Broadcasting in Hyper-cylinder graphs

Aria Adibi

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

October 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Aria Adibi**
Entitled: **Broadcasting in Hyper-cylinder graphs**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
Dr. Denis Pankratov

______________________ External

______________________ Examiner
Dr. Goswami

______________________ Examiner
Dr. Pankratov

______________________ Thesis Supervisor
Dr. Hovhannes Harutyunyan

Approved by ______________________
Dr. Leila Kosseim, Graduate Program Director

__________ ______________________
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Broadcasting in Hyper-cylinder graphs

Aria Adibi

Broadcasting in computer networking means the dissemination of information, which is known initially only at some nodes, to all network members. The goal is to inform every node in the minimal time possible. There are few models for broadcasting; the simplest and the historical model is called the *Classical model*. In the Classical model, dissemination happens in synchronous rounds, wherein a node may only inform one of its neighbors. The broadcast question is: What is the minimum number of rounds needed for broadcasting, and what broadcast scheme achieves it?

For general graphs, these questions are NP-hard, and it is known to be at least $3 - \epsilon$ inapproximable for any real $\epsilon > 0$. Even for some very restricted classes of graphs, the questions remain as an NP-hard problem. Little is known about broadcasting in restricted graphs, and only a few classes have a polynomial solution.

Parallel and distributed computing is one of the important domains which relies on efficient broadcasting. Hypercube and torus are the most used network topology in this domain. The widespread use is not only due to their simplicity but also is for their efficiency and high robustness (e.g., fault tolerance) while having an acceptable number of links. In this thesis, it is observed that the Cartesian product of a number of path and cycle graphs produces a valuable set of topologies, we called *hyper-cylinders*, which contain hypercube and Torus as well. Any hyper-cylinder shares many of the beneficial features of hypercube and torus and might be a suitable substitution in some cases. Some hyper-cylinders are also similar to other practically used topologies such as cube-connected cycles. In this thesis, the effect of the Cartesian product on broadcasting and broadcasting of hyper-cylinders under the Classical and Messy models is studied. This will add a valuable class of graphs to the limited classes of graphs which have a polynomially computable broadcast time. In the end, the relation between worst-case originators and diameters in trees is studied, which may help in the broadcast study of a larger class of graphs where any tree is allowed instead of a path in the Cartesian product.

Acknowledgments

I would like to sincerely thank my supervisor, professor Hovhannes Harutyunyan, for all his support and for helping me from the very beginning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To *broadcast* information is to transmit that information to many receivers. Three examples are a radio station broadcasting a signal to many receivers; Wi-Fi networks broadcasting their SSID[1] to all nearby wireless devices; and in computer networking, data packets being sent to all recipients all at once.

In computer networking, broadcasting is one of the fundamental operations and may be performed at any level of the network. For instance, as a high-level operation, some applications use broadcasting to share information or to coordinate between clients; as a middle-level operation, DHCP[2] clients use broadcasting to assign IP[3] addresses for networks running IPv4[4,5]; and as a low-level operation, broadcasting is used in Ethernet to resolve IP addresses to MAC addresses [6].

Several papers have been published about broadcasting; however, as is the case for any design in computing, the results function under different hypotheses and objectives, making some of them incomparable. To understand different hypotheses and to further illustrate the

[1] Service Set IDentifier
[2] Dynamic Host Configuration Protocol
[3] Internet Protocol
[4] Internet Protocol version 4, the network protocol used by most of today's internet
[5] In newer protocol IPv6 broadcasting is replaced by a more versatile method called multicasting.
[6] Media Access Control address

practicality of broadcasting, let us first narrow our discussion to an exceptionally important context for which efficient broadcasting is essential, namely, parallel and distributed computing architecture. Pierre Fraigniaud and Emmanuel Lazard have done a marvelous job in [45] providing a taxonomic framework and giving a survey of existing results in broadcasting. Their work influenced me in structuring and writing this introduction.

1.1 Parallel and Distributed Computing Architectures

Parallel and distributed computing systems are mainly composed of interconnected general-purpose processors, whose performance depends on both the computational power of the processors and the way they interact. If the processors are all linked to a shared memory, the main issues to solve are those concerning the concurrent access to the memory; however, if memory is distributed, data movements will depend on the topology of these connections. While some design variants are dedicated to a more niche type of computing, let us consider the more general distributed memory multiprocessors systems that can be connected using different topologies. The following is a quote form [45] explaining the typical architecture of such systems.

> Memories and processors are connected by a point-to-point interconnection networks; described in terms of nodes and links. A *node* in a typical network architecture consists of a processor, a memory, a fast bus, and several direct memory access (DMA) channels (see Figure 1). Each DMA channel connects the node to one of its neighbors. The memory and DMA channels are all connected to the fast bus, and the processor is connected to the memory. A processor communicates with a neighbor by writing the information in its memory. The information is then transmitted by the appropriate DMA channel via the bus to the neighbour's memory via its bus. This communication path

between two DMA-channels will be called a *link*.

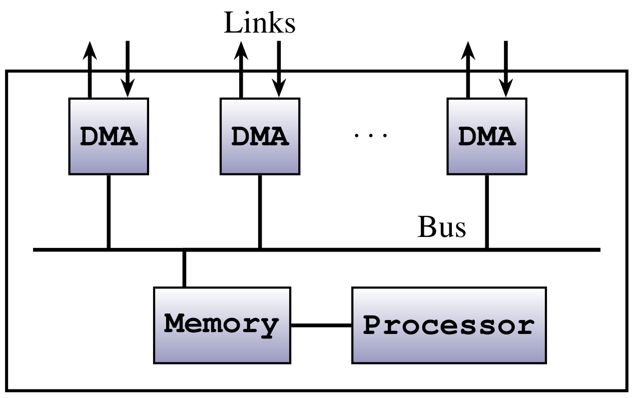


Figure 1: Architecture of a Node

1.1.1 Choice of the Topology

A good topology should strike a good enough balance between cost and performance based on the system's objectives. Examples of costs are, looking for the minimum number of wires (e.g., for financial reasons); efficient layout of the topology (e.g., wanting it to be efficiently planar for wire crossing issues); simple routing algorithms; simple construction; etc. On the performance side, examples are small diameter, extendability, short wires and redundant paths (e.g., for fault resistance, etc. In chapter 3, the topologies are discussed in more depth, and *d-dimensional (Directed) Hyper-cylinder* topologies are introduced, which is the main topic of this thesis.

1.1.2 Message Relaying Method

In this thesis, it is assumed that the relaying of the messages is done in *store-and-forward* (or *packet-switch*) fashion which means a node cannot send a message unless all bits of it are received. This is in contrast to *circuit-switched* (or *wormhole*) mode. Packet-switched networks move data in separate, small blocks – packets – based on the destination address in each packet. When received, packets are reassembled in the proper sequence to make up the message. Circuit-switched networks, on the other hand, require dedicated point-to-point connections during calls. The reasons for this choice are (1) any circuit-switched machine can also perform store-and-forward mode, (2) circuit-switch is a better option if uninterrupted connections are of utmost importance and the number of nodes is limited, which is an atypical situation; for other situations, packet-switch is more advantageous and preferred, (3) only a few algorithms have been found that work better in broadcasting under circuit-switch mode and the existence of many such algorithms have been deemed unlikely [97].

The communication links are usually made of *simplex* communication channels, which send information in one direction only. If the bidirectional flow is needed, two simplex channels operating in the opposite direction, called *duplex* communication channel, are used. There are two types of duplex communication channels (1) full-duplex and (2) half-duplex. If both processors can communicate with each other simultaneously, the link is *full-duplex*; otherwise, if both can communicate with each other but not simultaneously, the link is said to be *half-duplex*.

1.1.3 Communication Time and Synchronicity

Discussions about broadcasting time require a mathematical model for measuring time. Let communication time T be the time it takes to send a message through a link in the network. Many experiments have shown that T greatly depends on the length L of the message [19, 23, 74, 98]. For this reason, the link communication time is usually *modeled linearly* with L as the sole dependent variable:

$$T = \beta + L\tau$$

Other physical attributes of the link and network used in this equation are *latency* or *start-up time* β, and *data transfer time per element* or *propagation time* τ ($1/\tau$ is the *bandwidth*).

Another factor in the overall time measurement is whether the network uses synchronous or asynchronous transmissions. *Synchronous* transmissions are synchronized by an external clock, while *asynchronous* transmissions are synchronized by special signals along the transmission medium.

If the lengths of the messages is small, one can approximate T to a constant value. Taking this constant as the unit of time, $T = 1$ may be assumed. This makes the message relays behave synchronously; if no intentional delay is in place. Hence, no need to compare synchronous and asynchronous behaviors here. The fact that most applications of broadcasting send small messages for coordination purposes; makes this simplification a pragmatic one as well. Moreover, T was traditionally defined like this; more on this in subsection 1.2.1, The Origin.

1.1.4 Bottlenecks

Two of the main objectives studied for broadcasting are the time it takes to broadcast and the schemes used to achieve it; formally defined in section 1.4, Formal Definitions. These objectives are highly dependent on the bottlenecks of the network system. The three usual categories are:

1. If the system's nodes can only use one link for communication at each time unit, the system is said to be *processor-bound*, also called *1-port* or *whispering*.

2. In contrast, if the system's nodes can use all the available links, the system is called *link-bound*. This is because now, the number of links is the bottleneck. Other names are *n-ports*, *shouting*, or *flooding*.

3. In between, if the system's nodes can utilize k links at the same time, the system is called *DMA-bound*. This is because one scenario in which this behavior is possible is when the number of DMAs that can use the fast bus is limited. Although, it is also

possible that the system's nodes' internal links are bounded.

The study of broadcasting in link-bound systems is relatively simple. In this thesis, the processor-bound bottleneck is assumed.

1.2 Abstraction, Graph Notation

In section 1.1, one context in which broadcasting is used was given. One can abstract away many technicalities and work on a more general mathematical model framework, broadening the scope of contexts in which broadcasting is exploited. Usually, this abstraction is done through graph theory notions. Other models have also been used; for instance, matrix representation [106, 109], and integer/linear programming [8, 92, 95].

A simple connected graph G can model the communicating objects as vertices and the possible communications as edges. In parallel and distributed computing, a vertex corresponds to the network node and edges to network links. Simplex link is usually modeled with a directed edge, while full-duplex links with symmetric directed edges and half-duplex links with an undirected edge. In some contexts, it is natural for the communication links to be dynamic and frequently changing. For these cases, dynamic graphs (sequences of graphs) may be used to study the broadcasting; an example of such treatment is [49]. In this work, we only consider cases where links are not dynamic.

If v and u are two vertices, the edge between v and u is denoted with uv. If the edge is directed, the order describes the direction; from u to v. Otherwise, both uv and vu are the representation of the same edge. The *distance* between u and v is the length of the shortest path between u and v. The *diameter* is the greatest distance between vertices.

Convention 1. Sometimes, notations are slightly abused to minimize the repetition of ideas and definitions caused by slight technical differences between undirected and directed graphs. In addition to the same notation being used for directed and undirected edges explained above, the notation for classes of graphs like paths and cycles are also the same for directed and undirected graphs. Overall, there should be no confusion based on the context, or it will be explicitly clarified.

1.2.1 The Origin

In the 20th century, when the problem was first defined, parallel and distributed computing was not very common, and communication algorithms were studied with a graph-theoretic approach. In [41], Farley et al. first defined broadcasting alongside *minimum broadcast time* and *minimum broadcast graph* problems using graph-theoretic approach; the formal definition of which is provided in section 1.4. The following question, asked by A. Boyd, [52], also sparked some exuberance in the research of broadcasting and similar problems.

> “There are n ladies, and each one of them knows an item of scandal which is not known to any of the others. They communicate by telephone, and whenever two ladies make a call, they pass on to each other as much scandal as they know at that time. How many calls are needed before all the ladies know all the scandal?”

This problem was enjoyed and popularized with several mathematicians at the time and was called *gossiping* or *telephone problem*. Broadcasting and gossiping are similar concepts; [64] have studied the relation between solving for minimum broadcasting time and gossiping time for some families of graphs. In both problems, the constraints are similar to that of a (1) store-and-forward, (2) processor-bound communication with (3) communication-time $T = 1$ (and as a result synchronous, as explained in subsection 1.1.3). This model is still frequently used and studied not only because of its simplicity and theoretical graph properties but also for its fastidious modeling of communications in which small messages are exchanged. This model is called the *Classical model*, which is the hypothesis of this thesis in chapter 4.

1.3 Similar Problems

As was seen in The Origin, gossiping and broadcasting are similar problems; however, gossiping is not the only important problem resembling broadcasting. The following is a non-comprehensive list of pertinent procedures that resemble broadcasting and broadcasting:

- *Routing*: A node wishes to send a piece of information to another node. (*One-to-One*)
- *Broadcasting*: A node wishes to send a piece of information to all the other nodes. (*One-to-All*)
- *Multi-casting*: A node wishes to send a piece of information to some of the other nodes. (*One-to-Many*)
- *Scattering*: A node has different pieces of information for different nodes it wishes to sent. (*Personalized One-to-All*)
- *Gossiping*: All nodes have a piece of information that they need to send to every other node. (*All-to-All*)

1.4 Formal Definitions

The definitions are inspired by [51, Sec. 12.2].

Note that unlike in gossiping, there is no reason to differentiate between full and half-duplex links in broadcasting. This is because only one-directional flow between any pair of vertices suffices. For this reason, one can assume that the representative graph is directed or undirected and not hybrid; by modeling half-duplex links as symmetric directed edge, just like full-duplex representation.

Definition 1.4.1 (Broadcasting)**.** Broadcasting is the goal of sending a single message from a particular vertex (called the **originator**) to all of the other vertices in the graph.

Definition 1.4.2 (Broadcast protocol for an originator u, broadcast protocol tree)**.** The broadcast protocol is an algorithm describing a scheme used to broadcast information from u. Formally it is represented on the graph G by at most one positive integer labeling of edges representing that link being used at that time-unit after the process has started. This labeling should also satisfy

1. The set of edge labels incident to any vertex are disjoint.
2. There is exactly one path with increasing edge labels from u to any other vertex in G, forming a tree. This tree is called the *broadcast protocol tree*.

Definition 1.4.3 (Non-lazy broadcast protocol)**.** *Non-lazy broadcast protocol* is a broadcast protocol in which each vertex after receiving the information is not idle in sending the information unless there is no more sending to do. Alternatively, (1) the set of edge labels incident to any vertex consists of consecutive integers, and (2) one edge of the originator should be labeled with 1.

Definition 1.4.4 (Informed at time t, completion time)**.** A vertex v is *informed at time* t by a broadcast protocol for originator u if the last edge in the path from u to v is labeled with t. The *completion time* of a broadcast protocol for originator u is the largest edge label used in that broadcast protocol.

Definition 1.4.5 (Broadcast time of originator u, $b(u)$)**.** In graph G, *broadcast time of originator* u, is the *minimum* completion time that can be obtained from any broadcast protocol from u. It is denoted by $b(G, u)$ or $b_G(u)$; however, whenever there is no ambiguity about G it is denoted by $b(u)$.

It is clear that for every minimum broadcast protocol of any vertex, there exists another (not necessarily distinct) with the same completion time, which is non-lazy.

Definition 1.4.6 (Broadcast time of a graph G, $b(G)$)**.** The broadcast time of a graph G is the *maximum* of broadcast time of all the vertices.

Example 1.4.1. *Figure 2 shows an example of two broadcast protocols with u as the originator. In this graph $b(G) \geq \lceil \log_2(7) \rceil = 3$; hence, $b(u) = 3$. By symmetry, it is easy to verify that any vertex is able to disseminate a piece of information in three time-units; therefore, $b(G) = 3$.*

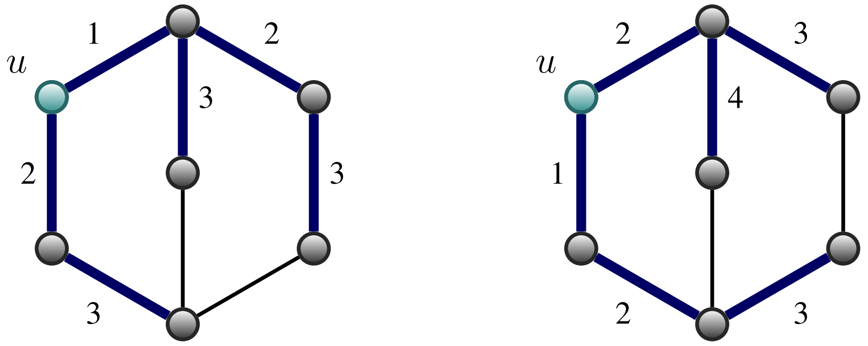


Figure 2: Two broadcast protocols with u as the originator; $b(u) = 3$ and $b(G) = 3$. Courtesy of [51, Sec. 12.2].

Definition 1.4.7 (Broadcast center of a graph G, $BC(G)$)**.** The broadcast center of a graph G is the set of vertices having the minimum broadcast time.

Example 1.4.2. *Figure 3 indicates the broadcast center (BC) in the example graph. The broadcast time of BC vertices is 6, while every other vertex has a larger broadcast time. For example, $b(x) = 7$.*

At each time-unit in the Classical model, the number of informed vertices cannot exceed double the previous amount; for this reason $b(G) \geq \lceil \log_2(|V|) \rceil$. It is easy to see that this bound is achieved by complete graph K_n (see subsection 2.2.3); yet, K_n does not have the minimum number of edges in all the graphs with n vertices that achieves this minimum. For example, the graph obtained by removing any edge from K_n also achieves this bound.

Definition 1.4.8 (Broadcast function, $B(n)$)**.** The *broadcast function, $B(n)$*, is the minimum number of edges in any graph G with n vertices such that $b(G) = \lceil \log_2(n) \rceil$.

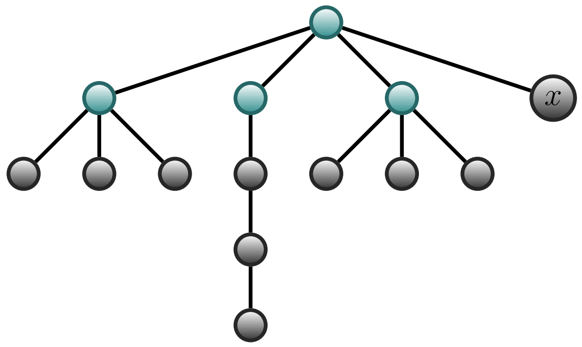


Figure 3: Light teal-colored vertices are the broadcast center (BC) of this graph. The broadcast time of BC vertices is 6, while other vertices have a larger broadcast time. For example, $b(x) = 7$.

Definition 1.4.9 (Minimum Broadcast Graph (mbg))**.** A *minimum broadcast graph (mbg)* is a graph G with n vertices and $B(n)$ edges such that $b(G) = \lceil \log_2(n) \rceil$.

Minimum broadcast graphs represent the cheapest possible communication networks (having the fewest communication links) in which broadcasting can be accomplished, from any vertex, as fast as theoretically possible.

1.5 Messy Models

In a network that knows how to broadcast in minimum time, it is tacitly assumed that either

1. an omniscient being is coordinating the actions of the vertices, or
2. each vertex has a set of coordinated actions (or can compute it) optimized for each originator. In this case, the vertices must have enough storage space and must also be able to determine the originator.

However, in many situations, this assumption is not desirable or realistic. For instance, the designers wanting the topology to be independently expandable, or they do not like to use high memory or processing power due to many costs, including but not limited to: high financial price, cyber security reasons, and even susceptibility to theft; etc. An example in point is the TCP/IP internet network currently in use.

It is, therefore, natural that nodes in networks have local knowledge of the network rather than a global one. Similar assumptions are made in [11], where the topology is not known; and in [6, 7, 9], where broadcasting with the same assumption in a particular type of networks called *radio networks* is studied. The exact such models used in this thesis were first defined in [3], named *Messy models*. In this model, the nodes do not know the state of the network, the originator, or the start time of the broadcasting. Moreover, they are independent agents with a limited view of their neighbors. There are three slightly different Messy models:

Definition 1.5.1 (Model M_1)**.** Each vertex knows the state of its neighbors at any time-unit, informed or uninformed. Each informed vertex must transmit the broadcast message to one of the uninformed neighbors, if any, in each time-unit.

Definition 1.5.2 (Model M_2)**.** Every informed vertex knows from which neighbor (neighbors) it received the broadcast message and to which neighbors it has sent the message. Each informed vertex must transmit the broadcast message to all the neighbors except the ones it knows are informed.

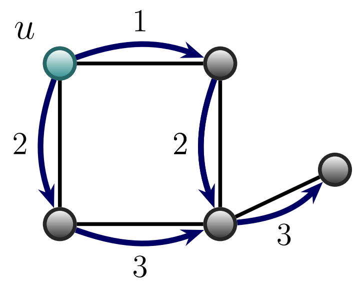
Definition 1.5.3 (Model M_3)**.** Each vertex knows to which neighbors it has sent the broadcast message. Each informed vertex must transmit the broadcast message to all the neighbors except the ones it knows are informed.

With the local knowledge, it is not natural to ask about the minimum broadcast time, but rather it is natural to ask what is the **worst case time** or broadcast scheme.

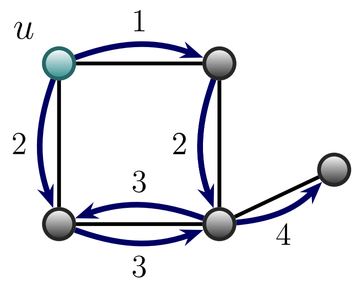
Definition 1.5.4 (Broadcast time of vertex u)**.** The *broadcast time of vertex* u in graph G using model M_i, denoted $t_i(G, u)$ or simply $t_i(u)$, for $i = 1, 2, 3$, is the **maximum** number of time-units required for the broadcast to inform all the vertices.

Definition 1.5.5 (Broadcast time of graph G)**.** The *broadcast time of graph* G using model M_i, denoted $t_i(G)$, for $i = 1, 2, 3$ is the maximum broadcast time for all vertices u of G; $t_i(G) = \max\{t_i(u) | u \in V(G)\}$.

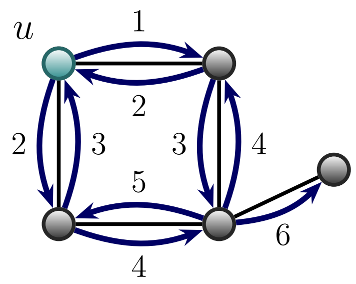
Example 1.5.1. *Figure 4 provide an example and highlights the difference between the three Messy models.*



(a) $t_1(u) \geq 3$



(b) $t_2(u) \geq 4$



(c) $t_3(u) \geq 6$

Figure 4: An example showing the difference between Messy models. The drawn scheme are optimal. It is easy to verify that $t_1(u) = 3$, $t_2(u) = 4$, $t_3(u) = 6$, and also $t_1(G) = 3$, $t_2(G) = 4$, $t_3(G) = 6$.

One might think of a more generalized version, in which instead of one level knowledge (only neighbors), each vertex has k levels knowledge. Informally, in most cases of concern, replacing an edge with a path of length k creates graphs with similar difficulties faced with the original graph with level one knowledge. Accordingly, the knowledge gained is not particularly useful. In [5], this difference was better analyzed, albeit the model there is slightly different than ours. Chapter 5 uses Messy models as its hypotheses.

1.6 Outline

The main objective of this thesis is to analyze broadcasting in the valuable class of topologies *d-dimensional (Directed) Hyper-cylinder* under Classical and Messy models. Consequently, hyper-cylinders will be added to the limited arsenal of graph classes for which broadcast time is known to be polynomially computable. Chapter 2 is the literature review where the “hardness” of finding the answer to minimum broadcast time, both in terms of NP-hardness and inapproximability, is explained. It also compiles the known results for particular classes of graphs, especially the known practical topologies. In chapter 3 some preliminary results regarding the effects of *Cartesian product* on broadcasting are provided, and then motivations and aims of this thesis are laid out; including the definition of Hypercylinder graphs. The thesis studies the polynomiality of the broadcast problem in hyper-cylinder graphs, expanding the classes of graphs for which the broadcast problem is polynomially computable. Chapters 4 and 5 explore the newly found results and conjectures in details within Classical and Messy models, respectively. Chapter 6 provides some results in regards to the relation between worst-case originators and diameters in trees, which are missing from the literature. This finding may help to generalize the result to a more general class of graphs. Finally, in chapter 7 all conclusions and conjectures are summarized, and few suggestions for future studies are suggested.

Chapter 2

Literature Review

Sources [45, 51, 62, 65] which include survey papers and book chapters are great resources for compilation of main results of broadcasting.

2.1 A Hard Problem

It turns out that finding $b(u)$ for an arbitrary vertex in graph G is a difficult task, both in terms of NP-hardness and lower bounds of approximation algorithms.

NP-hardness. D. S. Johnson proved that finding minimum broadcast time is NP-complete, and with his permission, the result was published by Slater et al. in [100]. Let us call the problem of finding the minimum broadcast time with multiple originators MB and a single originator SB problem.

Theorem 2.1.1 ([100])**.** *MB is NP-complete even with deadline $T \geq 4$.*

Theorem 2.1.2 ([100])**.** *SB is NP-complete even with deadline of $O(\sqrt[3]{|V|})$.*

The broadcast time of trees, however, can be found in linear time. The same paper [100] provides an algorithm to achieve this. Therefore, the question is, how complex is the problem for more restricted cases?

Restricting the graph to be planar and even with a very small bounded degree does not make the problem any easier. The best-known result in this matter are:

Theorem 2.1.3 ([67]). *MB restricted to 3-regular planar graphs and a deadline 2 is NP-complete.*

Theorem 2.1.4 ([85]). *SB restricted to planar graphs of degree 3 is NP-complete (the deadline grows like $\sqrt{|V|}$).*

NP-completeness of several restricted simple graphs for MB have also been proven [68]; namely, bipartite planar graphs, grid graphs, complete grid graphs[1], split graph[2], and chordal graph[3].

Polynomially computable. Only a few classes of graphs are known for which broadcast problem is polynomially solvable.

Many combinatorial optimization problems for graphs can be solved polynomially when the graph has bounded tree-widths or small connectivity (for examples, refer to [4] and [93].) In this respect, the broadcasting problem seems to be more complicated since it does not have finite-state property or a bounded number of equivalent classes (see [4] and [93] for the meaning); consequently, the methods used in [4] and [93] are not directly applicable. Despite this difficulty, [67] used a modified framework developed in [93] to define a Node and an Edge decomposition for graphs and presented a broadcasting algorithm. This algorithm proves polynomiality for restricted versions of Node and Edge decompositions and provides an intuitive sense of what makes the broadcasting problem difficult.

To the best of our knowledge, the exhaustive list of such classes are,

[1]The definition of grid graphs here differs from the definition in section 3.1. Here the definition is a planar graph with grid-like edges. Complete grid graph would be the same $2D$ version of grid defined in this work.

[2]A *split graph* is a graph in which the vertices can be partitioned into a clique and an independent set and possibly some edges in between.

[3]A *chordal graph* is one in which all cycles of four or more vertices have a chord.

1. Trees [100]

2. Tree of cycles; [59] Tree of cliques [60].

3. Unicyclic graphs; [56, 57] Fully connected trees [58]

4. Necklace graph [53],

5. k-cactus graph where each vertex is common between at most k cycles, with k being a constant [22] It is strongly believed, yet not proved, by our work that cactus graphs in general are NP-hard; main difficulty may be observed at [17]

6. Restricted versions of certain Node and Edge decomposition. [67]

7. Some basic usual and practical network topologies presented in section 2.2, Usual Topologies.

Approximation algorithms. Best general approximation algorithm known is $O\left(\frac{\log(n)}{\log(\log(n))}\right)$-approximation for minimum broadcasting on a n-node graph [38].

Two other influential approximation papers are: (1) an $O\left(\frac{\log^2(n)}{\log(\log(n))}\right)$-approximation

Theorem 2.1.5 ([92])**.** *There is an $O\left(nmlog^2 n\right)$-time algorithm that, given an undirected graph G on n nodes with m edges and a source node r, computes a broadcast scheme that completes in time $O\left(b(r)\frac{\log^2(n)}{\log(\log(n))}\right)$.*

and (2) an *additive* approximation algorithm [73], which computes a scheme that finishes in the minimum broadcast time plus an *additive factor* of $O\left(\sqrt{n}\right)$.

In [43] the authors analyzed the randomized broadcasting scheme in general graphs and gave the expected and almost sure coverage time for such scheme.

Inapproximability. Best general inapproximability lower bound so far is due to [39], which states:

Theorem 2.1.6 ([39])**.** *For undirected graphs (resp. directed) broadcast problem is NP-hard (resp., impossible unless $NP \subseteq DTIME\left(n^{O(\log n)}\right)$) to approximate it within a ratio of $3-\epsilon$ for any $\epsilon > 0$ (resp., $\Omega\left(\sqrt{\log n}\right)$.)*

The authors also provided a *combinatorial algorithmic* approximation with some advantages, which was missing in previous (mostly linear programming based) good approximation. The most important benefit is a better insight of the structure of the solution.

Another notable, yet generally weaker, inapproximability proof is due to [96], which proves the NP-hardness to distinguish between graphs $G = (V, E)$ with broadcasting time smaller than $b \in \theta(|V|)$ and larger than $(\frac{57}{56} - \epsilon)b$ for any $\epsilon > 0$. In this paper; inapproximability for 3-regular graphs was also proven for both MB and SB.

Theorem 2.1.7 ([96])**.** *For ternary graphs it is NP-hard to decide whether the broadcasting time is $b \in \theta(\log|V|)$ or $b+\theta(\sqrt{b})$ in the case of multiples sources. For ternary networks with single sources, it is NP-hard to distinguish between graphs with broadcasting time smaller than $b \in \theta(|V|)$ and larger than $b + c\sqrt{\log b}$.*

As a result of this difficulties some research has been made in approximation or heuristic algorithm (usually with no theoretical guarantee) to determine the broadcasting (see [10, 47, 48, 61, 95].) Another direction is to limit the study to a particular class of graphs, as it is the case in this thesis.

2.2 Usual Topologies

2.2.1 Path P_n

Path P_n is a sequence of n connected vertices, see Figure 5. The diameter is $n-1$.

For the originator to obtain optimal time, it should send the information to the longer side first. After that decision non-lazy scheme is unique. Accordingly, $b(P_n) = n - 1$. In computer networking, "buses" are mostly modeled with paths.

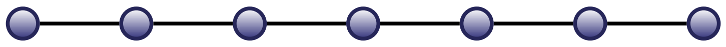


Figure 5: Path P_n for $n = 7$.

2.2.2 Cycle C_n

Cycle C_n is a wrap-around sequence of n connected vertices; see Figure 6. The diameter is $\lfloor n/2 \rfloor$.

After the first message relay, moves are uniquely determined in any non-lazy schemes. Due to symmetry, it does not matter which direction the first move is. Consequently, $b(u)$ is the same for all vertices and $b(C_n) = \lceil n/2 \rceil$. In computer networking, "rings" are mostly modeled with cycles.

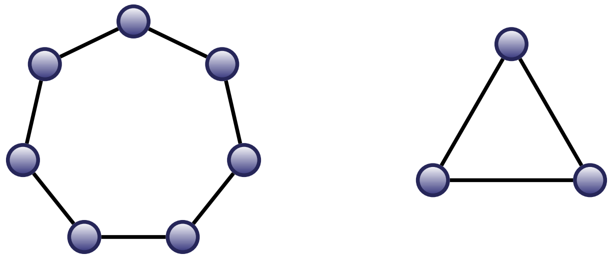


Figure 6: Cycle (Ring) C_n for $n = 7$ and $n = 3$.

2.2.3 Complete Graph K_n

In Complete graph K_n, all n vertices are linked together; see Figure 7. The diameter is 1.

Let the name of the vertices be from 0 to $n - 1$, with 0 as the originator. At step i have informed vertex p send a message to vertex $2^{i-1} + p$. In this way it can be seen that $b(K_n) = \lceil \log_2(n) \rceil$. The diameter is one, but the number of edges is far too high to be practical in a computing network connection when n is big enough. For small n; however, it might be beneficial to use K_n; for example, [46] uses such interconnection.

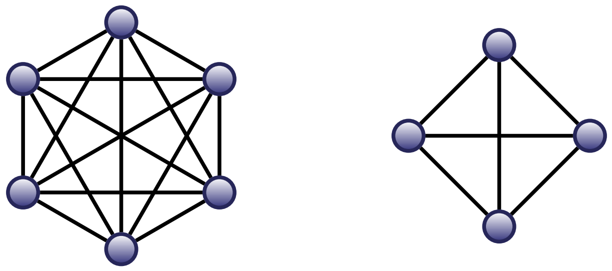


Figure 7: Complete graphs K_n for $n = 6$ and $n = 4$.

2.2.4 Hypercube H_d

Hypercube H_d is a graph recursively defined as follows. A 0-dimensional hypercube H_0 is a vertex. A $d + 1$-dimensional hypercube H_{d+1} consists of two copies of H_d, where same vertices in each is connected to each other; see Figure 8.

Hypercube is a minimum broadcast graph; therefore, $b(H_d) = d$. One optimal scheme is the following: at step i have informed vertex p send the message in its ith-dimention neighbor, the connecting edges for building H_i from two H_{i-1}.

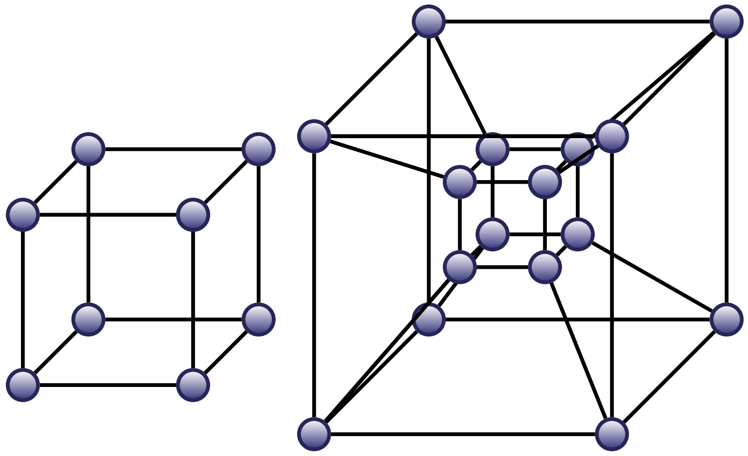


Figure 8: Hypercube graphs H_d for dimensions $d = 3$ and $d = 4$.

The hypercube is used in many parallel computers due to its many features, including but not limited to small diameter $\log_2 n$ (n is the number of vertices), fault tolerance capabilities, and easily simulating other topologies. For this reasons, many research have been done on hypercube for different objectives; few related examples are [18, 26, 94, 99].

Some researchers tried to build or find graphs with lower edges while preserving some of the features that hypercube provides; the following Cube-Connected Cycles CCC_d and De Bruijn Network $UB(d, D)$ are examples of such attempts.

2.2.5 Cube-Connected Cycles CCC_d

Cube-connected cycles graph of order d, CCC_d, can be defined as a graph with $d2^d$ vertices, indexed by pair of numbers (c, p) where $0 \leq c < 2^d$ and $0 \leq p \leq d$. Vertex (c, p) is connected to three neighbor: $(x,\ (y+1) \mod d)$, $(x,\ (y-1) \mod d)$, and $(x \oplus y,\ y)$, where $\oplus$ denotes the bitwise exclusive or operation on binary numbers; see Figure 9. Diameter is $2d + \lfloor d/2 \rfloor - 2$, for $d > 3$ [84].

By first relaying the message through hypercube neighbors and then through the cycle it can be seen that $b(CCC_d) = \lceil 5d/2 \rceil - 1$. Cube-connected cycles CCC_d can be viewed as d-dimensional hypercube H_d where every vertex is replaced by cycle of d. For this reason, as mentioned, cube-connected cycles have been proposed as a good alternative for hypercube for a large class of algorithms on general-purpose parallel processors with the additional benefit of having a small bounded degree (3 regular) [89].

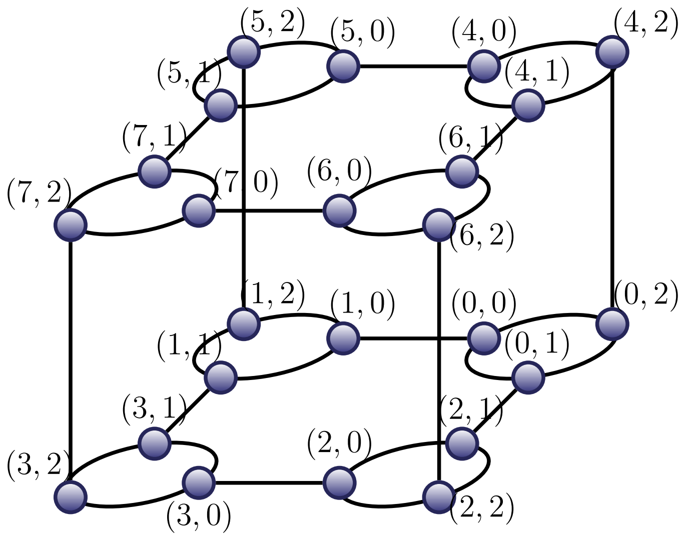


Figure 9: Cube-connected Cycle graph CCC_d for $d = 3$.

2.2.6 De Bruijn Network $UB(d, D)$

The de Bruijn Network $UB(d, D)$ is a digraph with indegree and outdegree d and diameter D with d^D vertices. The vertices are words of length D on an alphabet of d letters. There is an edge from v to u, called *shuffle-edge*, if and only if last $D - 1$ letters of v is the same as first $D - 1$ letters of u; see Figure 10. The undirected de Bruijn Network is obtained from removing the orientations of the edges.

The de Bruijn Network has been proposed and studied as a good competitor for hypercube [15]. Broadcasting has been studied on the directed version, and some bounds have been found; interested readers may refer to [14, 63, 70].

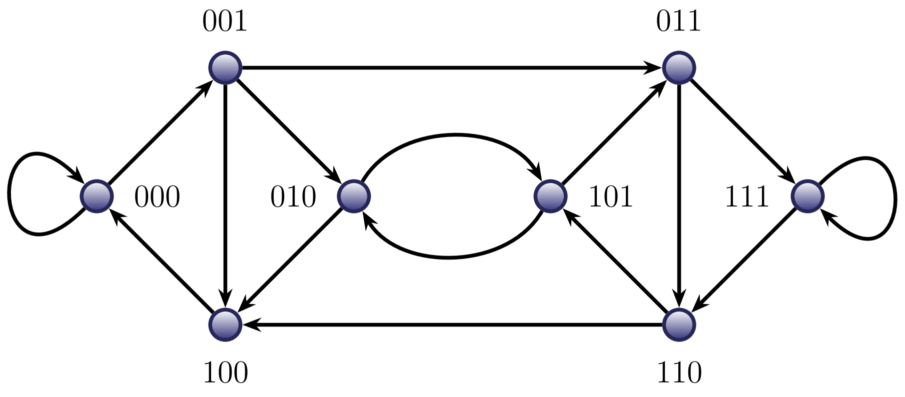


Figure 10: De Bruijn Network $UB(2, 3)$.

2.2.7 Shuffle-Exchange Network SE_d

Shuffle-Exchange Network SE_d is a 3-regular graph with 2^d vertices. The vertices are d-bits strings $x = x_0 \dots x_{d-1}$ whose neighbors are $x_1 \dots x_{d-1}x_0$ and $x_{d-1}x_0 \dots x_{d-2}$ via *shuffle* edge and to $x_0 \dots \overline{x_{d-1}}$ via *exchange* edge; see Figure 11. Diameter is known to be $2d - 1$.

It has been shown in [37] that $b(SE_d) = 2d - 1$. A small number of steps for routing and the existence of fast parallel algorithms for basic problems like sorting, matrix multiplication, polynomial evaluation, and Fourier transforms are some of the advantages of this

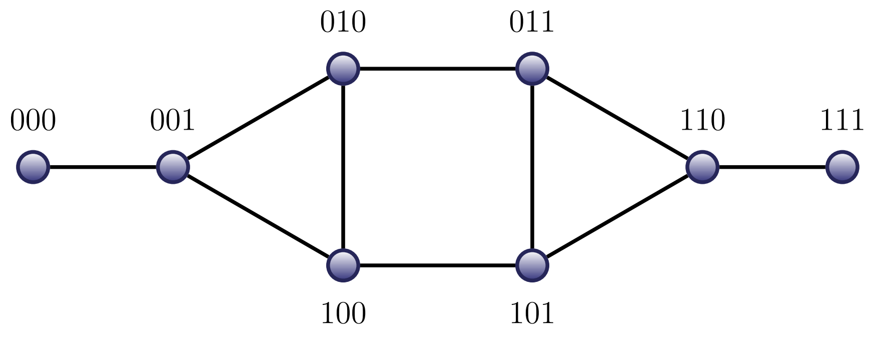


Figure 11: Shuffle-Exchange Network SE_3.

network [77, 78].

2.2.8 Butterfly Network BF_d

Butterfly Network BF_d is a graph with $d2^d$ vertices, diameter $\lfloor 3d/2 \rfloor$, and maximum degree 4. The vertices are labeled with a pair of numbers (l, x), l is called the level $(0 \leq l \leq d-1)$ and $x = x_0 \ldots x_{d-1}$ is a d bit string called the position-within-level. Each vertex (l, x) is connected by a straight edge to $(l+1 \mod d,\ x)$ and by a cross edge to $(l+1 \mod d,\ x_0 \ldots x_{l-1}\overline{x_l}x_{l+1} \ldots x_{d-1})$; see Figure 12.

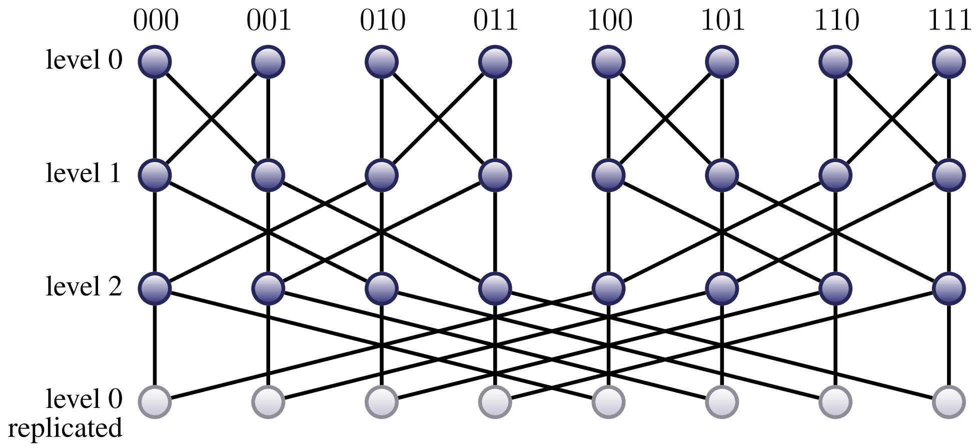


Figure 12: Butterfly Network BF_d for $d = 3$. For clarity, level 0 has been replicated.

First investigation of broadcasting were done by [101, 102], later the result was improved by [70] to $1.7417d \leq b(BF_d) \leq 2d - 1$.

In computer networking, Butterfly Networks are used primarily on *shared memory multiprocessors* having advantages like relatively low diameter and high link failure tolerance.

2.2.9 d-dimensional Grid $G[n_0 \times n_1 \times \ldots \times n_{d-1}]$

The d-dimensional grid graph of dimensions $n_0, n_1, \ldots, n_{d-1}$ denoted by $G[n_0 \times n_1 \times \cdots \times n_{d-1}]$ is the graph whose vertices are d-tuple of positive integers $(v_0, v_1, \ldots, v_{d-1})$ where $0 \leq v_i \leq n_i - 1$ for all $0 \leq i \leq d-1$. The edges connect d-tuples which differ in exactly one coordinate by one; see Figure 13. Diameter is $\sum_{i=0}^{d-1}(n_i - 1)$.

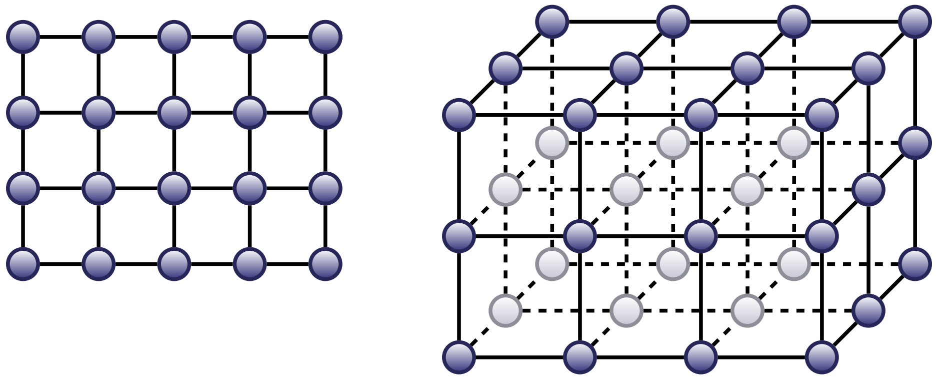


Figure 13: Grid graph $G[4 \times 5]$ and $G[3 \times 4 \times 3]$.

The first study of broadcasting in grids is by [40] in which $b(G[n_0 \times n_1]) = n_0 + n_1 - 2$ was proven, and more generally they showed:

- For a corner vertex v, $b(v) = n_0 + n_1 - 2$
- For a side vertex v, $b(v) =$ the maximum distance form v to a corner vertex.
- For an interior vertex $v = (i, j)$, $b(v) =$ the maximum distance form v to a corner vertex
  - plus 1 if $j = \frac{n_1+1}{2}$
  - plus 2 if $j = \frac{n_1+1}{2}$ and $i = \frac{n_0+1}{2}$

It can also be shown that for d-dimensional grids, $b(G[n_0 \times n_1 \times \cdots \times n_{d-1}]) = \sum_{i=0}^{d-1}(n_i - 1)$; however, $b(v)$ proves to be much more difficult and no work has been done about it. They also considered broadcasting in an *infinite 2-dimensional grid*. Let $f(n,t)$ be the maximum number of vertices than can be informed after t time-units in this grid. They have proved that $f(2,t) \leq 2t^2 - 6t + 8$ for $t \geq 2$ and conjectured equality. Inspired by this conjecture papers [29, 71, 72] improved the results to

$$\begin{aligned} f(n,t) \leq \frac{2^n}{n!}t^n + \frac{2-n2^n}{(n-1)!}t^{n-1} \\ + \frac{1}{(n-2)!}\left(4 - n - \frac{1}{3}(2^{n+1} - n2^{n-2}) + n^2 2^{n-1}\right)t^{n-2} + O(t^{n-3}). \end{aligned}$$

Infinite 2-dimensional hexagonal arrays, or triangular arrays, along with more conjectures and special case proofs have also been done; for more detailed information and references, please refer to [62]. Many computer networking systems employed grid structure for various reasons. All the examples provided in the Hypercube H_d section are applicable here since hypercubes are a special kind of grid network.

2.2.10 d-dimensional Torus $T[n_0 \times n_1 \times \ldots \times n_{d-1}]$

The d-dimensional Torus graph can be defined the same way as d-dimensional grid graph with the exception that the arithmetic of each dimension is done modulo to that dimension size. In this way, for dimension i, $n_i - 1$ and 0 in that dimension have a difference of one. Conceptually, d-dimensional torus can be viewed as forming a torus (hence the name); therefore, it is a toroidal graph; see Figure 14. The graph is $2d$-regular and its diameter is $\sum_{i=0}^{d-1}\lfloor n_i/2 \rfloor$.

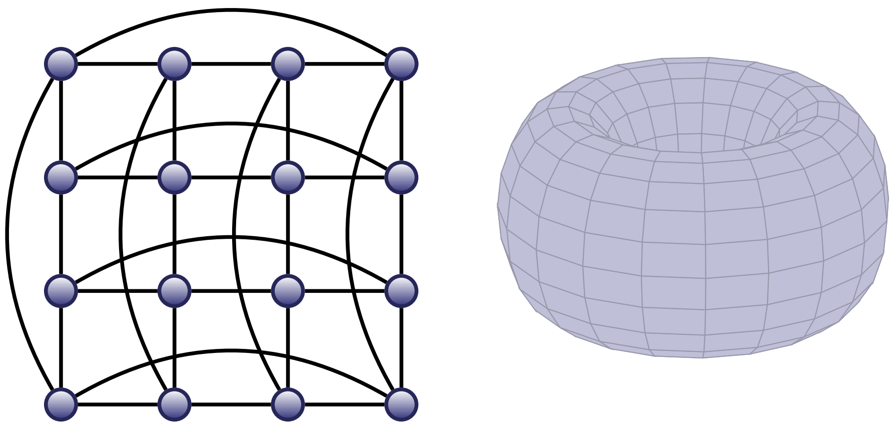


Figure 14: Torus graph $T[4 \times 4]$. Torus graph embeded on torus surface.

In [40], 2-dimensional torus was studied and they showed:

$$b(T[n_1 \times n_2]) = \begin{cases} \lfloor n_1/2 \rfloor + \lfloor n_2/2 \rfloor & \text{If } n_1 \text{ and } n_2 \text{ are even} \\ \lfloor n_1/2 \rfloor + \lfloor n_2/2 \rfloor + 1 & \text{If } n_1 \text{ and } n_2 \text{ are odd} \\ \lfloor n_1/2 \rfloor + \lfloor n_2/2 \rfloor + 1 & \text{Otherwise} \end{cases}$$

The following scheme obtained this result: if a vertex is informed by one of its vertical neighbors, it sends the message to its other vertical neighbor. Otherwise, it first sends the message to its other horizontal neighbor, then to its upper vertical neighbor, and finally to its lower vertical neighbor. However, this result is flawed for $T[3 \times 3]$, which will be discussed in chapter 4. For d-dimensional torus, it is known that

$$\sum_{i=0}^{d-1} \left\lfloor \frac{n_i}{2} \right\rfloor \leq b(T[n_0 \times n_1 \times \ldots \times n_{d-1}]) \leq \sum_{i=0}^{d-1} \left\lfloor \frac{n_i}{2} \right\rfloor + \max\{0, m-1\}$$

where m is the number of odd cycles. This can be proven using the 2-dimensional version and a simple induction. In computer networking, torus is a commonly used structure. Few examples are the use of 2-dimensional torus in [83, 91] and more generally use of torus in [1, 2, 66, 108].

2.2.11 Summary of the Known Results for the Usual Topologies

In [45] the status of the investigation of graphs mentioned above has been provided. Since then, some of the results may have been improved.

	K_n	C_n	2-dimensional T	T	G	H_d	CCC_d	$UB(d, D)$	SE_d	BF_d
	D	D	D	TI	D	D	D	TI	D	TI

(a) Broadcasting under the Classical model

	K_n	C_n	2-dimensional T	T	G	H_d	CCC_d	$UB(d, D)$	SE_d	BF_d
Full-duplex	D	D	D	TI	D	D	TI	?	?	TI
Half-duplex	D	D	TI	TI	TI	TI	TI	?	?	TI

(b) Gossiping under the Classical model

D	(Done) The exact value is known or tight bounds can be given.
TI	(To Improve) Results are to be improved in a significant way.
?	Open problem.

(c) Legend

Table 1: The status of known results of the usual topologies mentioned in section 2.2 according to [45].

2.3 Minimum Broadcast Graph

Historically, finding the broadcast function $B(n)$ and minimum broadcast graph (mbg) were the first investigated problems. By the discussion of section 2.1, A Hard Problem it is expected that finding $B(n)$ and mbg's should be extremely difficult. These problems indeed prove to be very difficult evident by the limitations of the proven results.

In [62], it is reported that the results of [41, 80, 86] collectively indicate the values of $B(n)$ for $n \leq 18$ and $B(2^k) = k2^{k-1}$. At the time no other result were known. Figure 15 and Table 2 summarize some of these findings. Since then some new results have been emerged; most notably, (1) [69], and [35] have independently showed that $B(2^k - 2) = (m-1)(2^{m-1} - 1)$ for $m \geq 2$; (2) the result for $n \leq 22$, $n = 26$ [110], $n = 63$ [76],

$n = 127$, $n = 1023$, $n = 4095$ and possibly more have also been found.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Number	1	1	1	1	4	1	1	4	6	21	2	1^*	1^*	2^*	3^*	8^*	5^*	1^*

$*$ Indicates the number known at the time of [62].

Table 2: Number of mbg's; courtesy of [62].

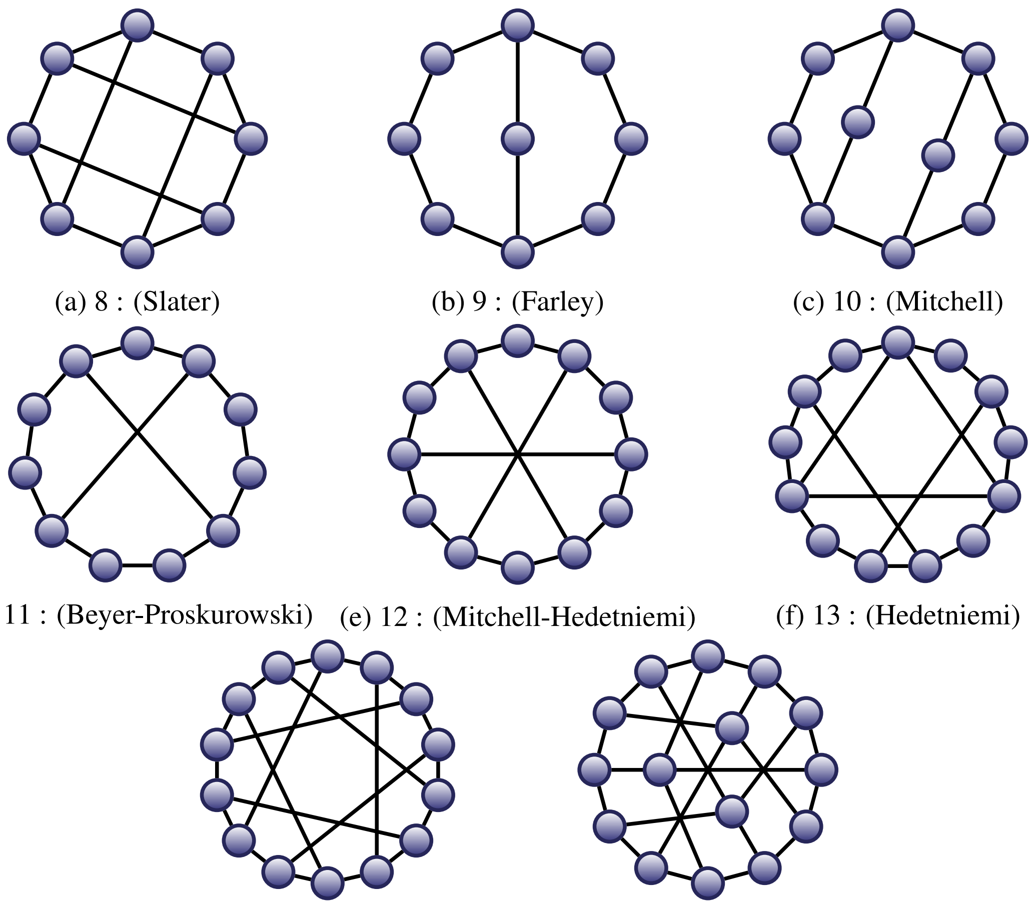

(a) 8 : (Slater)
(b) 9 : (Farley)
(c) 10 : (Mitchell)
(d) 11 : (Beyer-Proskurowski)
(e) 12 : (Mitchell-Hedetniemi)
(f) 13 : (Hedetniemi)
(g) 14 : (Bermond)
(h) 15 : (Hahn)

Figure 15: Example of minimum broadcast graphs; courtesy of [62].

Since finding the exact mbg is very difficult, some researchers have focused on giving sparse graphs and continually making them smaller to yield a better bound for $B(n)$. Some of these works are [12, 24, 25, 35, 36, 50, 55, 69]. In practice, for the design of the actual network, other considerations must be noted, particularly the average degree of the network. Examples of research on bounded degree graphs are [13, 21, 79].

2.4 Other Models

In addition to Classical and Messy models model and all the alternative models mentioned in the Introduction, including but not limited to the dynamic links model, alternative time models (e.g., linear model), or circuit-switch model, etc. there are many other hypotheses for which broadcasting/gossiping has been studied. Here a non-exhaustive list of such papers is presented.

In [44] the authors propose a linear model similar to the one discussed in which it also incorporates the number of communication links simultaneously used by each processor, and thus can also take into account the possibility of communicating in a restricted number of directions; in [16], the authors considered a model with constant T whereby splitting and combining of the packets are not allowed; in [20], broadcasting multiple messages in generalized Fibonacci trees is studied under the model that every processor can send one packet and receive one packet; in [81], authors study gossiping under the edge-coloring model in order to avoid the possibility of message collision and congestion; in [90], the authors discuss the problem of routing of several requests in all-optic networks, which has a high bandwidth; in [31, 88, 103], the authors analyze under reduced buffering; in [105], broadcasting of small number of messages in a square grid graph under different buffer sizes, in [42], consider a model between whispering and shouting; in [75], broadcasting and gossiping are generalized and studies on hypergraphs. Some networks with specific topologies have a property that passing a message between any pair of processors takes roughly the same time; for examples, see [27, 28, 32, 34]. To analyze this type, Bar-Noy and Kipnis have developed *postal model* [104], in which one can assume the processors are connected in a complete network, but there is a communication latency factor λ for sending a message. Broadcasting of multiple messages in the postal model is analyzed in [82].

Chapter 3

Cartesian Product in Broadcasting and Motivation

Hypercube and torus are the most common instances of k-ary n-cubes (cubes with n dimensions and k nodes in each dimension) in a network of multiprocessors [33]; some examples can be found in section 2.2, Usual Topologies. Hypercube topology has various beneficial features, which makes it ideal for multiprocessor systems, as it was pointed out in section 2.2. However, it has few drawbacks, such as not having a small fixed degree for any number of processors. Torus topology is also popular partly because of [33], in which they show that in some contexts, torus outperforms hypercubes. More information about why these two topologies are vastly used, considered beneficial, and the comparison between them can be found in [87]. Despite their simple form, one might wonder what feature(s) makes these topologies beneficial and commonplace.

It appears that the way edges are added in the recursive definition of the hypercube is what makes hypercube exceptionally good for parallel computing needs. The same kind of edges can be seen in torus if viewed as continually connecting one cycle to another copy of itself. This method of adding edges is reminiscent of the definition of Cartesian product in graphs. For this reason, we explored the definition of Cartesian product and investigated its effect on broadcast time.

Definition 3.0.1 (*Cartesian product*)**.** The Cartesian product of graphs G and H whether directed or undirected, written $G \times H$, is the graph with vertex set $V(G) \times V(H)$ specified by putting (u, v) adjacent to (u', v') if and only if (1) $u = u'$ and $vv' \in E(H)$, or (2) $v = v'$ and $uu' \in E(G)$.

Subgraph H of (a, b) or the *graph H associated with* (a, b) is defined as the induced subgraph on vertex set $\{(u, v)|u = a\}$, which by the definition is isomorphic to graph H. Similarly, *subgraph G of* (a, b) or the *graph G associated with* (a, b) is defined.

Example 3.0.1. *Figure 16 shows an example of the Cartesian product for undirected graphs.*

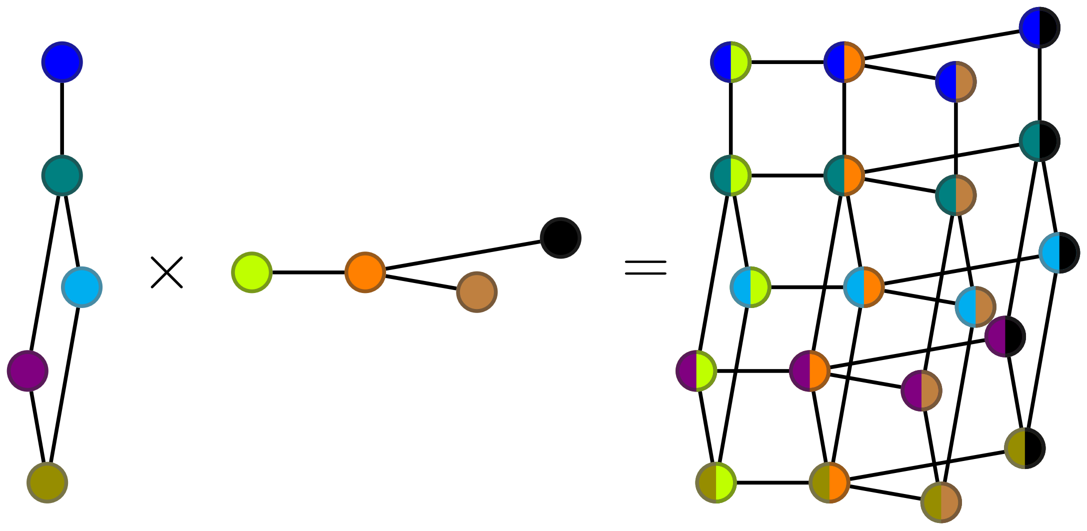


Figure 16: An example of the Cartesian product for undirected graphs. Courtesy of Wikipedia webpage [107].

Now that the definition is clear, let us explore to have a clearer picture of the effects of the Cartesian product in broadcasting under the Classical model starting with Theorem 3.0.1 which is our result.

Theorem 3.0.1. *Let G and H be two simple directed or undirected graphs then $b(G \times H) \leq b(G) + b(H)$.*

Proof. Let the originator be (o_G, o_H). The following broadcast scheme takes no more than $b(G) + b(H)$ time-units: Consider a non-lazy optimal broadcast scheme for each of G and H, namely S_G and S_H. After the vertex (a, b) is informed it follows S_G on associated G and then it follows S_H on associated H.

Let V_G be the set of vertices of the associated G of (o_G, o_H). All vertices in V_G are informed by $b(G)$ time-units. The vertices of $G \times H$ can be partitioned into $|V_G|$ copies of the graph H; expressly, for every vertex $x \in V_G$ the induced subgraph $\{(u, v)|u = x\}$. At time $b(G)$, since V_G is fully informed, all of these $|V_G|$ copies of H have a vertex x informed and these x's have no other vertex to inform in their associated G; i.e. V_G. Accordingly, all vertices will be informed no later than $b(G) + b(H)$ time-units. ■

Corollary 3.0.1.1. *Let $G = \Pi_{i=0}^{d-1} G_i$, then $b(G) \leq \sum_{i=0}^{d-1} b(G_i)$.*

Proof. Proof by induction on d. The case $d = 2$ is Theorem 3.0.1 and has been proven. Assume the result is true for $d - 1$, for some $d \geq 3$. Let $H = \Pi_{i=0}^{d-2} G_i$, by the induction hypothesis $b(H) \leq \sum_{i=0}^{d-2} b(G_i)$. Since $G = H \times G_{d-1}$, using theorem 3.0.1 we conclude that $b(G) \leq \sum_{i=0}^{d-1} b(G_i)$. ■

Remark 3.0.1. Corollary 3.0.1.1 may be tight for some graphs; for example, it is tight for 2d-grid as was seen in subsection 2.2.9.

The scheme used in Corollary 3.0.1.1 will be used in other parts of this document as well; therefore, let us name it *In order scheme*. The pseudocode is presented in In order scheme.

The proof of Theorem 3.0.1 in some sense suggests that the upper bound may not be tight even when either of G or H is slightly dense. Example 3.0.2 shows that the bound is indeed not tight.

Algorithm In order scheme Finds the next vertex to be informed from the given vertex using In order scheme in graph $G = \Pi_{i=0}^{d-1} G_i$

Input:

A vertex $v = (v_0, v_1, \ldots, v_{d-1})$ that has been informed.

Having access to fixed schemes $S(G_0), S(G_1), \cdots, S(G_{d-1})$

Output:

Next vertex to be informed from $(v_0, v_1, \ldots, v_{d-1})$

```
1: function nextInOrderScheme( $v$ )
2:     for $i = 0$ to $d - 1$
3:         if $v_i$ has a neighbor to inform in associated $G_i$ based on $S(G_i)$
4:             return next vertex in associated $G_i$ based on $S(G_i)$
5:     return NULL  //No next vertex to inform from $v$
```

Example 3.0.2. *Let $G = K_5 \times P_m$, where $m \geq 4$. The scheme in Figure 17 shows that $b(G) \leq m - 1 + 2 < m - 1 + 3 = b(P_m) + b(K_5)$.*

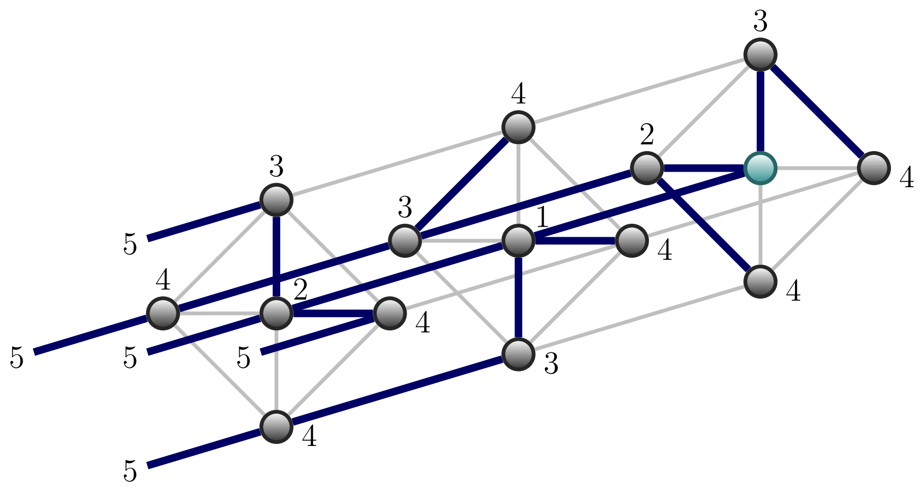


Figure 17: A scheme for $K_5 \times P_m$ where $m \geq 4$, which achieves broadcast time of $m-1+2$. The achieved time is less than $b(P_m) + b(K_5)$. For clarity, two edges of K_5 are not drawn.

To utilize the Cartesian product to find information about new classes of graphs, we must restrict the classes of "primitive" graphs, meaning the G_i's that are multiplied together. As it was seen when primitive graphs are even slightly dense, our current bound is not tight. Two useful types of graphs that are not dense are paths and cycles. The set of graphs obtained by the closure of the set of paths and cycles under Cartesian product contains Path P_n, Cycle C_n, d-dimensional Grid $G[n_0 \times n_1 \times \ldots \times n_{d-1}]$ (in particular Hypercube H_d,) and d-dimensional Torus $T[n_0 \times n_1 \times \ldots \times n_{d-1}]$. It also contains other beneficial topologies which might be useful in different fields where the study of broadcasting is of significance; see Figure 18 for some examples. Let us name this set *hyper-cylinder* graphs. In section 3.1, hyper-cylinder is formally defined, and an alternative definition of grid and torus graphs is presented.

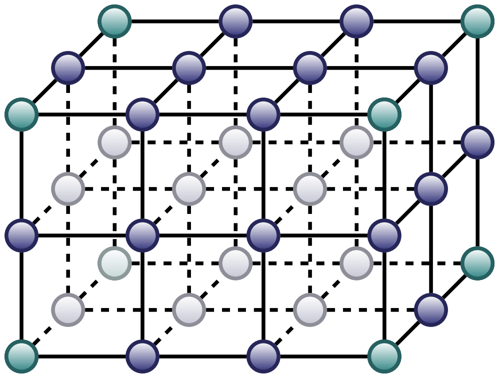
3.1 Definitions and Conventions

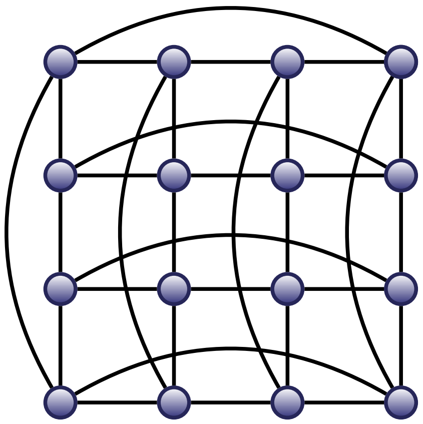
Definition 3.1.1 (*d-dimensional (Directed) Hyper-cylinder*)**.** The graph obtained by Cartesian product of d (directed) paths, or (directed) cycles, i.e. $\Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$, is called *$d$-dimensional (directed) hyper-cylinder* of size $\Pi_{i=0}^{d-1} l_i$.

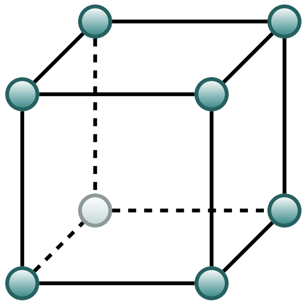
Definition 3.1.2 (*d-dimensional (Directed) Grid, d-dimensional (Directed) Torus*)**.** If all the graphs (H_i's) used in the definition of d-dimensional (directed) hyper-cylinder are (directed) paths the hyper-cylinder is a *d-dimensional (directed) grid*; and if all of them are (directed) cycles the graph is a *d-dimensional (directed) torus*.

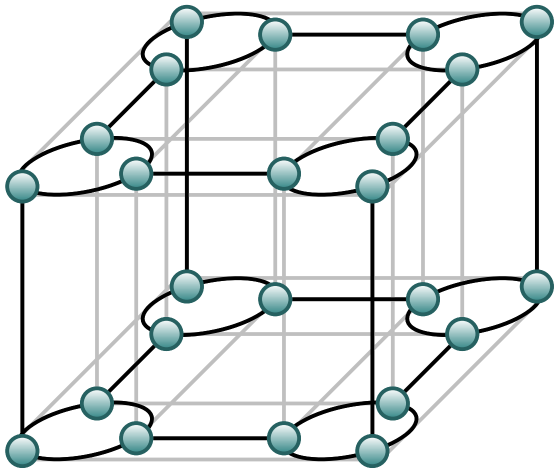
Convention 2. In the examples, the positive direction of vertical lines is taken to be downward in order for the vertex 0 to be at the top left.

Example 3.1.1. *Some examples of hyper-cylinder graphs are provided in the Figure 18.*


(a) Hyper-cylinder $P_3 \times P_4 \times P_3$, generating grid $G[3 \times 4 \times 3]$.


(b) Hyper-cylinder $C_4 \times C_4$, generating torus $T[4 \times 4]$.


(c) Hyper-cylinder $P_2 \times P_2 \times P_2$, generating hypercube H_3.


(d) Hyper-cylinder $C_3 \times P_2 \times P_2 \times P_2$, resembling CCC_3.

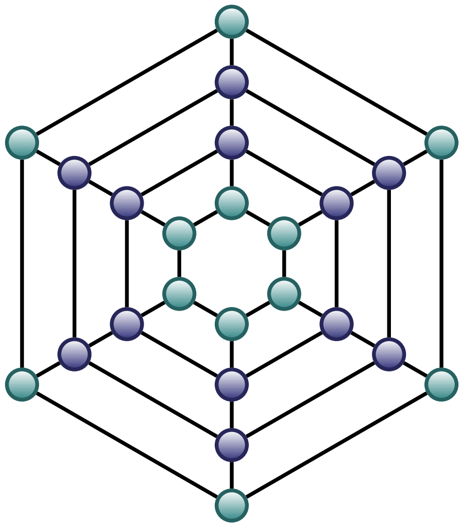
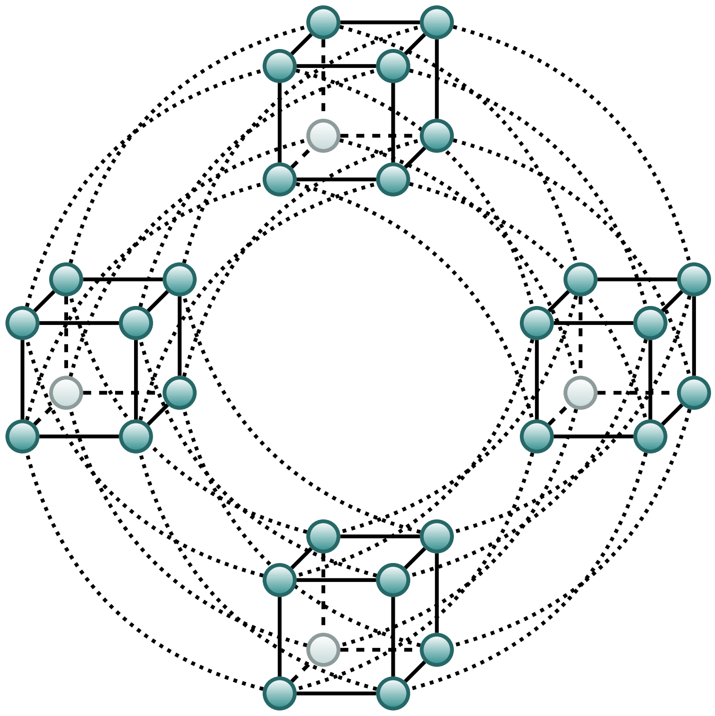

(e) Hyper-cylinder $C_6 \times P_4$.

Figure 18


(f) Hyper-cylinder $P_2 \times P_2 \times P_2 \times C_4$

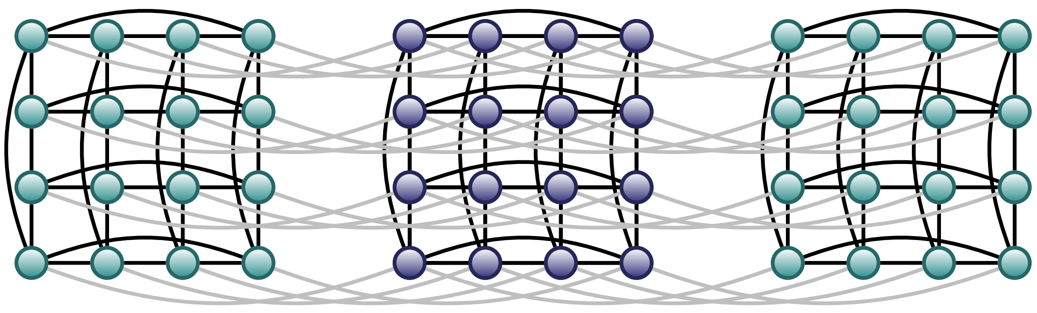

(g) Hyper-cylinder $C_4 \times C_4 \times P_3$

Figure 18: Examples of hyper-cylinder graphs. Corner vertices have been distinguished by light teal color.

Convention 3 (Lengths)**.** It is assumed that for all paths, their length is at least *one*, and for all cycles, the length is at least *two*. This is because the Cartesian product of a graph G with P_1 is isomorphic to G, and with C_1 is isomorphic to G with additional loops for each vertex which renders the result useless. The result of the Cartesian product with P_0 or C_0 is the "null" graph, which is a trivial structure for the study of broadcasting.

Convention 4 (*Naming of vertices*)**.** For every (not infinite) path and cycle, rename the vertices with non-negative integer numbers according to their ordering position, starting with 0. Consequently, when the length is l, modulo l and its arithmetic can be used to refer to the vertices. Additionally, without the loss of generality, assume that the ordering (in intuitive pictorial representation) is clockwise. To illustrate, vertex -1 refers to the vertex $l-1$, and vertex 1 is the next vertex of 0 clockwise.

In the case of (both side) infinite paths, 0 is assigned to an arbitrary vertex, and a positive direction is chosen; then, vertices are renamed as integer numbers. The positive direction is the outgoing direction in directed graphs, and it is randomly selected for undirected graphs.

Definition 3.1.3 (0 *vertex*)**.** The vertex of a d-dimensional hyper-cylinder whose all of its coordinate are 0, is called the 0 vertex.

Definition 3.1.4 (Corner Vertex)**.** Consider a d-dimensional hyper-cylinder $H = \Pi_{i=1}^{d} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$. Vertex $u = (u_1,\ u_2,\ \ldots,\ u_d)$ is a corner vertex of H, if for each H_i that is a path $u_i \in \{0, l_i\}$. If no H_i is a path, no corner vertex exists.

Example 3.1.2. *Figure 18 indicates the corner vertices in the provided examples.*

Definition 3.1.5 ((k-)Borders)**.** Let $H = \Pi_{i=0}^{d-1} H_i$ where $H_i \in \{P_{l_i+1}, C_{l_i}\}$ be a d-dimensional hyper-cylinder, by reindexing if necessary let us assume that $H_0, \ldots, H_{p-1}$ are paths and the rest are cycles provided that $p \geq 1$. For any fixed k indices from 0 to $p-1$, namely $i_0, i_1, \ldots i_{k-1}$ a k*-border* on these indices is the induced hyper-cylinder subgraph of H on the vertex set $V' = \{w = (w_0, \ldots, w_{d-1}) | w \in V(H),\ w_{i_j} = l_{i_j} \text{ for } 0 \leq j \leq k-1\}$.

Another name for 1-*border* is *border*. The name 0-*border* will be used to describe an induced subgraph on the set of vertices that are not in any border.

A *strict* k-border is a k-border which is not a $(k+1)$-border of G.

Definition 3.1.6 (*d-dimensional (Directed) Infinite Grid*)**.** If all the graphs (H_i's) used in the definition of d-dimensional (directed) hyper-cylinder are (directed) infinite paths (P_∞) this hyper-cylinder is called *d-dimensional (directed) infinite grid*.

Chapter 4

Broadcasting in Hyper-cylinder Graphs under Classical Model

4.1 Directed Graphs

Theorem 4.1.1. *Let* $H = \Pi_{i=0}^{d-1} H_i$*, where* $H_i \in \{P_{l_i+1},\ C_{l_i}\}$ *be a directed hyper-cylinder graph. With* 0 *as originator,* $b(0) = \sum_{i=0}^{d-1} l'_i$ *where* $l'_i = l_i$ *when* $H_i = P_{l_i+1}$ *and* $l'_i = l_i - 1$ *otherwise.*

Proof. The length of the shortest directed path from originator 0 to $(l'_0,\ l'_1,\ \ldots, l'_{d-1})$ is $\sum_{i=0}^{d-1} l'_i$; therefore, $b(0) \geq \sum_{i=0}^{d-1} l'_i$. In order scheme will reach all the vertices by $\sum_{i=0}^{d-1} l'_i$ time-units. ■

Corollary 4.1.1.1. *Let* $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ *and* $T = \Pi_{i=0}^{d-1} C_{l_i}$ *be a directed grid and torus respectively. With* 0 *as originator,* $b_G(0) = \sum_{i=0}^{d-1} l_i$ *and* $b_T(0) = \sum_{i=0}^{d-1}(l_i - 1)$.

Remark 4.1.1. The only viable choices for originator in a directed hyper-cylinder are the vertices that share one associate cycle with 0 vertex. This is because from other vertices, there is no other directed path to 0, and hence 0 will never get informed. It is easy to see that there is an automorphism such that each of these viable options becomes 0 vertex. This proves $b_H(0) = b(H)$.

Example 4.1.1. *Figure 19 illustrates the scheme and viable originators for directed hyper-cylinder $H = C_6 \times P_4$.*

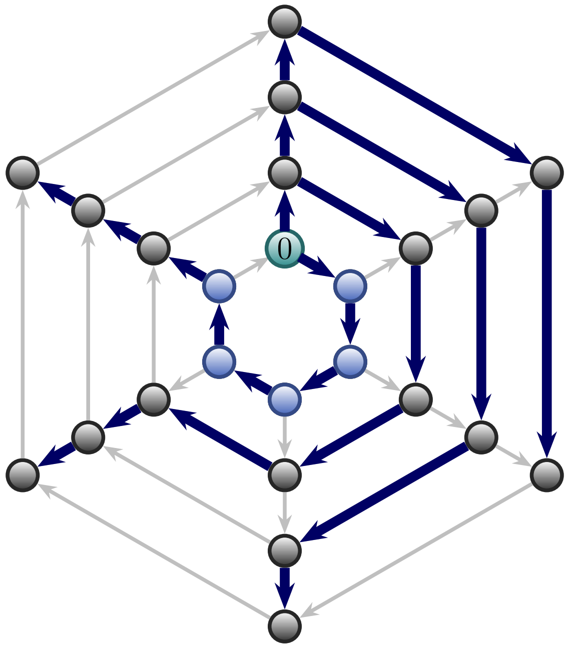


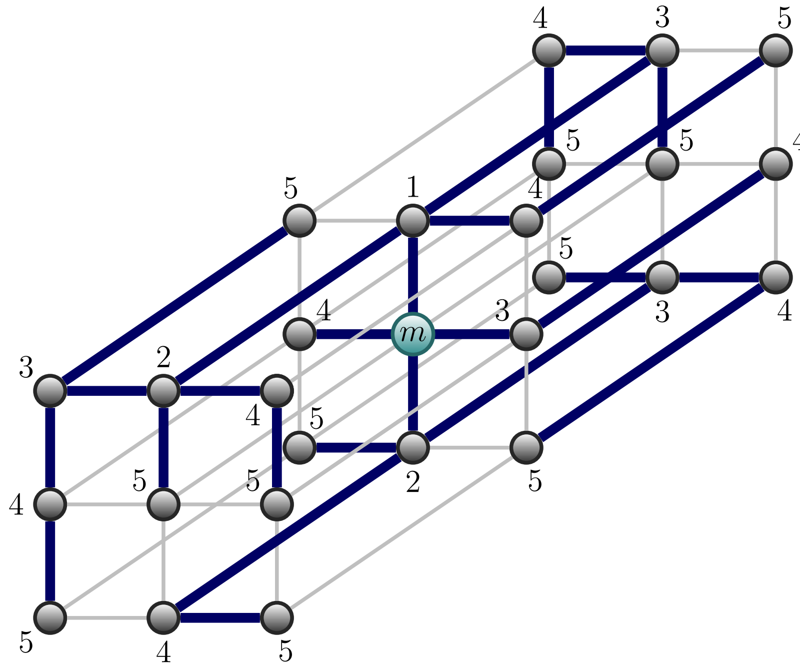
Figure 19: In order scheme for directed hyper-cylinder $H = C_6 \times P_4$; $b(H) = b_H(0) = 5 + 3 = 8$. Other viable originators have light blue color.

4.2 Undirected Graphs

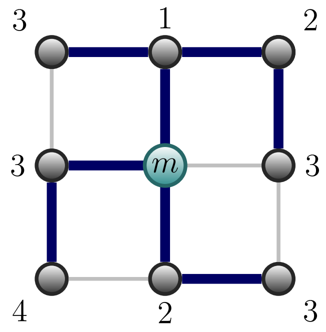
Definition 4.2.1. Let $u = (u_0,\ u_1,\ \ldots,\ u_{d-1})$ be a vertex in a d-dimensional hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$. Expression $mid(u) = k$ signifies that u is the middle vertex of exactly k of the odd-length paths (P_{l_i+1}'s); more precisely, it means that there is exactly k of i's such that $u_i = \frac{l_i - 1}{2}$ which is the index of middle vertex of odd-length path P_{l_i+1}.

Lemma 4.2.1. *Let $G = \Pi_{i=0}^{k-1} P_3$ be a grid and $m = (1, 1, \cdots, 1)$ be the middle vertex, then $b(m) \leq 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3)$. For $k \leq 3$ this upperbound is tight.*

Proof. The broadcast time of m for $k = 2$ is four, a special result of Farley and Hedetniemi which is mentioned in [62]. For $k = 3$, G has 27 vertices; consequently, $b_G(m) \geq \lceil \log_2(27) \rceil = 5$. In Figure 20 a broadcast scheme for both $k = 2$ and $k = 3$ is provided which achieves these bounds.



(a) $b(P_3 \times P_3 \times P_3,\ m) = 5$.



(b) $b(P_3 \times P_3,\ m) = 4$.

Figure 20: A minimum broadcast scheme for $P_3 \times P_3$, and for $P_3 \times P_3 \times P_3$.

Proof by induction on k for $k \geq 3$. For base case consider the aforementioned proof for $k = 3$. Assume the result is true for all values of n, which satisfies $3 \leq n < k$.

If $k \mod 3 \neq 0$, define $G_{k-1} = \Pi_{i=1}^{k} P_3$ and note that $G = G_{k-1} \times P_3$. Graph G can be though of as a path P_3 whose vertices are G_{k-1}. Consider the middle G_{k-1}, vertex m is the middle vertex of this subgraph as well. Therefore, the induction hypothesis indicates that starting from m the middle G_{k-1} can be informed in no more than $5 \times \lfloor (k-1)/3 \rfloor + 2 \times ((k-1) \mod 3)$ time-units. Applying Theorem 3.0.1 on $G_{k-1} \times P_3$ concludes that $b_G(m) \leq 5 \times \lfloor (k-1)/3 \rfloor + 2 \times ((k-1) \mod 3) + 2 = 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3)$.

Now consider $k \mod 3 = 0$. Letting $G_{k-3} = \Pi_{i=1}^{k-3} P_3$, G can be expressed as $G_{k-3} \times$

$(P_3 \times P_3 \times P_3)$. By the nature of Cartesian product one can imagine G as a $P_3 \times P_3 \times P_3$ graph whose vertices are a copy of G_{k-3}. Similar to before using the induction hypothesis the middle G_{k-3} can be informed in no more than $5 \times \lfloor (k-3)/3 \rfloor + 2 \times ((k-3) \mod 3)$ time-units. After that, the scheme mentioned in Figure 20 can be used to cover all G, this time; however, instead of a vertex informing another vertex a copy of G_{k-3} informs another copy of G_{k-3}. Accordingly, $b_G(m) \leq 5 \times \lfloor (k-3)/3 \rfloor + 2 \times ((k-3) \mod 3) + 5 = 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3)$. ■

Theorem 4.2.2. *In d-dimensional grid $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ with $u = (u_0,\ u_2,\ \ldots,\ u_{d-1})$ as originator,*

1. *If $mid(u) = 0$ then $b_G(u) = \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)$; the sum of broadcast time of u_i in P_{l_i+1} for every $0 \leq i \leq d-1$.*

2. *Otherwise, if $mid(u) = k \geq 1$, then*

$$\left[\sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)\right] - k + 1 \leq b_G(u) \leq \left[\sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)\right] - \frac{1}{3}(k - [k \mod 3])$$

Proof. Let $mid(u) = k$, C be the set of furthest corner vertices to u, and l'_i be the distance of u_i to one of the furthest end-vertices of the associated P_{l_i+1}. Cardinality of C is 2^k and the shortest distance from u to any member of C is $\sum_{i=0}^{d-1} l'_i$. If u_i is not the middle vertex of the associated P_{l_i+1}, then $b(P_{l_i+1},\ u_i) = l'_i$, else $b(P_{l_i+1},\ u_i) = l'_i + 1$. As a result, $b_G(u) \geq \sum_{i=0}^{d-1} l'_i = \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i) - k$.

Corollary 3.0.1.1 (using In order scheme) shows that $b_G(u) \leq \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)$ time-units. Therefore, if $k = 0$, $b_G(u) = \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)$.

Otherwise, if $k \geq 1$, consider the broadcast tree. Since $|C| \geq 2$ we have $b_G(u) \geq \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i) - k + 1$.

Without the loss of generality (with reindexing) assume that the associated P_{l_0+1} to $P_{l_{k-1}+1}$ are the paths in which the projection of u is in the middle. View G as $\Pi_{i=k}^{d-1} P_{l_i+1}$

connection of "hyper-vertices" $\Pi_{i=0}^{k-1} P_{l_i+1}$. Let H be the hyper-vertex that contains u. Note that by our choice u is the middle vertex of H, and $mid(H) = 0$ in the grid of hyper-vertices. We assert that H can be informed in $5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3) + \left\lceil \sum_{i=0}^{k-1} b(P_{l_i+1},\ u_i) \right\rceil - 2k$ time-units. If the assertion is true, by first informing H and then treating the problem as a similar problem for $\Pi_{i=k}^{d-1} P_{l_i+1}$ originating from H we get

$$\begin{aligned} b_G(u) &\leq b(H,u) + \sum_{i=k}^{d-1} b(P_{l_i+1},\ u_i) \\ &\leq \left\lceil \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i) \right\rceil - 2k + 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3) \\ &= \left\lceil \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i) \right\rceil - \frac{1}{3}(k - [k \mod 3]) \end{aligned}$$

concluding the proof.

The proof of the assertion. Associate a standard n-dimensional integer lattice ($\mathbb{Z}^k$) to vertices. Let vertex u be the 0 vertex, and $e_0, e_1, \ldots, e_{k-1}$ be the standard basis of our coordinate system. The new "coordinate" of vertex $v = (v_0, v_1, \ldots, v_{k-1})$ will be $(v_0 - u_0, v_1 - u_1, \ldots, v_{k-1} - u_{k-1})$. Having a vector notation facilitates the proof by allowing the use of vector operations.

By Lemma 4.2.1 all vertices of the middle subgraph $B = \Pi_{i=0}^{k-1} P_3$ can be informed in $5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3)$ time-units. The following is a partitioning of vertices of H into nine sub-multi-dimensional grids; in which each has a distinct vertex of B as a corner vertex. For each vertex $b \in B$, its multi-dimensional grid partition is defined as

$$P_b = \left\{ v \in V(H) \,\middle|\, v = b + \sum_{i=0}^{k-1} c_i b_i e_i \text{ where for all } i, c_i \in \mathbb{Z}^+ \right\}$$

Note that $b_i \in \{-1, 0, 1\}$; hence, $b_i e_i$ only dictates the direction away from the corner b in this multi-dimensional grid. To prove that this is a partition take a vertex $v = (v_0, v_1, \ldots, v_{k-1})$. Choose $b = (b_0, b_1, \ldots, b_{k-1}) \in B$, such that for each i, b_i is 1 if v_i is positive, 0 if v_i is zero and -1 if it is negative. Vertex v only resides in P_b.

Vertex b in P_b is the corner vertex and therefor $mid_{P_b}(b) = 0$. Consequently, the broadcast time of P_b with b as originator is $\sum_{i=0}^{k-1} |b_i|(l'_i - 1)$. After B is informed each $b \in B$ will act as the originator of its partition and exclusively broadcast there. Therefore,

$$\begin{aligned} b(H,u) &\leq 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3) + \max_{b \in B}\{b_{P_b}(b)\} \\ &= 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3) + \sum_{i=0}^{k-1}(l'_i - 1) \\ &= 5 \times \lfloor k/3 \rfloor + 2 \times (k \mod 3) + \left[\sum_{i=0}^{k-1} b(P_{l_i+1},\ u_i)\right] - 2k \end{aligned}$$

■

Corollary 4.2.2.1. *For every vertex v with $mid(v) \leq 1$ in $G = \Pi_{i=0}^{d-1} P_{l_i+1}$, $b(G,\ v) = \sum_{i=0}^{d-1} b(P_{l_i+1},\ v_i)$. Consequently, if the number of odd length paths does not exceed one the broadcast time of every vertex will be known.*

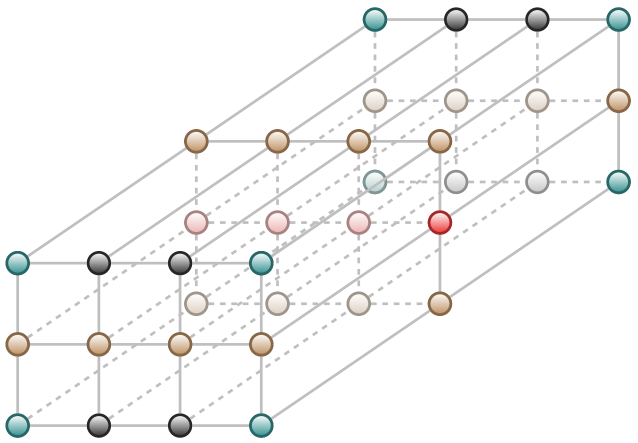
Corollary 4.2.2.2. *Let G be a d-dimensional grid $G = \Pi_{i=0}^{d-1} P_{l_i+1}$. By our knowledge of worst-case originators and broadcast center of paths we conclude that:*

1. *$b(G) = \sum_{i=0}^{d-1} l_i$; the sum of broadcast time of the paths.*

2. *Vertex $w = (w_0,\ w_1,\ \ldots,\ w_{d-1})$ is the worst-case originator, if and only if w_i is the worst-case originator of P_{l_i+1} for all $0 \leq i \leq d-1$. Consequently, the set of worst-case originators are the set of corner vertices.*

Proof. The result is the direct consequence of Theorem 4.2.2, noting that $\sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i) - \frac{1}{3}(k - [k \mod 3]) \leq \sum_{i=0}^{d-1} b(P_{l_i+1},\ u_i)$ and that $b(P_{l_i+1},\ u_i)$ are largest for the mentioned vertices. ■

Example 4.2.1. *Consider $G = P_4 \times P_3 \times P_3$ the information Theorem 4.2.2 and its corollary provide is summarized in Figure 21.*



🔵	$k = 0$	Worst-case originators $b(v) = 7$
⚫	$k = 0$	$b(v) = 6$
🟤	$k = 1$	Depending on the chosen vertex deterministically $b(v) = 6$ or 5
🔴	$k = 2$	For the two middle nodes the formula indicates $3 \leq b(v) \leq 4$ and for the two outer ones $4 \leq b(v) \leq 5$.

Figure 21: Summary of the information provided by Theorem 4.2.2 in $P_4 \times P_3 \times P_3$.

Theorem 4.2.3. *Broadcast time of $T = C_{l_1} \times C_{l_2}$(2d-torus) is:*

$$b(T) = \begin{cases} \lceil l_1/2 \rceil + \lceil l_2/2 \rceil = 4 & \textit{If } l_1 = l_2 = 3 \\ \lceil l_1/2 \rceil + \lceil l_2/2 \rceil - 1 & \textit{Else if } l_1 \textit{ and } l_2 \textit{ are odd numbers} \\ \lceil l_1/2 \rceil + \lceil l_2/2 \rceil & \textit{Otherwise} \end{cases}$$

Proof. Note that torus is vertex-transitive; therefore, only 0 vertex will be analyzed.

Consider the set $L = \{(v_1, v_2) \mid v_1 \in \{\lfloor l_1/2 \rfloor, \lceil l_1/2 \rceil\},\ v_2 \in \{\lfloor l_2/2 \rfloor, \lceil l_2/2 \rceil\}\}$ of vertices, which has the cardinality of $2^{\text{number of odd cycles}}$. The shortest path from origin to any vertex in L is $\lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor$, therefore $b(T) \geq \lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor$.

Theorem 3.0.1 (using In order scheme) guarantees that $b(T) \leq \lceil l_1/2 \rceil + \lceil l_2/2 \rceil$. This concludes the proof for the case where both l_1 and l_2 are even.

For other cases, consider the broadcast tree T^* of any optimal scheme. As mentioned vertices in L have distance $\lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor$ from the originator. Since $|L| > 1$, T^* cannot reach all elements in L in $\lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor$ time-units; thus, $b(T) \geq \lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor + 1$. This will conclude the result for the case when one cycle is of odd length and the other even.

For the case where both l_1 and l_2 are odd, if both are three we know that $b(T) \geq \lceil \log_2(3 \times 3) \rceil = 4 = \lceil l_1/2 \rceil + \lceil l_2/2 \rceil$. Else, the following broadcast scheme informs every vertex in $\lceil l_1/2 \rceil + \lceil l_2/2 \rceil - 1$ time-units, and since this number is equal to $\lfloor l_1/2 \rfloor + \lfloor l_2/2 \rfloor + 1$ the result is obtained. An example of this scheme is shown for $C_5 \times C_5$ in Figure 22.

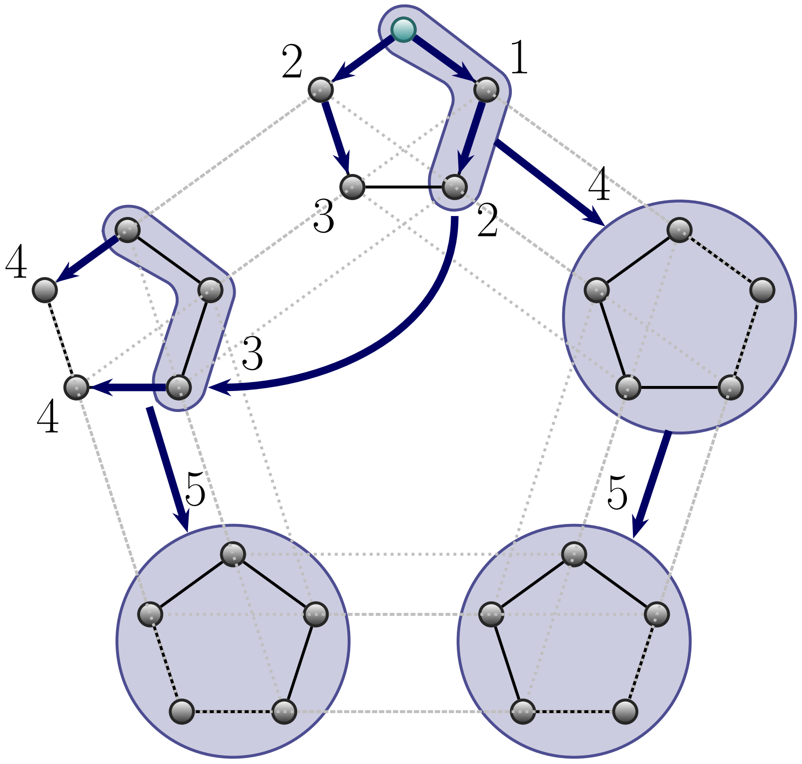


Figure 22: A broadcast scheme that informs $C_5 \times C_5$ in $\lceil 5/2 \rceil + \lceil 5/2 \rceil - 1$ time-units.

By symmetry, assume that $l_2 \geq l_1$; accordingly, $l_2 \geq 5$. Define C^i to be the associate C_{l_2} for the vertex $(i, 0)$. The scheme: Start by informing C^0 just like the regularly considered optimal scheme for cycles starting in clockwise direction. This means that first $(0, 1)$ is informed, then $(0, 0)$ progress counter-clockwise and $(0, 1)$ continues informing clockwise. At time $\lceil l_2/2 \rceil - 1$, only $(0, \lceil l_2/2 \rceil)$ is not informed in C^0. Utilizing Convention 4 notation, at time $\lceil l_2/2 \rceil$ vertices $\{(0, v_2) \mid -(\lfloor l_2/2 \rfloor - 1) < v_2 \leq \lfloor l_2/2 \rfloor\}$ will inform $\{(-1, v_2) \mid -$

$(\lfloor l_2/2 \rfloor - 1) < v_2 \leq \lfloor l_2/2 \rfloor\}$ and $(0, -(\lfloor l_2/2 \rfloor - 1))$ will inform $(0, -\lfloor l_2/2 \rfloor) = (0, \lceil l_2/2 \rceil)$, which makes C^0 fully informed. At time $\lceil l_2/2 \rceil + 1$, C^0 will inform C^1, and the only two uninformed vertices in C^{-1}; i.e. $(-1, -(\lfloor l_2/2 \rfloor - 1))$ and $(-1, \lceil l_2/2 \rceil)$, will get informed respectively by $(-1, -(\lfloor l_2/2 \rfloor - 2))$ and $(-1, \lfloor l_2/2 \rfloor)$. From now on, T is treated as C_{l_1} whose vertices are C^i's. At this time C^{-1}, C^0, C^1 are informed. Going clockwise from C^1 and counter-clockwise from C^{-1} all vertices will be informed in $\lceil l_2/2 \rceil + 1 + \lfloor l_1/2 \rfloor - 1 = \lceil l_1/2 \rceil + \lceil l_2/2 \rceil - 1$ time-units. ■

Remark 4.2.1. Unfortunately, we did not have access to [40] in which the 2d-torus is studied. However, we believe their method briefly mentioned in subsection 2.2.10 is different from ours; and more importantly, they did not distinguish the special case $T[3 \times 3]$.

Corollary 4.2.3.1. *Let $T = \Pi_{i=0}^{d-1} C_{l_i}$ be a d-dimensional torus with at most two odd cycles C_{l_i}. If both odd length are not 3, then the broadcast time and scheme of any vertex v is*

$$b(v) = \begin{cases} \sum_{i=0}^{d-1} \lceil l_i/2 \rceil - 1 & \textit{Two odd cycles} \\ \sum_{i=0}^{d-1} \lceil l_i/2 \rceil & \textit{Otherwise} \end{cases}$$

Proof. If two odd cycles exist, treat T as the connection of hyper-vertices isomorphic to the Cartesian product of the two odd cycles. In this case, by first informing the originator hyper-vertex using Theorem 4.2.3, and then using Corollary 3.0.1.1 the bound is achieved. The lower bound can be proven by noting that the eccentricity of any vertex is precisely this amount. Other cases are treated similarly; for one odd cycle, the hyper-vertex is that odd cycle, and for no odd cycle, the hyper-vertex is the vertex itself. ■

Corollary 4.2.3.2. *Let $T = \Pi_{i=0}^{d-1} C_{l_i}$ be a d-dimensional torus with k_1 odd cycles with* *length more than 3 and k_2 odd cycles with length 3. Then*

$$b(T) \leq \begin{cases} \sum_{i=0}^{d-1} \lceil l_i/2 \rceil - \left\lfloor \frac{k_1+k_2}{2} \right\rfloor & \textit{If } k_1 \geq k_2 \\ \sum_{i=0}^{d-1} \lceil l_i/2 \rceil - k_1 & \textit{Otherwise} \end{cases}$$

Proof. Note that the torus is a vertex-transitive graph; consequently, the proof for an arbitrary originator is sufficient. Let the number of odd l_i's be $k = k_1 + k_2$. The same techniques used in the proof of Theorem 4.2.3 is applicable here to get the result for the cases where $0 \leq k \leq 2$; in fact, it will prove that the broadcast time is exactly this bounds. The general case can be proven using induction on k. Take the correctness of $k = 0$ and $k = 1$ as the base case.

Assume the result is true for values less than k, where $k \geq 2$. There are two possibilities (1) $k_1 \geq k_2$ and (2) $k_1 < k_2$. For (1) if $k_2 \geq 1$, then take one odd cycle of length more than three and one with length three; and without the loss of generality (by reindexing) assume they are the first two cycles; i.e. C_{l_0} and C_{l_1}. Consider the graph $Q = C_{l_0} \times C_{l_1}$ associated with the origin. Base on Theorem 4.2.3 Q can be informed in $\lceil l_1/2 \rceil + \lceil l_2/2 \rceil - 1$ time-units. The graph G can be seen as $H = \Pi_{i=2}^{d-1} C_{l_i}$ connection of hyper-vertices isomorphic to Q. After the Q associated with the origin is informed, treat the broadcasting as informing H with the aforementioned hyper-vertex as originator. Note that H has $k_1 - 1$ odd cycles with length larger than three and $k_2 - 1$ with length three. By induction hypothesis H can be informed in $\sum_{i=2}^{d-1} \lceil l_i/2 \rceil - \left\lfloor \frac{k_1-1+k_2-1}{2} \right\rfloor$ time-units; hence, G can be informed in $\sum_{i=0}^{d-1} \lceil l_i/2 \rceil - \left\lfloor \frac{k_1+k_2}{2} \right\rfloor$ time-units. If however, $k_2 = 0$, then take two odd cycle with length more than three and similar arguments applies.

For (2) same technique is used; however, the case where $k_1 = 0$ needs to be proven separately. In this case, Corollary 3.0.1.1 is a scheme achieving the bound. ■

Conjecture 4.2.3.1. *The bounds provided in Corollary 4.2.3.2 are the exact broadcasting time of the torus graph under the Classical model.*

Remark 4.2.2. If Conjecture 4.2.3.1 is true, Corollary 4.2.3.2 provides an optimal broadcast scheme. Another optimal broadcast scheme will be presented shortly as well.

To justify our intuition for Conjecture 4.2.3.1, another optimal scheme with the same bound is provided. Based on this scheme, it can be intuitively seen that the bounds should be tight. Mainly it helps us to build the skeleton of a lower-bound argument. For this justification, the primary purpose is to present the intuition and the idea (the "leading messengers"); accordingly, the details are omitted.

Observation 4.2.1 (All odd cycles)**.** *As it was seen in Corollary 4.2.3.1 and Corollary 4.2.3.2 the presence of even cycles C_{2l} in the Cartesian product of T does not change our boundary gap of T's broadcast time. It increases both the upper and lower bound by l. Consequently, it is sufficient to only study the broadcast time of $T = \Pi_{i=0}^{k-1} C_{l_i}$ where all the cycles have odd length.*

The issue of odd cycles of length three will be discussed at the end. For now, assume that all the odd cycles have a length greater than 3. Let $T = \Pi_{i=0}^{k-1} C_{l_i}$ be such a graph. With these assumption, our new broadcast scheme should inform T in

$$\sum_{i=0}^{k-1} \lceil l_i/2 \rceil - \lfloor k/2 \rfloor = \sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor + k - \lfloor k/2 \rfloor = \sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor + \lceil k/2 \rceil$$

time-units. Let L be the set of vertices with maximum distance (which is $\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor$) from the originator. The cardinality of L is 2^k.

At first, *Scheme1* is presented which will informs T in $\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor + k$ time-units and introduces the imaginary concept of "replicating leader messengers." *Scheme 2* is the tweaked version of *Scheme 1* and will inform G in $\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor + \lceil k/2 \rceil$ time-units, the desired time.

Scheme 1. Figure 23 illustrates an example of this scheme. *Scheme 1* can be expressed by a story-like behavior using an imaginary concept named “replicating leader messengers.” For brevity, let us call them “leaders”.

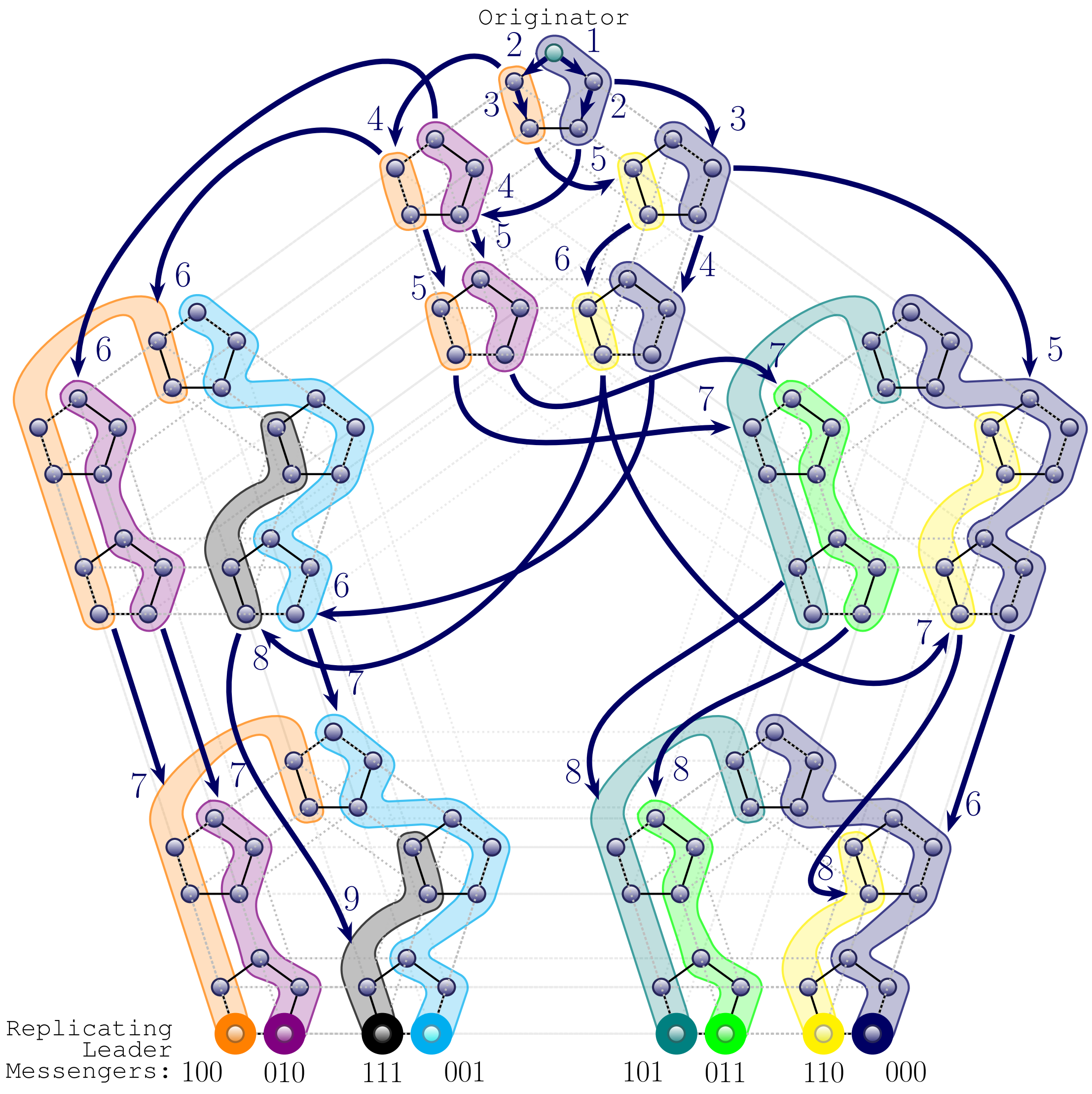


Figure 23: Scheme 1 applied to $C_5 \times C_5 \times C_5$ whose broadcast time is $\left[\sum_{i=0}^{2} \lfloor 5/2 \rfloor\right] + 3 = 9$. The leaders and their followers have been indicated with different colors. The completion time for each leader (and its followers) is $\left[\sum_{i=0}^{2} \lfloor 5/2 \rfloor\right]$ plus the number of 1’s in its binary representation. Let L be the set of maximum distance vertices to the originator. At the end of the scheme, there is a one to one correspondence between the leaders and the members of L.

The story is as follows: A message needs to be sent to all the vertices. A replicating leader messenger named Λ (null string) is chosen to complete this mission, with the addition of two priorities; (1) an ordering of the cycles (C_{l_i}) for informing purposes should be respected, and (2) within each cycle clockwise direction should be prioritized. Assuming that each action takes one time-unit, the behavior of a leader s is as follows: Before entering a new cycle, s duplicates itself. The new name for s will be $s0$, and the duplicate will have the name $s1$. The cycle ordering priority of $s1$ remains the same as $s0$; however, the direction priority changes to the other direction. The new leader $s1$ needs one time-unit to recover before it starts its mission. The first move of Λ is to enter the first cycle at the originator position. After entering a cycle, a leader starts informing the vertices of that cycle in the direction of its priority. If the priority is clockwise, it will stop at vertex $\lfloor l_i/2 \rfloor$ of that cycle; and if it is counter-clockwise, it will stop at vertex $-\lfloor l_i/2 \rfloor = \lceil l_i/2 \rceil$ of that cycle. Each vertex that is informed becomes a new follower of the leader who informed it. For the first vertex of each cycle where both $s0$ and $s1$ enter, the vertex becomes the follower of $s0$. The followers help their leader spread the message by mimicking their behavior, if possible, to a corresponding neighboring vertex. At the time of replication, followers of $s0$ after helping $s0$ will help $s1$ gain its first wave of followers. Figure 23 illustrates this behavior.

Observation 4.2.2. *It is clear that in* Scheme 1.

1. *All the vertices will be informed starting at the originator and that the scheme is a Classical broadcasting scheme.*

2. *In the end, there is a one-to-one correspondence between members of L and the leaders' positions.*

3. *Once a leader reaches a vertex in L, the leader and its followers remain idle for the rest of the scheme.*

4. *In the end, each leader and its followers partition the vertex set. The completion*

time in each partition is the distance traveled by the leader ($\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor$) plus all the recovery delays that the leader had. The recovery time is equal to the number of 1's in the leaders name. Consequently, the ending time unit is $\left[\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor\right] + k$.

Induced subgraph of L. Consider the induced subgraph Q of L. Since

$$L = \left\{ (v_0, v_1, \ldots, v_{k-1}) \;\middle|\; v_i \in \{\lfloor l_i/2 \rfloor, \lceil l_i/2 \rceil\} \text{ for } 0 \leq i \leq k-1 \right\}$$

Q is a k regular graph, one neighbor in each corresponding cycle. If the leaders' name are taken to be the names of vertices in Q, it is not hard to see that vertex v is connected to u if u's name can be obtained by "flipping" (negating) two consecutive bits of vertex v, or flipping the last (to the right) bit only. Since Q is k regular, these are the only connections for any vertex in Q.

Scheme 2. Figure 24 illustrates this scheme on $C_5 \times C_5 \times C_5$. *Scheme 2* follows *Scheme 1* except for two important changes, which allows it to have the completion time of $\left[\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor\right] + \lceil k/2 \rceil$ time-units. The changes are

1. A leader will not duplicate if its double will attain more than $\lceil k/2 \rceil$ ones in its (partial) name; however, it will append a 0 to its name as before. As was observed in item 4 of Observation 4.2.2, in this way when all the leaders reach L (the end) the completion time is $\left[\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor\right] + \lceil k/2 \rceil$. Unfortunately, however, due to this change, not all the vertices get informed. The next change will remedy this.

2. Assume v is a leader who is not created. In this scheme, the responsibility of informing the corresponding end vertex of v and all of its uninformed followers falls on leader u's shoulders, whose name is attained using the following procedure. Divide the (end) name of v into chunks of two bits from left to right; if k is odd, ignore the last bit. Go through these chunks one by one from left to right. If a chuck is 11, change it to 00 and immediately check if the number of 1's fall bellow or equal to

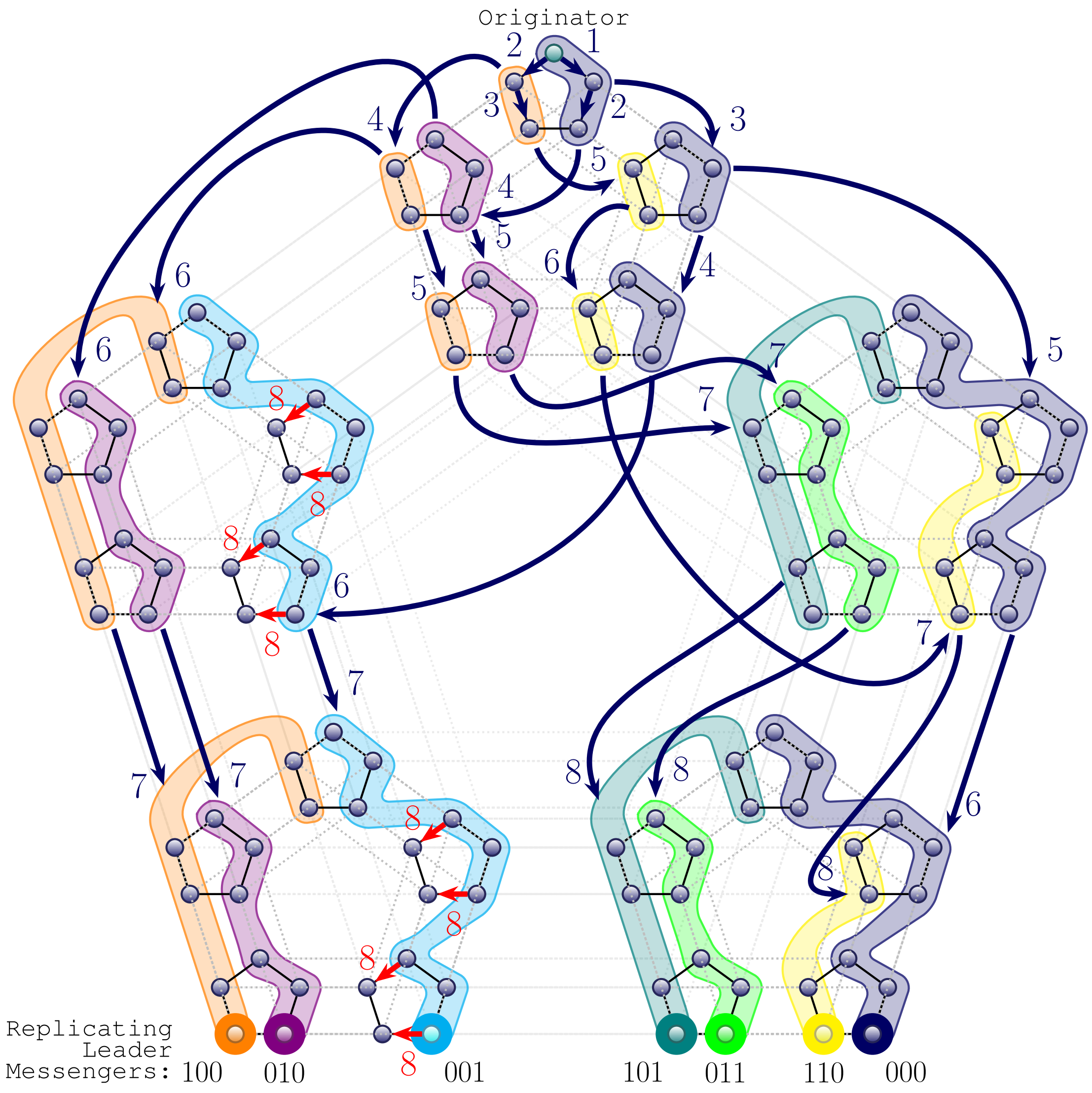


Figure 24: Scheme 2 applied to $C_5 \times C_5 \times C_5$ whose broadcast time is $\left\lceil \sum_{i=0}^{2} \lfloor 5/2 \rfloor \right\rceil + \lceil 3/2 \rceil = 8$. Leader 111 was not created and instead leader 001 took its responsibility.

$\lceil k/2 \rceil$ and if they do, stop the process. At some point, the stopping criterion happens. The arrived bit string is the name of u.

Let X_u be the set of all absent leaders whose duty has fallen on leader u's shoulders. Note that all such absent leaders share the same exact changed chunks of 2-bit string, with the only difference being one goes further to the right and changes one or a few more chunks than the other. Let $v \in X_u$ be the one who needed to change the rightmost chunk to reach u.

Leader u and its followers after reaching the corresponding vertex in L, start fulfilling their obligation to members of X_u. Leader $u = w_0$ and its relevant followers send the message to their neighboring vertices that reside in the partition associated with the leader whose name is obtained if the first (to the left) changed chunk of v is reversed back to 11 from u; let us call this leader w_1. Note that w_1 need not be an element of X_u. Then w_1 and its relevant followers send the message to neighboring vertices of the partition of the leader whose name is obtained by reversing the first two changed chucks of u; call it w_2. And so forth, until $w_p = v$ is reached.

First, note that it is not hard to see that (1) such neighboring exists, (2) $X_u \subseteq \{w_1, w_2, \ldots, w_p\}$, and (3) if these additional moves produce no conflict in the scheme, doing these additional moves will fulfill the duties of X_u members. Moreover, the total amount of time for leader u will not exceed $\left\lceil \sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor \right\rceil + \lceil k/2 \rceil$ time-units. Unfortunately; however, some conflict may arise which is addressed with the following change.

Because of the namings, the series of additional moves of one leader has no conflict with the additional move of another leader. The following modification resolves any conflict between the additional move of one leader and the Scheme 1 move of another. When u reaches L in Scheme 1, all of its followers and itself have no further moves. Therefore, for the first additional move, there is no conflict. In this way, w_1 and part of its followers are informed one time sooner than it was scheduled by *Scheme 1*. This is because the equivalent action of two ones in the name of w_1 was replaced by just one additional move from $w_0 = u$ to w_1. Generally, w_i and part of its partition are informed i time-units sooner than what was scheduled in Scheme 1. Consider the move of informing partition of w_{i+1} from w_i for any possible conflict.

The actions of followers of w_i can be categorized into two classes (1) they help to inform other vertices within the partition associated to w_i, or (2) they help to gain the initial followers for a double of their leader. When the w_1 is informed, all of its partition is informed. Accordingly, if there is any category one move in Scheme 1 from followers of w_1 *ignore* it since they are useless. In this way, followers of w_1 will have no conflict that can be caused due to their category one actions. For category two actions, note that not all followers of w_1 are needed to inform the partition of w_2, and those that are needed are the ones informed for the first time in the previous cycle (C_{l_j}) except the first hyper-vertex of this cycle. These vertices do not have any category two actions; therefore, no conflict will happen. This argument can be made formal with the help of mathematical induction.

Lower bound. Scheme 2 can help to gain intuition on the lower bound and to construct a skeleton of a possible proof for it. Let T^* be an optimal broadcast scheme tree. The paths in T^* that inform members of L are the main focus. The proof wants to say that under any scheme, the last member of L informed is informed no sooner than $\left\lceil \sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor \right\rceil + \lceil k/2 \rceil$. Without going through the details, there are two general approaches. The first approach highlights certain paths from originator to members of L in T^* inspired by the definition of leaders above. For example, (not very precisely) the paths that leave a cycle last, and the direction priority is determined by the first move after they enter a cycle. After, the time of the latest such path is investigated. The second approach is to trace the leaders' paths exactly as scheme 2 and compare it to T^*. Both methods have some success. Since we have failed to completely succeed in proving the lower bound using one of these methods, we will not explain our cumbersome effort in detail, which provides an even better intuition.

The concise overview of our attempt at the second approach is as follows. In some sense, consider the paths of leaders until they reach L we want to say that there is an optimal broadcast scheme that obeys the same paths (in some sense, maybe due to conflict similar to Scheme 2 some moves are not necessary). Assume not, for the closest optimal scheme to our claim, consider the first place they differ. One can change that position to make it closer to our paths while maintaining the optimality, contradicting the assumption. After this point, argue that there is an optimal scheme with the previous condition that only leaders should inform members of L. If not, the leader of the non-leader vertex who informed a certain member of L could inform that member as well without the loss of optimality, a contradiction. Finally, consider the position of the last informed leader, i.e., $11\dots1$. In the induced subgraph on L, as mentioned before, one leader is only connected to the other if two consecutive bits are flipped or the last bit is flipped. Any other 1 that is not obtained in these ways signifies an additional delay compared to the shortest distance. Consequently, to inform $11\dots1$'s position at least $\lceil k/2 \rceil$ delays are needed. Concluding that $b(T) \geq \left[\sum_{i=0}^{k-1} \lfloor l_i/2 \rfloor\right] + \lceil k/2 \rceil$.

Cycle of length 3. Consider the broadcast time of an odd length cycle. In the last move, there is only one uninformed vertex which can be informed by any of its two neighbors. This leaves one of the neighbors idle and unable to propagate the message to further distance. The following is the intuitive reasoning of what causes the delays and what is unique about C_3. The aforementioned idleness is the exact cause of delay. This delay is halved by pairing the odd cycles, one odd cycle utilizing its “idle” vertices, at least two, to send the information to the other cycle one time-unit sooner, eliminating the need for the last step in the other cycle. For this pairing to work, the number of odd idle vertices should be at least two. While C_3 can be the recipient of these benefits, it does not have enough “idle” vertices to reduce the time for others. This explained the bounds and the pairing treatment of the proof of Corollary 4.2.3.2.

Remark 4.2.3. The knowledge obtained from grids and torus provides bounds for any hyper-cylinder graph. Specifically, it gives the **exact** broadcast time and an optimal scheme of any vertex in any hyper-cylinder whose Cartesian product has at most two odd cycles and at most one odd-length path. Note that any number of even cycles or even length paths may be in the Cartesian product. The proof uses the hyper-vertex technique for the upper bound and the maximum distance with consideration of the optimal broadcast tree for the lower bound. Because of the similarity, the proof is omitted.

Example 4.2.2. *Broadcast time and an optimal broadcast scheme for every vertex of hyper-cylinder $P_3 \times C_5 \times C_5$ is known. This is because hyper-cylinder $P_3 \times C_5 \times C_5$ has two odd cycles and one odd-length path. The discussed optimal scheme for one vertex has been depicted in Figure 25, and based on our result, its broadcast time has been calculated.*

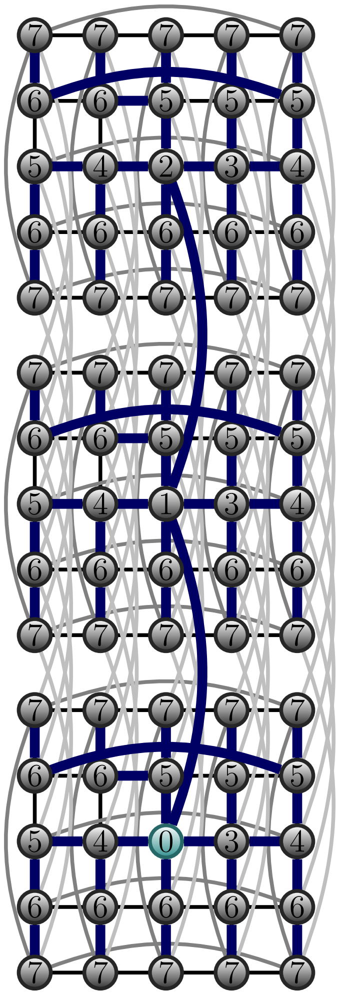


Figure 25: An optimal scheme from the indicated vertex in hyper-cylinder $P_3 \times C_5 \times C_5$ is shown. The broadcast time of the originator is $2 + \left[\sum_{i=0}^{1} \lceil 5/2 \rceil\right] - \lfloor 2/2 \rfloor = 7$.

Chapter 5

Broadcasting in Hyper-cylinder Graphs under Messy Models

5.1 Directed Graphs

The broadcast time of the directed multidimensional torus was studied in [30]. In this study, the authors successfully found the exact broadcast time for Messy models M_2 and M_3; however, for model M_1 failed to come up with an answer and only gave the exact time when the torus is two-dimensional. The following is the summary of their work for each model.

Remark 5.1.1. By symmetry, without the loss of generality, any (directed) torus $T = \Pi_{i=0}^{d-1} C_{l_i}$ can be expressed such that $2 \leq l_0 \leq l_1 \leq \cdots \leq l_{d-1}$.

- Model M_3:

  Theorem 5.1.1 ([30]). *Consider the d-dimensional directed torus $T = \Pi_{i=0}^{d-1} C_{l_i}$,*

$$t_3(T) = \sum_{i=0}^{d-1} (i+1)(l_i - 1) = (l_0 - 1) + 2(l_1 - 1) + 3(l_2 - 1) + \cdots + d(l_{d-1} - 1)$$

where $2 \leq l_0 \leq l_1 \leq \cdots \leq l_{d-1}$ *and* $d \geq 2$*.*

- Model M_2:

Theorem 5.1.2 ([30])**.** *Consider the d-dimensional torus* $T = \Pi_{i=0}^{d-1} C_{l_i}$*,*

$$t_2(T) = \left\lceil \sum_{i=0}^{d-1} (i+1)(l_i - 1) \right\rceil - j$$

where $2 = l_0 = l_1 = \cdots = l_{j-1}$ *for* $1 \leq j < d$*, and* $3 \leq l_j \leq \cdots \leq l_{d-1}$ *for* $d \geq 2$*.*

Remark 5.1.2. Note that in Theorem 5.1.2, all the dimensions cannot be two. If all the dimensions are 2 then $t_2(T) = t_2(\Pi_{i=0}^{d-1} C_2) = \left\lceil \sum_{i=0}^{d-1} (i+1)(1) \right\rceil - d + 1 = \frac{d(d-1)}{2} + 1$ which is one more than the formula expressed in Theorem 5.1.2. In this case the torus is isomorphic to a symmetric directed k-dimensional hypercube which under M_2 model can be viewed as an undirected hypercube. To prove the formula for $t_2(T)$ in this case, an almost identical proof to Theorem 5.1.2 can be used, or as mentioned, the result for undirected hypercube in [54] can be utilized; the latter is the choice of the authors in [30].

- Model M_1:

Theorem 5.1.3 ([30])**.** $t_1(C_{l_0} \times C_{l_1}) = l_0 + l_1 - 2$ *where* $2 \leq l_0 \leq l_1$*.*

Remark 5.1.3. Under Messy model M_1, in higher dimensions note that just like under model M_2, $T = \Pi_{i=0}^{d-1} C_2$ can be viewed as an undirected hypercube. In [54] it is hinted that it is not easy to know the exact broadcast time of undirected hypercube under M_1. The argument is based on the exact broadcast time and schemes obtained on small hypercubes. For this reason, the more general directed torus should not fair any better. It is, however, not obvious how much of this difficulty remains if we consider higher lengths for each dimension. Nevertheless, this difficulty can be intuitively sensed in higher lengths as well.

Inspired by [30], with a slightly different perspective, a proof of broadcast time and an optimal broadcast scheme can be given for d-dimensional hyper-cylinders. The result on both torus and grid can then be taken as corollaries. However, to make the proof easier to understand, first, the result for infinite grid and grid are found and then generalized to hyper-cylinder graphs. The idea is to investigate u, v-paths in any optimal broadcast tree where u is the originator. Each vertex in the path has some branches, some of which are useful in some sense which we named “improvements,” others can “delay” the path. Using this point, $t_i(u, v)$ for $2 \leq i \leq 3$ will be calculated. The equation $t_i(u)_{1\leq i\leq 3} = \max\{t_i(u, v)\}$ is used thereafter to conclude the broadcast time.

To explain the definition of “improvement,” and “delay” moves, let us further elaborate on the idea used on a d-dimensional directed grid $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ with u as the originator. The investigative objective is to find the value of $t_3(u, v)$ (or $t_2(u, v)$) for all reachable vertices v. Consider G' to be the subgraph induced by the vertex set

$$V' = \left\{w \;\middle|\; w \in V(G),\ u_i \leq w_i \leq v_i \text{ for } 0 \leq i \leq d-1\right\}$$

Note that (1) in any scheme, the path that conveys the information to v should reside entirely within G'. This is because it is not possible to come back to G' after exiting it. (2) Assuming there would be no call to outside of G', vertex v will get informed in exactly $\sum_{i=1}^{n}(v_i - u_i)$ time-units irrespective of the behavior of the used scheme. Accordingly, the only cause of delay on the path that conveys the information to v happens when the path delays by sending the message outside of G', which requires the path to be at that particular border of G'. For any vertex v and any scheme, refer to the movements of the u, v-path that are inside G' as “improvement” moves and movements that send the message out of G' as “delay” moves.

In the following, the broadcast time of some classes of hyper-cylinder is studied. Using their results at the end, a concluding general theorem for any hyper-cylinder is presented.

This is done to break the final proof, to make it easier to understand. All the previous results may be seen as corollaries of the general theorem.

Theorem 5.1.4 (Infinite directed grid)**.** *Consider a d-dimensional directed infinite grid $G = \Pi_{i=0}^{d-1} P_\infty$, and 0 as the originator. Under Messy models M_2 and M_3, for any reachable vertex v (non-negative coordinates, Convention 4)*

$$t_2(0, v) = t_3(0, v) = \sum_{i=0}^{d-1} (i + 1)(v_{\pi(i)})$$

where π is a permutation of $0, 1, \cdots, d-1$ such that $v_{\pi(0)} \leq v_{\pi(1)} \leq \cdots \leq v_{\pi(d-1)}$.

Proof. Since there are no pair of symmetric arcs, the analysis is similar under models M_2 and M_3; for this reason, only model M_3 is investigated.

Motivated by the prior discussions, consider the $0, v$-path P in a scheme S that maximizes $t_3(0, v)$. Denote 0 with u^0. First consider the case where G' is d-dimensional; i.e. for every i, $v_i > u^0_i = 0$.

Let u^1 be the first vertex on P that encounters a (1-)border (Definition 3.1.5) of G'; this should happen since u^0 is on no borders of G' and v is the only vertex on the d-border. No matter the scheme, it takes exactly $d(u^0, u^1)$ time-units to reach u^1 since all the movements are improving. Note that once P hits a border due to the directions it will remain on that border until it reaches v. Hitting the border means that for some i, $u^1_i = v_i$; without the loss of generality (by reindexing if necessary) assume $u^1_0 = v_0$. Now let u^2 be the first vertex that encounters a 2-border. It may be the case that $u^2 = u^1$ if u^1 is already on a 2-border. The required time-units to travel from u^1 to u^2 on P is no more that $2d(u^1, u^2)$; one possible delay move for each vertex which sends the message outside of G' using the first coordinate. Similar to above without the loss of generality assume $u^2_1 = v_1$. Generally, let u^k for $1 \leq k \leq d$ be the first vertex that hits a k-border. The amount of time needed to go from u^{k-1} to u^k does not exceed $kd(u^{k-1}, u^k)$; maximum $k-1$ delay moves (by sending the message outside of G' in $k-1$ direction) before an improvement move for each vertex on the path . Also $u^k_{k-1} = v_{k-1}$ is assumed. Consequently,

$$\begin{aligned}
t_3(0,v) &= t_3(u^0,u^d) \le \sum_{i=1}^{d} i\; d(u^{i-1},u^i) \\
&= (u^1_0 - u^0_0) + (2u^2_1 - u^1_1 - u^0_1) + (3u^3_2 - u^2_2 - u^2_2 - u^1_2 - u^0_2) + \cdots \\
&+ (du^d_{d-1} - u^{d-1}_{d-1} - \cdots - u^0_{d-1}) \le u^1_0 + 2u^2_1 + \cdots + du^d_{d-1} \\
&= 1v_0 + 2v_1 + \cdots + dv_{d-1}
\end{aligned}$$

In above argument some reindexing has been done for the ease of the explanation; therefore, the indices here are technically a permutation of $0, 1, \cdots, d-1$. This then tell us that without reindexing $t_3(0,v) \le \max_\sigma\{1v_{\sigma(0)} + 2v_{\sigma(1)} + \cdots + dv_{\sigma(d-1)}\}$ where σ is a permutation of $0, 1, \cdots, d-1$. Hence, $t_3(0,v) \le 1v_{\pi(0)} + 2v_{\pi(1)} + \cdots + dv_{\pi(d-1)}$, where π is a permutation that satisfies $v_{\pi(0)} \le v_{\pi(1)} \le \cdots \le v_{\pi(d-1)}$. The permutation π signifies that the border of a shorter dimension needs to be encountered sooner than a larger one.

If on the other hand G' is p-dimensional where $p < d$, the same argument as above can be made with $u^0 = u^1 = \cdots = u^{d-p}$ in mind (as if we are starting at u^{d-p} with $d-p$ delays available at the start). Like before after each hitting the border the number of delays available is increased by one. Note that being in this case means that (again with reindexing in mind) $v_0 = v_1 = \cdots = v_{d-p-1} = 0$. Accordingly, the same expression as above will work here as well.

The equality is obtained if a scheme has $t_3(0,v) \ge \sum_{i=0}^{d-1}(i+1)(v_{\pi(i)})$, where π is a permutation of $0, 1, \cdots, d-1$ such that $v_{\pi(0)} \le v_{\pi(1)} \le \cdots \le v_{\pi(d-1)}$. Consider this scheme: for each vertex the order of the neighbors to inform follows π; this means that the vertex $w = (w_0, w_1, \cdots, w_{d-1})$ will first inform $(w_0, \cdots, w_{\pi(0)} + 1, \cdots w_{d-1})$ and then $(w_0, \cdots, w_{\pi(1)} + 1, \cdots w_{d-1})$ and so forth. Indeed, the scheme is In order scheme with the imposed order of π. In this way vertex v is informed at exactly $\sum_{i=0}^{d-1}(i+1)(v_{\pi(i)})$

time-units for the first time. Figure 26 illustrates the behavior of this scheme.

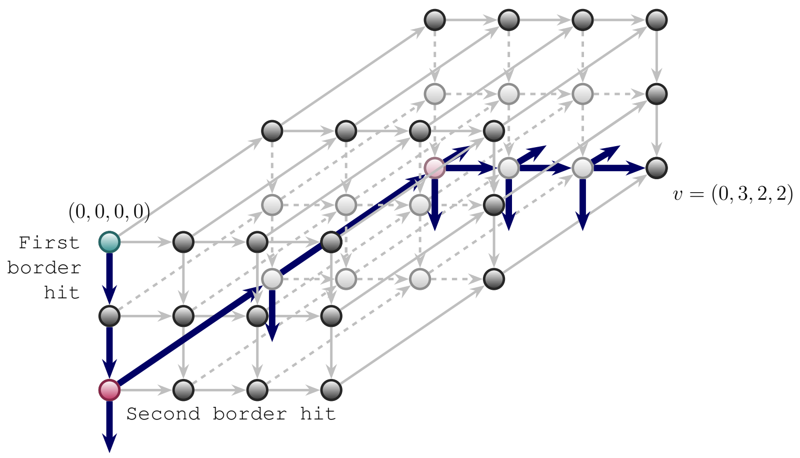


Figure 26: Let $G = \Pi_{i=0}^{3} P_{\infty}$ be a 4-dimensional directed infinite grid. The In order scheme with the imposed permutation $(0, 2, 3, 1)$ reaches $v = (0, 3, 2, 2)$ at $1\times0+2\times2+3\times2+4\times3$ time-units for the first time. The shown subgraph is G' which is 3-dimensional. Note that the delay move used by every vertex in the path which utilizes the first dimension is not shown.

■

Remark 5.1.4 (Grid originator). In directed grid graphs, the only suitable broadcast originator is vertex 0, since that is the only vertex that can reach every other vertex. In infinite grid graph, by vertex transitivity 0 is chosen as the originator.

Lemma 5.1.5 (k-border of directed grid)**.** *Consider a d-dimensional directed grid $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ with 0 as the originator. Under Messy models M_2 and M_3, for any vertex v which lies on a strict k-border of G, for some $k \geq 0$*

$$t_2(0, v) = t_3(0, v) \leq \left[\sum_{i=0}^{d-k-1} (i+1)(v_{\pi(i)})\right] + (d-k+1)\left[\sum_{i=d-k}^{d-1} v_{\pi(i)}\right]$$
$$= (1)v_{\pi(0)} + \cdots + (d-k)v_{\pi(d-k-1)} + (d-k+1)v_{\pi(d-k)} + \cdots + (d-k+1)v_{\pi(d-1)}$$

where π is a permutation of $0, 1, \cdots, d-1$ such that $v_{\pi(0)} \leq v_{\pi(1)} \leq \cdots \leq v_{\pi(d-1)}$.

The equality happens if v lies on the k-border whose indices are

$$\Big\{\pi(d-k), \pi(d-k+1), \cdots, \pi(d-1)\Big\}$$

Proof. First, note that $k = 0$ is also possibility, meaning that the vertex v lies on no border. In this case, $(d-k+1)\left[\sum_{i=d-k}^{d-1} v_{\pi(i)}\right] = 0$ since the starting index is greater than the ending index.

The proof is almost identical to the proof of Theorem 5.1.4 with one exception; the coefficients behind $d(u^{i-1}, u^i)$. Let c_i be the coefficient for $d(u^{i-1}, u^i)$ in this case. Like before $c_1 = 1$. For $i \geq 2$, we refer to the moment u^i was encountered. In the previous proof, it was noted that at this moment $u_i^i = v_i$ which opens up one delay move opportunity. The delay move is obtained by sending the message outside of G' using the ith dimension, causing the coefficient behind $d(u^{i-1}, u^i)$ to be one larger than that of $d(u^{i-2}, u^{i-1})$. In non-infinite grids; however, this opportunity does not present itself when $v_i = l_i$ since there are no vertices whose ith dimension exceed l_i. Since v is in a strict k-border this situation happens exactly k times. Consequently, $c_1, c_2, \ldots, c_d$ is a non-decreasing sequence in which, (1) $c_1 = 1$ (2) $c_{i+1} = c_i$ or $c_{i+1} = c_i + 1$ for $i \geq 2$, (3) $c_d = d - k + 1$. Hence,

$$\begin{aligned} t_3(0, v) = t_3(u^0, u^d) &\leq \sum_{i=1}^{d} c_i d(u^{i-1}, u^i) \\ &\leq c_1 u_0^1 + c_2 u_1^2 + \cdots + c_d u_{d-1}^d = c_1 v_0 + c_2 v_1 + \cdots + c_d v_{d-1} \end{aligned}$$

As in the previous proof if reindexing is not done we have $t_3(0, v) \leq \max_\sigma\{c_1 v_{\sigma(0)} + c_2 v_{\sigma(1)} + \cdots + c_d v_{\sigma(d-1)}\}$ where σ is any permutation of $0, 1, \cdots, d-1$. Since the sequence $[c_i]_{1 \leq i \leq d}$ is non-decreasing, $t_3(0, v) \leq c_1 v_{\pi(0)} + c_2 v_{\pi(1)} + \cdots + c_d v_{\pi(d-1)}$, where π is a permutation that satisfies $v_{\pi(0)} \leq v_{\pi(1)} \leq \cdots \leq v_{\pi(d-1)}$. Lastly, by the limitations of the sequence $t_3(0, v) \leq \left[\sum_{i=0}^{d-k-1} (i+1)(v_{\pi(i)})\right] + (d-k+1)\left[\sum_{i=d-k}^{d-1} v_{\pi(i)}\right]$.

If v lies on the k-border whose indices $\{\pi(d-k), \pi(d-k+1), \cdots, \pi(d-1)\}$, as indicated by the above inequalities In order scheme with imposed order of π (similar to the proof of Theorem 5.1.4) proves that this bound it tight for v. Figure 27 illustrates the behavior of this scheme, and provides a case where the inequality is not tight.

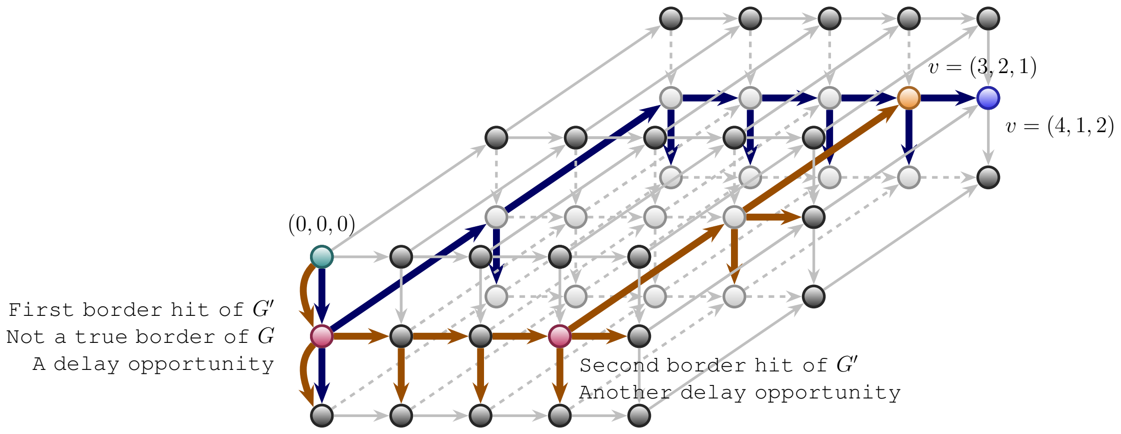


Figure 27: Let $G = P_5 \times P_3 \times P_3$ be a 3-dimensional grid. The vertex $(4,1,2)$ is on a strict 2-border and the bound of Lemma 5.1.5 is tight for it. The vertex $(3,2,1)$ is on a strict 1-border. The bound for this vertex yields 14 time-units, but the best possible is 13 time-units using the shown broadcast scheme.

■

By symmetry, without the loss of generality, directed grid G can be expressed as $\Pi_{i=0}^{d-1} P_{l_i+1}$ where $1 \leq l_0 \leq l_1 \leq \cdots \leq l_{d-1}$.

Theorem 5.1.6 (Directed grid)**.** *Consider a d-dimensional directed grid* $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ *where* $1 \leq l_0 \leq l_1 \leq \cdots \leq l_{d-1}$. *With* 0 *as the originator,*

$$t_2(0) = t_3(0) = \left[\sum_{i=0}^{d-k^*-1} (i+1)(l_i - 1)\right] + (d - k^* + 1)\left[\sum_{i=d-k^*}^{d-1} l_i\right]$$

$$k^* = \underset{0 \leq k < d}{\arg\max} \left\{ d - k - \sum_{i=d-k}^{d-1} l_i > 0 \right\} + 1$$

Proof. Due to the absence of symmetric arcs, $t_2(0) = t_3(0)$; therefore, it is suffice to only find $t_3(0)$. We have

$$\begin{aligned} t_3(0) &= \max_v \left\{ t_3(0,v) \;\middle|\; v \in V(G) \right\} \\ &= \max_{0 \le k \le d} \left\{ \max_v \{ t_3(0,v) \;\middle|\; v \text{ belonging to a strict } k\text{-border of } G \right\} \} \end{aligned}$$

For a fix k using Lemma 5.1.5 it is easy to see that

$$\max_v \left\{ t_3(0,v) \;\middle|\; v \text{ belonging to a strict } k\text{-border of } G \right\} \\ = \left[\sum_{i=0}^{d-k-1} (i+1)(l_i - 1) \right] + (d-k+1) \left[\sum_{i=d-k}^{d-1} l_i \right]$$

By definition $0 \le k \le d$. in the following the incremental changes of this function from k to $k+1$ is of interest; consequently, for k consider the range $0 \le k < d$. The change to this function from k to $k+1$ is $d - k - (l_{d-k} + \cdots + l_{d-1})$; therefore, it is a strictly decreasing function with respect to k. For $k = 0$ the change is $+d$ which is in accordance to the formula if we treat $(l_{d-k} + \cdots + l_{d-1})$ as 0. For $k = 0$ as mentioned the change is positive and there is a first k with which this change cease to be positive, due to the fact that for $k = d-1$ (for $d \ge 2$) the change is non-positive. This k is one more than the maximum integer k that satisfies $d - k - (l_{d-k} + \cdots + l_{d-1}) > 0$; call it k^*. Consequently, for $d \ge 2$

$$t_3(0) = \left[\sum_{i=0}^{d-k^*-1} (i+1)(l_i - 1) \right] + (d-k^*+1) \left[\sum_{i=d-k^*}^{d-1} l_i \right]$$

For $d = 1$ the formulas also works correctly. See Corollary 5.1.6.2 for an example. ■

Corollary 5.1.6.1 (Last vertices). *Consider a d-dimensional directed grid $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ where $1 \le l_0 \le l_1 \le \cdots \le l_{d-1}$. With 0 as the originator, under Messy models M_2 and M_3 in any non-lazy optimal broadcast scheme of G the set of last vertices informed* *are all in strict k^*-border(s) or strict (k^*+1)-border(s) of G, where k^* is the same as in Theorem 5.1.6. The strict (k^*+1)-border is only a possible if $d - k^* - \sum_{i=d-k^*}^{d-1} l_i = 0$.*

Proof. This is the direct consequence based on the proof of Theorem 5.1.6, especially the fact that the change function mentioned is strictly decreasing and cannot have two consecutive 0 values. Therefore, there can be only one or two consecutive maximizing k's. See Figure 28 for an example. ■

Corollary 5.1.6.2 (Hypercube)**.** *The broadcast time of a d-dimensional directed hypercube $G = \Pi_{i=0}^{d-1} P_2$ under Messy models M_2 and M_3 is $\lfloor (d+1)/2 \rfloor \lceil (d+1)/2 \rceil$.*

Proof. Directed hypercube is a type of directed grid and thus Theorem 5.1.6 applies to it. Let us find k^*.

$$d - k > \sum_{i=d-k}^{d-1} l_i \Rightarrow d > 2k \Rightarrow k \le \left\lfloor \frac{d-1}{2} \right\rfloor \Rightarrow k^* = \left\lfloor \frac{d+1}{2} \right\rfloor$$

Therefore, the broadcast time under Messy models M_2 and M_3 is

$$\left[\sum_{i=0}^{d-k^*-1} (i+1)(l_i - 1) \right] + (d - k^* + 1) \left[\sum_{i=d-k^*}^{d-1} l_i \right] = (d - k^* + 1)k^* = \left\lfloor \frac{(d+1)}{2} \right\rfloor \left\lceil \frac{(d+1)}{2} \right\rceil$$

Figure 28 shows one such optimal scheme in 3-dimensional directed hypercube.

■

Remark 5.1.5. By Corollary 5.1.6.1, if the usual binary naming of hypercube is used, the set of last vertices getting informed in any non-lazy optimal scheme will be those vertices that have $\lfloor (d+1)/2 \rfloor$ or $\lceil (d+1)/2 \rceil$ ones in their binary representation. The latter ($\lceil (d+1)/2 \rceil$ ones) is only possible if d is even. See Figure 28 for an example.

General theorems. Finally, the general theorems for hyper-cylinders are presented, first under Messy model M_3 and then under Messy model M_2. The theorems may seem

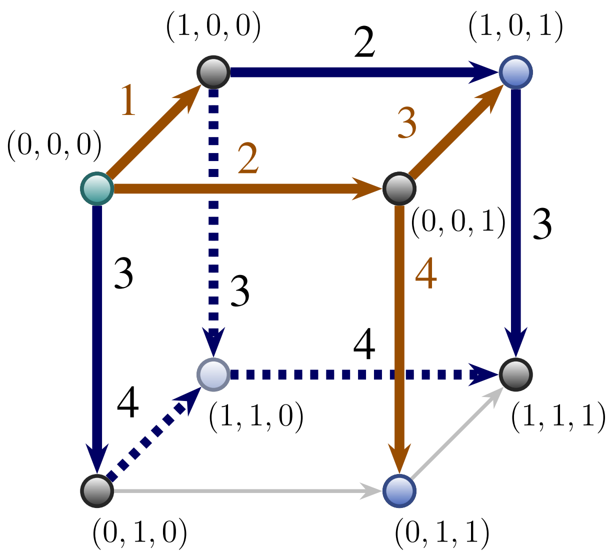


Figure 28: A non-lazy optimal broadcast scheme under model M_2 and M_3 for 3-dimensional directed hypercube, whose broadcast time is $\lfloor \frac{4}{2} \rfloor \lceil \frac{4}{2} \rceil = 4$. The possible last informed vertices are indicated with light blue color. In this scheme $(0, 1, 1)$ is the last informed. Its delaying path is displayed with a dark orange color.

complicated, but they describe an intuitive optimal broadcast scheme just like the ones seen for torus and grid. Using these theorems, one can be sure that an easy optimal broadcast scheme is known for any hyper-cylinder under Messy models M_2 and M_3. Furthermore, these theorems provide a means to calculate broadcast time easily. These consequences and other corollaries will be explained after the proof of the relevant theorem.

Remark 5.1.6. Just like the cases for torus and grid, by symmetry, without the loss of generality, hyper-cylinder H can be expressed as $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$ and $l_0 \leq l_1 \leq \ldots \leq l_{d-1}$.

Remark 5.1.7. The only possible originators for a d-dimensional hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$ are the vertices who share an associated cycle (C_{l_i}) with vertex 0. For any other vertex, there is no path to vertex 0. Due to symmetry, the results for 0 applies to any possible originator.

To make the general theorems more easily expressible, let us explore the idea under Messy model M_3 first and provide two definitions.

If one carefully observes the discussion of directed grid graph, it becomes apparent that, in short, the discussed optimal scheme works as follows: One by one, it traverses $d - k^*$ smaller dimensions until just before hitting the border of G, while utilizing any delay move possible along the way. Not hitting the border allows for one additional delay move in each such dimension for the rest of the path. The last k^* dimensions, which are the largest, will be traversed in the same way, except this time the border of G is hit. The formula expresses how to find an optimal k^*. See Figure 28 for such example example path, here $k^* = 2$. In the case of a torus, the problem of hitting the border is non-existence, and each dimension can be utilized for an additional delay move eventually. By observing this, the idea for an optimal broadcast scheme of general hyper-cylinder is to find a similar k^* such that only the k^* largest *paths* hit the border (without producing any delay move opportunity) and the rest utilized to the fullest extent while also providing an additional direction for delay move.

Consider a d-dimensional hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$ and $l_0 \leq l_1 \leq \ldots \leq l_{d-1}$. Let p be the number of paths in the Cartesian product of hyper-cylinder H. For any fix $0 \leq k < p$, a separate list similar to l_i's is desired such that in can be viewed as separating the k largest paths from the l_i's list.

Definition 5.1.1 (Sequence l'_i)**.** For a fix $0 \leq k < p$, let $l'_{d-k} \leq \cdots \leq l'_{d-1}$ be the length of k largest paths chosen from l_i's, and $l'_0 \leq \cdots \leq l'_{d-k-1}$ the remaining l_is.

Lastly, the index of paths (as opposed to cycles) in l_i's will also be helpful.

Definition 5.1.2 (Index of a path, I_j)**.** For $1 \leq j \leq p$, let I_j be the index of jth occurrence of a path in the l_i's list.

Both I_j for $1 \leq j \leq p$, and list l'_i for particular k are easy and efficient to calculate.

Theorem 5.1.7 (General theorem, Messy model M_3)**.** *Consider a d-dimensional directed hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$ as described above. Let p be the number of paths in this*

Cartesian product of H. With 0 as the originator, the broadcast time of H under Messy model M_3 is

$$t_3(0) = \left[\sum_{i=0}^{d-k^*-1} (i+1)(l'_i - 1)\right] + (d - k^* + 1)\left[\sum_{i=d-k^*}^{d-1} l'_i\right] \tag{1}$$

where l'_i is the list described in Definition 5.1.1 for $k = k^*$. If $p = 0$ or the argument inside $\arg\max$ *of the following Equation 2 is not positive for $k = 0$, $k^* = 0$; otherwise,*

$$k^* = \arg\max_{0 \le k < p} \left\{ \Big[(I_{p-k} + 1) + l_{I_{p-k}}(d - k - I_{p-k} - 1)\Big] - \sum_{i=I_{p-k}+1}^{d-1} l_i > 0 \right\} + 1 \tag{2}$$

Proof. The proof is similar to the proof of Theorem 5.1.6 with some important nuances. Here only these nuances are addressed, due to similarity the overall proof will easily follow.

Firstly, consider the vertices in a strict k-border of H. In the proof of Theorem 5.1.6 it has been concluded that the

$$\max\left\{t_3(0,v) \;\middle|\; v \text{ residing in a strict } k\text{-border}\right\}$$
$$= \left[\sum_{i=0}^{d-k-1} (i+1)(l_i - 1)\right] + (d - k + 1)\left[\sum_{i=d-k}^{d-1} l_i\right]$$

As discussed, the equality happens when only the k largest paths hit the border, which means no increasing the next coefficient, and these k paths are traversed at the end of the scheme. In the Cartesian product of a hyper-cylinder, in addition to paths, there might be some cycles. In cycle C, even if the “end” vertex (depending on which vertex is taken as first) in the associated C is reached, the delay opportunity within it still exists, since the end vertex still can send the message in this direction to the first vertex. A similar argument as Theorem 5.1.6 can be given, noticing that the border hitting paths should be traversed at the end of the scheme to utilize all the delay opportunities that will be created.

In Theorem 5.1.6, since the k border hitting paths do not increase the coefficient, it was easy to see that the maximum is achieved by considering the k largest length paths to be the border hitting ones. The same reasoning applies here. The k largest length path will be at the end of our summation and the rest (path and cycle) sorted in increasing order of their length, increasing the coefficient one by one. Note, for example, that the length of one cycle might be larger than the largest path length; nevertheless, for the maximum to be achieved, it will appear to the left term of our summation. The definition of l'_i helps to conclude that

$$\max\left\{t_3(0,v)\;\middle|\; v \text{ residing in a strict } k\text{-border}\right\}$$
$$= \left[\sum_{i=0}^{d-k-1}(i+1)(l'_i-1)\right] + (d-k+1)\left[\sum_{i=d-k}^{d-1} l'_i\right]$$

Secondly, in the proof of Theorem 5.1.6 the maximizing k^* was found by analyzing the incremental change function from k to $k+1$. In the end, it was concluded that

$$k^* = \underset{0\le k<d}{\arg\max}\left\{d-k-\sum_{i=d-k}^{d-1} l_i > 0\right\}+1$$

The argument here is similar. If $p=0$, then since all vertices reside in a strict 0-border the maximizing $k^*=0$. Otherwise, if $p>0$, consider the change from k to $k+1$ border hitting paths. The range of investigation for k is $0\le k<p$. Since l_i's are sorted the indices of k largest paths (the border hitting ones) within this list are $\{I_{p-k+1},\ldots,I_p\}$. In this change the I_{p-k}-th path with length $l_{I_{p-k}}$ joins the border hitting paths.

Note that for k the sequence $l'_0,\ldots,l'_{I_{p-k}}$ and sequence $l_0,\ldots,l_{I_{p-k}}$ are exactly the same because the k largest paths are to the right of I_{p-k} index. By adding I_{p-k}-th path to the border hitting paths, the change is

$$\Big[(I_{p-k}+1) + l_{I_{p-k}}(d-k-I_{p-k}-1)\Big] - \sum_{i=I_{p-k}+1}^{d-1} l'_i$$
$$= \Big[(I_{p-k}+1) + l_{I_{p-k}}(d-k-I_{p-k}-1)\Big] - \sum_{i=I_{p-k}+1}^{d-1} l_i$$

The first term is the changes with focus on I_{p-k}-th path only, and the second term are changes due to the one reduction of coefficients after I_{p-k} term. Note that $\sum_{i=I_{p-k}+1}^{d-1} l_i\prime = \sum_{i=I_{p-k}+1}^{d-1} l_i$, this is because in this portion the content of both l_i and l'_i are the same just not in the same order.

The change function is strictly decreasing with respect to k, since I_{p-k} is strictly decreasing and also due to the fact that l_is are increasing $\sum_{i=I_{p-k}+1}^{d-1} l_i \geq (d-k-I_{p-k}-1)l_{I_{p-k}}$. As a consequence, if the value for $k=0$ is not positive the maximizing $k^* = 0$. If positive; however,

$$k^* = \underset{0\leq k<p}{\arg\max} \left\{ \Big[(I_{p-k}+1) + l_{I_{p-k}}(d-k-I_{p-k}-1)\Big] - \sum_{i=I_{p-k}+1}^{d-1} l_i > 0 \right\} + 1$$

Note that unlike in Theorem 5.1.6 for $d \geq 2$, change function for $k = p-1$ may still be positive. Since, the maximum range of k is $p-1$, k^* still is well-defined. Example 5.1.1 illustrates this. ■

Corollary 5.1.7.1. *Theorem 5.1.1 and Theorem 5.1.6 under Messy model M_3 are special cases.*

Proof. Let us start with a directed torus $T = \Pi_{i=0}^{d-1} C_{l_i}$. In this case $p=0$; therefore, $k^*=0$.

For $k = 0$, the $l'_i = l_i$ for all $0 \leq i \leq d-1$. Consequently,

$$\begin{aligned} t_3(0) &= \left[\sum_{i=0}^{d-k^*-1} (i+1)(l'_i - 1)\right] + (d-k^*+1)\left[\sum_{i=d-k^*}^{d-1} l'_i\right] \\ &= \sum_{i=0}^{d-1}(i+1)(l_i-1) \end{aligned}$$

in accordance with Theorem 5.1.1.

Now let $G = \Pi_{i=0}^{d-1} P_{l_i+1}$ be a directed grid. In this case $p = d$, $I_j = j-1$ for $1 \leq j \leq d$, and for any k, $l'_i = l_i$. For $k = 0$ the argument inside $\arg\max$ Equation 2 is d a positive number. Consequently, by Equation 2

$$\begin{aligned} k^* &= \operatorname*{arg\,max}_{0 \leq k < p} \left\{ \left[(I_{p-k}+1) + l_{I_{p-k}}(d-k-I_{p-k}-1)\right] - \sum_{i=I_{p-k}+1}^{d-1} l_i > 0 \right\} + 1 \\ &= \operatorname*{arg\,max}_{0 \leq k < d} \left\{ d-k-\sum_{i=d-k}^{d-1} l_i > 0 \right\} + 1 \end{aligned}$$

in accordance with Theorem 5.1.6. According to Equation 1 the broadcast time is

$$\begin{aligned} t_3(0) &= \left[\sum_{i=0}^{d-k^*-1} (i+1)(l'_i - 1)\right] + (d-k^*+1)\left[\sum_{i=d-k^*}^{d-1} l'_i\right] \\ &= \left[\sum_{i=0}^{d-k^*-1} (i+1)(l_i - 1)\right] + (d-k^*+1)\left[\sum_{i=d-k^*}^{d-1} l_i\right] \end{aligned}$$

which is also in accordance with Theorem 5.1.6. ■

Example 5.1.1. *Consider a 4-dimensional hyper-cylinder $H = P_2 \times P_2 \times P_2 \times C_3$, as shown in Figure 29. Using Theorem 5.1.7, it is found that $k^* = 1$. The list l'_i for $k = k^*$ will be $1, 1, 3, 1$, and consequently by Theorem 5.1.7 the broadcast time is*

$$t_3(0) = \left[\sum_{i=0}^{d-k^*-1} (i+1)(l_i - 1)\right] + (d - k^* + 1)\left[\sum_{i=d-k^*}^{d-1} l_i\right]$$
$$= 1 \times 0 + 2 \times 0 + 3 \times 2 + 4 \times 1 = 10$$

The broadcast scheme has been partially drawn in Figure 29. The dark orange part is the maximum path with its delay. The transparent edges signify that the information has already reached the receiver, and this transmission will only cause a delay.

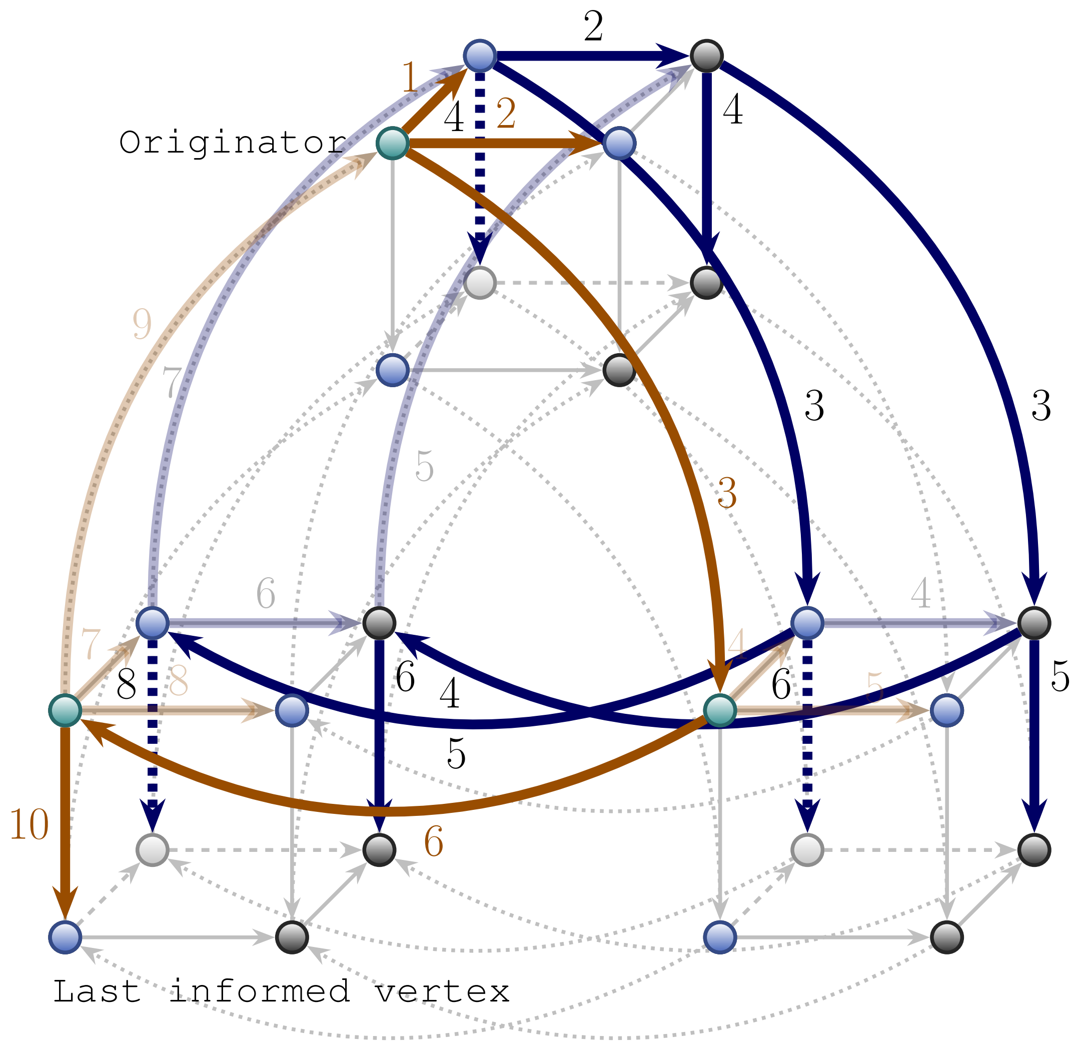


Figure 29: The hyper-cylinder and its partial optimal broadcast scheme of Example 5.1.1. The dark orange part is the maximum path with its delay. All possible originators are indicated with light teal color, and all possible end vertices with light blue color. In the chosen scheme the last informed vertex has been indicated.

Theorem 5.1.8 (General theorem, Messy model M_2)**.** *Consider a d-dimensional directed hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$ such that $l_0 \leq l_1 \leq \cdots \leq l_{d-1}$. Let x be the number of cycles with length two in the Cartesian product of H. With 0 as the originator, under the Messy model M_2*

$$t_2(0) = \begin{cases} t_3(0) - x + 1 & \textit{If } k^* = 0,\ l_{d-1} = 2, \textit{ and there is no path of length 2.} \\ t_3(0) - x & \textit{Otherwise} \end{cases}$$

where k^* is calculated the same way as in Theorem 5.1.7. Additionally, the optimal scheme is similar to that of Theorem 5.1.7 as well, with few minor changes explained in the proof.

Proof. The only obstacle under model M_2 compared to model M_3 is the behavior of cycles of length two (2-cycles). Consider the maximizing path with its delay in an optimal broadcast scheme under model M_2 (for example the dark orange part in Figure 29 is such path for model M_3). For each 2-cycle C, there is a point in which this path goes from corresponding coordinate 0 to 1, a move in associated C. At that point, unlike model M_3, a delay in the same direction C cannot be utilized because the next vertex is the previous sender, which under M_2 is known to be informed. Note that if the length of C exceeds 2, the next vertex would have been different from the previous sender, causing no trouble for the delay move. Consequently, one delay move which was possible under M_3 is not possible here. After any other move; however, the delay move becomes available since the next vertex in direction C is not the same as the previous sender. Accordingly, for each 2-cycle one time-unit less is required compared to model M_3. There is an exception; however, if the 2-cycle move is the last move, in this case, there was no delay under model M_3 to begin with. By the formulation of Theorem 5.1.7 this can happen only if $k^* = 0$, $l_{d-1} = 2$, and there is no path of length 2; meaning the largest component is forced to be a cycle of length 2. See Figure 30 for an example of each case. ■

Corollary 5.1.8.1. *Theorem 5.1.2 including the directed torus whose the length of all of its dimensions is two (symmetric directed hypercube), and Theorem 5.1.6 under model M_2 are special cases.*

Proof. The argument is similar to that of Corollary 5.1.7.1 with the correction mentioned in Theorem 5.1.8 in case cycles of length two exists. ■

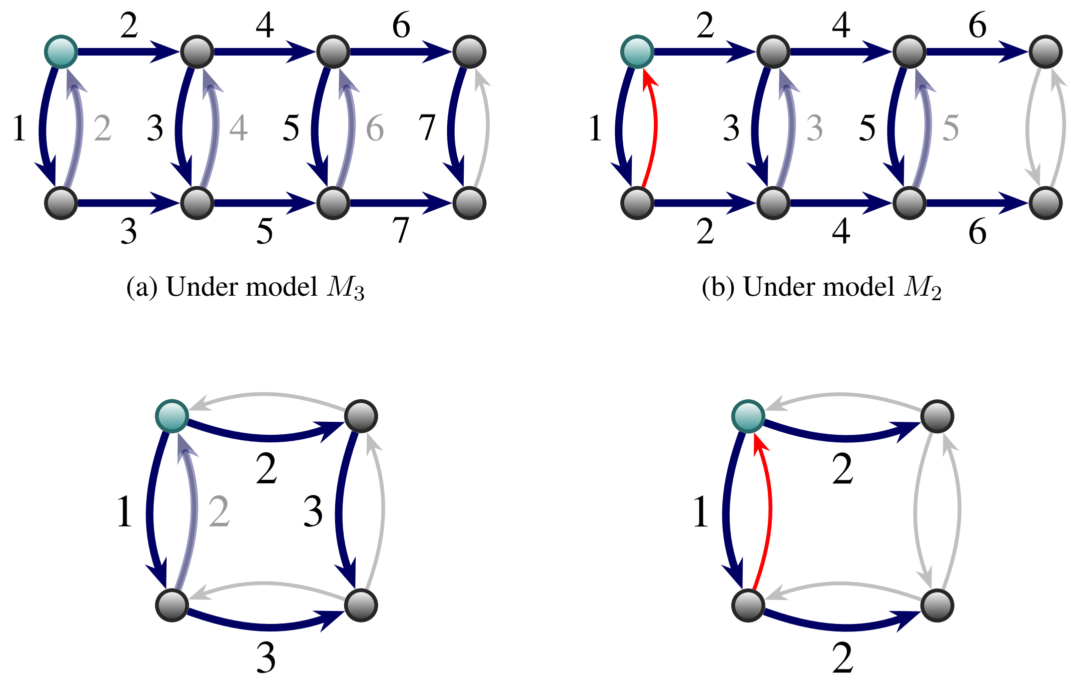
Corollary 5.1.8.2 (Last vertices)**.** *Consider a d-dimensional directed hyper-cylinder $H = \Pi_{i=0}^{d-1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$. With 0 as the originator, under Messy models M_2 and M_3 in any non-lazy optimal broadcast scheme of H the set of last vertices informed are all in strict k^*-border(s) or strict (k^*+1)-border(s) of H, where k^* is the same as in Theorem 5.1.7. The strict (k^*+1)-border is only a possible if $k^* \neq 0$ and $\Big[(I_{p-k^*}+1) + l_{I_{p-k^*}}(d - k^* - I_{p-k^*} - 1)\Big] - \sum_{i=I_{p-k^*}+1}^{d-1} l_i = 0$.*

Proof. This is the direct consequence based on the proof of Theorem 5.1.7 and Theorem 5.1.8, especially the fact that the change function mentioned is strictly decreasing and cannot have two consecutive 0 values. Therefore, there can be only one or two consecutive maximizing k's. See Figure 29 for an example. ■

Example 5.1.2. *Figure 30 provides an example for each case of Theorem 5.1.8, and compares it with the similar optimal scheme under Messy model M_3.*

Theorem 5.1.9 (2-dimensional hyper-cylinder, Messy model M_1)**.** *Consider a 2-dimensional directed hyper-cylinder $H = \Pi_{i=0}^{1} H_i$, where $H_i \in \{P_{l_i+1},\ C_{l_i}\}$. With 0 as the originator, $t_1(H) = l_0 + l_1 - x$ where $2 \leq l_0 \leq l_1$ and x is the number of cycles in Cartesian product of H.*

Proof. In [30], Theorem 5.1.3 was proved using a simple induction on $l_0 + l_1$. The main idea is the fact that if vertex (a, b) is informed at time-unit t then all vertices $\{(x, y) | 0 \leq x \leq a, 0 \leq y \leq b\}$ are informed by time-unit t as well. Consider graph H as grid whose top



(a) Under model M_3 (b) Under model M_2

(c) Under model M_3 (d) Under model M_2

Figure 30: Comparing the optimal scheme under models M_2 and M_3. The red edge is the delay that cannot be utilized under model M_2 at that particular time. Part (d) is the first case of Theorem 5.1.8, and part (b) is the second case.

left vertex is 0, with additional "loop back" edges for every cyclic dimension. For example, for $C_4 \times C_4$ the graph will look like directed version of Figure 14. By not utilizing the loop back edges, the graph becomes a grid and the above idea will apply; accordingly, here too by time-unit t all vertices in $\{(x, y)|0 \leq x \leq a, 0 \leq y \leq b\}$ are informed. Since the bottom right vertex will feasibly be informed at $l_0 + l_1 - x$ at the latest, the result is proved. ■

Corollary 5.1.9.1. *Theorem 5.1.3 is a special case.*

5.2 Undirected Graphs

This section starts with inquiring about the value of $t_3(u, v)$ in an infinite grid. Like before, define G' as the induced subgraph on the vertex set

$$V' = \left\{ w \;\middle|\; w \in V(G),\ u_i \leq w_i \leq v_i \text{ for } 0 \leq i \leq d-1 \right\}$$

where u is the originator. Due to vertex transitivity, it is suffice to just calculate $t_3(0, v)$. The idea used in finding this value and an optimal broadcast scheme for the directed infinite grid is helpful here; however, two main problems exist. The first problem is that if a path encounters (hit) a border of G' unlike in the directed infinite grid, the rest of the path is not forced to remain in that border. The second problem is that the improving move defined before is not applicable here, since a path may "circle back" yet informing new vertices and be crucial for the optimal path. To fix these problems, improving/delay move will be redefined, and an improving path will be considered, which may not be part of an optimal scheme.

Definition 5.2.1 (Saturated coordinate)**.** For $0 \leq i \leq d-1$, the ith coordinate of vertex w is called saturated if $w_i = v_i$.

Theorem 5.2.1 (Infinite grid, Messy model M_3)**.** *Consider a d-dimensional infinite grid $G = \Pi_{i=0}^{d-1} P_\infty$, and 0 as the originator. Under Messy model M_3, for any vertex v*

$$t_3(0, v) = \sum_{i=0}^{d-1} (d + i + 1)(|v_{\pi(i)}|)$$

where π is a permutation of $0, 1, \cdots, d-1$ such that $|v_{\pi(0)}| \leq |v_{\pi(1)}| \leq \cdots \leq |v_{\pi(d-1)}|$.

Proof. For simplicity, first, assume that all coordinates of v are non-negative. By our naming convention, at each message relay, the coordinate of the neighboring vertex is obtained by changing one of the coordinates of the current vertex by one. For each vertex $u \in V'$, a message relay is called an improving move if the receiver's distance to v is reduced. Alternatively, in an improving move, one unsaturated coordinated is increased by one. Call the remaining moves a "delay" move.

Starting at 0, *improving path* consists of first encountered improving move for each of its vertexes until it reaches v. This definition is well defined because (1) each vertex of the improving path has an improving move and since the degree of every vertex is $2d$ eventually for each such vertex an improving move will be made, and (2) after $\sum_{i=0}^{d-1} v_i$ improving moves the path is forced to be at v. Consequently, the length of the improving move is $\sum_{i=0}^{d-1} v_i$. If improving path hits any of the G' borders, the rest of the path will remain there. These properties allow us to use a similar argument as in the proof of Theorem 5.1.4.

In short, for $0 \leq i \leq d$ let u^i be the first vertex of the improving path that hits a i-border of G'. Note that $u^0 = 0$, $u^d = v$, and different u^i's may be the same vertex. Until hitting the first border only d moves of each vertex is a delay move. More generally, for the vertices between u^i and u^{i+1} including u^i but excluding u^{i+1} each vertex has $d + i$ delay moves. With similar argument to the proof of Theorem 5.1.4 it is easy to see that

$$t_3(0, v) \leq \sum_{i=0}^{d-1} (d + i + 1)(v_{\pi(i)})$$

where π is a permutation of $0, 1, \cdots, d-1$ such that $v_{\pi(0)} \leq v_{\pi(1)} \leq \cdots \leq v_{\pi(d-1)}$. An optimal scheme achieving this bound works as follows. Permutation π indicates an order on dimensions. Every vertex v after receiving the information sends it to the neighbors in the negative direction of dimensions by the order indicated by π. After that, v uses the same order of dimensions and sends the information to the neighbors in their positive direction. It is easy to see that this scheme achieves the bound mentioned above. See Figure 31 for an example.

For general vertex v, a coordinate may be negative. For such an unsaturated coordinate the distance is reduced if the coordinate is decreased by one; therefore, for any v

$$t_3(0, v) = \sum_{i=0}^{d-1} (d + i + 1)(|v_{\pi(i)}|)$$

where π is a permutation of $0, 1, \cdots, d-1$ such that $|v_{\pi(0)}| \leq |v_{\pi(1)}| \leq \cdots \leq |v_{\pi(d-1)}|$.

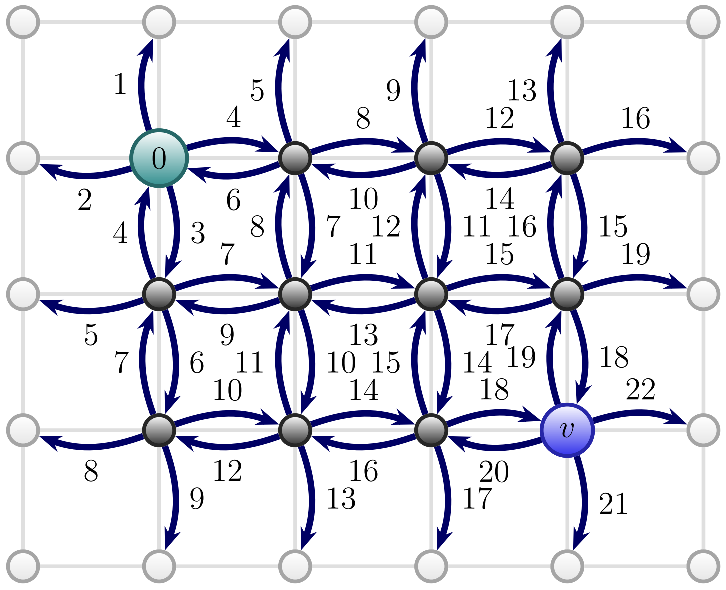


Figure 31: An optimal broadcast scheme for vertex $v = (3, 2)$ in 2-dimensional infinite grid $G = P_\infty \times P_\infty$. Vertex v is informed at time-unit $(2 + 0 + 1) \times 2 + (2 + 1 + 1) \times 3 = 18$.

■

Grid graphs. Consider a 2-dimensional grid $G = G[n+1,\ m+1] = P_{n+1} \times P_{m+1}$ for some integers $n \geq 1$ and $m \geq 1$. By symmetry, without the loss of generality, assume $n \leq m$. The difference between a grid and an infinite grid graph in Messy broadcasting is that in a grid, the maximum possible delay any k-border vertex could cause is smaller than that of $k - p$-border vertices, where $p > 0$. Intuitively, for a 2-dimensional case, since interior vertices may cause more delays, it is implied that there is an optimal broadcast scheme in which for any vertex v, the path that first informs v creeps through the borders first until it branches out in a straight line to v.

First move. Which direction should the originator 0 send the message first? Our intuition tells us that the shorter dimension (n, the vertical) should be informed first. The reason is as follows: As discussed, the bottleneck for delaying is the borders; therefore, the optimal scheme should delay informing them as much as possible. If the message is sent through the shorter dimension first, while it is true that the shorter border will be informed sooner, it is also true that it delays the progression of information toward the longer dimension (to the right) with the help of interior vertices and the fact that the bottom border already has significant enough delay. Based on our intuition, *assume* that the first move is downward along the shorter dimension. Later we will prove that this is an optimal choice.

For a vertex v, the maximum time of all possible border creeping paths is to be investigated. In our notation u means going up, d going down, r going right, and l going left.

Definition 5.2.2 (Border creeping paths)**.** Inspired by the above motivation for a vertex v consider all the 4 border creeping paths; namely, dr-path, dru-path, rd-path, and rdl-path. For example, dru-path starts at 0, creeping the border downward until it reaches the bottom border, then goes right on the bottom border until it reaches the horizontal position of v, then finally, it goes up (not necessarily on any border) until v is reached. The last change of direction is the only part of these paths that may not be on any border.

It is easy to see that for an interior vertex $v = (i, j)$ the maximum time it takes each border creeping path to reach v is:

$$\begin{aligned}
t_{dr}(v) &= 1 + 3(i-1) + 3 + 4(j-1)\\
t_{dru}(v) &= 1 + 3(n-1) + 2 + 3(j-1) + 3 + 4(n-i-1)\\
t_{rd}(v) &= 2 + 3(j-1) + 3 + 4(i-1)\\
t_{rdl}(v) &= 2 + 3(m-1) + 3 + 3(i-1) + 3 + 4(m-j-1)
\end{aligned}$$

For example, let us investigate the dru-path. The 1 is the maximum time it will take to reach the vertex one down of 0 (originator), the $3(n-1)$ to reach the bottom border, the 2 for changing the direction at the corner vertex and go one to the right, the $3(j-1)$ to reach the horizontal position of v, the 3 to change the direction upward and going one up, the $4(n-i-1)$ to go up and reach v. See such portion of the scheme in Figure 32. By these values, an easy upper bound to $t_3(0,v)$ can be given for interior vertices.

Lemma 5.2.2 (Upper-bound, Interior vertices)**.** *Let $v=(i,j)$ be an interior vertex, then*

$$t_3(0,v) \leq \min\{t_{dr}(v),\ t_{dru}(v),\ t_{rd}(v),\ t_{rdl}(v)\}$$

Similar upper-bound can be given to border vertices as well; here however, it is clear which border creeping path has the minimum time.

Lemma 5.2.3 (Upper-bound, Border vertices)**.** *Let $v=(i,j)$ be an border vertex, then*

$$t_3(0,v) \leq \begin{cases} 0 & i=0 \textit{ and } j=0 \\ 2+3(j-1) & i=0 \textit{ and } 0<j\leq m \\ 1+3(i-1) & j=0 \textit{ and } 0<i\leq n \\ 1+3(n-1)+2+3(j-1) & i=n \textit{ and } 0<j\leq m \\ 2+3(m-1)+2+3(i-1) & j=m \textit{ and } 0<i\leq n \end{cases}$$

Comparison between the maximum time of each two border creeping paths P_1 and P_2 separates the vertices into two groups; the vertices whose P_1 maximum time is less than equal to that of P_2, and the rest. The boundary is a line, let us denote it with

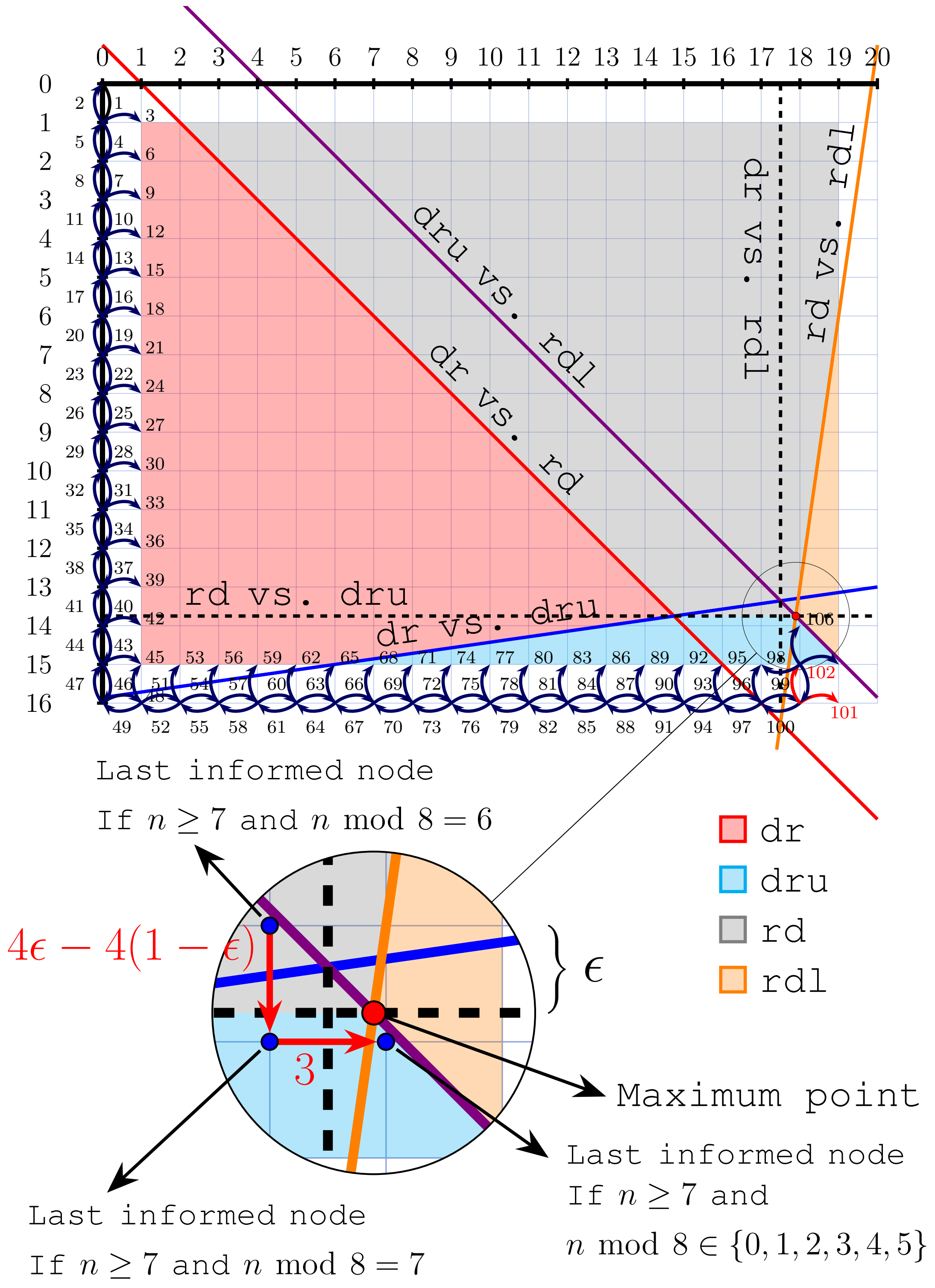


Figure 32: Broadcasting under Messy model M_3 in $G = P_{17} \times P_{21}$.

$L(\text{type }1, \text{type }2)$. The equations associated with each of these six lines is as follows.

$$
\begin{aligned}
L(dr,\ rd):&\quad i = j-1\\
L(dr,\ dru):&\quad i = n-\left(\frac{j+1}{7}\right)\\
L(rd,\ rdl):&\quad i = 7m-7j-1\\
L(dru,\ rdl):&\quad i = (n-m)+j-\frac{1}{7}\\
L(rd,\ dru):&\quad i = \frac{7n-2}{8}\\
L(dr,\ rdl):&\quad j = \frac{7m}{8}
\end{aligned}
$$

An example is shown in Figure 32 which also explain how to detect the regions that a particular type has the minimum, meaning that type reached those vertices first.

Lemma 5.2.4 (Common intersections)**.** *Each of the following 4 groups of three lines, have a common intersection.*

1. $\Big\{L(dr,\ rd),\ L(dr,\ dru),\ L(rd,\ dru)\Big\}$
2. $\Big\{L(dru,\ rdl),\ L(rd,\ rdl),\ L(rd,\ dru)\Big\}$
3. $\Big\{L(dru,\ rdl),\ L(dr,\ dru),\ L(dr,\ rdl)\Big\}$
4. $\Big\{L(dr,\ rd),\ L(rd,\ rdl),\ L(dr,\ rdl)\Big\}$

Proof. The common point for group 1 is $\left(\frac{7n-2}{8},\ \frac{7n+6}{8}\right)$, for group 2 is $\left(\frac{7n-2}{8},\ m-\frac{n+6/7}{8}\right)$, for group 3 is $\left(n-\frac{m}{8}-\frac{1}{7},\ \frac{7m}{8}\right)$, and for group 4 is $\left(\frac{7m}{8}-1,\ \frac{7m}{8}\right)$. ■

For a scheme S, let $t_S(v)$ be the time it takes S to inform v for the first time; clearly, $t_3(0,v) = \max_S\{t_S(v)\}$ for all schemes S. For any vertex v, let $x(v) \in \{dr,\ dru,\ rd,\ rdl\}$ be the type of border creeping path which achieves the aforementioned upper-bound for $t_3(0,v)$.

In any scheme S, if $t_S(v) = t_{x(v)}(v)$ the behavior of the scheme throughout $x(v)$-path is uniquely determined. This is because there is only one behavior that causes the vertices of $x(v)$-path to delay the information to v for this long, any other behavior will inform v

sooner. Knowing this, we ask whether there exists a scheme S such that for any vertex v, $t_S(v) = t_{x(v)}(v)$. If such scheme exists it would be an optimal scheme. Unfortunately, for large enough n and m, such scheme cannot exists. Consider a grid with $n \geq 2$ and $m \geq 1$, then $x((1,1)) = x((2,1)) = dr$. The dr-path for $(2,1)$ shares the dr-path for $(1,1)$ until vertex $(1,0)$. The requirements of the described unique behavior for vertex $(1,0)$ are different depending which of these two vertices are considered. Therefore, such S cannot exists.

Border evading scheme. For any interior vertex v, in the calculations of $t_{x(v)}(v)$ for the last direction change +3 was considered, which can only be achieved if at that point the message relays are first sending the message to the previous sender, then to the next vertex on the border, and lastly toward v. If from the unique behavior we only change this part so that it first sends to the previous sender, and then toward v we have a behavior that informs v in $t_{x(v)}(v) - 1$ time-units. It is as if the new behavior wanted to evade informing the rest of the border as much as possible. For any border vertex v, no changes is considered for the behavior; accordingly, the behavior will inform v at $t_{x(v)}(v)$ time-unit. In light of the new behavior for all vertices, it becomes clear that combining them produces no conflict (unlike before); therefore, describing a general scheme. Name this scheme *border evading scheme*. In the border evading scheme, for all interior vertices, all border creeping paths times are reduced exactly by one compared to their maximum. Accordingly, the regions associated with the earliest reaching border creeping paths (detected before) remain the same.

Thus far it is deduced that for border vertices $t_3(0, v) = t_{x(v)}(v)$ and for interior vertices $t_{x(v)}(v) - 1 \leq t_3(0, v) \leq t_{x(v)}(v)$. Let B be the set of border vertices, then since $t_3(0) = \max_{v \in V(G)} \{t_3(0, v)\}$ we have

$$\max\left\{\max_{v \in B}\left\{t_{x(v)}(v)\right\},\ \max_{v \in V(G)-B}\left\{t_{x(v)}(v) - 1\right\}\right\} \leq t_3(0) \leq \max_{v \in V(G)}\left\{t_{x(v)}(v)\right\}$$

Let

$$v^* = \underset{v \in V(G)}{\arg\max} \left\{ \max_{v \in B} \left\{ t_{x(v)}(v) \right\}, \max_{v \in V(G)-B} \left\{ t_{x(v)}(v) - 1 \right\} \right\}$$

with a preference to be an interior vertex. In this way, if $v^* \in B$, then for any interior vertex v, $t_3(0, v^*) \geq t_3(0, v)$; consequently, $t_3(0) = t_3(0, v^*)$ and border evading scheme is an optimal scheme. Otherwise, if $v^* \in V(G) - B$, if we can modify the border evading scheme in such a way that v^* in the new scheme is informed at time-unit $t_{x(v^*)}(v^*)$ then the new scheme will be an optimal scheme, and $t_3(0) = t_{x(v^*)}(v^*)$. To do that first let us find v^*.

Continuous view. Let us refer to the previously detected regions of the grid associated with the type of border creeping path that reaches a vertex first, by just regions. In the border evading scheme, if a message relay is entirely within one region, the change between the first informed time of the sender and receiver is known. For example, in the region associated with dr, a move to the right has +4 change while in dru has +3 and in rdl has -4. One can change the view from discrete movements (e.g., going up by one unit) to a continuous one. In this view, define the gradient of each point within a region based on the aforementioned uniform changes; in case a point belongs to multiple regions, choose one randomly. The vector field is conservative since it can be viewed as the gradient of a function composed of unions of linear equations. To know the first informed time of a vertex v, consider a point p at the coordinate of v. For any continuous path from 0 to v (due to path independence), the line integral over the vector field is the answer. This time can be associated with any other point in the grid. As an example, take a look at Figure 32, under the magnifying glass. As indicated, the change in the first informed time of the two vertical blue dots is $4\epsilon + (-4)(1 - \epsilon)$.

Using the continuous view, and Lemma 5.2.4 it is clear that point p^* with the most associated first informed time is the intersection of $L(dru,\ rdl)$, $L(rd,\ rdl)$, and $L(rd,\ dru)$ lines; $p^* = \left(\frac{7n-2}{8},\ m - \frac{n+6/7}{8}\right)$. This is true even for the case $n = m$, the only case in which $L(dru,\ rdl)$ comes before $L(dr,\ rd)$. **Important note:** The placement of p^* was determined by the separating lines and not by direct investigation of the line integrals. Take $L(dr,\ dru)$, for example, its equation was derived from $t_{dr}(v) = t_{dru}(v)$. The latent assumption in the value of $t_{dru}(v)$ is that v's vertical position is not n; which comes from the +2 applied for changing the direction upward. Consequently, if $\frac{7n-2}{8} > n-1$ or $m - \frac{n+6/7}{8} > m-1$ the above position might be problematic and additional checks are required. The two restrictions are false if $n \geq 7$.

For $n \geq 7$, based on the vector fields, the choice for v^* will be limited to three possible vertices in the vicinity of p^*;

1. $v_1^* = \left(\left\lfloor \frac{7n-2}{8} \right\rfloor,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil\right)$
2. $v_2^* = \left(\left\lfloor \frac{7n-2}{8} \right\rfloor + 1,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil\right)$
3. $v_3^* = \left(\left\lfloor \frac{7n-2}{8} \right\rfloor + 1,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil + 1\right)$

See Figure 32, under the magnifying glass. The red point is p^* and the possible three vertices have blue points. To understand which has the higher first informed time, the changes from v_1^* to v_2^* and then from v_2^* to v_3^* will be calculated. Let $\epsilon = \frac{7n-2}{8} - \left\lfloor \frac{7n-2}{8} \right\rfloor$.

By the slope of the lines, it is deduced that vertices v_2^* and v_3^* are in the dru region (cyan color in Figure 32), and vertex v_1^* in rd region (gray). Note that if $n \bmod 8 = 1$ then v_1^* is in the boundary of rd and dr regions, in this case we consider the rd region. As depicted in Figure 32, these positioning implies that the change from v_1^* to v_2^* is $4\epsilon + (-4)(1-\epsilon)$ and from v_2^* to v_3^* is $+3$. Consequently, vertex v_3^* is the sole last informed vertex except when (1) $n \bmod 8 = 5$, where v_1^* is also the last informed; (2) $n \bmod 8 = 6$, where v_1^* is the sole last informed vertex; and (3) $b \bmod 8 = 7$, where v_2^* is the sole last informed vertex.

On the other hand if $n \leq 6$, then a similar analysis points to 3 candidates for v^*; namely, $(n-1,\ m-1)$, $(n,\ m-1)$, and $(n,\ m)$. The change from an vertex at vertical position $n-1$ to n can be anything from $+3$ to -2, and the change from $(n,\ m-1)$ to $(n,\ m)$ is

$+3$; consequently, $(n,\ m)$ will be the sole last informed vertex.

Optimal scheme, changing the border evading scheme. Now that all the possible candidates for v^* (and their time) is determined for border evading scheme, we can describe a new similar scheme that will be completed in $t_3(0, v^*)$ and hence is an optimal scheme. If $n \le 6$, no change is necessary. Otherwise, (1) if $n \bmod 8 \in \{0,1,2,3,4,5\}$ change the behavior at $(n,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil + 1)$ so that at that point there is no evading the border as described before; see Figure 32 the modification of the behavior is in red. In this way, the dru-path time to v^* is increased by one and since dru-path was the only path that could inform the fastest, the desired scheme is achieved. (2) If $n \bmod 8 = 7$, do the similar modification but this time for $(n,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil)$ to match the v^*; (3) If $n \bmod 8 = 6$, in this case v^* resides (only) in the rd region; therefore, with the similar justification modify the behavior of $(0,\ m - \left\lceil \frac{n+6/7}{8} \right\rceil)$.

First move revisited. At first it was assumed that the first move is downward, through the smaller dimension. A similar analysis can be made if the first move is toward right. In that case the position of v^* is shifted to the left and since the best path for v^* is dru-path its completion time will be reduced. Consequently, sending though the smaller dimension as the first move is an optimal choice.

To summarize all the findings:

Theorem 5.2.5 (2-dimensional grid, Messy model M_3)**.** *Consider a 2-dimensional grid $G = P_{n+1} \times P_{m+1}$, where $n \le m$. With 0 as originator, under the Messy model M_3,*

$$t_3(0) = \begin{cases} 3m + 3n - 2 & \textit{if } n \le 6 \\ 3m + 3n - 5 + \left(4n - 3\left\lceil \frac{n+6/7}{8} \right\rceil - 4\left\lfloor \frac{7n-2}{8} \right\rfloor\right) & \textit{if } n \ge 7 \textit{ and } n \bmod 8 \in \{0,1,2,3,4,5\} \\ 3m + 3n - 4 + \left(4n - 3\left\lceil \frac{n+6/7}{8} \right\rceil - 4\left\lfloor \frac{7n-2}{8} \right\rfloor\right) & \textit{if } n \ge 7 \textit{ and } n \bmod 8 = 6 \\ 3m + 3n - 8 + \left(4n - 3\left\lceil \frac{n+6/7}{8} \right\rceil - 4\left\lfloor \frac{7n-2}{8} \right\rfloor\right) & \textit{if } n \ge 7 \textit{ and } n \bmod 8 = 7 \end{cases}$$

Moreover, an optimal scheme similar to border evading scheme *is known, in which the sole last informed vertex is*

$$\begin{cases} (n,\ m) & \textit{if } n \leq 6 \\ \left(\left\lfloor \frac{7n-2}{8} \right\rfloor + 1,\ \ m - \left\lceil \frac{n+6/7}{8} \right\rceil + 1\right) & \textit{if } n \geq 7 \textit{ and } n \bmod 8 \in \{0,1,2,3,4,5\} \\ \left(\left\lfloor \frac{7n-2}{8} \right\rfloor,\ \ m - \left\lceil \frac{n+6/7}{8} \right\rceil\right) & \textit{if } n \geq 7 \textit{ and } n \bmod 8 = 6 \\ \left(\left\lfloor \frac{7n-2}{8} \right\rfloor + 1,\ \ m - \left\lceil \frac{n+6/7}{8} \right\rceil\right) & \textit{if } n \geq 7 \textit{ and } n \bmod 8 = 7 \end{cases}$$

Proof. The optimal scheme and its last informed vertex v^* has been explained above. Knowing $v*$, $t_3(0) = t_{x(v^*)}(v^*)$. ■

Corollary 5.2.5.1. *Consider a 2-dimensional grid $G = P_{n+1} \times P_{m+1}$, where $n \leq m$. Under the Messy model M_3, for large enough n*

$$t_3(0) \approx 3m + 3\frac{1}{8}n$$

Chapter 6

Worst-case Originators and Diameters in Trees

One of the fundamental case studies of broadcasting is broadcasting in trees. Trees were among the first structures for which broadcasting was studied, not only because of their simple and common structure but also because every broadcasting scheme describes a tree that is a connected subtree of some binomial tree.

Consider a rooted tree T, with root vertex r. For every vertex $v \in V(T)$, let T_v be the subtree of T with root v. If $u_1,\ u_2,\ \ldots,\ u_p$ are children of vertex v such that $b(T_{u_1},\ u_1) \geq b(T_{u_1},\ u_1) \geq \cdots \geq b(T_{u_p},\ u_p)$, [100] showed that

$$b(T_v,\ v) = \max_{1 \leq i \leq p} \{b(T_{u_i},\ u_i) + i\} \tag{3}$$

Knowing this relation, a linear recursive algorithm could be given to calculate $b(T,\ r)$. The base cases are leaves v, where $b(T_v,\ v) = 0$. Equation (3) also describes an optimal scheme. After vertex v is informed, under this optimal scheme v informs u_1 to u_p in order; meaning in order of their broadcast time.

One follow-up question is then, what is the broadcast time of the tree? Of course, one might take each vertex as the root vertex run the previous algorithm to find the broadcast time of every vertex; and, consequently, the broadcast time of the tree. However, the time complexity will be of $O(n^2)$, where n is the number of vertices. Intuition might suggest that at least one of the end vertices of every diameter of the tree is the worst-case originator. If that is the case, then one can find both end vertices of a diameter in $O(n)$, run the aforementioned recursive algorithm for both and compare the results all in $O(n)$, and find $b(T)$ in only $O(n)$ time and space. Algorithms and situations like these make it worthwhile to study the relationship between the worst-case originators and the end vertices in diameters. Another situation to consider is when instead of a path, any tree is allowed as a primitive graph in the Cartesian product for a more general class of graphs to consider for broadcasting. Here, one might think that by studying only the end vertex of a diameter of trees, the broadcast time of the structure might be found. In the following, we analyze the relationship between worst-case originators and dimeters in trees, which seems to be missing from the literature, and show that the previous intuition is not correct.

Question 6.0.1. *Is it true that any worst-case originators is an end vertex of a diameter in tree?*

Answer. No, Figure 33 is an example. ■

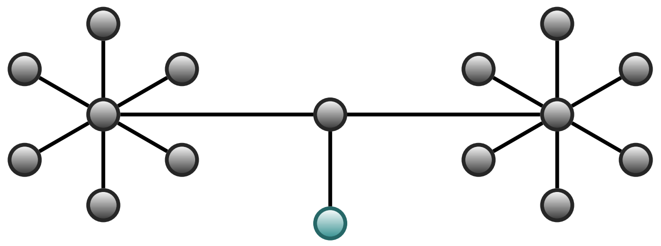


Figure 33: A worst-case originator may not reside in any diameter.

Question 6.0.2. *Is it true that at least one end vertex of any diameter in the tree T is a worst-case originator?*

Answer. No, Figure 34 is an example. ■

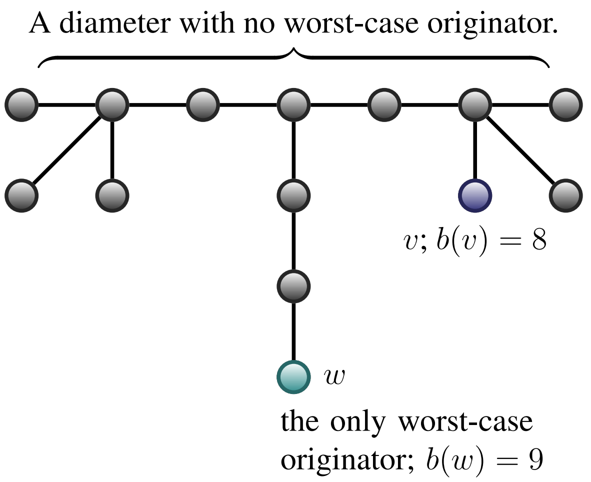

A diameter with no worst-case originator.
v; $b(v) = 8$
w
the only worst-case originator; $b(w) = 9$

Figure 34: A diameter may not contain any worst-case originators.

Question 6.0.3. *Is it possible that none of the diameters' end vertices are worst-case originator?*

Answer. No, at least one end vertex of a diameter is a worst-case originator. To prove this, by the sake of contradiction assume otherwise. Make the tree rooted from one of the worst-case originators, call it w; see Figure 35. Note that all the worst-case originators are leaves. Consider a diameter xyz where x and z are the leaves and y is the common ancestor. By our assumption $x \neq w$ and $z \neq w$. Let $d(x,\ y) = L$ and $d(z,\ y) = L'$; therefore, the length of the diameter is $L + L'$. Without the loss of generality assume $L' \leq L$.

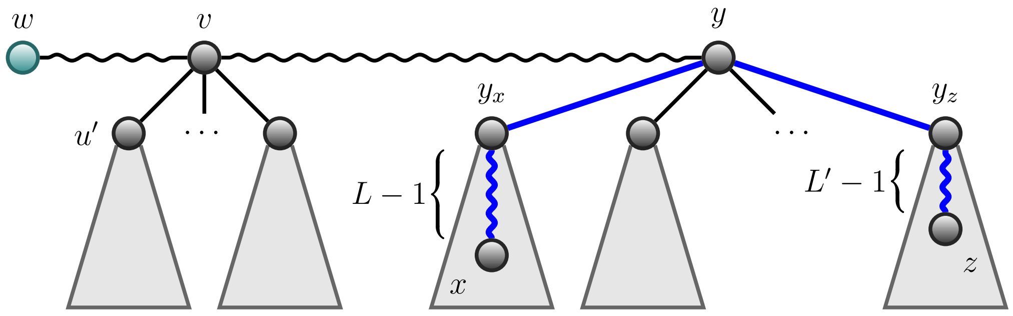


Figure 35: The tree rooted at the worst-case originator w. Path xyz is one diameter with the length $L + L'$.

Since w is not the end vertex of any diameter, $d(w,\ y) < L' \leq L$. Under any optimal scheme, any vertex v in the wy-path excluding y has no choice but to inform the next vertex in this path. For w this is true since w is a leaf. Assume this assertion is not true, in that case consider the first vertex v which does not obey this behavior. Let u be the child of v such that $y \in T_u$; define $T_{v-y} = T_v - T_u$. Lack of obedience means u is not the first child to get informed after v is informed. Accordingly, there is a child of v, called u' such that $b(T_{u'},\ u') \geq b(T_u,\ u)$ and the optimal scheme chose u' as the first child to be informed after v. Since $b(T_{u'},\ u') \geq b(T_u,\ u)$, by Equation 3

$$b(T_v,\ v) - 1 \leq b(T_{v-y},\ v) \leq b(T_v,\ v)$$

Since v is the first anomaly, $b(w) = d(w,\ v) + b(T_v,\ v)$. Consider vertex x as the originator, then $b(x) \geq d(x,\ v) + b(T_{v-y},\ v)$. This is true since the only way to inform vertices of T_{v-y} is through vertex v and the shortest distance to v is $d(x,\ v)$. By our first assumption, x is not a worst-case originator; therefore, $b(x) < b(w)$. However, since $d(y,\ v) \geq 1$ and $d(w,\ v) \leq d(w,\ y) \leq L$

$$b(x) \geq d(x,\ v) + b(T_{v-y},\ v) \geq L + d(y,\ v) + b(T_v,\ v) - 1 \geq d(w,\ v) + b(T_v,\ v) = b(w)$$

a contradiction.

This proves that $b(w) = d(w,\ y) + b(T_y,\ y)$. Note that z is in a different child subtree of y (under T_{y_z}) than x (under T_{y_x}). At least one of y_x or y_z will not get informed the next move after the optimal scheme reaches y. Assume y_z is not informed at that time. This implies that $b(T_{y_z},\ y_z)$ is not the sole greatest broadcast time of children of y; consequently, with the similar argument to above

$$b(y) \geq L' + b(T_y,\ y) - 1 \geq d(w,\ y) + b(T_y,\ y) = b(w)$$

This is a contradiction, since by our first assumption y should not be a worst-case originator. On the other hand if y_x is not informed at that time, similarly we have

$$b(x) \geq L + b(T_y,\ y) - 1 \geq d(w,\ y) + b(T_y,\ y) = b(w)$$

again a contradiction. As a result, our first assumption that no diameter's end vertex is a worst-case originator is incorrect. ■

Question 6.0.4. *Is there a diameter that both of its end vertices are worst case originator?*

Answer. Not necessarily, Figure 34 is an example. ■

Chapter 7

Conclusion and Future Work

Inspired by the practical and efficient hypercube and torus topologies in computing networks, a new, more general class of graph called *hyper-cylinders* was introduced, which have similar features. A hyper-cylinder is a graph that is obtained by the Cartesian product of some paths and cycles. In this thesis, the broadcast problem was studied for hyper-cylinder under both Classical and Messy models. As a consequence, some hyper-cylinders were added to the limited classes of graphs for which broadcast time is polynomially computable. For future work, it will be interesting to study the classes of graphs that can be obtained using the Cartesian product of other simple structures. In particular, the use of any tree (or arborescence in directed graphs) instead of paths. At the end of the thesis, the position of worst-case originators is studied, which might help with this generalization.

The broadcasting in hyper-cylinder graphs under the Classical model is studied in chapter 4. The result for grid and torus networks was improved, and some bounds for hyper-cylinders, in general, were provided. In particular, the exact result is found for directed hyper-cylinders and for a useful type of undirected hyper-cylinders described in Remark 4.2.3. For future work, the following can be improved. The exact broadcast time and optimal scheme, or a polynomial algorithm for calculating it for vertices in a grid that are at least in the exact middle of two dimensions. For d-dimensional torus where $d \geq 3$;

although a convincing conjecture is given, the formal proof for the exact broadcast time and the optimal scheme is missing. Accordingly, the exact result for general undirected hyper-cylinders is missing as well; however, we believe that if the result for torus and grid is found, the result for hyper-cylinders will follow.

In chapter 5, the broadcasting in hyper-cylinders under Messy models is studied. For the directed graphs, the exact result for Messy models M_2 and M_3 are found; however, for Messy model M_1 the result is only proved for 2-dimensional hyper-cylinders. For future work, Messy model M_1 can be studied further; especially, in hypercubes. For the undirected graphs, the result for the infinite d-dimensional grid and 2-dimensional grid under Messy model M_3 was found. For the latter, a continuous view technique was introduced, which we believe has a high potential in finding the broadcast time and optimal scheme of general d-dimensional grids. The result for any other type of hyper-cylinders is still unknown. Other broadcasting models or criteria, particularly the study of fault tolerance in hyper-cylinders, are also an interesting topic for future research.

Bibliography

[1] Dennis Abts. “The Cray XT4 and Seastar 3-D torus interconnect”. In: (2011).

[2] Narasimha R Adiga, Matthias A Blumrich, Dong Chen, Paul Coteus, Alan Gara, Mark E Giampapa, Philip Heidelberger, Sarabjeet Singh, Burkhard D Steinmacher-Burow, Todd Takken, et al. “Blue Gene/L torus interconnection network”. In: *IBM Journal of Research and Development* 49.2.3 (2005), pp. 265–276.

[3] R Ahlswede, H Haroutunian, and L Khachatrian. *Messy broadcasting in networks, Communications and Cryptography, eds. RE Blahut, DJ Costello Jr., U. Mauter, and T. Mittelholzer*. 1994.

[4] Stefan Arnborg, Jens Lagergren, and Detlef Seese. “Easy problems for tree-decomposable graphs”. In: *Journal of Algorithms* 12.2 (1991), pp. 308–340.

[5] Baruch Awerbuch, Oded Goldreich, David Peleg, and Ronen Vainish. “A trade-off between information and communication in broadcast protocols”. In: *Aegean Workshop on Computing*. Springer. 1988, pp. 369–379.

[6] Anindo Bagchi and S. Louis Hakimi. “Data transfers in broadcast networks”. In: *IEEE Computer Architecture Letters* 41.07 (1992), pp. 842–847.

[7] Anindo Bagchi, S. Louis Hakimi, and Edward F. Schmeichel. “Gossiping in a distributed network”. In: *IEEE transactions on computers* 42.2 (1993), pp. 253–256.

[8] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. “Multicasting in heterogeneous networks”. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 448–453.

[9] Reuven Bar-Yehuda, Amos Israeli, and Alon Itai. “Multiple communication in multihop radio networks”. In: *SIAM Journal on Computing* 22.4 (1993), pp. 875–887.

[10] René Beier and Jop F. Sibeyn. “A powerful heuristic for telephone gossiping”. In: *SIROCCO 7, Proceedings of the 7th International Colloquium on Structural Information and Communication Complexity, Laquila, Italy, June 20-22, 2000*. Ed. by Michele Flammini, Enrico Nardelli, Guido Proietti, and Paul G. Spirakis. Carleton Scientific, 2000, pp. 17–35.

[11] Amir Ben-Dor, Shai Halevi, and Assaf Schuster. *On greedy hot-potatoe routing*. Laboratory for Parallel Computing Research, Technion-Israel Institute of . . ., 1993.

[12] Jean-Claude Bermond, Pierre Fraigniaud, and Joseph G Peters. “Antepenultimate broadcasting”. In: *Networks* 26.3 (1995), pp. 125–137.

[13] Jean-Claude Bermond, Pavol Hell, Arthur L Liestman, and Joseph G Peters. “Broadcasting in bounded degree graphs”. In: *SIAM Journal on Discrete Mathematics* 5.1 (1992), pp. 10–24.

[14] Jean-Claude Bermond and Claudine Peyrat. “Broascasting in de Bruijn networks”. In: *Congressus numerantium* 66 (1988), pp. 283–292.

[15] Jean-Claude Bermond and Claudine Peyrat. “De Bruijn and Kautz networks: a competitor for the hypercube?” In: *First European Workshop on Hypercube and Distributed Computers*. Elsevier. 1989, pp–279.

[16] Dimitri P Bertsekas, C Özveren, George D Stamoulis, Paul Tseng, and John N. Tsitsiklis. “Optimal communication algorithms for hypercubes”. In: *Journal of Parallel and Distributed Computing* 11.4 (1991), pp. 263–275.

[17] Puspal Bhabak and Hovhannes A. Harutyunyan. “Constant Approximation for Broadcasting in k-cycle Graph”. In: *Conference on Algorithms and Discrete Applied Mathematics*. Springer. 2015, pp. 21–32.

[18] Laxmi N. Bhuyan and Dharma P. Agrawal. “Generalized hypercube and hyperbus structures for a computer network”. In: *IEEE Computer Architecture Letters* 33.04 (1984), pp. 323–333.

[19] Luc Bomans and Dirk Roose. “Communication benchmarks on the iPSC/2”. In: *Proceedings of the First European Workshop on Hypercube and Distributed Computers*. 1989, pp. 93–103.

[20] Jehoshua Bruck, Robert Cypher, and Ching-Tien Ho. *Multiple message broadcasting with generalized Fibonacci trees*. IEEE, 1992.

[21] Renato M Capocelli, Luisa Gargano, and Ugo Vaccaro. “Time bounds for broadcasting in bounded degree graphs”. In: *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer. 1989, pp. 19–33.

[22] Maja Čevnik and Janez Žerovnik. “Broadcasting on cactus graphs”. In: *Journal of Combinatorial Optimization* 33.1 (2017), pp. 292–316.

[23] T Champion and B Tourancheau. “Tnode: user document”. In: *Technical Report LIP-IMAG 89-02*. Ecole Normale Supérieure de Lyon France, 1989.

[24] Siu-Cheung Chau and Arthur L Liestman. “Constructing minimal broadcast networks”. In: *J. Combin. Inform. Syst. Sci* 10 (1985), pp. 110–122.

[25] X Chen. “An upper bound for the broadcast function B (n), Chinese J”. In: *Computers* 13 (1990), pp. 605–611.

[26] Edward Chow, Herbert Madan, John Peterson, Dirk Grunwald, and Daniel Reed. “Hyperswitch network for the hypercube computer”. In: *ACM SIGARCH Computer Architecture News* 16.2 (1988), pp. 90–99.

[27] Israel Cidon and Inder S Gopal. “Paris: An approach to integrated high-speed private networks”. In: *International Journal of Digital & Analog Cabled Systems* 1.2 (1988), pp. 77–85.

[28] D Clark, B Davie, D Farber, I Gopal, and B Kadaba. "D, Sincoskie, J. Smith, and D. Tennenhouse, The AURORA gigabit testbed". In: *Computer Networks and ISDN* (1991).

[29] EJ Cockayne and ST Hedetniemi. *A conjecture concerning broadcasting in m-dimensional grid graphs*. Tech. rep. Technical Report CS-TR-78-14, University of Oregon, 1978.

[30] Francesc Comellas, Hovhannes A. Harutyunyan, and Arthur L Liestman. "Messy broadcasting in multidimensional directed tori". In: *Journal of Interconnection Networks* 4.01 (2003), pp. 37–51.

[31] Kris Coolsaet, H De Meyer, and Veerle Fack. "Optimal algorithms for total exchange without buffering on the hypercube". In: *BIT Numerical Mathematics* 32.4 (1992), pp. 559–569.

[32] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. "LogP: Towards a realistic model of parallel computation". In: *Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming*. 1993, pp. 1–12.

[33] William J Dally. "Performance analysis of k-ary n-cube interconnection networks". In: *IEEE transactions on Computers* 39.06 (1990), pp. 775–785.

[34] William J Dally, Andrew Chien, Stuart Fiske, Waldemar Horwat, and John Keen. *The J-machine: A fine grain concurrent computer*. Tech. rep. Massachusetts inst of tech Cambridge microsystems research enter, 1989.

[35] Michael J Dinneen, Michael R Fellows, and Vance Faber. "Algebraic constructions of efficient broadcast networks". In: *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*. Springer. 1991, pp. 152–158.

[36] Michael J Dinneen, Jose A Ventura, Mark C Wilson, and Golbon Zakeri. "Compound constructions of broadcast networks". In: *Discrete applied mathematics* 93.2-3 (1999), pp. 205–232.

[37] S. Djelloul. "Etudes de certains rCseaux d'interconnexion: structures et communications". PhD thesis. Paris-Sud, Orsay, 1992.

[38] Michael Elkin and Guy Kortsarz. "Sublogarithmic approximation for telephone multicast: path out of jungle." In: *SODA*. Vol. 3. 2003, pp. 76–85.

[39] Michael Elkin and Guy Kortsarz. "A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem". In: *SIAM journal on Computing* 35.3 (2005), pp. 672–689.

[40] AM Farley and ST Hedetniemi. "Broadcasting in grid graphs". In: *Proc. 9th SE Conf. Combinatorics, Graph Theory, and Computing, Utilitas Mathematica*. 1978, pp. 275–288.

[41] Arthur Farley, Stephen Hedetniemi, Sandra Mitchell, and Andrzej Proskurowski. "Minimum broadcast graphs". In: *Discrete Mathematics* 25.2 (1979), pp. 189–193.

[42] Arthur M Farley and Andrzej Proskurowski. *Broadcasting: Between whispering and shouting*. University of Oregon. Department of Computer and Information Science, 1989.

[43] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. "Randomized broadcast in networks". In: *Random Structures & Algorithms* 1.4 (1990), pp. 447–460.

[44] Pierre Fraigniaud. "Complexity analysis of broadcasting in hypercubes with restricted communication capabilities". In: *Journal of Parallel and Distributed Computing* 16.1 (1992), pp. 15–26.

[45] Pierre Fraigniaud and Emmanuel Lazard. "Methods and problems of communication in usual networks". In: *Discrete Applied Mathematics* 53.1-3 (1994), pp. 79–133.

[46] Pierre Fraigniaud and Emmanuel Lazard. "Methods and problems of communication in usual networks". In: *Discrete Applied Mathematics* 53.1-3 (1994), pp. 79–133.

[47] Pierre Fraigniaud and Sandrine Vial. "Approximation algorithms for broadcasting and gossiping". In: *Journal of Parallel and Distributed Computing* 43.1 (1997), pp. 47–55.

[48] Pierre Fraigniaud and Sandrine Vial. "Comparison of heuristics for one-to-all and all-to-all communications in partial meshes". In: *Parallel Processing Letters* 9.01 (1999), pp. 9–20.

[49] Matthias Függer, Thomas Nowak, and Kyrill Winkler. "On the radius of nonsplit graphs and information dissemination in dynamic networks". In: *Discrete Applied Mathematics* 282 (2020), pp. 257–264.

[50] Luisa Gargano and Ugo Vaccaro. "On the construction of minimal broadcast networks". In: *Networks* 19.6 (1989), pp. 673–689.

[51] Jonathan L Gross, Jay Yellen, and Ping Zhang. *Handbook of graph theory*. CRC press, 2013.

[52] András Hajnal, Eric C Milner, and Endre Szemerédi. "A cure for the telephone disease". In: *Canadian Mathematical Bulletin* 15.3 (1972), pp. 447–450.

[53] Hovhannes A. Harutyunyan, George Laza, and Edward Maraachlian. "Broadcasting in necklace graphs". In: *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering*. 2009, pp. 253–256.

[54] Hovhannes A. Harutyunyan and Arthur L Liestman. "Messy broadcasting". In: *Parallel Processing Letters* 8.02 (1998), pp. 149–159.

[55] Hovhannes A. Harutyunyan and Arthur L Liestman. "More broadcast graphs". In: *Discrete Applied Mathematics* 98.1-2 (1999), pp. 81–102.

[56] Hovhannes A. Harutyunyan and Edward Maraachlian. "Linear algorithm for broadcasting in unicyclic graphs". In: *International Computing and Combinatorics Conference*. Springer. 2007, pp. 372–382.

[57] Hovhannes A. Harutyunyan and Edward Maraachlian. “On broadcasting in unicyclic graphs”. In: *Journal of combinatorial optimization* 16.3 (2008), pp. 307–322.

[58] Hovhannes A. Harutyunyan and Edward Maraachlian. “Broadcasting in fully connected trees”. In: *2009 15th International Conference on Parallel and Distributed Systems*. IEEE. 2009, pp. 740–745.

[59] Hovhannes A. Harutyunyan and Edward Maraachlian. “Linear Algorithm for Broadcasting in Networks With No Intersecting Cycles.” In: *PDPTA*. 2009, pp. 296–301.

[60] Hovhannes A. Harutyunyan and Edward Maraachlian. “Broadcasting in hierarchical tree cluster network”. In: *PDPTA 2010: proceedings of the 2010 international conference on parallel and distributed processing techniques and applications (Las Vegas NV, July 12-15, 2010)*. 2010, pp. 479–484.

[61] Hovhannes A. Harutyunyan and Bin Shao. “An efficient heuristic for broadcasting in networks”. In: *Journal of Parallel and Distributed Computing* 66.1 (2006), pp. 68–76.

[62] Sandra M Hedetniemi, Stephen T Hedetniemi, and Arthur L Liestman. “A survey of gossiping and broadcasting in communication networks”. In: *Networks* 18.4 (1988), pp. 319–349.

[63] Marie-Claude Heydemann, Jaroslav Opatrny, and Dominique Sotteau. “Broadcasting and spanning trees in de Bruijn and Kautz networks”. In: *Discrete applied mathematics* 37 (1992), pp. 297–317.

[64] Juraj Hromkovič, Claus-Dieter Jeschke, and Burkhard Monien. “Optimal algorithms for dissemination of information in some interconnection networks”. In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 1990, pp. 337–346.

[65] Juraj Hromkovič, Ralf Klasing, Burkhard Monien, and Regine Peine. “Dissemination of information in interconnection networks (broadcasting & gossiping)”. In: *Combinatorial network theory*. Springer, 1996, pp. 125–212.

[66] Cray Inc. *The Gemini Network*. URL: https://wiki.alcf.anl.gov/parts/images/2/2c/Gemini-whitepaper.pdf.

[67] Andreas Jakoby, Rüdiger Reischuk, and Christian Schindelhauer. “The complexity of broadcasting in planar and decomposable graphs”. In: *Discrete Applied Mathematics* 83.1-3 (1998), pp. 179–206.

[68] Klaus Jansen and Haiko Müller. “The minimum broadcast time problem for several processor networks”. In: *Theoretical Computer Science* 147.1-2 (1995), pp. 69–85.

[69] LH Khachatrian and OS Harutounian. “Construction of new classes of minimal broadcast networks”. In: *Conference on Coding Theory, Armenia*. 1990, pp. 69–77.

[70] Ralf Klasing, Burkhard Monien, Regine Peine, and Elena A Stöhr. “Broadcasting in butterfly and DeBruijn networks”. In: *Discrete Applied Mathematics* 53.1-3 (1994), pp. 183–197.

[71] C Ko. “Broadcasting, Graph homomorphisms and chord intersections.” In: (1980).

[72] Chen-Shung Ko. “On a conjecture concerning broadcasting in grid graphs”. In: *AMS Notices* (1979), A196–A197.

[73] Guy Kortsarz and David Peleg. “Approximation algorithms for minimum time broadcast”. In: *Israel Symposium on Theory of Computing and Systems*. Springer. 1992, pp. 67–78.

[74] S Kuppuswami and B Tourancheau. “Evaluating the performances of transputer based hypercube vector computer”. In: *La lettre du Transputer* 4 (1990).

[75] Roger Labahn. “Information flows on hypergraphs”. In: *Discrete mathematics* 113.1-3 (1993), pp. 71–97.

[76] Roger Labahn. “A minimum broadcast graph on 63 vertices”. In: *Discrete Applied Mathematics* 53.1-3 (1994), pp. 247–250.

[77] Tomas Lang and Harold S Stone. “A shuffle-exchange network with simplified control”. In: *IEEE Transactions on computers* 100.1 (1976), pp. 55–65.

[78] Frank Thomson Leighton. *Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI.* Tech. rep. Massachusetts inst of tech Cambridge lab for computer science, 1982.

[79] Arthur L Liestman and Joseph G Peters. “Broadcast networks of bounded degree”. In: *SIAM journal on Discrete Mathematics* 1.4 (1988), pp. 531–540.

[80] Arthur L Liestman and Joseph G Peters. “Minimum broadcast digraphs”. In: *Discrete Applied Mathematics* 37 (1992), pp. 401–419.

[81] Arthur L. Liestman and Dana Richards. “Network communication in edge-colored graphs: gossiping”. In: *IEEE Transactions on Parallel and Distributed Systems* 4.4 (1993), pp. 438–445.

[82] Philip D MacKenzie. “A lower bound for order-preserving broadcast in the postal model”. In: *Parallel Processing Letters* 3.04 (1993), pp. 313–320.

[83] David May. “The next generation transputers and beyond”. In: *European Conference on Distributed Memory Computing*. Springer. 1991, pp. 7–22.

[84] DS Meliksetian and CYR Chen. *Optimal routing algorithm and other communication issues of the cube-connected cycles*. Tech. rep. Tech. Rep. TR90-4, Dept. Elec. & Comp. Eng., Syracuse University, 1990.

[85] Martin Middendorf. “Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2”. In: *Information Processing Letters* 46.6 (1993), pp. 281–287.

[86] Sandra Mitchell and Stephen Hedetniemi. “A census of minimum broadcast graphs”. In: *J. Combin. Inform. System Sci* 5 (1980), pp. 141–151.

[87] Mohamed Ould-Khaoua. “On the optimal network for multicomputer: Torus or hypercube?” In: *European Conference on Parallel Processing*. Springer. 1998, pp. 989–992.

[88] Brigitte Plateau and Denis Trystam. "Optimal total exchange for a 3-D torus of processors". In: *Information Processing Letters* 42.2 (1992), pp. 95–102.

[89] Franco P Preparata and Jean Vuillemin. "The cube-connected cycles: a versatile network for parallel computation". In: *Communications of the ACM* 24.5 (1981), pp. 300–309.

[90] Prabhakar Raghavan and Eli Upfal. "Efficient routing in all-optical networks". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 134–143.

[91] Justin Rattner. "The new age of supercomputing". In: *European Conference on Distributed Memory Computing*. Springer. 1991, pp. 1–6.

[92] R Ravi. "Rapid rumor ramification: Approximating the minimum broadcast time". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE. 1994, pp. 202–213.

[93] R Reischuk. "An Algebraic Divide and Conquer Approach to Design Highly Parallel Solution Strategies for Optimization Problems on Graphs". In: *Technical Report*. TH Darmstadt, 1991.

[94] Youcef Saad and Martin H Schultz. "Topological properties of hypercubes". In: *IEEE Transactions on computers* 37.7 (1988), pp. 867–872.

[95] Peter Scheuermann and Geoffrey Wu. "Heuristic algorithms for broadcasting in point-to-point computer networks". In: *IEEE Computer Architecture Letters* 33.09 (1984), pp. 804–811.

[96] Christian Schindelhauer. "On the inapproximability of broadcasting time". In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2000, pp. 226–237.

[97] Steve R Seidel. "Circuit switched vs. store-and-forward solutions to symmetric communication problems". In: *Proceedings of the 4th Conference on Hypercube Computers and Concurrent Applications*. 1989, pp. 253–255.

[98] Steven R Seidel, Ming-Homg Lee, and Shivi Fotedar. "Concurrent bidirectional communication on the Intel iPSC/860 and iPSC/2". In: *The Sixth Distributed Memory Computing Conference, 1991. Proceedings*. IEEE Computer Society. 1991, pp. 283–284.

[99] Charles L Seitz. "The cosmic cube". In: *Communications of the ACM* 28.1 (1985), pp. 22–33.

[100] Peter J. Slater, Ernest J. Cockayne, and Stephen T. Hedetniemi. "Information dissemination in trees". In: *SIAM Journal on Computing* 10.4 (1981), pp. 692–701.

[101] Elena Stöhr. "Broadcasting in the butterfly network". In: *Information Processing Letters* 39.1 (1991), pp. 41–43.

[102] Elena Stöhr. “On the broadcast time of the butterfly network”. In: *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer. 1991, pp. 226–229.

[103] Abderezak Touzene and Brigitte Plateau. “Optimal multinode broadcast on a mesh connected graph with reduced bufferization”. In: *European Conference on Distributed Memory Computing*. Springer. 1991, pp. 143–152.

[104] Jonathan S Turner and Leonard F Wyatt. “A packet network architecture for integrated services”. In: *Proceedings of GLOBECOM*. Vol. 83. 1983, pp. 11–83.

[105] Frances Van Scoy. “Broadcasting a small number of messages in a square grid graph”. In: *Proc. Seventeenth Allerton Conf. on Communication, Control and Computing*. 1979.

[106] Emmanouel A Varvarigos and Dimitri P Bertsekas. “Communication algorithms for isotropic tasks in hypercubes and wraparound meshes”. In: *Parallel Computing* 18.11 (1992), pp. 1233–1257.

[107] Wikipedia. *Cartesian product of graphs*. URL: https://en.wikipedia.org/wiki/Cartesian_product_of_graphs.

[108] Wikipedia. *Torus interconnect*. URL: https://en.wikipedia.org/wiki/Torus_interconnect#cite_note-1.

[109] Ouri Wolfson and Adrian Segall. “The communication complexity of atomic commitment and of gossiping”. In: *SIAM Journal on Computing* 20.3 (1991), pp. 423–450.

[110] Jian-guo Zhou and Ke-min Zhang. “A minimum broadcast graph on 26 vertices”. In: *Applied mathematics letters* 14.8 (2001), pp. 1023–1026.