CHAIR OF ELECTRONIC DESIGN AUTOMATION

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Development of Signal Routing Algorithms for Multicast in Wavelength-Routed Optical Networks-on-Chips

Rei Haskaj

CHAIR OF ELECTRONIC DESIGN AUTOMATION

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Development of Signal Routing Algorithms for Multicast in Wavelength-Routed Optical Networks-on-Chips

Author:	Rei Haskaj
Supervisor:	Dr.-Ing. Tsun-Ming Tseng
Submission Date:	30.05.2025

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 30.05.2025 Rei Haskaj

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr.-Ing. Tsun-Ming Tseng, for his unwavering support, guidance, and availability throughout this work. His insightful feedback, patience, and readiness to answer all my questions have been invaluable to both the development of this research and my academic growth. I am truly thankful for his encouragement and expertise. I am profoundly grateful to my family — to my mother Erjena, my father Kastriot, and my brother Ted — whose unwavering love, sacrifices, and encouragement have been the foundation of all my efforts. Their support has given me strength in moments of difficulty and joy in moments of progress. I also extend heartfelt thanks to my close friends, whose presence and understanding have meant more than words can express during this journey.

Abstract

On-chip communication has become a topic of significant interest in recent years. Traditional electrical Networks-on-Chip (NoCs) may no longer be the optimal solution, as emerging research increasingly favors optical Networks-on-Chip (ONoCs) due to their better latency and bandwidth, which contribute to overall better performance. Optical NoCs can be broadly categorized into two types: active ONoCs and passive ONoCs. The latter are commonly known as wavelength-routed optical networks-on-chip (WRONoCs). In these networks, communication between a sender and a receiver typically involves the use of a specific wavelength. This work focuses on implementing multicast communication within WRONoCs, where a single sender transmits a message to multiple receivers using the same wavelength. Various network topologies can support this, including ring and mesh configurations. This thesis specifically examines mesh networks, exploring two distinct multicast strategies: the Bordered Areas Approach and the Layering Model. Each approach presents a unique method for enabling multicast in mesh networks, with potential applicability to ring networks as well.

Contents

1 Introduction

1.1 Optical Networks-on-Chip

Optical networks-on-chip (ONoC) tend to perform better at most tasks compared to traditional electrical networks-on-chip, offering a higher bandwidth and lower latency. The optical networks-on-chip are divided into two groups: the active ONoCs and the passive ONoCs. This categorization is done based on the routing mechanism [Li+18]. For an active ONoC, the paths that signals follow are not predefined in the design stage. A control system [XTS21] controls these paths while making real-time decisions. Passive ONoCs, on the other hand, function differently. The control system is not implemented in such cases because all routing options are predetermined in the design stage. That is also the main reason why the passive ONoCs are referred to as wavelength-routed optical networks-on-chip (WRONoCs). The absence of the control system in these NoCs results in a lower dynamic power consumption and network delay [Tru+20]. However, passive networks are not always more power-efficient than active networks. This can only hold for networks consisting of 16 nodes at most [Tru+20], which could be a ring network or a 4×4 mesh network, with the latter being the main target of this thesis.

1.2 Power Distribution Network and Splitters

The power distribution network and splitters are two important concepts when it comes to on-chip communication. The power distribution network (PDN) is tasked with powering the nodes in order to transmit optical signals. This principle is similar to a clock distribution network, which distributes the clock signal to the D Flip-Flops of a digital design, so that all registers are clocked with the same clock signal. There are different methods that can be applied to configure such a PDN, with one of them being the optimized version explained in [Ort+17]. Being able to split an optical signal is also vital for this thesis. In some cases, the Power Distribution Network (PDN) may provide a sending node with sufficient optical power to transmit a message using wavelength λ. If this node is equipped with two internal optical senders, and splitters are applied at the output, the node can transmit two optical messages simultaneously. Each sender within the node is capable of transmitting an optical signal at wavelength λ, allowing

the sending node to perform parallel communication with the same wavelength. The power loss of this splitting operation is in the vicinity of 3 dB [Zha+19]. Similarly to the PDN, different methods can be applied to construct such splitters, with one of them being the Y-branch shown in [Zha+19]

1.3 Micro-Ring-Resonators

Micro-ring-resonators (MRRs) are designed for redirecting optical signals, thus facilitating on-chip communication. MRRs are implemented to correspond to a specific wavelength. If the wavelength of the optical signal being transmitted matches that of the MRR, then a coupling takes place. This coupling operation redirects the optical signal either horizontally or vertically. For instance, if a signal enters the MRR from the western direction, a vertical coupling could take place, which could redirect the signal to the northern direction. More examples can also be found in the work from [Zhe+21].

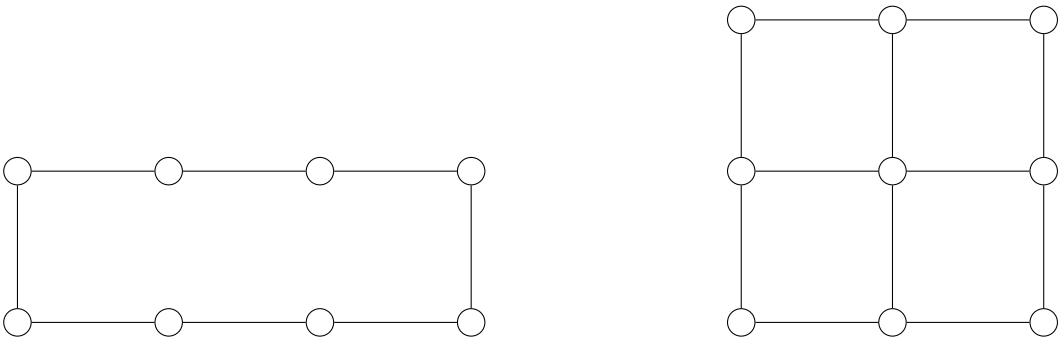
1.4 Multicast and Broadcast

The main goal of this thesis is the application of multicast to minimize the wavelength usage in a WRONoC. The traditional method of transmitting a message is using a single sender to send a signal, which is received by the receiving node. This method takes into consideration that each node of the network consists of a single sender and a single receiver as well. However, there are also multiple networks, where the nodes might consist of multiple senders and multiple receivers. This is where multicast and broadcast are more attractive alternatives for transmitting messages. Multicast takes into consideration nodes that possess multiple senders and can send one message to different receivers on different nodes. Broadcast, on the other hand, can be considered as an all-out communication method, where the nodes have multiple senders as well as multiple receivers. This will, however, not be part of this work.

1.5 Ring Networks and Mesh/Grid Networks

The two main groups that will be taken into consideration in this thesis are the ring networks and the mesh/grid networks. In a ring network, each communication node has a total of two neighbouring nodes, while in a mesh/grid network, the possibilities of communication are broader. For the 4×4 mesh network that will be examined in this paper, for instance, the communication nodes have a minimum of two and a maximum of four neighbouring nodes, which holds for a 3x3 mesh network as well, actually.

While a ring topology offers lower power consumption and lower wiring complexity, it lacks in connectivity. Meanwhile, the mesh topologies offer a higher connectivity as well as higher throughput, but unfortunately require larger areas to be implemented [Her+]. In Figure 1.1, there is a representative network for both cases. An important observation on the 3x3 mesh network is the different number of neighbouring nodes with respect to the most central node, which is directly connected to four other nodes. These other four nodes are themselves connected to three neighbours, with one of these neighbours being the central node, while the other two are positioned in the corners of the network. Lastly, the corner nodes are connected to only two neighbours.



(a) Ring (b) 3x3 Mesh

Figure 1.1: An example of a ring network alongside a 3x3 mesh/grid network.

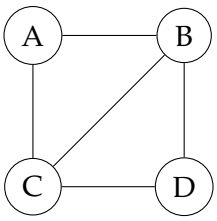
Apart from these two, there also exist other possibilities of configuring a network, such as a star network, where all nodes are connected to a central routing hub or switch, as can be seen in [FW17]. These, however, are not very common in constructing NoCs due to the fact that wiring can become a major issue in networks that contain a lot of nodes. Also, the central hub can become a bottleneck in these types of structures.

2 Background

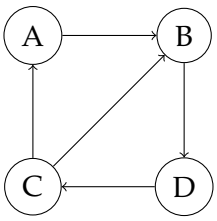
Before presenting the proposed models of applying multicast for a given mesh network and then adapting these for a ring network, some information regarding graph theory and mixed integer linear programming (MILP) is required. The networks, which are taken into consideration for this thesis, are all modeled as graphs with specific characteristics. Hence, it is important to discuss the general concepts of graph theory. Furthermore, mixed integer linear programming is used as a means of modelling the graph problems that will be proposed later on Ch. 3. A lot of problems encountered in this thesis can be compared to known problems for which an MILP model already exists. An MILP solver can be used to obtain a solution to these problems.

2.1 Graph Theory

Graphs are some of the most common data structures in multiple algorithms. A non-empty set of vertices and edges is considered to be a graph [Wal02]. One can form a subgraph of any given graph, with the condition that the sets of vertices and edges of the newly formed subgraph are subsets of the sets of vertices and edges of the original graph. There are generally two types of edges that can be found in different graphs: directed edges and non-directed edges. A directed edge between two vertices, A and B, indicates that one can only traverse in one direction, either from A to B or from B to A. A non-directed edge, on the other hand, indicates that both directions can be used, meaning that one could traverse from vertex A to vertex B and also from vertex B to vertex A. If a graph consists of non-directed edges, then it is considered to be a non-directed graph, while if it consists of directed edges, then it is considered to be a directed graph. An example of a directed and non-directed graph can be seen in Figure 2.1.



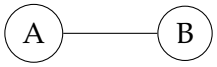
(a) Non-directed Graph



(b) Directed Graph

Figure 2.1: An example of a non-directed alongside a directed graph.

A non-directed edge can also be presented as a multi-edge, which would be composed of the directed edges A-B and B-A, as can be seen in Figure 2.2.



(a) Non-directed edge A-B

(b) Multi-edge between vertices A and B

Figure 2.2: Decomposition of a non-directed edge to a multi-edge.

2.1.1 Paths and Cycles

If a vertex is connected to an edge, then that vertex is considered to be incident to that particular edge. Similarly, two edges are considered to be incident to one another if they both share a vertex. For instance, if a graph consists of three vertices like a, b, and c and the edges (a,b) and (b,c), then these edges are considered to be incident to one another, because they both share the vertex b. Paths are also very important concepts when graphs are being discussed. A sequence of incident edges is considered to be a path. It needs to be noted that a path is allowed to visit the same vertex of a graph multiple times, but the same edge cannot be visited twice. Taking Figure 2.1a into consideration, a valid path would, for instance, be one path starting from vertex A and ending at vertex B: $(A, B) - -(B, C) - -(C, D) - -(D, B)$. As the example shows, vertex B has indeed been visited two times: once from node A at the beginning and once from node D at the very end. As mentioned, this is allowed. The most important detail for a path is that an edge is not passed by multiple times, which is not the case for the given example. If a path exists between two vertices of a graph, then these vertices are considered to be connected to one another. Furthermore, if every pair of vertices of a graph is connected, then the whole graph is called a connected graph. The graph from Figure 2.1a is clearly a connected graph. One could start a path from any vertex of the graph and visit every other vertex. Similarly, the graph from Figure 2.1b is also a connected graph, but one must be cautious in such cases. There clearly is

no direct edge starting from vertex A and ending at vertex C, for example, but there exists a path that can connect these two vertices to one another. That path would be $(A,B)->(B,D)->(D,C)$. This path actually proves that A can reach all other vertices of this graph, since vertices B and D were also visited along this path. Similarly, one could start at B and go through the path $(B,D)->(D,C)->(C,A)$, thus visiting all the vertices of the graph. The remaining two paths starting from the vertices D and C would be respectively $(D,C)->(C,A)->(A,B)$ and $(C,A)->(A,B)->(B,D)$. Starting from C, one could also follow the path that starts with the directed edge (C, B), but this path would never reach node A. This, however, does not imply that C and A cannot be connected. At least one path needs to exist that connects the two vertices, which was already presented above.

If a path starts and ends at the same vertex, then it is called a cycle. The graph from Figure 2.1a for example, contains cycles, like $(A,B)--(B,C)--(C,A)$, $(B,C)--(C,D)--(D,B)$ and $(A,B)--(B,D)--(D,C)--(C,A)$. On the other hand, even the graph from Figure 2.1b contains cycles, such as $(C,B)->(B,D)->(D,C)$ and $(A,B)->(B,D)->(D,C)->(C,A)$. However, there is no cycle that can be found in the subgraph containing the vertices A, B, C, and the edges $(A,B),(C,A),(C,B)$ due to the directions of the edges, which do not allow the formation of such a path starting and ending at the same vertex between the vertices of this group. Not every graph contains a cycle. In those cases, the graph is referred to as acyclic. Furthermore, an edge that starts and ends at the same node is not considered to be a cycle, but rather a loop. A graph that contains either loops or multiple edges is considered not to be a simple graph. Both graphs from Figure 2.1 do not contain loops or multiple edges, so they both are considered to be simple graphs.

2.1.2 Degree

When dealing with a graph, one encounters two types of degrees: the degree of a vertex and the degree of the graph. Considering that the graph is a non-directed graph, the degree of a vertex corresponds to the number of incident edges of this vertex. Different vertices have therefore different degrees, but at least one of the vertices has the highest degree in the graph. That particular value is considered to be the degree of the whole graph. In other words, the degree of the graph corresponds to the degree of the vertex that has the highest number of incident edges connected to it. In Figure 2.1a, the degree of the vertices are:

1. Vertex A : deg(A) = 2
2. Vertex B : deg(B) = 3

3. Vertex C : deg(C) = 3
4. Vertex D : deg(D) = 2

The highest degree value can be found in vertices B and C, and it is equal to 3. This means that the degree of this graph is also equal to 3.

If the graph were a directed graph, then the concepts of incoming and outgoing degrees would be introduced. For each vertex, one could calculate the number of incoming edges, and the total would be considered as the incoming degree of this vertex. Similarly, one could count the number of outgoing edges with respect to a vertex and find the outgoing degree of the particular vertex. As an example, the degrees of the vertices from the graph in Figure 2.1b can be shown in the following table:

Table 2.1: Incoming and outgoing degrees of the vertices from the graph in Figure 2.1b.

Vertex	Incoming Degree	Outgoing Degree
A	1	1
B	2	1
C	1	2
D	1	1

2.1.3 Other Attributes of Vertices and Edges

Apart from the name and degree, which are two of the most important attributes for a vertex, these could also contain other important information, like the geometrical position of a vertex in a plane. This position could especially be useful for particular graph algorithms like the A* algorithm (see Sec. 2.3.2), where heuristics are considered to find the shortest path from one starting vertex to a target vertex. Depending on the graph algorithm that is going to be applied, the edges of the graph could also have specific attributes such as weight. Dijkstra's algorithm [Dij59], for example, needs to take into consideration the weights of a graph in order to find the shortest path from a starting vertex to a target vertex.

2.1.4 Graph Algorithms

After introducing graphs as a common data structure, some typical algorithms need to be studied as well. The most usual task in a graph is to search for a vertex, starting at any graph vertex. Two of the most applied algorithms that would complete such a task are the depth-first search algorithm (DFS) and the breadth-first search algorithm (BFS),

which take two different approaches to solve the same problem [MMD17] [TL19b]. The preferred data structure that is applied for both algorithms also differs from one another. While the BFS uses a queue with the FIFO principle (First-In-First-Out), the DFS makes use of stacks with the LIFO principle (Last-In-First-Out). The time complexity for both of these algorithms is linear ($O(n)$). In the initial state, both these data structures are empty. The steps of the BFS algorithm would be:

1. Insert the starting vertex into the queue.
2. Check if this is the vertex being searched. If not, then insert all its neighbours into the queue and pop this vertex from the queue.
3. Check if the first neighbour is the vertex being searched. If not, then insert all its neighbours into the queue and pop this one from the queue.
4. Repeat until the vertex being checked is the target vertex or until the queue is empty.

If the queue is empty at the end of the algorithm and the target vertex is still not found, then it means that such a vertex is not part of this particular graph. This will also be the case for the DFS algorithm when the stack is empty, but the target vertex was not found. The steps of the DFS algorithm would be:

1. Push the starting vertex into the stack.
2. Pop the starting vertex from the stack to check it. If it is not the target vertex, push its neighbours into the stack.
3. Pop the last neighbour pushed into the stack from the previous step, and check if it is the target vertex. If not, insert its neighbours.
4. Repeat until the target vertex is found or until the stack is empty.

2.2 Mixed Integer Linear Programming

In order to fully comprehend the idea of mixed integer linear programming, one should first consider the steps that are normally followed in order to solve a given problem. The first step in finding a solution to a problem would be to model it, preferably as a graph. After the modelling part, an algorithm is applied to this graph, which would then lead to a solution. A simple example of this problem-solving flow in real life can be found in the concept of a GPS device. In this case, the problem formulation would be: Find the shortest path from the user's current location to their desired destination. This problem is then modelled as a graph, where the current location and the desired destination point are saved as vertices of a graph representing the city. All streets would, in this case, be represented as edges of this graph. The duty of the algorithm is to find the shortest way to the destination, while considering all possibilities along the way, like turns, traffic lights, heavy traffic, or even ongoing constructions on specific roads. The main issue of this flow is that modelling a problem as a graph is, unfortunately, not always possible. That is exactly the reason why mixed integer linear programming (MILP) is considered to be a safer approach to solving any given problem. MILP (MIP as an abbreviation) is a mathematical modelling method, meaning that it is a model type as well as a method. MIP models a problem as a set of linear equations and inequalities with different types of variables. This modelling method can be applied to every type of problem, unlike graph modelling. Instead of using an algorithm to find a solution, the MIP model is sent to an MIP solver, which then is able to find a solution to the problem. A MIP solver performs linear algebra to find the values of the variables that satisfy the equations and inequalities constructed in the earlier stage. An optional step would be to also send an optimization objective to the MILP solver, which typically refers to solving a maximization or minimization problem. The types of variables that one can encounter in an MIP model are binary variables, integer variables, and continuous variables. Binary variables take one value that is either equal to 0 or 1, while integer variables take an integer value from the set of integers Z, depending on the specified constraints. All the linear equations and inequations, which are part of an MIP model, are connected to one another by means of the logical AND ($\wedge$) operator. MIP is applicable for most large-scale engineering problems, but the main disadvantage of this method is that it can be quite slow.

2.2.1 MIP Model

An example of a real-life problem that cannot be modelled as a graph, but is easy to comprehend with an MILP model, would be designing a personal diet. Based on specific factors like gender, age, and activity level, people have different needs regarding

calorie intake per day. A young male who is trying to become a professional athlete, for example, would have a different diet compared to an older male who works in an office and works out less. In order to design the diet for the young athlete, one must try to maximize the intake of vitamins, fiber, and natural oils while also minimizing the intake of sugar and carbohydrates. With that being said, if the athlete consumes a specific type of food that is rich in natural oils, such as avocados, he would need to be careful not to consume a lot of seeds on the same day, because the seeds also contain a considerable amount of natural oils.

2.2.2 Propositional Logic

The last sentence from the previous subsection needs to be carefully examined before constructing an equation or inequation for the MIP model. This sentence presents two possible scenarios:

1. Avocados are consumed.
2. Seeds are not consumed.

In the context of discrete mathematics, these two are referred to as propositions. Propositions are statements to which a truth value can be assigned [MF24]. The two possibilities are true (t/T/1) or false (f/F/0). Normally, such a value cannot be assigned to every phrase or sentence, like "Good morning!". In the example above, the first statement should be assigned the value true, while the second one should be assigned the value false.

Apart from the propositions themselves, it is also very important to notice the way that these two statements are combined. In this case, if statement A takes place, then statement B should take place as well, or in less terms: If A, then B, with A being "Avocados are consumed" and B being "Seeds are not consumed". In propositional logic, such a connection of statements is referred to as an implication ($\Rightarrow$) [Gal11]. In discrete mathematics, however, an implication is simply a subjunction ($\rightarrow$) that always holds [Sch19]. Thus, studying the subjunction would be recommended for this part. In mathematical terms, a subjunction would be represented by the following equation:

$$A(A, B) \iff A \rightarrow B \tag{2.1}$$

Accordingly, the truth table of such a propositional form would be:

Table 2.2: Truth table of the propositional form in Equation 2.1

A	B	$A \rightarrow B$
t	t	t
t	f	f
f	t	t
f	f	t

In discrete mathematics, the most preferred presentation of propositional forms is by means of a conjunctive normal form (CNF) or disjunctive normal form (DNF). A CNF is a form that contains disjunctive terms that are connected through conjunctions, while in analogy, a DNF is a form that contains conjunctive terms that are connected through disjunctions [Tur+02]. A concrete transformation of the form presented in Equation 2.1 would be [Gal11]:

$$A(A, B) \iff \neg A \vee B \tag{2.2}$$

It is very simple to prove that Equation 2.2 is equivalent to Equation 2.1, because both forms share the same truth table, which is the one shown in Table 2.2. The logical value of Equation 2.2 can only be equivalent to false if and only if the proposition A is true and the proposition B is false, which is exactly the case for Equation 2.1 as well.

2.2.3 MIP Constarints

A MIP constraint is either an equation or an inequality of the MIP model. As mentioned earlier, all MIP constraints are logically connected to one another by the logical AND ($\wedge$) operator, which is the reason why it is recommended to avoid connecting constraints to one another by other connectors like the logical OR ($\vee$) connector. For the example of the young athlete, one can construct a constraint regarding the consumption of avocados and seeds at this point. The first step is to declare the types of variables that will be used. In this case, it would be easier to apply binary variables because each statement can only be assigned one value, either true or false. The first binary variable, which will be presented here, is b_{avocado}. This binary variable should represent the statement: Avocado is consumed. If the athlete indeed consumes an avocado, then the value of b_{avocado} would be equal to 1. Otherwise, if he does not consume any avocados, then the value of the binary variable would be equal to 0. Similarly, one could declare the binary variable b_{seeds}. If seeds are consumed by the athlete, then $b_{\text{seeds}} = 1$, otherwise $b_{\text{seeds}} = 0$. The whole constraint for the athlete is that if avocados are consumed, seeds should not be consumed. The discrete mathematical expression

for this whole statement would be $A \Rightarrow \neg B$, which, as mentioned, is equivalent to $\neg A \vee \neg B$. Normally, MIP uses inequalities for most of the calculations, so even the equalities are decomposed into two inequalities. So since avocado is consumed, then $b_{\text{avocado}} = 1$. This equation can be split into two inequalities:

1. $b_{\text{avocado}} \leq 1$
2. $b_{\text{avocado}} \geq 1$

An important fact to mention before moving forward is that both these new inequalities are connected to one another by the logical AND ($\wedge$) operator. Logically, these two inequalities combined with the logical AND ($\wedge$) operator represent $b_{\text{avocado}} = 1$, since this is the only case when both inequalities are satisfied. However, it is known at this point that b_{avocado} is a binary variable, which can either take the value 1 or 0. In other words, the range of this variable is fixed by definition. As a consequence, setting $b_{\text{avocado}} \geq 1$ is logically equivalent to $b_{\text{avocado}} = 1$. That means that the other constraint is redundant, and thus it can be simply deleted.

If the proposition/statement A is $b_{\text{avocado}} \geq 1$, then $\neg A$ is $b_{\text{avocado}} < 1$. Normally, a MIP solver works better with the $\leq$ and $\geq$ signs, thus it is recommended to transform the $b_{\text{avocado}} < 1$ inequality to $b_{\text{avocado}} \leq 1 - 1$ or simply $b_{\text{avocado}} \leq 0$. Since the propositions are connected by the logical OR ($\vee$) operator, the $\neg B$ should be positioned on the right-hand side of the inequality $b_{\text{avocado}} \leq 0$. In order to complete the constraint, $negB$ needs to be analysed as well.

Table 2.3: Linear expression for $\neg B$

b_{B}	1	0
$\neg b_{\text{B}}$	0	1
$1 - b_{\text{B}}$	0	1

Instead of $\neg b_{\text{B}}$, one must use $1 - b_{\text{B}}$. The first one is a logical expression, while the second one is a linear expression that can be understood by the MIP solver. Finally, the complete MIP constraint would be:

\\If avocados are consumed, then seeds should not be consumed.

$$b_{\text{avocado}} \leq 0 + 1 - b_{\text{seeds}} \tag{2.3}$$

$$b_{\text{avocado}} \leq 1 - b_{\text{seeds}} \tag{2.4}$$

As can be analysed in this inequality, for both cases, the conditions are fulfilled:

1. Avocado is consumed: $b_{\text{avocado}} = 1$ can only happen if seeds are not consumed: $b_{\text{seeds}} = 0$.
2. Avocado is not consumed: $b_{\text{avocado}} = 0$ means that the seeds may or may not be consumed: $b_{\text{seeds}} \leq 1$.

2.2.4 MIP Optimization

As mentioned earlier, it is also possible to set an optimization objective for the MIP solver, which typically means solving either a minimization or maximization problem in mathematical terms. However, this optimization is always optional. The solver would deliver a solution even without an optimization target, but that solution is highly likely not to be the best one. For the case of the athlete, an optimization objective could be to minimize the intake of industrial sugar on a daily basis, because this type of sugar could hinder his physique and overall performance. In order to set this optimization objective, the required variables need to be defined. Similarly to the example taken for the MIP constraints, binary variables will once again be used for this task. Let S be the set of all food articles that have a high amount of industrial sugar, like candies or fizzy drinks, then $\forall s \in S, b_{\text{s}}$ binary variables need to be declared. $b_{\text{s}} = 1$ indicates that the athlete has consumed the product s. Otherwise, if $b_{\text{s}} = 0$, the athlete has not consumed product s. The optimization objective would then be the following:

\\Minimize the daily intake of industrial sugar.

$$min \sum_{s \in S} b_{\text{s}} \tag{2.5}$$

The keyword "min" signals the MIP solver to perform a minimization problem. In other cases, one may want to perform a maximization. Thus, the keyword "max" would be used. In the current example, the athlete would opt to maximize the protein intake from his daily diet. Let F be the set of all food products that the athlete has purchased from the store, which include fruits, vegetables, fish, etc.. Then $\forall f \in F, b_{\text{f}}$ needs to be declared. $b_{\text{f}} = 1$ indicates that the athlete has consumed the product f. Otherwise, if $b_{\text{f}} = 0$, the athlete has not consumed product f. In this case, the optimization objective would be the following:

\\Maximize the daily intake of protein.

$$max \sum_{f \in F} b_{\text{f}} \tag{2.6}$$

2.3 Mixed Integer Linear Programming and Graphs

As discussed earlier, mixed integer linear programming is a mathematical modeling method that can be applied to solve any general problem, even if this particular problem cannot be modeled as a graph. However, that does not mean that MILP and graphs cannot be used in tandem to solve a particular problem. For a majority of engineering problems for which the problem can be modelled as a graph, it needs to be determined beforehand what the vertices and edges represent, since a graph is a set of vertices and edges. On specific occasions, the edges of the graph may represent physical connections between the vertices, like the wires, that may connect different modules with one another, which themselves would be the vertices of this graph. However, there are specific cases where the edges of a graph may not represent a real physical connection between the vertices, but rather a logical connection between the vertices. An example of this case would be a graph, whose vertices represent the lectures that a student desires to attend at the university. When the student is arranging their personal timetable, they need to be careful in choosing lectures that take place on the same day and at the same time. This way, the student can avoid collisions between lectures. The edges in this graph represent the logical connection between these lectures. Some of the typical graph problems that will be referred to later on in this thesis (more specifically in Ch. 3) are:

1. The vertex coloring problem.
2. One-source-one-target problem.
    a) Dijkstra's algorithm and A* algorithm.
3. Planar routing problem.

The first problem of this group considers the edges of the graph as logical connections between the vertices, while the second and third problems consider the edges as actual physical connections between the vertices.

2.3.1 Vertex Coloring Problem

The vertex coloring problem is a typical example of a problem where the edges are treated as logical connections between vertices. In order to fully understand the concepts behind this type of assignment, an example of an engineering problem shall be examined. This would be the layer assignment problem. In semiconductor manufacturing, masks need to be prepared. There are specific patterns that need to be printed on these masks. In the manufacturing process, a UV light is applied that

should be able to pass through these patterns. A functional pattern can be formed for the parts on the chip that are exposed to UV light. The issue in this example is that the patterns of the mask may be located near one another, which corresponds to the gaps between the patterns being quite small as well. This may cause a problem with the passage of the UV light. This light should pass through the gaps and normally move in a straight manner. These small gaps, however, may cause irregular movements from the UV light, which is not desired. This may cause the two different patterns of the mask to be treated as a single pattern, which may lead to a merging between the patterns after the manufacturing process is complete. This scenario should be avoided. In order to achieve a correct result from this manufacturing process, the patterns should be assigned to different layers, hence the name "layer assignment problem". In the sense of a printer, the solution to this problem would be to print these patterns in different layers, where in each layer, there are only patterns that do not conflict with one another. The conflict would be the spacing between specific patterns. Having assigned specific patterns to the first printed layer, one could repeat the same steps on the same sample to print the remaining layers. The second and all ongoing layers should satisfy the condition that only non-conflicting patterns are printed on the same layer. This problem can be modelled as a graph, and it would correspond to a vertex coloring problem, where the vertices represent the patterns of the mask and the edges represent the conflicts between these patterns [TL19b]. Since this problem can be modelled as a graph problem, there are two methods that can be applied to find a solution:

1. Applying a graph algorithm.
2. Building a MIP model and using a MIP solver.

Graph Algorithm for Solving the Vertex Coloring Problem

The first step of the graph algorithm that solves the vertex coloring problem would be to arrange the colors that will be assigned to the vertices in a specific order, forming a type of list. In this approach, some specific sets will also be used: the colorable set and the uncolorable set. The colorable set contains all vertices that are eligible for coloring with a specific color, while the uncolorable set contains vertices that are not eligible for coloring with that color. Notice that a node in the colorable set is not colored yet, but it is not uncolorable. At the beginning of the algorithm, the set of colorable vertices contains all vertices of the graph, while the set of uncolorable vertices is empty. The algorithm itself is then a greedy approach to the vertex coloring problem. A greedy algorithm consists of some iterative steps, which give priority to the current best choice without taking into consideration any factor in the future [Bar+08]. Therefore, the solution obtained from a greedy approach is not guaranteed to be optimal. For the

given problem, the optimality would be translated to the number of colors that are used in order to color every vertex without having two neighbouring vertices sharing the same color. The steps of the greedy approach would be the following [TL19b]:

1. Pick the first color from the list.
2. Use this color to color the starting vertex. All neighbouring vertices of the starting vertex are noted as uncolorable at this stage until further notice.
3. Use the same color once again to color the next possible colorable vertex and set its neighbouring vertices in the uncolorable set. Repeat this step until there are no more vertices in the colorable set.
4. Change the color from the list and mark all uncolorable vertices as colorable and repeat steps 2. and 3. shown above.

The time complexity of this algorithm would be linear ($O(n)$). A more optimal algorithm, that can be applied to solve the same problem, would be the Dsatur algorithm [Bré79], which itself is of a quadratic complexity ($O(n^2)$). This algorithm makes use of the degree of the vertices and the saturation degree of the vertices, which is equal to the number of colors assigned to the neighbours of the particular vertex.

MIP Model of the Vertex Coloring Problem

The first step in building an MIP model for any problem is to determine the types of variables that will be used for the linear constraints. In this case, binary variables shall be used once again, and these will be $b_{i,j}$ [TL19b]:

1. $b_{i,j} = 1$: i-th pattern is printed on the j-th layer.
2. $b_{i,j} = 0$: i-th pattern is not printed on the j-th layer.

There are also two sets that need to be declared: the set of patterns (P) and the set of layers (L). The first constraint would be:
\\Constraint 1: Each pattern needs to be printed in exactly one layer [TL19b].

$$\forall i \in P \quad \sum_{j \in L} b_{i,j} = 1 \tag{2.7}$$

For the other constraint, one needs to mathematically present the conflict between two patterns. This can be done by means of a relation in terms of discrete mathematics. In discrete mathematics, a relation sets a type of connection between two different

objects. A relation may not function in both directions. A relation may not function in both directions. In the sentence "The athlete likes avocados", two objects are involved: the athlete and avocados. The word "likes" represents the relation from the athlete to the avocados. Flipping this relation would mean checking whether it also holds in the opposite direction — in this case, asking whether "Avocados like the athlete". While this reversed statement is nonsensical in everyday language, it also fails mathematically. Even if avocados were capable of having preferences, there's no information suggesting that the athlete is an element of any set over which avocados express such a relation. For the conflict between the patterns in the layer assignment problem, however, the relation functions both ways. The two objects under inspection for this scenario are the two different patterns. If pattern a is in a conflictive relation with pattern b, then pattern b is also in conflict with pattern a. This relation in mathematics is presented as aRb and is translated as: "a is in conflict with b".

\\Constraint 2: Each pair of conflicting patterns needs to be printed in different layers [TL19b].

$$\forall aRb, \forall j \in L \quad b_{\mathrm{a,j}} + b_{\mathrm{b,j}} \leq 1 \tag{2.8}$$

In other words, either only pattern a is printed on layer j, only pattern b is printed on layer j, or neither of them is printed on layer j.

Having formulated these two constraints, one could plug them into an MIP solver and find a possible solution for the layer assignment problem. However, the solver may not deliver an optimal solution. In order to achieve such an optimal solution, the optimization target needs to be set. For the graph algorithm, the optimization target would be to use as few colors as possible to color every vertex of the graph, while for this model, it would be to minimize the total number of layers being used to print out the complete mask. In order to formulate this, some new binary variables need to be presented, which are the b_{j}. In this example, if $b_{\mathrm{j}} = 1$, it means that layer j has been used, while if $b_{\mathrm{j}} = 0$, it means that the j layer is not used.

\\Constraint 3: If at least one pattern is printed on layer j, then layer j has been used [TL19b].

$$\forall j \in L \quad \sum_{i \in P} b_{\mathrm{i,j}} \leq M * b_{\mathrm{j}} \tag{2.9}$$

In this case, M is a large constant to guarantee the validity of this constraint. In this particular example, the worst-case scenario would be that every layer contains only one pattern, thus meaning that the number of patterns would be equal to the number of used layers. Hence, for the constant M, one needs to ensure that this is greater than or equal to the cardinality of the set of patterns. However, in most MIP models where

such a variable is presented, it is simply set to a very large constant, like one billion, which would be very far from any possible worst-case scenario for different types of problems. Having connected the newly presented binary variables b_j to the already existing $b_{i,j}$, one could set the optimization target, which has already been described in words, in a mathematical form [TL19b]:

$$min \sum_{j \in L} b_j \tag{2.10}$$

As mentioned, this approach guarantees an optimal solution, but it may be slow to deliver it.

2.3.2 1-Source-1-Target Problem

The 1-source-1-target problem is a problem that can be modeled as an MIP model as well as a graph. In the latter case, the edges would be treated as physical connections between the vertices. As the name suggests, the 1-source-1-target problem is a typical problem for which a path is required to be found. This path should start at a specific starting vertex and finish at a specific target vertex. An example of such a problem in real life would once again be the GPS system that is commonly used to travel from one location to another. The starting vertex would be the current location of the user, while the ending vertex would be the desired destination. The edges would be all the roads that will be passed through in order to reach the final destination.

Graph Algorithm for Solving the 1-Source-1-Target Problem

There are two possible algorithms that can be applied to solve the 1-source-1-target problem, which are Dijkstra's algorithm [Dij59] and the A* algorithm [HNR68]. In reality, Dijkstra's algorithm not only solves the 1-source-1-target problem but also the 1-source-n-targets problem. On the other hand, the A* algorithm is very similar to Dijkstra's, with the difference that it applies heuristics to find the shortest path from one source to a target. The advantage of the A* algorithm is that it performs better in cases where the obstacles between the source and target are small, depending on the heuristics that are applied. If the obstacles are large, then Dijkstra's algorithm would perform better compared to the A* algorithm due to its searching wavefront. In order to understand the heuristics applied in the A* algorithm, Dijkstra's Algorithm needs to be examined first.

The first step of this algorithm would be to declare two sets of nodes: the processing set and the processed set. The value $dist_i$ represents the shortest distance from the

source node (s) to each possible target node (v_i). The steps that need to be followed are [TL19b]:

1. The source node is added to the set of processing nodes.
2. The node with the smallest value $dist_i$ is chosen from the set of processing nodes. This node is renamed as v_{min} and sent to the set of processed nodes.
3. All neighbours (v_{next}) of v_{min} are observed:
   a) If v_{next} is already in the set of processed nodes, ignore it.
   b) If v_{next} is in the processing set, check if $dist_{min}$ plus the edge weight from v_{min} to v_{next} is smaller than $dist_{next}$. If this is the case, update $dist_{next}$ = $dist_{min}$ + weight of the edge.
   c) If v_{next} is in neither the processed nor the processing set of nodes, it is added to the processing set with the value of $dist_{next}$ = $dist_{min}$ + weight of the edge.
4. Repeat steps 2 and 3 until the target node is stored in the processed set or until all reachable nodes from the source are stored in the processed set.

The main difference between the A* algorithm and Dijkstra's algorithm is the variable $dist_i$. Instead of having $dist_i$ represent the real shortest distance from the source node to each possible target, for the A* algorithm, $dist_i$ represents the estimated shortest distance, which is equal to the sum of the real shortest distance to a target node and the heuristic term. In the above algorithm, this change affects only the 2nd step, where instead of the term $dist_i$, the estimated shortest distance will be inserted.

Manhattan Distance

The heuristic that is applied in most cases for the A* algorithm is the calculation of the Manhattan distance. Finding the Manhattan distance between two points A and B, as shown in Figure 2.3, would be done by summing the length and width of the rectangle, considering that A and B are in the corners of the rectangle.

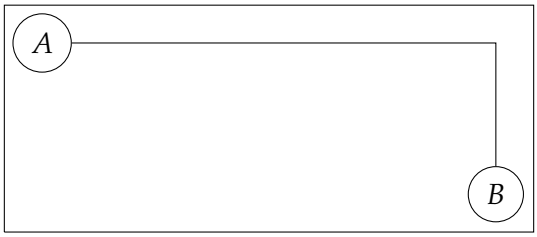


Figure 2.3: Example of the Manhattan distance between points A and B in a rectangle.

However, calculating the Manhattan distance in the A* algorithm refers to a different heuristic function. In this case, the algorithm needs to have information beforehand regarding the geometrical positions of the graph vertices. In other words, each vertex's x-coordinate and y-coordinate need to be saved for the calculation. The longest distance between each pair of nodes will be taken into consideration for the worst-case scenario in this problem. The Manhattan distance (also referred to as the L1 distance, originally proposed from Minkowsky [Her06]) between two vertices $A(x_a, y_a)$ and $B(x_b, y_b)$ would be calculated as follows:

$$M(A, B) = |x_a - x_b| + |y_a - y_b| \tag{2.11}$$

The heuristic function may not always refer to finding the Manhattan distance between two vertices. Other types of functions can be used as well, like the Euclidean distance [Her06]. As a rule of thumb, these functions should remain simple enough to obtain the heuristic term as soon as possible.

MIP Model of the 1-Source-1-Target Problem

In order to build a MIP model for the 1-source-1-target problem, it is essential to decide on the types and amount of variables needed beforehand. The main goal of the 1-source-1-target problem is to find a path that starts at a desired vertex and ends at a desired target vertex. Even though the graph models for this problem consist only of non-directed edges, in this case, it is recommended to decompose the non-directed edges into two directed edges, as shown in Figure 2.2. As mentioned in Subsec. 2.1.1, a path may pass through the same vertex multiple times, but it must not pass through the same edge multiple times. In order to track down the direction from which a vertex in the path is reached or from which direction the path leaves a vertex, it is essential to work with directed edges instead of non-directed edges. For any given vertex v_i, let $E_{i,\,in}$ be the set of incoming edges and $E_{i,\,out}$ be the set of outgoing edges. The binary variable b_k represents whether the k-th directed edge is part of the path or not. When building such MIP models, whose goal is to determine a specific path, the whole model typically consists of three parts [TL19a]:

1. The sending model.
2. The passing model.
3. The receiving model.

The constraint for the sending model refers to the starting vertex of the path. If a path starts at a specific vertex, then the number of outgoing edges with respect to this vertex

should be higher than the number of incoming edges. A mathematical formulation of this constraint would be:

\\Constraint 1: The number of outgoing edges from the starting vertex is one more than the number of incoming edges [TL19b].

$$\sum_{e_k \in E_{s,out}} b_k = \sum_{e_k \in E_{s,in}} b_k + 1 \tag{2.12}$$

The constraint for the receiving model refers to the target vertex of the path. If the path ends at a specific vertex, then the number of incoming edges with respect to this vertex should be higher than the number of outgoing edges.

\\Constraint 2: The number of incoming edges of the target vertex is one more than the number of outgoing edges [TL19b].

$$\sum_{e_k \in E_{t,in}} b_k = \sum_{e_k \in E_{t,out}} b_k + 1 \tag{2.13}$$

The constraint for the passing model refers to all intermediate vertices of the path that are neither the starting vertex nor the target vertex. For all these vertices, the number of incoming edges and outgoing edges should be equal. Another formulation for this constraint using attributes of vertices from Subsec. 2.1.2. In this case, the constraint should be applied to all vertices v for which the equality $indeg(v) = outdeg(v)$.

\\Constraint 3: The number of incoming and outgoing edges of all vertices that are neither a starting vertex nor a target vertex in the path should be equal to one another [TL19b].

$$\forall v_i \in V \backslash \{s,t\} \quad \sum_{e_k \in E_{i,in}} b_k = \sum_{e_k \in E_{i,out}} b_k \tag{2.14}$$

The optimization target for this problem would be to minimize the length of the path. In order to do so, a new set needs to be declared, namely E', which consists of all the directed edges of the graph. The assumption for this part of the problem is that to every edge, a weight factor (α_k) is assigned. The mathematical presentation of this optimization target would be [TL19b]:

$$min \sum_{e_k \in E'} b_k * \alpha_k \tag{2.15}$$

At first glance, there would be a problem with the formulation of such an optimization objective. As mentioned in Sec. 2.2, the constraints and objectives should be formulated in a linear manner. However, the presence of the multiplication sign in the former equation is not necessarily a contradiction. In this equation, the term b_k is a binary

variable, while α_k is a constant. Having a multiplication between a constant and a variable inside the equation does not make it non-linear. A problem would arise if α_k were a variable as well. The multiplication or division of two variables, whether binary or integer variables, is not linear, the same as having variables in specific powers or associated with functions like the exponential or logarithmic functions.

2.3.3 Planar Routing

The other type of problem that shall be examined in this part is also the planar routing problem. Similarly to the 1-source-1-target problem, planar routing treats the edges of a graph as physical connections, rather than logical connections. The difference between the 1-source-1-target problem and planar routing is that the latter can find a specific path from the starting vertex to an ending vertex, but also offers to find other paths. Let there be a set of starting vertices and a set of target vertices. The general idea of planar routing is to find a path from any of the starting vertices to any of the target vertices while not causing any conflicts. Conflicts in this sense would be crossings of different paths. In other words, if a vertex is part of one of the paths, it cannot be part of any of the other paths found by planar routing. Similarly, multiple paths cannot share the same edge of the graph. The reason why planar routing is preferred instead of solving the 1-source-1-target problem multiple times is that the latter may offer a bad solution where the paths intersect with one another. However, it cannot be denied that both problems are, in fact, very similar. This can also be deducted through the constraints of the MIP model of planar routing, which are slightly modified from the constraints of the MIP model presented for the 1-source-1-target problem.

MIP Model of Planar Routing

Similarly to the 1-source-1-target problem, the directed edges of the graph will be decomposed into two directed edges and for each vertex v_i, the sets $E_{i,\,in}$ and $E_{i,\,out}$ will be declared for the incoming and outgoing edges with respect to vertex v_i. Also, the binary variable b_k will be used once again to determine if the k-th directed edge is part of a path or not. Similarly to the former MIP model introduced in Subsec. 2.3.2, the current model will also consist of the sending model, the passing model, and the receiving model. However, the constraints from the model presented in Subsec. 2.3.2 need to be modified accordingly.

For the sending model, one needs to ensure that the sending node, which is chosen for any one of the paths, has exactly one outgoing edge and no incoming edges, rather than setting the number of incoming edges to be one more than the number of outgoing edges. Precisely, that would mean:

\\Constraint 1: The starting vertex has one outgoing edge and zero incoming edges [TL19b].

$$\sum_{e_k \in E_{s,out}} b_k = 1 \tag{2.16}$$

$$\sum_{e_k \in E_{s,in}} b_k = 0 \tag{2.17}$$

Similarly, for the receiving model, the target vertex of any of the paths must have exactly one incoming edge and no outgoing edges. In mathematical terms, one would have:

\\Constraint 2: The target vertex has one incoming edge and zero outgoing edges [TL19b].

$$\sum_{e_k \in E_{t,in}} b_k = 1 \tag{2.18}$$

$$\sum_{e_k \in E_{t,out}} b_k = 0 \tag{2.19}$$

For the passing model, the constraint that was used for the 1-source-1-target problem does not need to be modified at all and can be copied from the one presented in Subsec. 2.3.2:

\\Constraint 3: The number of incoming and outgoing edges of all vertices that are neither a starting vertex nor a target vertex in any of the paths should be equal to one another [TL19b].

$$\forall v_i \in V \backslash \{s,t\} \quad \sum_{e_k \in E_{i,in}} b_k = \sum_{e_k \in E_{i,out}} b_k \tag{2.20}$$

However, this is not the only constraint that needs to be dealt with in the passing model. As hinted above, the difference between planar routing and the 1-source-1-target problem is that planar routing offers to find multiple paths that do not intersect with one another. Until now, there has been no constraint that prevents these intersections from happening. In order to tackle this issue, a new constraint needs to be implemented:

\\Constraint 4: Two paths cannot enter the same vertex [TL19b].

$$\forall v_i \in V \backslash \{s,t\} \quad \sum_{e_k \in E_{i,in}} b_k \leq 1 \tag{2.21}$$

By setting the sum of the incoming edges for each vertex, apart from the starting and the target vertices, to smaller than or equal to 1, then if a specific intermediate vertex is

already part of one of the paths, then it cannot be part of any other paths, because the first found path has already entered this vertex and if another path enters the vertex, the sum would be equal to 2, which violates the constraint. This constraint is also fulfilled when the sum is equal to 0, which for the graph would mean that neither of the paths from the set of the starting vertices to the set of the target vertices has chosen to pass through this particular vertex.

The optimization target is also very similar to the one presented for the 1-source-1-target problem, with a slight change in the set of edges that it refers to. For this model, there is no need to declare a set of directed edges E', but the weight factors of the edges α_k are still used. The goal of this optimization is once again to minimize the length of the found paths [TL19b].

$$min \sum_{e_k \in E_{out}} b_k * \alpha_k \tag{2.22}$$

Changes of the Constraints from the 1-Source-1-Target MIP Model

As seen in the MIP model of planar routing, there are some differences from the proposed model of the 1-source-1-target problem. This is expected because generally the 1-source-1-target problem does not rule out the possibility that a cycle is formed. The first constraint of the 1-source-1-target problem is:

\\Constraint 1: The number of outgoing edges from the starting vertex is one more than the number of incoming edges.

$$\sum_{e_k \in E_{s,out}} b_k = \sum_{e_k \in E_{s,in}} b_k + 1 \tag{2.23}$$

Since the number of incoming edges of the starting vertex may not be equal to zero, it is implied that the path that starts from this vertex can also revisit this vertex at some point in the future. As mentioned in Subsec. 2.1.1, if a path enters the same vertex multiple times, then a cycle is formed [TL19b]. These cycles would not pose an issue in the case of the 1-source-1-target problem, but can be very problematic for planar routing. That is because in planar routing, the model has to ensure that no intersections occur between the different paths that need to be found. The presence of cycles, in this case, would increase the problem space in a non-necessary way. The formed cycles visit more vertices in the graph, which should be avoided. In order to do so, the constraints for the 1-source-1-target problem were modified for the MIP model of planar routing. Even without the intersection issue for planar routing, one could apply the same MIP model presented for planar routing (without the fourth constraint) in order to solve the 1-source-1-target problem.

2.4 Applying Multicast on a Wavelength-Routed Optical Network-on-Chip

The main goal of this thesis is to apply multicast on WRONoCs. Multicast refers to having a single vertex that sends signals to different receivers. These signals should all share the same wavelength λ (like 1000 nm, for instance). In a passive network / WRONoC, the paths that these signals follow from the sending vertex to the receiving vertices should be predefined in the design stage, which will be done with the two approaches from Ch. 3. Typically, applying multicast to a passive network relies on the strategic usage of splitters and MRRs.

The splitters can be placed in different locations on the network, which could be close to the starting vertex, as is the case for the first approach of this thesis, the bordered area approach. In this case, the splitting operation takes place at the very beginning, so the sending vertex has to be equipped with multiple senders (as will be explained in Sec. 3.1 there will be a maximum of four senders, meaning that a 1x4 splitter is required), each sending a message to different receivers. However, the splitting operation can be postponed, which is the case in the second approach, the layering model. In this case, there are multiple splitters (as will be explained in Sec. 3.2, these are 1x2 splitters) placed at specific vertices of the network.

On the other hand, placing MRRs in a passive network is also possible; however, it is only under the condition that the MRRs being used are passive as well. An MMR is considered to be passive if it cannot be switched during the transmission, meaning that it is either activated or deactivated all the time. Also, as mentioned in Ch. 1, these MRRs should also correspond to one single wavelength. For an active network, it is possible to reuse the same resources of MRRs and tune these to correspond to different wavelengths. With the help of the control system, it is also possible to turn the MRRs on or off depending on the specific case.

As will be explained in Ch. 3, in order to achieve multicast on passive mesh networks, there are specific subproblems that need to be solved for both of the proposed approaches (the bordered areas approach and the layering model). These subproblems mostly refer to finding a path or multiple paths, like the one-source-one-target problem or planar routing, respectively. In order to find the path from the sending vertex to a specific receiver, the paths found are connected to a specific vertex. These are the locations where the MRRs will be placed. The number of MRRs and splitters of different types varies for both approaches, and it is also dependent on other factors. These include the degree of the starting vertex in the bordered areas approach, and also the number of receivers per layer in the layering model.

2.4.1 Multicast in State-Of-The-Art Networks

Multicast in optical networks-on-chip (ONoCs) is not a new topic. However, most research has focused on active ONoCs rather than passive ones. One of the key differences between the two lies in their control systems: active ONoCs typically include dynamic control mechanisms to manage signal routing, while passive ONoCs lack such systems. In passive ONoCs, signal routes are generally predefined and fixed at design time, limiting their flexibility for dynamic communication patterns.

Applying multicast on an active ONoC normally relies on dynamically changing the state of the MRRs to either activated or deactivated. This is done by a control mechanism. Two leading methods that help with multicast and broadcast in mesh networks are FLONA [Wan+24], a network setup designed for that purpose, and GPRMM [Yan+23], a strategy that groups multicasts to reduce how many wavelengths are needed and to save power. This second strategy only applies to active NOCs, while FLONA also considers predefined data paths, which is an attribute of a passive ONoC. However, in the process of transmitting a message from a sender to a receiver, the control unit is notified, and this one is tasked with activating or deactivating certain MRRs, which is typical for an active ONoC. There are also methods that apply multicast on an active ring network, like DWRMR, which stands for Dynamically-established and Wavelength-Reusable Multicast Rings [Liu+16], and also makes use of MRRs being dynamically tuned by a control mechanism.

Attempting to adapt both of these strategies to a passive ONoC would result in failure due to the lack of a control unit. In this thesis, the two proposed approaches do not rely on MRRs that can be dynamically activated or deactivated.

3 Methodology

The main goal of this thesis is to apply multicast on a mesh network, where the example of a 4×4 mesh topology will be studied in detail. Since the target group of the ONoC that is treated for this thesis is the wavelength-routed ONoCs (WRONoCs), the proposed algorithms will run at the design stage, and some information should already be known to the algorithm. This type of information will be the location of the starting vertex, which from now on will be referred to as the sending vertex. The target vertices will also be considered as receiving vertices during this chapter. As is the case for the problems in Ch. 2, the new ones that will be presented for the multicast problem can be modelled using mixed integer linear programming, as well as graphs. For simplicity, the graph models and algorithms will be studied first, and then the MIP model of the problems will be presented. The graphs that will be presented for both approaches will consist of non-directed edges (until these are decomposed for specific reasons). The vertices of these graphs represent the routers of the NoCs, while the edges represent the wiring between these routers. For each vertex, the geometrical position in a two-dimensional plane will be used for both approaches, while for the edges, the attribute of weight will also be taken into consideration. The vertex located at position $(0, 0)$ is always in the bottom left corner. For simplicity reasons, each hop from one vertex to any of its neighbouring vertices will result in the weight of that specific edge connecting the two nodes to be equal to one. The approaches proposed by this thesis are: the bordered areas approach and the layering model. Both of these offer the possibility of applying multicast on a mesh network and, with some adjustments, can also be used for a ring network.

3.1 The Bordered Areas Approach

The bordered areas approach is one of the proposed approaches to solve the multicast problem for a mesh network, with the main focus being the example of the 4×4 mesh network. A graph representation of this network is shown in Figure 3.1:

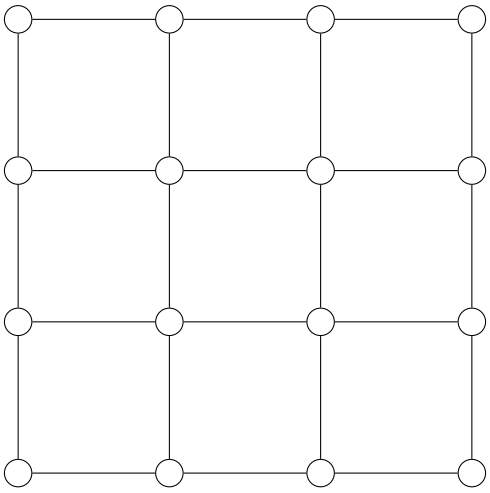


Figure 3.1: The graph of a 4×4 mesh network.

As mentioned, this approach takes into consideration that the sending vertex is known. The trademark of this approach is that the splitting operation takes place at the beginning. The number of receivers that can be reached by the sending vertex is determined by the splitting possibilities offered at this vertex. As can be seen in Figure 3.1, the vertices of the graph can be divided into a total of three different groups:

1. Vertices with a degree of two (four such vertices in total).
2. Vertices with a degree of three (eight such vertices in total).
3. Vertices with a degree of four (four such vertices in total).

The degree of the starting vertex determines the number of possibilities the algorithm has to split the light signal. For example, if the degree of the sending vertex is equal to two, then there are two possibilities for splitting the light signal. The two new signals have the same wavelength λ, so it is necessary for the power distribution network to provide a signal with the wavelength $2 * \lambda$ for the starting vertex. The degree of the starting vertex in this approach also determines the maximum number of receivers to which an optical signal with the wavelength λ can be transmitted. For a corner vertex

with a degree equal to two, the number of possible receivers is two at maximum, while it can also be one or zero, but not three or four. Apart from the number of receivers that is limited for this approach, there is also another limitation regarding the locations of the receivers. This second issue will be treated in more detail in the graph algorithm that proposes a solution for this problem.

3.1.1 Borders and Areas

Borders

As the name of the approach suggests, the algorithm that is proposed for this part will take into consideration two aspects: borders and areas. A border consists of a number of vertices and edges, which are chosen with regard to the sending vertex. It is therefore a subgraph of the mesh network. Similar to the maximum number of receivers and the possibility of splitting the light signal, the number of borders also depends heavily on the degree of the sending vertex. A corner starting vertex with a degree equal to two will form a total of two borders as well. Depending on how a border spans in a geographical sense, there are four possibilities of creating a border:

1. North Border (NB).
2. West Border (WB).
3. South Border (SB).
4. East Border (EB).

These borders contain vertices as well as edges, because they are subgraphs of the original graph. Since a graph is a non-empty set of vertices and edges, the borders are also a non-empty set of vertices and edges, namely:

1. $NB = \{NBV, NBE\}$.
2. $WB = \{WBV, WBE\}$.
3. $SB = \{SBV, SBE\}$.
4. $EB = \{EBV, EBE\}$.

NBV is a subset of the set of vertices V of the graph, and it contains all the vertices positioned in the subgraph NB. Similarly, NBE is a subset of the set of edges E of the graph, and it contains all edges that are part of the subgraph NB. This same logic applies to all the other borders, namely WB, SB, and EB. Taking the example

from Figure 3.1, if the sending vertex is the one in position (0,3), then a total of two borders can be formed. For this particular example, these borders will be the east border (EB) and the south border (SB). In this example, a border cannot span in the northern direction or the western direction, because the starting vertex is already at the northwestern corner of the topology. The eastern border vertices (EBV) in this case will be a subgraph that contains all the vertices located east of the sending vertex, while the southern border vertices (SBV) will contain all the vertices located south of the sending vertex. More precisely, that means:

$$EBV = \{(1,3),(2,3),(3,3)\} \tag{3.1}$$

$$SBV = \{(0,2),(0,1),(0,0)\} \tag{3.2}$$

To fully define the southern and eastern borders of this scenario, the sets EBE and SBE need to be filled as well:

$$EBE = \{(1,3)--(2,3),(2,3)--(3,3)\} \tag{3.3}$$

$$SBE = \{(0,2)--(0,1),(0,1)--(0,0)\} \tag{3.4}$$

All these values are based on a Cartesian system, where the x-axis spans from the origin to the right and the y-axis spans from the origin to the upper direction. A graph algorithm that would offer an opportunity to find these defined borders would be a modified version of the DFS and BFS algorithms, respectively. In order to find the western and eastern borders, the application of the BFS algorithm would be required, while finding the southern and northern borders would require the application of the DFS algorithm. The modified versions of these algorithms make it possible to differentiate between the northern and southern borders while applying the DFS algorithm. On the other hand, the modified BFS algorithm would be able to distinguish the western and eastern borders from one another. The modification of both algorithms will take place when the neighbouring vertices are inserted into either the queue for the BFS or the stack for the DFS. Compared to the algorithms presented in Ch. 2, the locations of all vertices are known for this problem. As a consequence, while checking for the neighbours, one can check whether these are below, above, to the right, or to the left of the sending vertex. These four possibilities correspond to finding the western vertices for the western bound, the eastern vertices for the eastern bound, the southern vertices for the southern bound, and the northern vertices for the northern bound. In other words:

1. If $x_\mathrm{s} = x_\mathrm{n}$ and $y_\mathrm{s} < y_\mathrm{n}$ then the neighbour n is in the norther border of the sending vertex s.

2. If $x_s = x_n$ and $y_s > y_n$ then the neighbour n is in the southern border of the sending vertex s.

3. If $x_s < x_n$ and $y_s = y_n$ then the neighbour n is in the eastern border of the sending vertex s.

4. If $x_s > x_n$ and $y_s = y_n$ then the neighbour n is in the western border of the sending vertex s.

The other modification in the formerly introduced algorithms would be to find the edges that connect the vertices of each bound to one another. The choice of including the sending vertex in the borders is arbitrary. If it is included, then all the inequalities presented above need to be modified from $<$ to $\leq$ and $>$ to $\geq$ respectively.

Areas

This approach also makes use of specific areas. Similar to borders, these areas are also subgraphs of the original graph representing the mesh network. Once again, the degree of the starting vertex determines the number of areas that will be formed. An area should always be formed between two borders. If there are two borders in total, then there is one area for this particular case. If there are three borders, then there are two areas in the graph, because they will share one of the borders with one another. In the case that four borders are found for a sending vertex, then the number of areas also corresponds to four. In this last case, each border is in contact with two areas. This is shown in Table 3.1.

Table 3.1: The relation between the degree of the starting vertex and the number of borders and areas obtained.

Degree of the starting vertex	2	3	4
# Borders	2	3	4
# Areas	1	2	4

These areas consist of a set of vertices and a set of edges as well. For simplicity reasons, these areas will be named A_1, A_2, A_3, and A_4. The decision whether A_1 for instance is between NB and EB or SB and EB is arbitrary. The choice of these areas is made to resemble the quadrants when working with trigonometric rules. Hence, throughout this thesis, the setting will be as shown in Figure 3.2.

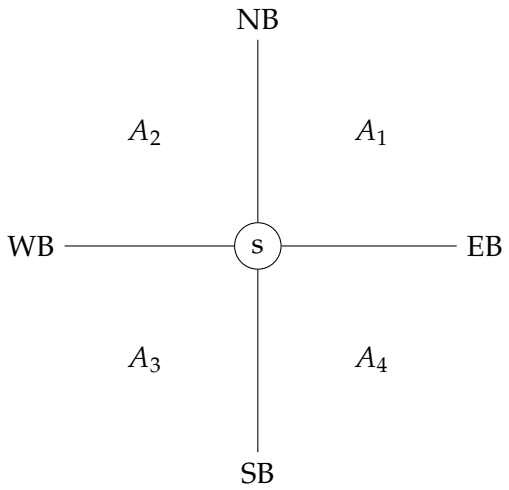


Figure 3.2: Configuration of areas for the bordered areas approach.

This means that A_1 is located between the eastern and northern borders, A_2 is located between the northern and western borders, A_3 is located between the western and southern borders and lastly A_4 is located between the eastern and southern borders. These areas, in a mathematical sense, are sets built as such:

1. $A_1 = \{A_1V, A_1E\}$.
2. $A_2 = \{A_2V, A_2E\}$.
3. $A_3 = \{A_3V, A_3E\}$.
4. $A_4 = \{A_4V, A_4E\}$.

Finding the elements of the set A_1V, for instance, would require a graph algorithm, similar to the one used for finding the elements of the sets NBV, WBV, SBV, and EBV. Compared to the border vertices, however, in this case, it is not necessary to differentiate between a BFS and a DFS algorithm. The time complexity of both algorithms is the same, and the order in which the vertices of A_1V are obtained is not important. The original BFS or DFS algorithm needs to be modified once again in the second step, where the neighbours first start to be inserted into the queue or the stack. In this insertion procedure, it needs to be ensured that the x-coordinate of the neighbour is greater than the x-coordinates of the vertices in the northern border, and the y-coordinate of the neighbour is also greater than the y-coordinate of the vertices in the eastern border. Note that in the northern border, all vertices share the same value for the x-coordinate, while in the eastern border, all vertices share the same value for

the y-coordinate. Apart from this modification, the new DFS or BFS algorithm should also deliver all edges connecting the found vertices. For the other areas, the vertex subsets will be defined as follows:

$$A_1V : \{v \in V, e \in EBV, n \in NBV | x_\mathrm{v} > x_\mathrm{n} \wedge y_\mathrm{n} > y_\mathrm{e}\} \tag{3.5}$$

$$A_2V : \{v \in V, w \in WBV, n \in NBV | x_\mathrm{v} < x_\mathrm{n} \wedge y_\mathrm{n} > y_\mathrm{w}\} \tag{3.6}$$

$$A_3V : \{v \in V, w \in WBV, s \in SBV | x_\mathrm{v} < x_\mathrm{s} \wedge y_\mathrm{n} < y_\mathrm{w}\} \tag{3.7}$$

$$A_4V : \{v \in V, e \in EBV, s \in SBV | x_\mathrm{v} > x_\mathrm{s} \wedge y_\mathrm{n} < y_\mathrm{e}\} \tag{3.8}$$

A rule for this algorithm is that each area must be bounded by one of the borders. Since each area is located between two borders, it is essential to give one of these borders priority over the other. Therefore, there are a total of three possibilities to prioritise the borders:

1. Right Border Priority.
2. Left Border Priority.
3. Both Borders Priority.

The third example will be referred to later on Subsec. 3.1.2. Between the first two, the choice is once again arbitrary. In Table 3.2, these possibilities of giving priority can be compared to one another.

Table 3.2: Comparing the left and right border priorities of the areas for the bordered areas approach.

Left Border Priority	Right Border Priority
$A_1 - NB$	$A_1 - EB$
$A_2 - WB$	$A_2 - NB$
$A_3 - SB$	$A_3 - WB$
$A_4 - EB$	$A_4 - SB$

Since this priority bounds an area to a specific border, the union of these two subgraphs will also be referred to in the discussion regarding the possible locations of the receivers in the network. Uniting the A_1 border to the eastern bound, would mathematically be presented as:

$$A_1EB = A_1 \cup EB \tag{3.9}$$

This new set will also include all the edges connecting the vertices of the border to the vertices of the area.

3.1.2 Number and Location of the Receivers

The total number of receivers should also be taken into consideration before presenting the following steps of the approach. As mentioned in Sec. 3.1, the maximum number of receivers is determined by the degree of the sending vertex. If this degree is equal to three, for instance, then there are three possible receivers in the network at most. The location of these receivers is also an important factor that needs to be taken into consideration. Since the whole graph is divided into the borders and areas subgraphs, a receiver can either be in an area or on a border. A limitation of this approach is that there cannot be multiple receivers on one single border. The first example that will be examined is that of a starting vertex with a degree equal to four.

Staring Vertex and Possible Locations of the Receivers

Considering the graph from Figure 3.1, there are four vertices that have a degree equal to four. These are located in positions (1,1), (2,1), (1,2), and (2,2). For this example, the vertex in position (1,2) will be analysed as the sending vertex. Since the degree of this vertex is equal to four, there are a total of four borders and four areas that can be found in the graph. These are all illustrated in Figure 3.3. The red colored vertex represents the sending vertex, while the blue colored vertices and edges represent the borders. The green color is used to present the areas.

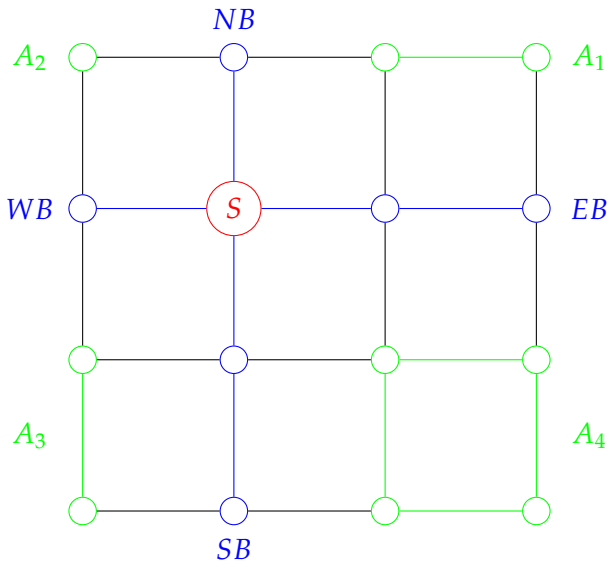


Figure 3.3: Sending vertex with degree four in the bordered areas approach.

This approach proposes that the maximum number of receivers in the case presented in Figure 3.3 is equal to four at most. If there are indeed four receivers, then there are a few possibilities for placing these in the network:

1. Each area has one receiver.
2. One area has two receivers.
    a) Two other areas have one receiver each, and the last area has no receivers.
    b) One other area has two receivers, and the remaining two have no receivers.

The first scenario is the trivial one, where the vertex in A_2, one of the two vertices in areas A_1 and A_3, respectively, and one of the larger areas A_4 would be the receiver. However, the receivers may also be placed on the borders. Taking into consideration the right border priority approach from Table 3.2, if a receiver is located exactly on the border and is given priority over the area, then the total number of receivers in this area decreases by one. If one area has two receivers in total, the scenario of both borders' priority needs to be considered for the approach. In the example shown in Figure 3.3, area A_4 may include two receivers in its set of vertices. In this case, both border priorities need to be triggered. The area prioritises the eastern and southern borders, meaning that the areas A_1 and A_2 need to decide which will be the one bound to the northern border (NB). Similarly, the areas A_2 and A_3 need to decide which will be bound to the western border (WB). If a receiver is located on the eastern border (EB), then the total number of receivers in area A_4 drops by one. This last receiver may then either be located in one of the green vertices of area A_4 or even in the southern border (SB). If the other remaining receivers are located in the areas A_2 and A_1, then the only possibility for reaching the receiver in area A_1 would be through the northern border (NB), while in order to reach the receiver in area A_2, the western border (WB) would be used. In this example, it would be impossible to place a receiver in area A_3 as well, because both the southern and western borders have already been utilized for the other receivers. In this scenario, the left border priority is actually applied for the areas A_1 and A_2. The situation would, however, be completely different if the remaining two receivers were located in area A_2 and A_3. Area A_3 would be the critical one, since one of its surrounding borders has already been used for the receivers in area A_4. This would mean that in order to reach the receiver in area A_3, the western border (WB) would be used, while for the area A_2, the northern border (NB) would be used. This other scenario corresponds to the right border priority. The conclusion from these scenarios is that if both border priorities are applied for one of the areas, then this will also be the case for all the remaining areas, which should make use of the next possible border. The areas that share one border with the one where the two receivers are located are the most critical ones and should be considered first.

Using the same example presented in Figure 3.3, it is also possible to visualize the cases where the sending vertex has a degree equal to three or two. If this vertex has a degree of three, then the first row in Figure 3.3 would be ignored (including the edges connecting the vertices of the first row to the vertices of the second row). In that particular case, the approach would only be able to distinguish three borders, these being the western (WB), the eastern (EB), and the southern (SB) borders. The total number of areas would be equal to two, and these would be the areas A_3 and A_4. The total number of receivers would be equal to three at maximum, with two possibilities of placing them:

1. Two receivers are located in area A_3 and one is located in area A_4.
2. Two receivers are located in area A_4 and one is located in area A_3.

The total number of receivers per area in this approach is limited to two per area. The area with the higher number of receivers would use both its surrounding borders to reach these receivers, while the other area would use the remaining border to reach the last receiver. Once again, the receivers may also be located on the borders themselves. In this case, the number of receivers in the area whose border has a receiver drops by one. There is also the possibility of each of these two areas having one receiver, which would make the approach easier. Using right border priority, the receiver in area A_4 would be reached from the southern border (SB) (if the receiver is itself not located exactly in the southern border), while the receiver in area A_3 would be reached from the western border (WB) (considering that the receiver is not located exactly on this border).

Lastly, the scenario of a sending vertex being the corner vertex can also be visualized using the same graph from Fire 3.3. In this case, the first row and first column would theoretically be deleted, leading to the starting vertex at the original (1,2) position to be a corner vertex with a degree equal to two. The number of borders in this case would be equal to two, and the ones found would be the eastern (EB) and southern (SB) borders. The only possible area to find in this example would be area A_4, which leads to two possible outcomes:

1. Two receivers are located in area A_4.
2. One receiver is located in area A_4.

If two receivers are located in the area A_4, then the eastern (EB) and southern (SB) borders would be used to reach these two receivers, considering that the receivers are not located directly in any of these two borders. Otherwise, if one receiver is located in area A_4 and right border priority is taken into consideration, then this receiver would be reached from the southern border (SB).

3.1.3 Reaching the Receivers

Up to this point, it has been mentioned that the receivers need to be reached from the starting vertex. In graph theory, this would mean that the problem formulation in the bordered areas approach is to find the paths that start at the sending vertex and end at the receiving vertices. The total number of these paths corresponds to the total number of receivers. Since paths need to be found, this problem will be compared to the two problems presented in Ch. 2, which proposed a solution for finding a path from a specific source to a specific target. These are the 1-source-1-target problem and planar routing. Before deciding which one of the two to use for the specific case, the exact location of the receivers is required. As mentioned in Subsec. 3.1.2, the receivers could either be at the border or in an area. Before considering multiple receivers in an area, let this part have a single receiver at most for each of the areas.

If the receiver is located on the border, then the path being searched is one that starts at the sending vertex and ends at the receiver without any need to redirect this path using an MRR, for example. This problem formulation is the same as the one for the 1-source-1-target problem and will be solved using the same methods. Solving it with a graph algorithm would mean applying either Dijkstra's algorithm or A* algorithm.

On the other hand, if the receiver is located in the area and not the border, then there are two issues to resolve for the desired path in this case:

1. Finding the path starting at the sending vertex and ending at one of the vertices of the respective border (using either right or left border priority).

2. Finding the path from this specific border vertex to the receiver in the area.

As shown, the more complicated problem can actually be split into two subproblems. Starting with the second subproblem, the main issue here is the specific vertex that is located on the border. This vertex should ideally function as the turning point for the path that is being searched. At first glance, it may seem like a multiple-sources-1-target problem. Changing the perspective to that of the receiving node in the area, this problem could also be formulated as a 1-source-multiple-targets problem, with the original receiver being considered as the source, while the targets are the vertices located on the border. However, there should be no multiple targets or sources for this problem. It is not required to find all paths starting from each vertex located on the border and ending at the receiver in the area, even though it is possible. The better method would be to first find the vertex on the border closest to the receiver in the area. In order to do so, a heuristic function can be applied. This would be the function for calculating the Manhattan distance, which was explained in Subsec. 2.3.2. This calculation would be done for all the vertices located on the border with respect to the

receiver located in the area. The vertex with the closest distance to the receiver will be the one chosen for solving both the presented subproblems. Usually, one would also consider the scenario of multiple border vertices having the same distance to the receiver, but that is not possible.

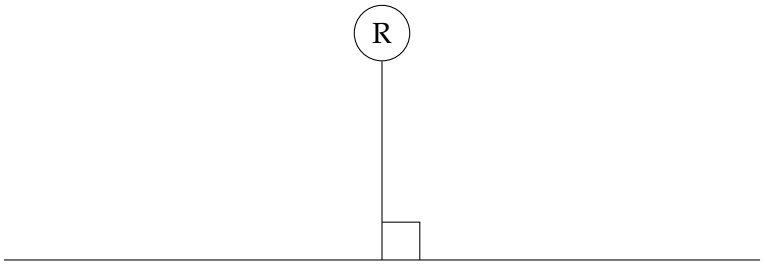


Figure 3.4: Shortest distance between the receiving vertex and a border.

All the vertices located on the borders either share the same x-coordinate value or the same y-coordinate value. More precisely, if the found bound is determined to be a southern (SB) or northern (NB) bound, then the vertices share the same x-coordinate. Depending on the starting vertex's y-coordinate and the y-coordinates of these vertices, it would be possible to determine whether these vertices are located in the southern (SB) or northern (NB) border. Similarly, all the vertices in the eastern (EB) and western (WB) borders share the same y-coordinate, while the x-coordinates compared to that of the starting vertex will determine whether these are located in the southern (SB) or northern (NB) border. The horizontal line in Figure 3.4 represents a random border, while the vertex R represents a receiver located in the area bound by this border. Mathematically, finding the shortest distance from a point to a line would mean finding the perpendicular between the point and the line. This perpendicular makes contact with the line at one specific point of the lane, forming a 90° angle between the two lines. This perpendicular cannot be found at any other position along the line and represents the closest distance from the point to the line. In other words, there is only one vertex on the border that is closest to the receiver. This vertex should be located in such a position that a perpendicular can be constructed from the receiving vertex to the border. Depending on the border, this vertex may either share the same x-coordinate or y-coordinate with the receiving vertex. In the case of the southern (SB) and northern (NB) border, this particular vertex will share the same y-coordinate with the desired border vertex, while for the eastern (EB) and western (WB) borders, this vertex will share the same x-coordinate with the desired border vertex. However, applying the Manhattan distance is the more general approach to finding the shortest distance. The concept from Figure 3.4 shall only explain why there is only one vertex on the border

whose distance is the smallest to the receiving vertex.

$$M(A, B) = |x_a - x_b| + |y_a - y_b| \tag{3.10}$$

If A is considered to be the border vertex and B is considered to be the receiving vertex, then in this equation, either the first or the second term will be equal to 0. For all the other vertices in the border, neither of these terms would be equal to zero, considering that the receiver is not located on the border itself.

Multiple Receivers per Area

As mentioned in Subsec. 3.1.2, an area might have either two, one, or zero receivers. If no receivers are located in the area, then no path needs to be found. In the case of one receiver, the specific border vertex closest to the receiver in the area needs to be found. However, the problem becomes more difficult when two receivers are located in the same area. Apart from the priority, which in this case changes to both borders priority for all the areas of the graph, one needs to consider the possibilities to reach the receivers. Finding these paths is a complicated problem that needs to be decomposed into three subproblems:

1. Finding the path starting at the sending vertex and ending at one of the vertices of the first border.

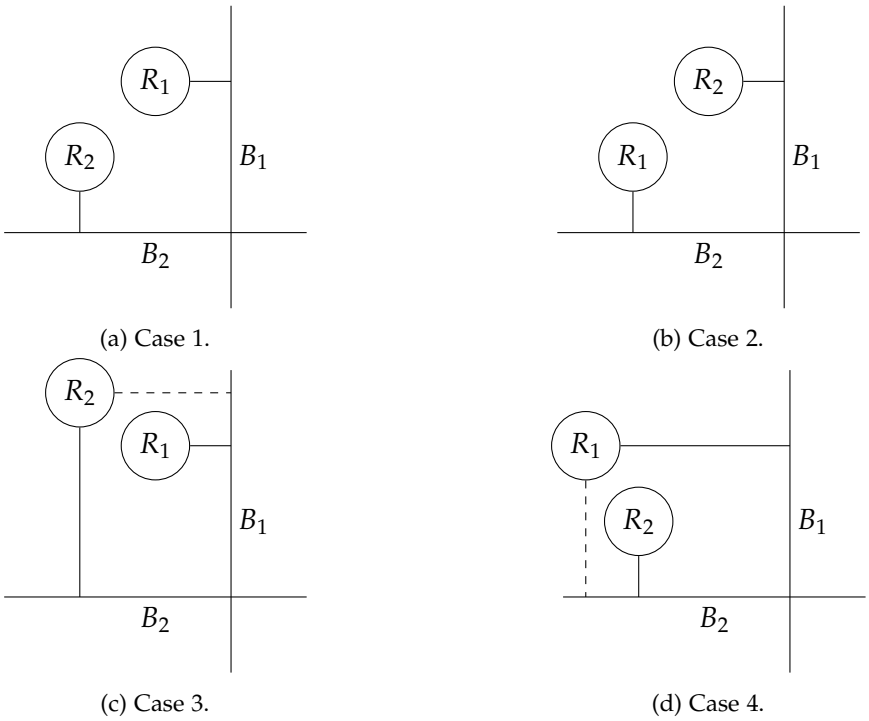
2. Finding the path starting at the sending vertex and ending at one of the vertices of the second border.

3. Finding the paths from these border vertices to the receivers in the area.

The first two subproblems resemble the first subproblem presented for the case of a single receiver at most in any area. The only difference here is that this should be done twice for both borders surrounding the area. In both cases, the 1-source-1-target problem needs to be applied. The third subproblem will be treated differently from all the other ones presented until this point. Finding these specific vertices in the borders is the challenging part of this case. Previously, there was only one receiver and one border to consider, but the number of receivers and borders for this case is equal to two. Let the receiver in this area be referred to as R_1 and R_2, while the borders be referred to as B_1 and B_2. The main issue for this part is whether R_1 or R_2 is closer to B_1. The furthest one will automatically be associated with B_2. There is a total of four cases for this scenario:

1. R_1 is closer to B_1 and R_2 is closer to B_2.

2. R_1 is closer to B_2 and R_2 is closer to B_1.
3. R_1 is closer to B_1 and R_2 is closer to B_1.
4. R_1 is closer to B_2 and R_2 is closer to B_2.



(a) Case 1.

(b) Case 2.

(c) Case 3.

(d) Case 4.

Figure 3.5: Scenarios with two receivers per area in the bordered areas approach.

These cases are shown in Figure 3.5. The first two cases are the trivial ones where the receivers are each closer to different borders, while the third and fourth cases are not that intuitive. In case three, for instance, the receiver R_1 is closer to the border B_1 compared to R_2 and the same border. The problem is, that the receiver R_2 is further from border B_2 compared to receiver R_1 and this same border. The question that arises for this particular case is whether it is better to connect R_2 with the B_1 border, even though R_1 is closer to B_1. In order to answer this question, this scenario will be analyzed in the 4×4 mesh network, as shown in Figure 3.6:

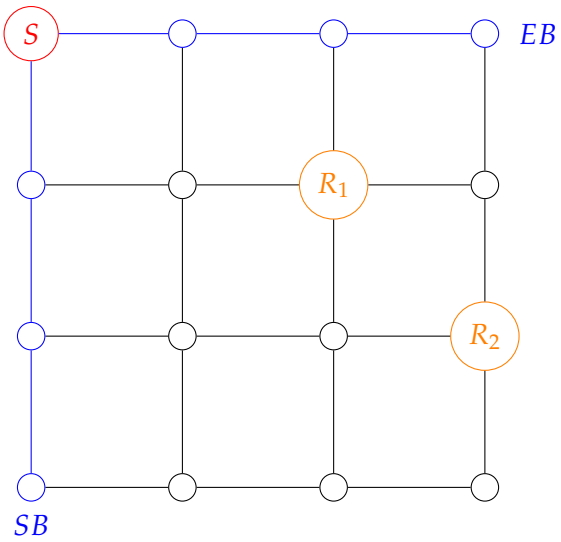


Figure 3.6: Bordered areas approach with two receivers in one area.

Figure 3.6 shows the bordered areas approach with the sending vertex located in position (0,3). With respect to this sending vertex, two borders can be found. These are the southern (SB) border and the eastern (EB) border. A northern border cannot be found because there is no vertex in the graph with a higher y-coordinate value than the starting vertex. Similarly, no western bound can be found, because the starting vertex has the lowest possible value in its x-coordinate. The area surrounded by the eastern (EB) and southern (SB) borders is area A_4, which consists of the remaining nine vertices of the graph. Among these vertices, there are two receivers, R_1 located in position (2,2) and R_2 located in position (3,1). R_1 is located closer to the eastern (EB) border then to the southern (SB) border. However, the receiver R_2 is also located closer to the eastern (EB) border than to the southern (SB) border. Connecting R_1 to the eastern (EB) border would cost one hop, from this vertex to the vertex in location (2,3) on the eastern (EB) border, which is the closest to it. If this is done, then R_2 remains to be connected to the southern (SB) border, although it is closer to the eastern (EB) border. Whether to connect R_1 to SB and R_2 to EB is under consideration. For both scenarios, the costs are calculated in Table 3.3. The second column for each method of the table shows the costs of reaching the receiver from the border, while the third one shows the costs of reaching the specific border vertex from the sending vertex S. As can be calculated in this table, the sum of the total costs is equal to eight (four in total for reaching both receivers from the borders and four for reaching the specific border vertices from the sending vertex). Choosing between these methods would be a complicated problem in terms of MIP, because the whole calculation needed to find the

closest vertex located on the border to the receiver would be repeated several times.

Table 3.3: Costs of both scenarios in Figure 3.6.

Method A			Method B		
Connection	Cost B-R	Cost S-B	Connection	Cost B-R	Cost S-B
$R_1 - EB$	1	2	$R_1 - SB$	2	1
$R_2 - SB$	3	2	$R_2 - EB$	2	3

In order to pair each receiver to one specific bound, even if this is not optimal, the heuristic function of the Manhattan distance can be applied once again. For each receiver, one would need to calculate the Manhattan distance to the vertices on the eastern (EB) and southern (SB) borders. Out of these four obtained values, the smallest one should be chosen. In the case from Figure 3.6, the smallest Manhattan distance found would be equal to one, which corresponds to the distance between R_1 and the vertex (2,3) in the eastern (EB) border. For R_2, the choice is limited to only one of the borders, which in this case is the southern (SB) one.

So, the specific vertices on the borders will act like starting vertices, while the receivers will be the target vertices. A similar problem was encountered in Subsec. 2.3.3 where planar routing was discussed. However, the original planar routing algorithm presented in Subsec. 2.3.3 mentions that the multiple paths found from the set of starting vertices to the set of target vertices should not intersect with one another. The paths required for this specific problem should also not intersect with one another, but they cannot start at a random starting vertex and finish at a random target vertex. These pairs, as explained earlier, are fixed at the beginning of the algorithm. Therefore, the algorithm for planar routing needs to be slightly modified in order to adapt to the new changes.

3.1.4 MIP Model

Now that the desired vertex on the border can be located, the problem can be solved using the algorithms presented in Ch. 2. For multiple subproblems, a path needs to be found that resembles the 1-source-1-target problem, while for the last subproblem presented in the scenario of multiple receivers in one area, a variation of the planar routing algorithm is required. In order to construct the MILP model for the whole problem, one could construct MILP models to solve the subproblems and connect these in order to obtain a solution for the whole problem. The only issue with the bordered areas approach regarding its MIP model is the heuristic function being used.

Manhattan Distance MIP Model

The Manhattan distance between two vertices was calculated using the following equation:

$$M(A,B) = |x_a - x_b| + |y_a - y_b| \tag{3.11}$$

As mentioned in Ch . 2, MIP is a mathematical modelling method consisting of a linear set of equations and inequalities that are connected logically using the logical AND ($\wedge$) operator. The main issue with the Manhattan distance formula is that it consists of two absolute values. The variables x_v and y_v representing the x- and y-coordinates of the graph's vertices will be declared as integer variables for building the MIP model. The absolute value of a variable is not a linear operation. Thus, before discussing the MIP model of the bordered areas approach, the absolute value of a variable should be linearized. The absolute value of the difference between two variables, like $X_{diff} = |x_a - x_b|$ from the former equation, can be split into two cases [TL19a]:

$$x_{diff} = |x_a - x_b| = \begin{cases} x_{diff} = x_a - x_b, & \text{if } x_a - x_b \geq x_b - x_a \\ x_{diff} = x_b - x_a, & \text{if } x_a - x_b \leq x_b - x_a \end{cases}$$

$$x_{diff} = |x_a - x_b| = \begin{cases} x_{diff} = x_a - x_b, & \text{if } x_{diff} \geq x_b - x_a \\ x_{diff} = x_b - x_a, & \text{if } x_a - x_b \leq x_{diff} \end{cases}$$

These two cases can be logically connected with the logical OR ($\vee$) operator. Splitting the two equations and using a binary auxiliary variable q to transform the logical OR ($\vee$) relationship to a logical AND($\wedge$) relationship would lead to a total of six constraints [TL19a]:

$$x_{diff} \geq x_a - x_b - q * M \tag{3.12}$$
$$x_{diff} \leq x_a - x_b + q * M \tag{3.13}$$
$$x_{diff} \geq x_b - x_a - (1-q) * M \tag{3.14}$$
$$x_{diff} \leq x_b - x_a + (1-q) * M \tag{3.15}$$
$$x_{diff} \geq x_b - x_a \tag{3.16}$$
$$x_{diff} \geq x_a - x_b \tag{3.17}$$

Out of these six constraints, the last one makes the first constraint redundant, and the second-to-last one makes the third constraint redundant. Therefore, constraints

one and three can be deleted, leaving four necessary constraints to model the absolute value:

$$x_{\text{diff}} \leq x_{\text{a}} - x_{\text{b}} + q * M \tag{3.18}$$
$$x_{\text{diff}} \leq x_{\text{b}} - x_{\text{a}} + (1 - q) * M \tag{3.19}$$
$$x_{\text{diff}} \geq x_{\text{b}} - x_{\text{a}} \tag{3.20}$$
$$x_{\text{diff}} \geq x_{\text{a}} - x_{\text{b}} \tag{3.21}$$

This model has to be repeated for the differences of the y values as well, thus leading to the constraints:

$$y_{\text{diff}} \leq y_{\text{a}} - y_{\text{b}} + q * M \tag{3.22}$$
$$y_{\text{diff}} \leq y_{\text{b}} - y_{\text{a}} + (1 - q) * M \tag{3.23}$$
$$y_{\text{diff}} \geq y_{\text{b}} - y_{\text{a}} \tag{3.24}$$
$$y_{\text{diff}} \geq y_{\text{a}} - y_{\text{b}} \tag{3.25}$$

This type of modelling, in terms of mixed integer linear programming, is referred to as hierarchical modelling. These two models can be used in tandem to fully model the Manhattan distance between two vertices of the graph.

1-Source-1-Target Problem

Some subproblems that were formulated for the bordered areas approach required finding a path from a specific sending vertex to a target vertex, which directly refers to the application of the 1-source-1-target problem. The subproblems that require such a path are:

1. Finding the path from the sending vertex to the receiver located on any of the borders.
   a) Vertex s represents the starting vertex while vertex t represents the receiver r.
   b) The set V of the third constraint will be replaced by the border: NB, EB, SB, or WB, depending on the location of the receiver.
2. Finding the path from the sending vertex to the specific border vertex (after applying the Manhattan distance).

a) Vertex s represents the starting vertex while vertex t represents the closest border vertex to the receiver in the area, c.

b) The set V of the third constraint will be replaced by the border: NB, EB, SB, or WB, depending on the location of the receiver and the priority used.

3. Finding the path from the specific border vertex (after applying the Manhattan distance) to the receiver in the area.

a) Vertex s represents the closest border vertex to the receiver in the area, c, while vertex t represents the receiver r.

b) The set V of the third constraint will be replaced by the area: A_1EB, A_2NB, A_3WB, or A_4SB, depending on the location of the receiver and assuming right border priority.

These changes will take place within the constraints of the 1-source-1-target problem, considering a modification in the first and second constraints that disables the formation of cycles. These three constraints are:

\\Constraint 1: The number of outgoing edges from the starting vertex is equal to one, and there are no incoming edges.

$$\sum_{e_k \in E_{s,out}} b_k = 1 \tag{3.26}$$

$$\sum_{e_k \in E_{s,in}} b_k = 0 \tag{3.27}$$

\\Constraint 2: The number of incoming edges of the target vertex is equal to one, and there are no outgoing edges.

$$\sum_{e_k \in E_{t,in}} b_k = 1 \tag{3.28}$$

$$\sum_{e_k \in E_{t,out}} b_k = 0 \tag{3.29}$$

\\Constraint 3: The number of incoming and outgoing edges of all vertices that are neither a starting vertex nor a target vertex in the path should be equal to one another.

$$\forall v_i \in V \backslash \{s,t\} \quad \sum_{e_k \in E_{i,in}} b_k = \sum_{e_k \in E_{i,out}} b_k \tag{3.30}$$

\\Optimization objective: Minimize this path being found.

$$min \sum_{e_k \in E'} b_k * \alpha_k \tag{3.31}$$

Planar Routing Variation

The last subproblem discussed in Subsec. 3.1.3 required finding two paths that start at one specific vertex on each border surrounding the area and end at the two receivers located in the area. For finding these specific vertices on the border, the MIP model of the Manhattan distance was presented. With the results of this model, it is possible to form the pairs c_1 - r_1 and c_2 - r_2, where the vertices c_1 and c_2 represent the closest vertices (located on the borders) to the receivers r_1 and r_2.

For each pair of vertices (i, j) for each path $k \in \{1, 2\}$, the binary variables ${b_{(i,j)}}^k$ and ${b_{(j,i)}}^k$ are introduced to determine whether the edge in direction (i, j) or (j, i) are located on path k or not. In total, there are four binary variables declared for each of the edges of the subgraph where this problem takes place. If the example from Figure 3.6 is taken into consideration, this subgraph would be:

$$A_4EBSB = A_4 \cup EB \cup SB \tag{3.32}$$

For simplicity, this subgraph will be referred to as G' with $G' = \{V', E'\}$ containing all the edges and vertices of the subgraph A_4EBSB.

\\Constraint 1: Each non-directed edge is visited at most once.

$$\forall i, j \in V' \wedge (i, j) \in E' \quad \sum_{1 \leq k \leq 2} ({b_{(i,j)}}^k + {b_{(j,i)}}^k) \leq 1 \tag{3.33}$$

\\Constraint 2: A vertex is visited at most once.

$$\forall v \in V' \quad \sum_{1 \leq k \leq 2} \sum_{(x,v) \in E'} {b_{(x,v)}}^k \leq 1 \tag{3.34}$$

\\Constraint 3: For all the vertices that are neither the starting vertex nor the target vertex of any of the paths, whenever the path enters the vertex, it must also leave the vertex.

$$\forall v \in V' \backslash \{c_1, c_2, r_1, r_2\}, \forall k \in \{1, 2\} \quad \sum_{(x,v) \in E'} {b_{(x,v)}}^k = \sum_{(v,x) \in E'} {b_{(v,x)}}^k \tag{3.35}$$

\\Conctraint 4: c_1 and c_2 have each one outgoing edge and no incoming edges.

$$\forall j \in V' \quad \sum_{(c_1,j)} {b_{(\mathrm{c_1,j})}}^1 = 1 \tag{3.36}$$

$$\forall j \in V' \quad \sum_{1 \leq k \leq 2} \sum_{(j,c_1)} {b_{(\mathrm{j,c_1})}}^k = 0 \tag{3.37}$$

$$\forall j \in V' \quad \sum_{(c_2,j)} {b_{(\mathrm{c_2,j})}}^2 = 1 \tag{3.38}$$

$$\forall j \in V' \quad \sum_{1 \leq k \leq 2} \sum_{(j,c_2)} {b_{(\mathrm{j,c_2})}}^k = 0 \tag{3.39}$$

$$\tag{3.40}$$

\\Constraint 5: Each target has one incoming edge in the corresponding path and no outgoing edges in any path.

$$\forall i \in V' \quad \sum_{(i,r_1)} {b_{(\mathrm{i,r_1})}}^1 = 1 \tag{3.41}$$

$$\forall i \in V' \quad \sum_{1 \leq k \leq 2} \sum_{(r_1,i)} {b_{(\mathrm{r_1,i})}}^k = 0 \tag{3.42}$$

$$\forall i \in V' \quad \sum_{(i,r_2)} {b_{(\mathrm{i,r_2})}}^2 = 1 \tag{3.43}$$

$$\forall i \in V' \quad \sum_{1 \leq k \leq 2} \sum_{(r_1,i)} {b_{(\mathrm{r_2,i})}}^k = 0 \tag{3.44}$$

$$\tag{3.45}$$

\\Optimization objective: Minimize the lengths of the found paths.

$$min \sum_{e \in E'} \sum_{1 \leq k \leq 2} {b_{\mathrm{e}}}^k * w_{\mathrm{e}} \tag{3.46}$$

The term w_{e} corresponds to the weight of the edge. This is assumed to be equal to one for all edges throughout the thesis.

3.1.5 Adaptation of the Bordering Areas Approach to a Ring Network

The proposed bordered areas approach can also be applied to a ring network. The main problem of a ring network compared to a mesh network is the location of a vertex. While in a mesh network, it is possible to determine the borders with respect to any starting vertex by comparing the x- and y-coordinates of the vertices, this strategy would not be appropriate in a ring network. Some vertices on the border may not

share any of the coordinate values with the ones from the starting vertex. Therefore, a new approach for determining the borders with respect to any starting vertex will be proposed. Similarly to the mesh network, the degree of the starting vertex determines the possible borders that can be found and also the maximum number of receivers. Since in a ring network all vertices have a degree equal to two, there are always two borders that can be found, and two receivers can be placed at most in this network. A method to find these borders would be by applying a clustering algorithm, like the K-means algorithm [Llo82] [Has24]. The two neighbouring vertices of the starting vertex can be set as the original means of the two clusters (borders), which will be found. After this step, the remaining vertices of the graph will be placed in one of the two clusters, depending on the distance between these vertices and the original means of the two clusters. This comparison will be done using the results of Dijkstra's algorithm on this graph. K-means will also recalculate the means of the clusters by searching for a better candidate inside the cluster. The distance between all vertices and these new means will then be calculated. It may occur that one or multiple specific vertices change clusters. These steps are repeated until no changes are found between the clusters of two consecutive steps of the algorithm. The two formed clusters represent the borders of the bordered areas approach. While a border in the bordered areas approach in the mesh network either contains one or no receivers, the clusters of the same approach in a ring network also contain either one or no receiver. In order to determine the path starting at the starting vertex and ending at any of the receivers, either the 1-source-1-target problem or the variation of planar routing will be applied. If there is only one receiver in the whole ring network, the 1-source-1-target model will be used in order to find the desired path. Otherwise, if there are two receivers in the ring, the model of the variation of planar routing will be applied, with the only difference that instead of two different sources and two different targets, in this case, there is a single source and multiple targets. This model will be presented in more detail in Sec. 3.2.

3.2 The Layering Model

The second proposed approach to solve multicast on a given mesh network is the layering model. This approach will be studied in a similar manner to the bordered areas approach. The first step will be to split the problem into a total of three subproblems. These three subproblems can be solved by either the exact methods presented in Ch. 2 or by modifying these problems. A possible graph algorithm and MIP model will be presented to solve multicast with the layering model. As is the case with the bordered areas approach, the layering model will also be constructed based on a starting vertex. The mesh network that will be used as an example for the layering model is the same 4×4 mesh network that was presented by the graph in Figure 3.1. The layering model postpones splitting the optical signal as much as possible. The maximum number of receivers in this case is determined by the size of the network. A 4×4 mesh network offers the possibility of using the same wavelength λ to reach a total of four receiving vertices at maximum. The sending vertex will be the only one on layer 0, and its degree determines the maximum number of layers that can be found. After the layers are determined, a sending path starting from the sending vertex and ending at a vertex in the last found layer will be found. Each node of the sending path will be equipped with splitters to send a message with a wavelength λ to each receiver. Depending on the total number of possible receivers, the Power Distribution Network (PDN) must ensure that the sending vertex has sufficient optical power to transmit a message using wavelength λ. This power will then be divided by a 1×4 optical splitter, assuming there are four reachable receivers in the network. This approach treats the whole mesh network as a series of ring networks built onto one another. The layering model consists of the following subproblems:

1. Building the layers with respect to a sending vertex s.
2. Finding the sending path starting at vertex s and ending at the last layer.
3. Finding the path from the vertex in the sending path to the receivers located on the layers.

3.2.1 Defining the Layers

The layers of this model are all formed based on the sending vertex. The sending vertex itself is placed on layer 0, while the other layers will be filled with all the remaining vertices of the graph. The first layer will be formed using the neighbours of the sending vertex. There are two types of neighbours: direct and diagonal neighbours. The direct neighbours are the vertices that are directly connected to the sending vertex on layer

0. In this case, an edge exists that connects the sending vertex to its neighbour. A diagonal neighbour, on the other hand, is a vertex that is directly connected to two direct neighbours. Figure 3.7 shall serve as an example for this explanation:

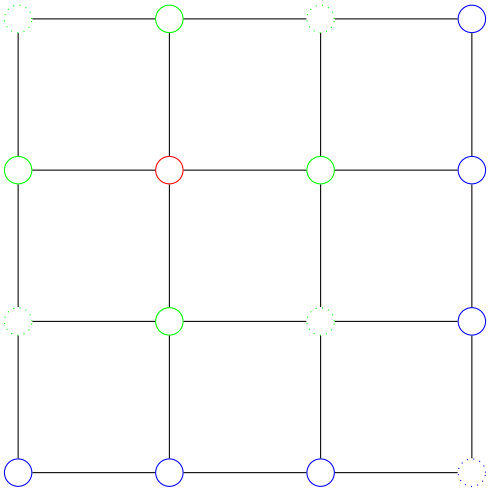
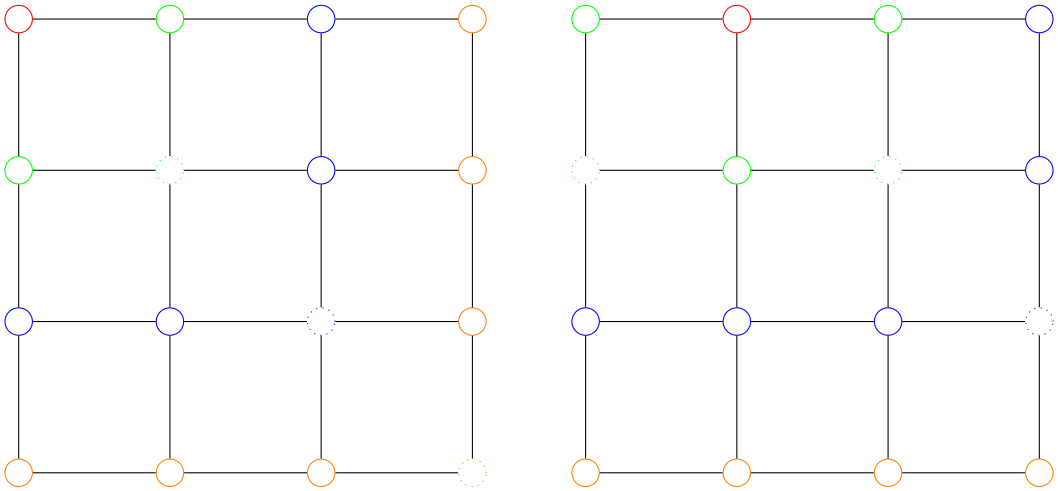


Figure 3.7: Direct and diagonal neighbours of the sending vertex for the layering model.

In Figure 3.7, the sending vertex is the one colored in red and located in position (1,2). The direct neighbours of this vertex are the ones located in positions (0,2), (1,1), (2,2), and (1,3). These are colored green. As can be observed by the differences in the x- and y-coordinate values of these vertices, the difference to the sending vertex is always equal to one. An important notice for these vertices is that either the x- or y-coordinate value changes compared to the sending vertex, but not both at the same time. The diagonal neighbours of the sending vertex are the vertices located in positions (0,3), (2,3), (0,1), and (2,1). These vertices are colored green and drawn by a dotted line. In this case, both the x- and y-coordinates change by one unit compared to those of the sending vertex. These four vertices are connected to two direct neighbours of the sending vertex. All the remaining vertices in the graph are either connected to one direct neighbour of the sending vertex or to no direct neighbour of the sending vertex. None of these vertices qualifies to be on the first layer of the layering model. The unified set of the directed and diagonal neighbours of the sending vertex represents the first layer of the layering model. Recursively, in order to determine the second layer of the layering model, the direct and diagonal neighbours of all the vertices in the first layer need to be determined. In the example from Figure 3.7, all vertices that are colored in blue are the direct neighbours of the vertices in layer one. This means that all these vertices are placed on layer two. The only diagonal neighbour of the vertices that are already placed in layer one is the vertex located at the position (3,0),

which is connected to two direct neighbours of the vertex (2,1) in layer one (namely the vertices (3,1) and (2,0)). This vertex is also part of the second layer. At the end of this assignment process, each vertex has to be assigned to one layer. In this example, the sending vertex has a degree equal to four, and the number of layers found is equal to two (excluding layer 0, which corresponds to the sending vertex itself). A different sending vertex may lead to a different number of layers that can be found.



(a) Building the layers for a sending vertex with a degree equal to two.

(b) Building the layers for a sending vertex with a degree equal to three.

Figure 3.8: Building the layers for a sending vertex with degree two and three.

The examples presented in Figure 3.8 present the layers constructed for the layering model when the degree of the sending vertex is equal to two and three. Once again, the sending vertex is colored in red, while the layers 1, 2, and 3 are colored in green, blue, and orange, respectively. The dotted vertices on each layer represent the diagonal neighbours of the vertices in the previous layer. In these examples, the number of layers found is equal to three (excluding layer 0). For a 4×4 mesh network, it would be impossible to find more layers using this method. There are only three possibilities when choosing a starting vertex: a vertex with degree 2, 3, or 4. The examples of the seeing vertex being with a degree of 2 and 3 offer the formation of three layers, as shown in Figure 3.8, while if the degree 4 is chosen for the sending vertex, the number of layers decreases by one, as was presented in Figure 3.7. Ultimately, the receivers of the layering model are all located on the different layers, with the limitation that a layer can contain at most two receivers. In other words, for each layer, there are at most two vertices that serve as receivers for this approach. Therefore, at first glance, it

would mean that the greater the number of layers of the layering model is, the more receivers can be reached using this approach. However, this does not necessarily mean that starting at a vertex with degree equal to four is not desired or should be avoided. Even though the number of layers for the examples in Figure 3.8 is one greater than the number of layers for the example shown in Figure 3.7, some problems may occur when the sending vertex is chosen with a degree of two or three.

Total Number of Vertices in a Layer

As mentioned, each layer represents a set of vertices and edges, where the vertices are chosen based on the neighbourhood condition (either direct or diagonal neighbours) and the edges are the ones connecting all vertices belonging to the same layer. Since layer 0 always consists of only the sending vertex, there is a maximum number of vertices that can be placed on layer 1. More precisely, there are four direct neighbours and four diagonal neighbours, which add up to a total of eight vertices. The first layer can be considered as a square where each side is three units. There are four sides in total, and on each side, there are three vertices. Each vertex on each side will have one direct neighbour in the upcoming layer, which adds up to twelve direct neighbours located on layer two. Apart from the direct neighbours, there are always four diagonal neighbours for each layer, meaning that the total number of vertices on layer two can be equal to sixteen at most. Recursively, calculating the maximum number of vertices per layer can be done with the following series:

1. Layer $0 \rightarrow 1$ vertex.
2. Layer $1 \rightarrow 1 * 4$ direct neighbours + 4 diagonal neighbours = 8 vertices.
3. Layer $2 \rightarrow 3 * 4$ direct neighbours + 4 diagonal neighbours = 16 vertices.
4. Layer $3 \rightarrow 5 * 4$ direct neighbours + 4 diagonal neighbours = 24 vertices.
5. Layer $4 \rightarrow 7 * 4$ direct neighbours + 4 diagonal neighbours = 32 vertices.
6. Layer $i \rightarrow (i + (i - 1)) * 4$ direct neighbours + 4 diagonal neighbours = $8 * i$ vertices.

This result means that for any layer i, there can be $8 * i$ vertices located on this layer. This rule excludes only layer 0, because by definition, this layer only contains the sending vertex s.

Full and Non-full Layers

Based on the total number of vertices located on a layer, these can be categorized into two groups: full layers and non-full layers. A layer is considered to be full if the number of vertices on this layer equals the maximum number of vertices located on this layer, which was calculated to be $8 * i$ for any layer i. If the number of vertices on a layer is smaller than $8 * i$, then the layer is referred to as a non-full layer. By definition, layer zero will always be a full layer, because it will always contain the sending vertex. After applying the layering model, the whole graph will consist of either full or non-full layers. It is expected that the last layer of a 4×4 mesh network will always be a non-full layer, since there is no central vertex in this kind of network. The last layer can only be full on odd square topologies, like 3x3 or 5x5. Even in these cases, only a single choice for the sending vertex would lead to the last layer being a full layer. Normally, the more layers are found in the network, the higher the probability that the last ones are non-full layers. If a found layer is determined as a full layer, this does not guarantee that the next layer will be a full layer as well. In the example from Figure 3.7, layer one is a full layer, because it consists of eight vertices, but layer two is non-full because it contains only seven vertices when it could contain a total of sixteen vertices. On the other hand, if a layer of the network turns out to be a non-full layer, then the upcoming layers will also be non-full layers. This can also be proven using the networks from Figure 3.8. In both cases, the first layer is non-full because it contains three and five vertices, respectively. There should have been eight vertices on this layer for it to be considered a full layer. As can be seen in both cases, all the upcoming layers, namely layers two, three, and four, are non-full layers. A full layer has the same structure as a ring network, which is the inspiration of this approach.

Number and Location of the Receivers

Determining if a layer is full or not is important for choosing the algorithm, which offers a path starting at the sending vertex and ending at the receiver. In the layering model, all vertices are assigned to specific layers. Among these, some serve as receivers. Each layer of the layering model has either zero, one, or two receivers. The total number of receivers that can be found on a layer is determined by the type of the layer. A full layer can contain two receivers, and there will always be a possibility of reaching both of the receivers. A non-full layer may contain two receivers as well, but there is no guarantee that both of these receivers can be reached. If the path that leads to the furthest receiver on the non-full layer can be constructed only by passing through the closest receiver, then in the current non-full layer, there is only one receiver that can be reached, which is the closest one. The concepts of closest and furthest refer to the

distance between the receivers in the current non-full layer and the vertex of the current layer that is also part of the sending path. The total number of receivers depends on the total number of layers found. In the worst-case scenario, the number of receivers is exactly equal to one when all the found layers are non-full layers. In the best-case scenario, the number of receivers is exactly equal to double the number of found layers when all these found layers are full layers. For most cases, the total number of receivers is between these two borders. The sooner the first non-full layer is found, the fewer receivers may be placed on this network. In order to maximize the connectivity, the first non-full layer is desired to be located as far away as possible from the sending vertex. In reality, the location of the first non-full layer is predetermined by the location of the starting vertex. Layer 0 is the only full layer that, by definition, will consist of no receivers because it must only contain the sending vertex.

3.2.2 Sending Path

A path should be determined where the splitting operation of the optical signal will take place. This path is referred to as the sending path. Mathematically, it is a subgraph of the original graph G. The conditions to form such a path are the following:

1. This path should start at the sending vertex on layer 0.
2. This path should end at a vertex located on the last found layer.
3. This path should pass through all the layers.
4. For each layer, only one vertex can be part of this path.
5. The vertices of the path must not be receivers.

The sending path will consist of the vertices, where the splitting operation will take place. The splitters located on each vertex depend heavily on the number of receivers that can be placed on a layer. If this number is equal to two, then the light signal should be split and two signals with the wavelength λ should be propagated to their final destinations. The method to do so will be presented in Sec. 3.2.3. In order not to complicate the algorithm any further, the location where the splitting takes place should not be a receiver. That is the reason why the sending path should not include any receivers. In other words, each vertex of the sending path will theoretically function as a sender on its layer. That is why it should not be a receiver. Such a sending path can be found for the example presented in Figure 3.7. As shown in Figure 3.9, the sending path starts at the sending vertex located on layer 0 in position (1,2) and it ends on layer two in position (3,2), with layer two being the last found layer. This path goes through

all layers (zero, one, and two), and all these layers have only one of their vertices placed on this path. All the vertices of the sending path are direct neighbours to one another as well. The yellow path fulfills all set requirements, and therefore, it is a valid sending path. This path consists of:

$$SP = \{(1,2),(2,2),(3,2)\} \cup \{(1,2)--(2,2),(2,2)--(3,2)\} \tag{3.47}$$

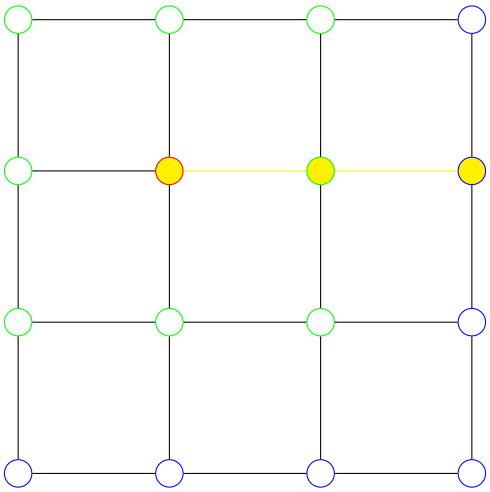
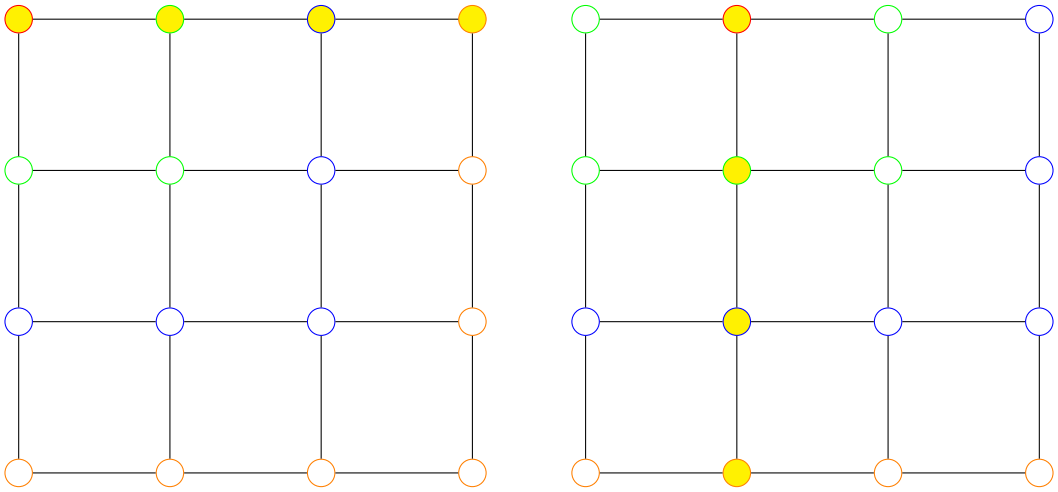


Figure 3.9: The sending path from the example shown in Figure 3.7.

However, the sending path colored in yellow is not the only possibility of constructing a sending path in the current example. Another valid path would be $(1,2)--(1,1)--(1,0)$. This path starts at the sending vertex and ends in the last found layer. It also contains only one vertex of all the found layers. This example proves that more than one sending path may exist for a given network. At this point, there is no real reason why one should be chosen over the other, so both are taken into consideration. An important property of the intermediate vertices, meaning those located on the sending path but excluding the source and target vertices, is their degree. These vertices must have a degree of at least three, typically varying between three and four. A degree of two is not allowed for intermediate vertices, as it would mean the vertex has only one incoming and one outgoing edge along the sending path, leaving no available connection to any receiver on its layer. Since intermediate vertices are expected to act as senders on their respective layers, they must have at least one additional edge to reach a receiver. If an intermediate vertex has a degree of three, it can transmit a signal to exactly one receiver on its layer. If the degree is four, it may reach two different receivers. Therefore, the degree of an intermediate vertex directly determines how

many receivers it can access within its layer. As mentioned, in these cases, the type of layer is also vital to fully determine if it is possible to have two receivers on this particular layer. The sending paths of the networks from Figure 3.8 serve as an example.



(a) Intermediate vertices of the sending path have a degree of two.

(b) Intermediate vertices of the sending path have a degree of four.

Figure 3.10: Sending paths of the graphs from Figure 3.8.

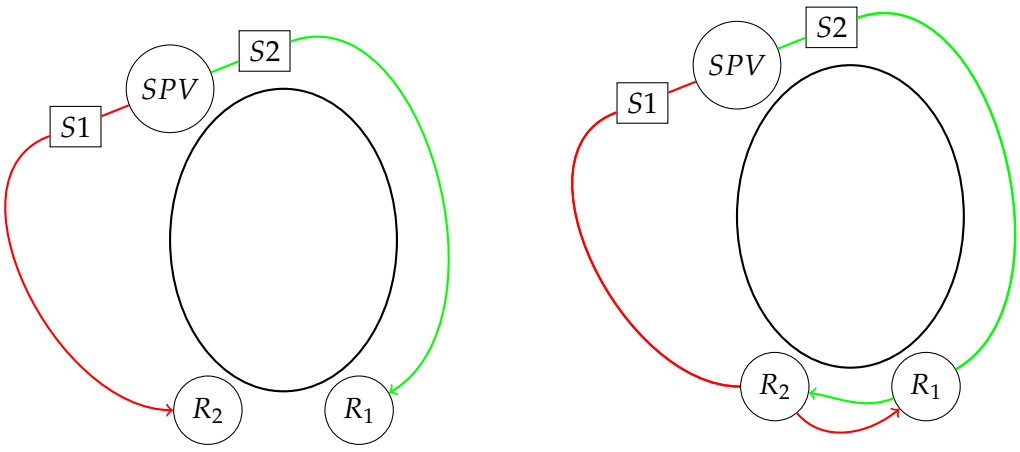
The sending paths in both cases are presented in yellow. These are $(0,3) - -(1,3) - -(2,3) - -(3,3)$ and $(1,3) - -(1,2) - -(1,1) - -(0,1)$ for the left and right graphs respectively. The left graph might also choose $(0,3) - -(0,2) - -(0,1) - -(0,0)$ as the sending path, while for the right graph, the presented sending path is the only one that can be found.

For the sending vertex, there is only the need to have a degree greater than or equal to one, which is always the case for the 4×4 mesh network. The target vertex, on the other hand, must have a degree of at least two. The sending path needs to enter this vertex but not leave it, which results in the other edge being used for sending the optical signal along the corresponding layer, which in this case is the last layer, where, under these conditions, only a single receiver can be placed. If the target vertex has a degree equal to three, then the sending path enters this vertex, and there can be two receivers on the last layer that could be reached from this vertex. As mentioned, this depends on the location of the receivers in the last layer as well. An important observation is that the target vertex of the sending path cannot have a degree equal to four. If this happens, then there is a contradiction with the conditions set for the sending path, namely that all the layers need to be visited. In this case, not all the layers

are visited if the degree is equal to four, because on one edge, the sending path enters the target vertex, and it can be transmitted to two receivers at most on the same layer. This leaves the last edge incident on this vertex unvisited, which means that the vertex on the other side of this edge has not yet been assigned to a layer in the worst-case scenario.

3.2.3 Reaching the Receivers

On each layer, there is a single vertex that serves as the sender. This is located on the sending path. In order to reach the receivers from the vertex on the sending path, the type of layer needs to be considered. As explained in Sect. 3.2.1, the layers are divided into two groups, which are the full layers and non-full layers, based on the number of vertices that are located on the layer under inspection. A full layer is similar in structure to a ring network, where all the vertices have a degree of two. The advantage of a full layer when using multicast is the fact that it is a cyclic graph. As explained in Subsec. 2.1.1, a cycle is a path that starts and ends on the same vertex. In a ring network, a cycle can be found regardless of where the path starts. In the case of a full layer, the paths to the receivers must start at the vertex on the sending path. Two receivers on the layer can be reached because there are only two possible edges from which to transmit a message. Optical signals, in this case, can either be transmitted in the clockwise direction or in the counterclockwise direction. This should be done in such a manner that no overlapping between optical signals occurs.



(a) No overlapping (b) Overlapping

Figure 3.11: Multicast for a full layer of the layering model.

As shown in Figure 3.11, an overlap may occur if no former measurements are taken. In this scenario, SPV, which stands for "Sending Path Vertex", is the sending vertex of the layer, while the vertices R_1 and R_2 are the receivers located on this layer. The sending vertex is equipped with two senders in order to send two signals of the same wavelength λ to two different directions at the same time. These are referred to as S_1 and S_2. As suggested in this example, the sending vertex SPV should reach the receiver R_1 in the clockwise direction and receiver R_2 in the counterclockwise direction. If this is not the case, then an overlap occurs, which must be avoided. The green and red colored lines in Figure 3.11 represent the clockwise and counterclockwise directions, not different wavelengths assigned to these paths. In both scenarios, the wavelength λ is the same. This problem is very similar to planar routing, with the modification that the starting vertex should be the same for both paths being found.

While placing two receivers on the layer and reaching these is possible for a full layer, the same does not always hold for a non-full layer. If the degree of the intermediate vertex of the sending path is equal to four, then two receivers might be placed on a non-full layer as well, but reaching these two is not always possible. It can only occur under some conditions. The main issue of a non-full layer is that it is an acyclic graph, meaning that no path can be found that starts and finishes on the same vertex. The starting vertex will be the one located on the sending path, while the two receivers R_1 and R_2 can be located in four different locations relative to this vertex:

1. R_1 is closer to the sending vertex SPV in the clockwise direction and R_2 is closer to the sending vertex SPV in the counterclockwise direction.

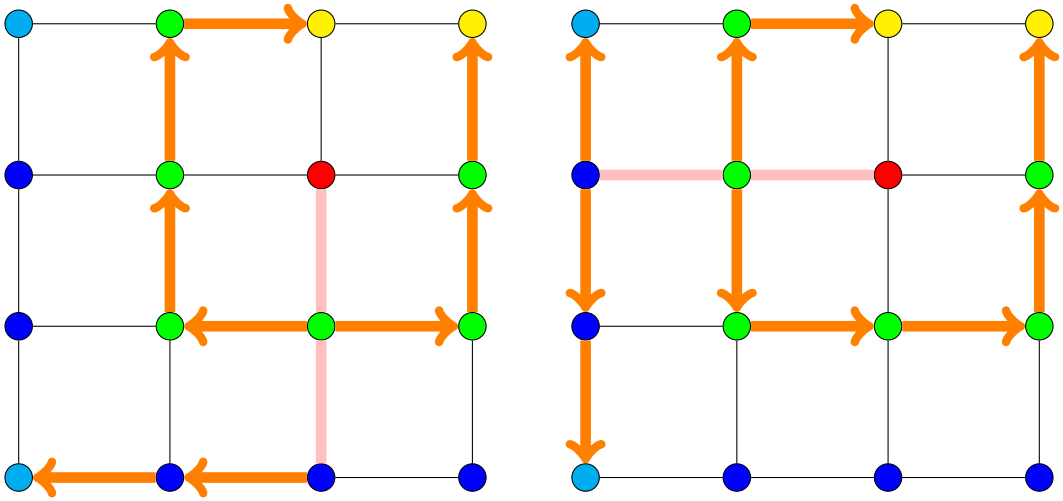
2. R_2 is closer to the sending vertex SPV in the clockwise direction and R_1 is closer to the sending vertex SPV in the counterclockwise direction.

3. Both R_1 and R_2 are closer to the sending vertex SPV in the clockwise direction.

4. Both R_1 and R_2 are closer to the sending vertex SPV in the counterclockwise direction.

The first two cases do not pose a problem, since the two paths can be constructed. Both start at the sending vertex SPV and reach their target vertex, which is either R_1 or R_2, in the clockwise and counterclockwise directions. The last two cases are problematic for a non-full layer. If both the receivers are located on one of the sides of the sending vertex SPV, then the closest of the two needs to be connected with the sending vertex directly, forming the first path. For the second path, one would theoretically start at the sending vertex once again and follow the other direction until the next receiver is reached. In the case of a full layer, this does not pose any problems,

while for a non-full layer, there is no possibility of resolving this issue. The absence of cycles in a non-full layer makes it impossible to use the other direction to reach the next receiver. If the receivers are placed in such a manner on a layer, then reaching two receivers on a non-full layer might be impossible. The only possibility of placing two receivers on this layer would be to start the sending operation from another vertex on the layer. For another sending vertex, this scenario might transform into either the first or second one, which are both acceptable. Changing the sending vertex would mean that the whole sending path must be replaced with another candidate. That is the reason why multiple sending paths are taken into consideration in Subsec. 3.2.2. The sending path candidates are important for this part of the procedure. If one of the candidates resolves this issue, then it is given priority over the other candidates and is chosen to be the final sending path. Such a scenario is shown in Figure 3.12.



(a) First found sending path.

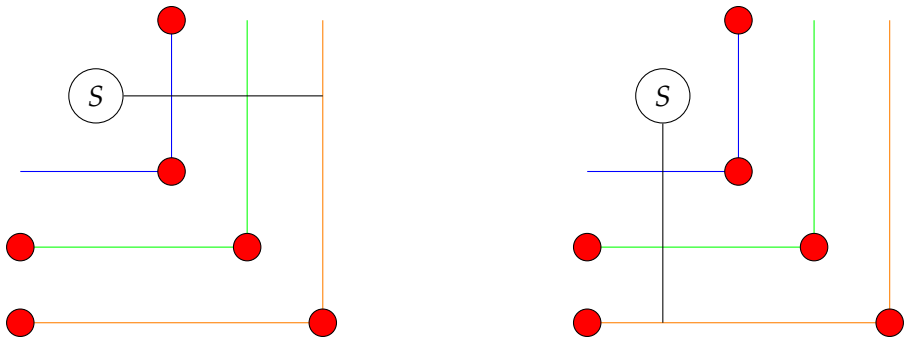
(b) Candidate sending path.

Figure 3.12: Choosing between two different sending paths.

In the example shown in Figure 3.12, the red vertex in position (2,2) represents the sending vertex, while the vertices colored in green and blue represent the vertices in layers one and two, respectively. The yellow vertices are located on the first layer, and these are the two receivers of this full layer, while the vertices colored in light blue are located on the second layer (which is a non-full layer) and represent the two receivers of this layer. The pink colored path represents the sending path in each case, while the orange paths are the ones that reach the receivers while starting at a vertex of the sending path. Since the first layer is full, the two receivers can be reached regardless of the chosen sending path. In the first case, the clockwise path costs four hops, while the

counterclockwise path costs three hops, and in the second case, the clockwise path costs two hops, while the counterclockwise path costs five hops, both adding up to a total of seven hops. The problem occurs in the second layer, which is a non-full layer. The first sending path ends at the vertex located in position (2,0). However, both receivers located in positions (0,0) and (0,3), respectively, are closer to the vertex of the sending path in the clockwise direction. Since the layer is not full, the counterclockwise direction cannot be used to reach the furthest receiver, which in this case is the one located in position (0,3). On the contrary, the sending path from Figure 3.12 (b) ending on vertex (0,2) is an eligible sending path which makes it possible to reach both receivers, because in this case the receiver in position (0,3) is closer to the vertex of the sending path in the clockwise direction, while the receiver in position (0,0) is closer to the vertex of the sending path using the counterclockwise direction. In such a scenario, the candidate sending path, namely $(2,2)--(1,2)--(0,2)$, is given priority over the first found sending path, which is $(2,2)--(2,1)--(2,0)$.

However, changing the sending path for this reason is a greedy approach to solving the problem. As mentioned earlier, if a non-full layer is encountered in the layering model, then all the upcoming layers will also be non-full layers. Assuming that five layers exist in a given network and the first non-full layer encountered is the third one, then the fourth and fifth layers are also non-full layers. For this case, it is also assumed that in the fourth and fifth layers, two receivers can be placed and reached without any issues. In the third layer, the receivers are placed in such a manner that one of them cannot be reached using the clockwise (or counterclockwise) direction. In this network, two sending paths can be initially found, and the first one is chosen. Choosing the second candidate as the sending path would resolve the issue on the third layer but cause problems in the fourth and fifth layers, where only one receiver can be reached. In total, the number of receivers reached drops by one, because for one additional receiver reached in the third layer, two already reachable receivers on layers four and five, respectively, cannot be reached with a different choice of the sending path. Therefore, choosing another candidate from the list of the sending paths candidates should only be done if the total number of receivers that can be reached is larger than the existing number; otherwise, swapping the sending path would not be optimal.



(a) First sending path. (b) Second sending path.

Figure 3.13: Choosing between two different sending paths for non-full layers.

The two examples from Figure 3.13 are an illustration of these scenarios. The blue colored layer is the third layer of the layering model, and the first one to be non-full as well. The green and orange colored layers are the fourth and fifth non-full layers of the approach. Each layer has two receivers, which are colored red, and the sending path starting from the sending vertex S and ending at the last found layer (in this case, layer five) is colored black.

3.2.4 MIP Model

Building the MIP model of the whole layering model will be done by building the MIP models of all the subproblems treated for this approach. All these subproblems require either grouping the vertices in specific clusters or finding a specific path/multiple paths. These models will be inspired by the ones presented in Ch. 1, where the vertex coloring problem, the 1-source-1-target problem, and planar routing were presented. Connecting the MIP models of the subproblems to one another forms the complete MIP model of the whole problem.

MIP Model of Building the Layers

Building the layers in the layering model is very similar to the vertex coloring problem presented in Subsec. 2.3.1. In this subproblem, the graph G of the network consists of edges that represent a logical connection between the vertices, rather than physical connections. While in the vertex coloring problem, a color was assigned to every vertex, the vertices should be assigned to specific layers in this problem. The sets that need to be declared in order to build the layers of the layering model are: V (the set of vertices of graph G), E (the set of edges of graph G), and L (the set of layers fo the graph G). This last set contains all possible layers that can be found, and it can be compared to the

set of layers presented in the vertex coloring problem. Similarly, the set of vertices V can be compared to the set of patterns P in the vertex coloring problem. The variables that are used to build the MIP model of this subproblem are binary variables of the form $b_{\mathrm{v,l}}$ which represent that the vertex v is located on layer l. The first constraint is very similar to the one presented for the vertex coloring problem.

\\Constraint 1: Each vertex is positioned in exactly one layer.

$$\forall v \in V \quad \sum_{l \in L} b_{\mathrm{v,l}} = 1 \tag{3.48}$$

The second constraint of the vertex coloring problem states that two patterns (vertices) with a conflicting relation cannot be placed on the same layer. The analogy to the current problem is that vertices that are neither direct nor diagonal neighbours of the vertices in the former layer cannot be placed on the current layer. So, in order to adapt the second constraint of the vertex coloring problem to the current subproblem of building the layers, some constraints to determine the direct and diagonal neighbours are required. If there exists an edge that directly connects a vertex in layer l_{i} to a vertex in layer $l_{\mathrm{i+1}}$, the vertex in layer $l_{\mathrm{i+1}}$ is a direct neighbour of the vertex in layer l_{i}. For this part of the model, some new variables need to be declared, namely:

1. $b^e_{v,l}$ which represent that vertex v is in layer l and is incident to the edge e.
2. b^e_v which represents that vertex v is incident to the edge e.

The first step of building the layers of the layering model is constructing the first layer, which consists of only the sending vertex of the graph.

\\Constraint 2: Layer 0 consists of only one vertex (which is the sending vertex).

$$\sum_{v \in V} b_{\mathrm{v,l_0}} = 1 \tag{3.49}$$

With the help of the new variables, it is also possible to determine the direct neighbours of the sending vertex, which are located on layer one. All vertices in the graph that are directly connected to the vertex in layer 0 are part of the first layer. In other words, if a vertex v_1 is in layer l_0 and is incident to edge e while the vertex v_2 is also incident to this same edge e, then it means that the vertex v_2 is in layer l_1.

$$\forall v_1, v_2 \in V, \forall e \in E \quad if \quad b^e_{v_1,l_0} = 1 \quad and \quad b^e_{v_2} = 1, \quad then \quad b_{v_2,l_1} = 1 \tag{3.50}$$

To bring this constraint in the form A implies B, the first two binary variables need to be merged into a sum of binary variables. This sum has to be equal to two in order for both conditions to be satisfied. This equation can then later be split into two inequalities combined by the logical AND ($\wedge$) operator.

$$b^e_{v_1,l_0} + b^e_{v_2} = 2 \quad \Leftrightarrow \quad b^e_{v_1,l_0} + b^e_{v_2} \geq 2 \quad \wedge \quad b^e_{v_1,l_0} + b^e_{v_2} \leq 2 \tag{3.51}$$

Since this is a sum of two binary variables (meaning that their values are either 0 or 1), it can never be greater than two, so the first inequality implies that the sum is exactly equal to two, since it is only satisfied when both the variables $b^e_{v_1,l_0}$ and $b^e_{v_2}$ are equal to 1. The second inequality implies that the sum can also be strictly smaller than two, which could happen if only $b^e_{v_1,l_0}$ is equal to 0, or only $b^e_{v_2}$ is equal to 0, or both are equal to 0. Neither of these cases would satisfy the equality $b^e_{v_1,l_0} + b^e_{v_2} = 2$. That means that A is represented by:

$$A : b^e_{v_1,l_0} + b^e_{v_2} \geq 2 \tag{3.52}$$

Since $A \Rightarrow B \Leftrightarrow \neg A \vee B$, $\neg A$ needs to be determined as well:

$$\neg A : b^e_{v_1,l_0} + b^e_{v_2} < 2 \quad \Leftrightarrow \quad b^e_{v_1,l_0} + b^e_{v_2} \leq 1 \tag{3.53}$$

With this information, it is possible to find the direct neighbours located on the first layer.

$$\forall v_1, v_2 \in V, \forall e \in E \quad b^e_{v_1,l_0} + b^e_{v_2} \leq 1 + b_{v_2,l_1} \tag{3.54}$$

This constraint should be generalized for all the layers of the approach. The only vertices that will be assigned to the upcoming layers are the ones that have not been visited and allocated to any of the existing layers. The risk of the generalized constraint is that in the search for the direct neighbours of the vertices of layer one, the vertex on layer zero, and the other neighbours located on layer one are also in the set of neighbours. These are, however, already placed onto a layer and should not be reallocated to another layer. Still, no additional term or constraint needs to be formulated. The first constraint of this problem guarantees that a vertex cannot be on multiple layers, which solves this issue. In order to find the direct neighbours of the upcoming layers recursively, the following constraint is applied:

\\Constraint 3: The direct neighbours of the upcoming layer are:

$$\forall v_1, v_2 \in V, \forall e \in E, \forall l \in L \quad b^e_{v_1,l_i} + b^e_{v_2} \leq 1 + b_{v_2,l_{i+1}} \tag{3.55}$$

The diagonal neighbours of a vertex in layer l_i are all the vertices that are incident to two direct neighbours of this vertex. The two direct neighbours and the diagonal neighbour are all located on layer $l_{\mathrm{i}+1}$. So if there exists a vertex which is incident to two direct neighbours on a layer l, then this vertex is part of layer l as well. In other words:

$$\forall v_1, v_2, v_3 \in V, \forall e_1, e_2 \in E, \forall l \in L \quad if \quad b^{e_1}_{v_1,l_i} = 1 \quad and \quad b^{e_2}_{v_2,l_i} = 1 \tag{3.56}$$

$$and \quad b^{e_1}_{v_3} = 1 \quad and \quad b^{e_2}_{v_3} = 1, \quad then \quad b_{v_3,l_i} = 1 \tag{3.57}$$

Using the same transformations as applied for the direct neighbours, the constraint for finding the diagonal neighbours on all layers recursively would be:
\\Constraint 4: The diagonal neighbours of the upcoming layer are:

$$\forall v_1, v_2, v_3 \in V, \forall e_1, e_2 \in E, \forall l \in L \quad b^{e_1}_{v_1,l_i} + b^{e_2}_{v_2,l_i} + b^{e_1}_{v_3} + b^{e_2}_{v_3} \leq 3 + b_{v_3,l_i} \tag{3.58}$$

These constraints are tasked with building the layers of the layering model by finding all the direct and diagonal neighbours in each layer. All vertices are located in exactly one layer. There is no need to add an optimization target for this subproblem, because the number of layers shall not be minimized or maximized, but calculated. In this sense, this subproblem is different from the vertex coloring problem, where the number of layers should be minimized.

MIP Model of Finding the Sending Path

Finding the sending path of the layering model is the second subproblem that can be solved with the help of mixed integer linear programming. There needs to be only one sending path for the layering model. This path starts at the sending vertex (s) and ends at a vertex in the last found layer. Due to the constraints that were presented for the sending path in Subsec. 3.2.2, this sending path is either horizontal or vertical to the sending vertex (s). This means that the candidates for the last vertex of the sending path are:

1. The vertex with the same x-coordinate value and a higher y-coordinate value compared to s.
2. The vertex with the same x-coordinate value and a lower y-coordinate value compared to s.
3. The vertex with the same y-coordinate value and a higher x-coordinate value compared to s.
4. The vertex with the same y-coordinate value and a lower x-coordinate value compared to s.

The objective is to find a path starting from s and ending at one of these target vertices. There could be multiple candidates from this group of possible target vertices that satisfy the requirements for constructing a sending path, but only one will be chosen at first. Finding such a path resembles the 1-source-1-target problem. However, the constraints must be modified accordingly. Let s be the starting vertex and t be the target vertex on the last layer. The binary variable b_k represents whether the k-th directed edge is part of the path or not.

\\Constraint 1: The number of outgoing edges from the starting vertex is equal to one, and there are no incoming edges.

$$\sum_{e_k \in E_{s,out}} b_k = 1 \tag{3.59}$$

$$\sum_{e_k \in E_{s,in}} b_k = 0 \tag{3.60}$$

\\Constraint 2: The number of incoming edges of the target vertex is equal to one, and there are no outgoing edges.

$$\sum_{e_k \in E_{t,in}} b_k = 1 \tag{3.61}$$

$$\sum_{e_k \in E_{t,out}} b_k = 0 \tag{3.62}$$

\\Constraint 3: The number of incoming and outgoing edges of all vertices that are neither a starting vertex nor a target vertex in the path should be equal to one another.

$$\forall v_i \in V \backslash \{s,t\} \quad \sum_{e_k \in E_{i,in}} b_k = \sum_{e_k \in E_{i,out}} b_k \tag{3.63}$$

These constraints do not exclude the possibility of visiting a vertex on the same layer, which is an important condition for the sending path. However, since the sending path is known to be a vertical or horizontal path starting from the sending vertex, it is possible to ensure that all requirements for the construction of the sending path are fulfilled. This can be done by solving the optimization problem:

\\Optimization objective: Minimize this path being found.

$$min \sum_{e_k \in E'} b_k * \alpha_k \tag{3.64}$$

Since this is a mesh network, the shortest path from the sending vertex to the target vertex is either a horizontal or vertical path. If this path contained a turn, then the length of the path would be greater than that of a straight path. The difference between this model and the one represented until this point for the 1-source-1-target problem is that it should save the best solutions (candidates), which may be useful after running the third model. This is done by the MILP solver.

MIP Model of Finding the Paths from the Vertex in the Sending Path to the Receivers

As mentioned in Subsec. 3.2.1, there are two types of layers: full layers and non-full layers. If the layer is full, then the subproblem is similar to a planar routing problem, with the modification that the sending vertex is the same for both paths that need to be found. For this model, the focus is on the specific full layer. This layer consists of the sending vertex, two receivers, and all the intermediate vertices that form a ring network. Before presenting the constraints of this model, it is essential to first define the subgraph that the model will consider. Let L_{i} be a subgraph of the original graph $G = (V; E)$, such that:

$$VL_i : \{v \in V \quad | \quad b_{v,l_i} = 1\} \tag{3.65}$$

$$EL_i : \{(v_1, v_2) \in E \quad | \quad v_1 \in VL_i \quad \wedge \quad v_2 \in VL_i\} \tag{3.66}$$

$$L_i = (VL_i, EL_i) \tag{3.67}$$

In order to form this subgraph, it is important that the vertices are assigned to their corresponding layers using the model of building the layers, which is similar to the vertex coloring problem. L_i represents the full layer where the model should be running. For each pair of vertices (i, j) and for each path $k \in \{1, 2\}$, two binary variables of the form ${b_{(\mathrm{i,j})}}^k$ and ${b_{(\mathrm{j,i})}}^k$ are introduced to represent whether the edge in direction (i, j) or (j, i) are in the path k or not. The two paths in the set k represent the clockwise and the counterclockwise paths. The constraints of this model are the following:

\\Constraint 1: Each non-directed edge is visited exactly once.

$$\forall i, j \in VL_i \wedge (i, j) \in EL_i \quad \sum_{1 \leq k \leq 2} ({b_{(\mathrm{i,j})}}^k + {b_{(\mathrm{j,i})}}^k) \leq 1 \tag{3.68}$$

\\Constraint 2: A vertex is visited at most once.

$$\forall v \in VL_i \quad \sum_{1 \leq k \leq 2} \sum_{(x,v) \in EL_i} {b_{(\mathrm{x,v})}}^k \leq 1 \tag{3.69}$$

\\Constraint 3: For all the vertices that are neither the starting vertex nor the target vertex of any of the paths, whenever the path enters the vertex, it must also leave the vertex.

$$\forall v \in VL_i \backslash \{c_1, c_2, r_1, r_2\}, \forall k \in \{1,2\} \quad \sum_{(x,v)\in EL_i} b_{(\mathrm{x,v})}{}^k = \sum_{(v,x)\in EL_i} b_{(\mathrm{v,x})}{}^k \tag{3.70}$$

\\Constraint 4: The sending vertex s has one outgoing edge in each path and no incoming edge.

$$\forall k \in \{1,2\}, \forall j \in VL_i \quad \sum_{(s,j)} b^k_{(s,j)} = 1 \tag{3.71}$$

$$\forall k \in \{1,2\}, \forall j \in VL_i \quad \sum_{(j,s)} b^k_{(j,s)} = 0 \tag{3.72}$$

\\Constraint 5: Each target has one incoming edge in the corresponding path and no outgoing edge in either of the paths.

$$\forall i \in VL_i \quad \sum_{(i,r_1)} b_{(\mathrm{i,r_1})}{}^1 = 1 \tag{3.73}$$

$$\forall i \in VL_i \quad \sum_{1\le k\le 2} \sum_{(r_1,i)} b_{(\mathrm{r_1,i})}{}^k = 0 \tag{3.74}$$

$$\forall i \in VL_i \quad \sum_{(i,r_2)} b_{(\mathrm{i,r_2})}{}^2 = 1 \tag{3.75}$$

$$\forall i \in VL_i \quad \sum_{1\le k\le 2} \sum_{(r_2,i)} b_{(\mathrm{r_2,i})}{}^k = 0 \tag{3.76}$$

$$\tag{3.77}$$

\\Optimization objective: Minimize the lengths of the found paths.

$$min \sum_{e\in EL_i} \sum_{1\le k\le 2} b_{\mathrm{e}}{}^k * w_{\mathrm{e}} \tag{3.78}$$

The term w_{e} corresponds to the weight of the edge. This is assumed to be equal to one for all edges throughout the thesis.

This model can also be used for a non-full layer, but without a guarantee that it would succeed in finding two paths starting at the sending vertex and each ending at one of the receivers. If the structure of the sending path or the locations of the receivers would make it impossible to reach two receivers in a non-full layer, then this subproblem transforms into a 1-source-1-target problem and can be solved using a similar model to

the one shown in Subsec. 3.2.4 with the only change, that the subgraph considered is L_i instead of the graph G. However, if the choice of a new sending vertex could possibly reach the two receivers of a non-full layer, then this sending vertex should be taken into consideration. In the model of the sending path, this new candidate is chosen, and this also affects the model for finding the path to the receivers because the sending vertex is changed.

3.2.5 Adaptation of the Layering Model to a Ring Network

Adapting the layering model to the ring network is not difficult, since the idea of the layering model is to treat the mesh network as a series of ring networks built onto one another. In this case, the problem does not require building the layers or even finding the sending path. The sending vertex is, by definition, located on the same layer as all the other vertices. Placing two receivers is always possible if the ring network is similar to a full layer. The only difference between the full layers of the layering model in a mesh network and in a ring network is the total number of vertices inside a layer. In the mesh network, depending on the layer l_i, the total number of vertices should be $8 * i$ for the layer to be considered as a full layer. However, a ring network may have seven vertices in total and still be a full layer in principle. In this case, it is sufficient to find a cycle for this graph. If a cycle exists, then two receivers can be placed onto the network without causing any problems. Reaching these two receivers would correspond to the planar routing problem, where both paths need to start at the sending vertex and end at the two receivers using the clockwise and counterclockwise directions. The model for solving this problem can be taken from the planar routing problem in the layering model, with the change that the graph under inspection is no longer L_i, but the original graph G representing the ring network in this case.

4 Implementation

The two strategies to apply multicast on a given 4×4 mesh network can also be implemented using the Python programming language [Pyt23], as well as an MIP solver, which for this thesis is Gurobi [Gur24]. The code in this thesis was developed with the help of ChatGPT, an AI tool by OpenAI [Ope25]. For both approaches, some subproblems will be solved by means of graph algorithms, while the others can be solved with an MIP model, as shown in Ch. 3.

4.1 Bordered Areas Approach

In the bordered areas approach, there are three subproblems to solve:

1. Finding the borders based on the starting vertex.
2. Finding the closest vertex of the border to the receiver in the area.
3. Finding the path from this vertex to the receiver or finding the paths from both borders to both receivers in the area (if two receivers are located in the same area).

The first subproblem is a typical case of applying functions in Python instead of an MIP model. Finding the vertices of the western border, for instance, would be done with the following function:

Algorithm 1 Find Western Border Vertices

Require: Starting vertex (x_s, y_s), set of all vertices V
Ensure: Western border vertices
FindWesternBorderVertices$(x_s, y_s), V$
1: **return** $\{v \in V \mid y_v = y_s \wedge x_v \leq x_s\}$

Setting $x_v \leq x_s$ means that the starting vertex (s) is also put in the border. In order to find the other three borders, the mathematical signs in line 1 would need to be changed accordingly. The two other subproblems are solved using an MIP model. Building an MIP model consists of three steps, which are:

1. Declare the type and number of variables that are required for the model. For instance, it is possible to declare $|V|$ binary variables b_v that represent whether vertex v is chosen for a specific task or not.

2. Formulate the constraints. These typically consist of a sum of variables, that should be $=$, $\leq$, or $\geq$ compared to another variable, sum of variables, or a fixed value.

3. Optionally, an optimization constraint can be set for the model. For instance, if the model is tasked with finding the shortest path from a source vertex to a target vertex, then the optimization constraint must be active, and it should minimize the number of edges selected for this particular path.

As mentioned in Subsec. 3.1.4, the MIP models that are required for the bordered areas approach are: the model of the Manhattan distance, the model of the one-source-one-target problem, and the model of the variation of planar routing. The Manhattan distance needs to be calculated with the input of the receiver (located in one of the areas) and the vertices of the corresponding border. The output of the model is the border vertex located closest to the receiver in the area. The required variables (type and number) are declared on lines 2 and 3 of the code. The constraints are located on lines 4,6, and 10, while the optimization objective is set on line 15. The constant M on lines 7, 8, 11, and 12 is set to be equal to 1000, making it possible to activate either one of the cases or the other. A pseudocode representation of this model is shown in the following algorithm:

Algorithm 2 Find Closest Border Vertex – Manhattan Distance

Require: Set of border vertices $B = \{(x_i, y_i)\}$, receiver vertex $r = (x_r, y_r)$
Ensure: Vertex in B closest to r by Manhattan distance
1: Initialize optimization model
2: Define binary variables b_i for each vertex $i \in B$
3: Define variables $xdiff_i, ydiff_i$ for each vertex $i \in B$
4: Add constraint: $\sum_i b_i = 1$ {Exactly one vertex selected}
5: **for** each vertex i in B **do**
6: Add constraints for $xdiff_i$ to model absolute difference in x:
7: $xdiff_i \geq x_i - x_r - M \times (1 - b_i)$
8: $xdiff_i \geq x_r - x_i - M \times (1 - b_i)$
9: $xdiff_i \leq M \times b_i$
10: Add constraints for $ydiff_i$ to model absolute difference in y:
11: $ydiff_i \geq y_i - y_r - M \times (1 - b_i)$
12: $ydiff_i \geq y_r - y_i - M \times (1 - b_i)$
13: $ydiff_i \leq M \times b_i$
14: **end for**
15: Set objective: minimize $\sum_i (xdiff_i + ydiff_i)$
16: Solve optimization model
17: **for** each vertex i in B **do**
18: **if** $b_i = 1$ **then**
19: **return** vertex i
20: **end if**
21: **end for**
22: **return** None

The second subproblem for which an MIP model can be constructed is the one-source-one-target problem. The requirements of this model are the subgraph where the path should be found, the source vertex, and the target vertex. The output of the model is the shortest path in the declared subgraph from the source vertex to the target vertex. The binary variables $b_{(u,v)}$ represent whether the edge from vertex u to vertex v is located on the path or not. Since both directions need to be checked, $2 * |E|$ such variables need to be declared. The constraint for the sending model is set on line 8, which states that the number of outgoing edges should be one more than the number of incoming edges. Similarly, on line 10, the constraint for the target vertex is placed, which states that the number of incoming edges should be one more than the number of outgoing edges. Lastly, on line 12, the constraint for all the intermediate vertices can be found, which states that the number of incoming and outgoing edges should be

equal to one another. The optimization objective is also set on line 3. For this particular example, this is necessary because otherwise, the path found might also include loops, which are not desired. A pseudocode representation of this model is the following:

Algorithm 3 Find Shortest Path from Source to Target – One_Source_One_Target

Require: Directed subgraph G, source node s, target node t
Ensure: Shortest path from s to t
1: Initialize optimization model
2: Define binary variables b_{uv} for each edge $(u, v) \in G$
3: Set objective: minimize $\sum b_{uv}$
4: **for** each node $n \in G$ **do**
5: Compute incoming_edges: $\sum b_{uv}$ where $v = n$
6: Compute outgoing_edges: $\sum b_{uv}$ where $u = n$
7: **if** $n = s$ **then**
8: Add constraint: outgoing_edges $-$ incoming_edges $= 1$
9: **else if** $n = t$ **then**
10: Add constraint: outgoing_edges $-$ incoming_edges $= -1$
11: **else**
12: Add constraint: outgoing_edges $-$ incoming_edges $= 0$
13: **end if**
14: **end for**
15: Solve the model
16: **if** solution is optimal **then**
17: Extract edges where $b_{uv} = 1$
18: Reconstruct path from s to t
19: **return** path
20: **else**
21: **return** None
22: **end if**

Lastly, the variation of planar routing that is applied for the bordered areas approach deals with the case when there are two different senders and each of these has to send a message to one of the two receivers located in the area. This model requires both the border vertices and receivers in the area as inputs, as well as the subgraph where these paths need to be found. In this case, the subgraph will consist of the area itself and both of its surrounding borders. The output of the model should be the two paths that do not overlap with one another. For simplicity reasons, these two paths are obtained by solving the one-source-one-target problem, and all of the constraints in the following pseudocode on lines 9, 11, 14, and 18 are used to ensure that there is no

overlap between the two paths on any vertex. The optimization objective is set on line 20. The binary variables b_n^k represent wether the vertex n is located on path $k \in \{1, 2\}$ or not.

Algorithm 4 Planar Routing

Require: Directed Subgraph G, pairs (s_1, t_1) and (s_2, t_2)
Ensure: Disjoint paths P_1 and P_2
1: $P_1 \leftarrow$ one_source_one_target(G, s_1, t_1)
2: Reserve nodes in P_1
3: $G' \leftarrow G$ without reserved nodes
4: $P_2 \leftarrow$ one_source_one_target(G', s_2, t_2)
5: Initialize optimization model
6: Define binary variables b_n^1 and b_n^2 for all $n \in G$
7: **for all** $n \in G$ **do**
8: **if** $n \in P_1$ **then**
9: Add constraint: $b_n^1 = 1$
10: **else**
11: Add constraint: $b_n^1 = 0$
12: **end if**
13: **if** $n \in P_2$ **then**
14: Add constraint: $b_n^2 = 1$
15: **else**
16: Add constraint: $b_n^2 = 0$
17: **end if**
18: Add constraint: $b_n^1 + b_n^2 \leq 1$
19: **end for**
20: Set objective: minimize total path weight using b_n^1 and b_n^2
21: Solve model
22: **return** Nodes with $b_n^1 = 1$ and $b_n^2 = 1$

4.2 Layering Model

For the layering model, the whole problem can also be divided into three subproblems:

1. Finding the layers depending on the sending vertex.
2. Finding the sending path based on the sending vertex and the last found layer.
3. Finding the paths on a layer from the vertex on the sending path to the receiver(s) located on the layer.

All of these subproblems can be solved using the models presented in Ch. 2. The first subproblem of finding the layers is similar to the vertex coloring problem:

Algorithm 5 Assign First Layer

Require: Vertices V, edges E, start vertex s
Ensure: Binary layer assignment $b_{v,\ell}$ for $\ell \in \{0,1\}$
for all $v \in V$, $\ell \in \{0,1\}$ **do**
  define binary variable $b_{v,\ell}$
end for
for all $v \in V$ **do**
  add constraint: $b_{v,0} + b_{v,1} = 1$
end for
add constraint: $b_{s,0} = 1$
$N \leftarrow$ neighbours of s in E
for all $n \in N$ **do**
  add constraint: $b_{n,1} = 1$
end for
for all $v \in V \setminus (\{s\} \cup N)$ **do**
  $C \leftarrow$ neighbours of v that are in N
  if $|C| \geq 2$ **then**
    add constraint: $b_{v,1} = 1$
  end if
end for
solve model
return layer assignment from $b_{v,\ell}$

This model requires the sets of vertices and edges of the graph as inputs, as well as the sending vertex. It assigns all of the direct neighbours and diagonal neighbours of the sending vertex to layer one, without performing an optimization. The required

binary variables are $b_{v,l}$ that represent whether vertex v is on layer l or not. Each vertex will be assigned to exactly one layer, which is the constraint presented on line 5 (in this case, the only layers observed are zero and one). On line 7, it is ensured that the sending vertex is always located on layer zero. Lines 9-12 serve to place the direct neighbours of the sending vertex at the first layer, while lines 12-17 ensure that the diagonal neighbours of the sending vertex are also placed on layer one. Recursively, the algorithm can be reapplied to find layers two, three, and so on. The only modification is that the vertices in the former layers should be excluded from the search. In other words, to find the second layer, the sending vertex and its incident edges need to be removed from the graph. Otherwise, the sending vertex would be considered to be a direct neighbour of the vertices on layer one, which would place it on layer two.

The second subproblem of finding the sending path can be solved by using the model of the one-source-one-target problem presented in Sec. 4.1. In this case, the original graph will be taken into consideration, with the known sending vertex and a target vertex located on the last found layer of this approach. Since there may be different possible sending paths, all the solutions will be saved and checked after running the planar routing model. Apart from finding the sending path, the model of the one-source-one-target problem can also be used in non-full layers to reach the closest receiver on the particular layer from the vertex of the sending path. In this scenario, the subgraph that is taken into consideration contains all the vertices and edges of the non-full layer.

Lastly, for full layers and sometimes non-full layers that satisfy the conditions mentioned in Sec.3.2.3, another variation of planar routing needs to be modeled. In this case, two paths sharing the same source need to be found. Each path ends at one of the two receivers located on the specific layer, which is the observed subgraph for this model. The source is a vertex that is shared between the sets of the sending path vertices and the current layer vertices. The required variables for this model are the binary $b_{e,k}$ variables. If the value is equal to one, then the edge e is part of path $k \in \{1, 2\}$, otherwise it is not. In total, there should be $2 * |E|$ for each $k \in \{1, 2\}$ binary variables, since the graph will be reconstructed as a directed graph. The constraint regarding the number of incoming and outgoing edges of the source is found on line 8, while for the target vertices it is found on line 10. Line 12 presents the constraint regarding the intermediate vertices of both paths. Each edge is either a part of one path, the other path, or neither of them, as mentioned on line 17. Furthermore, the first target vertex (receiver one) cannot be visited with the second path (path from the source to target two / receiver two) or the other way around. This is shown by the constraints on lines 20 and 23. The optimization objective of minimizing the length of both paths by having as few edges as possible for both paths is presented on line 25, and lastly, both paths are extracted as shown on the last two lines. All these steps are presented in the

following pseudocode:

Algorithm 6 Planar Routing from One Source to Two Targets

Require: Graph $G = (V, E)$, source s, targets t_1, t_2
Ensure: Two disjoint paths from s to t_1 and s to t_2

make G bidirectional
for all $(u, v) \in E \times \{1, 2\}$ **do**
  define binary variable $b_{e,k}$
end for
for all $k \in \{1, 2\}$ **do**
  for all $v \in V$ **do**
    if $v = s$ **then**
      outgoing_edges−incoming_edges= 1
    else if $v = t_k$ **then**
      incoming_edges−outgoing_edges= 1
    else
      incoming_edges=outgoing_edges
    end if
  end for
end for
for all $v \in V \setminus \{s\}$ **do**
  prevent node overlap: $\sum b_{(u,v),1} + b_{(u,v),2} \leq 1$
end for
for all edges e into t_2 **do**
  $b_{e,1} = 0$
end for
for all edges e into t_1 **do**
  $b_{e,2} = 0$
end for
minimize total $\sum b_{e,k}$
solve model and extract paths
return path from s to t_1, and from s to t_2

Depending on the choice of the sending path, the results of this model may differ. For all possible sending paths, these results are saved to make it possible to choose the best candidate as the sending path in the formerly introduced model of the one-source-one-target problem. The best candidate is the one that connects the sending vertex to the most receivers for the laying model.

4.3 Experimental Results

Both of these approaches were tested on a 4×4 mesh network and delivered the expected results as detailed on Ch. 3. For these examples, the sending vertex is always the one located at position (2,2). Two different scenarios are observed for the bordered areas approach:

1. There is at most one receiver on each area or its corresponding border.
2. There are two receivers in a single area.

The results of scenario one are presented in Figure 4.1.

```
The starting vertex is: (2, 2)
Western border: [(0, 2), (1, 2), (2, 2)]
Eastern border: [(2, 2), (3, 2)]
Northern border: [(2, 2), (2, 3)]
Southern border: [(2, 0), (2, 1), (2, 2)]
Area 1: [(3, 3)]
Area 2: [(0, 3), (1, 3)]
Area 3: [(0, 0), (1, 0), (0, 1), (1, 1)]
Area 4: [(3, 0), (3, 1)]
The receivers are: [(0, 0), (3, 1), (1, 3), (3, 2)]
Area 1 has 1 receiver(s)
Area 2 has 1 receiver(s)
Area 3 has 1 receiver(s)
Area 4 has 1 receiver(s)
Closest west border vertex to receiver (0, 0): (0, 2) in area 3
Path from the sending vertex (2, 2) to the vertex on the western border closest to the receiver is [(2, 2), (1, 2), (0, 2)]
Path from the closest border vertex to the receiver (0, 0) is [(0, 2), (0, 1), (0, 0)]
Closest south border vertex to receiver (3, 1): (2, 1) in area 4
Path from the sending vertex (2, 2) to the vertex on the southern border closest to the receiver is [(2, 2), (2, 1)]
Path from the closest border vertex to the receiver (3, 1) is [(2, 1), (3, 1)]
Closest north border vertex to receiver (1, 3): (2, 3) in area 2
Path from the sending vertex (2, 2) to the vertex on the northern border closest to the receiver is [(2, 2), (2, 3)]
Path from the closest border vertex to the receiver (1, 3) is [(2, 3), (1, 3)]
Receiever (3, 2) is in the eastern border.
The shortest path is: [(2, 2), (3, 2)]
```

Figure 4.1: The Python output of the bordered areas approach with at most one receiver for each area.

For the bordered areas approach, the first step would be to find the borders based on the starting vertex. The vertices that are contained in these sets are found directly underneath the starting vertex. With the known borders, the areas of the approach are defined as shown on the following four lines. As mentioned, in this example, there is at most one receiver located on each area or its corresponding border. The first receiver (located at position (0,0)) is part of the third area, and it is the first one that is found and

taken into consideration. Using the right border priority, the third area is bound to the western border (WB), as shown in Table 3.2. Along this border, the closest vertex to the receiver is the vertex located at position (0,2). This is calculated using the Manhattan distance model. After this, the one-source-one-target model is run twice. At first, it is used to find the path from the sending vertex (located on position (2,2)) to the closest border vertex (located on position (0,2)). The second time, it is used to find the path from the closest border vertex (located on position (0,2)) to the receiver (located on position (0,0)) in this area. The same process is repeated for the receivers located in areas two and four. The last receiver, namely located on position (3,2), is part of the eastern border (EB). The results from Table 3.2 show that the first area is bound to this border using the right border priority, so this is the only receiver found at area one and in this case the one-source-one-target model needs to be called only once to find the path from the sending vertex (located on position (2,2)) to this receiver (located on position (3,2)). These results are illustrated in Figure 4.2:

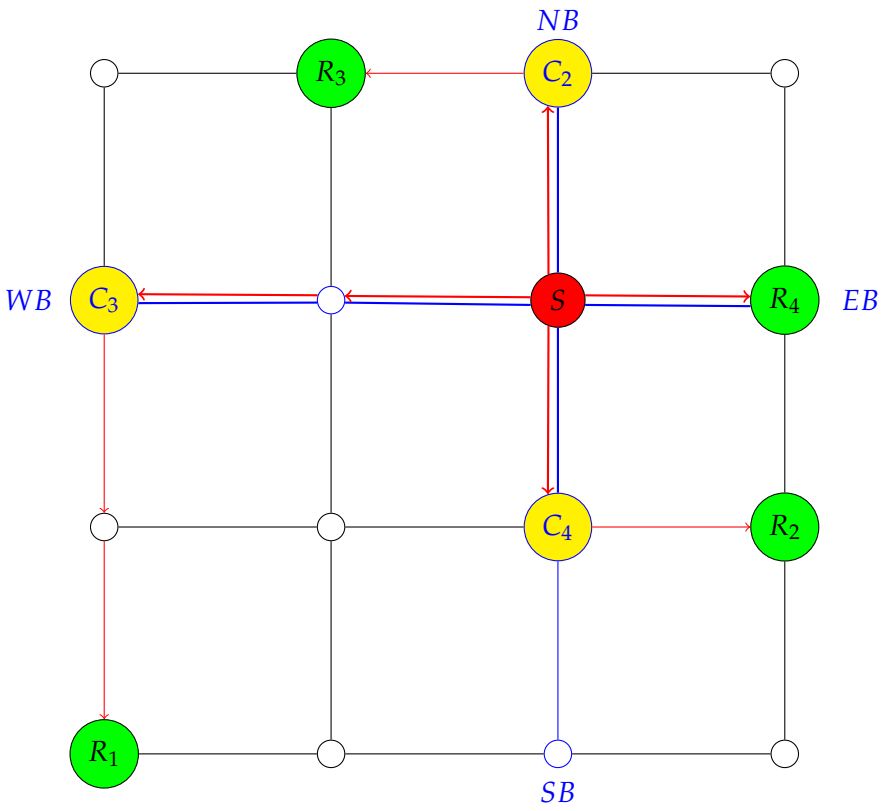


Figure 4.2: Graph representation of the output in Figure 4.1.

The starting vertex is colored red, while all the receivers are colored green. The yellow vertices represent the closest border vertex to the receiver in the respective area. For instance C_3 represents the vertex of the western border (WB) closes to receiver R_1 located on area A_3. The red colored edges represent the paths taken to reach the receivers based on the Python output shown in Figure 4.1.

The results of scenario two are presented in Figure 4.3:

```
The starting vertex is: (2, 2)
Western border: [(0, 2), (1, 2), (2, 2)]
Eastern border: [(2, 2), (3, 2)]
Northern border: [(2, 2), (2, 3)]
Southern border: [(2, 0), (2, 1), (2, 2)]
Area 1: [(3, 3)]
Area 2: [(0, 3), (1, 3)]
Area 3: [(0, 0), (1, 0), (0, 1), (1, 1)]
Area 4: [(3, 0), (3, 1)]
The receivers are: [(0, 0), (1, 0)]
Area 1 has 0 receiver(s)
Area 2 has 0 receiver(s)
Area 3 has 2 receiver(s)
Area 4 has 0 receiver(s)
Area 3 is under inspection.
Receiver: (0, 0) and Receiver: (1, 0)
Pairs of the border vertices closest to the receivers: [((0, 0), (0, 2)), ((1, 0), (2, 0))]
Path 1 planar routing from source (0, 0) to target (0, 2): [(0, 0), (0, 1), (0, 2)]
Path 2 planar routing from source (1, 0) to target (2, 0): [(1, 0), (2, 0)]
```

Figure 4.3: The Python output of the bordered areas approach with two receivers in a single area.

Since the graph and starting vertex are the same, the results for the borders and areas will be the same. The only difference in this scenario is the receivers R_1 and R_2 at positions (0,0) and (1,1) respectively, which are both located on area A_3. For both of these receivers, a choice is made regarding the surrounding borders (in this case, WB and SB). While the first receiver has an equal distance to both borders, the second receiver, namely the one at position (1,0), is closer to the southern border (SB). That is the reason why this receiver is paired with the southern border (SB), while the first one is paired with the western border (WB). The respective closest border vertices are C_{3,R_1} located on the western border (WB) at position (0,2) and C_{3,R_2} located on the southern border (SB) at position (2,0). The planar routing model is used to find both the paths and to avoid any overlap between the two, as shown in Figure 4.4

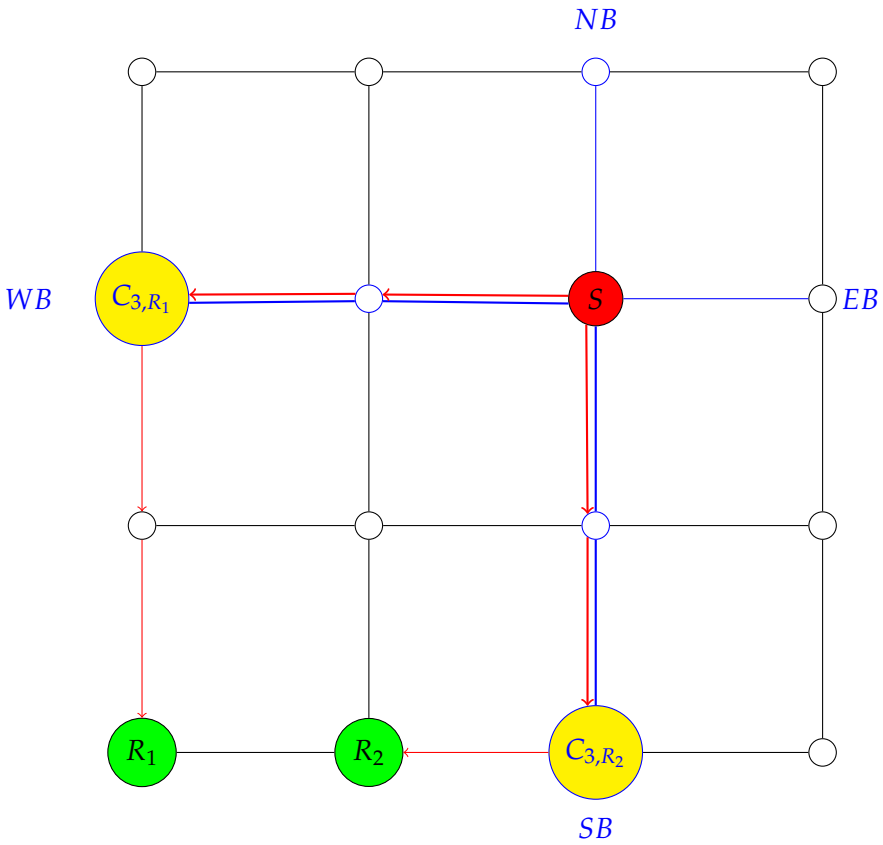


Figure 4.4: Graph representation of the output in Figure 4.3.

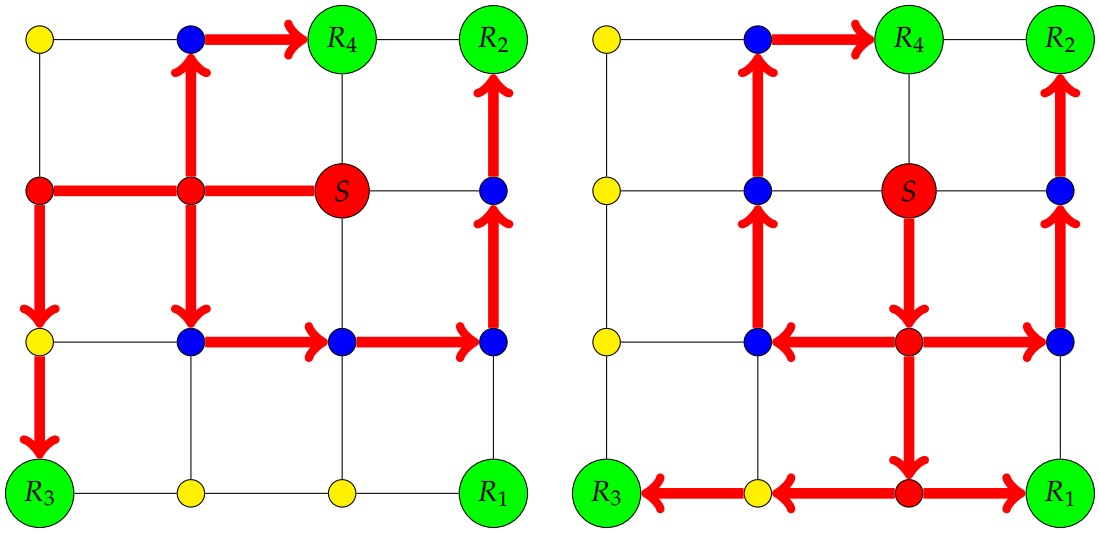
For the layering model, the same graph and starting vertex will be observed. This vertex represents layer zero of the layering model. In this case, two other layers can be found, which are layers one and two. Layer one consists of all direct and diagonal neighbours of the sending vertex, while layer two consists of all the direct and diagonal neighbours of all the vertices in layer one. The first layer is a full layer, while the second layer is a non-full layer. Two receivers are placed on each layer. It is possible to find two sending paths with these conditions, and for both candidates, the outcomes are presented. The Python output of this approach is shown in Figure 4.5:

```
The starting vertex (Layer 0) is:  (2, 2)
Vertices in layer 1: [(1, 1), (2, 1), (3, 1), (1, 2), (3, 2), (1, 3), (2, 3), (3, 3)]
Layer 1 is a full layer.
Vertices in layer 2: [(0, 1), (0, 2), (1, 0), (0, 0), (0, 3), (2, 0), (3, 0)]
Layer 2 is a non-full layer.
The receivers are: [(3, 0), (3, 3), (0, 0), (2, 3)]
Candidate vertices in layer 2: [(0, 2), (2, 0)]

--- Processing candidate (0, 2) ---
Sending path to (0, 2): [(2, 2), (1, 2), (0, 2)]
Start vertex in layer one: (1, 2)
Start vertex in layer two: (0, 2)
Receiver (3, 0) is in layer 2
Receiver (3, 3) is in layer 1
Receiver (0, 0) is in layer 2
Receiver (2, 3) is in layer 1
Layer 1 has 2 receivers: [(3, 3), (2, 3)]
Planar paths on layer 1: [(1, 2), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3)] and [(1, 2), (1, 3), (2, 3)]
Layer 2 has 2 receivers: [(3, 0), (0, 0)]
No solution found
Planar paths on layer 2: None and None
The only path found in this non-full layer is : [(0, 2), (0, 1), (0, 0)]

--- Processing candidate (2, 0) ---
Sending path to (2, 0): [(2, 2), (2, 1), (2, 0)]
Start vertex in layer one: (2, 1)
Start vertex in layer two: (2, 0)
Receiver (3, 0) is in layer 2
Receiver (3, 3) is in layer 1
Receiver (0, 0) is in layer 2
Receiver (2, 3) is in layer 1
Layer 1 has 2 receivers: [(3, 3), (2, 3)]
Planar paths on layer 1: [(2, 1), (3, 1), (3, 2), (3, 3)] and [(2, 1), (1, 1), (1, 2), (1, 3), (2, 3)]
Layer 2 has 2 receivers: [(3, 0), (0, 0)]
Planar paths on layer 2: [(2, 0), (3, 0)] and [(2, 0), (1, 0), (0, 0)]
```

Figure 4.5: The Python output of the layering model.

The first sending path candidate is $(2,2)--(1,2)--(0,2)$. The vertex located at position (1,2) will act as the sending vertex for the first layer, while the vertex located at position (0,2) will act as the sending vertex for the second layer. Since the first layer is a full layer, it is always possible to find two paths in order to reach both receivers on the layer using the MIP model for planar routing. The main issue with this candidate occurs in the second layer, where the MIP model for planar routing fails to deliver two paths that do not overlap with one another. In this case, the MIP model of the one-source-one-target problem is used to find the path ending at the closest receiver. In order to reach the furthest receiver in this construction, the path would need to pass through the closest receiver, which is not allowed. Hence, the model of planar routing delivers no solution. On the other hand, the second sending path candidate, $(2,2)--(2,1)--(2,0)$, offers a better solution, where all the receivers are reached.

The new location of the vertex in the sending path, which is also part of the second layer, makes it possible to use both the clockwise and counterclockwise directions to reach both receivers in the second (non-full) layer. These results are illustrated in the two mesh networks in Figure 4.6:



(a) First candidate for the sending path.

(b) Second candidate for the sending path.

Figure 4.6: Sending path candidates based on the output in Figure 4.5.

As shown in Figure 4.6, the two sending paths have no problem in reaching the receivers R_2 and R_4 located on the first layer, which is a full layer. The total number of hops in both cases is equal, and it adds up to seven. The problem, as mentioned, is at a non-full layer. The second candidate has its vertex of the sending path located in such a position in layer two, from which it is possible to find two non-overlapping paths leading to receivers R_1 and R_3 in the counterclockwise and clockwise directions, respectively. On the other hand, the first candidate does not offer the possibility of reaching both receivers R_1 and R_3. In this case, there is no path that starts from the vertex of the sending path in the second layer that reaches receiver R_1 without passing through receiver R_3. The only reachable receiver is therefore R_3 with the path shown in Figure 4.6a.

5 Conclusion

This thesis proposes two methods to perform multicast on a mesh network and how these can also be adapted for a ring network. While the bordered areas approach performed the split of the optical signals at the beginning of the process, the layering model delayed splitting the signal until the particular layer was reached. Both approaches can be applied on a 4×4 mesh network, and a given sender can reach at most four receivers in the best-case scenario. In the bordered areas approach, such a scenario would be placing one receiver in each of the areas, while for the layering model, it would be placing two receivers on two different layers. However, the possible locations of the receivers that will receive a message of the same wavelength from the sender are limited for both approaches. For instance, in the bordered areas approach, there cannot be three or more receivers located in the same area (including both surrounding borders). Placing two receivers in an area is always possible, but the expense is found in the remaining areas, where the number of receivers is limited to one (apart from the only area that does not share any border with the one where the two receivers are located). However, it is also not possible to have three or more receivers on the same layer in the layering model. In this case, even placing two receivers on a layer might be impossible if the layer is not full and the vertex on the sending path is closer to both receivers in the same direction (clockwise or counterclockwise). The maximum number of receivers in the bordered areas approach is equal to the degree of the sending vertex. For a 4×4 mesh network, the highest degree of a vertex is four. Hence, there are at most four receivers in the network. In the worst-case scenario, all of these four are placed in one single area of the approach. If this is the case, then only two of these can be reached. On the other hand, the maximum number of receivers for the layering model depends on the number of layers found (excluding layer zero, which contains only the sending vertex). In the worst-case scenario for a 4×4 mesh network, all the found layers are non-full layers, and the receivers are all placed on one single layer. The structure of the sending path may make it impossible to reach two receivers on this layer, leading to the whole approach to offer a single connection from the sending vertex to one receiver. In other words, the worst-case scenario of the layering model may lead to a trivial method of sending a message from a source to a single target, which can be done without using multicast at all. In this sense, the approach to border areas is safer since reaching at least two receivers is always guaranteed.

The bordered areas approach also performs better on smaller mesh networks, like the 2×2 and 3×3 networks. In the 2×2 mesh network, only two borders and one area can be found. Since all the vertices would have a degree equal to two, the maximum number of receivers in this graph is also set equal to two. Wherever these receivers are placed in this case, the bordered areas approach is able to find a path from the sending vertex to both of the receivers. On the same network, it is also possible to apply the layering model, but the results would differ from those of the bordered areas approach. Whichever vertex is the starting vertex, there will only be one upcoming layer for the layering model. This layer cannot be a full layer, since the first full layer of the layering model requires eight vertices to be considered a full layer. This network, however, only has a total of four vertices. Another setback of the layering model for this network is the structure of the sending path, which disables the vertex on the sending path and layer one from sending two signals on this layer. In this case, the sending path would only consist of one edge from the sending vertex to one of its direct neighbours, and the location of the single receiver would be limited to any of the two remaining vertices.

For a 3×3 mesh network, there is a single vertex that has a degree of four (Located in the middle of the network). For the bordered areas approach, four borders can be found (if this middle vertex is the sending vertex), and four areas can be found. In the best-case scenario, four receivers can be placed on the network (for instance, one in each area), and four paths from the sending vertex exist to these receivers. In the worst-case scenario, the sending vertex is one with a degree equal to two, and it is only possible to reach two receivers located in a single area or its surrounding borders. The central vertex of this network is also taken into consideration for the best-case scenario of the layering model. If this vertex is the sender, then the layer found on the next step of this approach is a full layer, where two receivers can be placed in total. In this scenario, it is possible to place the receivers wherever in the first layer and find a valid path from the vertex in the sending path (there are four possible sending paths in such a scenario) to both of the receivers using the clockwise and counterclockwise directions, respectively. In the worst-case scenario, the sending vertex is located on a corner, thus making it possible to find two layers, both of which are non-full layers. If the receivers are placed on the same layer, due to the structure of the sending path, it is impossible to find two paths, each starting at the sending vertex and finishing at the corresponding receiver. Only one of the receivers could be reached on such an occasion as well, similar to the other two networks observed earlier. All these results regarding the best-case (BC) and worst-case (WC) scenarios are presented in Table 5.1.

Table 5.1: Number of receivers reached for both methods in different networks.

Bordered Areas Approach			Layering Model		
Network	BC	WC	Network	BC	WC
2×2	2	2	2×2	1	1
3×3	4	2	3×3	2	1
4×4	4	2	4×4	4	1
$N \times N$	4	2	$N \times N$	$2 * (N - 2)$	1
Ring	2	2	Ring	2	2

In the case of a ring network, the methods are very similar to one another; hence, the results are the same in the best- and worst-case scenarios. The layering model performs worse than the bordered areas approach for smaller mesh networks. Typically, WRONoCs are more power efficient than the active NOCs for systems that consist of up to sixteen vertices. For a mesh network, this would correspond to a 4×4 network, which contains exactly sixteen vertices as seen in Figure 3.1. The preferred approach for these networks would be the bordered areas approach. However, if larger WRONoCs are considered, the layering model can perform better. The disadvantage of the bordered areas approach for larger networks is that the maximum number of receivers still depends on the highest degree of a vertex found in the network. Regardless of the total number of vertices in an N×N network, the highest vertex degree would still be equal to four. For the borderd areas approach, there are still four borders at maximum that can be found and four areas at maximum, which increase in size compared to the ones found for a 4×4 network. So, in the best-case scenario, it is possible to reach four different receivers in a large $N \times N$ network, and two receivers in the worst-case scenario. Compared to the bordered areas approach, the probability that the layering model is more effective rises with the growth of the network. For the layering model, in the best-case scenario, there are at most two receivers on a layer, so the maximum number of receivers depends on the number of found layers. The more vertices a network has, the more layers can be found, and thus, more receivers can be placed onto these layers. If N of the $N \times N$ network is an odd number, then there is a vertex in the middle of the network, which can be used to determine the best-case scenario. If the sending operation begins at this vertex, then the total number of layers that can be found is equal to $N - 2$ (same as in the 3×3 network, where only one layer can be found if the middle vertex is chosen as a sending vertex). In this case, all the layers are full and can contain two receivers, which leads to the total number of receivers to be equal to $2 * (N - 2)$. More concretely, in a 7×7 network, the bordered areas approach would be able to transmit a message with the same wavelength to four different receivers, while the layering model would make it possible for the sending

vertex to send a message with the same wavelength to ten vertices. The main problem with the layering model is always the choice of the sending vertex positioned on a corner, and all the found layers are non-full layers. If the receivers in this case are all placed inside a non-full layer and the structure of the sending path does not allow finding two paths that lead to two receivers on the same layer, then the sending vertex might communicate with a single receiver at a time, which defeats the purpose of applying multicast.

Apart from the worst-case scenario of the layering model, it also needs to be noted that it is very fragile. The case where all the layers are full is very rare, and in the larger networks, the probability of encountering the first non-full layer sooner rather than later is very high. As mentioned in Sec. 3.2, if a layer is determined to be non-full, then all the upcoming layers are non-full as well. The more non-full layers, the less likely it is to place two receivers on each of the upcoming layers. Also, the greedy method of choosing between the candidates for the sending path might lead to problems for the upcoming layers, as explained in Sec. 3.2.

5.1 Application

There are specific network topologies where applying the proposed multicast approaches in WRONoCs is advantageous. For example, the bordered areas approach is well-suited to network configurations in which a sender has one receiver located in each of the four diagonal directions: southwest, southeast, northeast, and northwest. This pattern matches the assumption of the bordered areas approach, where each defined area contains exactly one receiver.

This type of communication pattern is common in image processing applications, where different filters are applied to digital images to achieve effects such as deblurring. In such applications, each vertex in the optical network can represent a single pixel of the image. Applying a filter involves manipulating pixel values based on the weights defined in the filter matrix. These filters are typically represented as square matrices (for instance, 3×3 or 5×5), and the filtering operation corresponds to a mathematical procedure known as convolution.

Convolution involves sliding the filter matrix across the image and performing element-wise multiplication between the filter and the corresponding pixel values at each position, followed by summation. The result of the summation is then written back to the central pixel location in the image. In the context of a NoC-based implementation, the central vertex (representing the pixel where the filter is currently centered) must obtain values from neighbouring vertices to perform the convolution.

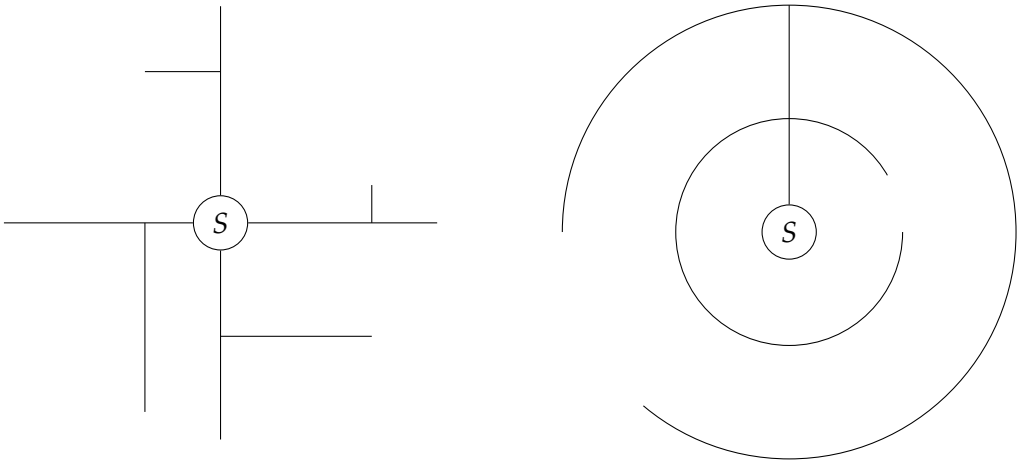
A typical multicast scenario arises when the central node sends signals only to those neighbouring nodes whose positions correspond to non-zero entries in the filter kernel, instructing them to send back their pixel values or the result of multiplying their pixel value by the kernel weight. These responses are then collected by the central node, which performs the final summation and updates its pixel value accordingly. This pattern aligns with the principle of multicast communication, where a single node simultaneously communicates with a selected subset of neighbours, which is supported by WRONoCs.

For example, when applying a Laplacian filter, the central vertex needs to notify and then obtain pixel values from its four direct neighbours (north, south, east, west), which fits the bordered areas approach. Alternatively, filters can be adapted to match the layering model, where the central vertex communicates with two receivers per layer.

Although these implementations are traditionally done using electrical NoCs [Zho+21], the same principles are applicable to optical NoCs (ONoCs). Nevertheless, many advanced image processing tasks today increasingly rely on optical neural networks (ONNs), which offer state-of-the-art performance for convolution operations.

5.2 Future Work

The two approaches proposed in this thesis, namely the bordered areas approach and the layering model, propose two different possibilities for applying multicast on a given mesh network. The main objective of this work is to use the same wavelength as often as possible for multiple communications between the sending vertex and multiple receivers. These two approaches deliver the paths that the optical signal takes starting from the sender and going through splitting operations or being coupled by MMRs until it reaches the receiver. These paths can be represented using these two wavefronts:



(a) Wavefront for the Bordered Areas Approach. (b) Wavefront for the Layering Model.

Figure 5.1: Wavefronts of both approaches.

As shown in Figure 5.1, the wavefront of the bordered areas approach consists of straight lines directed from the borders to the corresponding areas, while the wavefront of the layering model consists of circles representing the different layers of the approach. The blank spaces on the circle represent vertices where neither path goes through. The paths begin at the cross of the sending path (horizontal line over the starting vertex) with the corresponding path. From this point, the clockwise and counterclockwise directions are taken to reach the receivers in each layer. An interesting approach would be to somehow combine the bordered areas approach with the layering model to propose a new approach of applying multicast on a given mesh network. In the combined version, the bordered areas approach would be applied first. In the best-case scenario, the sending vertex has a degree of four, and there are four borders and four areas that can be found. In each area, instead of a single receiver, there will be multiple receivers. Inside the area, an algorithm will be applied to find the most central

vertex. This central vertex is the core of the layering model, which in this case will only be applied inside the particular areas obtained from the bordered areas approach. The central vertex has to be connected to the corresponding border by a path, which represents the sending path of the layering model. While in the layering model, the transmission of the optical signals starts at the sending vertex, in this combined version, the signals are transmitted along the sending path until this one reaches the most central vertex of the area. In this case, it would also be theoretically possible to place a receiver on the most central vertex of an area. The waveguide of this combined version is presented in Figure 5.2.

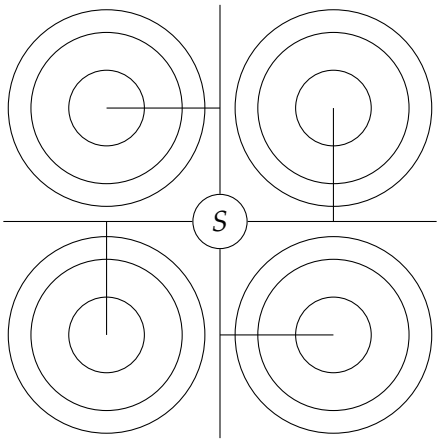


Figure 5.2: Combining the Bordered Areas Approach and the Layering Model.

It is recommended that this be performed on a large network to demonstrate the benefits of the layering model. If the network where this approach is being applied is an $N \times N$ mesh network, then four areas that can be found should theoretically be equivalent to four $(N-4) \times (N-4)$ networks, if N is chosen to be an odd number. Subtracting an even number from an odd number leads to another odd number, so for the layering model in each area, a central vertex exists. In one of the $(N-4) \times (N-4)$ networks, the layering model will deliver $(N-4)-2$ layers (or $N-6$ layers for simplicity). These are all full layers so that two receivers can be placed on each layer, which adds up to $2*(N-6)$ receivers. Since the central vertex of the area can be a receiver as well, the total number of receivers per area is equal to $2*(N-6)+1$. For the whole network, the possible number of receivers would therefore be $4*[2*(N-6)+1]$. More precisely, in the case of a 7×7 network, in the best-case scenario, the total number of receivers would be equal to twelve, which is nearly a quarter of the vertices in the whole network.

List of Figures

List of Tables

Bibliography

[Bar+08] A. R. Barron, A. Cohen, W. Dahmen, and R. A. DeVore. "Approximation and learning by greedy algorithms." en. In: *Ann. Stat.* 36.1 (Feb. 2008), pp. 64–94.

[Bré79] D. Brélaz. "New methods to color the vertices of a graph." In: *Commun. ACM* 22.4 (Apr. 1979), pp. 251–256. ISSN: 0001-0782. DOI: 10.1145/359094.359101.

[Dij59] E. W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/bf01386390.

[FW17] A. Fatima and S. M. Waseem. "Cellular Automata based Built-In-Self Test implementation for Star Topology NoC." In: *2017 11th International Conference on Intelligent Systems and Control (ISCO)*. 2017, pp. 45–48. DOI: 10.1109/ISCO.2017.7856039.

[Gal11] J. Gallier. *Discrete mathematics*. en. 2011th ed. Universitext. New York, NY: Springer, Jan. 2011.

[Gur24] L. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. https://www.gurobi.com. 2024.

[Has24] R. Haskaj. "Analysis of Wavelength Usage with Multicast in Wavelength-Routed Optical Networks-on-Chip." MA thesis. Technische Universität München, 2024.

[Her+] A. Herkersdorf, A. Lankes, T. Wild, S. Wallentowitz, and A. Sadighi. "Module 4: Interconnect." In: *Chip Multicore Processors*.

[Her06] T. Hertz. *Learning Distance Functions: Algorithms and Applications*. Hebrew University, 2006.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[Li+18] M. Li, T.-M. Tseng, D. Bertozzi, M. Tala, and U. Schlichtmann. "CustomTopo: A Topology Generation Method for Application-Specific Wavelength-Routed Optical NoCs." In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Nov. 2018.

[Liu+16] F. Liu, H. Zhang, Y. Chen, Z. Huang, and H. Gu. "Dynamic Ring-Based Multicast with Wavelength Reuse for Optical Network on Chips." In: *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*. 2016, pp. 153–160. DOI: 10.1109/MCSoC.2016.9.

[Llo82] S. Lloyd. "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[MF24] M. McGrath and D. Frank. *Propositions*. Ed. by E. N. Zalta and U. Nodelman. https://plato.stanford.edu/archives/fall2024/entries/propositions/. 2024.

[MMD17] A. Maratea, L. Marcellino, and V. Duraccio. "A GPU-Parallel Algorithm for Fast Hybrid BFS-DFS Graph Traversal." In: *2017 13th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. 2017, pp. 450–457. DOI: 10.1109/SITIS.2017.80.

[Ope25] OpenAI. *ChatGPT*. https://chat.openai.com. 2025.

[Ort+17] M. Ortín-Obón, M. Tala, L. Ramini, V. Viñals-Yufera, and D. Bertozzi. "Contrasting Laser Power Requirements of Wavelength-Routed Optical NoC Topologies Subject to the Floorplanning, Placement, and Routing Constraints of a 3-D-Stacked System." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.7 (2017), pp. 2081–2094. DOI: 10.1109/TVLSI.2017.2677779.

[Pyt23] Python Software Foundation. *Python Language Reference, version 3.x*. https://www.python.org/. Python Software Foundation. 2023.

[Sch19] U. Schlichtmann. "Chapter 1: Propositional Logic." In: *Discrete Mathematics for Engineers - Formulary*. 2019, pp. 3–23.

[TL19a] T.-M. Tseng and B. Li. "Part 2, Unit 2: Instance Modeling and Unit 4: Hierarchical Modeling." In: *Electronic Design Automation*. 2019.

[TL19b] T.-M. Tseng and M. Li. "Unit 2: Rooted - Tree, Unit 3: Graph - Edge as Relation, Unit 4: Graph - Edge as Physical Connection, and Unit 5: Graph Partition." In: *Mixed Integer Programming and Graph Algorithm for Engineering Problems*. 2019.

[Tru+20] A. Truppel, T.-M. Tseng, D. Bertozzi, J. C. Alves, and U. Schlichtmann. "PSION+: Combining logical topology and physical layout optimization for Wavelength-Routed ONoCs." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (Dec. 2020). DOI: 10.1109/TCAD.2020.2971536.

[Tur+02] I. Turksen, A. Esper, K. Patel, S. Starks, and V. Kreinovich. "Selecting a fuzzy logic operation from the DNF-CNF interval: how practical are the resulting operations?" In: *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*. 2002, pp. 28–33. DOI: 10.1109/NAFIPS.2002.1018025.

[Wal02] W. D. Wallis. *A beginner's guide to discrete mathematics*. en. 2003rd ed. Cambridge, MA: Birkhäuser, Nov. 2002.

[Wan+24] Y. Wang, L. Zhao, J. Gan, D. Yan, W. Wei, and K. Wang. "FLONA: A Low-Latency, Low-Loss Optical Network Architecture for Multicast and Broadcast." In: *2024 36th Chinese Control and Decision Conference (CCDC)*. 2024, pp. 5180–5187. DOI: 10.1109/CCDC62350.2024.10587956.

[XTS21] M. Xiao, T.-M. Tseng, and U. Schlichtmann. "FAST: A Fast Automatic Sweeping Topology Customization Method for Application-Specific Wavelength-Routed Optical NoCs." In: *Design, Automation and Test in Europe (DATE)*. Feb. 2021.

[Yan+23] W. Yang, Y. Chen, Z. Huang, H. Zhang, and H. Gu. "Routing and Wavelength Assignment for Multiple Multicasts in Optical Network-on-Chip (ONoC)." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42.12 (2023), pp. 4934–4947. DOI: 10.1109/TCAD.2023.3274951.

[Zha+19] Y. Zhang, X. Hu, D. Chen, L. Wang, M. Li, P. Feng, X. Xiao, and S. Yu. "Ultra-broadband, Low Loss and Ultra-Compact 3dB Power Splitter Based On Y-Branch With Step Waveguide." In: *2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*. 2019, pp. 1–3. DOI: 10.23919/PS.2019.8817638.

[Zhe+21] Z. Zheng, M. Li, T.-M. Tseng, and U. Schlichtmann. "Light: A Scalable and Efficient Wavelength-Routed Optical Networks-On-Chip Topology." In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2021.

[Zho+21] Y. Zhou, J. Wang, Y. Zhao, L. Huang, and S. Liu. "Improving the Performance of a NoC-based CNN Accelerator with Gather Support." In: *arXiv preprint arXiv:2108.02567* (2021).