Stony Brook University

Graph Theory based Design of Scalable Network Systems

A Dissertation Presented

by

Dongsoo Kim

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Engineering

Stony Brook University

August 2014

Stony Brook University

The Graduate School

Dongsoo Kim

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

Jie Gao, Associate 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
Dean of the Graduate School

Abstract of the Dissertation

Graph Theory based Design of Scalable Network Systems

by

Dongsoo Kim

Doctor of Philosophy

in

Computer Engineering

Stony Brook University

2014

Due to the Internet's enormous growth in size, interconnecting heterogeneous networks reliably and efficiently has been a major topic in computer networks research for decades. Currently, more than four billions users (hosts) are connected through the Internet and the number is still growing. To achieve scalable networking, many researchers have invented and improved communication topologies and network protocols. Among these efforts, Graph theory plays an important role in protocol design and topology development. In this dissertation, we discuss graph theory based network design to enhance scalability and efficiency, and we provide applications to Wireless Sensor Networks (WSNs).

Borel Cayley Graphs (BCGs) are regular and dense graphs with promising properties when used as a communication topology: efficient topological/spectral properties, Generalized Chordal Ring (GCR) representation and vertex transitivity. However, a BCG's topological properties such as diameter and average path length have large variations depending on the choice of its discrete generators. Further, despite having a short diameter, a BCG can result in large communication distances between nodes.

In this dissertation, we propose *Expanded* Borel Cayley Graphs (*Ex*-BCGs), a systematic expansion of BCGs that preserve BCG's advantageous network prop-

erties. *Ex*-BCGs are based on node ID space expansion and connection rule redefinition, without changing the associated BCG generating parameters. We found that the resulting *Ex*-BCGs have scalable network properties even after a large scale size expansion. Using these properties, we provide the Class-level Vertex Transitive (CVT) routing protocol to construct a distributed and optimal routing algorithm with a small routing table. We exploit the Cut-Through Rewiring (CTR) algorithm to resize *Ex*-BCGs to arbitrary sized networks without downgrading the network connectivity and performance. For fault-tolerant routing in resized *Ex*-BCGs, we present the Aggressive Multi-path Aware (AMA) routing protocol that involves routing and rewiring around node failures. Simulation results over a variety of node failure rates show that the AMA routing protocol produces reliable reachability and efficient average routing path length.

For *Ex*-BCGs network applications, we construct a communication topology and design a routing algorithm for WSNs. We developed the *Ex*-BCGs Topology Construction (EBTC) algorithm to discover physical neighbors without collision and to establish efficient logical connections. This was made with the cycle characteristic of *Ex*-BCGs connections. As a result, EBTC constructs an energy-efficient communication topology with a small nodal degree and a short average path length. For routing in WSNs, we present the *Ex* Clustering algorithm that forms clusters for hierarchical routing in the communication topology generated from the modified EBTC operation. The *Ex* Clustering algorithm enables a deterministic time schedule for data transmission without collisions and provides energy saving by allowing nodes to enter sleep mode when not clustered. From comparison studies with LEACH and Selective LEACH algorithms, we observed that *Ex* Clustering produces lower energy consumption, longer network lifetime and higher reported event ratio.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Wireless sensor networks (WSNs) are application specific network systems to collect information from tens to thousands of sensor nodes. WSNs are used for a huge number of applications in military [1], health care [2–4], environment investigation [5–7] and smart home appliances [8,9]. A sensor node has three components: a sensor, a radio and a processor. Sensors perform tasks such as measuring temperature or pressure and detecting movements within sensing area [10]. Processors process sensed data and radios are used to communicate with other sensor nodes or base station (BS) for data exchange. WSN's purpose is to collect (sense) interested data from the target area and send it to the base station or users. However, it is a challenging process due to challenges such as limited energy resource, unstable wireless communication and interference. Before discussing these challenges in detail, we introduce categories of data delivery models and network characteristics in WSN as follows:

- **Data delivery models**

  Data delivery models in WSNs can be categorized into event-driven [11–13], periodic [14, 15] and request/reply (query-driven) [16] models. In event-driven model, only sensor nodes detecting change (exceeding permitted range) within sensing area send the collected data to base station, users or next relaying nodes. In contrast, some WSNs applications require sensor nodes to periodically report the collected data for seamless monitoring. In request/reply model, BS or end user makes a request to sensor nodes for data collection and receives a response from them.

- **Network characteristics**

  Another feature of WSNs is that sensor nodes can be equipped with identically or non-identically functioned units/resources. On one hand, all sensor nodes in homogeneous WSNs have identical energy, communication functionality, sensing range and CPU capacity [15, 17, 18]. On the other hand, in heterogeneous WSNs, a subset of sensor nodes have more energy, better radio and CPU capability [19–21]. The heterogeneous WSNs utilize these distinct nodes to perform more energy and CPU consuming job such as long distance data transmission or complicated data aggregation in hierarchical network.

The selection of data delivery model and network characteristic is dependent on the WSN's purpose. In the following sections, we will discuss the factors to be considered in designing WSNs and how to construct a communication topology and to design a routing protocol.

1.2 From Neighbor Discovery to Topology Control in WSNs

To overcome the challenges discussed in the previous section, it is critical to build an efficient communication topology once deployed. While much research on this topic has been done, neighbor discovery and topology control are regarded as the two main approaches for building a WSN communication topology. Neighbor discovery is a process for a node to find physical neighbors within its transmission range while minimizing packet collision. In most of the papers on neighbor discovery, a node broadcasts 'HELLO' message and receives a response from its physical neighbors which may follow a schedule to prevent collision at the receiving node. For example, [22] proposes that nodes enter listening, transmitting or sleeping mode with a probabilistic round-robin wake-up schedule based on the birthday protocol. As another probabilistic wake-up schedule based neighbor discovery, [23] proposes that if a node does not receive a beacon due to collision, it transmits a feedback message. If no feedback messages are received, the node enters a passive state. Different from the probabilistic wake-up schedule based neighbor discovery protocols, Dutta *et al* [24] proposed DISCO, a neighbor discovery protocol with a deterministic wake-up schedule based on the Chinese Remainder theorem, and [25] improves DISCO's performance by using an activation

pattern. The neighbor discovery protocols described above aim at collecting physical neighbor information without collision. However, it is non-trivial for a node to find all its physical neighbors without collision as there are unknown number of physical neighbors when nodes are randomly deployed.

Topology control aims at achieving specific design goals such as energy efficiency and interference mitigation by selecting logical neighbors and adjusting the transmission power accordingly. In [26], Blough *et al* provide a *K*-Neighbor topology control algorithm in which a node chooses *k* closest physical neighbors as logical neighbors. Nodes in Local Minimal Spanning Tree (LMST) topology control algorithm [27] construct a local minimal spanning tree with a bounded logical node degree and establish bidirectional link with a guaranteed connectivity. The topology control algorithm proposed in [28] is based on the Relative Neighborhood Graph (RNG) where a logical connection between two nodes is established, only if there are no other nodes within the overlapped broadcast circles of the two nodes. Although the resultant communication topologies satisfy the pre-defined design objectives, most algorithms proposed so far assume that all physical neighbors are found without consideration of collision.

1.3 Routing Protocols in WSNs

Unlike wired network, network designers of wireless networks need to consider issues including such as interference, unstable link state and limited energy. From literatures on routing protocols for wireless network, ***ad-hoc*** routing has been widely researched [29–32]. In ad-hoc routing, two end nodes ($i.e.$, source and destination nodes) communicate directly or via relaying nodes without a centralized control.

There are sub-operation modes of ad-hoc routing: proactive, interactive, hybrid of proactive and interactive and hierarchical routing modes. Proactive ad-hoc routing maintains a list of destination and corresponding routes, which is periodically updated. Interactive ad-hoc routing is initiated by a source (user) demand followed by flooding Route Request packet to find a route and inform the source. Hybrid ad-hoc routing is a combination of reactive and interactive routing modes. Hierarchical ad-hoc routing involves layered routing such that a node needs to send a packet to a node at higher layer for data communication.

Unfortunately, the ad-hoc routing protocols introduced above can not be used for WSNs because of the following reasons: highly limited energy resource (unreplaceable battery), small data size, simple (modest) radio and CPU capability and

low cost sensor nodes. For example, Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) protocol widely used in wireless network is not appropriate to be used in WSNs since its large energy consumption and low throughput for reliable data exchange [33]. Moreover, WSNs should accomplish a selective Quality of Service (QoS) such as small delay, energy efficiency or minimum interference, which requires a specialized routing protocol [34]. To meet this goal, a variety of WSN routing protocols have been proposed [16, 35–37].

Among routing protocols for WSNs introduced so far, cluster based routing has been researched in depth because of its scalability, energy efficiency and data processing productivity properties [15, 38–40]. In cluster based routing, a certain number of nodes are elected as cluster heads (CHs) which are in charge of relaying data from cluster member nodes to base station.

Low Energy Adaptive Clustering Hierarchy (LEACH) [15] has been a widely adopted clustering algorithm. In LEACH, a predefined (optimal) ratio of nodes are *probabilistically* elected as CH and each node becomes CH exactly once during one round. In one variant of LEACH, Kang *et al* in [41] proposed a centralized CH selection algorithm to optimize energy saving and evenly distributed CHs. [42] added node's remaining energy as a factor in LEACH CH election mechanism.

Unlike LEACH and its variants, in Energy-Efficient Cluster Formation (EECF) proposed in [43], CHs are elected with a three-way message exchange between sensors and their neighbors to consider nodes' residual energy and degree (the number of neighbors). Kumar *et al* in [44] designed an energy efficient clustering algorithm based on a set of heterogeneous nodes of varying initial energy amount. The cluster based routing algorithms introduced so far have a common communication method such that cluster member nodes can only send data to base station through CHs, not directly, and CHs are periodically rotated to prevent early energy depletion.

1.4 Graph Theory based Communication Topologies

Since Euler introduced the concept of *graph*s in 1736 [45, 46], graph theory has made significant contribution to the development of networking systems. Researchers have been applying graph theory to solve practical problems in the design of multiprocessor interconnection becomes constructing networking system where processors with local memory and connections between these elements are regarded as nodes and edges, respectively [47–49]. In analysis of social networks, Robinson et al utilize *Exponential Random Graph* to model people's interaction in

social networks [50, 51]. Barabàsi *et al* introduce *Scale-free Networks* to analyze the World Wide Web and reveal that a few highly connected pages are essentially holding the World Wide Web together [52, 53].

Among a wide range of researches, building a graph theory based communication topologies for computer networks has been spotlighted. Diagonal and toroidal mesh (D-Mesh and T-Mesh) networks are devised for parallel multicomputers communication with a small nodal degree [54, 55]. De Bruijn graph models applied for network systems interconnections have a near optimal diameter and provide simple routing algorithms [56–58]. For data center networking, the fat-tree has been used for routing between nodes (racks) where the switch bandwidth grows proportionally as moving up to the root [59].

1.5 Borel Cayley Graphs based Communication Topology and Routing

Borel Cayley Graphs (BCGs) are dense, regular, bidirectional and vertex transitive graph [60, 61]. As a communication topology, BCGs have useful network properties: a small diameter, a short average path length (topological properties), robust algebraic connectivity, fast data dissemination speed and vertex transitivity [62]. Specifically, the vertex transitivity property of BCGs enables optimal and distributed routing algorithm with a small routing table [60]. Two-phase routing protocol proposed in [63] provides suboptimal routing using class congruency, but requires smaller routing table sizes.

To reduce the impacts from network size inflexibility and improve fault tolerance of BCGs, the Cut-Through Rewiring (CTR) algorithm has been proposed and CTR showed that the resultant resized BCGs have a robust connectivity, efficient topological properties and outstanding algebraic connectivity [64]. For WSNs application, Borel Cayley Graph-Topology Control (BCG-TC) algorithm in [65] builds a communication topology by establishing selective logical connections between sensor nodes to overcome communication range constraints.

However, there are limitations in applying BCGs as a communication topology for network applications as following:

- **Generating parameter dependence**

  BCGs have a set of generating parameters, p and k, where p is a prime number and k is a factor of $p - 1$. These parameters determine a network size

($p \times k$) of BCGs. Due to a limited choice of p and k, however, selection of network size with BCG is constrained (size inflexibility). Moreover, network properties of BCGs such as topological/spectral properties and connectivity rely on the generating parameters and it is not possible to predict until measurement (simulation) result is obtained. We will discuss this issue in Section 3.3.

- **Need of fault-tolerant routing**

  BCGs have beneficial properties including vertex transitivity and GCR representation which are a main idea for VT routing protocol. However, if one node or link in BCG fails, the VT routing becomes infeasible. Although a fault-tolerant routing for the resized BCGs is presented in [66], propagating control packets for routing table update requires a large network overhead.

- **Restriction for network application**

  When applying BCGs to wireless network applications, communication range requirements are constrained. For instance, in a $100m \times 100m$ target area, the BCG-TC algorithm requires sensor nodes to have transmission range of $60 \sim 70m$ to generate a connected communication topology when $1,081$ nodes are randomly deployed. Such a large transmission range configuration results in significant interference in data exchange.

1.6 Our Contributions

In this dissertation, we focus on eliminating the generating parameter dependence of BCG while preserving its useful network properties by proposing *Expanded* Borel Cayley Graphs (*Ex*-BCGs) with applications to WSNs. Details of our contributions are as follows:

- ***Ex*-BCGs and Class-level Vertex Transitive Routing Protocol**

  We propose the *Expanded* BCGs (*Ex*-BCGs) to theoretically expand BCGs and eliminate the performance dependence on the generating parameters. The *Ex*-BCGs preserve outstanding topological/spectral properties of BCGs even after significant size expansions and enable nodes to be represented as a Generalized Chordal Ring (GCR) independent of the expansion level. Furthermore, *Ex*-BCGs have class-level vertex transitivity property which allows the networks viewed by vertices in the same class have an identical

structure. From these properties, we present the *Class-level Vertex Transitive* (CVT) routing protocol for *Ex*-BCGs. The CVT routing algorithm enables optimal and distributed routing between nodes by incorporating with the routing table which is generated by the GCR constant with small size complexity $O(dk)$ where d is a node degree. To simplify the routing table lookup process in the CVT routing protocol, we introduce the routing table folding and merging schemes.

- **Size flexible *Ex*-BCGs and Aggressive Multi-path Aware Routing**

  The Aggressive Multi-path Aware (AMA) routing protocol is designed to enable routing in the resized BCGs and *Ex*-BCGs formulated by the CTR algorithm. When BCGs and *Ex*-BCGs are resized, the proposed AMA routing protocol resolves the single shortest path and multi-paths depletion problems of the VT) and CVT routing protocols. The AMA routing protocol updates the routing table to reflect topology changes with a control packet propagation. Based on the updated routing table, the AMA routing protocol delivers data with the Multi-path Aware routing and the Random Direction routing schemes. The multi-path aware routing exploits all available multi-paths with prioritized selection from the updated routing table. The random direction routing supplements the multi-path aware routing by relaying packets to randomly selected neighbor when the multi-path aware routing fails.

- ***Ex*-BCGs based Topology Construction**

  The *Expanded* Borel Cayley Graphs Topology Construction (EBTC) algorithm formulates an efficient communication topology for WSNs. The EBTC is an integrated process of neighbor discovery and topology control used to produce a communication topology while collision avoidance and energy saving. The EBTC consists of two phases: in Phase I) a node collects node IDs of their logical neighbor candidates (Neighbor Discovery) and in Phase II) a node establishes bidirectional links with the most qualified logical neighbor candidate(s) and adjusts transmission power (Topology Control). Nodes under the EBTC algorithm save energy and avoid collision during topology construction by following a selective and deterministic node wake-up schedule which is designed based on cycle connection characteristics of *Ex*-BCG.

- ***Ex* Clustering for WSNs Routing**

For routing in the communication topology constructed by the EBTC, we provide *Ex* Clustering algorithm that forms clusters with a scoring equation to elect CHs. The *Ex* Clustering algorithm takes into consideration node's remaining energy, communication distance to two-hop logical neighbors and number of two-hop logical neighbors when selecting CHs. In *Ex* Clustering, nodes sequentially by node IDs execute clustering operation while utilizing a limited transmission power to avoid collision and save energy. Once a node finds out that it is the most qualified, it becomes a CH and its two-hop neighbors belong to that cluster as parents and members. Then CHs distribute its parent and member nodes a unique code and time slot which will be used for data communication. Furthermore, the *Ex* Clustering algorithm controls the maximum cluster size by selecting an appropriate number of generators and saves energy by allowing non-clustered nodes to enter sleep (radio-off) mode.

1.7 Organization of Thesis

The thesis is organized as follows. In Ch. 2, we provide an overview of BCGs and its applications: vertex-transitive routing protocol and the Cut-Through Rewiring (CTR) algorithm. In Ch. 3, we present *Ex*-BCGs with the mathematical proof of its network properties. Then we show how to resize *Ex*-BCGs with the CTR algorithm and evaluate its network performance. In Ch. 4, we present the Class-level Vertex Transitive (CVT) routing algorithm for *Ex*-BCGs with simulation result. Furthermore, we design the AMA routing protocol for the resized *Ex*-BCGs and validate its routing feasibility. In Ch. 5, we provide the EBTC algorithm to construct a communication topology and *Ex* Clustering to form clusters and enable routing for WSNs. Finally, conclusions and future work are presented in Ch. 6.

Chapter 2

Preliminaries

In this chapter, we overview Borel Cayley Graphs (BCGs) and its routing protocol: *vertex transitive* routing protocol. This chapter is in pursuit of providing a background of our work. Note that we use node and vertex interchangeably and $< x >_p$ represents a modulo operation of x with p throughout the paper.

2.1 Borel Cayley Graphs

BCGs are vertex-transitive, regular and undirected graph derived from the Borel Subgroup [61]. The Borel Subgroup is defined as:

Definition 1 (Borel Subgroup)**.**

$$V = \left\{ \begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix}, \; x = < a^t >_p, \; y \in \mathbf{Z}_p, \; t \in \mathbf{Z}_k \right\}, \tag{2.1}$$

where p is a prime number and k is the smallest positive integer satisfying $< a^k >_p = 1$.

From Definition 1, Borel subgroup has k classes indexed from 0 to $k-1$ and each of them contains p elements, where a class index of the element is determined by t (See Figure 2.1).

Based on the Borel subgroup, Borel Cayley Graphs are defined as follows:

Definition 2 (Borel Cayley Graph [61])**.** Let V and $\mathbb{G} \subseteq V \setminus \{I\}$ be a Borel subgroup and a generator set, respectively. **C** is a BCG consisting of vertices represented by 2×2 matrices $\in V$ and a directed edge satisfying $u = v * g$, where $u \neq$

$$\begin{array}{|cccc|}
\hline
\begin{pmatrix} < a^0 >_p & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} < a^0 >_p & 1 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^0 >_p & p-2 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} < a^0 >_p & p-1 \\ 0 & 1 \end{pmatrix} \\
\begin{pmatrix} < a^1 >_p & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} < a^1 >_p & 1 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^1 >_p & p-2 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} < a^1 >_p & p-1 \\ 0 & 1 \end{pmatrix} \\
\vdots & \vdots & \ddots & \vdots \quad \vdots \\
\begin{pmatrix} < a^{k-1} >_p & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} < a^{k-1} >_p & 1 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^{k-1} >_p & p-2 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} < a^{k-1} >_p & p-1 \\ 0 & 1 \end{pmatrix} \\
\hline
\end{array}$$

Figure 2.1: Borel Subgroup with k classes and p elements in each class.

$v \in V, g \in \mathbb{G}$ and $*$ is a modulo-p multiplication chosen as the group operation as described as follows: when $v = \begin{pmatrix} < a^{t_v} >_p & y_v \\ 0 & 1 \end{pmatrix}$ and $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix}$, $v * g = \begin{pmatrix} < a^{t_v + t_g} >_p & < y_g < a^{t_v} >_p + y_v >_p \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix} = u$. Note that the generator set $\mathbb{G}$ excludes the identity element I to avoid self-loops.

Regarding the connection rule, BCGs are undirected graphs and have a generator dependent graph structure. Due to the fact that the set of generators $\mathbb{G}$ is closed under inverse, an inverse of a generator g creating a directed edge is also included in $\mathbb{G}$ ($i.e.$, $g^{-1} \in \mathbb{G}$). Thus BCGs are undirected graphs satisfying $v = u * g$ and $u = v * g^{-1}$ (See Figure 2.2).

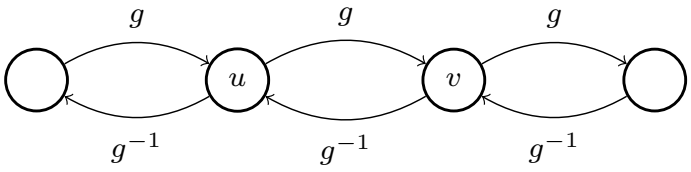


Figure 2.2: BCG connection rule

Generator selection always result in a pair of generator g and its inverse g^{-1}. Since the neighbors of node v in BCGs are obtained by $v * g$ and $v * g^{-1}$, where $v * g \neq v * g^{-1}$, a BCG node has always a multiple of two degree. In a network structure view, BCGs topological properties and connectivity are largely influenced by the choice of generators. [62] provides rules for selecting an appropriate generator g depending on the BCG generating parameters p and k. In the following subsections, we discuss the properties which are beneficial to network applications, especially in the development of network routing algorithms.

2.1.1 GCR Representation

As discussed in the previous section, BCGs are defined over a group of matrices. For simple ordering of vertices and concise description of connections, [67] proved that BCGs can be represented by Generalized Chordal Rings (GCR) which converted vertices represented by matrix into integer node IDs and provided a systematic description of connections as follows:

Proposition 1. All Borel Cayley Graphs have GCR representation.

The proof is included in [67] and not repeated here. The GCR representation of BCGs is based on selecting a transform element **T** and class representing elements a_i, $i = 0, \cdots, k-1$ for k classes.

Consider, a BCG with parameters $p = 7$, $k = 3$, $a = 2$ and generator set $G = \{A, B, A^{-1}, B^{-1}\}$, where $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$. Selecting a transform element $\mathbf{T} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ ($\mathbf{T}^7 = \mathbf{I}$) and class representing elements $a_i = \begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix}$ for class i, the conversion to an integer node ID as follows:

$$\begin{pmatrix} a^t & y \\ 0 & 1 \end{pmatrix} := yk + t \tag{2.2}$$

where the vertex ID ranges from 0 to $pk-1$ and the number of vertices is $N = p \times k$. As a result, vertices in this example are labeled with integers $V = \{0, 1, \cdots 20\}$.

For a systematic description of connection, GCR constants define integer ID difference between intermediately connected vertices for each class as:

For any $j \in V$, if $< j >_k = i$, j is connected to $< j + \alpha_i >_N, < j + \alpha_i^{-1} >_N$, $< j + \beta_i >_N, < j + \beta_i^{-1} >_N$,

where N is a network size ($N = p \times k$). The GCR constants $\alpha_i, \alpha_i^{-1}, \beta_i, \beta_i^{-1}$ for the example with $p = 7$, $k = 3$, $a = 2$ and generators $A = \begin{pmatrix} 2^0 & 1 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}$ are listed in Table 2.1.

The resultant BCG represented by integer ID and connection from GCR constants is illustrated in Figure 2.3.

i	α	α^{-1}	β	β^{-1}
0	3	-3	4	-10
1	6	-6	7	-4
2	-9	9	10	-7

Table 2.1: GCR constants for a BCG with parameters $p = 7$, $k = 3$ and $a = 2$

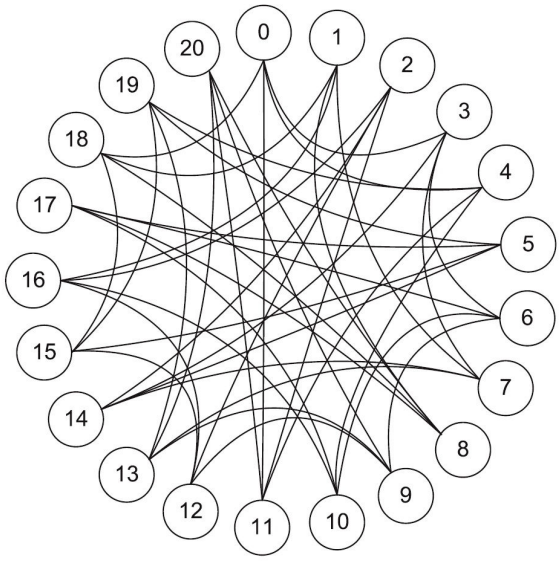


Figure 2.3: GCR Representation of BCG example

2.1.2 Vertex Transitivity

A graph **C** is *vertex transitive* if its automorphism group acts transitively on **V**(**C**) [68]. In other words, when graphs viewed by every vertex look identical, that graph is vertex transitive. In the previous section, we mentioned that BCGs are vertex transitive. We restate the vertex transitivity of BCGs as following [60]:

Corollary 1. Let **C**=(**V**,**G**) be a Borel Cayley Graph. Assume $u, v, w \in \mathbf{V}$. If u and v are connected through a certain sequence of generators, then w and $w * u^{-1} * v$ are connected through the same sequence of generators.

The proof is included in [60] and not repeated here. Corollary 1 implies that the vertex transitivity of BCGs enables finding a path between two arbitrary vertices by finding a path between corresponding vertices. In other words, a path

between u and v is obtained by finding the path between the identity element $\mathbf{I}$ and $u^{-1} * v$. Furthermore, the GCR representation of BCGs simplified path finding by converting matrix representation of vertex to integer representation. We will discribe a routing algorithm based on the vertex transitivity later.

2.1.3 Class Congruency

Class congruency, an important property of BCGs, means that every node in a class has the same class-connectivity [63]. We restate that *class congruence* property as follows:

Proposition 2. *Class congruency*. The GCR constants of BCGs with $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and class representing element $\begin{pmatrix} a^i & 0 \\ 0 & 1 \end{pmatrix}$ are congruent modulo k. Parameter k is also the number of classes in the GCR. Specifically, with generators

$$\begin{aligned} A &= \begin{pmatrix} a^{t_1} & y_1 \\ 0 & 1 \end{pmatrix}, A^{-1} = \begin{pmatrix} a^{k-t_1} & -a^{k-t_1} y_1 \\ 0 & 1 \end{pmatrix} \\ B &= \begin{pmatrix} a^{t_2} & y_2 \\ 0 & 1 \end{pmatrix}, B^{-1} = \begin{pmatrix} a^{k-t_2} & -a^{k-t_2} y_2 \\ 0 & 1 \end{pmatrix} \end{aligned} \tag{2.3}$$

the class constants are as follows:

$$\begin{array}{rcll} c_A & = & t_1 & \forall\, i \\ c_{A^{-1}} & = & k - t_1 & \forall\, i \\ c_B & = & t_2 & \forall\, i \\ c_{B^{-1}} & = & k - t_2 & \forall\, i \end{array} \tag{2.4}$$

The proof is included in [63] and therefore not repeated here. The class constants mean that a class index of an arbitrary vertex's neighbor in BCGs is determined by the generators and the vertex's class index. For example, with $p = 7$, $k = 3$, $a = 2$ and $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$, a neighbor of vertex $u = \begin{pmatrix} 2^t & y \\ 0 & 1 \end{pmatrix}$ in B direction belongs to the $t+1$ class independent of the y. For network application, the class congruence property plays a key role in *two-phase* routing protocol, which will be discussed later.

	A	B	A^{-1}	B^{-1}
1	0	0	1	0
2	0	0	0	1
3	1	0	0	0
4	0	1	0	0
5	1	1	1	0
6	1	0	0	0
7	1	0	0	0
8	0	0	1	0
9	1	0	0	1
10	0	1	0	0

	A	B	A^{-1}	B^{-1}
11	0	0	0	1
12	0	0	1	1
13	1	1	0	1
14	1	0	0	0
15	0	0	1	0
16	0	1	1	1
17	1	1	1	0
18	0	0	1	0
19	0	1	0	0
20	0	0	0	1

Figure 2.4: Routing table of BCG example

2.2 Vertex-Transitive Routing Protocol

As mentioned in Section 2.1.2, BCGs are vertex transitive. So finding a path between two arbitrary vertices (source and destination) is equivalent to finding a path between the two vertices, where one is a class representing node and another is a corresponding node from the class representing node's view to the destination.

Since a matrix domain representation makes finding a path a non-trivial work, the vertex-transitive routing protocol based on the GCR representation of BCGs is used. In GCR format, finding the path between source u and destination v is equivalent to finding the path between the u's class representing node ($i.e.$, $< u >_k$) and node v'. In other words, once we find the paths between 0 and all vertices, routing is simplified to mapping source and destination to the equivalent destination v'. The vertex-transitive routing protocol makes it possible to use the same routing table in all vertices of a BCG. The routing algorithm is summarized in Table 2.2. We observed that the routing algorithm allows for multiple, shortest paths and results in a routing table of size $O(N)$ where N is the network size. The routing algorithm is iterative and its time complexity for packet transmission between source and destination is $O(D)$, where D is a diameter.

As an example of routing, assume that we need to send a message from source 0 to destination 16 in a BCG of parameters $p = 7$, $k = 3$, $a = 2$ and generators $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$. We identify $j' = 16$ from Step 1 of the routing algorithm from Table 2.2. From the routing table in Figure 2.4, at row 16 we found three choices: $\mathbf{B}$, $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$. When we select a link $\mathbf{B}$, the new source i'

For a degree-4 Borel Cayley Graph in GCR representations with

$\mathbf{T}=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$ and $a_i=\begin{pmatrix}a^i & 0\\ 0 & 1\end{pmatrix}$,

we have k classes, where $a^k=1\ (mod\ p)$.

Assume the generators to be $\mathbf{A}$, $\mathbf{B}$, $\mathbf{A}^{-1}$, $\mathbf{B}^{-1}$,

where $\mathbf{A}=\begin{pmatrix}a^{t_1} & y_1\\ 0 & 1\end{pmatrix}$ and $\mathbf{B}=\begin{pmatrix}a^{t_2} & y_2\\ 0 & 1\end{pmatrix}$.

Given source $i=m_1q+c_1$ and destination $j=m_2q+c_2$.

While $(i\neq j)$
Step 1 : Identify a new destination, $j'=<a^{q-c_1}(m_2-m_1)>_p q+<c_2-c_1>_q$
Step 2 : From row j' of routing table, determine which link to take.
Step 3 : Identify new source $i'=mq+c$ and
$m=y_1$, $c=t_1$, if link $\mathbf{A}$ is chosen
$m=y_2$, $c=t_2$, if link $\mathbf{B}$ is chosen
$m=q-<a^{q-t_1}>_p$, $c=q-t_1$, if $\mathbf{A}^{-1}$ is chosen
$m=q-<a^{q-t_2}>_p$, $c=q-t_2$, if $\mathbf{B}^{-1}$ is chosen
Step 4 : $i=i'$ and $j=j'$

Table 2.2: Iterative Routing for Borel Cayley Graphs

is 4. Since $i\neq j$, iteration with $i=4$ and $j=16$ results in $j'=6$. At row 6 of the routing table, $\mathbf{A}$ is selected, which determines the new source $i'=3$. Since $i\neq j$, one more iteration is needed. We find $i'=3$ and $j'=3$ by selecting $\mathbf{A}$. Finally, the path between source 0 and destination 16 is to be $\mathbf{BAA}$. We summarize all that paths from source 0 and destination 16 in Table 2.3.

We observed that the vertex-transitive routing protocol utilizes vertex transitivity and GCR representations of BCGs. The vertex transitive routing protocol results in a routing table that guarantees the shortest path(s). The routing table size is $O(Nd)$, where N is a network size and d is the node degree, while the routing algorithm has a time complexity $O(D)$, where D is a diameter.

path 1 :	0	$\xrightarrow{\mathbf{B}}$	4	$\xrightarrow{\mathbf{A}}$	10	$\xrightarrow{\mathbf{A}}$	16
path 2 :	0	$\xrightarrow{\mathbf{A}^{-1}}$	18	$\xrightarrow{\mathbf{B}}$	1	$\xrightarrow{\mathbf{A}^{-1}}$	16
path 3 :	0	$\xrightarrow{\mathbf{B}^{-1}}$	11	$\xrightarrow{\mathbf{A}}$	2	$\xrightarrow{\mathbf{B}^{-1}}$	16

Table 2.3: Multiple paths from source 0 to destination 16

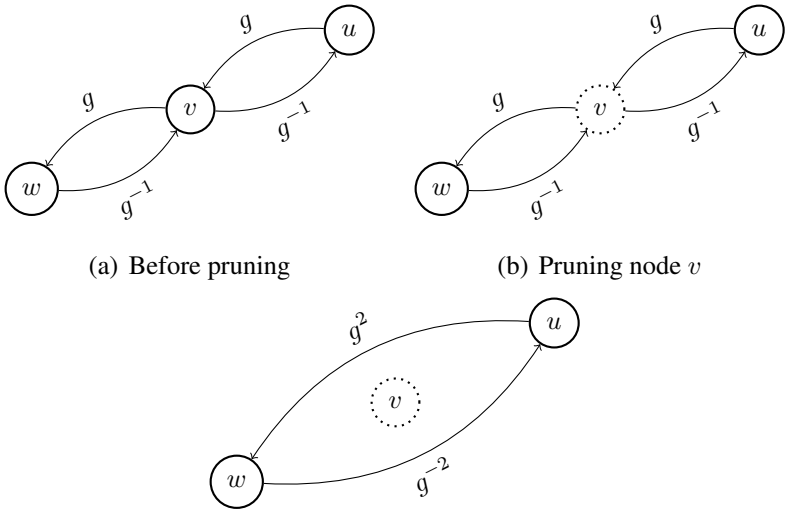
2.3 Cut-Through Rewiring Algorithm

As discussed in the previous section, BCG's size is determined by parameters p and k. Since p is a prime number and k a factor of $p-1$, BCGs have limited sizes. To alleviate this size inflexibility, [64] proposes a *Cut-Through Rewiring* (CTR) algorithm to build arbitrarily sized networks while conserving a constant nodal degree. Once node(s) of BCGs are pruned, the CTR algorithm connects neighboring nodes of the pruned nodes following the BCG connection rule.

Figure 2.5 illustrates the CTR algorithm. Once v is pruned, where $u * g = v$ and $v * g = w$ (See Figure 2.5(a) and Figure 2.5(b)), the CTR algorithm connects u to w and u by $u * g^2 = w$ and $w * g^{-2} = u$ in Figure 2.5(c). The resized BCGs have good topological properties, robust connectivity and fast information distribution performance while improving the BCG size flexibility [64].

2.4 Discussion

In this chapter, we reviewed BCGs and its applications. BCGs have beneficial network properties: outstanding topological/spectral properties, GCR representation, class congruency and vertex transitivity. Based on those properties, vertex-transitive routing protocol provides optimal and distributed routing path. However, BCGs have poor scalability by severe generating parameters p, k dependence resulting in inconsistent network performance. Moreover, selection pool of the parameters p, k has small size. Thus, its network size inflexibility limits applicability of BCGs in real network applications. In the next chapters, we propose how to eliminate scalability issue of BCGs with consistent network performance and widen the network size selection pool.



(a) Before pruning

(b) Pruning node v

(c) Applying CTR between nodes u and w

Figure 2.5: Illustration of Cut-Through Rewiring in BCG

Chapter 3

Expanded Borel Cayley Graphs

In this chapter, we introduce *Expanded* Borel Cayley Graphs (*Ex*-BCGs) how to resolve scalability limitation of BCGs while preserving BCGs' useful network properties. Then we provide how to resize *Ex*-BCGs to an arbitrarily sized network with CTR algorithm.

3.1 Extended Borel Subgroup

As discussed in Chapter 2, the size of BCGs with the parameters p, k is $p \times k$ nodes of IDs ranging from 0 to $p \times k - 1$. In order to systematically expand the node ID space of BCGs, we define the Expanded Borel subgroup to increase the node ID space.

$$\begin{array}{|ccccccc|}
\hline
\begin{pmatrix} < a^0 >_p & 0 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^0 >_p & p-1 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} \boldsymbol{< a^0 >_p} & \boldsymbol{p} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} & \cdots & \begin{pmatrix} \boldsymbol{< a^0 >_p} & \boldsymbol{np-1} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} \\
\begin{pmatrix} < a^1 >_p & 0 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^1 >_p & p-1 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} \boldsymbol{< a^1 >_p} & \boldsymbol{p} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} & \cdots & \begin{pmatrix} \boldsymbol{< a^1 >_p} & \boldsymbol{np-1} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\begin{pmatrix} < a^{k-1} >_p & 0 \\ 0 & 1 \end{pmatrix} & \cdots & \begin{pmatrix} < a^{k-1} >_p & p-1 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} \boldsymbol{< a^{k-1} >_p} & \boldsymbol{p} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} & \cdots & \begin{pmatrix} \boldsymbol{< a^{k-1} >_p} & \boldsymbol{np-1} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix} \\
\hline
\end{array}$$

Figure 3.1: Node ID space of Expanded Borel subgroup (Bold is the expanded part from original BCG)

Definition 3 (Expanded Borel subgroup)**.** Let V' be an Expanded Borel subgroup of $\mathbf{GL}_2(\mathbf{Z}_p)$ with parameter $a \in \mathbf{Z}_p \setminus \{0, 1\}$, then

$$V' = \left\{ \begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix} : x = < a^t >_p, y \in \mathbf{Z_{n \times p}}, t \in \mathbf{Z}_k \right\} \tag{3.1}$$

where p is a prime number, k is the smallest positive integer satisfying $< a^k >_p = 1$, and n is an *expansion factor*.

The expansion factor n is an integer so that y of the subgroup elements of the Expanded Borel subgroup can be expanded from 0 to $np - 1$ as illustrated in Figure 3.1. Hence, the Expanded Borel subgroup contains $n \times p \times k$ node IDs (when $n = 1$, V' is an original Borel subgroup).

3.2 Expanded Borel Cayley Graphs

Based on the Expanded Borel subgroup, we define *Expanded* Borel Cayley Graphs (*Ex*-BCGs) as follows:

Definition 4. Let V' and $\mathbb{G}' \subseteq V' \setminus \{I\}$ be an Expanded Borel subgroup and a generator set, respectively. $\mathbf{C}'$ is an *Ex*-BCG consisting of vertices represented by 2×2 matrices $\in V'$ and a directed edge satisfying $u = v * g$, when $u = \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix}$, $v = \begin{pmatrix} < a^{t_v} >_p & y_v \\ 0 & 1 \end{pmatrix}$, $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix} \in \mathbb{G}'$, where $u \neq v \in V'$ and $0 \leq t_u, t_v, t_g \leq k-1$, $v*g = \begin{pmatrix} < a^{t_v + t_g} >_p & < y_g < a^{t_v} >_p + y_v >_{np} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix} = u$.

For an undirected connection, an inverse generator g^{-1} is defined as:

Definition 5. With a generator $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix} \in \mathbb{G}'$, g^{-1} is defined as $g^{\lambda - 1}$, where $g^\lambda = \underbrace{g * \cdots * g}_{\lambda} = I$.

From Definition 5, we derive the following proposition:

Proposition 3. For any generator $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix} \in \mathbb{G}'$, when $g^{-1} = g^{\alpha} \in V'$, $\alpha \in \{m-1, 2m-1, \cdots, nm-1\}$, where n is an expansion factor and m is such that $g^m = I$ in BCGs.

Proof. Any generator $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix} \in V$, where $g^m = I$ in BCGs, satisfies

$$< a^{mt_g} >_p = < a^{lk} >_p = 1 \tag{3.2}$$

where l is a positive integer. If we define α_ϕ as

$$\alpha_\phi = < a^{(\phi-1)t_g} >_p, \tag{3.3}$$

a sequence of class indices in connections rooted at node $i = \begin{pmatrix} < a^i >_p & y_i \\ 0 & 1 \end{pmatrix}$ with g ($i.e.$, $i * g, \cdots, i * g^{nm}$) is cycled due to $mt_g = lk$ from Eq (3.2). This is illustrated in Table 3.1. From $g^m = I$ in BCG, we obtain

$$\sum_{\phi=1}^{m} \alpha_\phi = \beta p \tag{3.4}$$

where β is a positive integer. Since

$$< a^{\gamma t_g} >_p = < a^{(m+\gamma)t_g} >_p \tag{3.5}$$

where $0 < \gamma < m$,

$$\begin{array}{ccccccc} \alpha_1 & = & \alpha_{m+1} & = & \cdots & = & \alpha_{(n-1)m+1} \\ \alpha_2 & = & \alpha_{m+2} & = & \cdots & = & \alpha_{(n-1)m+2} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \alpha_m & = & \alpha_{2m} & = & \cdots & = & \alpha_{nm} \end{array} \tag{3.6}$$

Then,

$$\begin{array}{rcl} \alpha_1 + \alpha_2 + \cdots + \alpha_m & = & \beta p \\ \alpha_m + \alpha_{m+1} + \cdots + \alpha_{2m} & = & \beta p \\ \vdots & & \\ \alpha_{(n-1)m} + \alpha_{(n-1)m+1} + \cdots + \alpha_{nm} & = & \beta p \end{array} \tag{3.7}$$

$$
\begin{array}{c}
i \rightarrow < i + t_g >_k \rightarrow < i + 2t_g >_k \rightarrow \cdots \rightarrow < i + mt_g >_k = i \rightarrow \\
< i + (m+1)t_g >_k = < i + t_g >_k \rightarrow \cdots \rightarrow < i + 2mt_g >_k = i \rightarrow \\
\vdots \\
< i + ((n-1)m+1)t_g >_k = < i + t_g >_k \rightarrow \cdots \rightarrow < i + nmt_g >_k = i
\end{array}
$$

Table 3.1: Class sequence of cycle in *Ex*-BCGs with $g^{nm} = I$

$$
\begin{array}{c}
g^m = \begin{pmatrix} < a^{mt_g} >_p & < y_g(\alpha_1 + \alpha_2 + \cdots + \alpha_m) >_{np} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & < y_g \beta p >_{np} \\ 0 & 1 \end{pmatrix} \\
g^{2m} = \begin{pmatrix} < a^{2mt_g} >_p & < y_g(\alpha_1 + \alpha_2 + \cdots + \alpha_{2m}) >_{np} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & < y_g 2\beta p >_{np} \\ 0 & 1 \end{pmatrix} \\
\vdots \\
g^{nm} = \begin{pmatrix} < a^{nmt_g} >_p & < y_g(\alpha_1 + \alpha_2 + \cdots + \alpha_{nm}) >_{np} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & < y_g n\beta p >_{np} \\ 0 & 1 \end{pmatrix}
\end{array}
$$

Table 3.2: Power sequence of generator g in the *Ex*-BCGs

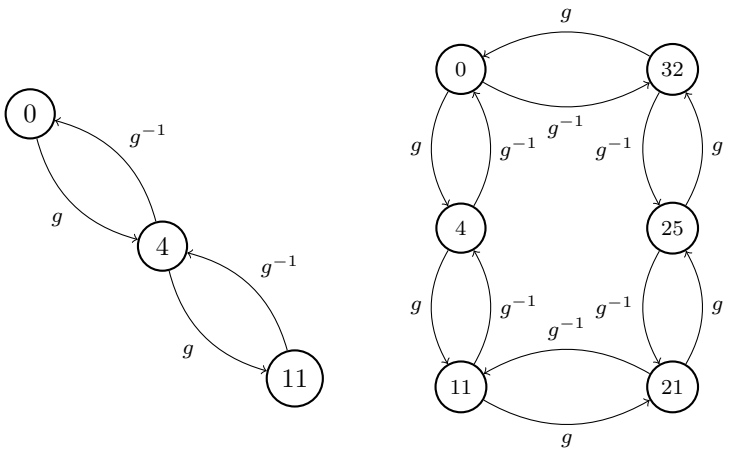
In *Ex*-BCGs with expansion factor n, any generator $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix}$ selected from $\mathbb{G}'$ satisfies Eq (3.2). Using Eq (3.3), g^m can be expressed as

$$
\begin{pmatrix} < a^{mt_g} >_p & < y_g(\alpha_1 + \cdots + \alpha_m) >_{np} \\ 0 & 1 \end{pmatrix}. \tag{3.8}
$$

Thus, $g^{nm} = \begin{pmatrix} 1 & < y_g n\beta p >_{np} \\ 0 & 1 \end{pmatrix} = I$ independent of y_g and β (See Table 3.2). Hence, any generator $g \in \mathbb{G}'$ has inverse g^{-1} with maximum order $nm - 1$. □

From Def. 4 and Def. 5, BCGs can be theoretically expanded with any expansion factor n.

In Fig. 3.2, we illustrate an example of a sequence of connections of *Ex*-BCG from BCG. Note that each integer number of circle represents a node ID obtained from Eq (2.2) and Eq (3.17). Fig. 3.2(a) shows a sequence of connections rooted at node 0 in BCG with generating parameters $p = 7, k = 3, a = 2$ and generator $g =$



(a) A sequence of connections rooted at node 0 with $g^3 = I$

(b) A sequence of extended connections rooted at node 0 with $g^6 = I$

Figure 3.2: An example of connection sequences in BCG and *Ex*-BCG with parameters $p = 7$, $k = 3$, $a = 2$ and $n = 2$.

$\begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}$. After expansion with $n = 2$, the sequence of connection is extended as shown in Fig. 3.2(b).

Based on the discussion so far, we state and prove several useful properties of *Ex*-BCGs in the following sections.

3.3 Network Properties and Evaluation of *Ex*-BCGs

3.3.1 Class-level Vertex Transitivity

Unlike BCGs, *Ex*-BCGs lose the vertex transitivity property after expansion. However, *Ex*-BCGs preservevertex transitivity after expansion at the class-level.

Proposition 4. *Ex*-BCGs are *class-level vertex transitive*.

Proof. To prove the proposition, we use Theorem 3.1.2 of [68]. Let $\alpha = \begin{pmatrix} < a^t >_p & y_\alpha \\ 0 & 1 \end{pmatrix}$, $\beta = \begin{pmatrix} < a^t >_p & y_\beta \\ 0 & 1 \end{pmatrix}$ and $g = \begin{pmatrix} < a^g >_p & y_g \\ 0 & 1 \end{pmatrix}$. Note that α and β are the vertices in the same class t.

For each $g \in \mathbb{G}'$, where $\mathbb{G}'$ is a generator set of the *Ex*-BCGs, the mapping

$$\epsilon_g : \alpha \mapsto \alpha g \tag{3.9}$$

is a permutation of the elements of $\mathbb{G}'$. We prove the class level vertex transitivity by showing $(\beta * g) * (\alpha * g)^{-1} = \beta * \alpha^{-1}$ [68]. If $\alpha^m = I$, $\alpha^{-1} = \alpha^{m-1}$ is

$$\begin{pmatrix} < a^{(m-1)t} >_p & < y_\alpha(a^{(m-2)t} + \cdots + 1) >_{np} \\ 0 & 1 \end{pmatrix} \tag{3.10}$$

and $\beta * \alpha^{-1}$ is

$$\begin{pmatrix} < a^{mt} >_p & < y_\alpha(\gamma p - 1) + y_\beta >_{np} \\ 0 & 1 \end{pmatrix}, \tag{3.11}$$

where $\gamma p - 1 = a^{(m-1)t} + \cdots + a^t$. Since $< y_\alpha \gamma p >_{np} = 0$ from Table 3.2, Eq. (3.11) becomes

$$\begin{pmatrix} < a^{mt} >_p & < -y_\alpha + y_\beta >_{np} \\ 0 & 1 \end{pmatrix}. \tag{3.12}$$

From Def. 5, if $(\alpha * g)^l = I$ and l is a positive integer, $(\alpha * g)^{-1} = (\alpha * g)^{l-1}$. Thus $(\beta * g) * (\alpha * g)^{-1}$ becomes

$$\begin{pmatrix} < a^{l(t+g)} >_p & < (y_g a^t + y_\alpha)(\rho p - 1) + y_g a^t + y_\beta >_{np} \\ 0 & 1 \end{pmatrix}, \tag{3.13}$$

where $\rho p - 1 = a^{(l-1)(t+g)} + \cdots + a^{(t+g)}$. Since $< y_g a^t \rho p + y_\alpha \rho p >_{np} = 0$, Eq (3.13) becomes

$$\begin{pmatrix} < a^{l(t+g)} >_p & < -y_\alpha + y_\beta >_{np} \\ 0 & 1 \end{pmatrix}. \tag{3.14}$$

Since $< a^{l(t+g)} >_p = < a^{mt} >_p = 1$, we get

$$(\beta * g) * (\alpha * g)^{-1} = \beta * \alpha^{-1}. \tag{3.15}$$

This is an automorphism of *Ex*-BCGs such that $\alpha * g$ and $\beta * g$ are adjacent, if and only if α and β are adjacent, where α and β are the vertices of the ***same class***. The permutations ϵ_g form a subgroup of the automorphism group of the *Ex*-BCGs isomorphic to $\mathbb{G}'$. This subgroup acts transitively on the vertices, since for any two vertices a and b in the same class, the automorphism $\epsilon_{a^{-1} * b}$ maps a to b. □

3.3.2 GCR representation

In this section, we show that *Ex*-BCGs can be represented by Generalized Chordal Rings (GCRs).

Proposition 5. *Ex*-BCGs can be represented as GCR with a transform element $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, where $T^{np} = I$.

Proof. To prove the proposition, we use Proposition 1 and Def. 3 of [67]. Let $\alpha_i = \begin{pmatrix} < a^i >_p & 0 \\ 0 & 1 \end{pmatrix}$ be a representing element of the i^{th} class in the Expanded Borel subgroup. Since $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ satisfies $T^{np} = I$ independent of n, the elements of V' can be partitioned into k classes: $V'(\alpha_0), V'(\alpha_1), \cdots, V'(\alpha_{k-1})$ as follows:

$$\begin{array}{rcl} V'(\alpha_0) & = & \{\alpha_0, T * \alpha_0, \cdots, T^{np-1} * \alpha_0\} \\ V'(\alpha_1) & = & \{\alpha_1, T * \alpha_1, \cdots, T^{np-1} * \alpha_1\} \\ & \vdots & \\ V'(\alpha_{k-1}) & = & \{\alpha_{k-1}, T * \alpha_{k-1}, \cdots, T^{np-1} * \alpha_{k-1}\}. \end{array} \tag{3.16}$$

In other words, for all $\alpha \in V'$, there exists $\alpha = T^s * a_i$ such that $s \in \{0, 1, \cdots, np-1\}$ and $i \in \{0, 1, \cdots, k-1\}$. Hence, each matrix element of *Ex*-BCG is converted to an integer node ID as follows:

$$\begin{pmatrix} < a^i >_p & s \\ 0 & 1 \end{pmatrix} := sk + i. \tag{3.17}$$

For any two distinct $a, b \in V'$ where $a = T^s * \alpha_i$ and $b = T^{s'} * \alpha_{i'}$ for some $s, s' \in \{0, 1, \cdots, np-1\}$ and $i, i' \in \{0, 1, \cdots, k-1\}$. Thus, $a \rightarrow i + sk$, $b \rightarrow i' + s'k$. Then $a \neq b \Rightarrow s \neq s'$ or $i \neq i'$ or both. Finally, $i + sk \neq i' + s'k$. So, the matrix element of *Ex*-BCG is mapped to a unique integer ID (one-to-one).

We restate the definition of GCR [67] as follows:

Definition 6. A graph G has a GCR representation if the vertices of G can be labeled with integers $mod\ N$ (N is the number of vertices), and there is a divisor q of N such that vertex i is connected to vertex j if and only if vertex $< i + q >_N$ is connected to vertex $< j + q >_N$.

We assume node u represents a matrix element or an integer node ID of *Ex*-BCG, interchangeably (*i.e.*, $u = \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix} \Leftrightarrow u = y_u k + t_u$). If $u = \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix}$ is connected to $v = \begin{pmatrix} < a^{t_v} >_p & y_v \\ 0 & 1 \end{pmatrix}$ with generator $g = \begin{pmatrix} < a^{t_g} >_p & y_g \\ 0 & 1 \end{pmatrix}$, then $u*g = \begin{pmatrix} < a^{t_v} >_p & y_v \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a^{<t_u+t_g>_k} & < y_g < a^{t_u} >_p + y_u >_{np} \\ 0 & 1 \end{pmatrix}$ is satisfied.

Thus,

$$< t_u + t_g >_k = t_v, \quad < y_g < a^{t_u} >_p + y_u >_{np} = y_v. \tag{3.18}$$

So, node $u + \sigma k$ with $\sigma \in \{1, 2, \cdots, np-1\}$ is defined as

$$T^\sigma * u = \begin{pmatrix} 1 & \sigma \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} < a^{t_u} >_p & y_u \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} < a^{t_u} >_p & < y_u + \sigma >_{np} \\ 0 & 1 \end{pmatrix} \tag{3.19}$$

and $T^\sigma * u * g$ is

$$T^\sigma * u * g = \begin{pmatrix} < a^{<t_u+t_g>_k} >_p & < y_g < a^{t_u} >_p + < y_u + \sigma >_{np} >_{np} \\ 0 & 1 \end{pmatrix}. \tag{3.20}$$

Likewise, node $v + \sigma k$ is defined as

$$T^\sigma * v = \begin{pmatrix} 1 & \sigma \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} < a^{t_v} >_p & y_v \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} < a^{t_v} >_p & < y_v + \sigma >_{np} \\ 0 & 1 \end{pmatrix}. \tag{3.21}$$

From Eq (3.18), Eq (3.20) and Eq (3.21),

$$\begin{aligned} &<< y_g < a^{t_u} >_p + y_u >_{np} + \sigma >_{np} \\ =&< y_g < a^{t_u} >_p + < y_u + \sigma >_{np} >_{np} \end{aligned} \tag{3.22}$$

As a result, in an *Ex*-BCG, if u is connected to v, then $u + \sigma k$ is also connected to $v + \sigma k$. Therefore, *Ex*-BCGs can be expressed in a GCR representation with a transform element $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. □

As an example of a GCR representation for an element of a *Ex*-BCG, with generating parameters $p = 7, k = 3, a = 2$ and expansion factor $n = 2$, the matrix node ID $\begin{pmatrix} 2 & 7 \\ 0 & 1 \end{pmatrix}$ is uniquely represented by the integer node ID: $7 \times 3 + 1 = 22$.

3.3.3 Network Properties Evaluation

In this section, we investigate topological/spectral properties and consensus performance of *Ex*-BCGs (of size N_{Ex-BCG}). Then we compare the result with that

of BCGs (of size N_{BCG}).

1. Comparison with the simulation results of *all generator sets* of *Ex*-BCGs and BCGs of similar sizes.

2. Comparison of simulation results for *the same subsets of generator sets* of *Ex*-BCGs and BCGs of similar sizes.

The separated analysis is due to the nature of *Ex*-BCGs and BCGs which yields different number of possible generator sets. The number of possible generator sets is determined by parameter k and each set of generator produces distinct network performance. Because the results from different generator sets vary, we analyze the result of *Ex*-BCGs and BCGs with 1) all possible generator sets and 2) the same subset of the generator sets.

Benchmark network topologies

In [64], comparison between BCGs, toroidal and diagonal mesh networks and small-world networks shows that BCGs have the smallest diameter, shortest average path length, largest algebraic connectivity and fastest convergence speed of the average consensus protocol. By comparing *Ex*-BCGs and BCGs of similar sizes, we can make a relative comparison to toroidal and diagonal mesh networks and small-world networks of similar sizes. Thus, we select BCGs of sizes close to *Ex*-BCGs sizes as benchmark networks with the constraint of $a \in \{2, 3, 5, 6, 7\}$.

Metrics

For evaluation and comparison study, we use the following metrics throughout this evaluation:

- **Topological properties: diameter and average path length**

  Topological properties are significant network characteristics and consist of diameter and average path length [69]. Diameter is the largest hop count between an arbitrary pair of nodes. Average path length is the average of the hop counts across all possible pair of nodes. The topological properties are an effective metric in evaluating the communication efficiency of multi-agent systems [70, 71]. We measure the topological properties of *Ex*-BCGs, and compare them with that of BCGs of similar sizes.

- **Convergence speed of average consensus protocol**

  We quantify the convergence speed of *Ex*-BCGs and BCGs by counting the number of steps for all nodes to reach the average consensus value with a predefined precision. Each node is initialized with a random state value $[-5, 5]$. Since we consider non-weighted and undirected networks, nodes exchange their state value according to [72]

$$r_v(t+1) = r_v(t) + \frac{1}{w} \sum_{u \in N(v)} (r_u(t) - r_v(t)), \tag{3.23}$$

  where $w = 2deg_{max}(G) + 1$ for a guaranteed asymptotic convergence, and $deg_{max}(G)$ denotes the maximum degree of graph G. When the state values of all nodes reach the average of the initial state values within a precision of 10^{-4}, we declare a consensus to be achieved.

- **Graph-spectral property: algebraic connectivity**

  The algebraic connectivity is the measure of speed of performance of the consensus algorithm [73] and defined as the second smallest eigenvalue of Laplacian matrix L of graph G when L is defined as

$$L := D - A, \tag{3.24}$$

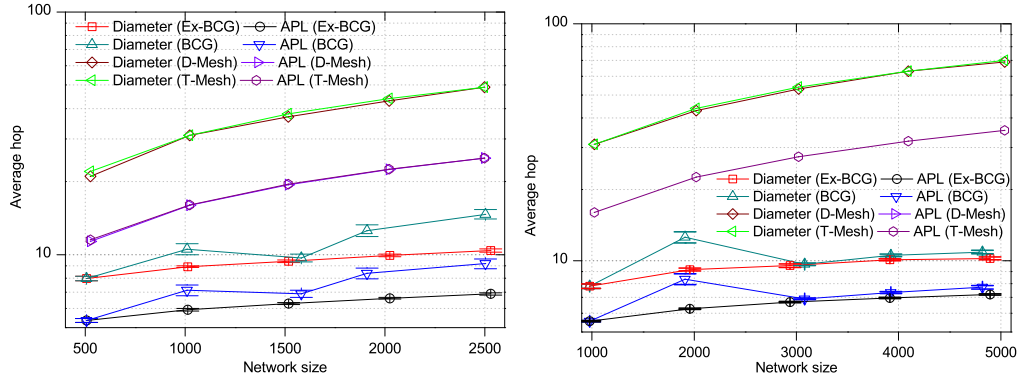
  A is an adjacency matrix of graph G, and D is the diagonal degree matrix of graph G. The algebraic connectivity quantifies the speed of coordination and information distribution of multi-agent systems [74]. Generally, the larger the algebraic connectivity, the faster the consensus speed of the network. Furthermore, the algebraic connectivity is an indirect measure of the robustness of node-failures and edge-failures [73, 75].

Simulation setup

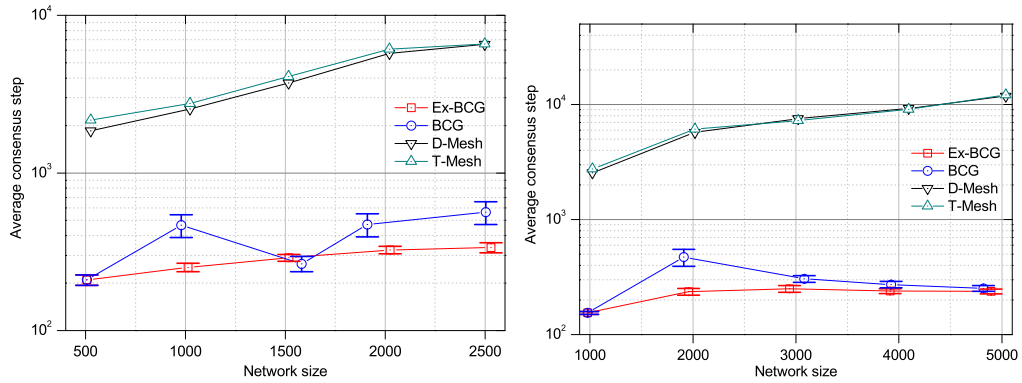
In the simulation, we expand $N_{BCG} = 506$ and $N_{BCG} = 979$ BCGs up to 5 times, and measure the diameter, average path length, algebraic connectivity and consensus steps of the average consensus protocol at each expansion. For performance comparison, we select BCGs, Diagonal and Toroidal Mesh (D-Mesh and T-Mesh) networks of similar sizes. For BCGs and *Ex*-BCGs, we examine *all* possible generator sets (*i.e.*, $g_1 \neq g_2, g = \begin{pmatrix} < a^t >_p & 1 \\ 0 & 1 \end{pmatrix}$, where $0 < t < k$) and

Ex-BCGs							BCGs						Meshes			
N_{Ex-BCG}	p	k	a	n	generator sets	sets	N_{BCG}	p	k	a	generator sets	sets	N_{Mesh}	n'	k'	sample
506	23	22	5	1	$(1,2),\cdots,(20,21)$	210	506	23	22	5	$(1,2),\cdots,(20,21)$	210	525	21	25	1
1012	23	22	5	2	$(1,2),\cdots,(20,21)$	210	1010	101	10	6	$(1,2),\cdots,(8,9)$	36	1023	31	33	1
1518	23	22	5	3	$(1,2),\cdots,(20,21)$	210	1582	113	14	7	$(1,2),\cdots,(12,13)$	78	1517	37	41	1
2024	23	22	5	4	$(1,2),\cdots,(20,21)$	210	1910	191	10	7	$(1,2),\cdots,(8,9)$	36	2021	43	47	1
2530	23	22	5	5	$(1,2),\cdots,(20,21)$	210	2504	313	8	5	$(1,2),\cdots,(6,7)$	21	2499	49	51	1
979	89	11	2	1	$(1,2),\cdots,(9,10)$	45	979	89	11	2	$(1,2),\cdots,(9,10)$	45	1023	31	33	1
1958	89	11	2	2	$(1,2),\cdots,(9,10)$	45	1910	191	10	7	$(1,2),\cdots,(8,9)$	36	2021	43	47	1
2937	89	11	2	3	$(1,2),\cdots,(9,10)$	45	3081	79	39	2	$(1,2),\cdots,(37,38)$	703	3021	53	57	1
3916	89	11	2	4	$(1,2),\cdots,(9,10)$	45	3924	109	36	2	$(1,2),\cdots,(34,35)$	595	4095	63	65	1
4895	89	11	2	5	$(1,2),\cdots,(9,10)$	45	4820	241	20	6	$(1,2),\cdots,(18,19)$	171	5037	69	73	1

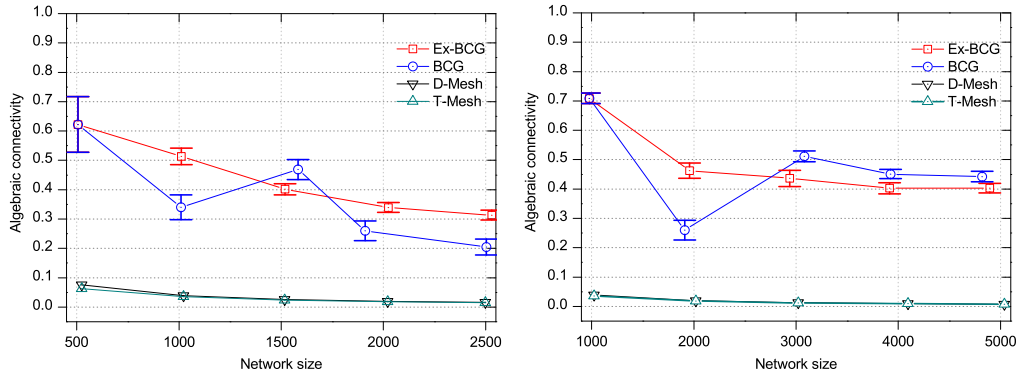
Table 3.3: Network parameters and the number of sets (samples)



(a) Topological properties comparison



(b) Consensus performance comparison



(c) Algebraic connectivity comparison

Figure 3.3: Performance comparison with 95% confidence interval: *Ex*-BCGs expanded from $N_{BCG} = 506$ BCG and BCGs of similar size (left figure) and *Ex*-BCGs expanded from $N_{BCG} = 979$ BCG and BCGs of similar size (right figure)

use their average for the comparison. For example, $N_{Ex\text{-}BCG} = 1,012$ expanded from $N_{BCG} = 506$ BCG has 210 generator sets calculated by $(k-1)(k-2)/2$ with $k = 22$. In the charts, we show a 95% confidence interval for each metric. Table 3.3 summarizes the parameters and the number of samples of the target *Ex*-BCGs, BCGs and D-Mesh and T-Mesh networks used in our comparison. Note that the generating parameters of the target Mesh networks (n' and k') determine the network size $N = n' \times k'$ [55]. From the simulation results, we found that *Ex*-BCGs and BCGs have much better topological properties, consensus steps and algebraic connectivity than D-Mesh and T-Mesh networks.

Topological properties

Figure 3.3(a) shows the average hops of the diameter and average path length of the target *Ex*-BCGs and BCGs of similar sizes. From the experimental results, as the size increases, we observed that the BCGs have non-smooth (unpredictable) average hops over the investigated diameter and average path length. Similarly, BCGs of other sizes display inconsistent performance in topological properties as the parameters p and k change for size expansion. In contrast, the average hops of the diameter and average path length of the target *Ex*-BCGs are short and have small increase rate along the size expansion as illustrated in the figure. Specifically, the increase rate of the average hops of diameter and average path length between $N_{Ex-BCG} = 506$ and $N_{Ex-BCG} = 2,530$ *Ex*-BCGs are 30.4% and 28.6%, respectively. On the other hand, the hop increase rate between $N_{BCG} = 506$ and $N_{BCG} = 2,530$ BCGs have 83.8% and 71.4% increase rates, respectively.

Convergence speed of the average consensus protocol

In analysis of convergence speed, we observed that *Ex*-BCGs have fast and predictable convergence speed when compared to BCGs. Along the size increase, *Ex*-BCGs expanded from $N_{BCG} = 506$ and $N_{BCG} = 979$ BCGs in Figure 3.3(b) keep the average consensus steps below 350, and have 60.1% and 53.9% increase rate, respectively. Unlike *Ex*-BCGs, BCGs of similar sizes exhibit unpredictable consensus steps and large increase rate. For example, the consensus steps of BCGs between $N_{BCG} = 506$ and $N_{BCG} = 2,504$ BCGs show 169% increase rate, which is almost three times larger steps than that of *Ex*-BCGs of similar size. In terms of performance variability, 95% confidences interval of the average consensus step of *Ex*-BCGs are smaller than that of BCGs independent of the size, which means *Ex*-BCGs have more invariable and faster consensus performance.

Algebraic connectivity

Figure 3.3(c) shows the average algebraic connectivity of *Ex*-BCGs and BCGs of similar sizes. As illustrated in the figure, the algebraic connectivity of BCGs fluctuates as the network size grows ($i.e.$, changing the generating parameters). For instance, the average algebraic connectivity of size $N_{BCG} = 1,582$ BCGs is 0.47, but it is sharply reduced to 0.26 in $N_{BCG} = 1,910$ BCG. These inconsistent algebraic connectivity results display evidence of extensive dependence of BCGs on the generating parameters in graph-spectral property. On the contrary, *Ex*-BCGs exhibit a small decrease rate along the size expansion and less confidence interval. Along the expansion from $N_{BCG} = 506$ BCG to its 5 times, *Ex*-BCG displays only 50% decrease rate while the benchmark BCG has 70% decrease rate.

Data statistics of the same subset of generator sets

In this section, we discuss the simulation results from the same subset of generator sets of *Ex*-BCGs and BCGs of similar sizes. For example, BCG of size $N_{BCG} = 1,010$ has 36 generators sets $\{(1,2),(1,3),\cdots,(8,9)\}$, while *Ex*-BCG of size $N_{Ex-BCG} = 1012$ expanded from a BCG of size $N_{BCG} = 506$ has 210 generator sets $\{(1,2),(1,3),\cdots,(20,21)\}$. Among those generator sets, we compare simulation result over the same generator sets $\{(1,2),(1,3),\cdots,(8,9)\}$. The analysis is based on the average, the standard deviation σ and (*min, max*) of the data. The results are summarized in Table 3.4.

Table 3.5 is the statistics of the result from the same subset of generator sets. Overall, we found that *Ex*-BCGs have smaller average hops in diameter and average path length and faster convergence speed than those of BCGs of similar sizes. Also, the standard deviations of *Ex*-BCGs for the investigated metrics are smaller than those of BCGs of similar sizes. In analysis of minimum and maximum values, we observed *Ex*-BCGs not only have less difference between minimum and maximum values, but also show superior optimal performance in most metrics to that of BCGs. In other words, *Ex*-BCGs are more invariable and efficient communication topology than that of BCGs independent of the size.

Ex-BCGs							BCGs					
N_{Ex-BCG}	p	k	a	n	generator sets	number of sets	N_{BCG}	p	k	a	generator sets	number of sets
1012	23	22	5	2	$(1,2),\cdots,(8,9)$	36	1010	101	10	6	$(1,2),\cdots,(8,9)$	36
1518	23	22	5	3	$(1,2),\cdots,(12,13)$	78	1582	113	14	7	$(1,2),\cdots,(12,13)$	78
2024	23	22	5	4	$(1,2),\cdots,(8,9)$	36	1910	191	10	7	$(1,2),\cdots,(8,9)$	36
2530	23	22	5	5	$(1,2),\cdots,(6,7)$	21	2504	313	8	5	$(1,2),\cdots,(6,7)$	21
1958	89	11	2	2	$(1,2),\cdots,(8,9)$	36	1910	191	10	7	$(1,2),\cdots,(8,9)$	36
2937	89	11	2	3	$(1,2),\cdots,(9,10)$	45	3081	79	39	2	$(1,2),\cdots,(9,10)$	45
3916	89	11	2	4	$(1,2),\cdots,(9,10)$	45	3924	109	36	2	$(1,2),\cdots,(9,10)$	45
4895	89	11	2	5	$(1,2),\cdots,(9,10)$	45	4820	241	20	6	$(1,2),\cdots,(9,10)$	45

Table 3.4: BCGs and *Ex*-BCGs parameters and common generator sets

Ex-BCGs expanded from $N_{BCG} = 506$ BCG	expansion factor $n = 2$		expansion factor $n = 3$		expansion factor $n = 4$		expansion factor $n = 5$	
	$N_{Ex-BCG} = 1012$	$N_{BCG} = 1010$	$N_{Ex-BCG} = 1518$	$N_{BCG} = 1582$	$N_{Ex-BCG} = 2024$	$N_{BCG} = 1910$	$N_{Ex-BCG} = 2530$	$N_{BCG} = 2504$
	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)
Diameter	8.67 (0.91, 36) (8, 11)	10.5 (1.54, 36) (9, 14)	9.41 (0.70, 78) (8, 11)	9.63 (1.42, 78) (8, 13)	9.9 (0.83, 36) (9, 12)	12.6 (1.91, 36) (9, 17)	10.4 (0.89, 21) (9, 12)	14.7 (1.41, 21) (12, 17)
Average path length	5.78 (0.30, 36) (5.50, 6.57)	7.13 (1.03, 36) (5.80, 9.08)	6.33 (0.40, 78) (5.87, 7.12)	6.86 (0.94, 78) (6.09, 8.90)	6.51 (0.19, 36) (6.22, 6.98)	8.37 (1.20, 36) (6.56, 10.7)	6.75 (0.24, 21) (6.49, 7.42)	9.17 (0.90, 21) (7.70, 10.8)
Consensus step	233.3 (65.7, 36) (160, 439)	465.7 (214.2, 36) (228, 935)	294.9 (90.0, 78) (168, 616)	265.4 (119.1, 78) (173, 553)	329.4 (47.3, 36) (182, 711)	471.9 (221.9, 36) (200, 958)	316.3 (132.8, 21) (195, 699)	563.4 (202.2, 21) (340, 1026)
Algebraic connectivity	0.53 (0.14, 36) (0.27, 0.73)	0.34 (0.18, 36) (0.15, 0.56)	0.40 (0.12, 78) (0.16, 0.67)	0.47 (0.14, 78) (0.19, 0.63)	0.33 (0.11, 36) (0.14, 0.51)	0.26 (0.09, 36) (0.11, 0.47)	0.32 (0.09, 21) (0.11, 0.42)	0.20 (0.06, 21) (0.10, 0.31)

Ex-BCGs expanded from $N_{BCG} = 979$ BCG	expansion factor $n = 2$		expansion factor $n = 3$		expansion factor $n = 4$		expansion factor $n = 5$	
	$N_{Ex-BCG} = 1958$	$N_{BCG} = 1910$	$N_{Ex-BCG} = 2937$	$N_{BCG} = 3081$	$N_{Ex-BCG} = 3916$	$N_{BCG} = 3924$	$N_{Ex-BCG} = 4895$	$N_{BCG} = 4820$
	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)	average (σ, N) (min, max)
Diameter	9.14 (0.35, 36) (9, 10)	12.6 (1.91, 36) (9, 17)	9.53 (0.54, 45) (9, 11)	9.38 (0.90, 45) (9, 12)	10.09 (0.28, 45) (10, 11)	10.3 (0.96, 45) (9, 13)	10.2 (0.47, 45) (10, 12)	11.1 (1.42, 45) (10, 15)
Average path length	6.24 (0.09, 36) (6.12, 6.45)	8.37 (1.20, 36) (6.56, 10.7)	6.68 (0.13, 45) (6.48, 7.21)	6.75 (0.27, 45) (6.51, 7.58)	6.97 (0.13, 45) (6.80, 7.55)	7.2 (0.42, 45) (6.8, 8.5)	7.2 (0.14, 45) (7.04, 7.77)	7.9 (0.95, 45) (7.06, 10.4)
Consensus step	228.6 (42.9, 36) (175, 374)	471.9 (221.9, 36) (200, 958)	249.4 (57.5, 45) (173, 418)	258.1 (168.2, 45) (160, 801)	238.6 (41.2, 45) (182, 366)	235.5 (97.3, 45) (168, 588)	237.3 (38.7, 45) (193, 363)	262.6 (103.9, 45) (174, 549)
Algebraic connectivity	0.47 (0.08, 36) (0.27, 0.60)	0.26 (0.09, 36) (0.11, 0.47)	0.44 (0.09, 45) (0.25, 0.64)	0.58 (0.19, 45) (0.13, 0.83)	0.40 (0.07, 45) (0.26, 0.55)	0.49 (0.13, 45) (0.15, 0.64)	0.40 (0.06, 45) (0.26, 0.47)	0.44 (0.11, 45) (0.19, 0.60)

Table 3.5: Data statistics from identical generator sets of BCGs and *Ex*-BCGs expanded from a BCG of size $N_{BCG} = 506$ (upper) and a BCG of size $N_{BCG} = 979$ (lower). Note N denotes the number of generator sets

3.4 Resizing Ex-BCGs

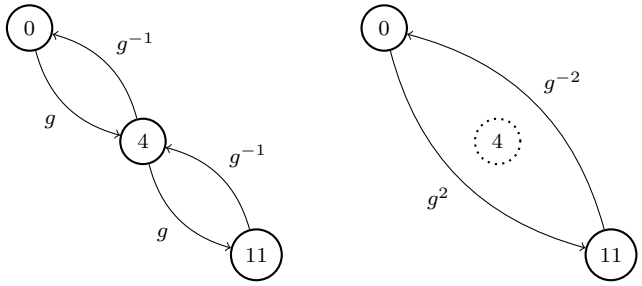
3.4.1 Resizing algorithm

In Section 3.2, we studied the connection rule of *Ex*-BCGs and identified the characteristics of consecutive connections of generator g as following:

$$\underbrace{g * g * \cdots * g}_{m} = g^m, \tag{3.25}$$

$$\underbrace{g^{-1} * g^{-1} * \cdots * g^{-1}}_{m'} = g^{-m'}, \tag{3.26}$$

where m and m' are an integer.



(a) A sequence of connections rooted at node 0 in 21-node BCG.

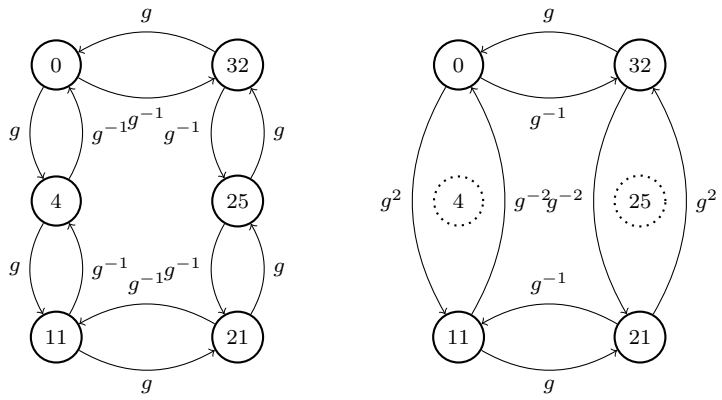
(b) Pruning node 4 and applying the CTR algorithm between nodes 0 and 11

Figure 3.4: CTR algorithm illustration on a 21-node BCG

In this section, we describe the Cut-Through Rewiring (CTR) algorithm of *Ex*-BCGs and show differences from that of BCG with an example based on Eq. (3.25) and Eq. (3.26).

Suppose we have 21-node BCG with $p = 7, k = 3, a = 2$ and $g = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}$. A sequence of connections rooted at node 0 with generator g produces edges from node 0 to node 4 and to node 11 as shown in Figure 3.4(a) ($0 * g = 4$ and $4 * g = 11$). If we prune node 4 and apply the CTR between nodes 0 and 11, the resulting connection is as shown in Figure 3.4(b).

Similarly, with 42-node *Ex*-BCG expanded from 21-node BCG using the same generator g, the sequence of connections rooted at node 0 is formed in Fig. 3.5(a).



(a) A sequence of connections rooted at node 0 in 42-node *Ex*-BCG.

(b) Pruning nodes 4 and 25, and applying the CTR algorithm between nodes 0 and 11 and between nodes 21 and 32.

Figure 3.5: CTR algorithm illustration on a 42-node *Ex*-BCG

If we prune nodes 4 and 25 and apply the CTR algorithm to nodes 0 and 11, nodes 21 and 32 while conserving degree-4 (we obtain the sequence of connections in Fig. 3.5(b)). Analogous to the CTR algorithm for BCGs in [64], the CTR algorithm for *Ex*-BCGs is a resizing methodology to build an arbitrarily sized network with a constant degree.

The number of the prunable nodes of *Ex*-BCGs without node isolation is affected by the expansion factor n. In BCGs with a generator $g = \begin{pmatrix} < a^t >_p & y \\ 0 & 1 \end{pmatrix}$, where $0 < t < k$, the number of prunable nodes in a sequence of connections in the g direction is $m - 2$ with $g^m = I$, where m is a positive integer and smaller than k [64]. When BCGs with generator g are expanded with expansion factor n, the number of prunable nodes in a sequence of connections without node isolation in the g direction can be bounded. Applying Proposition. 3 ($g^{nm} = I$) as follows

$$m - 2 \leq \text{number of prunable nodes} \leq nm - 2 \tag{3.27}$$

For example, a sequence of connections rooted at node 0 in Figure 3.4 is $0 \rightarrow 4 \rightarrow 11 \rightarrow 0$ with $g^3 = I$. Thus, the number of prunable nodes without node isolation is $1 = (3 - 2)$. After expansion with $n = 2$, a sequence of connections rooted at node 0 are extended as $0 \rightarrow 4 \rightarrow 11 \rightarrow 21 \rightarrow 25 \rightarrow 32 \rightarrow 0$ with $g^6 = I$, which means that the number of prunable nodes without node isolation is

$4 = (2 \times 3 - 2)$. Based on the discussion so far, we illustrated how to improve *Ex*-BCGs by the CTR algorithm.

3.4.2 Performance Evaluation

Simulation Setup

We use simulation model to quantify size flexibility and fault-tolerance of BCGs and *Ex*-BCGs after applying the CTR algorithm. In the simulation, we expand 506-node and 979-node BCGs up to 5 times. We chose the *Ex*-BCGs with a set of generators (g_1, g_2) that has fastest consensus speed (from in Section 3.3.3). We randomly prune nodes of each *Ex*-BCG from 10% to 90%. At each pruning cycle, we apply the CTR algorithm to neighbors of the pruned nodes and measure the connectivity, diameter, average path length and consensus step out of 30 network samples. For comparison study, we use BCGs of similar sizes with an optimal set of generators. For investigating topological properties and consensus step, we put 95% confidence interval for an accurate analysis. Tab. 3.6 captures the detail of the simulation parameters.

Ex-BCGs								BCGs						
N_{Ex-BCG}	p	k	a	t_1	t_2	n	samples	N_{BCG}	p	k	a	t_1	t_2	samples
1012	23	22	5	1	3	2	30	1010	101	10	6	3	7	30
1518	23	22	5	3	12	3	30	1582	113	14	1	1	9	30
2024	23	22	5	3	21	4	30	1910	191	10	1	9	9	30
2530	23	22	5	4	21	5	30	2504	313	8	5	1	7	30
1958	89	11	2	2	8	2	30	1910	191	10	7	1	9	30
2937	89	11	2	4	5	3	30	3081	79	39	2	14	33	30
3916	89	11	2	4	5	4	30	3924	109	36	2	5	25	30
4895	89	11	2	6	9	5	30	4820	241	20	6	11	13	30

Table 3.6: Parameters and number of samples

Connectivity

Fig. 3.6 and Fig. 3.7 show the connectivity of the resized *Ex*-BCGs and BCGs by the CTR algorithm. In Fig. 3.6, the pruned *Ex*-BCGs expanded from 506-node BCG generate almost fully connected networks after size reduction from 10% to 90%. Unlike, the resized networks of the target BCGs have less connectivity when compared to that of *Ex*-BCGs. For example, the resized networks from 2504-node

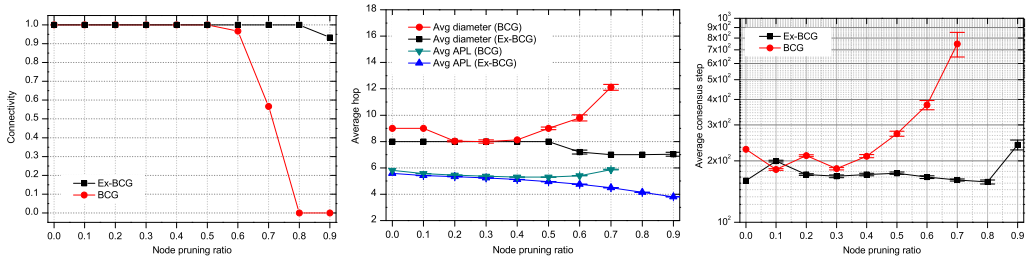
BCG do not produce connected network beyond 70% node pruning, whereas the pruned networks of the 2530-node *Ex*-BCG are fully connected after pruning up to its 10% its original size.

Topological properties & Convergence Speed

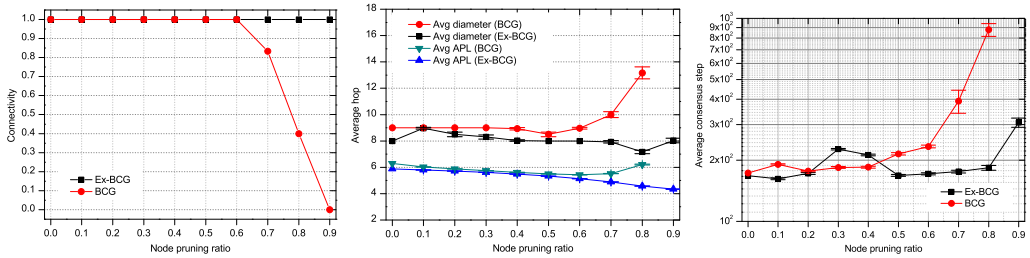
In Fig. 3.6 and Fig. 3.7, we observe that the average hop counts of the diameter and average path length of the pruned *Ex*-BCGs remain low, even when the node pruning ratio of *Ex*-BCGs expanded from a 506-node and a 979-node BCGs increases up to 90% and 60%, respectively. Also, the pruned *Ex*-BCGs expanded from 506-node BCG result in fast convergence speed and almost full connectivity over the pruning range 10% to 90%. From the comparison study, the pruned *Ex*-BCGs have almost the same or superior performance to that of BCGs along the node pruning ratio.

3.5 Discussion

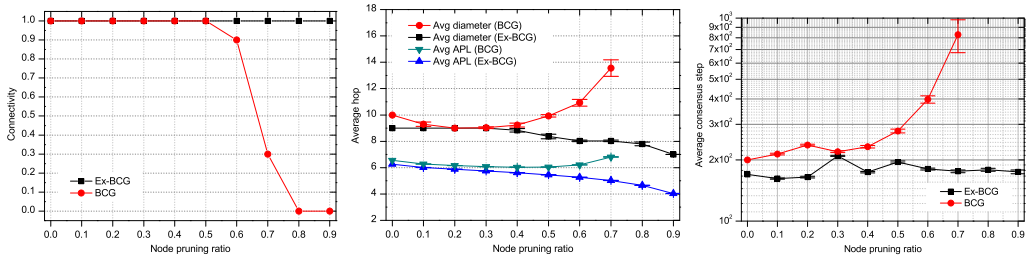
In this chapter, we presented *Ex*-BCGs and resized it using the CTR algorithm. The *Ex*-BCGs deterministically expand BCGs while preserving their beneficial network properties: topological/spectral properties, class-level vertex transitivity and GCR representation. The simulation proved that *Ex*-BCGs have a consistent network performance along large size expansion unlike BCGs. Furthermore, the resized *Ex*-BCGs with the CTR algorithm and the resultant networks have a strong connectivity (maintaining connectivity up to 80% node removal) and efficient topological/spectral network properties.



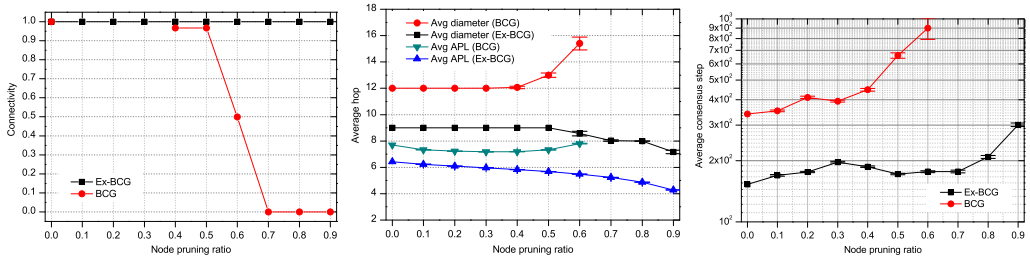
(a) Comparison of 1012-node *Ex*-BCG and 1010-node BCG: connectivity (left), topological properties (center) and consensus step (right)



(b) Comparison of 1518-node *Ex*-BCG and 1582-node BCG: connectivity (left), topological properties (center) and consensus step (right)

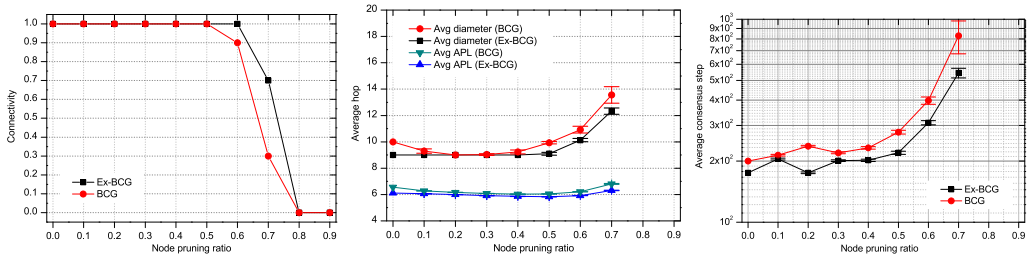


(c) Comparison of 2024-node *Ex*-BCG and 1910-node BCG: connectivity (left), topological properties (center) and consensus step (right)

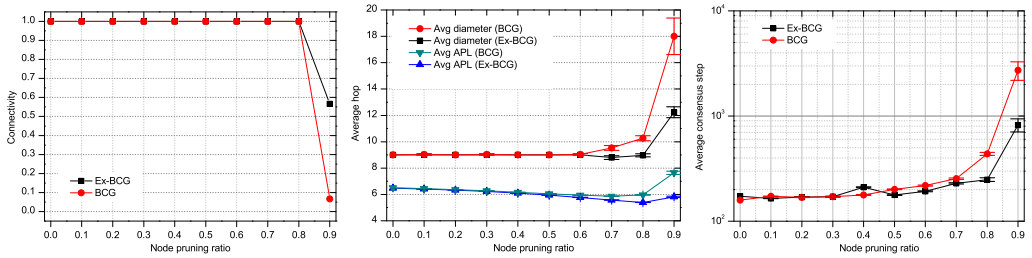


(d) Comparison of 2530-node *Ex*-BCG and 2504-node BCG: connectivity (left), topological properties (center) and consensus step (right)

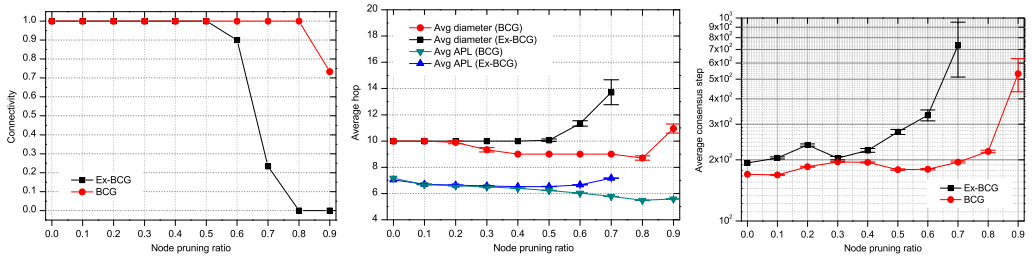
Figure 3.6: Size flexibility and fault-tolerance comparison of *Ex*-BCGs expanded from a 506-node BCG and BCGs of similar size 95% confidence interval for average hops and average consensus steps. x-axis is the ratio of the pruned nodes.



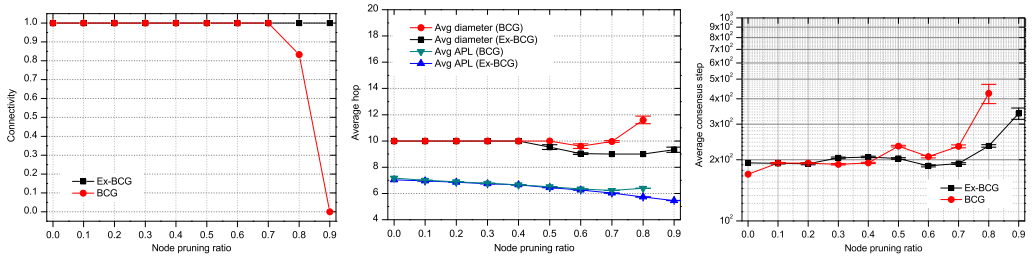
(a) Comparison of 1958-node *Ex*-BCG and 1910-node BCG: connectivity (left), topological properties (center) and consensus step (right)



(b) Comparison of 2937-node *Ex*-BCG and 3081-node BCG: connectivity (left), topological properties (center) and consensus step (right)



(c) Comparison of 3916-node *Ex*-BCG and 3924-node BCG: connectivity (left), topological properties (center) and consensus step (right)



(d) Comparison of 4895-node *Ex*-BCG and 4820-node BCG: connectivity (left), topological properties (center) and consensus step (right)

Figure 3.7: Size flexibility and fault-tolerance comparison of *Ex*-BCGs expanded from a 979-node BCG and BCGs of similar size 95% confidence interval for average hops and average consensus steps. x-axis is the ratio of the pruned nodes.

Chapter 4

Routing Protocols for Expanded Borel Cayley Graphs

In this chapter, we propose the Class-level Vertex Transitive (CVT) routing protocol for *Ex*-BCGs and the Aggressive Multi-path Aware (AMA) routing protocol for the resized *Ex*-BCGs.

4.1 Class-level Vertex Transitive Routing Protocol

In this section, we present the *class-level vertex transitive* (CVT) routing algorithm for *Ex*-BCGs. Then, we provide the routing table generation and routing table folding schemes to reduce an overhead in routing.

4.1.1 Routing Protocol

In the previous chapter, we showed *Ex*-BCGs are class-level vertex transitive. In other words, once paths from a class representing node to all other nodes are found, these paths can be used by nodes in the **same class**. However, memorizing all paths is not efficient. Thus, our CVT routing algorithm is an iterative process of informing a node that has a packet to route of a direction to forward it along the optimal path by referring to the predefined routing table (We will discuss how to generate this routing table in the next section).

With the CVT routing protocol, the path between a source $i = m_1k + c_1$ and a destination $j = m_2k + c_2$ is identical to that path between i_c and j_c, where i_c is

a class representing node of i and j_c is a corresponding node to j from i_c's view (Eq. 3.17). To find j_c, we provide the following Corollary:

Corollary 2. For an *Ex*-BCG with a GCR representation and transform element T, if $i \Rightarrow T^{m_1} * \alpha_{c_1}$ is connected to $j \Rightarrow T^{m_2} * \alpha_{c_2}$ through a sequence of generators, $i_c \Rightarrow \alpha_{c_1}$ is connected to j_c through the same sequence of generators and j_c is defined as

$$j_c =< j - (i - i_c) >_N, \tag{4.1}$$

where N is a network size.

Proof. If we assume a sequence of generators between i and j as A, the following is satisfied

$$T^{m_1} * \alpha_{c_1} * A = T^{m_2} * \alpha_{c_2},\ A = \alpha_{c_1}^{-1} * T^{-m_1} * T^{m_2} * \alpha_{c_2}. \tag{4.2}$$

Since i_c is connected to j_c through A,

$$\alpha_{c_1} * A = T^{m_2 - m_1} * \alpha_{c_2} = T^{m'_2} * \alpha_{c'_2} \tag{4.3}$$

where $T^{-m_1} = T^{np-m_1}$ and then $T^{np-m_1} * T^{m_2} = T^{m_2-m_1}$. Finally, Eq (4.3) is translated to $j_c = (m_2 - m_1)k + c_2 = m_2k + c_2 - m_1k$ identical to Eq (4.1). □

Based on the identified direction (generator) from finding j_c, i forwards a packet to the neighbor in the selected direction by referring to the routing table. This process continues until $i = j$. We will discuss how to generate the routing table informing an optimal direction incorporating with Corollary 2.

4.1.2 GCR Constant based Routing Table Generation

The CVT routing algorithm 1) identifies a class representing node i_c of i and a corresponding node j_c to j from i_c's view and 2) i forwards a packet to the neighbor in the selected direction by look up the routing table. To inform an optimal direction, ***the routing table should reflect the view from a class representing node to other nodes classified with directions (generators)***. The routing table of the vertex-transitive routing algorithm of BCGs is obtained by organizing a Minimal Spanning Tree (MST) rooted at node 0 and recording nodes at each branch based on generators, which is carried out by referring to the network adjacency list. The basic idea of generating the routing table for the CVT routing algorithm

is to construct an MST equivalent to that of the vertex-transitive routing algorithm. However, the adjacency list based scheme results in a routing table size $O(N + M)$, where N is a network size and M is the number of edges. To generate a routing table with a lower size complexity, we introduce the GCR constant based routing table generation scheme.

In Section 3.3, we showed that *Ex*-BCGs can be expressed with a GCR representation. This is reinterpreted as the connection between nodes in *Ex*-BCGs can be described with GCR constants. The GCR constant is the integer node ID difference between two immediately connected nodes in the GCR representation. Thus, every node in *Ex*-BCGs has class and generator specific GCR constants for all its connections.

For example, in a 42-node *Ex*-BCG expanded from a 21-node BCG with $p = 7, k = 3, a = 2, n = 2$ and $g_1 = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}$, $g_2 = \begin{pmatrix} 2^2 & 1 \\ 0 & 1 \end{pmatrix}$, each node has GCR constants deriving from the transform element $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ as described in Table 4.1.

	g_1 direction	g_1^{-1} direction	g_2 direction	g_2^{-1} direction
If node i of class 0 has :	$+4$	-10	$+5$	-5
If node i of class 1 has :	$+7$	-4	$+5$	-11
If node i of class 2 has :	$+10$	-7	$+11$	-5

Table 4.1: GCR constants of 42-node *Ex*-BCG

Based on these GCR constants, we explain how to generate a routing table with the 42-node *Ex*-BCG example mentioned previously.

Utilizing the GCR constants in Table 4.1, we construct an MST rooted at node 0 (a class representing node of class 0). As shown in Fig. 4.1, on top of the tree, the node ID differences between node 0 and its neighbors are $+4, -10, +5, -5$ in $g_1, g_1^{-1}, g_2, g_2^{-2}$ directions, respectively. Resulting neighbors' constants are designated in the circles. At each step of expanding the tree, the routing table entries in Fig. 4.2 are identified by modulo 42 operation of the resultant integer, and then corresponding direction column is filled with 1 when there is a match, 0 otherwise. For example, -10 at step 1 is equivalent to $32 =< -10 >_{42}$, and then g_1^{-1} column of 33^{th} row (Note that the routing entry starts with 0) of the routing table is filled with 1. This process is iterated until all the entries are filled with 1 or 0. As a result, the routing table contains information of optimal direction (generator) from node 0 to all other nodes. Since there are 3 classes, the routing table generation

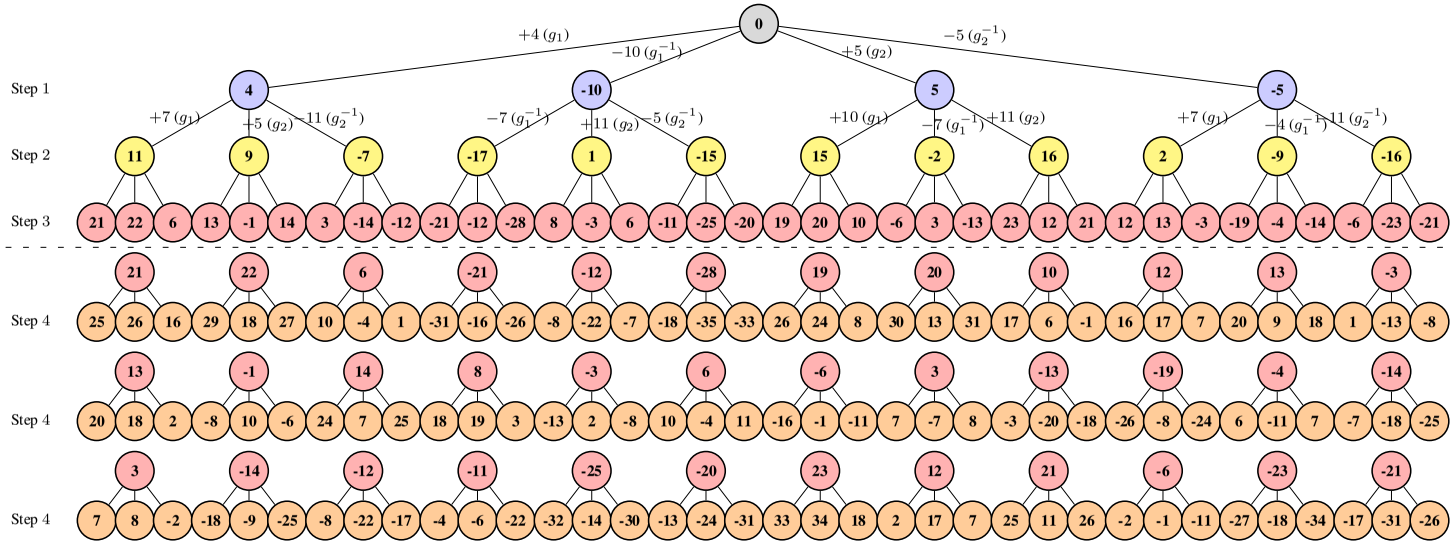


Figure 4.1: 42-node *Ex*-BCG routing tree rooted at node 0

	g_1	g_2	g_1^{-1}	g_2^{-1}		g_1	g_2	g_1^{-1}	g_2^{-1}
0	0	0	0	0	21	1	1	1	1
1	0	0	1	0	22	1	0	1	0
2	0	0	0	1	23	0	1	0	1
3	1	1	0	0	24	1	1	1	1
4	1	0	0	0	25	0	0	1	0
5	0	1	0	0	26	0	0	0	1
6	1	0	1	0	27	0	0	1	0
7	1	1	1	1	28	1	0	0	1
8	0	0	1	0	29	0	1	0	0
9	1	0	0	0	30	1	0	1	0
10	0	1	0	0	31	0	0	1	0
11	1	0	0	0	32	0	0	1	0
12	0	1	0	1	33	0	0	0	1
13	1	0	0	1	34	1	1	1	1
14	1	0	1	0	35	1	0	0	0
15	0	1	0	0	36	0	1	0	1
16	0	1	0	0	37	0	0	0	1
17	0	0	1	0	38	0	0	0	1
18	1	1	1	1	39	0	0	1	1
19	0	1	0	1	40	0	1	0	0
20	0	1	0	0	41	1	0	0	0

Figure 4.2: 42-node *Ex*-BCG's Routing table from node 0

process is operated for nodes of class 1 and 2 by organizing MSTs rooted at node 1 and node 2. Based on these routing tables, the CVT routing algorithm identifies the neighbor to forward a packet with j_c of Eq (4.1). For example, when node 6 needs to find the neighbor to forward a packet to node 10, corresponding j_c is 4 from Eq (4.1). Then, node 6 checks 5^{th} row and figures out from the routing table in Fig. 4.2 that the neighbor in g_1 direction is one to forward a packet (g_1 column has 1).

Analytically, our GCR constant based routing table generation algorithm needs only GCR constants, whose size complexity is $O(dk)$, where d is a node degree and k is the number of classes, which is much smaller than $O(N + M)$ of the adjacency list based scheme.

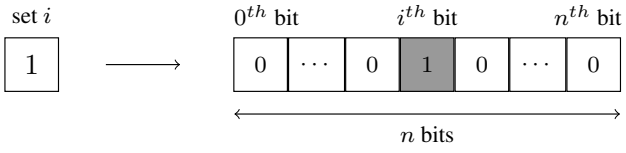


Figure 4.3: Padding 0 to routing entry of set i to make n bits

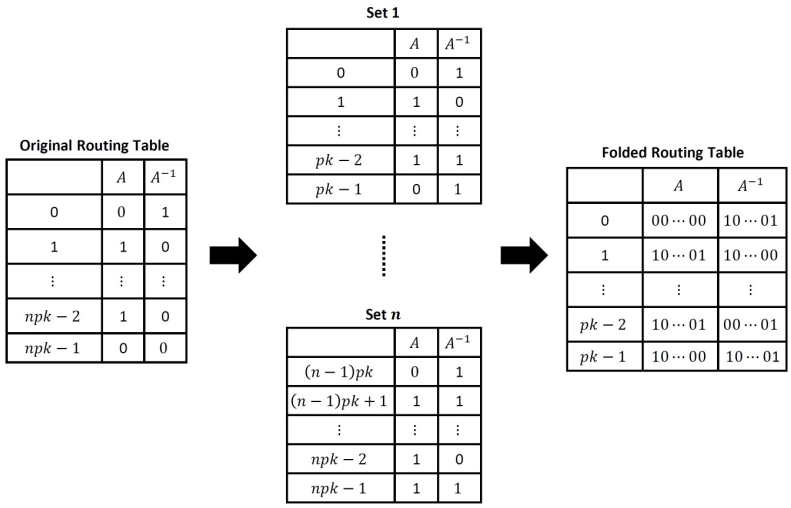


Figure 4.4: Routing table folding

4.1.3 Routing Table Folding

The routing table discussed in the previous section has $N = npk$ routing entries, where n is an expansion factor, p and k are generating parameters. This is inefficient when the network size increases. In this section, we provide the routing table folding to reduce N to N/d.

Ex-BCGs with an expansion factor n have routing entries ranging from 0 to $npk - 1$. Let us split the routing table into n subtables, each of which with pk entries. So, x^{th} $(i = 1, 2, \cdots, n)$ subtable has routing entries ranging from xpk to $(x + 1)pk - 1$. Next, we pad each routing entry with 0 until it has n bits as shown in Fig. 4.3.

Then, the routing tables are folded to its $1/n$ size by the **OR** operation between the folded entries. Fig. 4.4 shows an example of how to fold routing tables for generators A and A^{-1}, and Table 4.2 explains how to look up the folded routing table.

Step 1 : Identify the routing table set index x by $xpk \leq j_c < (x + 1)pk$.
Step 2 : Find a with $a = j_c - xpk$
Step 3 : At $a + 1^{th}$ row of the table, select a generator (direction) whose x^{th} bit is 1.

Table 4.2: How to look up the folded routing table.

Therefore, when d is the node degree and N is the network size, the folded routing table has Nd/n routing entries, compared to Nd routing entries without folding.

As an example, we consider the routing tables of a 1081-node BCG with $p = 47, k = 23, a = 2$ and a 1100-node *Ex*-BCG expanded from 110-node BCG with $p = 11, k = 10, a = 2$ and $n = 10$. With a network node degree of 4, the 1100-node *Ex*-BCG needs $(110 \times 4)/10 = 44$ routing entries, whereas 1081-node BCG requires $1081 \times 4 = 4324$ routing entries.

The CVT routing algorithm requires each node to track optimal routing through its one hop neighbors with each neighbor identified by Eq (4.1 and its own routing table (the distributed and self-routing algorithm). Based on the discussion so far, we can summarize the CVT routing algorithm in Table 4.3.

Given source i and destination j, while $(i \neq j)$
Step 1 : Identify $i_c =< i >_k$ and $j_c =< j - (i - i_c) >_N$.
Step 2 : Select an outgoing link by lookup the folded routing table.
Step 3 : Let i be the neighbor from the selected outgoing link at step 2

Table 4.3: CVT routing algorithm

4.1.4 Routing Example

Let us consider routing from node 7 to node 29 in a 42-node *Ex*-BCG expanded from a 21-node BCG with $g_1 = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}$ and $g_2 = \begin{pmatrix} 2^2 & 1 \\ 0 & 1 \end{pmatrix}$. Table 4.4 describes the routing iterations and Fig. 4.5 shows the routing tables. The resultant routing path is $7 \xrightarrow{g_1} 14 \xrightarrow{g_1} 24 \xrightarrow{g_2} 29$.

1^{st} iteration :	$i = 7, j = 29 \Rightarrow i_c = 1, j_c = 23$, then
	select g_1 by lookup of 3^{th} row of $class$ 1 routing table.
2^{nd} iteration :	$i = 14, j = 29 \Rightarrow i_c = 2, j_c = 17$, then
	select g_1 by lookup of 18^{th} row of $class$ 2 routing table.
3^{rd} iteration :	$i = 24, j = 29 \Rightarrow i_c = 0, j_c = 5$, then
	select g_2 by lookup of 6^{th} row of $class$ 0 routing table.

Table 4.4: Routing iterations from node 7 to node 29

	g_1	g_2	g_1^{-1}	g_2^{-1}
0	01	01	01	01
1	01	00	11	00
2	00	01	00	11
3	11	11	01	01
4	10	00	01	00
5	00	10	00	01
6	10	00	11	00
7	11	10	10	11
8	00	01	10	00
9	11	00	01	00
10	00	10	01	00
11	10	00	01	00
12	00	10	00	11
13	11	01	01	11
14	11	00	10	00
15	00	11	00	01
16	00	10	00	01
17	00	00	10	01
18	10	10	11	11
19	00	11	00	10
20	01	10	00	00

(a) Class 0 routing table

	g_1	g_2	g_1^{-1}	g_2^{-1}
0	00	01	00	11
1	01	01	01	01
2	01	00	11	00
3	11	00	01	00
4	00	10	00	11
5	01	00	00	10
6	00	10	00	01
7	11	11	01	01
8	10	00	01	00
9	10	10	11	11
10	00	11	00	01
11	00	10	00	01
12	00	01	10	00
13	10	00	11	00
14	01	00	00	10
15	11	11	01	01
16	10	10	11	11
17	00	11	00	10
18	10	00	01	00
19	11	00	01	00
20	10	11	11	10

(b) Class 1 routing table

	g_1	g_2	g_1^{-1}	g_2^{-1}
0	01	00	11	00
1	00	01	00	11
2	01	01	01	01
3	10	00	00	01
4	01	11	11	01
5	10	00	11	00
6	01	00	00	10
7	11	00	01	00
8	00	10	00	11
9	00	11	00	01
10	10	01	00	00
11	10	10	11	11
12	10	00	01	00
13	00	10	00	01
14	11	11	01	01
15	00	10	01	00
16	10	00	01	00
17	11	00	01	00
18	00	10	00	01
19	00	00	10	01
20	00	11	00	01

(c) Class 2 routing table

Figure 4.5: Routing tables of 42-node *Ex*-BCG

4.1.5 Performance Evaluation

In this section, we evaluate the CVT routing algorithm of *Ex*-BCGs. For performance comparison, we select BCG [61], Diagonal Mesh (D-Mesh) [55], Toroidal Mesh (T-Mesh) [55] and Undirected De Bruijn (UDB) [76] networks of a similar size as benchmark networks and utilize their optimal routing algorithms.

Network Model

For the simulation, we consider the network model of [77] with a small modification. A network consists of nodes illustrated in Fig. 4.17 and zero delay links. Each node has a traffic extractor, d_n^i receive buffers (Rx buffers), a switching fabric and d_n^o transmit buffers (Tx buffers). The traffic extractor diverts incoming packets arriving at the destination to the local station. The switching fabric maps packets from Rx buffers to Tx buffers following the network routing algorithm. To model the Rx and Tx buffers, we use finite size buffers of sizes ranging from 1 to 4. Since we consider degree 4 networks in the simulation, the sum of Tx and Rx buffer sizes is 8, 16 and 24. The local packet generator uses a *Bernoulli* process with packet generation ratio as parameter.

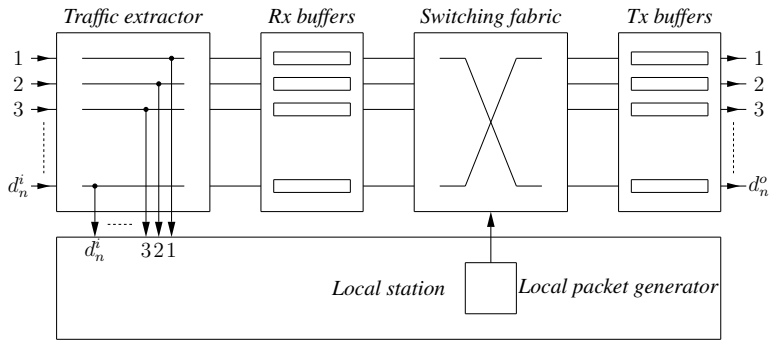


Figure 4.6: Node model

Similar to [77], time is slotted and synchronized such that nodes can receive and transmit packets in the same time slot. Each time slot has two phases: a packet switching phase (switching packets from Rx buffers to Tx buffers) and a packet transmitting phase (sending packets from Tx buffers to Rx buffers). During each time slot, the local station can receive at most d_n packets. When Rx buffer is full, new packets are dropped (overflow). In mapping the packet from Rx buffer to Tx buffer, if the selected Tx buffer is full, the switching fabric seeks another Tx buffer (alternative shortest path). When all alternate Tx buffers are full, the packet is not dropped, but stays in the Rx buffer.

Metrics

For evaluation, we use the following metrics:

- **Reachability**

  Reachability is a *reliability* metric for the routing algorithm under study. It measures the number of successfully routed packets:

$$Reachability = \frac{number\ of\ succeeded\ packets}{number\ of\ generated\ packets} \tag{4.4}$$

- **End-to-End (ETE) delay**

  Evaluates the *efficiency* of the routing algorithm:

$$ETE = \frac{\textit{number of time slots required for successfully routed packets}}{\textit{number of successfully routed packets}} \tag{4.5}$$

- **The mean number of packets at node**

  With a given traffic, the mean number of packets at node is a critical metric of routing capacity quantifying tolerance to bottleneck and network structural efficiency against congestion and queuing delay. The mean number of packets at a node is measured as following:

$$\frac{\textit{number of packets in Rx and Tx Buffers at a node for simulation time slots}}{\textit{number of simulation time slosts}} \tag{4.6}$$

Target networks

For the simulation, we expand a 506-node BCG to a 1012-node *Ex*-BCGs of a degree 4 with an expansion factor $n = 2$. For comparison study, we select a 1010-node BCG, a 1015-node T-Mesh, a D-Mesh and a 1024-node UDB each of degree 4. The detail of the network parameters for *Ex*-BCG and BCG are described in Table 4.5.

	Size	Generating parameters	Generator sets
Ex-BCG	1012	$p = 23, k = 22, a = 5, n = 2$	$g_1 = \begin{pmatrix} 2^3 & 1 \\ 0 & 1 \end{pmatrix}, g_2 = \begin{pmatrix} 2^7 & 1 \\ 0 & 1 \end{pmatrix}$
BCG	1010	$p = 101, k = 10, a = 6$	$g_1 = \begin{pmatrix} 2^3 & 1 \\ 0 & 1 \end{pmatrix}, g_2 = \begin{pmatrix} 2^7 & 1 \\ 0 & 1 \end{pmatrix}$

Table 4.5: Network parameters

Traffic patterns

We consider two traffic patterns: *All-to-All* and *All-to-100*. The *All-to-All* traffic pattern assumes that a source uniformly and randomly selects a destination among

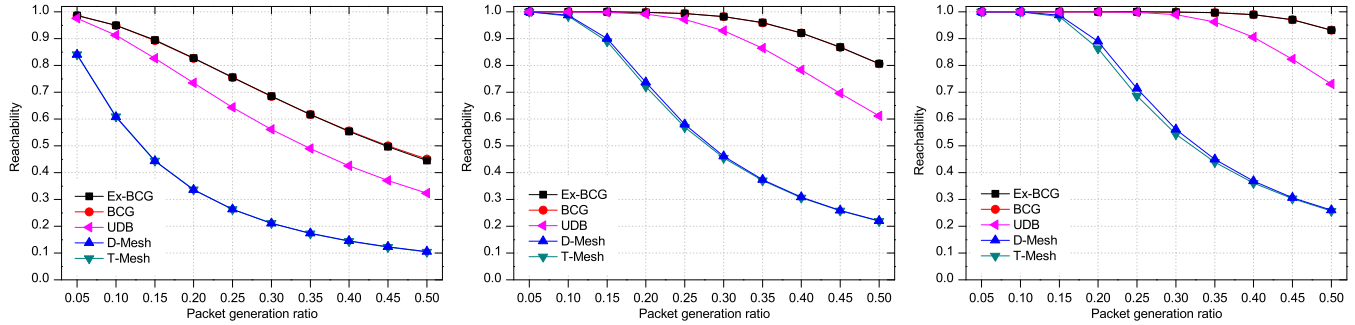
Traffic pattern	Packet generation rate
All-to-All traffic	$0.05, 0.1, \cdots, 0.5$
All-to-100 traffic	$0.02, 0.04, \cdots, 0.2$

Table 4.6: Traffic patterns and packet generation rate

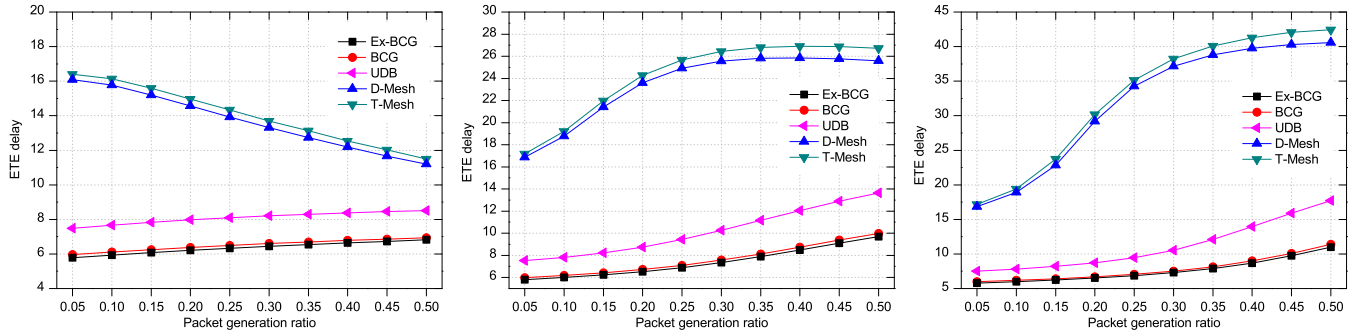
all nodes. To evaluate the ability for the routing algorithms to cope with congestions, we employ the *All-to-100* traffic pattern. There, a source uniformly and randomly chooses a destination out of 100 nodes randomly selected at the beginning of the simulation. The packet generation rates for the two traffic patterns are described in Table 4.6. We run each simulation for $10,000$ time slots, obtain the average of the metrics collected from 10 network samples (where applicable) and compare them.

All-to-All Traffic

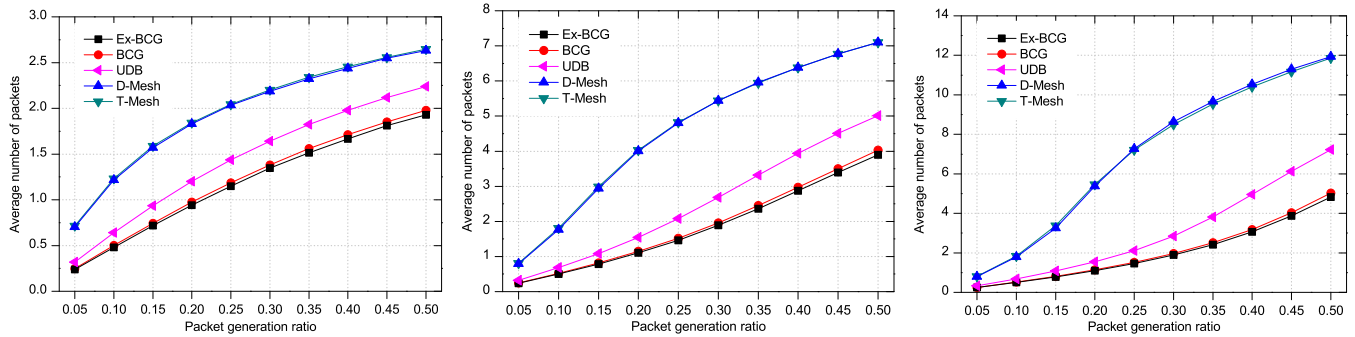
Fig. 4.7 shows simulation results for All-to-All traffic pattern. Metrics shown are reachability, ETE delay and mean number of packets per node. From the simulation, we found that the *Ex*-BCG with the CVT routing algorithm has best performance when compared to the benchmark networks. In Fig. 4.7(a), the reachabilit of the networks decrease as the packet generation ratio increases. However, the reachability of the *Ex*-BCG remains higher than that of the Meshes and UDB networks independent of the packet generation ratios and the buffers sizes. In Fig. 4.7(b), the *Ex*-BCG with the CVT routing algorithm maintains the smallest ETE delay. With a buffer size 8 packets, the ETE delays of for the D-Mesh and T-Mesh networks decrease as a function of the packet generation ratio whereas those of larger buffer sizes increase. This is because the ability that the networks can handle the packets reaches saturation with a packet generation of rate 0.05 (low reachability). From the mean number of packets per node (See Fig. 4.7(c)), we observe that the *Ex*-BCG with the CVT routing algorithm has the least number of packets as a function of the packet generation rates. In comparison with the BCG, although the *Ex*-BCG has almost the same performance as that of BCG in all the metrics, the *Ex*-BCG requires node to have 506×4 routing entries while the BCG needs 1010×4 routing entries with the vertex-transitive routing algorithm. Therefore, with consideration to routing complexity, *Ex*-BCG with the CVT routing algorithm is more efficient than BCG.



(a) Reachability: Results with buffer sizes 8, 16 and 24 from the left

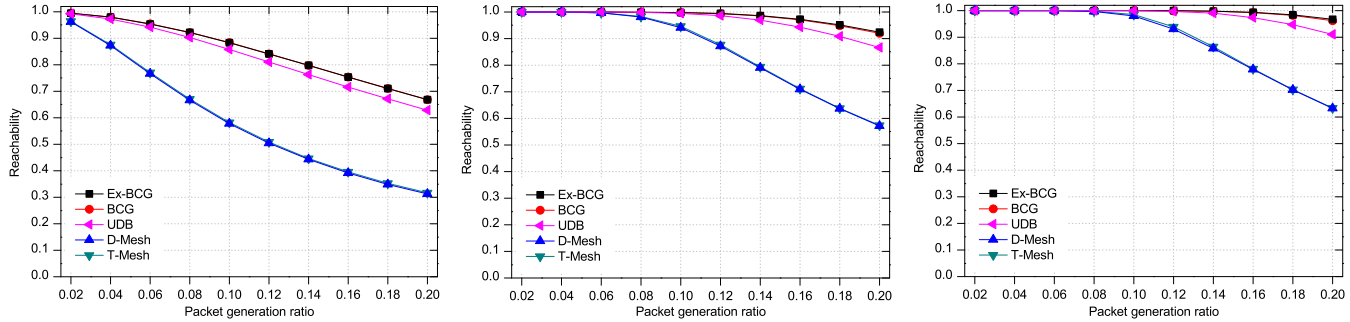


(b) End-to-End (ETE) delay: Results with buffer sizes 8, 16 and 24 from the left

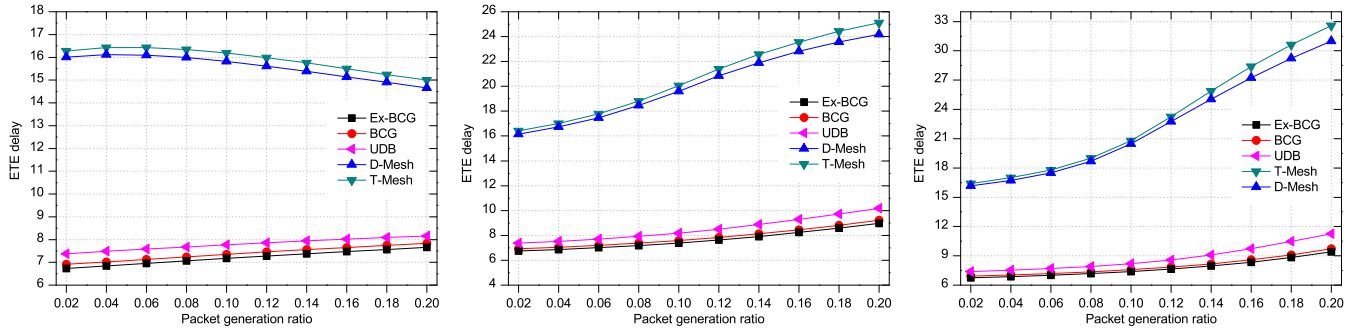


(c) Average number of packets at node: Results with buffer sizes 8, 16 and 24 from the left

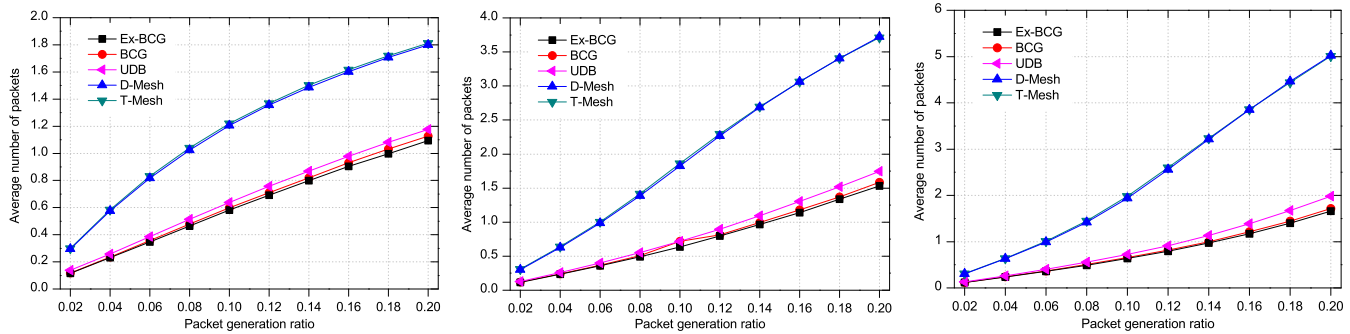
Figure 4.7: Simulation results for the *All-to-All* traffic pattern



(a) Reachability: Results with buffer sizes 8, 16 and 24 from the left



(b) End-to-End (ETE) delay: Results with buffer sizes 8, 16 and 24 from the left



(c) Average number of packets at node: Results with buffer sizes 8, 16 and 24 from the left

Figure 4.8: Simulation results for the *All-to-100* traffic pattern

All-to-M Traffic

Fig. 4.8 shows simulation results for All-to-All traffic pattern. Metrics shown are reachability, ETE delay and mean number of packets per node. In the result, the *Ex*-BCG with the CVT routing algorithm is most tolerant to the concentrated traffic among the networks. In terms of reachability in Fig. 4.8(a), the reachabilities of the networks become less as the packet generation rate increases. However, the decreasing rate of *Ex*-BCG is lower than those of the benchmark networks, and in particular, the *Ex*-BCG with the buffer size 24 shows more than 95% reachability along the packet generation rate. Moreover, ETE delay of *Ex*-BCG also remains as lowest regardless of the buffer sizes and packet generation rates as shown in Fig. 4.8(b). For investigation of the routing capacity and network structure, the least mean number of packets at node of the *Ex*-BCG with the CVT routing algorithm is an evidence of the robust tolerance to the concentrated traffic (See Fig. 4.8(c)).

4.2 Aggressive Multi-path Aware Routing Protocol

4.2.1 Problem Statement

In this section, we present problem statement that the VT and CVT routing protocols introduce with the resized *Ex*-BCGs and BCGs. Throughout the paper, we consider the unfolded routing table of the CVT routing protocol for simplicity ($i.e.$, one element of each routing entry has only one bit for relaying direction [78]).

Single Shortest Path Problem

The VT and CVT routing protocols provide shortest (optimal) path(s) by looking up the BCGs and *Ex*-BCGs routing tables. Due to the fact that those routing protocols deal with only optimal paths, there may be a unique shortest path between given source and destination. If there is only a single path available in the routing table, resizing with the CTR algorithm can trigger a routing failure.

Fig. 4.9 shows an example of routing failure when there is only a single path between src and dst and one of the intermediate nodes is pruned for resizing. Originally, the path between src and dst is $src \rightarrow i \rightarrow j \rightarrow dst$ (single shortest path). However, after resizing ($i.e.$, pruning j and performing the CTR operation between i and k), i is rewired to k, and then i becomes incapable of routing to dst.

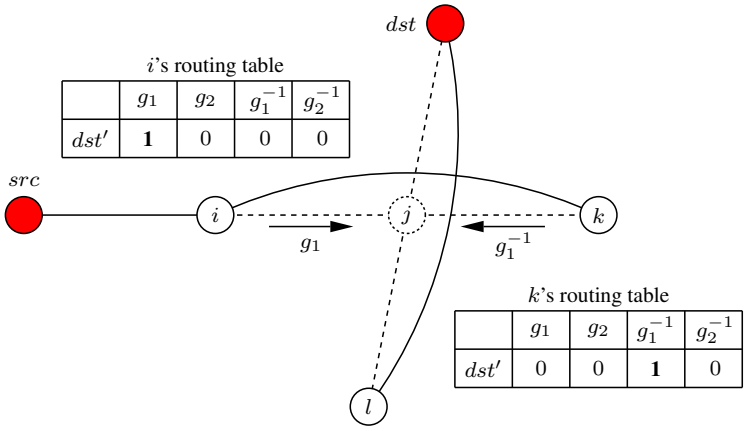


Figure 4.9: Example of routing failure by single shortest path problem. Note that dst' represents a corresponding node to dst at the node and is obtained from Tab 2.2 and Eq (4.1).

Specifically, i looks up its routing table and relays a packet to k in g_1 direction, and then k returns the packet to i in g_1^{-1} direction by following its routing table. Consequently, i and k repeat relaying the packet to each other (looping effect).

Ex-BCGs						
Network size	p	k	a	n	Generator set	Single shortest paths
1012 (253-4)	23	11	2	4	$(3, 7)$	$325,956$ (31.9%)
1012 (506-2)	23	22	5	2	$(3, 7)$	$227,608$ (22.2%)

BCGs					
Network size	p	k	a	Generator set	Single shortest paths
1010	101	10	6	$(3, 7)$	$205,030$ (20.1%)
1081	47	23	2	$(3, 7)$	$410,780$ (35.2%)

Table 4.7: The number of single shortest paths

In Tab. 4.7, we analyze the number of single shortest paths and its ratio to all routing paths. More than 20% of total paths are single shortest paths, which means that if the CTR operation is performed along those paths, routing is likely to fail.

Multi-path Depletion Problem

Multiple shortest routing paths between source and destination nodes are advantageous properties by providing alternative paths when there is a node failure or a bottleneck along the path. The VT and CVT routing protocols allow multiple shortest paths from the routing table. However, when a large amount of resizing is performed with CTR, those multi-paths may be depleted and routing failures increase. To resolve these problems, the routing table update scheme was introduced in [66]. It enables nodes rewired by CTR to inform new neighbors of routing table updates by sending Backward Advertisement (BA) packets. However, this scheme produces significant overhead when the BA packet propagates multi-hops and each node must have two hop neighbor information. Moreover, this routing table update scheme is unable to route when an intermediary node along a single shortest path fails as shown in Fig. 4.9.

4.2.2 Aggressive Multi-path Aware Routing

In this section, we propose the Aggressive Multi-path Aware (AMA) routing protocol for reliable and efficient routing in resized *Ex*-BCGs and BCGs while resolving the problems stated in the previous section. The AMA routing protocol is based on updated routing table that reflects the topology changes. We first introduce how to update the routing table and then present the detail of AMA routing operation.

4.2.3 Routing Table Update

Control packets

Our routing table update scheme aims at reflecting topology changes in the routing table with a small network overhead. As illustrated in Fig. 4.10, we use two types of control packets to update the routing table.

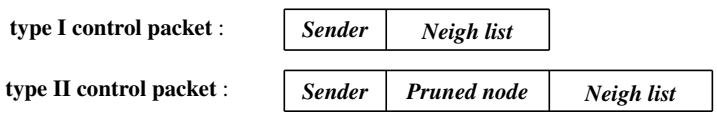


Figure 4.10: Control packets

The type I control packet is used by nodes that lose neighbors due to pruning and are rewired by the CTR algorithm. It allows nodes that obtain a new neighbor by the CTR operation to inform its new neighbor of the node ID (***Sender*** field) and its neighbors' node IDs (***Neigh list*** field).

The type II control packet is propagated to neighbors of the newly rewired node in order to advertise routing infeasibility to the pruned node (***Pruned node*** field) and routing availability for the newly rewired neighbor and its neighbors (***Neigh list*** field).

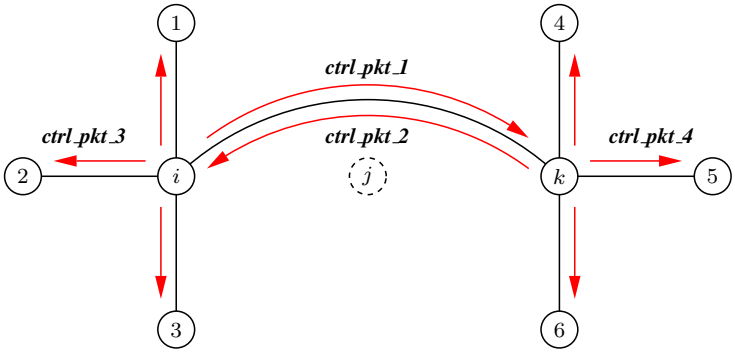
4.2.4 Control packet propagation

Once the CTR operation is performed between nodes, the control packets defined in Fig. 4.10 are propagated. Fig. 4.11 shows an example of how the control packets are propagated and how the routing tables are updated. In Fig. 4.11(a), once node j is pruned and i and k are rewired, i and k exchange type I control packets (***ctrl_pkt_1*** and ***ctrl_pkt_2***) to inform a new neighbor of node IDs of its current one hop neighbors (See Fig. 4.11(b)). Based on the received control packets, i and k update their routing tables as follows: Note that g_j represents a direction to node j from the node.

- Identify the direction g_j to the pruned node j.
- Change the routing entry j' corresponding to j in g_j to 0. j' is obtained from Tab 4.7 and Eq (4.1).
- Change all routing entries with 1 in g_j to 2.
- Change the routing entries corresponding to node IDs of ***Neigh list*** field in g_j to 1.

Since j is removed, routing to nodes that are reachable through j before its removal may become infeasible. In other words, the routing entries in g' direction, which originally have 1 before resizing, may route the packet to the wrong path if the packet travels through the g' direction. Therefore, we set the routing entries with 1 in g' direction to 2, which means *less prioritized* option than neighbor(s) of 1.

Then, based on the received control packets, i and k send type II control packets ***ctrl_pkt_3*** and ***ctrl_pkt_4*** to their neighbors, respectively. By receiving these control packets, neighbors update their routing tables as follows.



(a) Control packets propagation

ctrl_pkt_1 :	i	$1, 2, 3$	
ctrl_pkt_2 :	k	$4, 5, 6$	
ctrl_pkt_3 :	i	j	$k, 4, 5, 6$
ctrl_pkt_4 :	k	j	$i, 1, 2, 3$

(b) Control packets

i's table

	g_j
⋮	$1 \rightarrow \mathbf{2}$
⋮	0
j'	$1 \rightarrow \mathbf{0}$
k'	$1 \rightarrow \mathbf{1}$
$4'$	$0 \rightarrow \mathbf{1}$
$5'$	$0 \rightarrow \mathbf{1}$
$6'$	$1 \rightarrow \mathbf{1}$

k's table

	g_j
⋮	$1 \rightarrow \mathbf{2}$
⋮	0
j'	$1 \rightarrow \mathbf{0}$
i'	$1 \rightarrow \mathbf{1}$
$1'$	$0 \rightarrow \mathbf{1}$
$2'$	$0 \rightarrow \mathbf{1}$
$3'$	$1 \rightarrow \mathbf{1}$

$1, 2, 3$'s table

	g_i
⋮	1
⋮	0
j'	$1 \rightarrow \mathbf{0}$
k'	$1 \rightarrow \mathbf{1}$
$4'$	$0 \rightarrow \mathbf{1}$
$5'$	$1 \rightarrow \mathbf{1}$
$6'$	$1 \rightarrow \mathbf{1}$

$4, 5, 6$'s table

	g_k
⋮	1
⋮	0
j'	$1 \rightarrow \mathbf{0}$
i'	$1 \rightarrow \mathbf{1}$
$1'$	$0 \rightarrow \mathbf{1}$
$2'$	$0 \rightarrow \mathbf{1}$
$3'$	$1 \rightarrow \mathbf{1}$

(c) Update the routing tables: Note that n' in the first column represents a corresponding node to n at a current node.

Figure 4.11: Routing Table Update

- Identify the direction ($i.e.$, g_i for nodes $1, 2, 3$ and g_k for nodes $4, 5, 6$) to the ***Sender*** field.
- Change the routing entry j' corresponding to j in the found direction to 0.
- Change the routing entries corresponding to node IDs of ***Neigh list*** field in the direction to 1.

Fig. 4.11(c) illustrates how to update the routing tables of nodes i, k and $1 - 6$ based on the exchanged control packets. When propagating the control packets, we limit the number of hops to 2 in order to minimize control packet overhead.

4.2.5 AMA Routing Operation

Starting from the updated routing table, we will discuss the AMA routing operation in detail. The AMA routing operation is composed of two schemes: 1) Multi-path Aware routing and 2) Random Direction routing. These two schemes rely on a data packet header defined in the following section.

Data packet header

For the AMA routing operation, the data packet header has 5 fields as follows:

Src	***Dst***	***Path***	***Parent***	***Hop***	***Number of multi-paths***	***Return flag***

Figure 4.12: Data packet header

- ***Src*** & ***Dst*** : source and destination node IDs.
- ***Path***: a set of node IDs that the packet traversed.
- ***Parent***: a node ID to return the packet when there is no available neighbor for relaying.
- ***Hop***: the number of hops the packet traversed (optional).
- ***Number of multi-paths***: the number of available multi-paths that the packet has.

- ***Return flag***: indication of whether the packet is returning or not.

In the following sections, we describe how the header fields are used with the two routing schemes. Note that we omit ***Src*** and ***Dst*** fields in the following figures for simplicity.

Multi-path Aware routing

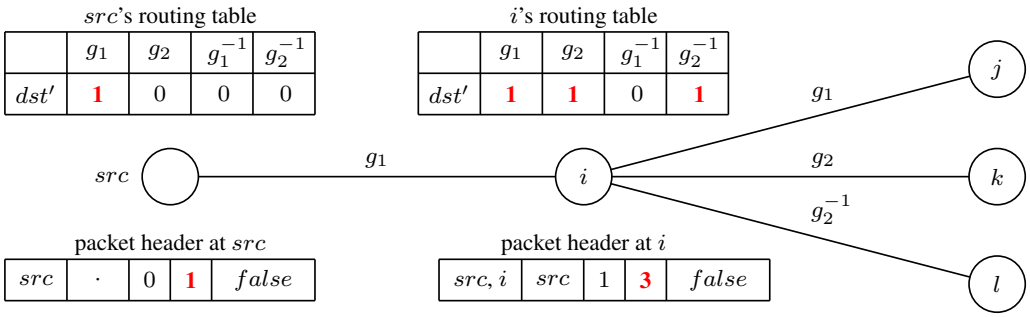


Figure 4.13: Counting the number of multi-paths

The multi-path aware routing aims at exploiting all possible multi-paths which are obtained from the updated routing table. In the multi-path aware routing, the packet counts the number of multi-paths from the routing table in propagation to the destination. Counting the number of multi-paths while routing is performed as illustrated in Fig. 4.13. Note that dst' is a corresponding node to dst at the current node. For example, if dst is 10 in *Ex*-BCG with parameters $p = 7, k = 3, a = 2$ and $n = 2$, dst' is $7 = 10 - (3 - 0)$ at node 3 from Eq (4.1). At src, there is only one path along g_1 direction to dst. Thus, the ***Number of multi-paths*** field is 1. When i receives the packet, it identifies from the routing table that there are 3 possible directions g_1, g_2 and g_2^{-1}. Then i updates the ***Number of multi-paths*** field to 3.

When routing with the routing table and the *relaying node* has no available direction to a *next node*, the relaying node returns the packet to its *parent node* which may have alternative path. Fig. 4.14 shows an example of when and how to return the packet to the parent node. In Fig. 4.14(a), i sends the packet to j by following the routing table and j finds out that its neighbor k has been pruned,

thus routing is no longer feasible. However, the ***Number of multi-paths*** field of the packet indicates there is an alternative path. Then, j returns the packet to its parent i while changing the ***Return flag*** field to *true* and reduces the ***Number of multi-paths*** to 1. When the packet is returned to i, i searches for an alternative path ($i.e.$, g_2 direction) and sends the packet to m while changing the ***Return flag*** field to *false* as shown in Fig. 4.14(b). This multi-path aware routing operation continues until the packet arrives at destination (routing success) or the following conditions are satisfied:

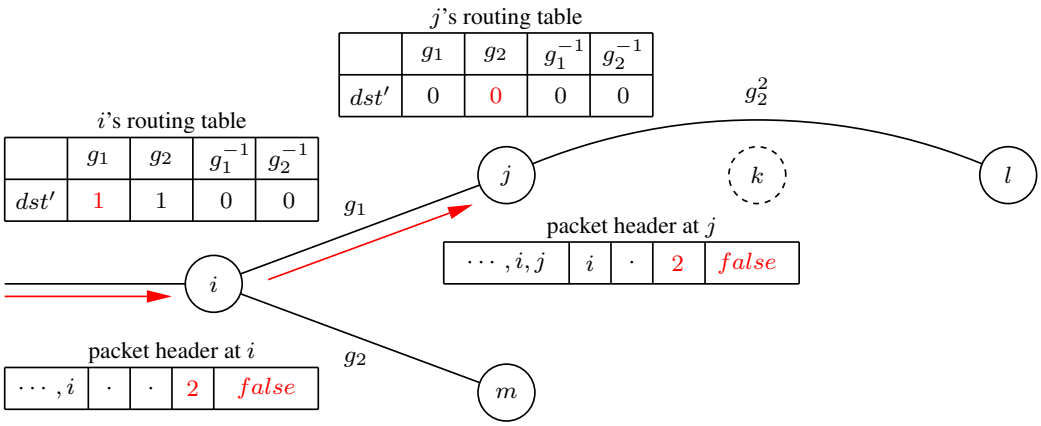
- The ***Number of multi-paths*** field is 1 ($i.e.$, no more alternative path).
- The routing entry corresponding to destination at the current node does not have 1 or 2.

Random Direction Routing

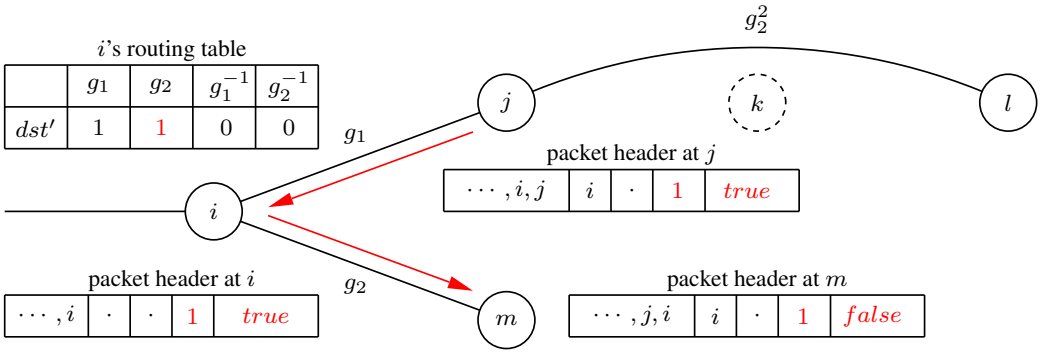
During the multi-path aware routing operation, there may be a situation that there is no more multi-paths ($i.e.$, ***Number of multi-paths*** $= 1$) and the current relaying node has no available neighbor for relaying. This is likely to occur when a large number of nodes are pruned for resizing since the routing table update is unable to fully reflect the topology changes.

Fig. 4.15 provides an example of when i finds out that it has no neighbor for relaying and the packet indicates no more available multi-path. In this case, the AMA routing protocol ***randomly*** selects the next relaying node other than the traversed neighbors identified from the ***Path*** field. If the ***Path*** field does not contain k and l, i randomly selects one of them. This random selection routing is in pursuit of resolving the problems pointed out in Section 4.2.1 by *providing alternative path* to continue the routing rather than dropping the packet, when the multi-path aware routing operation fails. Therefore, if the selected node (either node k or l) has 1 or 2 at the corresponding routing entry in its routing table, the node restarts the multi-path aware routing operation by updating the packet header fields. Otherwise, it performs the random direction routing again. This process of finding alternative path is iterated until the conditions of the routing protocol failure defined in the following section are met.

There are other options in selecting the next relaying node such as the number of packets in buffer or link state. In this paper, however, we focus on a graph theoretic routing protocol which can be further modified for specific network applications. Moreover, the random selection is a simple and efficient option to



(a) When a packet arrives at j, there is no available path due to rewiring. Since the *Number of multi-path* field is 2, the packet is returned to its parent while changing the *Return flag* field to *true*.



(b) The packet is returned to its parent i which has another optimal path. Note that when the parent node has no path, returning the packet continues with updating the parent node by lookup the *Path* field. i relays the packet to m in the direction of g_2 while changing the *Return flag* field to *false*.

Figure 4.14: Packet propagation in multi-path aware routing

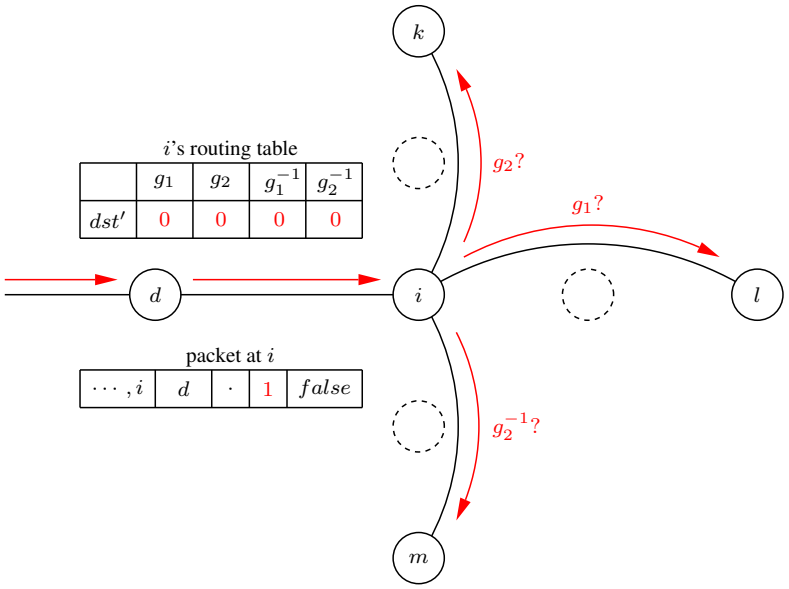


Figure 4.15: Example of Multi-path Aware Routing failure

implement when there are a large number of possible neighbors. We tried other heuristic approaches, but the results are similar to that of the random selection (We do not show the other approach's result due to space limitation).

4.2.6 Definition of Routing Failure

In the AMA routing protocol, we define the conditions of routing protocol failure as follows:

- The ***Number of multi-paths*** field is 1.
- All neighbors of the current node are traversed (identified from the ***Path*** field).
- The routing entry corresponding to destination at the current node does not have 1 or 2.

The above routing protocol failure conditions are defined at graph level considerations. In addition to those conditions, depending on the network applications, the routing failure can also be defined with other conditions such as Time-to-Live (TTL), routing table invalidation and buffer overflow *etc*. Based on the discussion so far, Fig. 4.16 shows a flow chart of the AMA routing protocol.

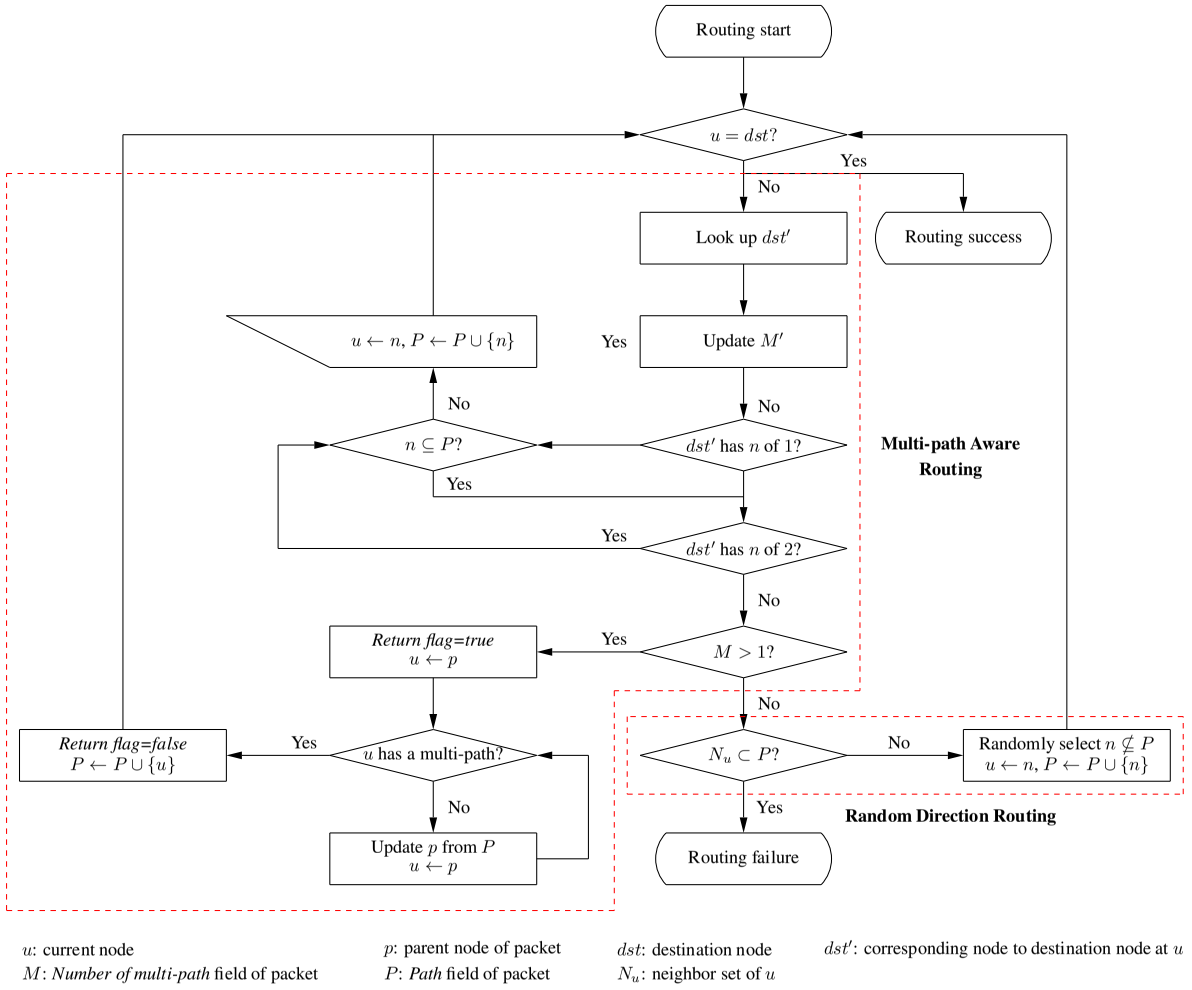


Figure 4.16: AMA routing protocol flow chart

4.2.7 Performance Evaluation

In this section, we use a simulation model to examine the AMA routing protocol for resized BCGs and *Ex*-BCGs with and without network traffic.

Network Model

In the simulation, we assume that the BCGs and *Ex*-BCGs are resized to an arbitrary size network before running the simulation (static network). Thus, there is no pruning nodes or CTR operation during routing. We also assume that the routing table of each node is updated.

We evaluate the AMA routing protocol with two network scenarios. First of all, we examine the AMA routing protocol without traffic and buffer consideration. In other words, there is no dropped packet by overflow and no delay since only one packet is routed at one time. This scenario aims at investigating the routing feasibility of AMA routing protocol in the resized BCGs and *Ex*-BCGs.

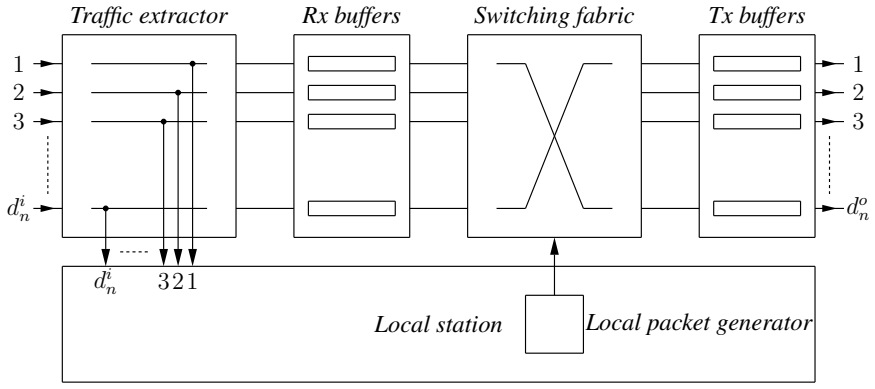


Figure 4.17: Node model

Secondly, we adopt more practical network scenario with traffic and buffer consideration. For this simulation, we consider the network model of [78]. Each network consists of nodes as illustrated in Fig. 4.17 connected with zero delay links. The node has a traffic extractor, d_n^i receiving Rx buffers (per incoming link), a switching fabric and d_n^o transmitting Tx buffers (one per outgoing link). The traffic extractor diverts incoming packets arriving at the destination to the local station. The switching fabric moves packets in Rx buffers to Tx buffers using the routing protocol. We employ a finite size buffer for Rx and Tx buffers ranging from 1 to 2. Since we consider degree 4 networks in the simulation, the

sums of Tx and Rx buffer sizes are 8 and 16 per node, respectively. The local packet generator creates packets following the *Bernoulli* process parameterized by the packet generation rate. In the simulation, we assume that time is slotted and synchronized. This allows nodes to receive and transmit packets at the same time. Each time slot has two phases: a packet switching phase (switching packets from Rx buffers to Tx buffers) and a packet transmitting phase (sending packets from Tx buffers to Rx buffers). At each time slot, the local station can receive at most d_n packets. When the Rx buffer has no space to accept it, a newly incoming packet is dropped (overflow). In switching packets from Rx buffer to Tx buffer, if the selected Tx buffer has no space, the switching fabric seeks another Tx buffer (alternative shortest path). When all designated Tx buffers are full, the packet is not dropped, but stays in the Rx buffer.

Simulation Setup

In the simulation, we target BCGs of $1,010$ nodes and $1,081$ nodes and *Ex*-BCGs of $1,012$ nodes expanded from 253-node and 506-node BCGs, and reduce the network size by $0\%, 5\%, \cdots, 40\%$. For each size reduction, we obtain the average of the metrics collected from 10 network samples by randomly selecting the removed nodes for each sample. Tab. 4.8 shows the network generating parameters.

Ex-BCGs						
Network size	p	k	a	n	Generator set	Removed node percentage
1012 (253-4)	23	11	2	4	$(3,7)$	$0\%, 5\%, \cdots, 40\%$
1012 (506-2)	23	22	5	2	$(3,7)$	$0\%, 5\%, \cdots, 40\%$

BCGs					
Network size	p	k	a	Generator set	Removed node percentage
1010	101	10	6	$(3,7)$	$0\%, 5\%, \cdots, 40\%$
1081	47	23	2	$(3,7)$	$0\%, 5\%, \cdots, 40\%$

Table 4.8: Network generating parameters

For the simulation with network traffic pattern, we utilize *All-to-All* traffic pattern with different packet generation probabilities. We assume that a source uniformly and randomly selects a destination among all nodes with a given packet generation rate. For each size reduction, we run a simulation for $5,000$ time slots

and obtain the average of the metrics collected from 10 network samples by randomly selecting the removed nodes for each sample. Tab. 4.9 shows the detail of the network traffic pattern, buffer size and packet generation rate.

Traffic pattern	Buffer size	Packet generation rates
All-to-All	$8, 16$	$0.05, 0.1$

Table 4.9: Network traffic parameters

Metrics

For evaluation, we utilize the following metrics:

- **Reachability** The reachability is used and to evaluate the *reliability* of the routing protocol. We define the reachability as follows:

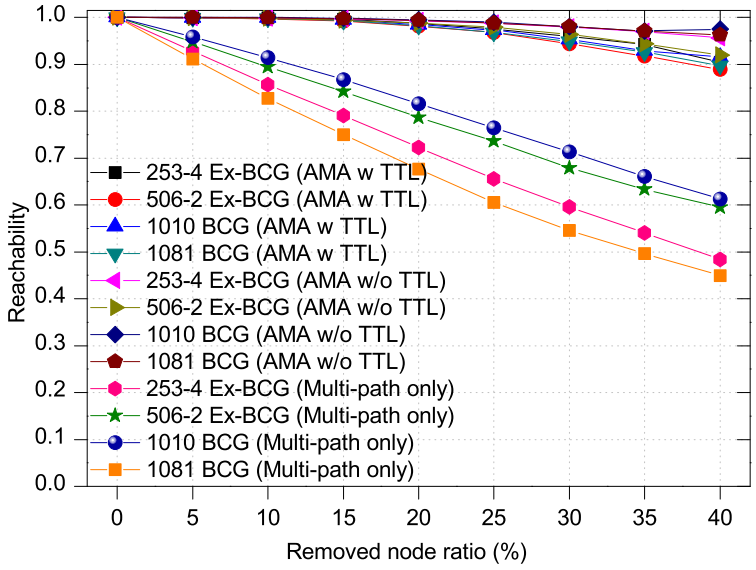
$$\textit{Reachability} = \frac{\textit{Number of arrived packets}}{\textit{Number of generated packets}} \tag{4.7}$$

- **Average routing path length**

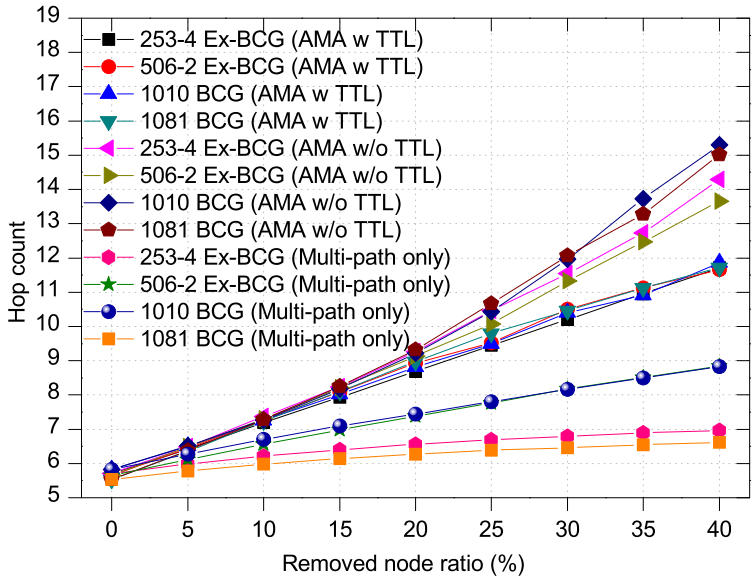
  The average routing path length is the average hop count between all possible pairs of source/destination nodes allowed by the routing protocol. That metric evaluates the routing *efficiency* of the network. In the average routing path length evaluation, we set TTL (the maximum number of hops) to $diameter \times 4$.

- **End-to-End Delay**

  The End-to-End (ETE) delay is an important metric to evaluate the *efficiency* of the routing protocol when network traffic pattern is applied. Obviously, the smaller the ETE delay the more efficient the routing ability. In the ETE delay evaluation, we set TTL (the maximum number of time slots) to $diameter \times 4$ so as to prevent packets from floating the network.



(a) Reachability



(b) Average routing path length

Figure 4.18: Result without traffic and buffer consideration

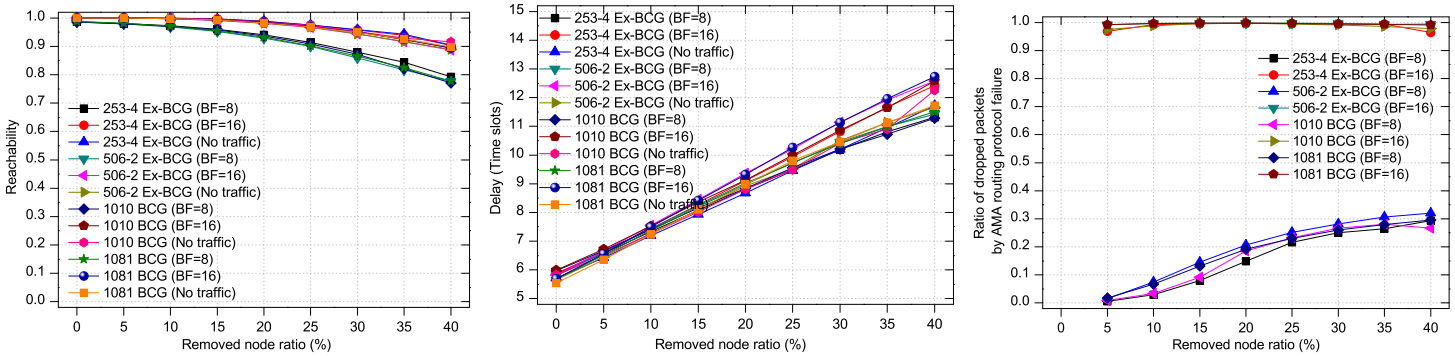
Simulation result without traffic and buffer consideration

Fig. 4.18 shows the reachability and average routing path length of the AMA routing protocol with and without TTL in the target networks. For comparison, we simulated the target networks with only *Multi-path aware* routing scheme discussed in Section 4.2.5. As pointed out in Section 4.2.1, the reachabilities from *Multi-path aware* routing show faster decreasing than those of AMA routing protocol along the removed node ratios. Especially, the $1,081$ BCG shows the lowest reachability among target networks (the highest ratio of single shortest paths from Tab. 4.7). However, we found that the reachabilities from the AMA routing protocol with/without TTL are more than 90% independent of the target networks along the removed node ratios (Reliability). These high reachabilities are achieved by effectively providing alternative routing paths from the random direction routing operation.

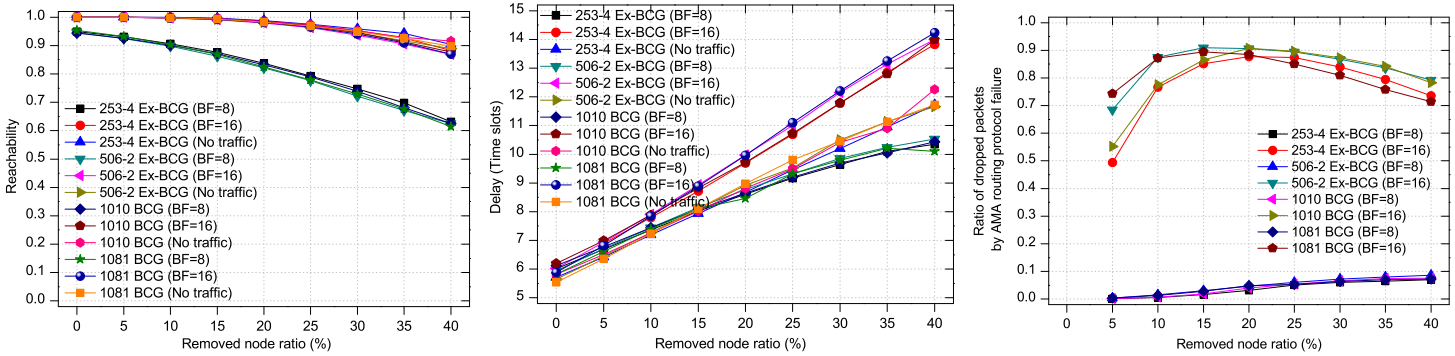
In evaluation of the average routing path length as shown in Fig. 4.18(b), the hop counts from the results increase as more nodes are removed. This is because a large amount of node removal triggers disfunctions in the routing table and results in random direction routing. However, the increasing rate of hop counts is maintained low (Efficiency). Specifically, the hop count grows by less then 1 hop per additional 5% of nodes removed. Although the routing path lengths from the multi-path aware routing algorithm show smaller hop counts when compared to those of the AMA routing protocol, their low reachabilities highlight its routing limitation.

Simulation Results for the All-to-All traffic pattern

Fig. 4.19 summarizes reachability, End-to-End delay and dropped packets results for the All-to-All traffic pattern with packet generation rates of $0.05, 0.1$ and buffers of size 8 and 16. From the reachability results, we observe that although the removed node ratio increases, the reachabilities of the target networks remain as higher than 90% with buffer size 16 when considered packet generation rates are 0.05 and 0.1 (Reliability). Specifically, the reachabilities of buffer size 16 are almost identical to that without traffic pattern. In other words, with buffer size 16, the AMA routing protocol shows almost optimal performance when the packet generation rate is less than 0.1. With buffer size 8, the reachabilities of packet generation rates $0.05, 0.1$ shrink to 77% and 62.5%, respectively. In the simulation with a buffer of size 8, most of the routing failures are caused by insufficient buffer length to cope with the incoming packets.



(a) Packet generation probability $= 0.05$: Reachability, End-to-End delay and Dropped packets analysis from the left



(b) Packet generation probability $= 0.1$: Reachability, End-to-End delay and Dropped packets analysis from the left

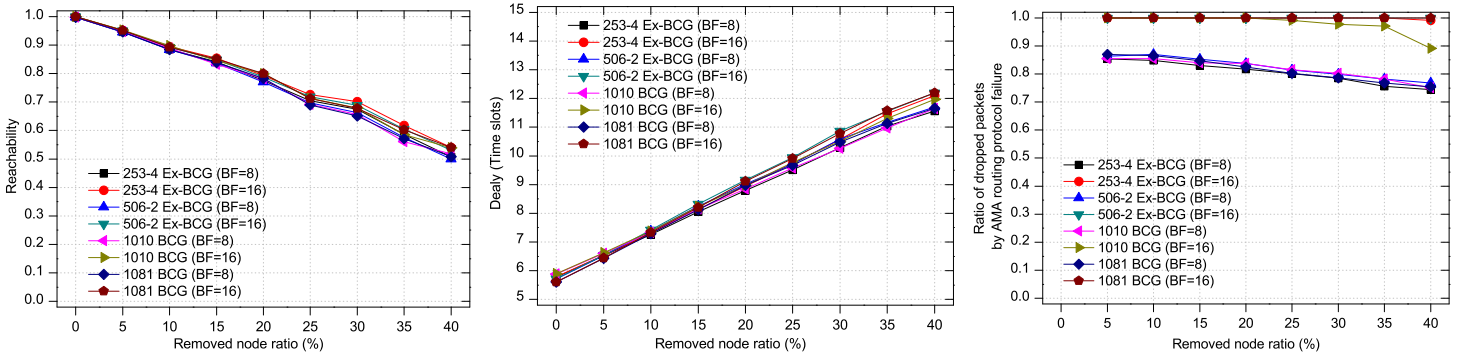
Figure 4.19: Results for the All-to-All traffic patterns and finite buffer sizes (In the dropped packets analysis, we omit the results when no nodes are removed. The *No traffic* results correspond to the case of no traffic and no buffer.)

In ETE delay result, we identified that the ETE delays of the target networks increase as the removed node ratio grows regardless of packet generation rates and buffer sizes. However, the ETE delay with buffer size 16 and packet generation rat 0.05 shows slight difference from the average routing path length (Efficiency). Although the ETE delays of buffer size 16 with packet generation rates $0.05, 0.1$ have increasing delay as the removed node ratios increment, their reliable reachabilities (higher than 90% for packet generation rate 0.05 and 85% for packet generation rate 0.1) represent that the AMA routing protocol efficiently deliver the network traffic by using multi-paths and random direction routing. From the simulation result, we found that the AMA routing protocol achieves almost identical reachabilities and average routing path lengths in the target networks along the removed node ratios, although their generating parameters are different.

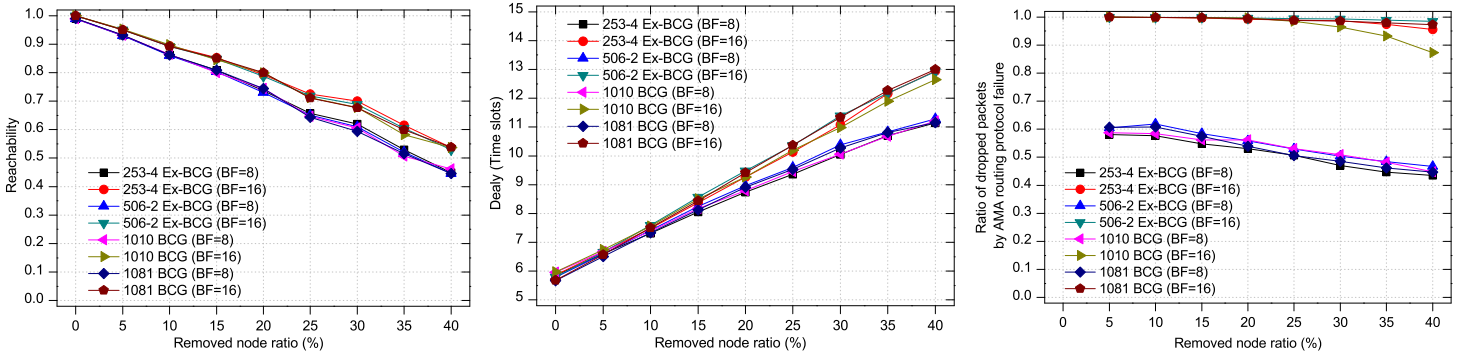
We also analyze the dropped packets to investigate the routing failure. In the simulation, we categorize the causes of routing failure into 3 categories: 1) the routing failure conditions defined in Section 4.2.6, 2) exceeding Time-To-Live (TTL) in End-To-End (ETE) delay and 3) buffer overflow. We regard the routing failure conditions and TTL as AMA routing protocol failures and show the rate of dropped packets from those factors. From the simulation results, we observed that with buffer size 16 and packet generation rate 0.05, the routing failure is mostly triggered by the AMA routing protocol failure. However, with buffer size 8, more than 70% of dropped packets in routing are due to the buffer overflow. This shows that the AMA routing protocol with buffer size 16 performs its optimal routing ability at packet generation rate 0.05. With packet generation rate 0.1, the routing failure from the AMA routing protocol failure is slightly degraded, since buffer of size 16 is not sufficient to cope with all incoming packets.

Simulation Results for the All-to-0.25 traffic pattern

In Fig. 4.20, we show the simulation results for reachability, ETE delay and dropped packets analysis of All-to-0.25 traffic pattern ($i.e.$, Destination nodes are randomly selected from 25% of network nodes) with packet generation rates $0.02, 0.04$ and buffer sizes $8, 16$. Note that we do not include the simulation results for reachability and average routing path length without traffic, since the All-to-0.25 traffic pattern has limited destination nodes. From the reachability results, as the removed node ratio increases, the reachabilities with the considered packet generation rates decrease independent of buffer size. This is because the concentrated traffic forwarded to the small number of destinations triggers a bottleneck along the routing path. However, we found that even if the packet generation



(a) Packet generation probability $= 0.02$: Reachability, End-to-End delay and Dropped packets analysis from the left



(b) Packet generation probability $= 0.04$: Reachability, End-to-End delay and Dropped packets analysis from the left

Figure 4.20: Simulation result of All-to-0.25 traffic: In dropped packets analysis, we omit the result of removed node ratio 0%.

rate is doubled from 0.02 to 0.04, the reachability reduces slightly. Especially, with buffer size 16, the reachability for packet generation rates 0.02 and 0.04 have almost identical value.

From the ETE delay simulation results, we observed that the ETE delay grows as the removed node ratio increases for both packet generation rates independent of buffer sizes. Although the ETE delay of both packet generation rates are maintained low, the relatively low reachabilities represent load balancing limitations of the AMA routing protocol. However, we aim at designing a general routing scheme for the resized BCGs and *Ex*-BCGs rather than routing with specific traffic pattern.

In the analysis of dropped packets, we found that most dropped packets with buffer size 16 are caused by the AMA routing failure independent of packet generation rates. This is because nodes along the routing path have insufficient buffer size to receive all the incoming packets. On the contrary, the simulation result with buffer size 8 shows that the rate of dropped packets by the AMA routing protocol failure decreases as the packet generation rate increases, which implies larger number of the dropped packets occurred due to the buffer overflow.

4.3 Discussion

In this chapter, we presented the CVT and AMA routing protocols for *Ex*-BCGs and the resized *Ex*-BCGs. The CVT routing protocol utilized the class-level vertex transitivity and GCR representation for optimal and distributed routing. The proposed routing table folding and merging schemes simplified routing table generation and lookup processes. Furthermore, we provided the AMA routing protocol to allow for routing in resized *Ex*-BCGs by exploiting available multi-paths from the updated routing table and using random direction routing when no more options are available. The simulation result with and without traffic patterns showed that the AMA routing protocol has reliable reachability and short average path length.

Chapter 5

Ex-BCGs based Communication Topology Construction and Routing Protocol for WSNs

In this chapter, we propose the *Ex*-BCG Topology Construction (EBTC) algorithm to formulate an *Ex*-BCG based communication topology for WSNs that eliminates packet collision and saves energy. Based on the communication topology from the EBTC algorithm, we devise the clustering based routing algorithm for reliable and efficient data reporting in event-driven WSNs.

5.1 EBTC: *Ex*-BCG Topology Construction

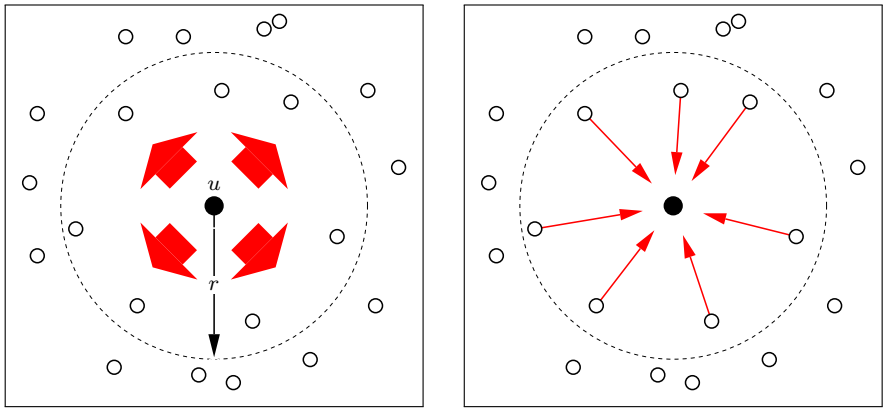
5.1.1 Problem Statement

In this section, we present the challenges associated with neighbor discovery and topology control algorithm design.

Collision in neighbor discovery

Most known distributed topology control algorithms are based on local information obtained from physical neighbors. However, if nodes are randomly deployed in a target area, it becomes difficult for a node to predict how many physical neighbors reside in its transmission range.

In Fig. 5.1(a), node u broadcasts a request for neighbor discovery that results in response from its physical neighbors. However, these replies may trigger a



(a) u broadcasts a request for neighbor discovery.

(b) Replies to u result in collision.

Figure 5.1: Collision problem in neighbor discovery.

significant amount of collisions, where there are not enough time gaps between replies (See Fig. 5.1(b)). Even if the physical neighbors reply with random delay, collisions are inevitable in dense WSNs. As a result of collision during neighbor discovery, nodes are unable to obtain intact local information, which leads to an incomplete topology construction such as non-symmetric links or disconnected communication topologies.

Selection of logical neighbors

In designing a topology control algorithm, how to select logical neighbors out of physical neighbors is a key factor in optimizing WSN energy consumption. In [79], we found that *Ex*-BCG is a promising topology with efficient topological properties and robust connectivity. However, its beneficial properties are based on the assumption that a node can communicate with any node in the network, which is not always practical in wireless networks. To construct a communication topology based on *Ex*-BCG in WSNs while accomplishing the objectives: 1) robust connectivity, 2) energy efficiency and 3) short hop counts between logically connected nodes; a new approach that considers limited transmission range and constrained energy support is required. In the next section, we design a topology construction algorithm to resolve these problems.

5.1.2 Design objectives

The EBTC algorithm operates in a *distributed manner* where a node establishes logical connections based *only on one hop neighbor information* collected from its physical neighbors. Our EBTC aims at achieving the following objectives:

- Avoid collision and save energy consumption with a deterministic and selective node wake-up schedule.
- Establish a reliable bidirectional link in topology construction.
- Short average hop counts in resultant communication topology.
- Constrain a logical node degree as low.

The existing neighbor discovery algorithms [22–25] are unable to find physical neighbors without collision when nodes are randomly deployed in a target area. Differing from other neighbor discovery algorithms, however, nodes under the EBTC algorithm search for *only qualified physical neighbors rather than all physical neighbors*. In other words, our EBTC is not just a combined work of neighbor discovery and topology control but an integrated work of those two processes.

5.1.3 Network model

For network modeling, we assume the following:

- Before deployment, nodes are assigned a unique integer node ID ranging from 0 to $N-1$, (where N is the total number of nodes). The EBTC operation is performed starting from node 0 to node $N-1$. For its sequential operation, we assume time is slotted and that nodes have a synchronized timer. In practice, clock drift among timers is a non-trivial problem that may trigger false interactions between nodes. However, we assume that each time slot is significantly longer than the clock drift.
- Nodes are equipped with an RSSI (Received Signal Strength Indication) detector. Initially nodes are assigned identical transmission power, but once the EBTC operation is completed, nodes adjust their transmission power to cover their farthest logical neighbor.

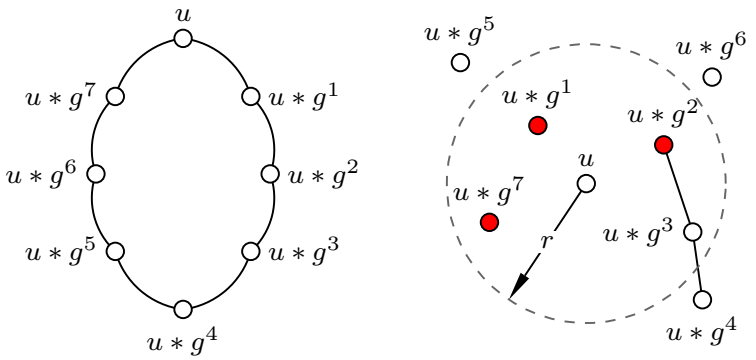
- Nodes are randomly and uniformly distributed in a target area and are not mobile.
- Nodes do not have a hardware like GPS. Thus they are unaware of the geographic location of other nodes.

Next, we present the EBTC algorithm in the following sections. The EBTC algorithm consists of two phases: I) Logical Neighbor Candidates Discovery and II) Logical Neighbor Selection.

5.1.4 Phase I: Logical Neighbor Candidates Discovery

In Phase I, each node collects its one hop neighbor information to find its logical neighbor candidates, where a logical neighbor candidate is defined as follows:

Definition 7. The set of C_u of logical neighbor candidates of a node u contains nodes defined as physical neighbors iff they belong to a cycle rooted at u with generator g in *Ex*-BCG and have less than 2 logical neighbors (in forward and inverse directions).

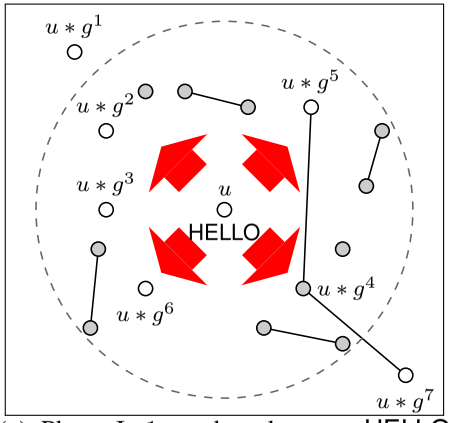


(a) Cycle rooted at u with g in *Ex-BCG*.

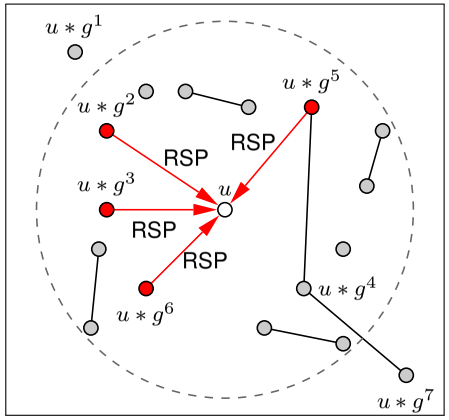
(b) Logical neighbor candidates (red colored nodes).

Figure 5.2: Example of logical neighbor candidates

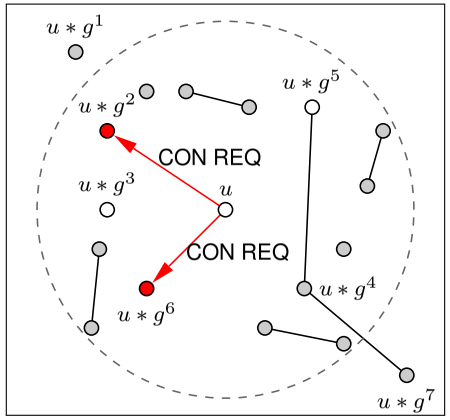
Fig. 5.2 shows an example of logical neighbor candidates. In Fig. 5.2(a), a cycle rooted at u with generator g is illustrated (u is connected to $u * g^1$ and $u*g^7 = u*g^{-1}$ in forward and inverse directions, respectively). From Definition 7, u has logical neighbor candidates $C_u = \{u * g^1, u * g^2, u * g^7\}$ in its transmission



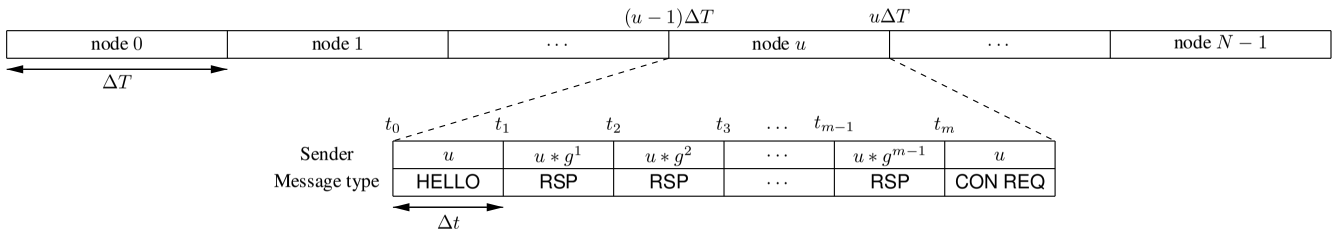
(a) Phase I−1: u broadcasts a HELLO message while non-logical neighbor candidates (gray colored nodes) are in sleep mode.



(b) Phase I−2: Logical neighbor candidates (red colored nodes) reply with a RSP message at their exclusively assigned time slot.



(c) Phase II: u selects the logical neighbors according to POW. Then, $u*g^2$ and $u*g^6$ update their neighbor list and the bidirectional links are established.



(d) Time slot assignment: ΔT is time assigned to one node and Δt is a single time slot.

Figure 5.3: EBTC algorithm and assigned time slots

range r as shown in Fig. 5.2(b) ($u * g^3$ already has logical neighbors $u * g^4$ and $u * g^2$).

Next we show *how a node discovers its logical neighbor candidates without collision* during Phase I of the EBTC. To collect node IDs of logical neighbor candidates, u broadcasts a HELLO message containing its integer ID at time t_0 (See Fig. 5.10(b) and Fig. 5.3(d)). Then, only logical neighbor candidates receiving the HELLO message reply to u with a RSP message containing its *node ID* and *neighbor list*. Other physical neighbors not qualified to be logical neighbor candidate are in sleep mode during u's EBTC operation as shown in Fig. 5.10(c). Also, $u*g^1$ and $u*g^7$ can't hear HELLO from u, thus they enter the sleep mode and wake up when its EBTC operation initializes or it involves other node's EBTC operation (The sleep mode minimizes energy consumption by preventing non-qualified nodes from overhearing).

The replying order of RSP messages from logical neighbor candidates is determined by the *power index* in the connection, which is a logical distance ($i.e.$, g^1, g^2 and g^7) from node u in the cycle. In other words, if u has all $m-1$ logical neighbor candidates ($C_u = \{u * g^1, u * g^2, \cdots, u * g^{m-1}\}$), $u * g^1$ replies at time t_1, $u * g^2$ replies at time t_2 and so on, as shown in Fig. 5.3(d)). Since each logical neighbor candidate can figure out which time slot for reply is assigned to it from its power index in connection obtained by the *modulo* operation of u's matrix node ID and generator g, replies do not result in collisions ($i.e.$, $u * g^1$ replies at first time slot and $u * g^2$ at second time slot). Therefore, u can reliably collect the node IDs of all its logical neighbor candidates without collisions. Once Phase I is finished, u proceeds to Phase II.

5.1.5 Phase II: Logical Neighbor Selection

In Phase II, a node selects logical neighbors from the logical neighbor candidates according to the POW connection rule. From our previous work in [79], we found that the connection characteristics of *Ex*-BCGs result in efficient topological properties and robust connectivity. In order to achieve these beneficial properties in wireless networks, the POW connection rule elects the most qualified logical neighbors from the logical neighbor candidates obtained in Phase I. The POW connection rule is inspired by the Cut-Through Rewiring (CTR) algorithm which is used for resizing *Ex*-BCGs in [79]. The POW connection rule operates as described in Fig. 5.4.

Why do we select the nodes with the minimum and maximum power indexes out of the logical neighbor candidates from POW connection rule? In *Ex*-BCGs, u

From Phase I, u obtains $C_u = \{u * g^{min}, \cdots, u * g^{max}\}$.
We define L_u as a length of cycle rooted at u.
If u doesn't have any logical neighbor,
  u selects $u * g^{min}$ and $u * g^{max}$ as logical neighbors.
  ($u * g^{min}$: logical neighbor in forward direction.)
  ($u * g^{max}$: logical neighbor in inverse direction.)
Else if u has a logical neighbor,
  If power index of u's logical neighbor $\geq L_u/2$,
    u selects $u * g^{min}$ as a logical neighbor.
  Otherwise,
    u selects $u * g^{max}$ as a logical neighbor.
Otherwise (u already has two logical neighbors),
  Do nothing.

Figure 5.4: POW connection rule

is connected to $u*g$ (forward direction) and $u*g^{-1} = u*g^{m-1}$ (inverse direction), where $g^m = I$. If $u * g$ is removed to resize *Ex*-BCGs, the CTR rewires u with $u*g^2$ in the forward direction with the highest priority. In case $u*g^{m-1}$ is removed, u is rewired to $u * g^{m-2}$ in inverse direction with the highest priority. With this rewiring rule, the resultant resized *Ex*-BCGs can conserve the efficient topological properties and robust connectivity. By applying this prioritizing connection rule in wireless networks, we select the nodes of the minimum and maximum power indexes as logical neighbors in the forward and inverse directions, respectively. Once completing the selection of logical neighbors, u broadcasts CON REQ message containing u's node ID and its logical neighbor list and adjusts its transmission power to cover the farthest logical neighbor. Then, the selected logical neighbor(s) updates its logical neighbor list according to received CON REQ. When the EBTC is completed, bidirectional links are established by adjusting its transmission power (See Fig. 5.10(d)).

Based on the Phase I and II operation, Fig. 5.5 shows wake-up schedule of u and its logical neighbor candidates of Fig. 5.3. From the figure, all elements of cycle rooted at u are in wake-up mode to receive HELLO at the first time slot of Phase I and send RSP to u in their time slot only if qualified to become a logical neighbor. Thus, $u * g^1$ and $u * g^7$ do not send RSP since they are out of u's transmission range (no receiving HELLO from u). Theoretically, there is no collision in message exchange in topology construction during EBTC execution.

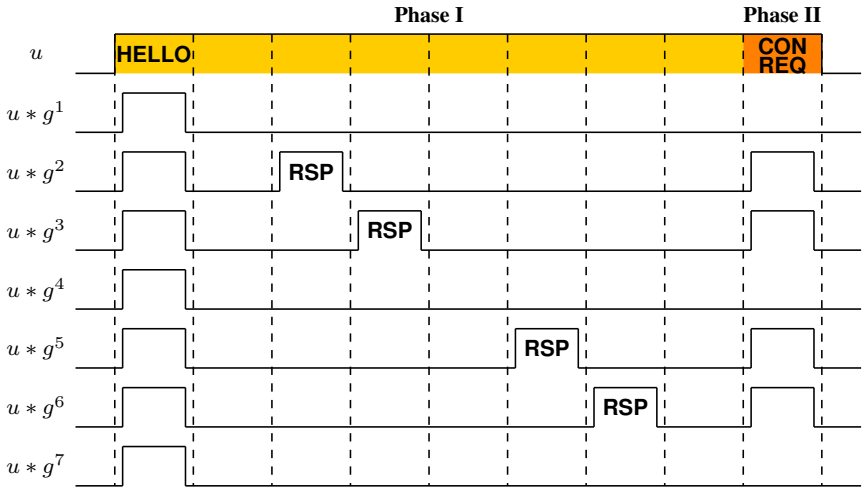


Figure 5.5: Coordinated wake-up schedules of u and its logical neighbor candidates: $u * g^1$, $u * g^4$ and $u * g^7$ are not qualified to be a logical neighbor ($i.e.$, $u * g^1$ and $u * g^7$ are out of u's transmission range and $u * u^4$ already has two logical neighbors in g direction). Thus, they are in sleep mode during the rest of u's Phase I and Phase II. Based on RSP messages from $u * g^2$, $u * g^3$, $u * g^5$ and $u * g^6$, u selects logical neighbors and sends CON REQ message.

5.1.6 Topology Construction Convergence Time Analysis

As discussed in Section 3.2 that the maximum length of cycle in *Ex*-BCGs is nk ($i.e.$, $g^{nk} = I$), the time ΔT for one node with a generator set $G = \{g_1, g_2, \cdots, g_d\}$ is bounded as follows: Note that one generator produces 2 logical neighbors in the maximum forward and inverse directions.

$$\Delta T \leq \Delta t(|G|^2(nk-1) + 2|G|), \tag{5.1}$$

where $n \geq 1$ and $k \geq 2$. Δt is a time slot for packet exchange between two nodes.

For nodes to perform the EBTC sequentially without collisions, they should be aware that how many time slots are assigned. The assigned time slots for one node is determined by cycles rooted at that node produced by a generator set G, which is obtained by the $moudlo\ np$ operation between its matrix node ID and generator g. Thus, we set the required time ΔT for one node to complete EBTC operation as:

$$\Delta T = \Delta t(|G| \sum_{g \in G} (\lambda_g - 1) + 2|G|), \tag{5.2}$$

where λ_g is the maximum length of cycles obtained from generating parameters p, k, a and generator g in *Ex*-BCG. As a result, $N\Delta T$ is the time needed to construct a communication topology under EBTC, where N is the total number of nodes. Based on this discussion, each node can figure out when it should enter the wake-up and sleep modes. For example, node 3 wakes up during $2\Delta T \sim 3\Delta T$ to perform the EBTC operation, and if node 3 belongs to a cycle rooted at node 5, it wakes up at $4\Delta T$ to receive HELLO from node 5. If node 3 doesn't receive HELLO from node 5 between $4\Delta T \sim 4\Delta T + \Delta t$ ($i.e.$, node 3 is not a physical neighbor of node 5), it goes to sleep mode. In all other time slots, node 3 is in sleep mode.

5.1.7 Performance Evaluation

For performance evaluation, we examine the EBTC algorithm and compare the results to three other topology control algorithms: *K*-Neighbor [26], Local Mimimal Spanning Tree (LMST) [27] and Relative Neighborhood Graph (RNG) [28].

Simulation Setup

In the simulation, we assume that nodes are ***uniformly and randomly*** distributed in an area of $100m \times 100m$. The nodes communicate with each other using a predefined transmission range. We collect the simulation results from 100 network samples at transmission ranges. For EBTC, we use a 990-node *Ex*-BCG expanded from a 110-node BCG with generating parameters $p = 11, k = 10, a = 2, n = 9$ and generator set $G = \{g_1 = \begin{pmatrix} 2^3 & 1 \\ 0 & 1 \end{pmatrix}, g_2 = \begin{pmatrix} 2^5 & 1 \\ 0 & 1 \end{pmatrix}, g_3 = \begin{pmatrix} 2^7 & 1 \\ 0 & 1 \end{pmatrix}, g_4 = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}\}$. Tab. 5.1 summarizes the details of the simulation setup.

Note that in the communication topology construction in [26–28] did not specify how to collect physical neighbor information in detail (no specific neighbor discovery process for collision avoidance). So we assume that a node is assigned enough time slots to find its physical neighbor information without collision.

In addition, we assume that a node should be awake while its physical neighbors are performing the topology control operation. This assumption is reasonable, because a node in those topology control algorithms does not know whether or not it becomes a logical neighbor of its physical neighbors until all its physical neighbors complete the neighbor discovery and topology control phases.

Table 5.1: Simulation parameters

TC algorithms	Degree bound	Network size	Transmission range
EBTC	$4\ (g_1, g_2)$, $6\ (g_1, g_2, g_3)$, $8\ (g_1, g_2, g_3, g_4)$,	990	$5, 10, \cdots, 30m$
K-Neigh	$6, 8$	1000	$5, 10, \cdots, 30m$
RNG		1000	$5, 10, \cdots, 30m$
LMST (G^-)		1000	$5, 10, \cdots, 30m$
LMST (G^+)		1000	$5, 10, \cdots, 30m$

Also, we assume that if none of the physical neighbors of a node are performing the topology control operation, the node goes to sleep mode. Therefore, we focus on the energy consumption only during message exchanges for transmission, reception and overhearing during the topology control operation while excluding energy consumption in node's idle-listening.

Metrics

For evaluation, we utilize the following metrics:

- **Energy consumption for topology construction**

  Energy consumption is an important metric to quantify the efficiency of topology control algorithm. Energy consumption represents the network overhead, since it is determined by the number of exchanged messages between nodes. For the simulation, we adopt the energy consumption model for transmitting (E_t) and receiving (E_r) data size k bits messages at distance d [15], that is:

$$E_t(k, d) = E_{elec} \cdot k + E_{amp} \cdot k \cdot d^2 \tag{5.3}$$

$$E_r(k) = E_{elec} \cdot k \tag{5.4}$$

  where $E_{elec} = 50nJ/bit$ and $E_{amp} = 100pJ/(bit\ m^2)$. We assume that all exchanged messages in the simulation are 10 bytes long including header and data.

- **Reachability**

  Reachability is used to examine the connectivity of a communication topology. We define the reachability as follows:

$$Reachability = \frac{Largest\ connected\ component\ size}{Network\ size} \tag{5.5}$$

  A component is defined as a subset of a network such that there exists at least one route from each node of that subset to each other node [80].

- **Average path length**

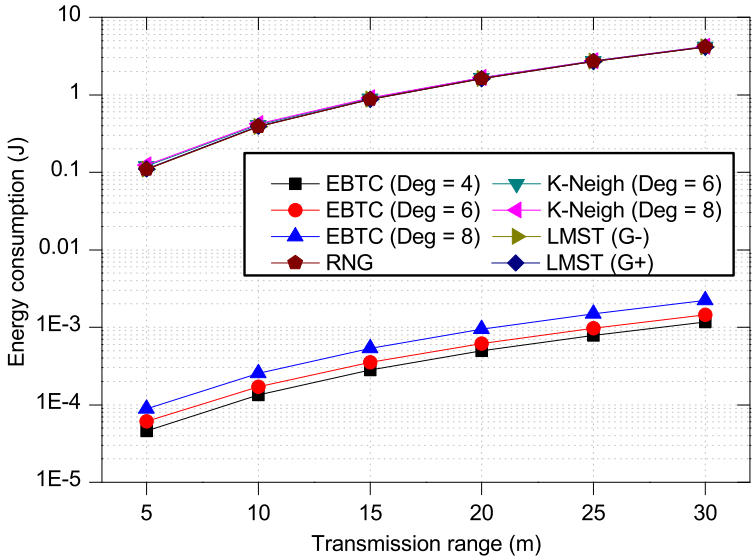
  The average path length is an average hop count between all pairs of nodes. This metric is a significant indicator for efficiency of data routing and relaying.

- **Average logical node degree**

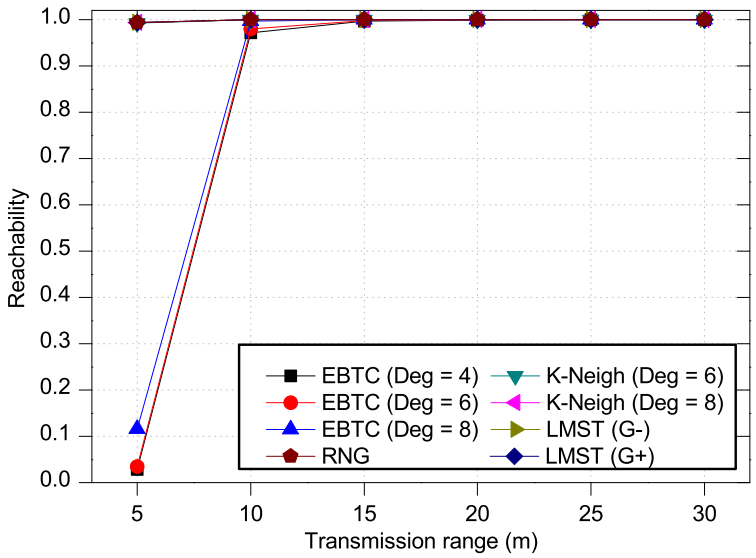
  In WSNs, it is desirable for nodes to have a small number of logical neighbors for communication to obtain a larger spatial reuse and better throughput.

Energy consumption for topology construction

Fig. 5.6(a) shows the energy consumption required to construct the topologies with the target algorithms. From the results, we found that the K-Neighbor, LMST and RNG algorithms consume almost an identical amount of energy to complete their topologies than the EBTC algorithm for two reasons: 1) a node that broadcasts a HELLO message should receive responses from all its physical neighbors and 2) all physical neighbors should reply to a HELLO message (with RSP) and should receive a response from the HELLO message originator (with CON REQ), even if they are not qualified to be a logical neighbor. On the contrary, the EBTC requires nodes to deterministically and selectively wake up and receive HELLO messages during their pre-defined time slots. Moreover, if there is no received HELLO message (out of the transmission range of the HELLO message originator), nodes remain in sleep mode. The result demonstrates that the deterministic and selective wake-up schedule of the EBTC prevents overhearing of nodes and reduces energy consumption considerably.

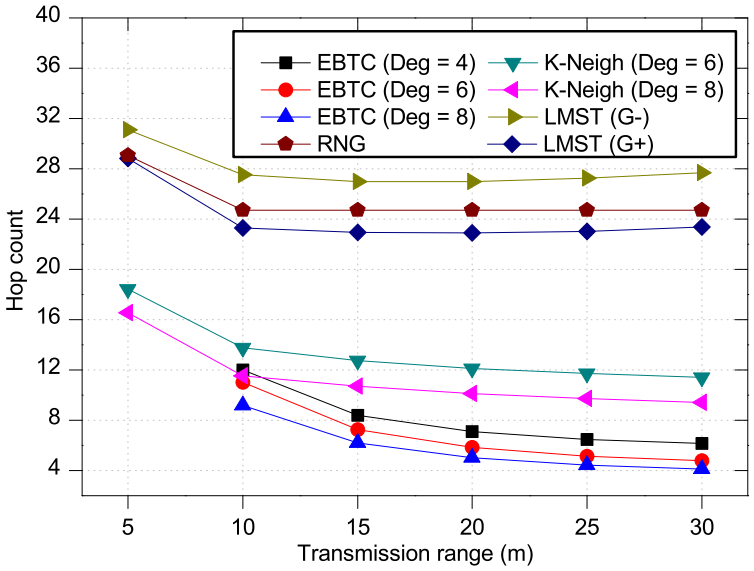


(a) Power consumption for topology construction

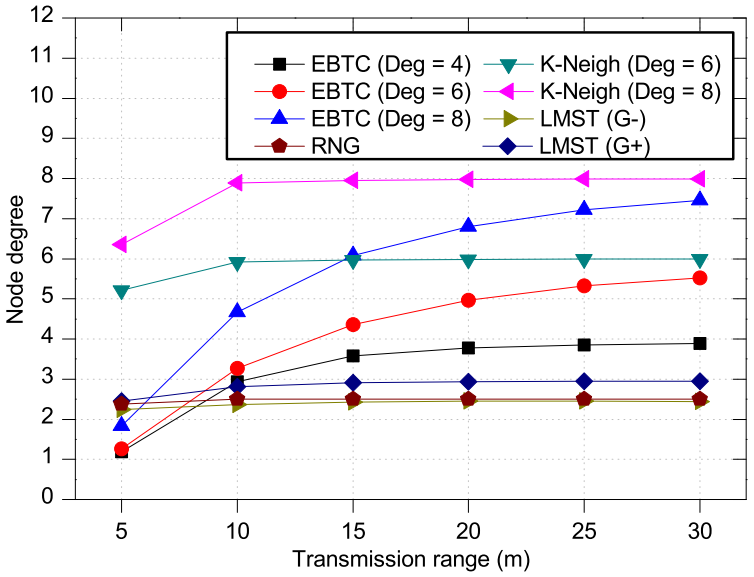


(b) Reachability

Figure 5.6: Simulation results of power consumption for topology construction and reachability



(a) Average Path Length



(b) Average logical node degree

Figure 5.7: Simulation results of average path length and average logical logical node degree

Reachability

Fig. 5.6(b) shows the reachability of the communication topologies from the topology control algorithms considered. From the results, we found that the EBTC generates almost fully connected communication topologies as a function of transmission range except at the $5m$ transmission range. At $5m$ transmission range, the nodes are not able to find enough logical neighbor candidates to generate a connected communication topology. However, more than 98% of network nodes are connected with a transmission range of $10m$, which is a reasonable transmission range for a $100m \times 100m$ target area. Even if the other topology control algorithms produce communication topologies with higher reachability than that of the EBTC, they are not facilitated with collision avoidance neighbor discovery scheme. From this consideration, the EBTC is a more effective topology construction algorithm that yields a reliable communication topology.

Average path length

In Fig. 5.7(a), the communication topologies from the EBTC and *K*-Neighbor topology control algorithms have much shorter average path length than those of LMST and RNG. However, if the average logical node degree in Fig. 5.7(b) is considered, the communication topologies from the EBTC have more efficient average path length (9.18 hop count with degree 8 of EBTC and 11.5 hop-count with degree 8 of *K*-Neighbor topology control at transmission range $10m$) since smaller nodal degree means more usability of channel reuse application. This result shows that the communication topology from our EBTC is beneficial for end-to-end communication in WSNs, because the small average path length with a reasonably short transmission range reduces interferences and retransmission occurrences.

Average logical node degree

From the results shown in Fig. 5.7(b), the average logical node degree of all communication topologies are bounded to small values between 3 and 8. The communication topologies from the RNG and LMST algorithms have the logical node degree bounded by 3 while the EBTC and *K*-Neighbor topology controls are bounded to predefined values between 4 and 8 and between 6 and 8, respectively. However, even if the average logical node degrees of the RNG and LMST algorithms are smaller, the EBTC is a more efficient topology construction algorithm

when we consider the results for energy consumption and average path length result.

5.2 Clustering based Routing Algorithm for WSNs

In this section, we propose a clustering based routing algorithm for WSNs. We first suggest a problem of other clustering algorithms with simulation results. Then, we provide our solution to tackle that problem and show its performance evaluation and comparison with other benchmark clustering algorithms.

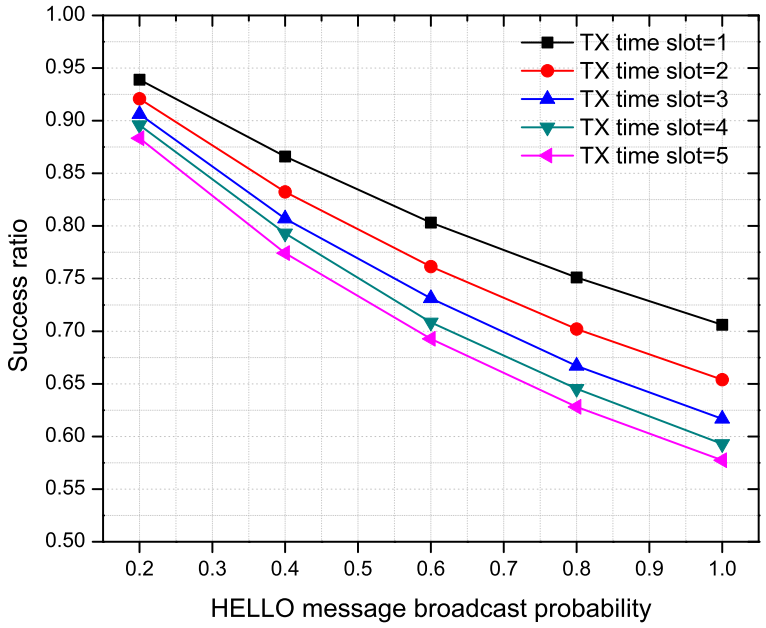
5.2.1 Problem Statement

Clustering based routing algorithms in WSNs have been widely researched for decades. However, most clustering algorithms start from the assumption that after deployment of nodes, each node is aware of its physical neighbors or can collect physical neighbor information without collisions [81]. Strictly speaking, this assumption is not valid in most in wireless network environment. Especially, in dense wireless networks, collecting physical neighbor information without collisions is a non-trivial issue. Therefore, clustering algorithms without accurate physical neighbor information can not produce the result as their authors claim. The following simulation illustrates the issue and highlight the need for a more practical scheme to collect physical neighbor information.

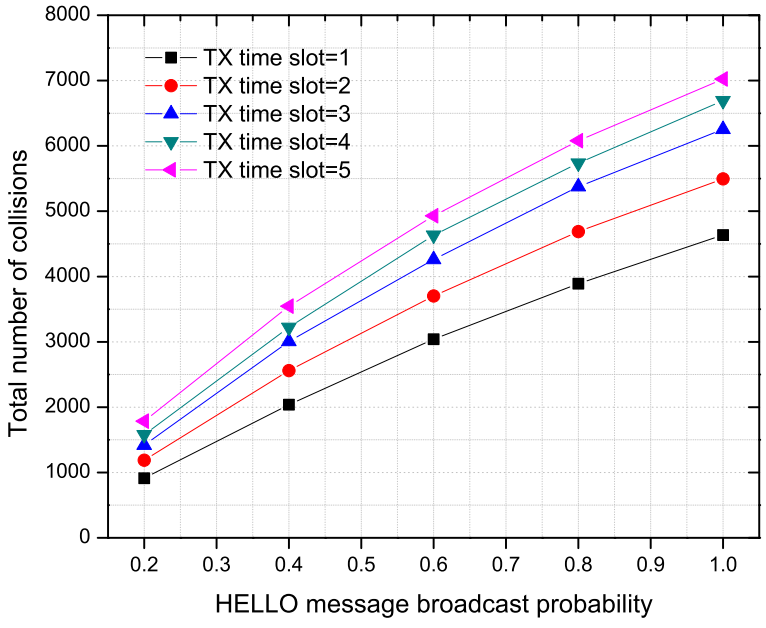
In the simulation, we assume that $1,000$ nodes are uniformly and randomly deployed in a $100m \times 100m$ area with identical initial energy. Nodes are labeled with unique integer ID and equipped with a synchronized timer. Once deployed, nodes start broadcasting to advertise its existence to physical neighbors with p-persistent CSMA protocol (widely used in WSNs for data transmission). Tab 5.2 summarizes the details of the simulation parameters. Note that transmission time slots represent the number of time slots for one broadcast. Each node broadcasts only once with a given probability p, if the channel is idle until waiting time is expired. In order to collect physical neighbor information, all nodes should be awake until the end of broadcasting for receiving.

As metrics, we measure the success ratio as defined in Eq (5.6) and the total number of collisions once all nodes completed broadcasting HELLO message.

$$\text{Success ratio} = \frac{\text{Number of found physical neighbors}}{\text{Number of physical neighbors}} \tag{5.6}$$



(a) Success ratio



(b) Total number of collisions

Figure 5.8: Simulation results for success ratio and total number of collisions

Parameters	Values
Network size	$1,000$ nodes
Transmission range	$5m$
Network samples	30
Probability p	$0.2, 0.4, \cdots, 1$
Back-off	$(min, max) = (2, 10)$
Transmission time slots	$1, 2, \cdots, 5$

Table 5.2: Simulation parameters

From the simulation results, we found that the success ratio along the broadcast probabilities never reaches 1, which means that collision prevents nodes from finding all their physical neighbors. Therefore, the resulting clusters can't produce the claimed performance. To tackle and resolve this physical neighbor discovery issue, in the next sections, we propose the *Ex* Clustering algorithm that is based on the EBTC algorithm to avoid collisions and save energy in the cluster formulation process.

5.2.2 Design Objectives and Network Model Assumptions

Before providing design objectives and network model assumptions, we describe the reason for focusing on dense WSNs. Generally, since WSNs aim on simple and specific applications such as measuring temperature and detecting movement in a target area, cost of a single sensor is inexpensive. Moreover, the capacity of radio transceivers for sensors has dramatically improved over the last decades. Thus, the deployment of a large number of sensor nodes in a target area to form dense WSNs has become a spotlighted method to achieve following purposes [81, 82].

- **Network life time extension**

  Operations of nodes in WSNs such as sensing, communicating, and processing of data are energy consuming work. Since a node in WSNs is equipped with a limited power source (battery) and normally not field-rechargeable, performing a given task becomes impossible after energy depletion. Although there have been many algorithms to save energy, deploying extra sensor nodes is one because of its inexpensive cost.

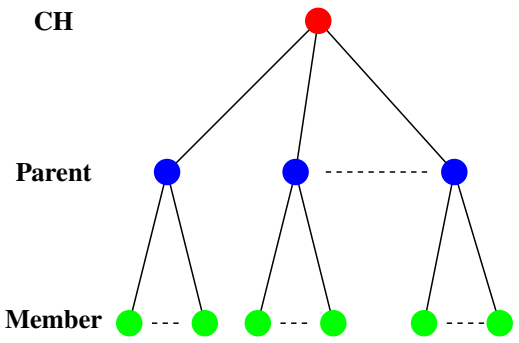


Figure 5.9: Cluster layer

- **Fault-tolerance** By deploying dense sensor nodes, the network can be more faulty tolerant. The dense network coverage by a large allowing it to sustain operations despite some of the nodes died out due to energy depletion.
- **Data reliability and accuracy**

  Data exchanged between nodes are prone to be lost if intermediate nodes along the data routing path die out. This data communication failure deteriorates the WSN application accuracy and reliability. Therefore, the extra nodes in a dense network can help prevent data loss and improve data accuracy.

To obtain the benefits stated above, it is necessary to design an efficient routing algorithm for dense networks. In the following sections, we propose the *Ex* Clustering algorithm.

Design Objectives

To allow for energy efficient and reliable routing in dense WSNs, the *Ex* Clustering algorithm must operate in a *distributed* manner with a three-layer cluster: cluster head (CH), parent and element (See Fig 5.9). Our *Ex* Clustering algorithm aims at maintaining 2 hops between CH and cluster member at most. The design objectives are as following:

- **Self-organization based on Local information**

The *Ex* Clustering algorithm operates in a distributed manner and nodes construct clusters based on local information with limited transmission power. Since there is no centralized command needed to form clusters, the resultant clustering performance is reliable and scalable. The *Ex* Clustering operation is sequentially performed in the order of node ID at each node.

- **Energy efficiency in cluster formulation and data transmission**

  Generally, operations consuming energy in clustering algorithms is categorized into two parts: **cluster formation** and **data communication**. Although most clustering algorithms focus on the latter, energy consumption in cluster formation is non-negligible. Our *Ex* Clustering algorithm improves energy efficiency in cluster formation as well as data communication. Moreover, our *Ex* Clustering allows nodes to stay in sleep mode when they are not clustered. This expands network lifetime by preventing early energy depletion of nodes.

- **Reliable and fast data communication with collision avoidance**

  Clusters generated from the *Ex* Clustering algorithm have an upper bound in size. Since the EBTC algorithm produces communication topologies where nodes have a limitation in the number of logical neighbors, the size of cluster formulated from that communication topology is bounded and controllable. Due to the bounded and controllable cluster size, the *Ex* Clustering algorithm can design a deterministic time schedule for reliable and fast data communication. Further, assigning a unique code and time slot to each cluster member prevents collisions since the number of required codes for nodes is limited to a small number.

- **Limited transmission power**

  Non-CH nodes in *Ex* Clustering, which should perform data communication with a base station, use limited transmission power reaching its next data relay node.

Network Model

Now we provide the assumptions used to construct our network model, which are effective throughout this section.

- Before deployment, nodes are assigned a unique integer node ID ranging from 0 to $N-1$, where N is the total number of nodes and perform the clustering operation sequentially ($e.g.$, from node 0 to node $N-1$).

- Nodes are equipped with a synchronized timer. In practice, clock drift among timers is a non-trivial problem that may trigger false interactions between nodes. However, we assume that each time slot is significantly longer the that clock drift and ensures enough time for transmission, reception and process of data.

- Nodes are equipped with an RSSI (Received Signal Strength Indication) detector. Initially nodes are assigned identical transmission power, but nodes adjust the transmission power to cover their farthest logical neighbor once clustering operation is completed.

- Nodes are homogeneous ($e.g.$, equal energy, identical transmission and sensing capacity, etc.), and randomly and uniformly distributed in a target area and are not mobile.

- Nodes do not have a hardware like GPS. Thus they are unaware of the geographic location of other nodes.

- A base station is a device that has a permanent power source and located outside of the target area. For example, if the coordinates of a network area are ranging from $(0, 0)$ to $(100, 100)$, the based station is located at $(50, 150)$.

Based on these assumptions, we present the details of the *Ex*-Clustering algorithm in the following sections.

5.2.3 *Ex* Clustering Algorithm

The *Ex* Clustering algorithm aims at formulating clusters based on the logical communication topology and achieving energy efficiency and effective data routing. The operation of the *Ex* Clustering algorithm is composed of multiple number of rounds, where a round is a CH rotation period. One round is composed of two phases: cluster setup and data communication. The following sections discuss the details on how to formulate clusters and establish data communication with energy saving and collision avoidance.

Neighbor ID	Hop	Remaining Energy	Communication Distance	Degree
1	1	e_1	c_{u1}	d_1
2	1	e_2	c_{u2}	d_2
3	1	e_3	c_{u3}	d_3
4	1	e_4	c_{u4}	d_4
5	2	e_5	c_{u5}	d_5
6	2	e_6	c_{u6}	d_6
7	2	e_7	c_{u7}	d_7
8	2	e_8	c_{u8}	d_8
9	2	e_9	c_{u9}	d_9
10	2	e_{10}	c_{u10}	d_{10}
11	2	e_{11}	c_{u11}	d_{11}
12	2	e_{12}	c_{u12}	d_{12}
13	2	e_{13}	c_{u13}	d_{13}

Table 5.3: Collected two-hop logical neighbor information at node u

Phase I: Cluster setup

As discussed in the previous section, the underlying communication topology of the *Ex* Clustering is formulated by the EBTC algorithm. However, we assign additional tasks to each node when it collects logical neighbor candidates information. During collecting logical neighbor candidates information, logical neighbor candidates are required to include *their logical neighbor* information in the RSP message so that the collecting node obtains two-hop logical neighbor information, once the communication topology construction is completed. The included logical neighbor information contains the following attributes of itself and its one-hop neighbors:

- Remaining energy
- Communication distance measured by RSSI
- The number of logical neighbors (degree)

From the collected two-hop logical neighbor candidates, each node stores its local information and Tab 5.3 shows an example of the collected two-hop logical neighbor information.

Note that c_{uv} represents the sum of the communication distance from node u to node v, not geographic distance between nodes u and v.

Notation	Definition
e	energy factor
E_u	Remaining energy of node u
E_{max}	Initial energy
n	cluster size factor
N_u	the number of two-hop neighbors from node u
N_{max}	the maximum number of two-hop neighbors
c	communication distance factor
C_u	the sum of communication distance from node u to two-hop neighbors
C_{max}	the sum of *maximum* distance to two-hop neighbors

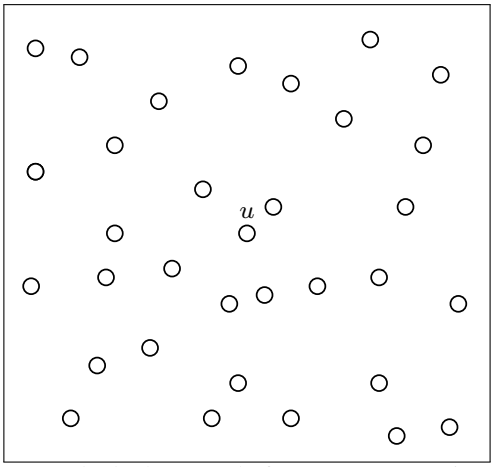
Table 5.4: Notations definition

Based on the collected local information, each node executes the cluster formation process. The cluster formation in the *Ex* Clustering algorithm is a self CH election process. The CH election utilizes a scoring equation which measures qualification to be CH. Each node i calculates its scores and its two-hop logical neighbors according to Eq (5.7). Note that Tab. 5.4 shows the notations definitions.

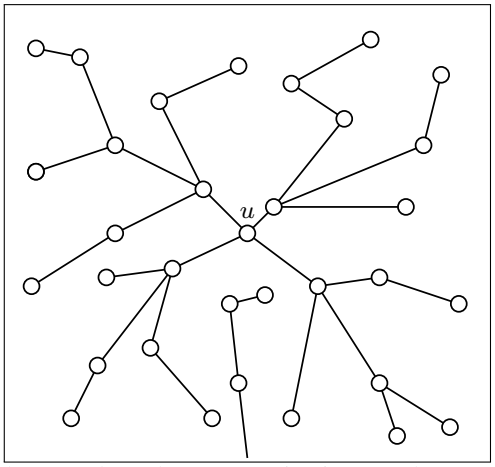
$$score = e \cdot E_u/E_{max} + n \cdot N_u/N_{max} + c \cdot C_u/C_{max} \tag{5.7}$$

Eq (5.7) is a weight function which takes factors as parameters that affect network life time and data communication efficiency during cluster formulation: remaining energy, the number of member nodes and communication distance.

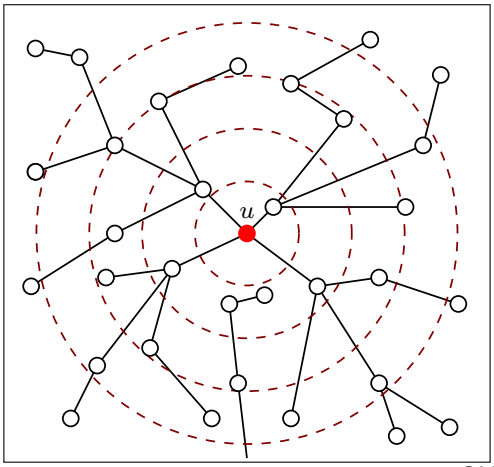
From the obtained scores, if node u's score is highest or more than 90% of the best score among the calculated scores and there is no clustered neighbor within its two hops, it becomes a CH and advertises (broadcasts) CH ELEC message with a transmission power set to reach all nodes in the network, which keeps clustered nodes from being clustered in other cluster. The CH ELEC message contains code and time slot information: node ID list of clustered nodes, a unique code for each parent and its member nodes and time slot for data communication. The node ID list distribution is, as mentioned previously, to prevent overlapping clusters. The code assignment to each parent and its member nodes is for simultaneous data communication with avoiding collision. The time slot is given to each node according to Fig. 5.11. Basically, all CHs are given a time slot 0 so that they send



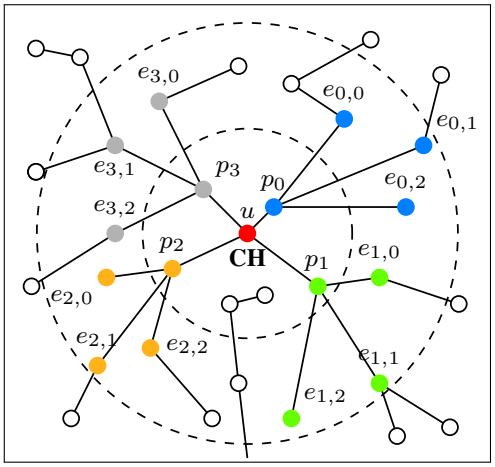
(a) Node deployment before EBTC operation.



(b) Based on the communication topology resulting from the EBTC operation, u calculates scores of two-hop neighbors.



(c) u has the best score and broadcasts CH ELEC message to its two-hop neighbors and the rest of nodes.



(d) With codes obtained from base station, u distributes (broadcasts) codes and time slots for data communications.

Figure 5.10: Phase I: Cluster setup

data to the BS with the assigned code when time t is $t\ mod\ (d+1) = 0$. The purpose of code and time slot assignment is to take advantage of a combination of Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA). Fig. 5.10 shows an illustration of Phase I, which consists of 4 parents ($p_0 \sim p_3$) and 12 member nodes ($e_{0,0} \sim e_{0,2}, e_{1,0} \sim e_{1,2}, e_{2,0} \sim e_{2,2}$ and $e_{3,0} \sim e_{3,2}$).

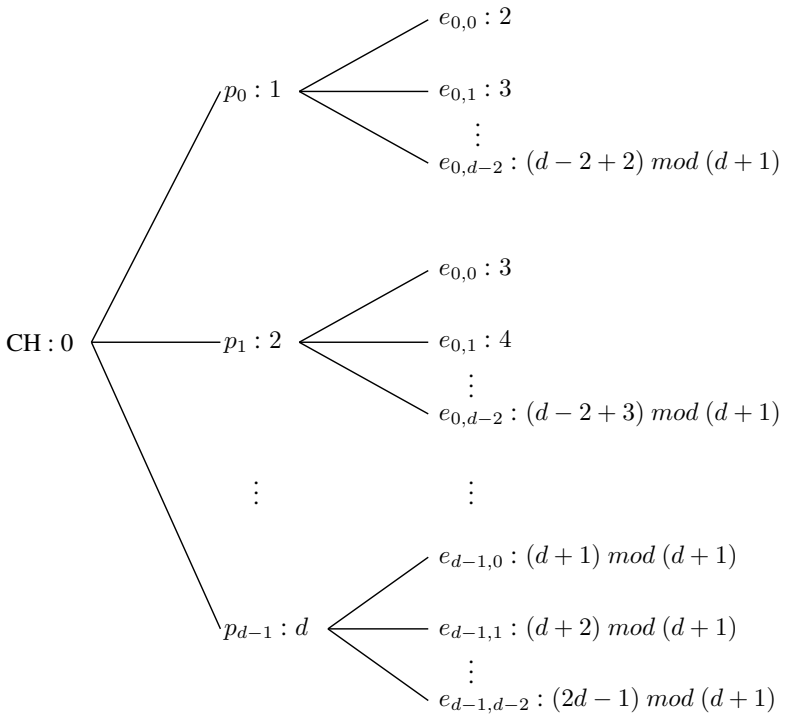


Figure 5.11: Time slot assignment: note that d represents the maximum number of logical node degrees.

In Fig. 5.10, node u elects itself as a CH and broadcasts the CH ELEC message containing node IDs list of cluster nodes, codes (colors) for CH and each parent and time slot for data transmission. The one-hop neighbors of node u receiving the CH ELEC record node u as a CH and a parent, while its two-hop neighbors record node u as CH and the intermediate node (one hop neighbor from node u) as a parent. The receiving parent and member nodes store their codes and time slot for data communication. Other nodes that are not a member of the cluster remember the node IDs of the announced list and if the list includes their one-hop or two-hop neighbors, the node does nothing during its clustering setup

Time	0	1	2	3	4	5	6	7
CH	tx to BS	rx from p_0	rx from p_1	rx from p_2	rx from p_3	tx to BS	rx from p_0	rx from p_1
p_0		tx to **CH**	rx from $e_{0,0}$	rx from $e_{0,1}$	rx from $e_{0,2}$		tx to **CH**	rx from $e_{0,0}$
$e_{0,0}$			tx to p_0					tx to p_0
$e_{0,1}$				tx to p_0				
$e_{0,2}$					tx to p_0			
p_1	rx from $e_{1,2}$		tx to **CH**	rx from $e_{1,0}$	rx from $e_{1,1}$	rx from $e_{1,2}$		tx to **CH**
$e_{1,0}$				tx to p_1				
$e_{1,1}$					tx to p_1			
$e_{1,2}$	tx to p_1					tx to p_1		
p_2	rx from $e_{2,1}$	rx from $e_{2,2}$		tx to **CH**	rx from $e_{2,0}$	rx from $e_{2,1}$	rx from $e_{2,2}$	
$e_{2,0}$					tx to p_2			
$e_{2,1}$	tx to p_2					tx to p_2		
$e_{2,2}$		tx to p_2					tx to p_2	
p_3	rx from $e_{3,0}$	rx from $e_{3,1}$	rx from $e_{3,2}$		tx to **CH**	rx from $e_{3,0}$	rx from $e_{3,1}$	rx from $e_{3,2}$
$e_{3,0}$	tx to p_3					tx to p_3		
$e_{3,1}$		tx to p_3					tx to p_3	
$e_{3,1}$			tx to p_3					tx to p_3

Figure 5.12: Code assignment and time schedule for data communication

operation. In the figure, same colored nodes mean that they will use the same code in communication. Fig. 5.12 shows code assignment and time slot allocation to node u's parents and its member nodes. For instance, member $e_{0,0}$ uses blue colored code (specific orthogonal code to other code) to send data to its parent p_0. For transmission of data CH to base station, CH uses the code of CH (red colored code).

Nodes sequentially perform the CH election process in the order of node IDs ($i.e.$, from node ID 0 to $N-1$). In the network model assumptions, we assumed that nodes have a synchronized timer and aware of time slot for data exchange. Therefore, the sequential operation does not trigger collision in the cluster setup phase. Furthermore, in order to save energy, *Ex* Clustering adopts a passive node sleep mode, which means that when the cluster formulation is completed, the non-clustered nodes will enter sleep mode ($i.e.$, radio and sensor off mode) during data transmission phase.

Phase II: Data communication

Data communication in *Ex* Clustering is simple that each node uses its own time slot and code assigned during Phase I. We define a frame as the number of time slots that each node has a chance to transmit data at least once. Therefore, the time slots for data transmission phase is determined by how to assign the time slot

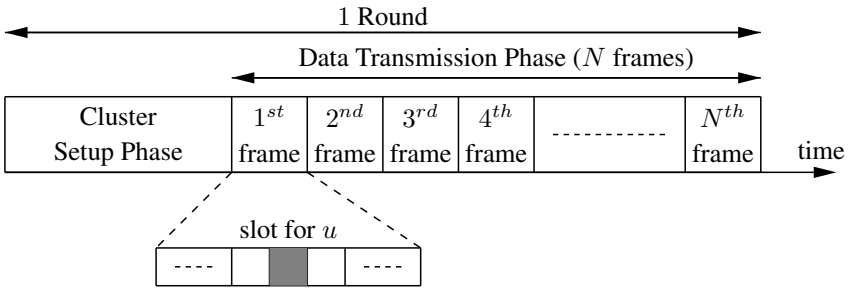


Figure 5.13: One round of Cluster Setup phase and Data Transmission Phase with N frames

to node. In our clustering algorithm, the number of required time slots for one frame is

$$\text{time slots of one frame} = 2d_{max} \tag{5.8}$$

where d_{max} is the number of maximum logical node degrees. Generally, one round has multiple number of frames since too frequent CH rotation, which requires many number of cluster setup phases, triggers energy waste and network complexity increase when network size is very large. Fig. 5.13 shows one round composition when there are N frames. When N^{th} frame time is expired, a new round starts.

Cluster Size and Cluster Formulation Time Convergence Analysis

Cluster size bound of the *Ex* Clustering algorithm is determined by the number of maximum logical node degrees. If the maximum logical node degree is d_{max}, the maximum cluster size C_n is as follows:

$$C_n \leq d_{max}^2 + 1 \tag{5.9}$$

In case of cluster formation time convergence, using Eq (5.2) for EBTC operation time convergence, the time convergence T for cluster setup phase is

$$T = N(\Delta T + \Delta t) \tag{5.10}$$

where Δt is for broadcasting CH ELEC for nodes. Based on the discussion so far, we investigate the performance of *Ex* Clustering algorithm with comparison

models.

5.2.4 Performance Evaluation

For evaluation of *Ex* Clustering and comparison study, we select LEACH and its variants as benchmark clustering algorithms. Since LEACH describes the formation mechanism in detail such as when to announce CH election result for CHs, how to find and select CHs for member nodes without collision and how to assign time slots for data communication, the LEACH is most qualified comparison model for our *Ex* Clustering algorithm. In the following section, we provide an overview of LEACH and its variants.

LEACH overview

Low-Energy Adaptive Clustering Hierarchy (LEACH) is one of the most well known clustering algorithms for WSNs. Its one round consists of two phases: Set-up and Steady-state phases. In the Set-up phase, node i probabilistically elect itself as a CH with the following equation:

$$P_i(t) = \begin{cases} \frac{k}{N-k\times(r \bmod \frac{N}{k})} & : C_i(t) = 1 \\ 0 & : C_i(t) = 0 \end{cases} \tag{5.11}$$

where r is an index of round, k is the intended number of CHs and N is the number of nodes. The LEACH algorithm ensures each node to become a CH only once during k/N rounds to save energy. Once node has been elected, it is not accounted for CH during the rest of rounds. If a node self-elect as a CH, it advertises with transmission power covering all other nodes. The advertisement receiving nodes record all CHs and calculate distances to each one based on RSSI. Among them, nodes select the closest one as its CH and sends a Join request to become a member of the selected CH. After each CH collects the Join request messages, it distributes TDMA schedule and code to its cluster member nodes, which will be used in data transmission. The LEACH considers MAC protocol, which is a combination of TDMA and CDMA, to avoid collision during cluster setup and data transmission.

As a variant, we modified the LEACH algorithm that nodes selectively enter sleep mode with a probability to further save energy (Sel-LEACH). This is for comparison fairness with our algorithm since in the *Ex* Clustering algorithm, non-clustered nodes enter sleep mode during the following data transmission phase.

We derive the selection probability based on the percentage of live nodes in *Ex* Clustering and compare those performance.

How to find the optimal numbers of CHs and frames in LEACH, Sel-LEACH and Ex Clustering

Differing from LEACH in [15] where deals with 100 nodes and periodic data communication, we target the larger number of nodes ($i.e.$, $1,000$ nodes) and event-driven data communication in the same size of network area. Although [15] describes how to find the optimum numbers of CHs and frames, our simulation result shows that the equation which was used to find those optimal values, is not appropriate. Thus we found the optimal numbers of CHs and frames according to the selected simulation parameters.

Parameter	Value
Target area size	$100m \times 100m$
Network size	$1,000$ nodes
Network samples	10
Simulation time slots	$10,000$
Total events	$2,000$
Active mode selection probability	$0.25, 0.4, 0.5, 0.75, 1$
CHs	$5, 6, \cdots, 50$

Table 5.5: Simulation parameters for finding optimum number of CHs

First, we find the optimum number of CHs with the simulation parameters described in Tab 5.5. Based on the result, we measure the efficiency of each case as following: Note that the higher CH efficiency is the better number of CHs.

$$\text{CH Efficiency} = \frac{\text{Number of reported events}}{\text{Total energy consumption}}. \tag{5.12}$$

Once the optimal number of CHs is obtained, we find the optimal number of frames by using simulation parameters of Tab 5.6. We select the number of frames generating the best efficiency using

$$\text{Frame Efficiency} = \frac{\text{Reported event ratio}}{\text{Energy consumption per one event report}}. \tag{5.13}$$

Parameter	Value
Target area size	$100m \times 100m$
Network size	$1,000$
Network samples	10
Simulation runtime	1 round
Event generation ratio	0.2
Number of frames	$50, 100, \cdots, 700$

Table 5.6: Simulation parameters for finding optimum number of frames

From the simulation conducted to find the optimum number of CHs and frames, we acquired the result as shown in Tab 5.7.

Selection probability	Optimum number of CHs	Optimum number of frames
0.25	14	550
0.5	22	500
0.75	32	550
1	49	150

Table 5.7: Optimum numbers of CHs and frames for LEACH and Sel-LEACH

Similarly, we also found the optimum number of frames for *Ex* Clustering as 450. Note that the number of CHs in *Ex* Clustering can not be explicitly controlled unlike LEACH and Sel-LEACH, which is determined by score calculated with weighted CH election function. Based on the result, we investigate the performance of LEACH and Sel-LEACH and compare it with that of *Ex* Clustering.

Simulation parameters

For performance comparison study, we assume that nodes are uniformly and randomly distributed in the area of $100m \times 100m$. In the simulation, we generate an event with ratio 0.2 per each time slot which is randomly located at random time. The simulation run time is until $2,000,000$ time slots or there is no CH elected. The detail of network parameters are described in Tab. 5.8 and Tab. 5.9.

For *Ex* Clustering, we deploy 990 nodes derived from 990-node *Ex*-BCGs with generators $g_1 = \begin{pmatrix} 2^1 & 1 \\ 0 & 1 \end{pmatrix}, g_2 = \begin{pmatrix} 2^3 & 1 \\ 0 & 1 \end{pmatrix}, g_3 = \begin{pmatrix} 2^5 & 1 \\ 0 & 1 \end{pmatrix}$ and $g_4 = \begin{pmatrix} 2^7 & 1 \\ 0 & 1 \end{pmatrix}$. In term of CH election factors, since nodes have identical energy at the deployment,

Parameter	Value
Target area size	$100m \times 100m$
Network size	990
Maximum logical node degrees	8
Initial transmission range	$10m$
Factors of CH election at round 1	$n = 0.5$, $c = 0.5$
Factors of CH election after round 1	$e = 0.7$, $n = 0.2$, $c = 0.1$
Number of frames	450
Base station location	$(50, 175)$
Initial energy	$2J$
Data size	500 bytes
Event generation ratio per time slot	0.2
Network samples	30

Table 5.8: Network parameters for *Ex* Clustering

we use $e = 0$, $n = 0.5$, $c = 0.5$ for first round. The data size for all cluster setup and data transmission phases is constant as 500 bytes. The power consumption model to transmit (E_t) and receive (E_r) data derived from [15] is defined as follows:

Parameter	Value
Target area size	$100m \times 100m$
Network size	$1,000$
(CHs ratio, selection probability, frame)	$(0.014, 0.25, 550), (0.019, 0.4, 600), (0.022, 0.5, 500),$ $(0.032, 0.75, 550), (0.049, 1, 150)$
Base station location	$(50, 175)$
Initial energy	$2J$
Data size	500 bytes
Event generation ratio per time slot	0.2
Network samples	30

Table 5.9: Network parameters for LEACH and Sel-LEACHs

$$E_t(k,d) = \begin{cases} kE_{elec} \cdot + k \cdot d^2 \cdot 10pJ/bit/m_2 & \text{if } d \leq 87m \\ kE_{elec} \cdot + k \cdot d^4 \cdot 0.0013pJ/bit/m^4 & \text{if } d > 87m \end{cases} \tag{5.14}$$

$$E_r(k) = E_{elec} \cdot k \tag{5.15}$$

where $E_{elec} = 50nJ/bit$.

Simulation Result

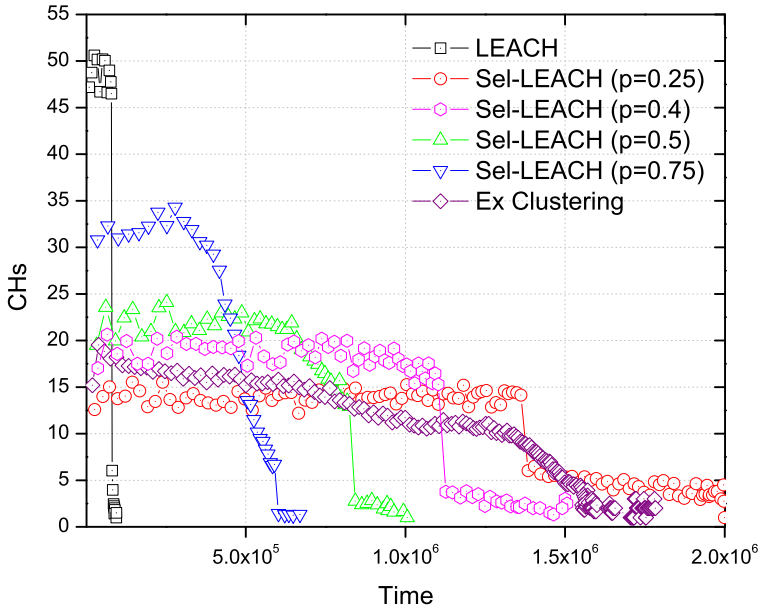
Based on the simulation parameters discussed in the previous section, we measure the following metrics.

- **Number of CHs**
- **Average cluster size**
- **Remaining node energy**
- **Live node ratio**
- **Reported event ratio**
- **Average report delay**

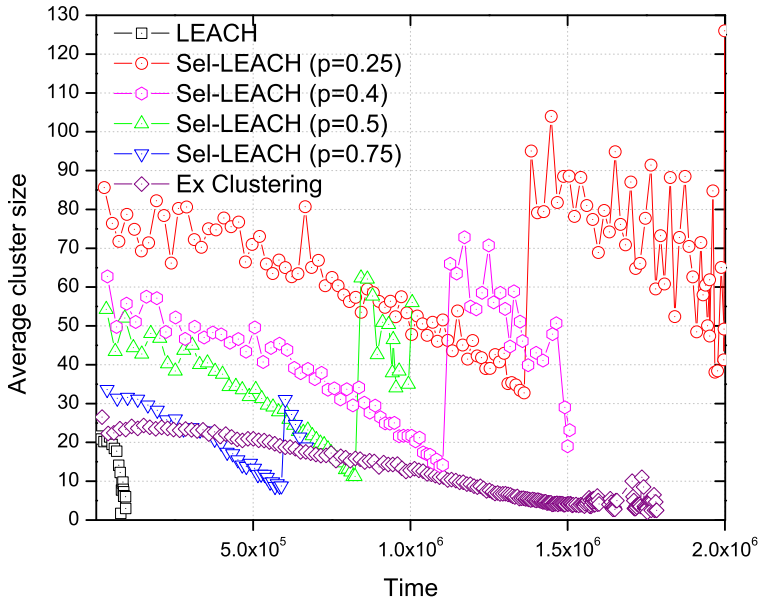
Fig. 5.14(a) and 5.14(b) show the average number of CHs and cluster size of investigated clustering algorithms. As expected, the average numbers of CHs of all algorithms decrease along the time. LEACH and Sel-LEACH clustering models show a quick drop in the number of CHs at a specific time point without regard to p. Those quick drops are caused by dead nodes due to energy depletion, which are a CH candidate only allowed in that round. However, *Ex* Clustering shows smooth decreasing since there is no restriction such that a node can not be elected as a CH for a specific time duration since it becomes a CH.

As expected from the result of the average number of CHs, the average cluster sizes of LEACH and Sel-LEACH models shown in Fig. 5.14(b)have a quick increase in the number of clustered nodes at time points when the average numbers of CHs have a quick drop. It is straightforward because the number of CHs is small, but active nodes should select a CH for data transmission.

In Fig 5.15(a), we depict the average of remaining node energy along the time. From the result, we identified that Sel-LEACH with $p = 0.25$ has the largest remaining energy along the simulation runtime when comparing with other models. Also, in the result of live node ratio shown in Fig. 5.15(b), we found that Sel-LEACH with $p = 0.25$ keeps nodes alive as many as possible during the simulation runtime. As far as the energy is concerned, the Sel-LEACH with $p = 0.25$ is the most stable clustering algorithm. However, the result of reported event ratio

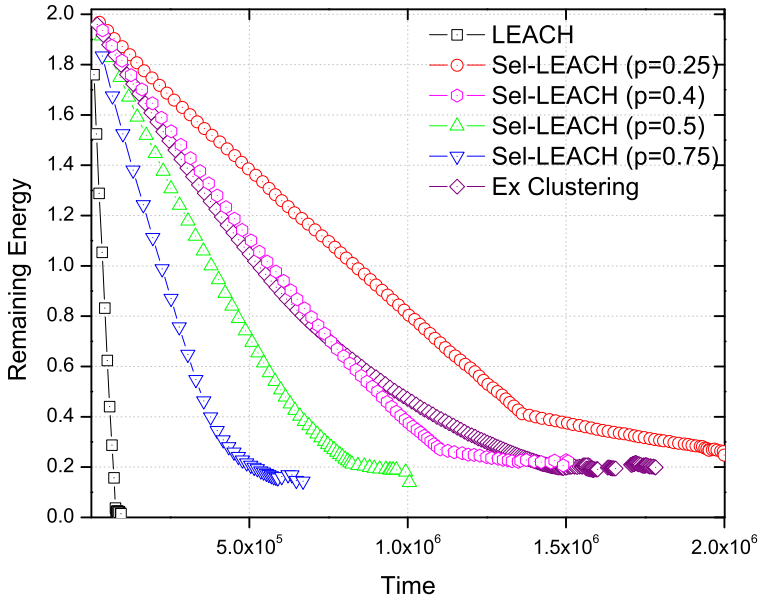


(a) CHs

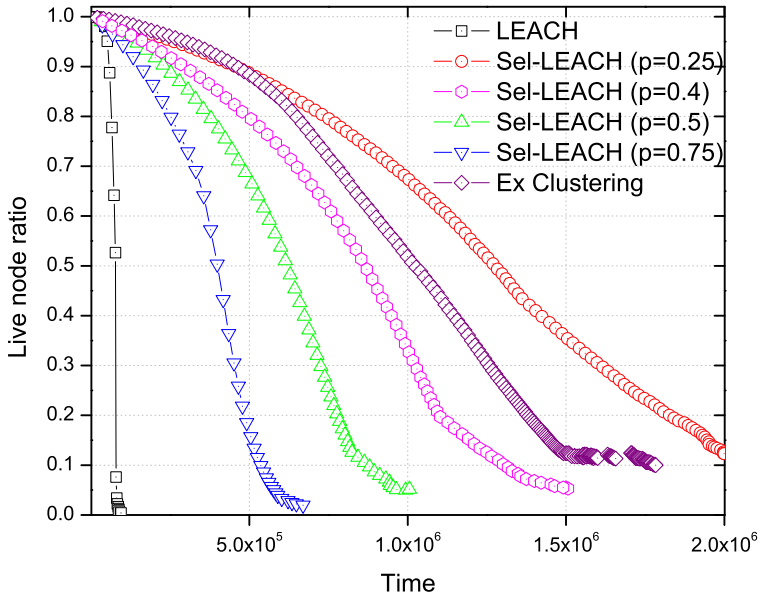


(b) Cluster size

Figure 5.14: CHs and cluster size

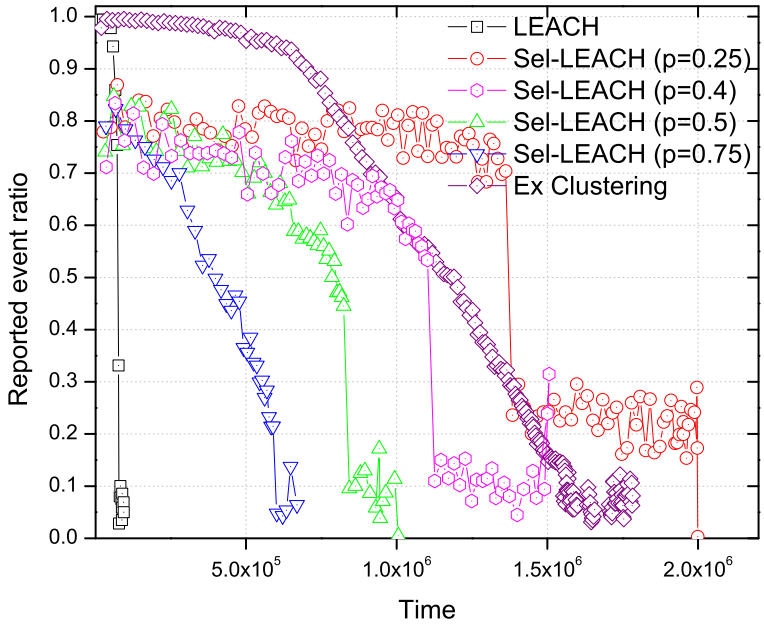


(a) Remaining node energy

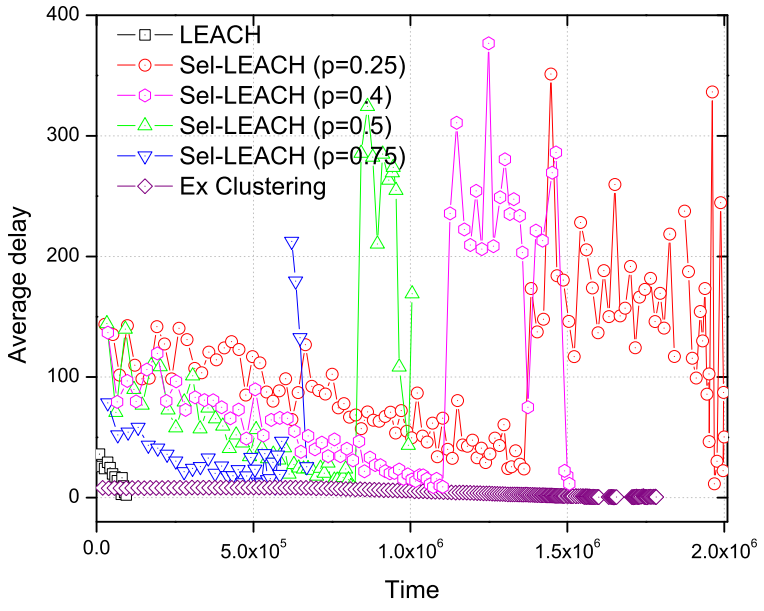


(b) Live node ratio

Figure 5.15: Remaining node energy and live node ratio



(a) Reported event ratio



(b) Average report delay

Figure 5.16: Reported event ratio and average delay

and average report delay reveals that Sel-LEACH with $p = 0.25$ is not the best clustering algorithm.

Fig 5.16(a) and 5.16(b) depict the reported event ratio and average report delay. The result shows that *Ex* Clustering has the highest reported event ratio among the considered clustering algorithms, but the reported event ratio of the Sel-LEACH with $p = 0.25$ reverses that of *Ex* Clustering around 8.0×10^5 time slots. According to the average report delay, however, the *Ex* Clustering has much lower delay than that of Sel-LEACH with $p = 0.25$.

5.3 Discussion

In this chapter, we proposed the EBTC algorithm to construct a *Ex*-BCGs based communication topology which avoids collision and saves energy consumption during formulation. The EBTC algorithm utilizes a deterministic node scheduling scheme for message exchange derived from connection characteristic of *Ex*-BCG, which enables nodes to avoid collision. Further, the node scheduling scheme allows nodes to find their sleep schedule according to **POW** connection rule in order to save energy and prolong network lifetime. From performance evaluation, we observed that our EBTC algorithm produce more efficient energy consumption, reliable reachability and shorter average path length than the communication topologies from other benchmark topology control algorithms.

As an extended work of EBTC, we designed the *Ex* Clustering algorithm and investigated its performance with comparison study. Our *Ex* Clustering bases on the communication topology generated by EBTC algorithm and forms clusters according to a score function which takes energy and communication efficiencies into account. In the simulation, we compared the *Ex* Clustering with LEACH and Sel-LEACHs in event-driven WSNs application. From the simulation results evaluating clustering, energy and communication efficiency, we observed that even if the Sel-LEACH with $p = 0.25$ shows better remaining node energy and larger ratio of live nodes, *Ex* Clustering outperformed it in reported event ratio and average report delay, which are important metrics to quantify clustering efficiency in event-driven WSNs application.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this dissertation, we proposed *Ex*-BCGs to improve scalability and eliminate the generating parameter dependence of BCGs for network performance. *Ex*-BCGs are composed of systematic expansion of node ID space and connection redefinition. The resultant *Ex*-BCGs have useful network properties such as a GCR representation, class-level vertex transitivity, fast consensus convergence speed and efficient topological/spectral properties. From simulation results, we confirmed that *Ex*-BCGs have consistent and outstanding performance along 5 times size expansion (sizes of $2, 3, 4$ and 5 times) from 506-node and 979-node BCGs. To build an arbitrary size networks from *Ex*-BCGs, we applied the CTR algorithm to rewire nodes that lost neighbors due to node failures. The connection rule of *Ex*-BCGs allowed CTR to preserve excellent network properties even after a large ratio of node failures and removal. The simulation results revealed that the resized *Ex*-BCGs maintain a robust connectivity. Further, topological/spectral properties and consensus convergence protocol speed of the resized *Ex*-BCGs are superior to those of BCGs.

As one of *Ex*-BCGs based applications, we presented the CVT routing algorithm which is a distributed algorithm that enables an optimal (shortest) routing path based on class-level vertex transitivity. Moreover, our routing table folding and merging schemes simplified routing table lookup process. The simulation results with *All-to-All* and *All-to-M* traffic patterns showed that the CVT routing achieved reliable reachability and efficient routing path length. For fault-tolerant routing of the resized *Ex*-BCGs, the AMA routing algorithm addressed the sin-

gle shortest path and multi-path depletion problems of the CVT routing. The AMA routing exploited all available routing paths from the updated routing table by counting the number of multi-paths along the selected path. When there is no more available options from the updated routing table, the AMA routing randomly selects the next forwarding neighbor which has not been traversed. From the simulation with and without traffic patterns and finite buffer consideration, we identified the reliability (routing success) and efficiency (short average routing path length) of the AMA routing algorithm in the considered range of node removals.

To construct *Ex*-BCGs based communication topologies in WSNs, we designed the EBTC, a topology construction algorithm, which integrates neighbor discovery and topology control operations with collision avoidance and energy saving. To create a node's advertisement schedule, the EBTC algorithm employed a cycle characteristic of *Ex*-BCGs. In the EBTC operation, nodes reduce energy consumption in topology construction by entering the sleep mode when not qualified to be a logical neighbor candidate. Through comparison study of the EBTC and other topology control algorithms, energy consumption in the EBTC required less energy consumption than those of considered RNG, k-Neigh and LMST topology control algorithms. In addition, through simulation we investigated reachability, average path length and logical node degree with a variety of transmission ranges and we verified that our EBTC produced an efficient communication topology in WSNs.

We provided the *Ex* Clustering algorithm, an extension of the EBTC operation, for efficient data communication in event-driven WSNs applications. The *Ex* Clustering algorithm formed three-layer clusters in the communication topology derived from the slightly modified EBTC operation. Each node self-elects CH based on the scoring equation which quantifies its remaining energy, two-hop communication distance and number of two-hop neighbors. Once a node is elected as a CH, its one-hop and two-hop neighbors become parents and member nodes, respectively. Otherwise, non-clustered nodes enter the sleep mode to save energy. This CH election operation is conducted in sequential order of node IDs and periodically rotates CHs to expand network lifetime. In comparison studies with LEACH and selective LEACH with event-driven WSN simulations, we observed that our *Ex* Clustering provides longer network lifetime, lower energy consumption, shorter data reporting delay and reliable data reporting ratio than those of the comparison clustering algorithms.

In conclusion, our *Ex*-BCG and its applications reduce the scalability problem of BCGs when used in network systems. The EBTC and *Ex* Clustering algorithms

prove applicability of *Ex*-BCGs for WSNs application by constructing an efficient communication topology and a cluster based data reporting scheme.

6.2 Future Research

Multi-channel assignment for *Ex*-BCG based WSNs

As radio models for a sensor has evolved over the last decades, multi-channel radio is becoming a general choice in WSNs. The IEEE $802.15.4$ (LR-WPAN) standard in [83, 84] defines physical and MAC layers for low cost, low rate personal area networks, especially WSNs, with a multi-channel radio. Differing from a single channel based WSNs where interference is inevitable if more than two data transmissions occur within each other's transmission range at the same time, the multi-channel radio enables sensor nodes to communicate with other nodes without interference if they are using different channel, which enhances communication efficiency by allowing simultaneous data communications [85–87]. Since our *Ex*-BCG based communication topology has a bounded node degree determined by the number of generators, this feature is conducive in designing a multi-channel based routing. Moreover, the communication topology from *Ex*-BCGs has small diameter and short average path length, which improves communication throughput in the multi-channel radio communication.

***Ex*-BCG based Mobile Ad-hoc Networks (MANET)**

In a mobile environment, it is not trivial for the network to maintain network connectivity while its nodes move in and out of range [88–90]. For example, in cellular communication, handling a mobile user moving over two adjacent cells, which is called *handoff*, has been a major research issue [91–95]. Registering a new neighbor as a data relaying node and eliminating a leaving node from a list of neighbors requires real-time detection of topology change. To interact with topology changes without detection failure, mobile nodes should perform periodic or on-demand neighbor discovery to keep their neighbor information up-to-date. To achieve this goal, Phase I (logical neighbor candidate discover) of our EBTC algorithm can be used since it is a relatively simple operation to update logical neighbor information. But more research would be required for designing a dynamic discovery operation, not a sequential one. Moreover, due to frequent topology changes, routing between mobile nodes should be distinguished from static routing.

Bibliography

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Networks (Elsevier)*, vol. 38, pp. 393–422, 2002.

[2] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," in *International Symposium on Communication Theory and Applications (ISCTA 2001)*, (Ambleside, UK), July 2001.

[3] J. Kahn, R. Katz, and K. Pister, "Next century challenges: mobile networking for smart dust," in *Proceedings of the ACM MobiCom99*, (Washington, USA), pp. 271–278, 1999.

[4] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor," in *IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, pp. 607–610, 2000.

[5] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, "Wireless sensor networks for habitat monitoring," in *ACM Workshop on Sensor Networks and Applications*, 2002.

[6] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, (Salt Lake City, UT), May 2001.

[7] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Mobile Computing and Networking*, pp. 263–270, 1999.

[8] C. Herring and S. Kaplan, "Component-based software systems for smart environments," in *IEEE Personal Communications*, pp. 60–61, 2000.

[9] I. Essa, "Ubiquitous sensing for smart and aware environments," in *IEEE Personal Communications*, pp. 47–49, 2000.

[10] G. Wener-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Walsh, "Deploying a wireless sensor network on an active volcano," in *Data-Driven Applications in Sensor Networks (Special Issue)*, Apr 2006.

[11] N. Bouabdallah, M. E. Rivero-Angeles, and B. Sericola, "Continuous monitoring using event-driven reporting for cluster-based wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3460–3479, 2009.

[12] M. Noori and M. Ardakani, "Lifetime analysis of random event-driven clustered wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 10, pp. 1448–1458, 2011.

[13] H. Sabbineni and K. Chakrabarty, "Datacollection in event-driven wireless sensor networks with mobile sinks," *International Journal of Distributed Sensor Networks*, 2010.

[14] H. Li, H. Yu, L. Li, and A. Liu, "On the cascading data collection mechanism in wireless sensor networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 07)*, pp. 2479–2482, Sep 2007.

[15] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *33rd Hawaii International Conference on System Sciences (HICSS00)*, vol. 8, pp. 1–10, Jan. 2000.

[16] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the ACM MobiCom99*, (Seattle, WA), pp. 174–185, 1999.

[17] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of ACM MobiCom00*, (Boston, MA), pp. 56–67, 2000.

[18] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, (Atlanta, GA), pp. 22–31, 2002.

[19] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 878–890, 2005.

[20] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 293–313, 2002.

[21] K. Lu, Y. Qian, M. Guizani, and H.-H. Chen, "A framework for a distributed key management scheme in heterogeneous wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 2, pp. 639–647, 2008.

[22] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 137–145, Oct. 2001.

[23] R. Khalili, D. L. Goeckel, D. Towsley, and A. Swami, "Neighbor discovery with reception status feedback to transmitters," in *the 29th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 10)*, pp. 1–9, March 2010.

[24] P. Dutta and D. Culler, "Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications," in *Sensys'08*, pp. 71–83, Nov. 2008.

[25] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 10)*, pp. 350–361, Apr 2010.

[26] D. Blough, M. Leoncini, G. Resta, and P. Santi, "The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks," *IEEE Trans. Mobile Comput*, vol. 5, no. 9, pp. 1267–1282, 2006.

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

[28] S. Borbash and E. Jennings, "Distributed topology control algorithm for multihop wireless networks," in *Proc. 2002 World Congress on Computational Intelligence (WCCI 2002)*, pp. 355–360, 2002.

[29] C. Perkins and E. Royer, "Ad hoc on demand distance vector (aodv) routing (Internet-draft)," *Mobile Ad-hoc Network (MANET) Working Group, IETF*, 1998.

[30] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," *IEEE/ACM Transactions on Networking (ToN)*, vol. 14, no. 3, pp. 479–491, 2006.

[31] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networks," *Computer Communications Review - Proceedings of SIGCOMM '96*, Aug. 1996.

[32] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pp. 63–72, 2003.

[33] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[34] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: A survey," in *International Conference on Wireless Networks*, vol. 233, pp. 1–7, 2004.

[35] K. Akkaya and M. Younis, "A survey of routing protocols in wireless sensor networks," *Ad Hoc Network (Elsevier)*, vol. 3, pp. 325–349, 2005.

[36] R. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, (Orlando, FL), pp. 350–355, 2002.

[37] Y. M. Lu and V. W. Wong, "An energy-efficient multipath routing protocol for wireless sensor networks," *International Journal of Communication Systems*, vol. 20, no. 7, pp. 747–766, 2007.

[38] X. Mina, S. Wei-rena, J. Chang-jianga, and Z. Yinga, "Energy efficient clustering algorithm for maximizing lifetime of wireless sensor networks," *International Journal of Electronics and Communications*, vol. 64, no. 4, pp. 289–298, 2010.

[39] N. Dimokas, D. Katsaros, and Y. Manolopoulos, "Energy-efficient distributed clustering in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 371–383, 2010.

[40] D. Kumar, T. C. Aserib, and R. Patel, "EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks," *Journal of Computer Communications*, vol. 32, no. 4, pp. 662–667, 2009.

[41] F. Xiangning and S. Yulin, "Improvement on leach protocol of wireless sensor network," in *International Conference on Sensor Technologies and Applications*, pp. 260–264, 2007.

[42] M. J. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *Mobile and Wireless Communications Network, 2002. 4th International Workshop*, pp. 368–372, 2002.

[43] A. Chamam and S. Pierre, "A distributed energy-efficient clustering protocol for wireless sensor networks," *Computers and Electrical Engineering*, vol. 2, no. 36, pp. 303–312, 2010.

[44] D. Kumar, T. C. Aseri, and R. B. Patel, "EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks," *Computer Communications*, vol. 32, no. 4, pp. 662–667, 2009.

[45] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, 2003.

[46] A.-L. Barabási, *Linked*. Plume, 2003.

[47] S. B. Akers and B. Krisnamurthy, "A group-theoretic model for symmetric interconnection networks,," in *Int. Con5 Parallel Processing*, 1986.

[48] D. Cvetkovič and T. Davidovič, "Application of some graph invariants to the analysis of multiprocessor interconnection networks," *Yugosl. J. Oper. Res*, vol. 18, pp. 173–186, 2008.

[49] D. Cvetkovič, T. Davidovič, A. Ilič, and S. K. Simič, "Graphs for small multiprocessor interconnection networks," *Applied Mathematics and Computation*, vol. 217, pp. 2468–2480, 2010.

[50] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An Introduction to Exponential Random Graph (*p**) Models for Social Networks," *Social Networks*, vol. 29, pp. 173–191, 2006.

[51] O. Frank and D. Strauss, "Markov graphs," *Journal of the American Statistical Association*, vol. 81, pp. 831–842, 1986.

[52] A. L. Barabási and E. Bonabeau, "Scale-free networks," *Scientific American*, vol. 288, pp. 60–69, 2003.

[53] A. L. Barabási, E. Bonabeau, and H. Jeong, "Scale-free characteristics of random networks: The topology of the World Wide Web," *Physica A*, vol. 281, pp. 69–77, 2000.

[54] S. Scott and G. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," in *Hot Interconnect Symp. IV*, 1996.

[55] K. W. Tang and S. A. Padubidri, "Diagonal and toroidal mesh networks," *IEEE Trans. Computers*, vol. 43, no. 7, pp. 815–826, 1994.

[56] Y. Liu and Y. Yang, "Reputation propagation and agreement in mobile adhoc networks," in *Proc. IEEE Wireless Communication and Networking Conf.(WCNC)*, vol. 2, pp. 1510–1515, 2003.

[57] K. Sivarajan and R. Ramaswami, "Lightwave networks based on de Bruijn graphs," vol. 2, pp. 70–79, Feb. 1994.

[58] K. N. Sivarajan and R. Ramaswami, "Multihop lightwave networks based on De Bruijn Graphs," in *IEEE INFOCOM*, pp. 1001–1011, 1991.

[59] C. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Computers*, vol. 10, no. 10, pp. 892–901, 1985.

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

[61] K. W. Tang and B. W. Arden, "Representations of Borel Cayley graphs," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 4, pp. 655–676, 1993.

[62] 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.

[63] K. Tang and B. Arden, "Class-congruence property and two-phase routing of Borel Cayley graphs," *IEEE Transactions on Computers*, vol. 44, pp. 1462 –1468, Dec. 1995.

[64] 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.

[65] 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.

[66] J. Yu, *Quasi Borel Cayley Graphs for Ultrafast Information Dissemination in Large and Dense Networks*. PhD thesis, Stony Brook University, Stony Brook, USA, 2011.

[67] B. W. Arden and K. W. Tang, "Representations and routing for Cayley graphs," *IEEE Transactions on Communications*, vol. 39, pp. 1533–1537, Nov. 1991.

[68] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, 2001.

[69] P. Carrington, J. Scott, and S. Wasserman, *Models and Methods in Social Networks Analysis*. Cambridge University Press, 2005.

[70] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 4, 1998.

[71] Z. Wu and R. Wang, "The consensus in multi-agent system with speed-optimized network," *International Journal of Modern Physics*, vol. 23, pp. 2339–2348, 2009.

[72] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," vol. 95, pp. 215–233, Jan. 2007.

[73] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *Proceedings of the American Control Conference (ACC'05)*, vol. 4, pp. 2371–2378, June 8–10, 2005.

[74] R. Olfati-Saber, "Algebraic connectivity ratio of ramanujan graphs," in *Proceedings of the American Control Conference (ACC'07)*, pp. 4619–4624, July 9–13, 2007.

[75] B. Bollobas, *Random Graphs*. Cambridge University Press, 2001.

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

[77] E. Noel and K. W. Tang, "Performance modeling of multihop network subject to uniform and nonuniform geometric traffic," *IEEE/ACM Trans. Networking*, vol. 8, pp. 763–774, 2000.

[78] D. Kim, E. Noel, and K. W. Tang, "Graph theoretic expansion of borel cayley graphs with an optimal and distributed routing algorithm," in *Proceedings of International Performance Computing and Communications Conference (IPCCC 2012)*, pp. 236–245, Dec. 2012.

[79] D. Kim, E. Noel, and K. W. Tang, "Expanded Borel Cayley Graphs (ExBCGs): A Novel Communication Topology for Multi-Agent Systems," *Journal of Network and Computer Applications*, vol. 37, pp. 47–61, 2014.

[80] M. Newman, *Networks*. Oxford University Press, 2010.

[81] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, Nov 2003.

[82] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks.," in *Proceedings of International Conference of Distributed Computing Systems*, (Vienna, Austria), July 2002.

[83] *Standard for Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*. New York, IEEE Press, 2003.

[84] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[85] Özlem Durmaz Incel, *Multi-channel wireless sensor networks: protocols, design and evaluation*. University of Twente, 2009.

[86] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *Proceedings of The 27th Conference on Computer Communications (INFOCOM '08)*, pp. 1193–1201, Apr 2008.

[87] H. K. Le, D. Henriksson, and T. Abdelzaher, "A practical multi-channel media access control protocol for wireless sensor networks," in *Proceedings of The 7th international conference on Information processing in sensor networks (IPSN '08)*, pp. 70–81, Apr 2008.

[88] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-layering in mobile ad hoc network design," *Computer*, vol. 37, no. 2, pp. 48–51, 2004.

[89] S. Nesargi and R. Prakash, "MANETconf: Configuration of hosts in a mobile ad hoc network," in *Proceedings of The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1059–1068, 2002.

[90] M. Mohsin and R. Prakash, "IP address assignment in a mobile ad hoc network," in *Proceedings of The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 856–861, 2002.

[91] D. Hong and S. S. Rappaport, "Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures," *IEEE Transactions on Vehicular Technology*, vol. 35, no. 3, pp. 77–92, 1986.

[92] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *IEEE Transactions on Wireless Networks*, vol. 1, no. 3, pp. 469–481, 1995.

[93] N. D. Tripathi, J. H. Reed, and H. F. VanLandinoham, "Handoff in cellular systems," *IEEE Transactions on Personal Communications*, vol. 6, no. 5, pp. 26–37, 1998.

[94] M.-H. Chiu and M. A. Bassiouni, "Predictive schemes for handoff prioritization in cellular networks based on mobile positioning," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 510–522, 2000.

[95] J. M. Holtzman and A. Sampath, "Adaptive averaging methodology for handoffs in cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 44, no. 1, pp. 59–66, 1995.