley graphs in the integer domain of GCR representation.

Proposition 4. Consider a Borel Cayley graph in the GCR representation with transform element $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and representing elements of each class i defined by $a_i = \begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix}$. Let $i = m_1q + c_1, j = m_2q + c_2$ 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 $j' = \langle m' + a^{\langle c'-c_1 \rangle_q}(m_2 - m_1) \rangle_p q + \langle c' - c_1 + c + 2 \rangle_q$ [59].

Table 2.2 shows an optimal routing algorithm, Vertex-transitive Routing, that exploits the inherent symmetry of Cayley graphs and uses an identical routing table at any node.

Table 2.2: Vertex-transitive routing algorithm for Borel Cayley Graphs.

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 $a^k = 1$ (mod p). Assume the generators to be g_1, g_2, g_1^{-1} and g_2^{-1}, where

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

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 $j = j'$
```

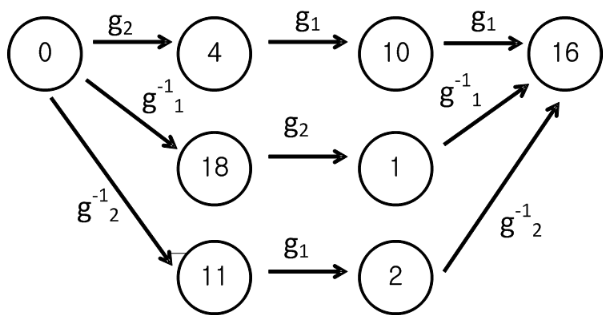


Figure 2.3: Example of Vertex-transitive routing from node 0 to node 16.

Table 2.3: A summary of Vertex-transitive routing example.

$Iteration$	i	j	j'	i'	$link$
0	0	16	16	4	g_2
1	4	16	6	3	g_1
2	3	6	3	3	g_1
3	3	3	-	-	-

As an example, we consider the Borel Cayley graph shown in 2.1(a). At each node, we store the size 20×4 routing table shown in Figure 2.2 where the number of nodes is 20 and the number of generators is 4. Suppose we need to route a message from node 0 to node 16. There are three shortest paths between nodes 0 and 16 as shown in Figure 2.3. The iterations for this example are summarized in Table 2.3.

Chapter 3

Node ID Assignment for Borel Cayley Graph

3.1 Introduction

In this chapter, we propose two node ID assignment algorithms for applying BCGs as logical topologies in WSNs. The first one, the chordal ring node ID assignment algorithm, uses a specific representation (the chordal ring representation) of BCG to assign node IDs of the entire graph. Its goal is to minimize edges associated with long distance radio links. The second algorithm, the distributed node ID swapping assignment, accounts for network nodes' finite radio transmission range. In this case, the imposition of the entire BCG is not always possible.

We will show that the proposed chordal ring node ID assignment algorithm requires a smaller radio range (57% that of the random node ID assignment). When the nodes are constrained to a finite transmission range and the imposition of an entire BCGs is not feasible, the proposed distributed node ID swapping assignment imposes more edges of the original graph (43% more edges in comparison to random ID assignment). There are other node ID assignments related researches that mostly focus on scalability and energy consumption of an ad hoc WSN [73, 74]. Their main goal is to support dynamic and temporary node ID assignment when unicast routing is needed. They do not consider topology controls with node ID's connections.

3.2 Problem Statement

A WSN can be represented as a graph where each node of the graph corresponds to a sensor node and each edge represents a radio connection between nodes. Obviously, for dense WSNs, the number of physical neighbors (nodes within radio range) is large. For ease of description, we call the topology representing the physical neighbors a *host* graph. Besides representing the number of neighbors within communication reach of each other, an edge of a host graph is also weighted by the communication distance or communication cost between nodes.

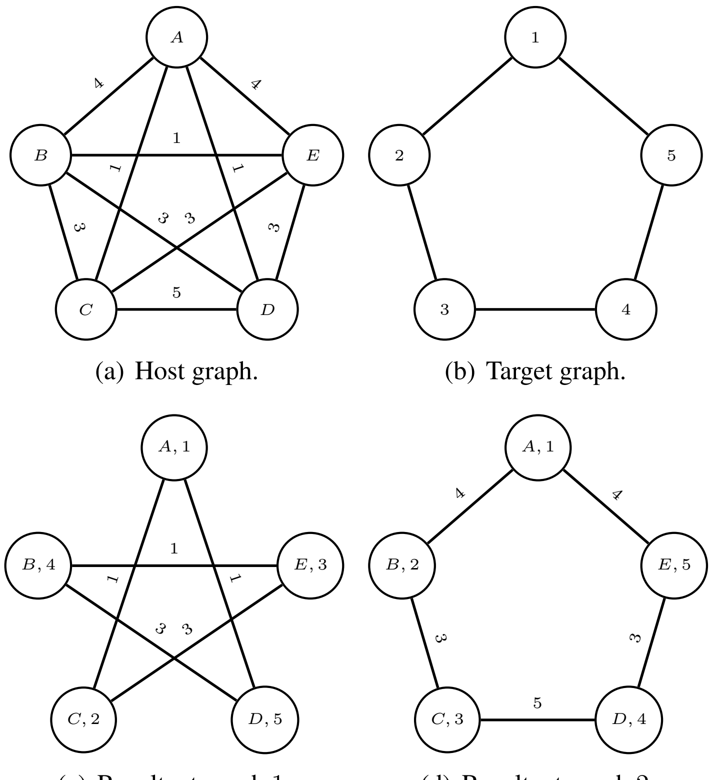
We are interested in the problem of imposing a BCG with a small degree on a dense WSN. We call the BCG topology the *target* graph; and the host graph after the imposition, the *resultant* graph. We consider two cases:

- The radio range of each node covers the whole sensor deployment area. In this case, any node ID assignment always produces a resultant graph with fully connected BCG topology. But depending on a node ID assignment, the communication distances between neighbors vary. So in this case, the goal of our node ID assignment is to reduce the communication distance between connected nodes.

- The radio range does not cover the whole sensor deployment area. Careful node ID assignment is required to produce a resultant graph as similar as possible to the target graph (a BCG topology). In this case, the goal is to find the ID assignment that establishes most communication edges following BCG connection rules.

3.2.1 Average communication distance minimization when the radio range covers the deployment area

Our node ID assignment problem can now be described in terms of finding the assignment that yields the minimum sum of weights of the resultant graph after imposing a target graph onto the host graph. Figure 3.1 illustrates these terminologies. Figure 3.1(a) is the host graph with each node's physical neighbors and weights on the edges that represents the connections communication cost. Figure 3.1(b) is the target graph. Figure 3.1(c) and Figure 3.1(d) show two example resultant graphs, where each graph has a different sum of weights, depending on how node IDs were assigned.



(a) Host graph. (b) Target graph.

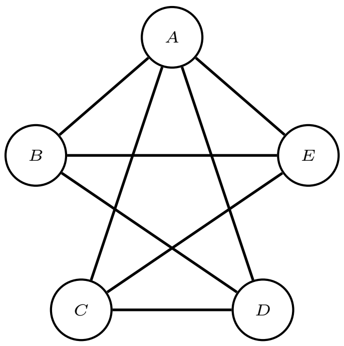
(c) Resultant graph 1. (d) Resultant graph 2.

Figure 3.1: Graph representation with radio range enough to cover whole deployment area.

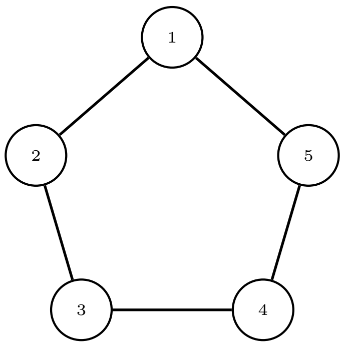
Let a host graph $G = (V, E)$ in the Euclidean space represents the underlying network before applying our proposed methods with V being the set of sensor nodes and E representing the set of communication distances between nodes. We assume that the radio range of each node in the host graph is large enough to cover the whole deployment area (So the host graph is a fully connected graph). The weight of $E(u, v)$, denoted by w_{uv}, represents the Euclidean distance between nodes u and v. The weight between two nodes u and v with coordinates (u_x, u_y) and (v_x, v_y) is computed as

$$w_{uv} = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2}. \tag{3.1}$$

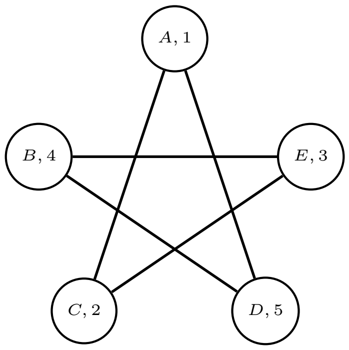
The weights are symmetric because if node u is in the radio range of node v, then



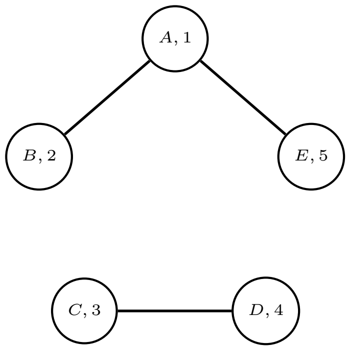
(a) Host graph.



(b) Target graph.



(c) Resultant graph 1.



(d) Resultant graph 2.

Figure 3.2: Graph representation with finite radio range.

node v is also in the radio range of node u. Thus, the edges are undirected.

3.2.2 Completing communication edges with finite radio range

This node ID assignment problem can be described as finding the maximum number of edges of the resultant graph after imposing a target graph onto the host graph. Figure 3.2 illustrates these terminologies. Figure 3.2(a) is the host graph that shows the physical neighbors of each node. Figure 3.2(b) is the target graph. Figure 3.2(c) and Figure 3.2(d) show two different ways to impose the target graph on the host graph where each results in a different number of edges, depending on how node IDs were assigned.

The edge $E(u, v)$ is determined by the radio range and the Euclidean distance between nodes u and v. The distance between two nodes u and v with coordinates

(u_x, u_y) and (v_x, v_y) is w_{uv}. The edge $E(u, v)$ is defined as follows:

$$E(u,v) = \begin{cases} 1 & \text{if } w_{u,v} \leq \text{Radio range} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In the remaining sections, we will propose algorithms to solve these two problems: (a) reducing the average communication distance of BCG based networks and (b) completing as many as possible communication edges that follow the BCG connection rules. Each problem is dealt with by a distributed method. Regardless of the application type, a distributed method is preferred in wireless sensor networks.

3.3 Simulated Annealing based Node ID Assignment

Simulated annealing (SA) is a probabilistic and iterative algorithm used in several domains [75]. We apply simulated annealing to solve our BCG node ID assignment problem (SA assignment). Readers interested in more details on SA and its applications are referred to [76]. Algorithm 1 represents the Simulated Annealing algorithm. We only focus on defining the application dependent functions in SA used for our node ID assignments.

The SA assignment has some known limitations. Its convergence is sensitive to SA parameters t_s, t_e, k_{max}, and temperature decrement interval α defined in Algorithm 1. Note that it is a centralized algorithm that requires positions of sensor nodes, heavy computation load, and a routing protocol for notifying node IDs from a central machine to sensor nodes. However it gives us guidelines for the optimal performance when evaluating our proposed methods.

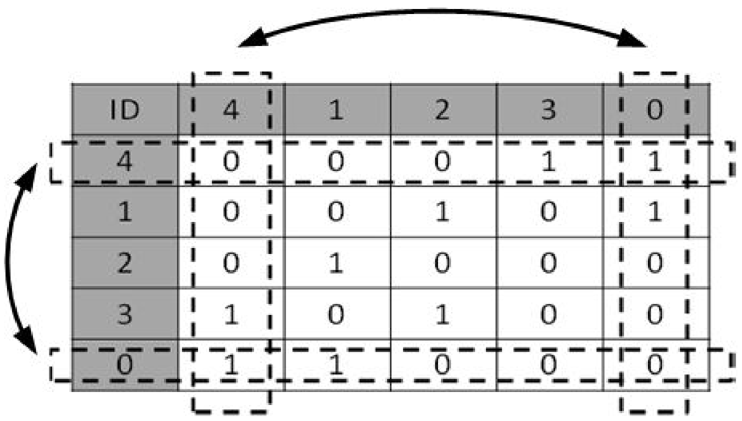
Case of finite radio range

Recall that one of our goals is to minimize the communication distance between nodes when imposing a BCG target graph on a host graph with sufficient long-distance radio range. We denote the weighted adjacency matrix of the host graph by H and the adjacency matrix of the target graph by M. As mentioned previously, a host graph is assumed to be a fully connected graph from which we obtain the distance matrix for all pairs of nodes in a network. Thus, a matrix element h of H is as follows:

$$h_{uv} = w_{uv}, \forall u, v \in V. \tag{3.3}$$

Algorithm 1 Simulated Annealing.

```
Require: $H$: Adjacency matrix of host graph
Require: $M_0$: Adjacency matrix of Borel Cayley graph
 1: procedure SA($H, M_0$)
 2:     $t \leftarrow t_s$                                          ▷ $t_s$: initial temperature
 3:     $M \leftarrow M_0$
 4:     $curDistance \leftarrow$ SCORE($H$, $M$)
 5:     while $t_e \leq t$ do                                    ▷ $t_e$: finished temperature
 6:         while $k \leq k_{max}$ do                            ▷ $k_{max}$: iterative count
 7:             $v_1, v_2 \leftarrow$ SELECT($M$)
 8:             $M_T \leftarrow$ SWAP($v_1, v_2, M$)                 ▷ $M_T$ is a trial set
 9:             $trialDistance \leftarrow$ SCORE($H$, $M_T$)
10:             $\Delta S \leftarrow trialDistance - curDistance$
11:             if $\Delta S > 0$ then
12:                 $curDistance \leftarrow trialDistance$
13:                 $M \leftarrow M_T$                                ▷ Update $M$
14:             else
15:                 $trialProb \leftarrow RANDOM(0, 1)$
16:                 if $trialProb < e^{-\frac{\Delta S}{t}}$ then
17:                     $curDistance \leftarrow trialDistance$
18:                     $M \leftarrow M_T$
19:                 end if
20:             end if
21:             $k \leftarrow k + 1$
22:         end while
23:         $t \leftarrow \alpha t$                                   ▷ $0 < \alpha < 1$
24:     end while
25:     return $M, curDistance$
26: end procedure
```



ID	4	1	2	3	0
4	0	0	0	1	1
1	0	0	1	0	1
2	0	1	0	0	0
3	1	0	1	0	0
0	1	1	0	0	0

Figure 3.3: Illustration of matrix M when node ID 0 and 4 are swapped.

A BCG target graph has an adjacency matrix M_0 of elements m_{uv} defined by:

$$m_{uv} = \begin{cases} 1 & \text{if there is a BCG edge between nodes } u, v \in V \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The SELECT function chooses two node IDs uniformly at random. The SWAP function interchanges rows of selected nodes, and then interchange columns in M as follows:

$$\forall k \in [0, N), \begin{cases} m_{ik}, m_{jk} & = & m_{jk}, m_{ik} \\ m_{ki}, m_{kj} & = & m_{kj}, m_{ki} \end{cases} \tag{3.5}$$

where $m_{\alpha\beta}$ is a matrix element of M, i and j are selected nodes, k is the row or column index, and N is the number of nodes.

Figure 3.3 shows an example of M for the graph in Figure 3.1(b) when node ID 0 and 4 are exchanged with the SWAP function. The first node of ID 0 was connected to the second node of ID 1 and the fifth node of ID 4 before exchange. After the exchange, the first node having node ID 4 is connected to the fourth node of ID 3 and the fifth node of ID 0. Therefore, each sensor node is assigned a different node ID while the target graph topology is preserved.

The SCORE function calculates the sum of weights as followings:

$$SCORE(H, M) = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} h_{ij} * m_{ij}/2. \tag{3.6}$$

Equation (3.6) is the sum of communication distances once the edges that do not follow the BCG connection rules are removed.

Case of infinite radio range

The simulated annealing based node ID assignment for completing connections is almost identical to the simulated annealing node ID assignment for reducing the average communication distance. The difference is with the SCORE function and the update condition. Line 11 of Algorithm 1 is changed from $\Delta S > 0$ to $\Delta S < 0$ because the SCORE function calculates the sum of disconnected edges in a target graph as follows:

$$r_{ij} = \begin{cases} 1 & \text{if } h_{ij} - m_{ij} = -1 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$SCORE(H, M) = \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} r_{ij}/2. \tag{3.8}$$

In this thesis, for the SA assignment, we set t_e = 30, t_s = 0.1, $k_{max} = N$, and $\alpha = 0.98$. SA parameter values were selected heuristically based on simulation results.

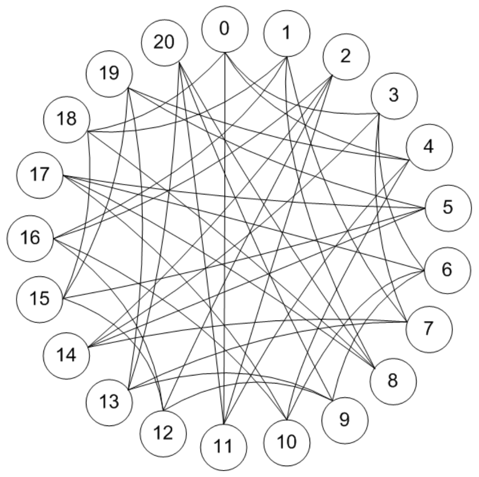
3.4 Chordal Ring Based Node ID Assignment

3.4.1 CR representation and node ID conversion between GCR and CR

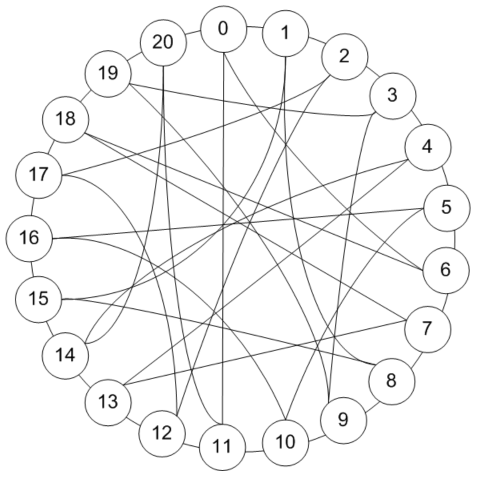
Recall that nodes of ID n in a chordal ring (CR) have neighbors ID $n + 1$ mod N and $n - 1$ mod N. In other words, all nodes on the circumference of the ring are connected to form a Hamiltonian cycle. In the CR representation of BCG, the transform element and class representing elements are function of BCG parameters. Let $\mathbb{T}$ be a transform element in CR and $\eth$ be the set of class representing elements in CR. Then, any vertex $v \in V$ is represented by $\mathbb{T}$ and $\eth_{i'}$ in $\eth$ as follows:

$$v = \mathbb{T}^{j'} * \eth_{i'}. \tag{3.9}$$

Since there is no systematic representation for CR and no conversion method between GCR ID and CR ID, we propose (a) a CR representation in the integer



(a) Generalized chordal ring.



(b) Chordal ring.

Figure 3.4: Examples of generalized chordal ring and chordal ring.

domain and (b) a conversion method between GCR ID and CR ID. The number of classes, q, can be different in the CR and GCR domains [56]. Therefore, we use q_g for the number of classes in the GCR domain and q_c in the CR domain. The node ID representation in the CR domain ($ID_c(v)$) is given as follows:

$$ID_c(v) = q_c * j' + i', \tag{3.10}$$

where q_c is k or p, and $ID_c(v)$ is the node ID of v in the CR domain. Figure 3.4(b) shows the CR representation of a Borel Cayley graph in the integer domain. The graphs shown in Figure 3.4 represent the same BCG represented in both the CR and GCR domains.

By combining Eq. (2.5) and Eq. (3.9), the conversion formulation from CR ID to GCR ID is

$$\begin{aligned} ID_c(v) &= q_c * j' + i', \\ v &= \mathbb{T}^{j'} * \eth_{i'} = \begin{pmatrix} a^i & j \\ 0 & 1 \end{pmatrix}, \\ ID_g(v) &= q_g * j + i, \end{aligned} \tag{3.11}$$

where $ID_g(v)$ is the node ID representation in the GCR domain.

Similarly, the conversion formulation from GCR ID to CR ID is

$$\begin{aligned} ID_g(v) &= q_g * j + i, \\ v &= T^j * a_i = \begin{pmatrix} a^i & j \\ 0 & 1 \end{pmatrix} = \mathbb{T}^{j'} * \eth_{i'}, \\ ID_c(v) &= q_c * j' + i'. \end{aligned} \tag{3.12}$$

Based on the set of class representing elements $\eth$ given in [56], integers i' and j' can be calculated by Algorithm 2. Note that Algorithm 2 does not have any infinite loop since the matrix v is $\mathbb{T}^{j'}$ multiplied by $\eth_{i'}$.

Algorithm 2 Calculating j' and i'

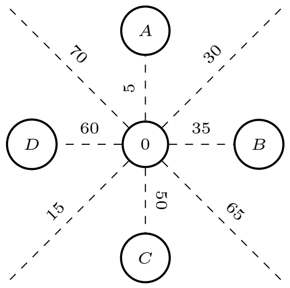
```
Require: $\mathbb{T}$ is nonsingular.
 1: procedure CALCULATING $j'$ AND $i'$($\mathbb{T}$, $v$)
 2:     $j', i' \leftarrow 0$
 3:     while Not $v \in \eth$ do                    ▷ Find $j'$ of $\mathbb{T}$ using Eq. (3.9)
 4:         $v \leftarrow \mathbb{T}^{-1}v$
 5:         $j' \leftarrow j' + 1$
 6:     end while
 7:     while $v \neq \eth_{i'}$ do                  ▷ Find $i'$ of $\eth_{i'}$ using Eq. (3.9)
 8:         $i' \leftarrow i' + 1$
 9:     end while
10: return $j', i'$
11: end procedure
```

We already discussed the relationship between GCR ID and the CR ID. The GCR and CR representations of BCG have different advantages. The GCR representation of BCG has an optimal routing algorithm to identify the shortest paths between any sources and destination pairs [59]. On the other hand, the CR representation supports a Hamiltonian cycle and a simple sub-optimal routing algorithm. Therefore, we assign IDs in the CR domain to minimize the communication distance between neighboring nodes and then map them to the GCR domain for optimal routing performance.

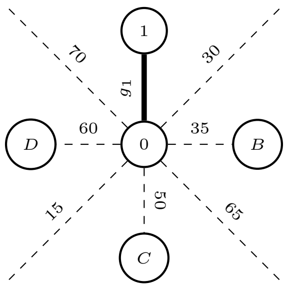
3.4.2 Algorithm

The chordal ring based method (CR assignment) consists of three main steps:

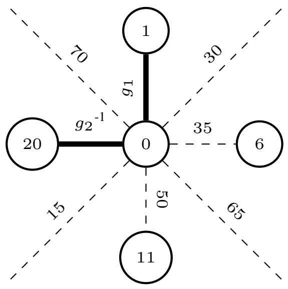
1. Making a Hamiltonian cycle in the CR domain,



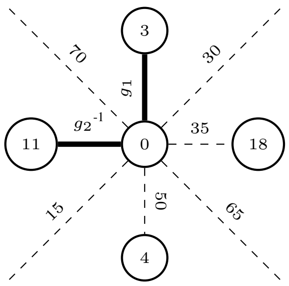
(a) Select the closest edge.



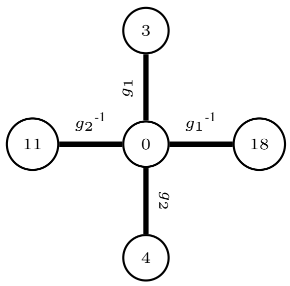
(b) Assign the next node ID to the selected node.



(c) All sensors are assigned node IDs using a Hamiltonian cycle.



(d) Convert to GCR



(e) Establish connections.

Figure 3.5: Illustration of CR assignment. Note that each alphabet represents a physical sensor node and a dashed line represents distance.

2. Converting node IDs from the CR domain to the GCR domain, and
3. Establishing edges.

Figure 3.5 illustrates the CR assignment algorithm. There, the algorithm starts by choosing the lowest weighted edge of sensor node ID 0 and then assigns 1 plus node ID 0 to the corresponding node (Figure 3.5(b)). Selection of the closest sensor can be implemented at a reasonable computational cost in today's sensor technology. Existing localization methods include global positioning system (GSP), beacon nodes, and proximity-based localization [77–81]. The next node selects the edge with the lowest weight among edges that are not already connected to assigned nodes. The algorithm repeats until the last node ID is assigned

(Figure 3.5(c)). Then, the node IDs are mapped from the CR domain to the GCR domain (Figure 3.5(d)). Finally, the actual connections are established using GCR connection constants following the BCG connection rule (Figure 3.5(e)). Algorithm 3 summarizes the CR assignment.

Algorithm 3 CR assignment.

```
 1: procedure CR ASSIGNMENT($\mathbb{T}$, $\eth$)
 2:     $v \leftarrow v_0$                                    ▷ $v_0$: starting node
 3:     $ID_c(v) \leftarrow 0$
 4:     while $ID_c(v) \neq N$ do                   ▷ Until all nodes are assigned
 5:         $v_n \leftarrow$ select unassigned node closest to $v$
 6:         $ID_c(v_n) \leftarrow ID_c(v) + 1$
 7:         $v \leftarrow v_n$
 8:     end while
 9:     Convert all IDs from CR to GCR                        ▷ Using Eq. (3.11)
10:     Establish the remained edges using GCR constants
11: end procedure
```

The CR assignment guarantees that at least the lowest weighted edges of consecutive node IDs are selected as communication links, except for the edge between the starting node and the last assigned node. As a result, we optimize two out of four edges per node excluding first and last ones. We expect the total weights of edges from the CR assignment to be smaller than those of a random node ID assignment. Moreover, the CR assignment is a fully distributed algorithm and does not require pre-assigned node IDs.

3.4.3 Average communication distance

We define the average communication distance as the average weights of the target graph communication distance between pairs of nodes. Our goal is to minimize the average communication distance. To evaluate our assignment algorithm, we calculated the expected communication distance analytically. From Eq. (3.1), the expected communication distance between randomly chosen positions is as follows:

$$E(D) = \frac{1}{a^2b^2} \int_0^b \int_0^b \int_0^a \int_0^a w_{uv} \, du_x dv_x du_y dv_y \ , \qquad (3.13)$$

where a and b represent the horizontal and vertical dimensions (m), respectively.

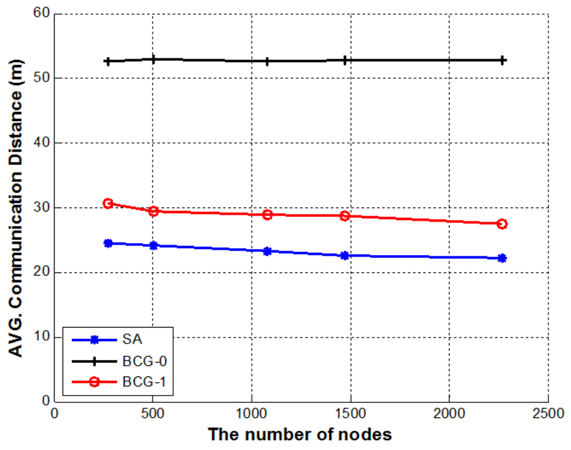
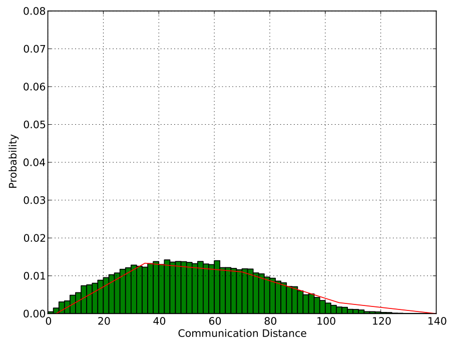


Figure 3.6: Average communication distance of resultant graphs by node ID assignments in the case of sufficient radio range to cover whole sensor deployment area.

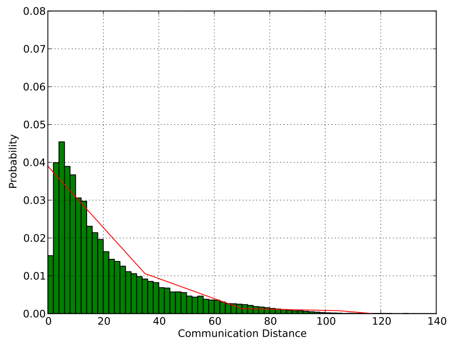
Figure 3.6 shows the average communication distance for each node ID assignment algorithm where $a, b = 100$m. The Random assignment algorithm uniformly and randomly assigns a unique BCG node ID to all the sensors. We call the resultant network topology with Random assignment BCG-0 and the resultant network topology with CR assignment BCG-1. The average weight of the Random assignment and the expected distance are both approximately equal to 52.

We also calculated the standard deviation for the average communication distance from 100 CR assignment samples. The results show that selection of the initial node does not affect the average communication distance. For graphs of sizes $N = 272$, 506, 1081, 1474, and 2265, the standard deviations for the CR assignment were 0.58, 0.53, 0.62, 0.72, and 0.60, respectively.

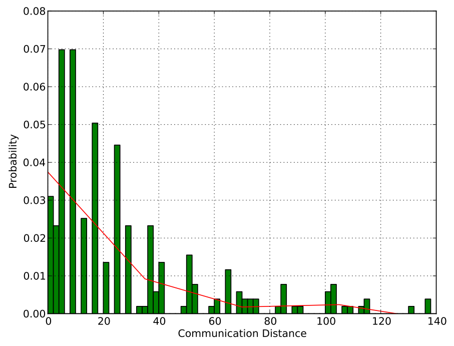
Figure 3.7 shows the histograms of the communication distance for $N = 272$ generated by the proposed node ID assignment algorithms. The histogram of the SA assignment exhibits a right skewed distribution with a high frequency of short edges. The CR assignment also show a right skewed distribution although it is less regular. We also obtained similar histograms for other network sizes listed in Table 2.1.



(a) Random assignment.



(b) SA assignment.



(c) CR assignment.

Figure 3.7: Histograms of communication distance by the proposed algorithms with $N = 272$ and a solid line shows polynomial fit line.

3.5 Distributed Node ID swapping Assignment

In the previous section, we showed the CR assignment algorithm for reducing average communication distance with infinite radio range. In this section, we propose the distributed node ID swapping assignment for completing communication edges when radio range is finite. The distributed node ID swapping assignment (Dist-swap assignment) consists of four main steps:

1. Broadcast its node ID to its physical neighbors,
2. Collect IDs of physical neighbors that can be swapped,
3. Select the best-fit node ID to be swapped with, and
4. Swap node IDs and update each logical neighbor table of its physical neighbors.

3.5.1 Terminologies

We denote the logical node ID of node x by x_{id} and define the following:

- $D(x_{\text{id}})$: Set of logical neighbors of node ID x_{id} in the target graph,
- $T(x)$: Set of logical IDs of node x's physical neighbors, and
- $L(x_{\text{id}}, y) = |D(x_{\text{id}}) \cap T(y)|$: The number of logical neighbors of node ID x_{id} that are also physical neighbors of node y.

We define four different packet types:

1. Token packet: Forwarded by nodes and used to initiate the node swapping algorithm by the node recipient. Also contains counter $reptCnt$.
2. Info packet: Used by the current node holding the token to broadcast x_{id} to its physical neighbors so as to identify candidate nodes to be swapped.
3. Request packet: Supports collection of candidate nodes. If a physical neighbor node y receiving an Info packet from node x determines it can be swapped, it sends a Request packet back to node x. The Request packet contains y_{id}, $L(y_{\text{id}}, y)$, and $L(x_{\text{id}}, y)$.
4. Swap packet: Used by the current node x to announce to its physical neighbors swapping with node y. Contains the pair $(x_{\text{id}}, y_{\text{id}})$. Nodes having those node IDs in $T()$ need to update the swapped IDs.

3.5.2 Assumption

We assume that (a) a host graph is a connected graph (There is a path between all node pairs), (b) nodes have pre-assigned unique IDs ranging from 0 to $N-1$, (c) $T(x)$ (a set of logical IDs of its physical neighbors) at each node $x \in V$ is obtained before the swapping process, and (d) the order of nodes to perform operation is based on token method.

3.5.3 Algorithm

The Dist-swap assignment is performed by one node at a time using a token procedure (i.e., there is one token in the network, and only the node holding the token executes the swapping algorithm). To prevent an infinite loop, we rely on a token counter ($repCnt$) to track the number of times of the token being passed around. At the beginning of the Dist-swap assignment, the node that starts the operation sets $repCnt$ to $repTotal$ that was heuristically selected based on previous simulation studies. As the token travels around the network, each node decreases $repCnt$ by one before forwarding the token to the next node. The Dist-swap assignment ends once $repCnt$ reaches zero.

We describe in details the Dist-swap assignment as follows:

- Table 3.1 describes the Dist-swap assignment operation after node x receives a Token packet. First, node x determines whether or not it needs to execute the swapping process based on its current number of logical neighbor IDs among its physical neighbors and based on its target number of logical neighbors within the target graph. If those are not the same, node x sends an Info packet to its physical neighbors. After receiving Request packets from its physical neighbors, node x determines which node ID swapping is most beneficial. Because the Request packet from node y contains y_{id}, $L(y_{\text{id}}, y)$, and $L(x_{\text{id}}, y)$, node x can calculate the change in the number of logical neighbors from the physical neighbors when node IDs x_{id} and y_{id} are swapped. If a candidate node for swapping exists, node x sends a Swap packet with its own node ID and the selected node ID to its neighbors.

- Table 3.2 illustrates node y operation after receiving an Info packet. To determine whether or not to reply with a Request packet, node y checks the number of its logical neighbors that would be assigned to its physical neighbors if it were assigned node ID x_{id}. If it is larger than the number

Table 3.1: Dist-swap assignment operation after a node receives the Token packet.

1. A node x receives a Token packet.
   (a) If $|D(x_{\text{id}})| = L(x_{\text{id}}, x)$, forward the Token packet randomly to one of its physical neighbors.
   (b) Otherwise, sends an Info packet to all its physical neighbors.
2. Node x collects the Request packets.
   (a) If there is no Request packet received, forward the Token packet randomly to one of its physical neighbors.
3. Node x creates a list of candidate nodes. A node y is a candidate node if $L(y_{\text{id}}, x) + L(x_{\text{id}}, y) - L(x_{\text{id}}, x) - L(y_{\text{id}}, y) > 0$.
   (a) If there is no candidate node, forward the Token packet randomly to one of its physical neighbors.
   (b) Otherwise, select the node y maximizing the number of logical neighbors when two node IDs are swapped.
4. Node x sends Swap packet with $(x_{\text{id}}, y_{\text{id}})$.
5. Node x sends Token packet to node y.

of its logical neighbors with the current node ID, node y sends a Request packet.

- Table 3.3 shows the operation when node z receives a Swap packet. Swap packet contains node IDs $(x_{\text{id}}, y_{\text{id}})$. If y_{id} is the same as z_{id}, then the Request packet sent by z is accepted. Next, node z changes its node ID to x_{id} and sends a Swap packet with $(z_{\text{id}}, x_{\text{id}})$ to its physical neighbors so that they update their $T()$s. If x_{id} is the same as z_{id}, node z ignores that packet. Otherwise, node z updates $T(z)$ to reflect that node ID x_{id} is changed to y_{id}.

3.5.4 Example

Next, we use an example to illustrate the Dist-swap assignment operation. The physical nodes of the host graph are denoted by nodes x, y, and z, and the logical

Table 3.2: Dist-swap assignment operation after a node receives the Info packet.

1. A node y receives an Info packet.
2. Node y calculates the number of its logical neighbors among its physical neighbors if it were assigned node ID x_{id}.
   (a) If that number is larger than or equal to the number of logical neighbors of the current ID, send the Request packet.
   (b) Otherwise, ignore the Info packet.

Table 3.3: Dist-swap assignment operation after a node receives the Swap packet.

1. A node z receives a Swap packet that contains node IDs $(x_{\text{id}}, y_{\text{id}})$.
   (a) If z_{id} is equal to y_{id}, change z_{id} to x_{id} and send a Swap packet with node IDs $(z_{\text{id}}, x_{\text{id}})$ to its physical neighbors.
   (b) If z_{id} is equal to x_{id}, ignore the Swap packet .
   (c) Otherwise, change x_{id} to y_{id} in $T(z)$.

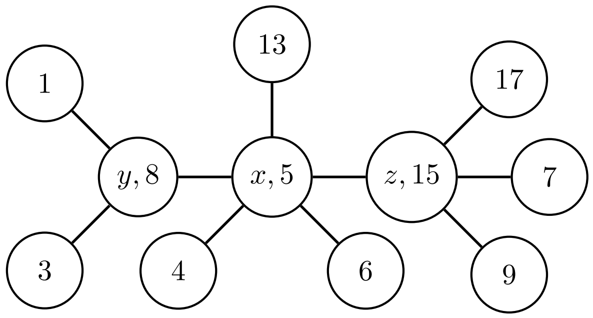


Figure 3.8: Host graph used as example of Dist-swap assignment.

node IDs of the target graph are denoted by 1, 2,...,etc. Figure 3.8 shows that logical node IDs 5, 8, and 15 have been preassigned to nodes x, y, and z, respectively. We assume the target graph has a simple connection rule, namely: node ID i is connected to $i-4$, $i-2$, $i+2$, and $i+4$. Given the connection rule in the target graph, the neighbors of node ID 5 in the target graph are: $D(5) = \{1, 3, 7, 9\}$. Similarly, $D(8) = \{4, 6, 10, 12\}$, and $D(15) = \{11, 13, 17, 19\}$.

Let node x receive a Token packet first. Then node x decides whether or not

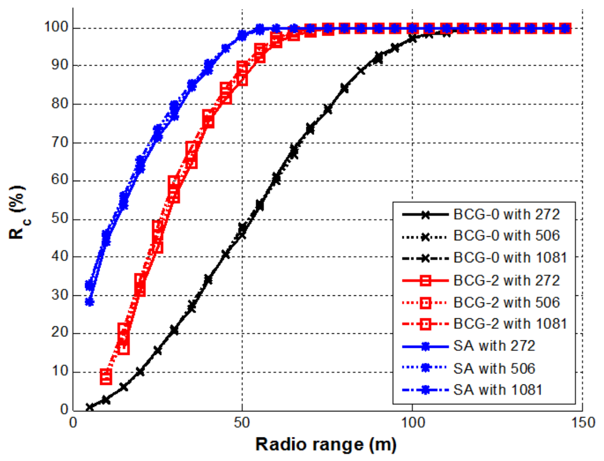


Figure 3.9: Edge construction percentage with distributed node ID assignment.

to perform the swapping process based on the current number of logical neighbors among its physical neighbors ($L(5, x)$). Since $T(x) = \{4, 6, 8, 13, 15\}$, $D(5) = \{1, 3, 7, 9\}$, and $L(5, x) = 0$, the number of node x logical neighbors among its physical neighbors is not the same as the maximum number of logical neighbors 4. So, node x sends an Info packet to its physical neighbor nodes. Swapping node IDs is beneficial to node y because $L(8, y)$ is zero and $L(5, y)$ is two, which means node y gains two more logical neighbors after swapping IDs with node ID 5. Thus, node y sends the Request packet to node x. Node z also sends a Request packet to node x through the same process. Finally, node x determines to swap IDs with node y because $L(8, x) + L(5, y) = 4$ is greater than $L(15, x) + L(5, z) = 3$.

3.5.5 Completeness of resultant graph

Recall that an efficient ID assignment method should incorporate as many as possible target graph edges in the resultant graph from a given host graph where sensor nodes are limited to a specific radio range. Therefore, to measure the performance of our assignment methods, we define R_c, the ratio of the number of edges of the resultant graph over that of the target graph. That is,

$$R_c = \frac{\text{Number of edges in resultant graph}}{\text{Number of edges in target graph}}. \tag{3.14}$$

Table 3.4: Summary of our proposed algorithms.

Resultant graphs	Algorithms
BCG-0	Random assignment
BCG-1	Chordal ring based node ID assignment
BCG-2	Distributed node ID swapping assignment
BCG-3	Distributed node ID swapping assignment & Random node selection

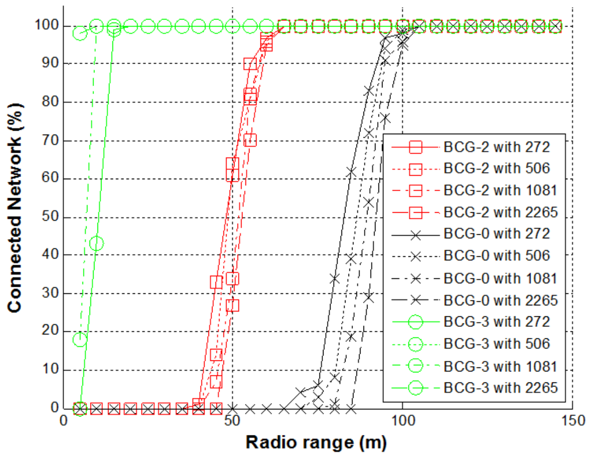


Figure 3.10: Connected network percentage with node ID assignments.

For a given radio transmission range, the assignment method with the largest R_c indicates that the most edges of the target graphs are incorporated in the resultant graph. We define the resultant network topology with Dist-swap assignment BCG-2. Figure 3.9 shows R_c for BCGs target graphs with $N = 1081$. The Dist-swap assignment (BCG-2) produces an R_c larger than that of the Random assignment (BCG-0) and less than that of the SA assignment. Especially, with 50m radio range in 100m by 100m area, it showed 43% more edges than the Random assignment.

When applying BCGs to wireless sensor networks, we also need to consider network connectivity. Because the Dist-swap assignment maximizes R_c without considering network connectivity, we apply a random node selection algorithm on processed nodes that do not have the right number of logical neighbors. We define the resultant graph from this method BCG-3. Table 3.4 summarizes resultant

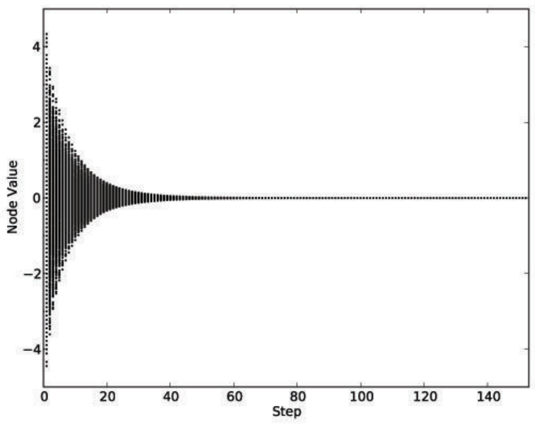


Figure 3.11: Example of status agreement with consensus protocol on BCG.

graphs from our proposed algorithms.

Figure 3.10 shows the percentage of connected network function of assignments for 100 network samples. BCG-0 with a 105m radio range produces a 100% connected network while BCG-2 and BCG-3 requires 65m and 10m radio range, respectively to produce a connected network. All connections of BCG-2 are satisfied by the connection rules of BCGs. However, BCG-3 has non-BCG connections since some are logical neighbors randomly selected for improved connectivity.

3.6 Data Dissemination

3.6.1 Consensus protocol

We use the consensus protocol as a benchmark to evaluate our topology control algorithms. Consensus protocol is a distributed node-to-node message exchange rule to achieve a network-wide agreement over a quantity of interest (e.g., average of sensory data). Consensus protocol has a long history in distributed computing and has been used in a variety of applications. Moreover, it has been widely accepted as a reliable measure of data fusion performance of network topologies [82, 83]. In WSN, consensus protocol research has been focusing on time synchronization and gossip algorithms [84–86]. Readers further interested with consensus protocol and its application are referred to [60].

The consensus agreement value can be an average, a maximum, a minimum, or any function. Consensus protocol only depends on the initial states of the nodes in the network. Furthermore, the speed of consensus is a good measure of the efficiency of a network topology to distribute information. In this study, we consider the distributed average consensus protocol proposed in [60] which we summarize next.

Let us consider a network system of which the underlying logical network topology is represented by an unweighted, undirected graph. Each node $v \in V$ in the system communicates its state value x_v to its immediate neighbors $N(v) := u$ where $e_{v,u} \in E(V)$ and E is an edge set. At each iteration τ, nodes exchange their current state value $x_v(\tau)$ with their immediate neighbors. Given the state values $x_u(\tau)$ received from their neighbors $u \in N(v)$, each node u updates its state according to

$$x_u(t+1) = x_u(t) + \frac{1}{\omega} \sum_{v \in N(u)} (x_v(t) - x_u(t)), \tag{3.15}$$

where ω is typically $0 < \frac{1}{\omega} < \frac{1}{2d_{max}(G)}$ and $d_{max}(G)$ is the largest nodal degree. We set ω to $2 \times d_{max}(G) + 1$ in this thesis. Consensus convergence can be controlled by ω. However, finding an optimal value for ω is out of scope in this thesis.

Following the method in [87], we use the average consensus protocol (Eq. 3.15) to measure the information fusion performance of the network generated by the proposed topology control protocol. Figure 3.11 shows an example of agreement steps by a consensus protocol. BCGs have already been shown to exhibit better consensus protocol performance than mesh, torus, and small world networks [68]. However, that research compared only topology level without considering nodes's physical geographical information and radio range. In this thesis, we compare a BCG with other network topologies when applied to a wireless sensor network.

3.6.2 Simulation setup

We evaluated our proposed topology control algorithms in terms of consensus speed and power consumption. The simulations were executed on 100 host graphs; each with sensors uniformly and randomly distributed over a 100m $\times$ 100m area. Consensus protocol was initialized with nodes state value set to integers randomly chosen between -5 and 5 inclusive. We declared a network topology to have reached an agreement once all node values equal the average of all initial state

Table 3.5: Radio model parameters.

$Parameters$	$Values$
n	2
β_1	$45nJ/bit$
β_2	$10pJ/bit$
γ	$135nJ/bit$
B	$320bits(=40bytes)$

values within a precision of 0.001. The performance of the consensus protocol was measured as the number of steps needed to reach a network-wide consensus.

We also computed the energy needed for all the nodes in the network to reach a consensus using the average consensus protocol. To do this, we utilized the radio model described in [88]. For the sake of simplicity, we only considered the energy consumptions from data transmission (E_{tx}) and reception (E_{rx}) defined as follows:

$$E_{tx} = B(\beta_1 + \beta_2 \omega(v,u)^n), \tag{3.16}$$

$$E_{rx} = B\gamma, \tag{3.17}$$

where n is the path loss exponent which ranges from 2 to 6. The constants β_1, β_2, and γ correspond to the energy dissipated by the transmitter module, transmit amplifier, and receiver module, respectively. We denote the estimated distance between nodes v and u by $\omega(v,u)$ and the length of the transmitted messages by B. We set $n = 2$, equivalent to the free-space pass loss model and assume that a sensor node v, regardless of the topology control protocols considered, can adjust its transmission power to reach its neighbors w. The parameters used in the simulation are summarized in Table 3.5.

Using the average consensus protocol, a node transmits and receives to and from each of its neighbors at every iteration. Thus the node's energy consumed by the network at the τ^{th} iteration is given by

$$P_0(v) = \tau B\Big(\sum_{\forall(v,u)\in E v\neq u} \big(\beta_1 + \beta_2\omega(v,u)^n\big) + \gamma deg(v)\Big), \tag{3.18}$$

where $deg(v)$ is the number of logical neighbors, (v,u) is a logical connection and E denotes the edge set. The average nodal energy consumption is given by $\overline{P_0} = \sum_{v\in V} P_0(v)/|V|$. We call this power consumption model *Power model* 0.

In wireless sensor networks, a network can support multi-casting routing to transmit at once to multiple nodes inside the radio range. In this case, power consumption is calculated not with each communication edge but with the maximum distance communication edge as follows.

$$P_1(v) = \tau B\big(\beta_1 + \beta_2 \mathrm{Max}\{r(v,u), u \in adj(v)\}^n + \gamma deg(v)\big). \tag{3.19}$$

We call this power consumption model *Power model* 1.

3.6.3 Comparison between BCG-0 and BCG-1

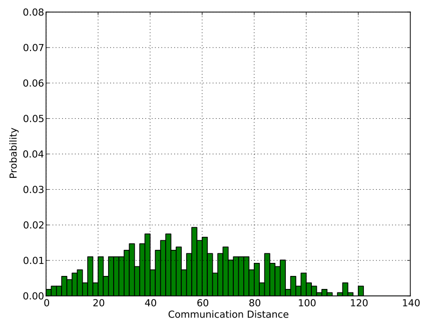
Figure 3.12 shows the histograms of the communication distance for $N = 1081$ generated by the proposed node ID assignment algorithms. The histogram of the CR assignment (BCG-1) exhibits a right skewed distribution with a high frequency of short edges. The Dist-swap assignment (BCG-2) shows all connection distance are under the maximum radio range 80m. We obtained similar histograms for the network sizes listed in Table 2.1. We compared the average power consumption between the Random assignment and the CR assignment because they require a radio range that covers the whole node deployment area. Figure 3.13 summarizes the resulting consensus protocol energy consumption. We found BCG-1 consumed 8% less energy than BCG-0 with power model 0 and 2% less with power model 1.

3.6.4 Comparison between our proposed topology control and existing topology controls

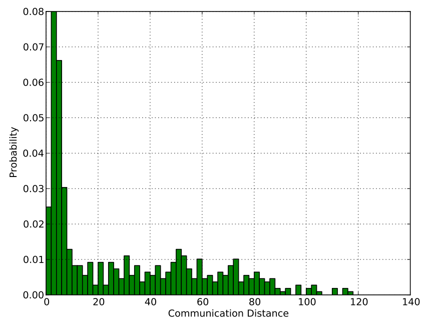
We compared our topology control algorithms with existing topology controls. The evaluation was done in terms of diameter, average path length, consensus steps, and nodal energy consumption.

We used the following topologies for comparison against our approach:

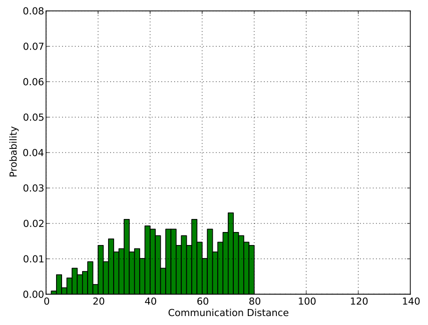
- Max: Network where all nodes within the maximum radio range are logical neighbors. Without any topology control, a network is called Max topology.
- E-MST [89]: The Euclidean minimum spanning tree is a minimum spanning tree whose edge weight is the distance between nodes. This algorithm minimizes the summed weight of edges and generates a connected network.



(a) Random assignment(BCG-0).



(b) CR assignment (BCG-1).



(c) Dist-swap assignment (BCG-2).

Figure 3.12: Histograms of communication distance by the proposed algorithms with $N = 1081$.

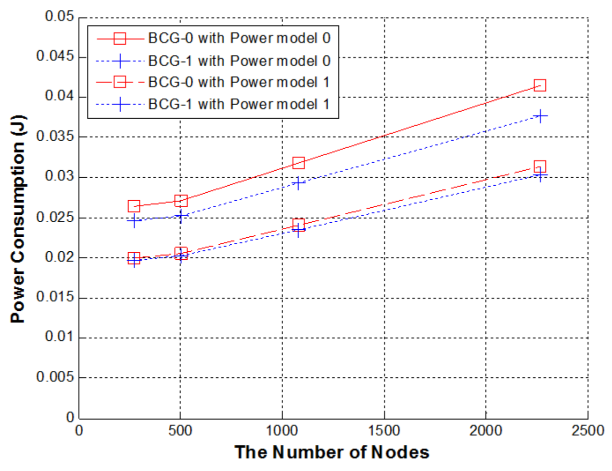


Figure 3.13: Power consumption comparison between BCG-0 and BCG-1.

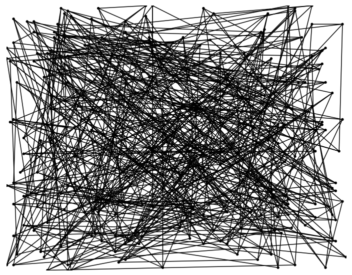
- K-Neigh [34]: Network where the number of neighbors of a node equal to, or slightly below a given value *k*.

- Gabriel Graph [90]: A planar graph (no edges cross one another) supports geographical routings and is defined such that an edge (v, u) exists if no other node is inside the circle with the diameter uv.

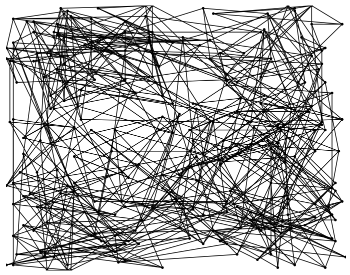
Figure 3.14 provides illustrations of the topologies constructed with (a) the Random assignment (BCG-0), (b) the Dist-swapping assignment (BCG-2), (c) Max, (d) E-MST, (e) K-Neigh, and (f) Gabriel graph. (For ease of description, we show sample graphs with $N = 272$. However, we simulated networks of size $N = 1081$ in this section.)

In Figure 3.15 and Figure 3.16, we show the diameter and average path length function of the transmission range, respectively. We only compared the cases where the resultant graph is a connected graph. From these figures, we confirm again that BCGs have the smallest diameter and the shortest average path length for a given radio range except Max topology. This is because as a radio range increases, BCGs have a constant number of logical neighbors while Max topology has much more logical neighbors than others.

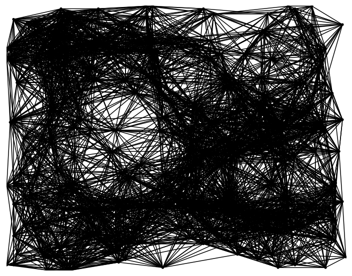
From Figure 3.17, at radio range 30m, consensus protocol convergence for BCG-3 was 35.5, 2932, and 15.9 times faster than that of Gabriel, E-MST, and K-Neigh topologies. At radio range 50m, convergence of BCG-3 was 81.2, 6756.9, and 39.9 times faster than that of Gabriel, E-MST, and K-Neigh topologies. Even



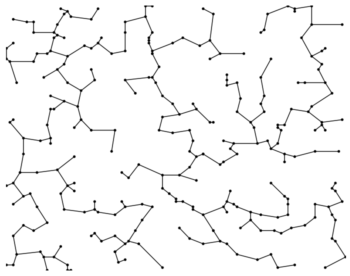
(a) BCG-0.



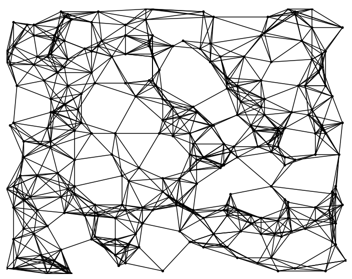
(b) BCG-3.



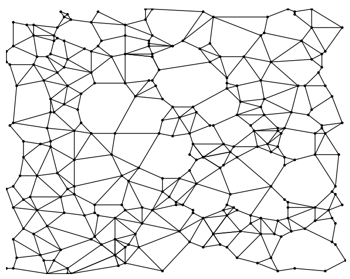
(c) Max.



(d) E-MST.



(e) K-Neigh.



(f) Gabriel Graph.

Figure 3.14: BCG-0, BCG-3 with radio range 40m, Max with 25m, E-MST, K-Neigh with $k = 10$ and 25m, Gabriel Graph. $N = 272$

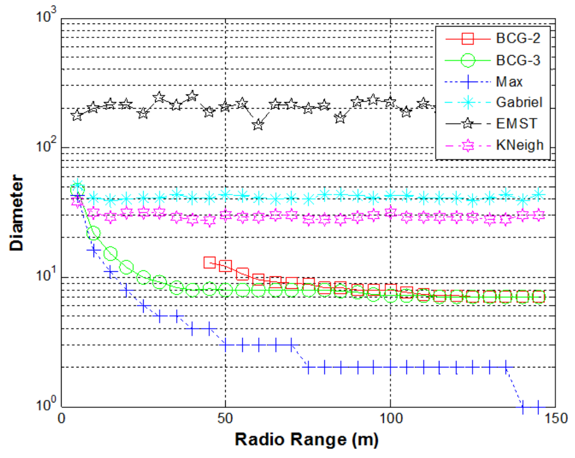


Figure 3.15: Comparison of diameters.

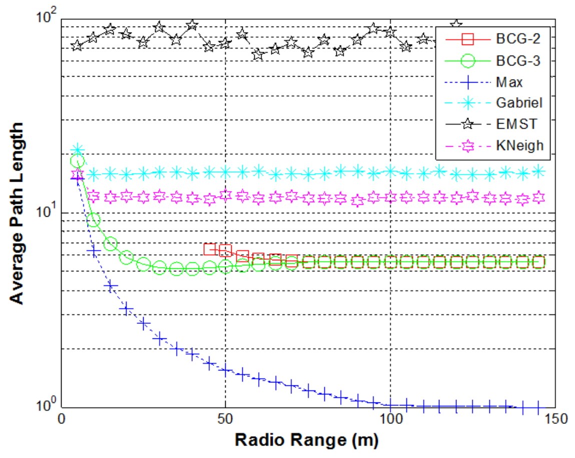


Figure 3.16: Comparison of average path lengths.

though Max topology has the best consensus steps, due to its large number of logical neighbors, it consumed 23 times more nodal power than BCG in Figure 3.18. When considering multi-cast routing (Power model 1), the Max topology has an obvious advantage as only the maximum radio range is used in the power consumption computation. However, even with this advantage, our simulations showed that BCG continues to outperform the Max topology in Figure 3.19. This better performance is explained by the fact that the Max topology have a larger

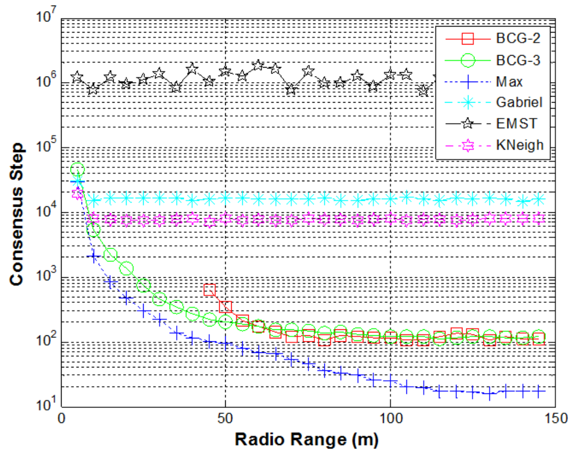


Figure 3.17: Comparison of consensus steps.

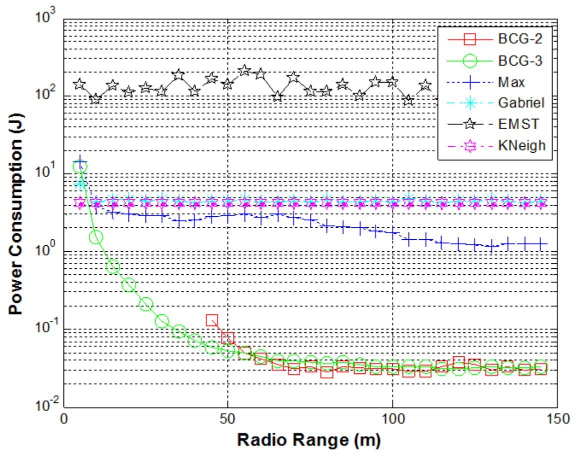


Figure 3.18: Comparison of power model 0 for consensus protocol.

number of logical neighbors and hence larger power is consumed at the receiver modules.

3.6.5 Consensus protocol analysis with considering interferences

In wireless sensor networks, signals travel from a sender to a receiver through radio channels. These signals sent at a particular transmit power by the transmitter

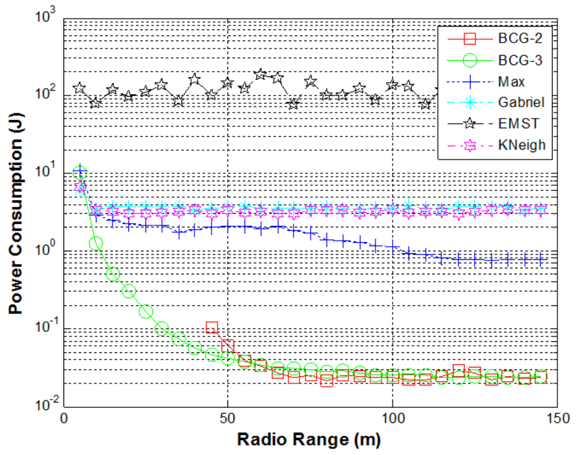


Figure 3.19: Comparison of power model 1 for consensus protocol.

suffer attenuation through the radio channel. The receiver on the other end is only able to receive signals with power levels greater than the sensitivity of its transceiver. The attenuation, commonly known as the path loss of the channel, directly depends on the distance between sender and receiver, the frequency of operation and the radio channel interference. Those factors are considered for design of efficient topology controls.

We simulated consensus protocol on a 1081-nodes BCG-3 (Distributed node ID swapping assignment & Random node selection) and on other topology control algorithms with a simple interference model. Sensors uniformly and randomly distributed over a 100m $\times$ 100m area. Consensus protocol was initialized with nodes state value set to integers randomly chosen between -5 and 5 inclusive. Our interference model is based on the signal-to-interference-ratio (SIR) and capture threshold model presented in [91].

When the following two conditions are satisfied, we assume that a packet is delivered successfully:

- Capture threshold model: A received signal power is larger than some threshold (set to -96dBm by default).

- SIR condition: A received signal power is larger than the sum of interference noise power from other nodes. (The typical minimal SIR for correct signal decoding is 10dB.)

We use the log-distance path loss model as the radio propagation model to

Table 3.6: Consensus protocol performance after 10000 ticks with 30m initial radio range and a packet generation rate of 5×10^{-3}.

	Mean	STD	AVG. Power (J)	$\frac{CollidedPkts}{GeneratedPkts}$
BCG-3	-0.063	0.102	0.013	4.71%
K-Neigh	0.044	0.292	0.027	0.85%
MST	0.069	0.948	0.006	0.52%
Gabriel	-0.070	0.514	0.014	0.78%

predict the signal power loss [92]. The signal energy loss is calculated as follows:

$$P_l(d) = P_l(d_0) + n10\log_{10}(\frac{d}{d_0}), \tag{3.20}$$

where the path loss exponent n is set to 2, the reference distance d_0 is set to 1.0m, and the reference energy $P_l(d_0)$ is based on a Friis propagation model:

$$P_l(d_0) = \frac{P_t G_t G_r \lambda^2}{16\pi^2 d_0^2 L}, \tag{3.21}$$

where P_t represents the transmission power, G_t the transmission gain (set to 1 dbym by default), G_r the reception gain (set to 1 dbym by default), λ the carrier wavelength, and L is the system loss (set to 1 by default).

An agreement of consensus protocol used in 3.6.2 is difficult to achieve from packet collision by radio interference. We tried different simulation termination conditions. One was to measure performance after running for a fixed simulated time. The other was to record the number of consensus steps when the standard deviation (STD) of the current state of all nodes is less than a certain value.

When packets are sent at every simulation ticks, a consensus protocol cannot be achieved due to a lot of packet collisions. So we send packets with a packet generation rate that nodes generate packets at a constant average rate of R packets per time slot. First, we used a packet generation rate of 5×10^{-3} and then used various packet generation rates.

In Table 3.6, BCG-3 produced larger packet collision percentage because it sends packets with higher transmission power to support a longer communication radio than the other networks. Even though it has more collision packets, BCG-3 showed the smallest STD calculated from status of all nodes. It means that nodes in a network have more similar values than the other topologies. Figure 3.20

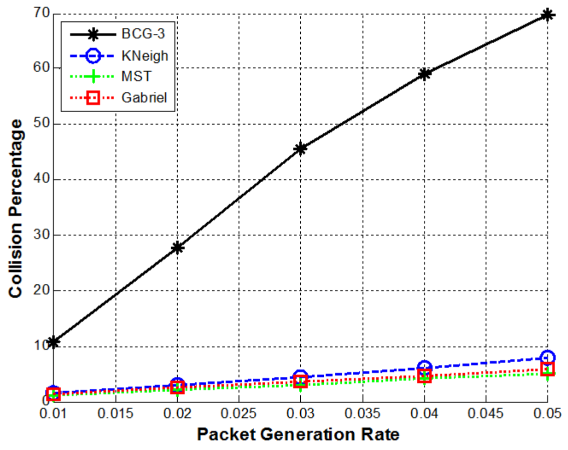
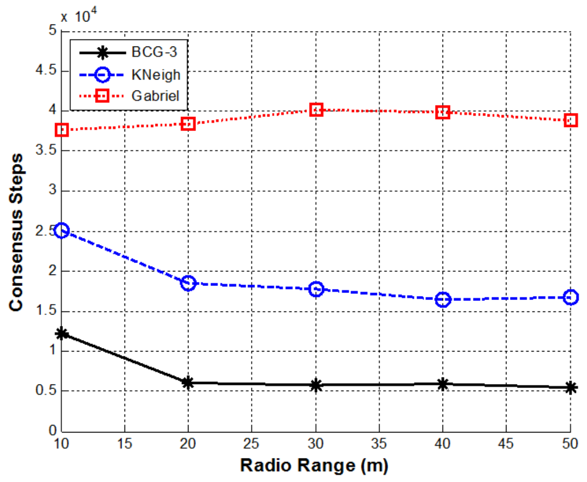


Figure 3.20: Collision percentage as a function of packet generation rate.

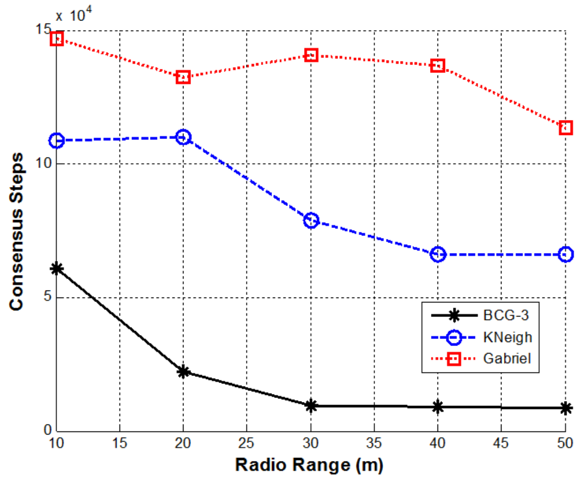
shows that BCG-3 produced much more collisions as packet generation increases.

Figures 3.21(a) and 3.21(b) show consensus steps required to achieve each STD of 0.25 and 0.1 with a packet generation rate of 5×10^{-3}, as a function of radio range. BCG-3 showed the best performance in both conditions. BCG-3 required 71% and 88% less consensus steps when compared against K-Neigh and Gabriel graphs with 30m radio distance and a STD of 0.25, respectively. In case of a STD of 0.1, BCG-3 with 30m radio distance required only 80% and 90% less consensus steps than K-Neigh and Gabriel graphs, respectively.

We also simulated with various packet generation rates with 30m radio range and a STD of 0.1. In Figure 3.22, BCG-3 is efficient in terms of consensus steps and power consumption under a packet generation rate of 0.04. However, as increases a packet generation rate, BCG-3 required more consensus steps. It is because the number of collisions are rapidly increased more than others due to longer communication radio constructed. Consensus protocol on BCG-3 showed worse performance than K-Neigh and Gabriel above a 0.04 packet generation rate.

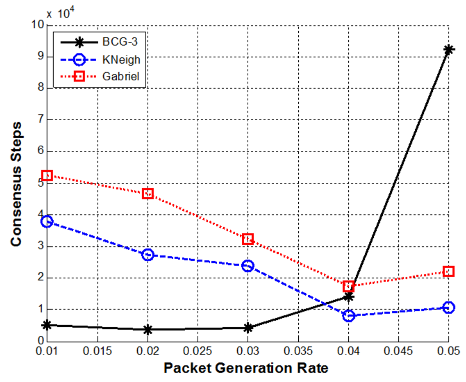


(a) Comparison of consensus step for a STD of 0.25 with 0.005 packet generation rate.

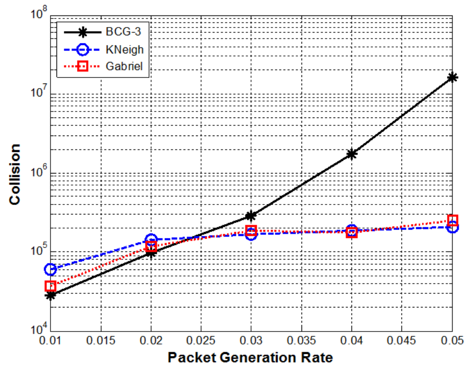


(b) Comparison of consensus step for a STD of 0.1 with 0.005 packet generation rate.

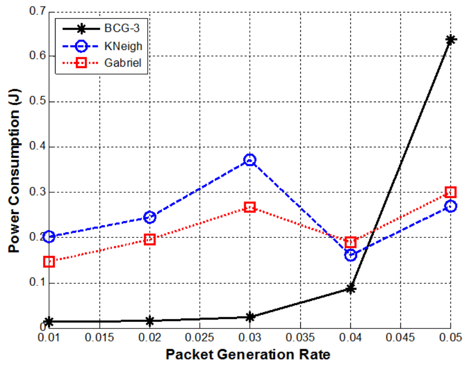
Figure 3.21: Comparison of consensus step for STDs of 0.25 and 0.1 with 0.005 packet generation rate.



(a) Consensus step.



(b) Packet collision.



(c) AVG. Power consumption.

Figure 3.22: Comparison of consensus step, packet collision and power consumption with various packet generation rates and 30m radio range.

Chapter 4

Distributed and Fault-tolerant Routing for Borel Cayley Graph

4.1 Introduction

Deterministic characteristics for connections between nodes in structured graphs allow theoretical analysis and guarantee global properties such as diameter and average path length [93]. Also graph based networks can have symmetry, hierarchy, connectivity, and hamiltonicity [36,50]. Those properties are desirable comparing random graph based networks.

Even though BCGs have such favorable properties, there are practical limitations in applying BCGs. One of them is the lack of fault-tolerant routing algorithms: existing BCG routing algorithms do not account for node or communication link failures.

In this section, we present a fault-tolerant routing algorithm for BCGs, which accounts for communication link failures. For our proposed 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 packet delivery performance from 20% to 350%. We also investigate packet congestion with our proposed algorithms for different packet generation rates.

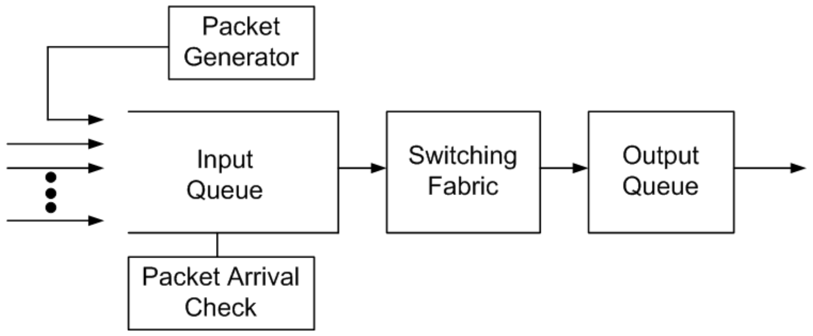


Figure 4.1: Node model.

4.2 Topology Comparison

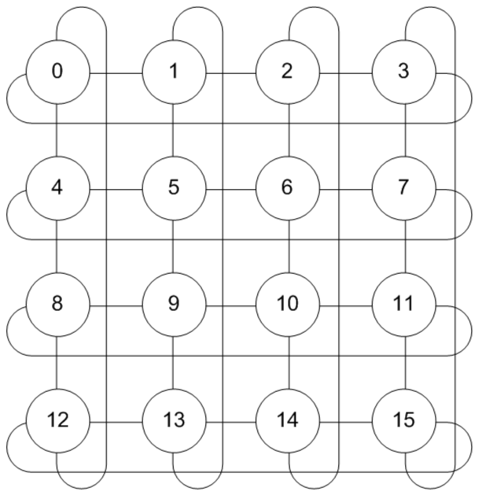
4.2.1 Network model

A network consists of a set of nodes connected by full duplex point-to-point links. Our node model is depicted in Figure 4.1. 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.

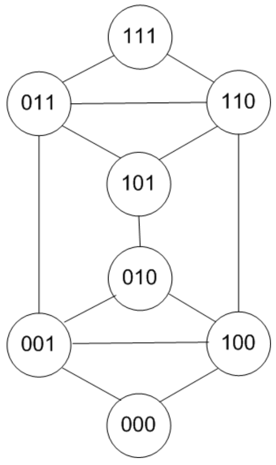
The input queue is FIFO served and accepts up to the number of packets equal to the degree of a node at a single time unit. The output buffer size is one and one packet is transmitted from a node at each time slot. This is consideration of our potential application of wireless sensor networks. Each wireless sensor node has one transmitter and one receiver as usual. The node model receives multiple packets simultaneously. On the other hand in section 4.5.3 it considers interference effect; only one packet is received successfully when there are multiple receiving packets. Depending on our simulation setup, the input buffer size ranges from one to infinity. Incoming packets are discarded when the input buffer is full.

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 with uniform distribution (Packet generator).

Three traffic patterns are considered in this paper: All-to-All traffic pattern



(a) 4 x 4 toroidal mesh network.



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

Figure 4.2: Toroidal mesh network and de Bruijn graph.

(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 generates 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 [94]. In Pattern 2, nodes send packets to a group of nodes, 5% of the total nodes in this thesis. This traffic pattern model depicts a situation having multiple sink nodes. It is also used in integrated devices. Nodes located on the borders of device chips can be connected to high-capacity transmission lines.

4.2.2 Topologies

Toroidal mesh networks and de Bruijn graphs are popular topologies for interconnection networks [37] [41] [49]. 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.

Toroidal Mesh Network

Figure 4.2(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 [95]. 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.

De Bruijn

The undirected de Bruijn graph (UDB) has $N = d^k$ nodes of degree $2d$ [37]. 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 4.2(b) shows an undirected de Bruijn graph for $k = 3$ and $d = 2$ where self-loops are removed.

Algorithm 4 Undirected de Bruijn graph routing algorithm.

```
 1: procedure ROUTING ALGORITHM FOR UDB($cur$, $dst$)    ▷ $cur$: the current
    node, $dst$: the destination node
 2:     $i \leftarrow match\_fwd(cur, dst)$
 3:     $j \leftarrow match\_bwd(cur, dst)$
 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 4 corresponds to the routing algorithm for UDB. The routing algorithm consists of transmitting a packet to either its left or right neighbors [96]. Algorithm 4 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 4.2(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 [96], 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.

Performance comparison

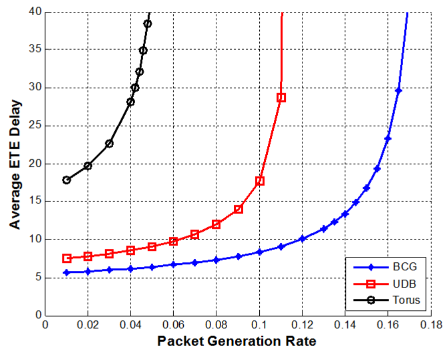
The parameters for the mesh networks we used for performance comparison are described in Table 4.1. BCG and Torus are degree 4 graphs. Most nodes of UDB have degree 4 except the few nodes with self-loops. Table 4.2 shows our benchmark mesh networks topological properties such as the diameter and the 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, BCGs have the smallest diameter and the shortest average path length.

Table 4.1: Mesh networks parameters.

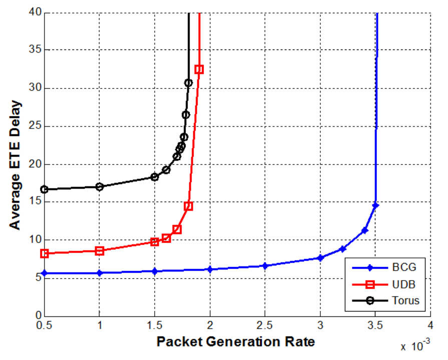
	N	Parameters
BCG	1081	$p = 47$, $k = 23$, $a = 2$, $t_1 = 1$, $t_2 = 7$, $y_1 = 1$, $y_2 = 1$
Torus	1088	$R = 32$, $C = 34$
UDB	1024	$d = 2$, $k = 10$

Table 4.2: Static property.

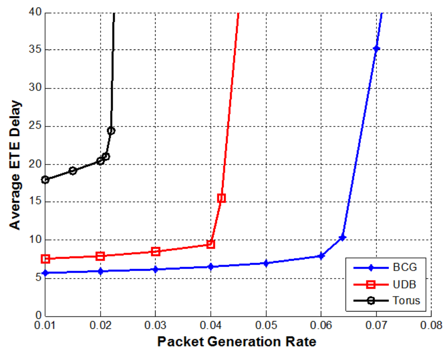
	AVG. Path length	Diameter
BCG	5.54	7
Torus	16.52	33
UDB	6.77	10



(a) Pattern 0 (All-to-All).

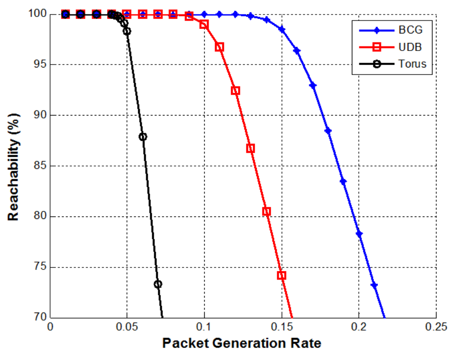


(b) Pattern 1 (All-to-One).

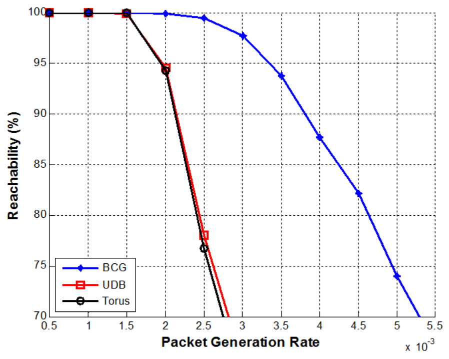


(c) Pattern 2 (All-to-M).

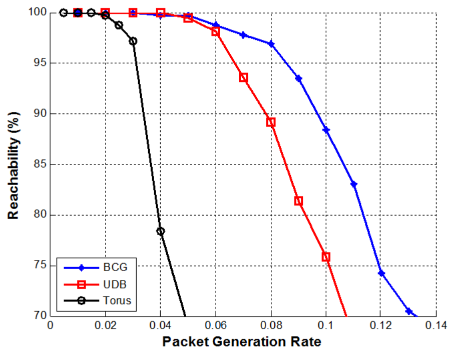
Figure 4.3: End-to-End Delay with infinite buffer.



(a) Pattern 0 (All-to-All).

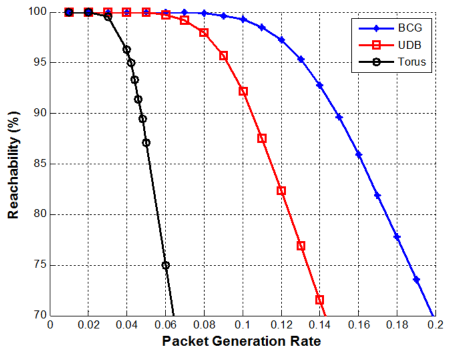


(b) Pattern 1 (All-to-One).

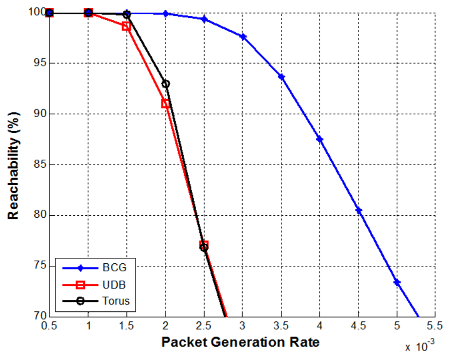


(c) Pattern 2 (All-to-M).

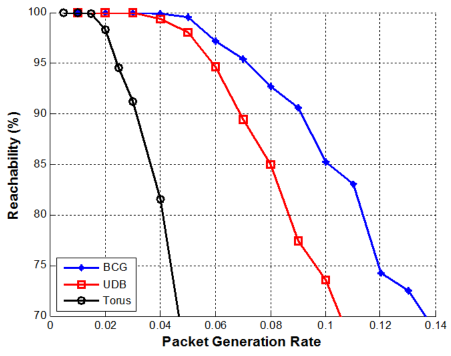
Figure 4.4: Reachability with buffer length 10.



(a) Pattern 0 (All-to-All).



(b) Pattern 1 (All-to-One).



(c) Pattern 2 (All-to-M).

Figure 4.5: Reachability with buffer length 5.

We also considered the performance of our network models using the following two metrics: (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 4.2.1. We set the input buffer length to infinite, 5, and 10. Our simulation time was $100,000$ ticks.

Figure 4.3 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 degrades. Figures 4.4 and 4.5 show reachabilities with buffer length 10 and 5, respectively. Reachability with finite buffer decreases above certain packet generation rate. Packet generation rate of decreasing reachability of BCG is higher than others even though they have almost the same number of nodes and edges.

4.3 Data Structure of Exhaustive Routing

Vertex-transitive routing guarantees the shortest path between any source-destination pair based on a routing table identical at every node. 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. We define two types of routing tables to account for 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.

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 in addition of BCG shortest paths (Static Routing Table). A Dynamic Routing Table for Type 2 packets is created to indicate "next best" paths as well as any unusable links due to failures. Basically in Phase 2 packets are routed according to the path length information and are used to update unusable links in the Dynamic Routing Table.

4.3.1 Static Routing Table

The Static Routing Table is pre-calculated and identical across all nodes. The Static Routing Table is defined for reference source node 0. Each time a message needs to be routed from a node different than node 0, the destination ID is mapped from an absolute ID (Global ID of the destination in a network) to a relative ID (ID of the destination in the view of the current node regarded as a reference node) as follows, referring to Table 2.2.

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

where the current node's absolute ID is $m_1q + c_1$, the destination node's absolute ID is $m_2q + c_2$, and j' is the relative destination ID. We denote the absolute ID of node u by $aID(u)$ and the relative ID of node u at node v by $rID(u, v)$. Row indexes in the 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 4.6(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 [97].

The first row of the Vertex-transitive routing table in Figure 4.6(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 4.6(b), the shortest path is through generator g_1^{-1} and is two hops 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 SRTBL$(rID(v, u))$ and the hop count through generator g from node u to node v by SRTBL$(rID(v, u), g)$. For example, SRTBL$(rID(3, 0)) = (1, 3, 3, 3)$ and

	g_1	$g^{-1}{}_1$	g_2	$g^{-1}{}_2$
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 Vertex-transitive routing.

	g_1	$g^{-1}{}_1$	g_2	$g^{-1}{}_2$
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 4.6: Original Vertex-transitive routing table and Static Routing Table of BCG in Figure 2.1(a). Note that some part of routing tables are omitted for brevity.

SRTBL$(rID(3,0), g_2) = 3$.

4.3.2 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 4.4. 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 the 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 4.3 summarizes routing tables for Exhaustive routing.

Table 4.3: Summary of routing tables for Exhaustive routing.

SRTBL	Pre-calculated Cells are not changed Cells are path lengths via generator Used in Phase 1 and Phase 2
DRTBL	Generated dynamically Cells are bits indicating the available links Cells are updated when node/link failures occur
D1RTBL	For Phase 1 Initialized with cells indicating shortest path links
D2RTBL	For Phase 2 Initialized with cells indicating all links are to be explored

4.4 Phases of Exhaustive Routing

4.4.1 Phase 1

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

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 each destination defined 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, because one or more shortest path links are disconnected, 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:

$$\text{Packet frame} = \{\text{Source}, \text{Destination}, \text{Packet Type}, \text{Path history}\}.$$

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 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, when node a determines the packet whose destination is node b ($rID(b, a) = 5$) is a returned packet via generator g_1, it generates the D1RTBL for node b using Figure 4.6(b). we get D1RTBL$(aID(b)) = (1, 1, 1, 0)$ since the smallest number is 3 in the SRTBL$(rID(b, a) = 5) = (3, 3, 3, 4)$. This sets the entry indicated by generator g_1 to zero. Finally, node a has D1RTBL$(aID(b)) = (0, 1, 1, 0)$

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 5 summarizes Phase 1 of our Exhaustive routing algorithm.

4.4.2 Phase 2

Table generation rules and packet forwarding rules for Type 1 and Type 2 packets are different. A Type 1 packet returns to its source when there is no shortest path within D1RTBL. At the point 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 entry for the packet's destination.

A packet gets stuck at a node having no available shortest paths in Phase 1. In Phase 2, the node receiving Type 2 packets updates or generates a D2RTBL entry for the destination node. The D2RTBL is initialized to 1 for 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

Algorithm 5 Phase 1 of Exhaustive routing.

```
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 $pkt$ is a returned packet then
7:          Update Dynamic Routing Table              ▷ Case: node in Path history
8:      end if
9:      if $pkt.dst$ is in 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
```

an entry of 1 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 by exploiting more available paths to destination. Figure 4.7 shows our Exhaustive routing algorithm flow.

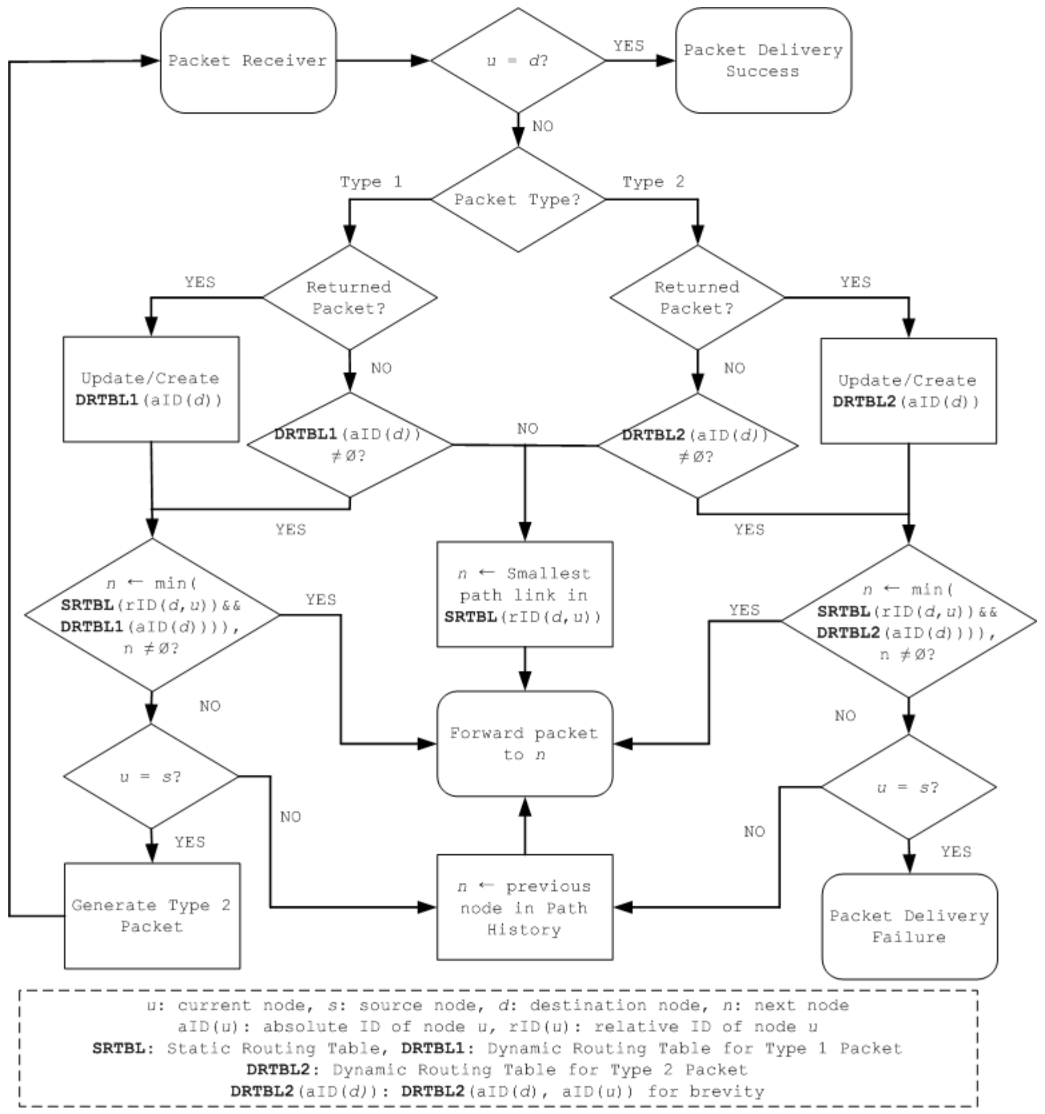

Packet Receiver
u = d?
YES
Packet Delivery Success
NO
Packet Type?
Type 1
Type 2
Returned Packet?
YES
NO
Update/Create DRTBL1(aID(d))
DRTBL1(aID(d)) ≠ Ø?
YES
NO
DRTBL2(aID(d)) ≠ Ø?
Update/Create DRTBL2(aID(d))
n ← min(SRTBL(rID(d,u)) && DRTBL1(aID(d)))), n ≠ Ø?
n ← Smallest path link in SRTBL(rID(d,u))
n ← min(SRTBL(rID(d,u)) && DRTBL2(aID(d)))), n ≠ Ø?
Forward packet to n
u = s?
Generate Type 2 Packet
n ← previous node in Path History
Packet Delivery Failure
u: current node, s: source node, d: destination node, n: next node
aID(u): absolute ID of node u, rID(u): relative ID of node u
SRTBL: Static Routing Table, DRTBL1: Dynamic Routing Table for Type 1 Packet
DRTBL2: Dynamic Routing Table for Type 2 Packet
DRTBL2(aID(d)): DRTBL2(aID(d), aID(u)) for brevity

Figure 4.7: Exhaustive routing algorithm flow.

The phase 2 of Exhaustive routing can create loops as shown in Figure 4.8.

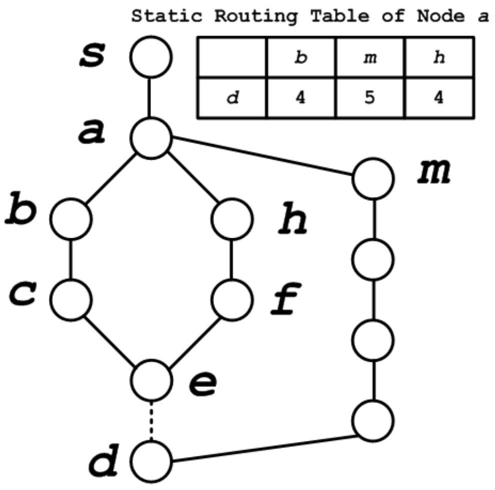


Figure 4.8: Network to illustrate a loop of Exhaustive routing. The dot line between nodes indicates that the communication is disconnected

Assume node s sends a packet to node d via a. The packet reaches node e via c but the communication link is disconnected. Then, after excluding the incoming link, the packet can only be forwarded to node f. The packet follows in a circle like $a \rightarrow b \rightarrow c \rightarrow e \rightarrow f \rightarrow h \rightarrow a \rightarrow b$. To prevent loops, Phase 2 uses a different method to check whether or not 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 when node a receives a returned packet, it refers to the Path history (s, a, b, c, e, f, h). Node a sets to zero the entry for the generator link to node b in $\mathsf{D2RTBL}(d, a)$. Node a also sets the entry for the generator link to node h to zero through the same method. Finally, node a delivers the packet via node m using Path history (s, a).

4.5 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 4.1. 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

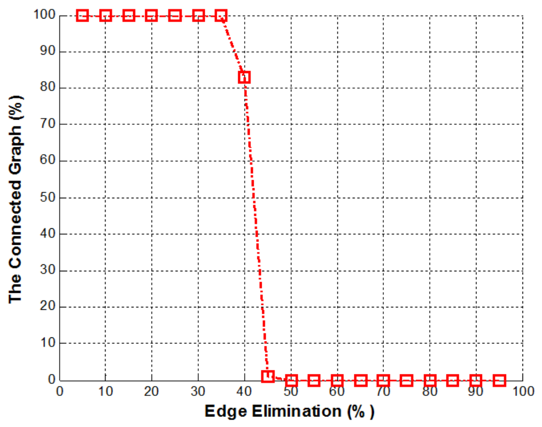


Figure 4.9: Connected BCG percentage as a function of edge elimination.

the second generator. Using two different generators and their inverse generators, we constructed an undirected degree-4 BCG. We arbitrarily chose parameters t_s and y_s for generators.

4.5.1 Static performance analysis

First, we simulated network disconnection by edge eliminations on BCG. We randomly selected edges to be eliminated. For each simulated case and each $\%$ edges eliminated, we generated 100 networks, then ran our routing algorithms, and evaluated their performance. The BCG is originally a connected graph. When we eliminate some edges, the network can consist of multiple network components (Components are not connected to one another). We measured packet delivery performance to the largest component only if the largest component has over 95% of the total nodes. Figure 4.9 shows the percentage of connected graphs among the 100 network samples for each edge elimination fraction. From those results, we simulated BCG ranging from 5% to 35% elimination of edges because the connected graph is kept within that range.

We showed the following two metrics to evaluate our proposed routing algorithms:

- Routability: the number of reachable source and destination pairs among all pairs of nodes in the largest component of a network.
- Average Hop Count: the average number of nodes traversed by a packet

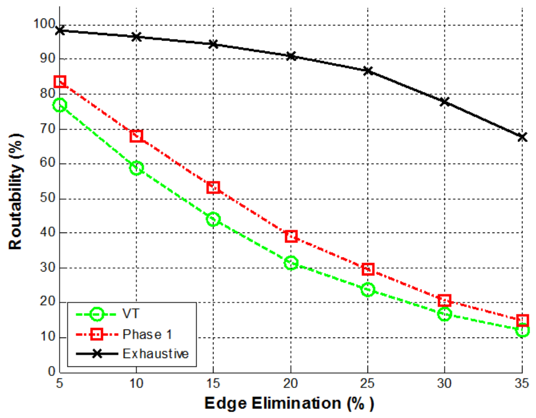


Figure 4.10: Routability with 1081 BCG after edges are eliminated.

between its source and destination within the largest component.

In this section, the simulator did not generate a packet before the previously sent packet was dropped or reached to its destination (only one packet exist in the network), which help measure the routing algorithm performance regardless of packet congestion and buffer length.

We compared Exhaustive routing with Phase 1 Exhaustive routing (Phase 1 routing) and Vertex-transitive routing (VT routing). Results for the Exhaustive routing were acquired once the dynamic routing tables stabilized. For comparison purpose, in our implementation of the original VT routing, an optimal link is randomly selected chosen in cases where multiple optimal links exist. When there is no available link, the packet is discarded.

Routability

Figure 4.10 shows the routability metric for a 1081 ndoes BCG after edges elimination. Phase 1 routing has a larger routability than Vertex-transitive routing because Phase 1 routing exploits all shortest paths between source and destination. Exhaustive routing has the best routability.

Average Hop Count

Figure 4.11 shows the average hop count metric for a 1081 nodes BCG with 35% edge elimination. Comparing the average hop count of our proposed algorithms

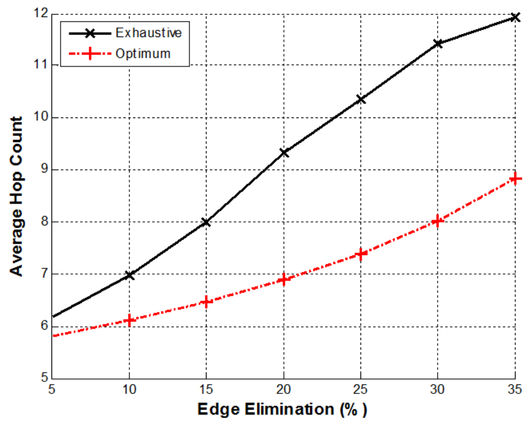


Figure 4.11: Average hop count with 1081 BCG after edges are eliminated.

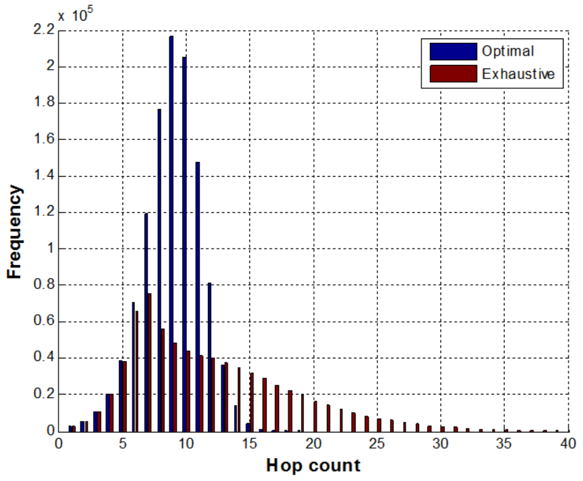


Figure 4.12: Hop count distribution of BCG with 1081 nodes after 35% edge elimination. Note that hop counts exceeding 40 are not shown for brevity.

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, the average hop count of the second algorithm is larger than the former. So we only compared the 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 are used. When a packet is not delivered to the destination, we exclude it from the average hop count computation.

Distribution of Hop Counts

We also investigated the frequency of hop counts after node elimination. Figure 4.12 shows the hop counts distribution. The histogram of hop counts exhibits a right skewed distribution with a high frequency of short hop counts.

4.5.2 Dynamic performance analysis

Next, we investigate the dynamic properties of our routing algorithms. This consists of measuring the performance when multiple packets are flowing simultaneously in the network while edges are being eliminated, as opposed to the static case in Section 4.5.1. In this case, packet generation rate and buffer length become important parameters. We used the network model described in Section 4.2 and removed a randomly chosen edge each time unit according to the edge failure rate. Simulation run time was $1,000,000$ ticks that is ten times longer than in the static case. This was required since more time is needed to observe the effects of link failures. The evaluation was done in terms of reachability function of packet generation rate, edge failure rate, and buffer length.

Table 4.4: Routing comparison with infinite buffer, 0.05 packet generation rate, and 0.0005 edge failure rate. Note that the result is accumulated during the whole simulation time.

	Reachability	ETE Delay	Buffer Length
VT	56.97%	6.14	0.23
Phase1	60.71%	6.23	0.19
Exhaustive	92.75%	9.19	0.43

Table 4.4 shows reachability, the average ETE delay, and average occupied buffer length with infinite buffer and an edge failure rate of 0.0005 during the whole simulation time. Exhaustive routing produced the highest reachability but also the longest End-to-End delay. End-to-End delay excludes non-reached packets.

Figure 4.13 shows reachability as a function of simulation time with a packet generation of 0.05 and a failure rate of 0.001. Exhaustive routing shows much

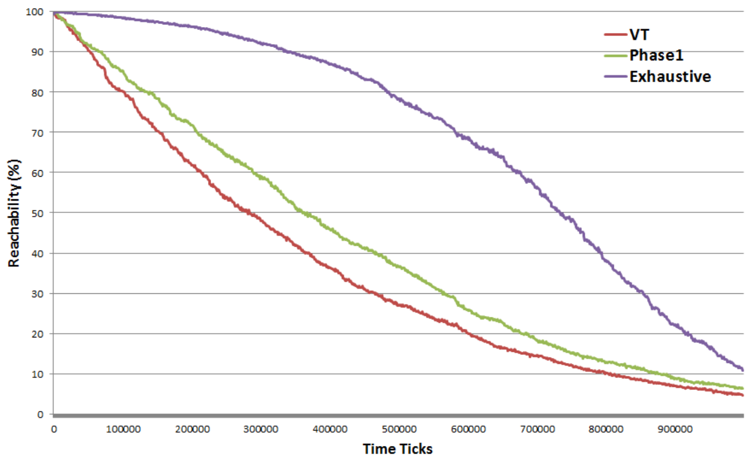


Figure 4.13: Reachabilities of routings with a packet generation rate of 0.05 and an Edge failure rate of 0.001 by simulation ticks.

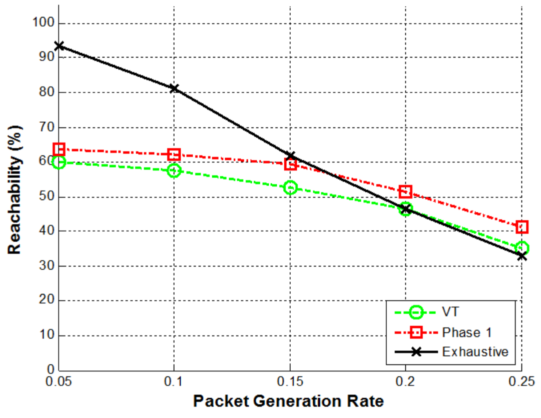
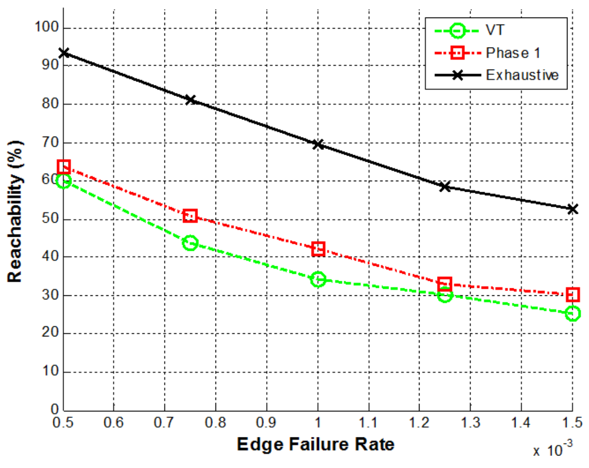
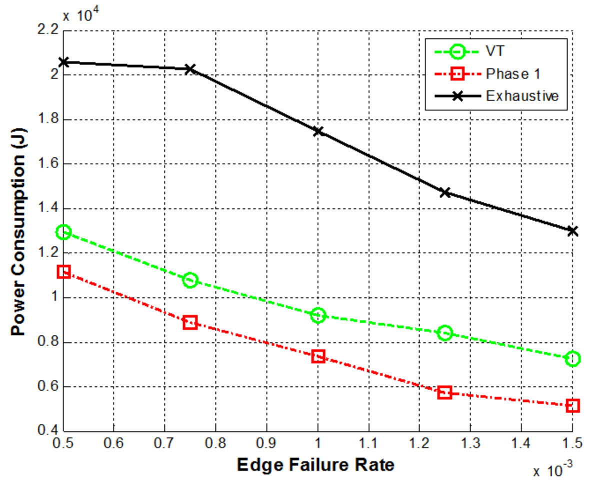


Figure 4.14: Comparison of fault-tolerant routing reachability with buffer length 5 and 0.0005 edge failure rate according to packet generation rate. Note that the result is accumulated during the whole simulation time.

better reachability up to simulation time of about $600,000$. However, after that point, the advantage decreased continuously. It is because a BCG network can be multiple components when over 30% edges are removed. Among nodes at each component, it does not have a path to route. The number of node pair having no path rapidly increases as the number of components increases.



(a) Routing reachability.



(b) Power consumption.

Figure 4.15: Comparison of fault-tolerant routing reachability and power consumption with buffer length 5 and 0.05 packet generation rate according to edge failure rate. Note that the result is accumulated during the whole simulation time.

Figure 4.14 showed the reachability result as a function of packet generation rate ranging from 0.05 to 0.25. Exhaustive routing with a 0.05 packet generation rate produced 56% and 47% better reachability than VT routing and Phase 1, respectively. However as a packet generation rate increases, the performance of our routing algorithms becomes similar because of network capacity and overflowed packets.

Phase 1 routing with a packet generation rate of 0.25 produced 26% better reachability than Exhaustive routing because Exhaustive routing has a longer average hop count to support high reachability in the static case. With finite input buffers, long average hop count and high reachability make buffer length full, which causes more dropped packets. Phase 1 routing produced better reachability than VT routing even at high packet generation rates. The only shortest path length guaranteed packets are generated because Phase 1 routing uses the Dynamic Routing Table to send packets. This results in the buffer length occupancy of Phase 1 smaller than one of VT.

In Figure 4.15(a), we investigated the reachability metric as a function of edge failure rate for a packet generation rate of 0.05. When we changed edge failure rate, the frequency of edge failure and the total number of fault edges during the whole simulation time changes. 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 the edge failure rate, Exhaustive routing shows 50% to 100% better reachability. Reachabilities of VT routing, Phase 1 routing, and Exhaustive routing decreased by 42%, 47%, and 56%, respectively, as edge failure rate increases from 0.0005 to 0.0015. In Figure 4.15(b), Exhaustive routing consumed more power than other routing algorithms because of its better reachability and longer average hop count.

4.5.3 Comparison Exhaustive routing with AODV

AODV

Ad hoc On Demand Distance Vector protocol (AODV) is a routing algorithm between mobile computers, which is basically an improvement of DSDV [17, 98]. As opposed to DSDV, AODV is a reactive protocol that establishes paths when needed. Nodes only know the next hop to destination and send packets through their neighbors to destination nodes with which they cannot directly communicate.

AODV achieves routing by discovering the routes along which messages can be passed. When a source node needs to send packets to a destination and cannot get the routes from its routing table, it will broadcast a Route Request (RREQ). If the receiver has a route to the destination, it unicasts a Route Reply (RREP) back

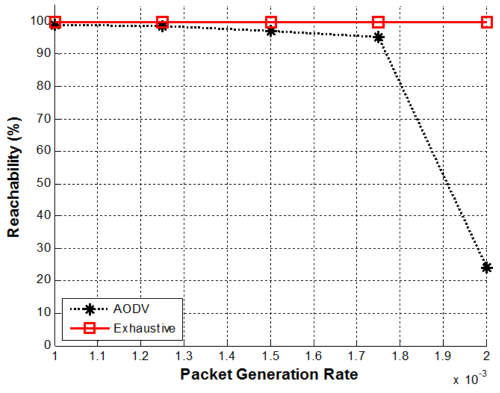


Figure 4.16: Reachabilities of AODV and Exhaustive routing according to packet generation rate.

to the source node. Otherwise, the RREQ is rebroadcasted further. If a RREP is sent, all nodes along that path record the route to the destination from this packet. A node can receive the same RREQ more than once when there exist multiple paths between two nodes.

To prevent the same request from being broadcast repeatedly, RREQ is uniquely identified by a source node ID and a broadcast ID. A node keeps track of its neighbors by listening for a HELLO packet that each node broadcasts at set intervals. Nodes send out Route Error (RERR) packets to their neighbors when they do not receive a HELLO packet. To avoid routing loops and identify the freshness of a route, sequence number is used. A larger sequence number denotes a fresher route. More details about AODV can be found in [17].

Simulation

We simulated 506 nodes BCG networks for $10,000$ simulation ticks. We set parameters $p = 23$, $a = 7$, $t_1 = 1$, and $t_2 = 7$. In AODV simulation, we assume nodes immediately recognize disconnection with neighbors without HELLO packets.

In Figure 4.16, the reachability of AODV decreases rapidly beyond a packet generation rate of 0.00175. On the other hand, Exhaustive routing reachability is stable for a packet generation rate up to 0.17. Because AODV uses broadcasting to communicate, the network using AODV saturates earlier than the network using Exhaustive routing. Regarding power consumption, AODV consumed more 400

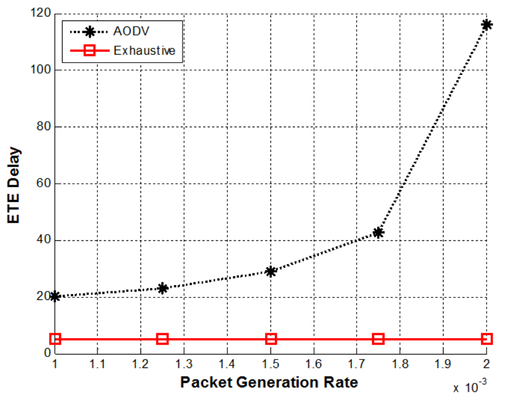


Figure 4.17: End-to-End Delay of AODV and Exhaustive routing according to packet generation rate.

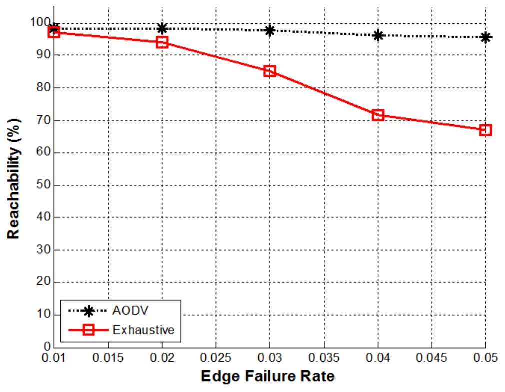


Figure 4.18: Reachabilities of AODV and Exhaustive routing according to edge failure rate with 0.001 packet generation rate.

times than Exhaustive routing. Although Exhaustive routing showed better ETE delay than AODV in Figure 4.16, AODV is more robust than Exhaustive routing regarding topology changes by edge failure. Figure 4.18 shows both reachabilities function of edge failure rate. At 0.04 edge failure rate, AODV produced a 96% reachability while Exhaustive routing produced a 71% reachability.

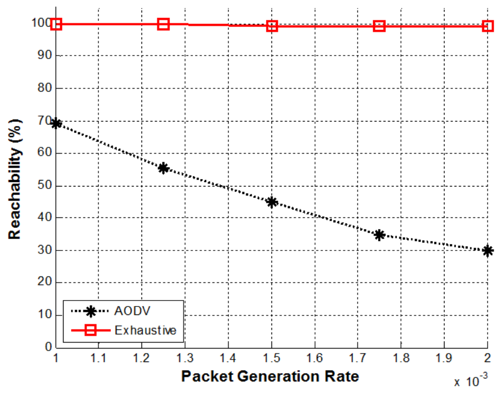


Figure 4.19: Reachabilities of AODV and Exhaustive routings as a function of packet generation rate with interferences.

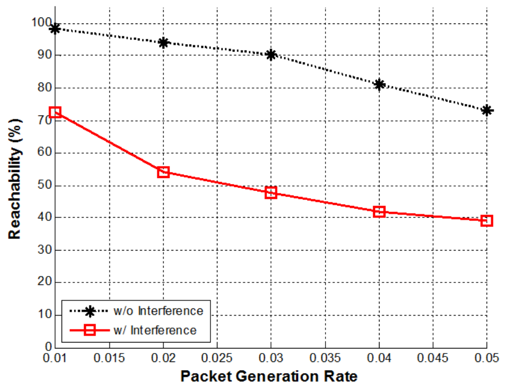


Figure 4.20: Reachabilities of Exhaustive routing function of packet generation rate and with interferences.

Simulation with interference

We used the same interference model as the one described in Section 3.6.5. When considering interference, packets can be lost due to communication distance or transceivers near the same receiver which send packets simultaneously. Figure 4.19 shows reachabilities of AODV and Exhaustive routing for a packet generation rate ranging from 1×10^{-4} to 5×10^{-4}. Exhaustive routing shows better performance because the network using AODV has more transmitting packets, which means the chance of packet collision increases.

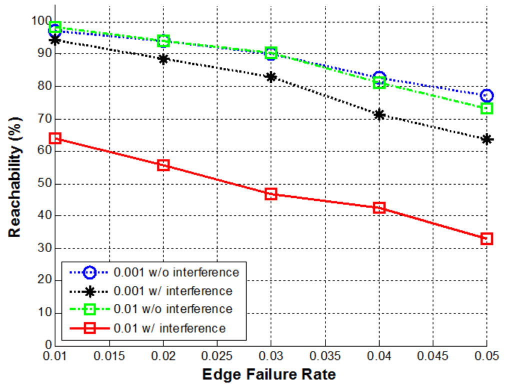


Figure 4.21: Reachabilities of Exhaustive routing function of edge failure rate and considering interference.

Figure 4.20 shows the reachability of Exhaustive routing for a packet generation rate ranging from 0.01 to 0.05. It shows a performance degradation of about 30% due to packet collisions. Figure 4.21 compares reachability for different edge failure rates and packet generation rates. For packet generation rates 0.001 and 0.01, their reachabilities are similar. When considering interferences, the network with a packet generation rate 0.01 shows a large performance degradation. The performance of packet generation rate 0.01 decreased more 30% than that of packet generation rate 0.001.

Chapter 5

Conclusion and Future Research

In this dissertation, we investigated structured graph based wireless sensor networks to achieve fast information dissemination and efficient data exchange in large and dense networks. In particular, we focused on the strengths, limitations and solutions, and applications of Borel Cayley graphs (BCG). We summarize our contributions and provide future research plans.

5.1 Contribution

We applied Borel Cayley graphs to wireless sensor networks for topology control. We proposed node ID assignment methods to reduce the communication distance between nodes or to distributively increase the number of logical connections following the predefined graph connection rules. In particular, we proposed the Chordal Ring based node ID assignment method and the distributed node ID swapping assignment method.

In the Chordal Ring based method, we exploited the fact that all BCGs have CR representations which imply the existence of a Hamiltonian cycle with consecutive integer numbers. With this observation, node IDs can simply be progressively assigned in numerical order based on their physical locations. Once the node IDs are assigned in the CR domain, we provide an explicit formulation to convert the CR domain ID to the GCR domain ID; which supports efficient routing. Our simulation results over networks of sizes ranging from 300 to 2000 nodes showed that the CR based node ID assignment requires a smaller radio range (57%) than that of the random ID assignment.

In the distributed node ID swapping assignment method, we consider that nodes have a limited transmission range. In WSNs, it is challenging to impose an entire logical graph to a physical network of finite radio range. An efficient node ID assignment will allow more connections to be imposed. For a range of network sizes from 200 to $1,000$ nodes, simulation showed that our proposed method achieved 43% more desired connections than that of the Random assignment.

We quantified the performance of our proposed topology control algorithms by consensus protocol when compared to existing topology controls such as K-Neigh, Gabriel, and E-MST. We also investigated the effects of interferences on BCG based networks for consensus protocol. Wireless packet transmission is interfered by simultaneous transmission of packets among nodes in the network.

In networks, communication failures are common. The existing Vertex-transitive routing for Borel Cayley graphs 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 paths available in the Borel Cayley graph. When BCG shortest paths are disconnected, Phase 2 of Exhaustive routing is used to exploit non-shortest paths in the Borel Cayley graph. Through simulation, we found that the proposed Exhaustive routing produced 20% to 350% better routability than Vertex-transitive routing with various amount of link failures. Regarding the average hop count, Exhaustive routing produced paths 30% longer than Optimal routing. We also compared BCG network topology properties to torus and de Bruijn graphs with various traffic patterns.

When we consider simultaneous multiple packets flowing, Exhaustive routing for certain packet generation rates produced over 50% better reachability than Phase 1 and VT routings. However, with a high packet generation rate, Phase 1 routing produced better reachability because it sends packets along the guaranteed shortest paths of the original Borel Cayley graph. We also characterized the interference effects due to simultaneous transmission.

In summary, our proposed fault-tolerant routing and node ID assignment algorithms make it possible for Borel Cayley graphs to be deployed in realistic network scenarios.

5.2 Future Research

Wireless Sensor Networks

When wireless sensor nodes are mobile (mobile ad hoc networks), nodes may move in and out networks. When a node moves out the network or is deactivated due to power depletion, its node ID should be deregistered and stored for the next new incoming node. So that when a new node joins the network, it needs to be assigned a new node ID that is unique to the network. Especially within structured graph based networks, it is important to efficiently assign new node IDs due to limitation of node ID range. Size inflexibility of BCG makes this problem more difficult. Also a flexible node ID assignment can be used to reduce communication distance when nodes move.

In addition to flexible node ID assignment, hierarchial network construction is an issue. When a sensor deployment area becomes larger and radio range is limited, it may be difficult to map all nodes within one network component. In this case, we need to divide our network into multiple clusters. We envision that BCG topology can be applied within a cluster or between cluster heads. This concept could be used as basis for developing hierarchial behavior within BCGs.

Another application in wireless sensor networks is to apply BCG to distributed hash table (DHT). DHT can be used to routing algorithms, which is achieved by multi-hop communications between adjacent logical neighbors [24] or to resource management like P2P [99].

Network-on-Chip

In Network-on-Chip (NoC), a switching network consists of the interconnection of many switching units. The function of the switching is to enable these units to communicate with each other. A NoC interconnection is characterized by its topology, which is basically the underlying graph. Various topologies such as mesh networks, hypercubes, shuffle networks, butterfly networks, binary trees and fat trees have been proposed to build interconnection networks for many-core systems [42–44]. Network diameter, connectivity, bandwidth and latency are important metrics in NoC. BCGs are good candidates as the topology for NoC with those desired properties.

Bibliography

[1] D. Culler, D. Estrin, and M. Srivastava, “Guest editors’ introduction: Overview of sensor networks,” *Computer*, vol. 37, no. 8, pp. 41 – 49, 2004.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *Communications Magazine, IEEE*, vol. 40, pp. 102 – 114, aug 2002.

[3] C. E. Perkins, “Ad hoc networking,” ch. Ad hoc networking: an introduction, pp. 1–28, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[4] T. Schmid, H. Dubois-ferriere, and M. Vetterli, “Sensorscope: Experiences with a wireless building monitoring sensor network,” in *In Proc. First Workshop on Real-World Wireless Sensor Networks (REALWSN05)*, 2005.

[5] K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, “Deploying a wireless sensor network on an active volcano,” in *IEEE Internet Computing*, pp. 18–25, 2006.

[6] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, “Luster: wireless sensor network for environmental research,” in *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys ’07, (New York, NY, USA), pp. 103–116, ACM, 2007.

[7] K. Martinez, J. Hart, and R. Ong, “Environmental sensor networks,” *Computer*, vol. 37, no. 8, pp. 50 – 56, 2004.

[8] S. Coleri, S. Y. Cheung, and P. Varaiya, “Sensor networks for monitoring traffic,” in *In Allerton Conference on Communication, Control and Computing*, 2004.

[9] T. Yan, T. He, and J. A. Stankovic, “Differentiated surveillance for sensor networks,” pp. 51–62, 2003.

[10] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, “The platforms enabling wireless sensor networks,” *Commun. ACM*, vol. 47, pp. 41–46, June 2004.

[11] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, pp. 325–349, 2005.

[12] J. Al-Karaki and A. Kamal, “Routing techniques in wireless sensor networks: a survey,” *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 6 – 28, 2004.

[13] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom ’99, (New York, NY, USA), pp. 174–185, ACM, 1999.

[14] S. M. Hedetniemi and A. Liestman, “A survey of gossiping and broadcasting in communication networks,” *Networks*, vol. 18, pp. 319–349, 1988.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion: A scalable and robust communication paradigm for sensor networks,” in *MOBICOM*, pp. 56–67, ACM, 2000.

[16] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, pp. 153–181, Kluwer Academic Publishers, 1996.

[17] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, WMCSA ’99, (Washington, DC, USA), pp. 90–, IEEE Computer Society, 1999.

[18] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, DIALM ’99, (New York, NY, USA), pp. 48–55, ACM, 1999.

[19] F. Zhang, H. Li, A. Jiang, J. Chen, and P. Luo, "Face tracing based geographic routing in nonplanar wireless networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 2243 –2251, May 2007.

[20] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognition*, vol. 12, pp. 261–268, 1980.

[21] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology*, vol. 18, no. 3, pp. pp. 259–278, 1969.

[22] J. M. Keil and C. A. Gutwin, "Classes of graphs which approximate the complete euclidean graph," *Discrete Comput. Geom.*, vol. 7, pp. 13–28, January 1992.

[23] T. Watteyne, I. Augé-Blum, M. Dohler, S. Ubéda, and D. Barthel, "Centroid virtual coordinates - a novel near-shortest path routing paradigm," *Comput. Netw.*, vol. 53, pp. 1697–1711, July 2009.

[24] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by dhts," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, (New York, NY, USA), pp. 351–362, ACM, 2006.

[25] M. Gerla, X. Hong, and G. Pei, "Landmark routing for large ad hoc wireless networks," in *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, vol. 3, pp. 1702 –1706 vol.3, 2000.

[26] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, "Glider: gradient landmark-based distributed routing for sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, pp. 339 – 350 vol. 1, 2005.

[27] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, p. 10 pp. vol.2, jan. 2000.

[28] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom ’01, (New York, NY, USA), pp. 272–287, ACM, 2001.

[29] R. Rajaraman, “Topology control and routing in ad hoc networks: a survey,” *SIGACT News*, vol. 33, pp. 60–73, June 2002.

[30] C. Bettstetter, “On the minimum node degree and connectivity of a wireless multihop network,” in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc ’02, (New York, NY, USA), pp. 80–91, ACM, 2002.

[31] R. Ramanathan and R. Rosales-Hain, “Topology control of multihop wireless networks using transmit power adjustment,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 404 –413 vol.2, 2000.

[32] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, “Distributed topology control for power efficient operation in multihop wireless ad hoc networks,” in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1388 –1397 vol.3, 2001.

[33] P. Santi, “Topology control in wireless ad hoc and sensor networks,” *ACM Comput. Surv.*, vol. 37, pp. 164–194, June 2005.

[34] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, “The k-neigh protocol for symmetric topology control in ad hoc networks,” in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc ’03, (New York, NY, USA), pp. 141–152, ACM, 2003.

[35] R. Wattenhofer and A. Zollinger, “Xtc: a practical topology control algorithm for ad-hoc networks,” in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, p. 216, Apr. 2004.

[36] S. B. Akers and B. Krishnamurthy, “A group-theoretic model for symmetric interconnection networks,” *IEEE Trans. Comput.*, vol. 38, pp. 555–566, April 1989.

[37] K. N. Sivarajan and R. Ramaswami, “Lightwave networks based on de bruijn graphs,” *IEEE/ACM Trans. Netw.*, vol. 2, pp. 70–79, February 1994.

[38] K. Zhu and B. Mukherjee, “Traffic grooming in an optical wdm mesh network,” *Selected Areas in Communications, IEEE Journal on*, vol. 20, pp. 122 –133, jan 2002.

[39] R. Duncan, “A survey of parallel computer architectures,” *Computer*, vol. 23, pp. 5–16, February 1990.

[40] R. D’Andrea and G. Dullerud, “Distributed control design for spatially interconnected systems,” *Automatic Control, IEEE Transactions on*, vol. 48, pp. 1478 – 1495, sept. 2003.

[41] J. Sun and E. Modiano, “Capacity provisioning and failure recovery for low earth orbit satellite constellation,” 2003.

[42] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of network-on-chip,” *ACM Comput. Surv.*, vol. 38, June 2006.

[43] H. Moussa, A. Baghdadi, and M. Jezequel, “Binary de bruijn interconnection network for a flexible ldpc/turbo decoder,” in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 97 –100, may 2008.

[44] A. El Gamal, “Trends in cmos image sensor technology and design,” in *Electron Devices Meeting, 2002. IEDM ’02. Digest. International*, pp. 805 – 808, 2002.

[45] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, “A survey and comparison of peer-to-peer overlay network schemes,” *Communications Surveys Tutorials, IEEE*, vol. 7, no. 2, pp. 72 – 93, 2005.

[46] 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 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM ’01, (New York, NY, USA), pp. 149–160, ACM, 2001.

[47] M. F. Kaashoek and D. R. Karger, “Koorde: A simple degree-optimal distributed hash table,” 2003.

[48] M. Naor and U. Wieder, "Novel architectures for p2p applications: The continuous-discrete approach," *ACM Trans. Algorithms*, vol. 3, August 2007.

[49] D. Loguinov, J. Casas, and X. Wang, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 1107–1120, October 2005.

[50] C. Qu, W. Nejdl, and M. Kriesell, "Cayley dhts - a group-theoretic framework for analyzing dhts based on cayley graphs," in *In International Symposium on Parallel and Distributed Processing and Applications (ISPA)*, Springer-Verlag, 2004.

[51] E. Noel and W. Tang, "Novel sensor mac protocol applied to cayley and manhattan street networks with crossbow mica2," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, vol. 2, pp. 626 –631, Sept. 2006.

[52] A. Taleb, J. Mathew, and D. Pradhan, "Fault diagnosis in multi layered de bruijn based architectures for sensor networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 456 –461, 2010.

[53] S. S. Iyengar, D. N. Jayasimha, and D. Nadig, "A versatile architecture for the distributed sensor integration problem," *IEEE Trans. Comput.*, vol. 43, pp. 175–185, February 1994.

[54] A. A.-B. Al-Mamou and H. Labiod, "Scatterpastry: An overlay routing using a dht over wireless sensor networks," *Intelligent Pervasive Computing, International Conference on*, vol. 0, pp. 274–279, 2007.

[55] F. Araujo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri, "Chr: a distributed hash table for wireless ad hoc networks," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pp. 407 – 413, 2005.

[56] K. W. Tang and B. W. Arden, "Representations of borel cayley graphs," *SIAM J. Discret. Math.*, vol. 6, pp. 655–676, November 1993.

[57] B. Arden and K. Tang, "Representations and routing for cayley graphs [computer networks]," *Communications, IEEE Transactions on*, vol. 39, pp. 1533 –1537, Nov. 1991.

[58] J. Yu, E. Noel, and K. W. Tang, "Degree constrained topology control for very dense wireless sensor networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM'10)*, pp. 1–6, Dec. 06–10, 2010.

[59] K. W. Tang and B. W. Arden, "Vertex-transitivity and routing for cayley graphs in gcr representations," in *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's*, SAC '92, (New York, NY, USA), pp. 1180–1187, ACM, 1992.

[60] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, pp. 1520 – 1533, sept. 2004.

[61] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215 –233, 2007.

[62] J. Yu, E. Noel, and K. W. Tang, "Pseudo-random graphs for fast consensus protocol," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'09)*, pp. 828–834, July 13–16, 2009.

[63] J.-C. Bermond, D. C., and J.-J. Quisquater, "Strategies for interconnection networks: some methods from graph theory," *J. Parallel Distrib. Comput.*, vol. 3, pp. 433–449, December 1986.

[64] M. Miller and J. Siran, "Moore graphs and beyond: A survey of the degree/diameter problem, electron," *Electronic Journal of Combinatorics, Dynamic survey D*, vol. 14, 2005.

[65] G. Panchapakesan and A. Sengupta, "On a lightwave network topology using kautz digraphs," *Computers, IEEE Transactions on*, vol. 48, pp. 1131 –1137, oct 1999.

[66] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Comput. Surv.*, vol. 34, pp. 313–356, September 2002.

[67] J. Yu, D. Kim, E. Noel, and K. W. Tang, "Dense and symmetric graph formulation and generation for wireless information networks," in *Proceedings of the International Conference on Wireless Networks and Information Systems (WNIS'09)*, pp. 379–384, Dec. 28–29, 2009.

[68] 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(17), pp. 2093–2104, 2010.

[69] K. Tang and B. Arden, "Class-congruence property and two-phase routing of borel cayley graphs," *IEEE Trans. Comput.*, vol. 44, no. 12, pp. 1462–1468, 1995.

[70] M. Ben-Ayed, *Dynamic routing for regular direct computer networks*. PhD thesis, Rochester, NY, USA, 1990.

[71] J. P. Hayes, T. N. Mudge, and Q. F. Stout, "Architecture of a hypercube supercomputer.," in *ICPP'86*, pp. 653–660, 1986.

[72] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Commun. ACM*, vol. 24, pp. 300–309, May 1981.

[73] J. Lin, Y. Liu, and L. Ni, "Sida: Self-organized id assignment in wireless sensor networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pp. 1 –8, Oct. 2007.

[74] E. Ould-Ahmed-Vall, D. M. Blough, B. H. Ferri, and G. F. Riley, "Distributed global id assignment for wireless sensor networks," *Ad Hoc Netw.*, vol. 7, pp. 1194–1216, August 2009.

[75] C. Sechen, "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *Proceedings of the 25th ACM/IEEE Design Automation Conference*, DAC '88, (Los Alamitos, CA, USA), pp. 73–80, IEEE Computer Society Press, 1988.

[76] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, *Optimization by simulated annealing*, pp. 551–567. Cambridge, MA, USA: MIT Press, 1988.

[77] D. Moore, J. Leonard, D. Rus, and S. Teller, “Robust distributed network localization with noisy range measurements,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys ’04, (New York, NY, USA), pp. 50–61, ACM, 2004.

[78] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, A. Lédeczi, G. Balogh, and K. Molnár, “Radio interferometric geolocation,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys ’05, (New York, NY, USA), pp. 1–12, ACM, 2005.

[79] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, “A high-accuracy, low-cost localization system for wireless sensor networks,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys ’05, (New York, NY, USA), pp. 13–26, ACM, 2005.

[80] L. Lazos and R. Poovendran, “Serloc: secure range-independent localization for wireless sensor networks,” in *Proceedings of the 3rd ACM workshop on Wireless security*, WiSe ’04, (New York, NY, USA), pp. 21–30, ACM, 2004.

[81] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, “Mobile-assisted localization in wireless sensor networks,” in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, pp. 172 – 183 vol. 1, march 2005.

[82] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 63 – 70, april 2005.

[83] Y. Hatano and M. Mesbahi, “Agreement over random networks,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, pp. 2010 – 2015 Vol.2, dec. 2004.

[84] L. Schenato and G. Gamba, “A distributed consensus protocol for clock synchronization in wireless sensor network,” in *Decision and Control, 2007 46th IEEE Conference on*, pp. 2289 –2294, dec. 2007.

[85] P. Braca, S. Marano, and V. Matta, “Running consensus in wireless sensor networks,” in *Information Fusion, 2008 11th International Conference on*, pp. 1 –6, 30 2008-july 3 2008.

[86] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, pp. 1847 –1864, nov. 2010.

[87] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 2371 – 2378 vol. 4, june 2005.

[88] M. Marta and M. Cardei, "Using sink mobility to increase wireless sensor networks lifetime," in *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pp. 1 –10, june 2008.

[89] N. Li, J. Hou, and L. Sha, "Design and analysis of an mst-based topology control algorithm," *Wireless Communications, IEEE Transactions on*, vol. 4, pp. 1195 – 1206, may 2005.

[90] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, (New York, NY, USA), pp. 243–254, ACM, 2000.

[91] H. Li and P. Mitchell, "Full interference model in wireless sensor network simulation," in *Wireless Communication Systems, 2009. ISWCS 2009. 6th International Symposium on*, pp. 647 –651, sept. 2009.

[92] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, (New York, NY, USA), ACM, 2006.

[93] G. Barrenetxea, B. Berefull-Lozano, and M. Vetterli, "Lattice networks: capacity limits, optimal routing, and queueing behavior," *Networking, IEEE/ACM Transactions on*, vol. 14, pp. 492 –505, june 2006.

[94] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 2–16, Feb. 2003.

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

[96] Z. Feng and O. Yang, "Routing algorithms in the bidirectional de bruijn graph metropolitan area networks," in *Military Communications Conference, 1994. MILCOM '94. Conference Record, 1994 IEEE*, pp. 957 –961 vol.3, oct 1994.

[97] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[98] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Proceedings of the conference on Communications architectures, protocols and applications*, SIGCOMM '94, (New York, NY, USA), pp. 234–244, ACM, 1994.

[99] C. Canali, M. Renda, P. Santi, and S. Burresi, "Enabling efficient peer-to-peer resource sharing in wireless mesh networks," *Mobile Computing, IEEE Transactions on*, vol. 9, pp. 333 –347, march 2010.