ARBEITSGRUPPE VERNETZTE SYSTEME
FACHBEREICH INFORMATIK
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Modeling and Analysis of Optimal Slot Allocations in Wireless Ad-Hoc Networks

Mattias Nissler

Diploma Thesis

July 28, 2008

Supervisors:

Prof. Dr. Reinhard Gotzhein
Dipl. Inf. Thomas Kuhn
Dipl. Inf. Philipp Becker

I hereby certify that the material and results presented in this thesis are my own work and have not previously been submitted for a degree at any institution. All published and unpublished sources are acknowledged and referenced.

Kaiserslautern, July 28, 2008

Mattias Nissler

Acknowledgments

I would like to thank everyone who has helped and supported me while preparing this thesis. In particular I am grateful for the helpful comments and the constructive discussions I had with my supervisors, Prof. Dr. Reinhard Gotzhein, Thomas Kuhn and Philipp Becker. Their input has often helped me to refocus my thoughts and to consider new aspects that I had missed before. I also would like to thank the Service Center Informatik staff who let me run the solver program on their hardware. Additional thanks go to the nice people at Fraunhofer ITWM who gave me the possibility to use ILOG's cplex software to try and solve the mixed integer formulation of the optimization problem. Finally, I thank my friends and my family for finding the right words whenever I was frustrated about the smaller and larger problems I encountered and for supporting me in many different ways.

Contents

1 Introduction

Almost 40 years have passed since the creation of ALOHAnet [1], the first packet-based wireless computer network created in 1970 at the University of Hawaii. Since then, advances in microelectronics have led to small and cheap wireless transceivers that enable devices such as very compact mobile phones and matchbox-sized wireless sensor nodes. Wireless networks have grown more and more popular due to their unique advantages over wired networks: The wireless nature allows to deploy wireless networks very rapidly, reducing the effort or even avoiding to set up infrastructure as it is required for wired networks. Furthermore, the ubiquitous availability of the wireless medium allows to reconfigure the network easily by moving nodes around, even during operation of the network.

Still, wireless networks are subject of numerous research efforts. The central issue of coordinating transmissions by different network nodes such that these do not interfere with each other has already been identified in the ALOHAnet experiments. While the competition based approach that allows transmissions at any time if the medium is sensed idle has proved viable and is being used in current wireless networks in refined forms, more coordinated medium access control techniques have been developed. These mechanisms enable many nodes to use the wireless medium by separating transmissions by different nodes in the domains of time, space, frequency, or encoding. The advantage these techniques offer are deterministic guarantees about the network resources available to a network node and thus to the data flows it handles as long as the environment does not disturb the system, e.g. by foreign interference caused by radio transmitters not belonging to the network.

In this thesis, the problem of multi-hop data transmissions through a wireless network, possibly via several paths, is considered. Transmissions are coordinated network-wide through time division multiple access (TDMA). Time is split into intervals called slots, which can be reserved for transmissions. The reservations are made in a way such that concurrent transmissions in a slot do not interfere with each other.

The first part of this thesis defines the mathematical model that allows a formal treatment of the networks under consideration. A graph structure provides the network topology, i.e. potential links between the network nodes. Whether transmissions actually succeed is determined by a noninterference model which formalizes conditions that have to be met for a transmission to be received successfully in the situation that concurrent transmissions are being performed in the network. The noninterference model is exchangeable; three different possible models are introduced that differ in the precision they capture the underlying physical effects. A formal notion of scenarios (i.e. sets of data flows between pairs of nodes that are to be handled by the network) is defined and the conditions that must be met by a set of slot reservations to solve a scenario are developed. In the prospect of evaluating the relative performance, the model allows solutions to handle data flows via a single path or multiple paths. Finally, metrics that allow to compare the performance of solutions w.r.t. different aspects such as the end-to-end delay a flow experiences when handled by a solution or the amount of energy that is consumed under a given solution are introduced.

An important aspect of our TDMA-based network model is that the effects of queuing and indeterminism at the medium access layer are largely avoided. On the one hand, this allows to capture important system parameters such as throughput and end-to-end latency in a mathematically exact way. On the other hand, this reduced indeterminism is a very desirable property for applications that have stringent Quality-of-Service requirements, since the system allows to give guarantees on throughput and latency bounds.

The second part of this thesis examines the question whether multi-path routing, i.e. allowing data frames belonging to the same flow to take different paths through the network, is beneficial over the more restricted approach that uses a single path for all frames of a flow. Using multiple paths for a flow promises some advantages, such as better balancing of the load the network has to handle, thus possibly allowing the network to handle more data flows. Furthermore, multi-path routing can be used to improve the reliability of the data transmissions by adding redundancy as well as enhancing throughput and latency by avoiding hot spots in the network. First, some constructed example networks are investigated which prove that there are situations in which significant performance gains can be realized by using multi-paths instead of a single one. Then, topological network features are identified that influence the relative performance of multi-path and single-path solutions, respectively. Based on the system model developed in the first part of this thesis, the performance of the approaches is evaluated by studying algorithmically obtained optimal solutions to a large number of randomly generated scenarios.

Our results indicate that multi-path approaches generally yield only very little performance gains in the random networks that have been considered, both in terms of network load and end-to-end delay. The actual percentage of scenarios that multi-path approaches can solve better than an optimal single-path solution is only in the order of 10%. These findings complement other studies [14, 18] which also indicate multi-path routing does not provide significant benefit in wireless networks. However, the abstract network models used in these studies limit the applicability of the results to actual networks. In contrast, the network model used in this thesis is much more realistic and does account for physical characteristics of the wireless channel. Furthermore, while [14, 18] only examine network load, the approach used in this thesis allows to study the effect of multi-path routing also under different metrics such as end-to-end delay and energy consumption.

Based on our results, several conclusions can be drawn. Since the performance improvement of multi-path routing approaches was found to be insignificant for random networks, any additional overhead during route discovery for finding multiple paths does not pay off in terms of increased performance but should focus on improving reliability. Our evaluation indicates that the key feature of wireless networks that prevents multi-path techniques from realizing performance gains is the shared wireless medium due to the resulting interference problem. Consequently, ways to avoid the interference problem might help to increase the performance achievable with wireless networks. This includes e.g. transmission power reduction schemes in order to reduce interference at distant nodes as well as the use of directional transmitter hardware instead of the omnidirectional antenna characteristic assumed in this thesis.

The structure of the thesis is as follows: The wireless network model is defined in Chapter 2. Chapter 3 analyzes the relative performance of single-path and multi-path routing approaches and presents performance results from an empirical study of random networks. After reviewing related work in Chapter 4, Chapter 5 concludes the thesis.

2 System model

This chapter establishes the wireless network model that is used for the analysis and evaluation presented later. The goal is to develop a formal model that captures the important physical effects that characterize wireless networks while providing a sound base for reasoning about the system and its properties.

2.1 Wireless Networks

A wireless network is a collection of computing devices that have hardware for communicating over a wireless channel. The members of the network are referred to as nodes and are typically positioned in a bounded geographical area. This understanding of wireless networks includes many types of wireless networks currently discussed in literature, e.g. wireless ad-hoc networks, mobile ad-hoc networks (MANETs) and wireless sensor networks (WSNs). All of these have different characteristics, e.g. regarding available hardware, level of mobility, energy resources etc. Also, the network can either be homogeneous, i.e. consist of identical devices exclusively, or inhomogeneous.

The hardware resources that are available to the wireless nodes vary significantly depending on the type of wireless network. In wireless access networks and metropolitan area networks as specified in IEEE 802.16 [19], there are dedicated base stations, and the user nodes are personal computers that can provide the necessary energy and computing power to drive a decent wireless network interface. In contrast, WSN nodes require cheap low powered hardware packed into tiny devices. Thus, the nodes' available energy, computing power and size limit many characteristics such as transmission range, complexity of hardware and antenna design.

The level of mobility also varies greatly from nodes being placed at fixed locations to settings where every node moves in an unpredictable fashion. For example, nodes in wireless access networks are stationary most of the time. The same is true for many flavors of WSNs. But networks formed by mobile communication equipment carried around by humans, e.g. mobile phones or military applications that interconnect soldiers on the battlefield, exhibit a very high degree of mobility. Mobility induces significant additional problems due to the network topology being highly dynamic. The main problem here is finding and maintaining routes for data streams from one node to another, even when source and destination (and also the network) undergo continuous changes in topology.

As mentioned above, wireless networks can be homogeneous or inhomogeneous. For example, in a wireless access network, there are special nodes that are interconnected by wired infrastructure. In the case of a WSN some nodes might serve a special purpose, e.g. act as an interface node that is accessed by the user to retrieve data from the network. Also, transmission range and energy resources can be different amongst nodes. There are situations in which these characteristics can and should be exploited. For example, routing algorithms

should prefer high-powered nodes as intermediate hops for data transmissions in order to keep other nodes having tighter energy constraints alive.

From a more abstract point of view, an interesting parameter of wireless networks is the arrangement of the nodes. When discussing the capacity of wireless networks, most authors assume the nodes are embedded in a plane or surface of some kind [17, 29]. This is a valid assumption for a large number of wireless networks. However, setups that position the nodes in 3D volumes are not unusual, for example a wireless network setup by a company in an office building. Gupta and Kumar show that the theoretical bound on network capacity is larger in the 3D case than it is for the 2D case [16].

2.2 Formalization

For formal treatment, it is useful to have an abstract concept of wireless networks. The abstraction should enable convenient examination and reasoning about the network and its properties, thereby abstracting from the details while preserving the important problem characteristics. The usual approach is to model a wireless network as a directed connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The wireless network nodes form the set of nodes $\mathcal{V}$ of the graph. If a wireless node is in the transmission range of another node, there is an edge in the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ that connects the corresponding graph nodes. The sets of neighbor nodes of a node v in the graph $\mathcal{G}$ are referred to by the sets $\mathcal{G}_v^i$ and $\mathcal{G}_v^o$ for nodes that are connected by incoming or outgoing edges, respectively, and are defined as follows:

$$\mathcal{G}_v^i := \{w \in \mathcal{V} \mid (w, v) \in \mathcal{E}\} \tag{2.1}$$

$$\mathcal{G}_v^o := \{w \in \mathcal{V} \mid (v, w) \in \mathcal{E}\} \tag{2.2}$$

Note that if $(a, b) \in \mathcal{E}$, i.e. node b is in the transmission range of node a, the inverse edge (b, a) does not necessarily exist in $\mathcal{E}$. This can be due to the transmitter of node a sending at a higher energy level than b's transmitter. However, there may also be more subtle physical effects leading to such a situation, e.g. more background noise at node a, resulting in node a not being able to separate b's signal from the background noise. On the abstract level, this means $\mathcal{G}$ is accurately modeled as a directed graph. However, unidirectional links between nodes impose problems for wireless network protocols, so many approaches ignore unidirectional links, resulting in the assumption of the connectivity graph $\mathcal{G}$ being undirected (e.g. [22, 5, 36]).

The connectivity graph can easily be determined from an actual wireless network by simply assigning each node an identifier, thereby forming the node set $\mathcal{V}$, and checking each pair of nodes whether they can transmit data to each other, which defines $\mathcal{E}$. However, it is useful to be able to work with hypothetic models without having to set up and measure an actual network. The important point is to make sure the model is realistic, i.e. that there actually could be a physically existing wireless network exhibiting the model's properties. To construct such hypothetical models, one can choose a set of nodes $\mathcal{V}$ arbitrarily. The problematic aspect is to make sure $\mathcal{E}$ models a realistic situation. In other words, not every possible connectivity graph models a realistic wireless network.

This point is best demonstrated by examples. Figures 2.1 and 2.2 show two graphs that are not realistic, i.e. actual networks will not have the connectivity graphs as depicted in

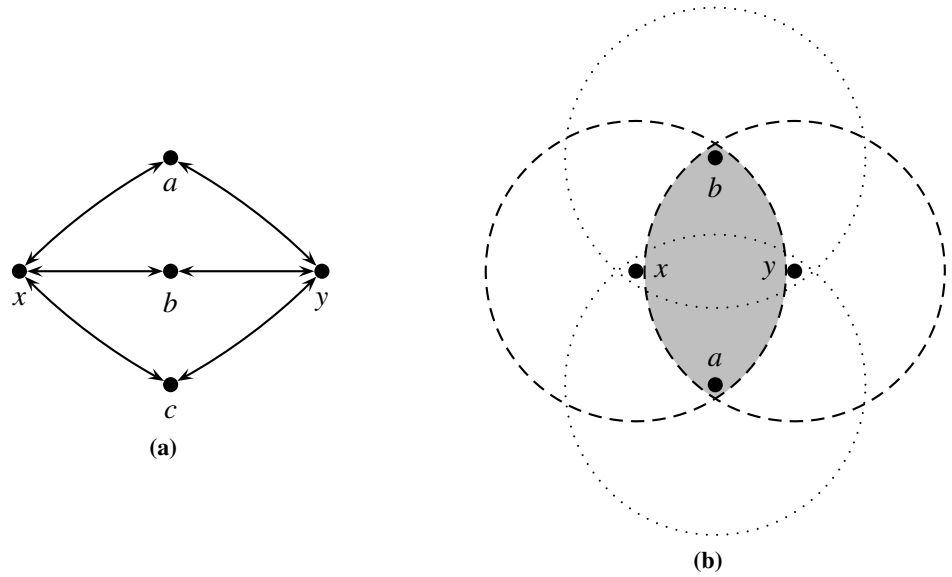


Figure 2.1: An example of a connectivity graph that is unrealistic if perfect conditions (see text) are assumed. Due to the graph topology in (a), the nodes a, b, and c must be located within the shaded area in (b). Due to its limited size, a, b and c cannot be placed in such a way that they are all pairwise out of transmission range.

the figures, if certain assumptions about the networks are made. For the examples, it is presumed that the wireless nodes are placed in a plane and all have the same transmission range and reception sensitivity. Transceiver and antenna characteristics are assumed to be identical among the nodes. Under the assumption of the antennas in use being perfectly omnidirectional, the area in which a transmission from a node can be picked up is a unit disk. Hence, the connectivity graph is a unit disk graph.

Consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ shown in Figure 2.1a. From the graph, it follows that nodes x and y cannot communicate with each other, i.e. they are not located within transmission range of each other. However, nodes a, b and c each have edges to both x and y, so they are in transmission range of both x and y. Therefore, they must be located within an area where x's and y's unit disks overlap. This is illustrated by the shaded area in Figure 2.1b. However, the limited size of the shaded area does not allow a placement of a, b and c in a way such that they are out of transmission range of each other. This can be seen by observing the following: Assume a is placed below the line connecting x and y. Then, a's unit disk covers the lower half of the shaded area. Hence, the next node, say b, must be placed in the upper half of the shaded area and its unit disk will cover this area. This leaves no possible location for the remaining node c. All other cases are analogous. Also note that the size of the overlap area of x's and y's transmission ranges depends on the distance between x and y; the case of maximum overlap area is depicted in the figure.

Another situation of an unrealistic connectivity graph can be found in Figure 2.2a. Node m has edges to all nodes in the set $B = \{a, b, c, d, e, f\}$. However, it is a fundamental property of unit disk graphs that a node can have at most 5 neighbors that are pairwise non-adjacent. Thus, there must exist an edge between at least one pair of nodes in B. This can be seen in the illustration in Figure 2.2b. Nodes a, b, c, d and e have already been placed. No matter where

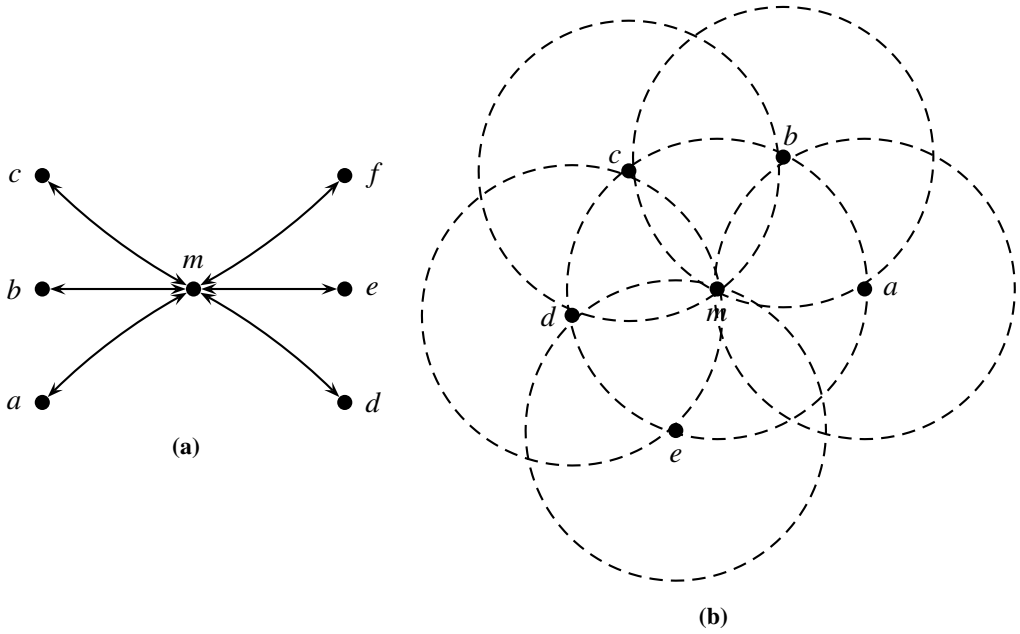


Figure 2.2: (a) shows an unrealistic connectivity graph violating the unit disk graph property that a node cannot have 6 pairwise non-adjacent neighbors. On the setup as depicted in (b), there is no location where node f can be placed without being in one of the other nodes' transmission ranges.

node f is placed in the transmission range of m it will also be in the transmission range of at least one of the other nodes.

The examples also show that the class of valid connectivity graphs depends on the assumptions made about the physical wireless networks that should be modeled by the graphs. The examples assumed unit disks as transmission ranges. This assumption is invalid in many situations, e.g. when obstacles inhibit signal propagation in certain directions. For example, a possible wireless network setup that yields the connectivity graph as shown in Figure 2.1a can be obtained by placing walls between the nodes a, b and c (see Figure 2.3).

In order to be able to generate realistic connectivity graphs, it is useful to extend the model with additional information and use it to construct a connectivity relation. A possible approach is to assign a location in 2- or 3-dimensional space to every network node in $\mathcal{V}$. Node locations are given by a function $\mathcal{L} : \mathcal{V} \rightarrow \mathbb{R}^n$ that maps a node to its location in 2- or 3-dimensional space. Then, $(a, b) \in \mathcal{E}$ iff b is in the transmission range of a. Assuming a disk model in 2-dimensional space, this can be checked by computing the distance between nodes a and b from their locations $\mathcal{L}(a)$ and $\mathcal{L}(b)$. It is important to note that an edge (a, b) in the connectivity graph does by no means guarantee that node a can always successfully transmit messages to b. Whether a's transmission is successfully received by b depends on the behavior of the other nodes at the time a transmits its message. If there are other nodes that transmit concurrently, collisions or increased noise might prevent b to receive the message. This effect is further discussed in Section 2.3.

Note that the model could be further extended with information such as interference caused by radio transmitters in the area that do not belong to the wireless network. This can be

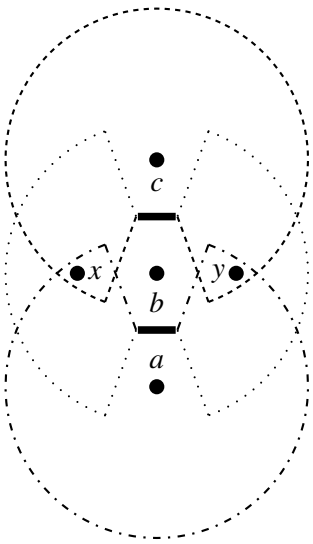


Figure 2.3: Obstacles that inhibit radio signal propagation can yield connectivity graphs that would be unrealistic under perfect conditions (compare Figure 2.1)

modeled as background noise and included in the model as a function NOISE : $\mathbb{R}^n \rightarrow \mathbb{R}$ that gives the amount of background noise at a given location. Another aspect are antennas with directional characteristics. Both factors result in the transmission ranges to be different from a perfect disk and influence the resulting connectivity graph.

2.3 Noninterference Models

If several nodes in the wireless network concurrently transmit messages, the situation can arise that a message is not successfully received at its destination even though the corresponding edge in the connectivity graph exists. This can be due to collisions that occur when a node overhears multiple transmissions from several nodes that are all received at approximately the same energy level. But even if there is only a single transmission at a reasonably high energy level on the medium, reception can fail due to increased background noise caused by other concurrently transmitting nodes that are typically located farther away.

Therefore, conditions for successful reception of a message (other than the receiving node being in transmission range) need to be defined. This is what a *Noninterference Model* is about. Several approaches can be found in previous work.

Connectivity Graph Based Model

This method uses only topological information from the connectivity graph. From the graph topology, colliding transmissions can be identified. These have been classified in the literature as *Primary* and *Secondary Conflicts* [41]. Primary Conflicts occur at node b when there is a neighbor a that transmits a message to b but b also transmits a message to another neighbor c. Obviously, b cannot both receive the message from a and transmit the message to c at the same time in a TDMA network in which all nodes operate on the same radio channel. Secondary Conflicts are due transmissions from multiple neighbors that interfere at the intended receiver.

Definition 1 (Graph-Based Noninterference Model)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $a, b \in \mathcal{V}$ and $\text{TX}(v,t)$ be a predicate that decides whether node v transmits at time t. Then the condition for successful reception of a transmission from node a to b at time t in the Graph-Based Noninterference Model is:

$$\Phi_1(a,b,t) \equiv \neg\text{TX}(b,t) \wedge \forall v \in \mathcal{G}_b^i \setminus \{a\} : \neg\text{TX}(v,t) \tag{2.3}$$

The advantage of this connectivity graph based noninterference model is its simplicity. It only needs topological information that is contained in the connectivity graph. However, it does not account for increased background noise due to non-adjacent nodes that transmit concurrently. This can be mitigated by considering not only the one-hop neighborhood of b but also requiring the 2- or even 3-hop neighbors of b not to transmit at the same time.

Protocol Model

The Protocol Model [17] uses geometric information to decide whether a transmission from node a to node b is successfully received.

Definition 2 (Protocol Noninterference Model)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $a, b \in \mathcal{V}$ and $\text{TX}(v,t)$ be a predicate that decides whether v transmits at time t. The condition for successful reception of a transmission from a to b at time t in the Protocol Noninterference Model is:

$$\Phi_2(a,b,t) \equiv \forall v \in \mathcal{V} \setminus \{a\} : (\text{TX}(v,t) \rightarrow (|\mathcal{L}(v) - \mathcal{L}(b)| \geq (1+\Delta)|\mathcal{L}(a) - \mathcal{L}(b)|)) \tag{2.4}$$

$|x - y|$ is the Euclidean distance in $\mathbb{R}^n$. $\Delta \geq 0$ is a system parameter that specifies a guard zone.

The transmission is successfully received at node b if there is no other concurrently transmitting node within the transmission distance and the guard zone. The purpose of the guard zone is to add distance between transmitters. This is necessary due to the fact that transmissions arriving from nodes farther away may be at a too low energy level to be picked up but still add background noise that interferes with local transmissions. Note that the Protocol Model assumes all nodes to transmit at the same energy level, such that their transmission ranges can be assumed to be about equal. Otherwise, the direct correlation between distance and interference cannot be justified.

Physical Model

The physical characteristics of the wireless medium are more accurately captured by the Physical Model [17]. It considers the energy levels at which the nodes transmit their messages.

Definition 3 (Physical Noninterference Model)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $a, b \in \mathcal{V}$, $\text{TXP}(v,w,t)$ be node v's transmission energy level at time t when transmitting to node w and N, α and γ be system constants. For a transmission from a to b, the received energy level at node c is:

$$\rho(a,b,c,t) = \text{TXP}(a,b,t) \cdot \left(\frac{1}{|\mathcal{L}(a) - \mathcal{L}(c)|}\right)^{\alpha} \tag{2.5}$$

The condition for b to successfully receive a transmission from a at time t in the Physical Noninterference Model is:

$$\Phi_3(a,b,t) \equiv \frac{\rho(a,b,b,t)}{N + \sum\limits_{\substack{v \in \mathcal{V} \\ v \neq a}} \max\limits_{w \in \mathcal{V}} \rho(v,w,b,t)} \geq \gamma \tag{2.6}$$

Here, N is a constant describing the ambient noise and the parameter α controls the influence of distance on the energy level outside a small neighborhood of the transmitter. The inverse square law states $\alpha = 2$, larger α values can be used to account for additional path loss. Equation 2.6 effectively calculates the signal-to-interference-and-noise ratio (SINR) by summing up the ambient noise energy and the energy picked up from concurrent transmissions and relating that to the energy level of the incoming transmission from a. If the SINR is above a receiver-specific threshold γ, the message is received successfully.

2.4 Medium Access Control

Many different Medium Access Control mechanisms have been put forward over the years. There are different techniques that address the arbitration problem of the shared wireless medium, the most popular choices being Carrier Sense Multiple Access (CSMA), Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA).

CSMA is a very simple approach, e.g. used in traditional Ethernet-type wired networks. The idea is to monitor the medium and wait until it is free before starting a transmission. Collisions can still happen and must be taken care of. While CSMA is particularly easy to implement it cannot use the full bandwidth of the medium due to collisions.

Frequency division techniques are in widespread use in radio communications. For example, stereo FM radio transmissions combine two signals on different frequencies from which the two stereo channels can be reconstructed by the receiver. In wireless networks, FDMA often refers to systems that have a set of available frequencies from which each node or link is assigned one to use. Given that neighboring nodes or links use different frequencies to transmit messages, the transmissions do not interfere and can be successfully received. However, receiving nodes need to know in advance that they are supposed to receive a transmission, so they can tune their receiver hardware to the frequency of the sending node.

TDMA splits time into slots and assigns these to nodes or links. The result is a TDMA schedule that determines the time intervals during which each node may transmit. While an FDMA receiver has to know in advance about a transmission that is to be received in order to tune the receiver to the right frequency, no such difficulties arise with TDMA. Receivers can constantly monitor the channel and pick out all transmissions that are relevant to them. However, one promise of TDMA is that of energy efficiency, which is achieved by powering down the receiver hardware during time slots that are not assigned to incoming transmissions.

Time division techniques often divide time into time intervals that are called macro slots. All macro slots share the same internal structure made up of so called micro slots available for assignment. Some TDMA variants reserve special time intervals within macro slots for management purposes such as micro slot allocation, other approaches use regular micro slots for management. There are also static TDMA systems that use fixed schedules constructed

in advance of system deployment such that no online micro slot management functionality is required.

CDMA uses a single radio channel on which the network nodes are permitted to send concurrently. However, all nodes sending concurrently need to use a different code. The idea is to have the transmitter encode a single bit by the positive or negative version of the sender's code, the different codes being orthogonal signals of a fixed length. A receiving node can then reconstruct an individual bit from the combination of all concurrently transmitted signals that it received by correlating the received signal with the codes of the senders. The correlation result will indicate which bit value was transmitted by the sender.

FDMA, TDMA and CDMA effectively divide the shared medium into a number of subchannels. The problem of assigning subchannels to nodes or edges in the connectivity graph can be modeled as a graph coloring problem [40]. The vertex coloring problem (i.e. assigning colors to vertices with the restriction that adjacent vertices receive different colors, thereby using a minimum number of colors) is known to be NP-hard [25]. In systems that use dynamic channel allocation, i.e. assign communication channels at runtime, there is the additional need for a distributed channel assignment algorithm. The algorithm should not depend on global information about the network but use only information available to the local network node and its neighborhood. Furthermore, wireless networks can be dynamic due to nodes joining and leaving and node mobility. This imposes additional requirements on the channel assignment algorithm.

It should be noted that FDMA, TDMA and CDMA can be combined. This has been done in a number of mobile telecommunication systems, e.g. GSM which uses a combination of TDMA and CDMA.

2.5 TDMA model

In this thesis, we focus on wireless networks using TDMA as medium access control technique. The choice of TDMA can be justified by different aspects: TDMA can be implemented relatively easily on current low cost hardware. A clock of sufficient accuracy is available on virtually every hardware platform. Time synchronization between network nodes can also be achieved at the required precision (see Section 4.1). As mentioned before, TDMA systems can be made very energy-efficient by turning off the radio hardware during time intervals in which no data is to be sent or received. This is very important in the context of sensor networks due to the limited energy resources available to each node.

This section introduces the formal model of TDMA as used in a wireless network. Time is divided into time intervals called *macro slots*. These are of fixed duration and repeat indefinitely. It is assumed that the network employs a time synchronization mechanism that allows the nodes to agree on the starting time of each macro slot. In the formal model, the sequence of macro slots $\underline{\mathcal{S}}$ is assumed to be consecutively numbered, i.e. $\underline{\mathcal{S}} = \{\underline{s}_1, \underline{s}_2, \ldots\}$. Internally, a macro slot is subdivided into $n_{\mathcal{S}}$ *micro slots* or *slots* for short. $\mathcal{S}$ refers to the set of micro slots which are indexed by positive numbers, i.e. $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_{\mathcal{S}}}\}$. A micro slot can be assigned to an edge (a, b), such that the node a can use it for data transmissions to b. An illustration of the time division scheme employed by the network can be found in Figure 2.4.

In some of the following definitions, a consecutive slot numbering scheme is needed that allows to distinguish between instances of the same micro slot in different macro slots. This

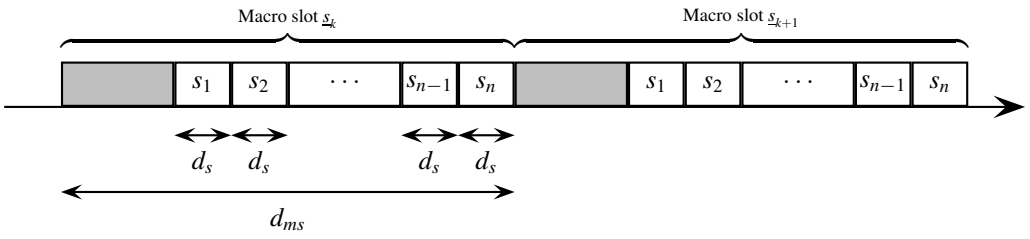


Figure 2.4: TDMA model: Macro slots of fixed length that are divided into micro slots are repeated indefinitely. The shaded areas indicate time intervals not available for TDMA. Micro slots are consecutively numbered $s_1, \ldots, s_n$, starting at the first micro slot in the macro slot. d_s is the duration of a micro slot, d_{ms} the duration of a macro slot.

numbering $\vec{\mathcal{S}} = \{\vec{s}_1, \vec{s}_2, \ldots\}$ is obtained by combining slot number pairs $(\underline{s}_i, s_k) \in \underline{\mathcal{S}} \times \mathcal{S}$. The fact that every macro slot contains $n_{\mathcal{S}}$ micro slots suggests a correspondence that counts each macro slot as $n_{\mathcal{S}}$ microslots, i.e. the pair $(\underline{s}_i, s_k)$ corresponds to slot $\vec{s}_{i \cdot n_{\mathcal{S}} + k}$ in the consecutive numbering.

Note that the time synchronization protocol and possibly other underlying network maintenance protocols need to communicate with other network nodes. The time intervals used for these maintenance communications are not under the control of the TDMA system. This situation can be captured by setting aside a fixed number of slots in each macro slot for these purposes. For the formal model, we will simply ignore these slots by not including them in $\mathcal{S}$. Potentially existing time intervals that are not available to TDMA contribute e.g. to multi-hop latency. However, the formal model ignores these effects for simplicity. Instead of a continuous time model a discrete time model that is based on the consecutive slot numbering is employed.

The messages carried by the TDMA network are termed *frames*. All frames are of a fixed size of n_{F} bytes that must be chosen such that each node can transmit at least one complete frame during a single micro slot. Using constant size frames has the advantage that there is no need to further fragment frames. Moreover, splitting and merging data flows can be done on a per frame granularity level. The necessary meta data (i.e. a sequence number) is generated by the source node and is carried within each frame, so nodes that forward frames do not need to reformat the data that flows through them.

Nodes are allowed to transmit messages on a per slot basis. Slots cannot be subdivided, thus a node is either allowed to transmit for the time of the entire slot or it has to stay passive for that time. However, depending on the encoding and transmission rate used, the time required to send a frame may vary among different nodes. If a node can fit more than one frame transmission into the time interval provided by a macro slot, it is allowed to send multiple frames during that interval. The time required to transfer a frame is given by the function $\mathcal{T} : \mathcal{E} \times \mathcal{S} \to \mathbb{R}^+$. Thus, $\mathcal{T}((a,b),s)$ specifies the time it takes node a to transfer a frame to node b during slot s. It is assumed that $\mathcal{T}(e,s) \leq d_s$ for all edges e and slots s. This ensures a node can transfer at least one frame during a slot.

A micro slot s is assigned to a set of edges $\mathcal{M} \subseteq \mathcal{E}$. $(a,b) \in \mathcal{M}$ states that node a can use the time intervals corresponding to slot s to transmit frames. If $|\mathcal{M}| > 1$, multiple nodes are allowed to transmit during the micro slot. Formally, the assignment of slots to edges is modeled by a function:

Definition 4 (Slot assignment)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{S}$ be the set of slots available in the TDMA scheme. A function $\text{SA} : \mathcal{S} \to 2^{\mathcal{E}}$ *that maps each slot to a set of connectivity graph edges is called* slot assignment.

Local to a network node v, a slot assignment is interpreted as follows: If $(v, w) \in \text{SA}(s)$ for a time slot $s \in \mathcal{S}$, node v has the opportunity to transmit a message to w during s. Node w should be prepared to receive a message from v during slot s. If $\text{SA}(s) = \emptyset$, micro slot s is unassigned.

The definition of a slot assignment does not include information about the macro slots in which the function is to be used. For static TDMA systems, there is only one slot assignment that is agreed upon before the network is deployed. Dynamic TDMA systems allow the assignment of slots to transmissions to be changed while they are running. In the model, this can be captured by switching between different slot assignments, e.g. at macro slot boundaries.

Not all mathematically possible slot assignments can be allowed. For example, $\text{SA}(s) = \{(v, w), (w, v)\}$ must be avoided since that would mean nodes v and w can both transmit and receive at the same time, which is not possible in the model. Generally, it must be ensured that the slot assignment is *consistent*, i.e. that there can be no interference from concurrent transmissions which would prevent a message from being received. This can be formally captured with the help of the noninterference models that have been defined in Section 2.3:

Definition 5 (Consistent slot assignment)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and SA *a slot assignment.* SA *is* consistent *under a noninterference model Φ if all transmissions allowed by the slot assignment satisfy the noninterference condition Φ:*

$$\forall s \in \mathcal{S}\, \forall (a, b) \in \text{SA}(s) : \Phi(a, b, s) \tag{2.7}$$

Note that this definition deviates in the use of the noninterference model from how the models were originally defined. Instead of passing a point in time as the third parameter, a slot is given. This can be justified by the observation that at any point in time within the interval specified by the slot the set of potentially concurrently transmitting nodes is the same. Thus, we can extend the definition of the noninterference models so they also accept slots and will evaluate to true if the noninterference condition is true for all points in time within the slot.

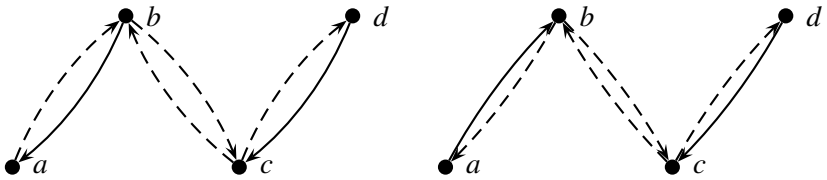
All noninterference models already prevent a node from transmitting concurrently when it is supposed to receive a message, so this restriction need not be stated explicitly. What is still missing for Φ to be properly defined is the definition of the TX predicate for the connectivity graph based and the protocol noninterference model, and the TXP function definition for the physical noninterference model. The definition of $\text{TX}(a, s)$ for slot s is as follows:

$$\text{TX}(a, s) \equiv \exists b \in \mathcal{V} : (a, b) \in \text{SA}(s) \tag{2.8}$$

Note that the predicate is defined to be true not only when node a actually transmits in slot s, but when the slot assignment assigns slot s to node a so it can potentially transmit. The reason is that successful reception of concurrent transmissions arriving at other nodes must be ensured no matter if a actually transmits in slot s or not.

Likewise, the definition of the TXP function must account for potential transmissions instead of actual transmissions. Hence, $\text{TXP}(a, b, s)$ in Φ_3 is the power level at which node a will transmit a message to node b if a decides to do so in slot s. Note that the TXP function is conceived here not as a model property, but is a parameter that can be tuned in conjunction with the slot assignment.

Figure 2.5 gives an example that shows two slot assignments. When considered under the graph-based noninterference model, the slot assignment in Figure 2.5b is consistent, while the situation depicted in Figure 2.5a shows an inconsistent slot assignment.



(a) inconsistent slot assignment **(b)** consistent slot assignment

Figure 2.5: Examples for slot assignments. Solid and dashed arrows represent the edges of the connectivity graph. Solid lines indicate edges that are in $\text{SA}(s)$. Consider the graph-based noninterference model. The slot assignment in (a) is not consistent, since a transmission from node b would interfere with a possible transmission on the edge (d, c). An example of a consistent slot assignment is given in (b).

2.6 Flows and Scenarios

At this point, the model is expressive enough to think about data flowing through the network. For simplicity, we consider only unicast data streams that originate at a node called *source* and flow to a destination node that is called *sink* node.

Definition 6 (Data flows and scenarios)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph. A scenario *$\mathcal{F}$ is a finite set of flows. The flows are numbered and referred to by f_i, i.e. $\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{F}|}\}$. Each* flow *$f_i$ is assigned a pair of source and sink nodes, denoted by $\text{FSOURCE}(f_i)$ and $\text{FSINK}(f_i)$, respectively, such that $\text{FSOURCE}(f_i) \neq \text{FSINK}(f_i)$.*

The formal definition of a flow does not indicate how the network is supposed to actually handle the flow. It is up to a routing algorithm to decide on which paths the flow data is to be transferred from the source to the sink. Cycle-free Paths are defined as finite sequences of nodes without loops and are subject of the following definitions:

Definition 7 (Cycle-free Path)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. A non-empty sequence of nodes $p = (v_1, \ldots, v_n) \in \mathcal{V}^+$ is called cycle-free path *if the following conditions are satisfied:*

- *Adjacent nodes in the path form an edge in $\mathcal{G}$:*

$$\forall i \in \{1, \ldots, n-1\} : (v_i, v_{i+1}) \in \mathcal{E} \tag{2.9}$$

- *No node occurs more than once in p:*

$$\forall i, j \in \{1, \ldots, n-1\} : i \neq j \rightarrow v_i \neq v_j \tag{2.10}$$

The length of p is referred to by $|p|$ and is defined to be the number of edges in p, i.e. $|p| = n - 1$. In the following, the notation v_k^p denotes the kth node in p, thus $p = (v_1^p, \ldots, v_{|p|+1}^p)$. We define $\mathcal{P}_\mathcal{G}$ to be the set of all cycle-free paths in $\mathcal{G}$.

Definition 8 (Hop-distance)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. The function $\text{DIST}_\mathcal{G} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{N}$ gives the hop distance between two nodes v and w in graph $\mathcal{G}$ defined as:

$$\text{DIST}_\mathcal{G}(v, w) := \min_{p \in P_{v,w}} |p| \tag{2.11}$$

where $P_{v,w} := \{p \in \mathcal{P}_\mathcal{G} \mid v_1^p = v \wedge v_{|p|+1}^p = w\}$ is the set of all paths starting at v and ending in w.

Definition 9 (Subpath)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. $\wr \subseteq \mathcal{P}_\mathcal{G} \times \mathcal{P}_\mathcal{G}$ denotes the subpath relation*, i.e. $p \wr q$ is true if p is part of q:*

$$\wr := \left\{(p, q) \in \mathcal{P}_\mathcal{G} \times \mathcal{P}_\mathcal{G} \mid \exists k \in \mathbb{N}_0 \, \forall i \in \{1, \ldots, |p| + 1\} : v_i^p = v_{i+k}^q\right\} \tag{2.12}$$

Note that due to the path definition, edges can be dealt with as paths of length 1. Moreover, for an edge $e \in \mathcal{E}$ and a path $p \in \mathcal{P}_\mathcal{G}$, e is part of the path if $e \wr p$. Frames belonging to a flow are transferred through the network along a path that starts at the source and ends at the sink. The flow routing notion describes which paths are used for a flow. We allow *multipath routing*, thus there can be a set of paths that connect source and sink and the system can use all these paths to transfer the frames belonging to the flow.

Definition 10 (Flow routing)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $f \in \mathcal{F}$ a data flow in $\mathcal{G}$. A finite set of pairs $\Psi_f \subseteq \mathcal{P}_\mathcal{G} \times \mathbb{N}$ is called routing of f *if all paths start at f's source node and end at the sink node:*

$$\forall (p, \kappa) \in \Psi_f : v_1^p = \text{FSOURCE}(f) \wedge v_{|p|+1}^p = \text{FSINK}(f) \tag{2.13}$$

Ψ_f describes the how the network handles the flow. $(p, \kappa) \in \Psi_f$ indicates that κ frames per macro slot of capacity are provided on path p.

After assigning a flow routing to each flow in a scenario, micro slots must be allocated to the paths in the flow routing in such a way that all paths receive enough slots so their capacities are realized. When allocating slots, the slot assignment must be respected, i.e. slots can only be allocated if the slot assignment permits the edge be used in the slot.

Definition 11 (Flow mapping)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, SA *a slot assignment and $\mathcal{F}$ a scenario. A function* $\text{FMAP} : \mathcal{F} \times \mathcal{P}_\mathcal{G} \times \mathcal{E} \times \mathcal{S} \rightarrow \mathbb{N}$ *is called* flow mapping *if the following conditions hold:*

- *Capacity must only be allocated to edges on paths that start at the respective flow's source and end at the sink:*

$$\forall f \in \mathcal{F}\ \forall p \in \mathcal{P}_\mathcal{G}\ \forall e \in \mathcal{E}\ \forall s \in \mathcal{S}: \\ \text{FMAP}(f,p,e,s) > 0 \rightarrow (\text{FSOURCE}(f) = v_1^p \vee \text{FSINK}(f) = v_{|p|+1}^p) \quad (2.14)$$

$$\forall f \in \mathcal{F}\ \forall p \in \mathcal{P}_\mathcal{G}\ \forall e \in \mathcal{E}\ \forall s \in \mathcal{S}: \text{FMAP}(f,p,e,s) > 0 \rightarrow e \wr p \quad (2.15)$$

- *The flow mapping may only allocate capacity for a combination of slot and edge if* SA *allows to use this edge during the slot:*

$$\forall f \in \mathcal{F}\ \forall p \in \mathcal{P}_\mathcal{G}\ \forall e \in \mathcal{E}\ \forall s \in \mathcal{S}: \text{FMAP}(f,p,e,s) > 0 \rightarrow e \in \text{SA}(s) \quad (2.16)$$

- *No more capacity than available in a slot must be allocated:*

$$\forall v \in \mathcal{V}\ \forall s \in \mathcal{S}: \sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_\mathcal{G}} \sum_{w \in \mathcal{G}_v^o} \text{FMAP}(f,p,(v,w),s) \cdot \mathcal{T}((v,w),s) \leq d_s \quad (2.17)$$

The idea behind the FMAP definition is that $\text{FMAP}(f,p,e,s)$ gives the number of frames that are transfered via edge e during slot s for path p handling flow f. The flow mapping definition deliberately allows flow mappings that assign only fractions of the capacity available over an edge during a slot to a single path. This is useful when two paths of the flow routings share a common subpath, in which case they can share a slot. Furthermore, a flow mapping may spread the capacity available in a slot to edges that start at the same node but differ in the second node. Depending on what the underlying technology allows, this aspect can be ruled out by an additional condition that makes sure that for every node v in the network there is only one destination node w that v needs to talk to during a single slot:

$$\forall f \in \mathcal{F}\ \forall p \in \mathcal{P}_\mathcal{G}\ \forall (v,w) \in \mathcal{E}\ \forall s \in \mathcal{S}: \text{FMAP}(f,p,(v,w),s) > 0 \rightarrow \\ \neg \left(\exists f' \in \mathcal{F}\ \exists p' \in \mathcal{P}_\mathcal{G}\ \exists w' \in \mathcal{V} \setminus \{w\}: \text{FMAP}(f',p',(v,w'),s) > 0\right) \quad (2.18)$$

Obviously, the flow mapping should allocate the slots to flows and paths in a way such that every flow can be transmitted from the source to the sink. For a single flow, this is achieved by taking a flow routing and allocating slots to the edges on the paths of the routing, such that each path in the routing receives enough transmission opportunities to provide its announced capacity.

Definition 12 (Supporting flows and scenarios)
Let $\mathcal{G} = (\mathcal{V},\mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario, SA *a slot assignment and* FMAP *a flow mapping.*

1. *Let $f \in \mathcal{F}$ a flow and Ψ a routing of f. f is called* supported by $(\text{SA}, \text{FMAP}, \Psi)$ *if the following conditions hold:*
   - *The flow mapping only allocates capacity to edges that are actually part of a path in Ψ:*

$$\forall p \in \mathcal{P}_\mathcal{G}\ \forall e \in \mathcal{E}\ \forall s \in \mathcal{S}: \text{FMAP}(f,p,e,s) > 0 \rightarrow \exists (p,\kappa) \in \Psi: e \wr p \quad (2.19)$$

- *For each path in Ψ, every edge belonging to the path is allocated the capacity share per macro slot as required by the path:*

$$\forall (p, \kappa) \in \Psi \, \forall e \in \mathcal{E} : e \wr p \rightarrow \sum_{s \in \mathcal{S}} \text{FMAP}(f, p, e, s) = \kappa \tag{2.20}$$

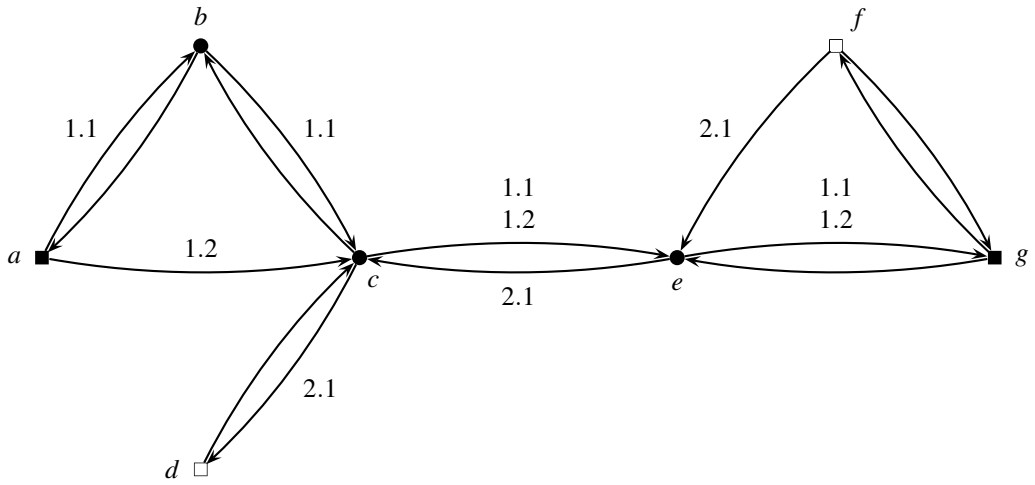
2. *Let $\psi : \mathcal{F} \rightarrow 2^{\mathcal{P}_\mathcal{G} \times \mathbb{N}}$ be a function that maps each flow f to a routing of f. The scenario is called* supported by $(\text{SA}, \text{FMAP}, \psi)$ *if every flow $f \in \mathcal{F}$ is supported by* $(\text{SA}, \text{FMAP}, \psi(f))$.

Figure 2.6 gives an example that illustrates the conditions that must be met for a flow f to be supported by a triple $(\text{SA}, \text{FMAP}, \Psi)$ of slot assignment, flow mapping and routing of f. The connectivity graph is given in Figure 2.6a. We assume that all nodes use the same transmission rate and that the chosen rate allows exactly one frame to be transmitted during a single slot. In the example, there are two flows defined: Flow f_1 describes a data stream from node a to node g, the other flow f_2 has node f as its source node and flows to node d. The flow routings that are considered are given in 2.6b and are also indicated by the edge labels in Figure 2.6a. The routings specify that f_1 should receive a capacity of 1 frame per macro slot on each of the two paths (a, b, c, e, g) and (a, c, e, g) while 2 frames per macro slot of capacity should provided for the single path (f, e, c, d) that handles f_2. We now investigate how the scenario can be supported by a pair of suitable slot assignment and flow mapping with the given flow routings.

Consider the slot assignment and flow mapping given in Figures 2.6c and 2.6d, respectively. To verify the triple $(\text{SA}, \text{FMAP}, \psi)$ actually supports the scenario, several aspects must be checked. First of all, ψ should actually map each flow to a flow routing, i.e. all paths must start at the corresponding flow's source node and end at its sink node. Furthermore, we need to check that the slot assignment is consistent w.r.t. the chosen noninterference model. The function SA as given in Figure 2.6*c* is indeed consistent under the graph-based noninterference model. The flow mapping FMAP must be valid as of Definition 11, which is obviously the case. Additionally, all flows $f \in \mathcal{F}$ must be supported by $(\text{SA}, \text{FMAP}, \psi(f))$, i.e. meet the conditions of Definition 12. Consider the first flow. Its routing specifies the paths (a, b, c, e, g) and (a, c, e, g), each providing a capacity of 1 frame per macro slot. Obviously, the flow mapping only allocates capacity to edges of the paths, so the first condition of Definition 12 is true. What is left is to check the second condition for each edge that is part of the first flow's routing. For example, at edge (c, e) the two paths each require capacity of 1 frame per macro slot each. As can be seen in Figure 2.6d, FMAP allocates slots 6 and 10 to the first and second path of flow 1's routing, respectively. This can also be checked for all other combinations of flow, edge and routing path, so the condition is true. Thus, scenario $\mathcal{F}$ is supported by $(\text{SA}, \text{FMAP}, \psi)$.

2.7 Frame transmission behavior

In this section, the characteristics of frames that flow along a path in a flow's routing are studied. Based on the assumptions that the source node continuously sends frames, that frames are forwarded as soon as possible by the nodes on a path, and that the nodes use a FIFO



(a)

f	$\psi(f)$
f_1	$\{((a,b,c,e,g),1),$
	$((a,c,e,g),1\}$
f_2	$\{((f,e,c,d),2\}$

(b)

s	$\mathrm{SA}(s)$
1	$\{(a,b),(f,e)\}$
2	$\{(a,c),(f,e)\}$
3	$\{(c,d),(b,a),(e,g)\}$
4	$\{(b,c),(f,g)\}$
5	$\{(c,d),(e,g)\}$
6	$\{(c,e),(b,a)\}$
7	$\{(e,c),(g,f)\}$
8	$\{(e,c)\}$
9	$\{(b,c),(g,e)\}$
10	$\{(c,e)\}$

(c)

	Mapping of flows to slots									
Edge	1	2	3	4	5	6	7	8	9	10
(a,b)	1.1									
(a,c)		1.2								
(b,a)			-			-				
(b,c)				1.1					-	
(c,b)										
(c,d)			2.1		2.1					
(c,e)						1.1				1.2
(d,c)										
(e,c)							2.1	2.1		
(e,g)			1.1		1.2					
(f,e)	2.1	2.1								
(f,g)				-						
(g,e)									-	
(g,f)							-			

(d)

Figure 2.6: Two flows in a network. (a) shows the network topology. The edge labels $x.y$ indicate flow routings; x gives the flow number and y the path number. (b) defines a function ψ that maps flows to flow routings as indicated in (a). (c) shows a slot assignment that is consistent under the graph-based noninterference model. (d) gives a flow mapping that supports the scenario in combination with the given flow routings and the slot assignment. The mapping of micro slots to routing paths is given for each edge in the graph. The boxes indicate the available slots and show the index of the flow and path to which a slot is allocated. Slots with hyphens are allocated by the slot assignment but not mapped to a flow.

queue, we find that after a startup phase containing a finite number of frames, the transmission settles to a stable state in which the forwarding behavior follows a well-defined pattern. This section focuses on single paths being part of a flow's routing. Results for a complete flow can then be obtained by combining the results for the individual paths. For example, the definition of latency defines the latency of a flow to be the maximum latency of the individual paths in the flow's routing.

This section is structured as follows: First, the relationship between frames that are transferred via a path and the consecutive micro slot numbers of the slots during which the frames are handled is established. Then, Proposition 1 shows that the slots during which frames on a path are transferred are arranged according to a regular pattern after the startup phase. For the proof of the proposition, two lemmata are presented and proved. Then, the transmission shift concept which is important in the proofs of the lemmata is revisited and an alternative way of calculating the transmission shift values is given.

The formal analysis starts by numbering the frames of a single flow that travel via a single path of the flow's routing. Then, the notion of transmission opportunities is introduced. A *transmission opportunity* refers to the allocation of resources to a particular path in a flow routing. Each transmission opportunity provides exactly the capacity to transfer one frame. Transmission opportunities are associated with micro slots, i.e. every transmission opportunity occurs during a particular micro slot. Depending on the transmission rates and micro slot durations in effect, it is possible to have multiple transmission opportunities during a single micro slot. Transmission opportunities are referred to by numbers. They are numbered in a way that is compatible with the chronological ordering of the slots that are associated with the transmission opportunities. However, no constraints exist w.r.t. the numbering of transmission opportunities that are part of the same micro slot within a single macro slot.

Definition 13 (Frame numbering and transmission opportunities)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and the triple $(\text{SA}, \text{FMAP}, \psi)$ support the scenario. Let $f \in \mathcal{F}$ and $(p, \kappa) \in \psi(f)$.

1. $\text{THREAD}(f, p)$ *denotes the sequence of frames in flow f that travel on path p. The notation $c_i \in \text{THREAD}(f, p)$ refers to the ith frame w.r.t. transmission order. Thus,* $\text{THREAD}(f, p) = (c_1, c_2, \ldots)$. $\text{THREAD}(f, p)$ *is called the* thread of f defined by p.

2. *The function* $\text{TSL} : \mathcal{F} \times \mathcal{P}_{\mathcal{G}} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *maps transmission opportunities for* $\text{THREAD}(f, p)$ *to consecutive slot numbers.* $\text{TSL}(f, p, j, i) = n$ *specifies that slot $\vec{s}_n$ provides the ith transmission opportunity for* $\text{THREAD}(f, p)$ *on edge (v_j^p, v_{j+1}^p). The definition of* TSL *is:*

$$\text{TSL}(f, p, j, i) := \min \left\{ r \in \mathbb{N} \;\middle|\; \sum_{l=1}^{r} \text{FMAP}(f, p, (v_j^p, v_{j+1}^p), s_l) \geq \text{REM}(i-1, \kappa) + 1 \right\} + \text{DIV}(i-1, \kappa) \cdot n_{\mathcal{S}} \tag{2.21}$$

The functions $\text{DIV}(a, b)$ *and* $\text{REM}(a, b)$ *denote the integer quotient and remainder, respectively, of the division of two integers a and b.*

There are some caveats in the TSL definition worth pointing out: Due to the minimization, $\text{TSL}(f, p, j, i)$ is only defined if p is actually part of f's routing. If so, the minimization is always well defined due to Definition 12. Furthermore, the third argument j must be a valid index w.r.t. the edges in path p, i.e. j must be in the range $\{1, \ldots, |p|\}$.

The TSL function plays an important role in the following sections. Intuitively, it specifies the consecutive number of the slot that provides the ith transmission opportunity for a thread. Some straightforward properties of TSL are of practical use within the following propositions and lemmata. The TDMA principle of repeating macro slots results in a periodicity of the TSL function. This is used to reason about transmission opportunities that belong to the same micro slot within different macro slots. Furthermore, it is easy to see that TSL is monotonic in its fourth parameter. These characteristics are the subject of the following lemma:

Lemma 1 (Properties of TSL)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and the triple $(\text{SA}, \text{FMAP}, \psi)$ support $\mathcal{F}$. Let $(p, \kappa) \in \psi(f)$ be a path in f's routing and $j \in \{1, \ldots, |p|\}$.

1. *The* TSL *function is non-decreasing in its fourth parameter:*

$$\forall i, l \in \mathbb{N} : i \leq l \rightarrow \text{TSL}(f, p, j, i) \leq \text{TSL}(f, p, j, l) \tag{2.22}$$

2. *Let $i \in \mathbb{N}$, $m \in \mathbb{Z}$ and $i + m \cdot \kappa > 0$.* TSL *is periodic due to the fact that there are κ transmission opportunities in each macro frame that consists of $n_{\mathcal{S}}$ micro slots:*

$$\text{TSL}(f, p, j, i + m \cdot \kappa) = \text{TSL}(f, p, j, i) + m \cdot n_{\mathcal{S}} \tag{2.23}$$

PROOF (LEMMA 1)
Let $r_i := \min\{r \in \mathbb{N} \mid \sum_{l=1}^{r} \text{FMAP}(f, p, (v_j^p, v_{j+1}^p), s_l) \geq \text{REM}(i-1, \kappa) + 1\}$. Note that by Definition 12 we have:

$$\sum_{l=1}^{n_{\mathcal{S}}} \text{FMAP}(f, p, (v_j^p, v_{j+1}^p), s_l) = \kappa \tag{2.24}$$

Since $\text{REM}(i-1, \kappa) + 1 \in \{1, \ldots, \kappa\}$ for all $i \in \mathbb{N}$, we obtain $r_i \leq n_{\mathcal{S}}$.

1. Let $i, l \in \mathbb{N}$ and $i \leq l$. Two cases can be distinguished:
   - $\text{DIV}(i-1, \kappa) < \text{DIV}(l-1, \kappa)$:

     Since we are dealing with integers, this is equivalent to $\text{DIV}(i-1, \kappa) + 1 \leq \text{DIV}(l-1, \kappa)$. Thus, we have:

$$\begin{aligned} \text{TSL}(f, p, j, i) &= r_i + \text{DIV}(i-1, \kappa) \cdot n_{\mathcal{S}} \\ &\leq n_{\mathcal{S}} + \text{DIV}(i-1, \kappa) \cdot n_{\mathcal{S}} \\ &= (\text{DIV}(i-1, \kappa) + 1) \cdot n_{\mathcal{S}} \\ &\leq \text{DIV}(l-1, \kappa) \cdot n_{\mathcal{S}} \\ &< \text{DIV}(l-1, \kappa) \cdot n_{\mathcal{S}} + r_l \\ &= \text{TSL}(f, p, j, l) \end{aligned} \tag{2.25}$$

- $\text{DIV}(i-1,\kappa) = \text{DIV}(l-1,\kappa)$:

  Let $\alpha = \text{DIV}(i-1,\kappa) = \text{DIV}(l-1,\kappa)$. Then, we have:

  $$\alpha \cdot \kappa + \text{REM}(i-1,\kappa) = i-1 \tag{2.26}$$
  $$\alpha \cdot \kappa + \text{REM}(l-1,\kappa) = l-1 \tag{2.27}$$

  Comparing these equations, we obtain $\text{REM}(i-1,\kappa)+1 \leq \text{REM}(l-1,\kappa)+1$ which implies $r_i \leq r_l$. Thus, we have:

  $$\text{TSL}(f,p,j,i) = r_i + \alpha \leq r_l + \alpha = \text{TSL}(f,p,j,l) \tag{2.28}$$

  Note that $\text{DIV}(i-1,\kappa) > \text{DIV}(l-1,\kappa)$ is impossible due to $i \leq l$.

2. Let $i \in \mathbb{N}$, $m \in \mathbb{Z}$ and $i + m \cdot \kappa > 0$. Due to $\text{REM}(i+m\cdot\kappa-1,k) = \text{REM}(i-1,\kappa)$ and $\text{DIV}(i+m\cdot\kappa-1,\kappa) = \text{DIV}(i-1,\kappa)+m$, we have:

   $$\begin{aligned} \text{TSL}(f,p,j,i+m\cdot\kappa) &= r_{(i+m\cdot\kappa)} + \text{DIV}(i+m\cdot\kappa-1,k)\cdot n_{\mathcal{S}} \\ &= r_i + (\text{DIV}(i-1,k)+m)\cdot n_{\mathcal{S}} \\ &= \text{TSL}(f,p,j,i) + m\cdot n_{\mathcal{S}} \end{aligned} \tag{2.29}$$

   This property will be referred to as *periodicity of* TSL in the following. □

For our analysis, it is not relevant at which time (i.e. in which macro slot) the flow actually starts. Thus, it is assumed that the flow starts at the beginning of macro slot $\underline{s}_1$. For every path (p,κ) of the flow routing, the source node a will send κ frames in each macro slot. $\text{TSL}(f,p,1,i)$ gives the consecutive micro slot number in which a will send frame $c_i \in \text{THREAD}(f,p)$ to node v_2^p. Once frames arrive at v_2^p, it can forward them. Note that just after the start of the flow, some transmission opportunities at forwarding nodes on the path can possibly not be used if no frames are available at the time. Thus, we need to express formally at which transmission opportunity a frame c_i will be transmitted over a given edge that is part of p. In the following it is also assumed that the nodes forward the frames in FIFO order and that they forward in a greedy fashion, i.e. the nodes transmit frames if there are frames in the local queue. There are two conditions that decide whether a frame $c_i \in \text{THREAD}(f,p)$ to be forwarded via an edge e can be transmitted: Frame c_{i-1} must already have been transmitted via e (with the exception of the first frame, i.e. $i=1$). This is required by the FIFO ordering. Furthermore, c_i can only be forwarded over $e=(v,w)$ if c_i has already arrived at v, i.e. c_i has already been transferred over the previous edge. These considerations lead to the following definition:

Definition 14 (Mapping of frames to transmission opportunities)
Let $\mathcal{G}=(\mathcal{V},\mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario supported by $(\text{SA},\text{FMAP},\psi)$. Let $f \in \mathcal{F}$ and $(p,\kappa) \in \psi(f)$. The transmission function $\text{TRANS} : \mathcal{F}\times\mathcal{P}_{\mathcal{G}}\times\mathbb{N}\times\mathbb{N} \to \mathbb{N}$ describes the mapping of frames to transmission opportunities over a given edge. $\text{TRANS}(f,p,j,i)$ gives the number of the transmission opportunity in which frame $c_i \in \text{THREAD}(f,p)$ is forwarded via (v_j^p, v_{j+1}^p):

$$\text{TRANS}(f,p,j,i) := \begin{cases} i & j=1 \\ \min\{l \in \mathbb{N} \mid C_1(f,p,i,j,l)\} & j>1, i=1 \\ \min\{l \in \mathbb{N} \mid C_1(f,p,i,j,l) \wedge C_2(f,p,i,j,l)\} & j>1, i>1 \end{cases} \tag{2.30}$$

The predicates C_1 and C_2 formalize the constraints that must be met for a frame to be forwardable. $C_1(f,p,i,j,l)$ is true if c_i which is to be forwarded via (v_j^p, v_{j+1}^p) during transmission opportunity l has already arrived at that time at v_j^p, i.e. has been forwarded over (v_{j-1}^p, v_j^p) during a transmission opportunity preceding l:

$$C_1(f,p,i,j,l) \equiv \text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,i)) < \text{TSL}(f,p,j,l) \tag{2.31}$$

$C_2(f,p,i,j,l)$ states that if c_i is to be forwarded via (v_j^p, v_{j+1}^p) during transmission opportunity l, c_{i-1} must have been forwarded via (v_j^p, v_{j+1}^p) during a previous transmission opportunity:

$$C_2(f,p,i,j,l) \equiv \text{TRANS}(f,p,j,i-1) < l \tag{2.32}$$

The definition of TRANS is heavily based on recursion. This approach was chosen deliberately because it closely resembles the behavior exposed by the nodes. Thus, it is trivial to verify that an actual system will actually behave according to the definition. Every system that implements FIFO queueing and greedy forwarding (i.e. frames are always forwarded when possible) obviously conforms to the TRANS definition.

As mentioned above, we assume that the source node continuously sends frames. Thus, the source node will send one frame in each transmission opportunity that comes up. At the forwarding nodes, TRANS decides in which transmission opportunities and thus slots the frames get forwarded. Due to the periodicity of the slot allocations in consecutive macro slots, we can expect that forwarding follows a pattern: If a frame arrives during a particular micro slot s_i of macro slot $\underline{s}_k$ and is forwarded in micro slot s_l (possibly in a later macro slot), the frame arriving in micro slot s_i during the following macro slot $\underline{s}_{k+1}$ will also be sent out in micro slot s_l. This expectation turns out to be true for all frames that come after an initial sequence of frames that make up a startup phase. Thus, it is reasonable to introduce equivalence classes of frames based on the transmission opportunities they are sent over an edge. This is the subject of the following proposition.

Proposition 1 (Frame transmission equivalence classes within a thread)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ be a scenario supported by $(\text{SA}, \text{FMAP}, \psi)$. Let $f \in \mathcal{F}$ and $(p,\kappa) \in \psi(f)$. Let $\mathcal{M}$ be the set of integers larger than $\kappa \cdot (|p|-1)$, i.e. $\mathcal{M} = \{i \in \mathbb{N} \mid i > \kappa \cdot (|p|-1)\}$. Let $j \in \{1,\ldots,|p|\}$ be the index of an edge in p. Consider the family of equivalence relations $\sim_j \subseteq \mathcal{M}^2$ defined by the κ transmission opportunities within a macro slot in which the frames $c_i \in \text{THREAD}(f,p)$ are forwarded via the edge $(v_j^p, v_{j+1}^p) \wr p$:

$$\sim_j := \{(i,l) \in \mathcal{M}^2 \mid \text{REM}(\text{TRANS}(f,p,j,i),\kappa) = \text{REM}(\text{TRANS}(f,p,j,l),\kappa)\} \tag{2.33}$$

$\sim_j$ is obviously an equivalence relation, thus it generates a set of equivalence classes:

$$[i]_j := \{l \in \mathcal{M} \mid i \sim_j l\} \tag{2.34}$$

For the equivalence classes, the following holds:

1. *The elements of the equivalence classes are the frame numbers within distance of multiples of κ:*

$$\forall i \in \mathcal{M}\ \forall j \in \{1,\ldots,|p|\} : [i]_j = \{\min [i]_j + m \cdot \kappa \mid m \in \mathbb{N}_0\} \tag{2.35}$$

2. *The equivalence classes are the same at every edge in p:*

$$\forall j, l \in \{1, \ldots, |p|\}\ \forall i \in \mathcal{M} : [i]_j = [i]_l \tag{2.36}$$

Thus, in particular $[i]_1 = [i]_{|p|}$ *for every* $i \in \mathcal{M}$ *and we define* $\sim := \sim_1$ *and* $[i] := [i]_1$.

3. *There are* κ *equivalence classes.*

For technical reasons, the definition of the equivalence relation $\sim$ is based on the frame numbering of the frames within a thread. What we are actually interested in is the way frames are handled whose numbers belong to the same equivalence class. In the following, we will find that these frames are transferred in the same micro slots at all nodes. This also means they experience the same delay which is important when reasoning about the latency a flow experiences when transferred through the network.

Before giving the proof of Proposition 1, two lemmata are presented. The important statement of these lemmata is that starting with frame $c_{\kappa\cdot(j-1)}$, all transmission opportunities at node v_j^p will be used for forwarding frames. Thus, at this point node v_j^p's transmission queue is never empty for any transmission opportunity that comes up. Lemma 2 considers all frames for a fixed node on the path. Then, Lemma 3 extends the statement to all nodes on the path that need to forward frames.

In the lemmata, the notion of a transmission shift is used, which is introduced in the following definition. The transmission shift value gives the difference between the transmission opportunity indices in which a frame is received and forwarded, respectively, at a node:

Definition 15 (Transmission shift)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *be the connectivity graph and* $\mathcal{F}$ *a scenario supported by* $(\text{SA}, \text{FMAP}, \psi)$. *Let* $f \in \mathcal{F}$ *be a flow,* $(p, \kappa) \in \psi(f)$ *a path in its routing and* $j \in \{2, \ldots, |p|\}$ *the index of a node being part of p.*

1. Frame i's transmission shift d_i^j *at node* v_j^p *is defined as:*

$$d_i^j := \text{TRANS}(f, p, j, i) - \text{TRANS}(f, p, j-1, i) \tag{2.37}$$

2. d^j *is called the* transmission shift at node v_j^p*:*

$$d^j := d^j_{\kappa\cdot(j-1)} \tag{2.38}$$

The meaning of the constant $\kappa \cdot (j-1)$ in the definition of d^j will be clarified in the proofs of the Lemmata 2 and 3. Intuitively, it gives the number of the frame at which the transmission of the flow has reached the stable state. Once this state is reached, the slots used by node v_j^p for forwarding frames follow a pattern and the transmission shift value does not change any more for later frames.

Figure 2.7 gives an example that illustrates the flow of frames belonging to a single path in a flow's routing over the first three edges in the path. The slot structure for each edge is given by the box rows. Individual boxes represent slots. For each slot, the slot number as well as the consecutive slot number are given. Furthermore, the transmission opportunities provided by the slots are indicated along with the frames that are transmitted during these transmission opportunities. The transmission shifts can be calculated by comparing the transmission opportunitiy numbers of a frame between two consecutive edges. For example, $d_3^4 = 6 - 4 = 2$.

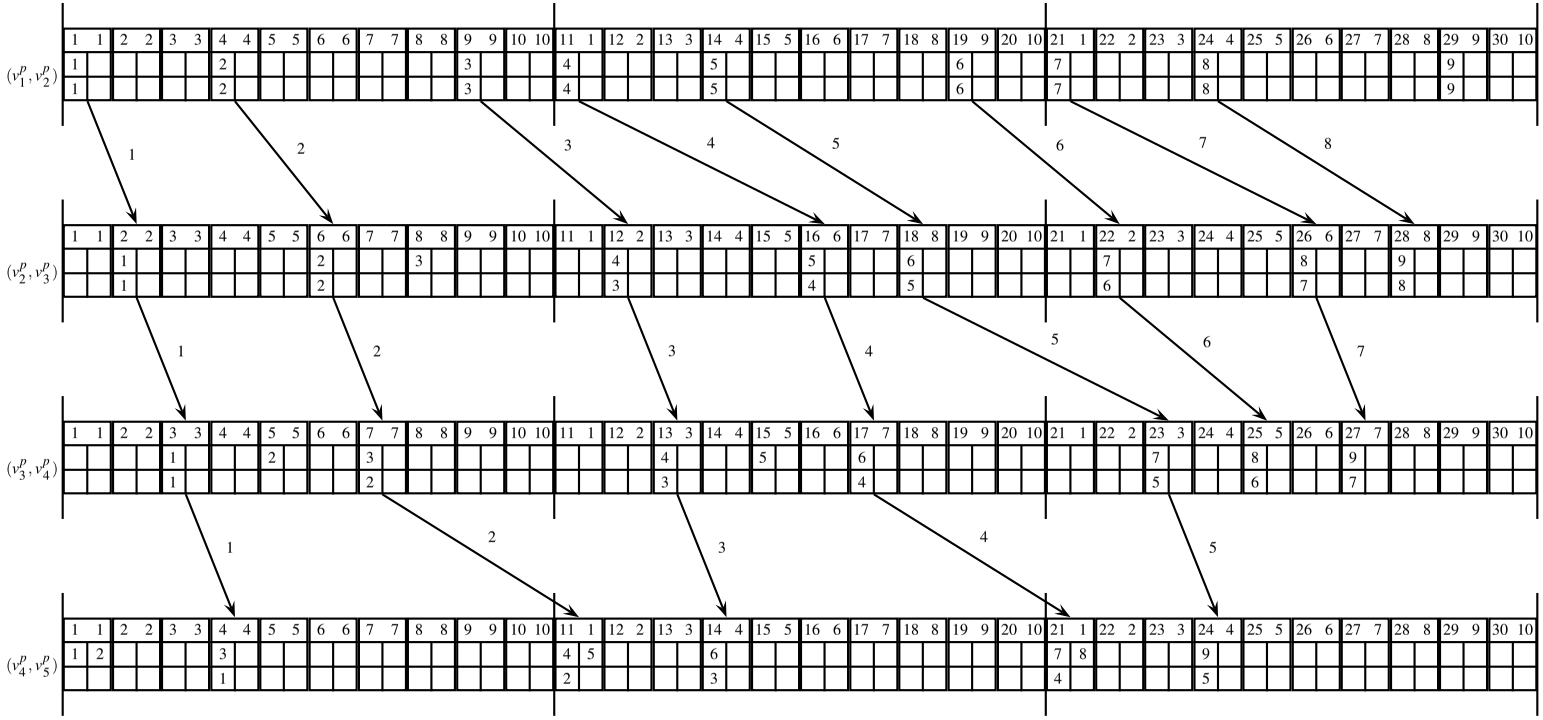


Figure 2.7: Transmission of the first few frames of a thread via the first four edges in the path. Frames are assumed to be forwarded in FIFO order at each node. Three consecutive macro slots are shown, frames start to arrive in the first macro slot shown.

Lemma 2
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario supported by (SA, FMAP, ψ). *Let $f \in \mathcal{F}$ be a flow, $(p, \kappa) \in \psi(f)$ a path in f's routing, $j \in \{2, \ldots, |p|\}$ and $m = \kappa \cdot (j-2)+1$. If the condition*

$$\forall l \in \mathbb{N} : (l \geq m) \rightarrow (\text{TRANS}(f, p, j-1, l) = \text{TRANS}(f, p, j-1, m) + l - m) \tag{2.39}$$

holds, then for all $i \geq \kappa \cdot (j-1)$, the following is true:

1. *Each frame c_i arrives early enough so it can potentially be forwarded during transmission opportunity* $\text{TRANS}(f, p, j-1, i) + d^j$ *on the outgoing edge:*

$$\text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i)) < \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i) + d^j) \tag{2.40}$$

2. *Starting with frame $c_{\kappa \cdot (j-1)}$ all frames will be forwarded during consecutive transmission opportunities:*

$$\text{TRANS}(f, p, j, i) = \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + i - \kappa \cdot (j-1) \tag{2.41}$$

3. *The transmission shift for c_i is d^j, i.e. $d_i^j = d^j$.*

PROOF (LEMMA 2)
The proof is by induction on i. Assume $i = \kappa \cdot (j-1)$.

1. We have:

$$\begin{aligned} &\text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i)) \\ &\qquad = \quad \text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, \kappa \cdot (j-1))) \\ &\qquad \overset{\text{Def. 14}}{<} \quad \text{TSL}(f, p, j, \text{TRANS}(f, p, j, \kappa \cdot (j-1))) \\ &\qquad \overset{\text{Def. 15}}{=} \quad \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, \kappa \cdot (j-1)) + d^j) \\ &\qquad = \quad \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i) + d^j) \end{aligned} \tag{2.42}$$

2. Trivially true for $i = \kappa \cdot (j-1)$.

3. Trivially true for $i = \kappa \cdot (j-1)$.

Suppose that claims 1-3 of the lemma hold for all i' not greater than some fixed i. We need to prove:

1. $\text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i+1)) < \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i+1) + d^j)$
2. $\text{TRANS}(f, p, j, i+1) = \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + (i+1) - \kappa \cdot (j-1)$
3. $d_{i+1}^j = d^j$

The proofs follow:

1. Consider frame $c_{i+1-\kappa}$. Due to Equation 2.39, $c_{i+1-\kappa}$ is forwarded by v_{j-1}^p during the same slot as c_i, but in the previous macro slot.

   First, we prove $d_{i+1-\kappa}^j \leq d^j$. There are two cases:

a) $(i+1-k) \in \{m, \ldots, \kappa \cdot (j-1)\}$

For all $l \in \{m, \ldots, \kappa \cdot (j-1)\}$ it can be proved by induction that $d_l^j \leq d^j$. Assume $l = \kappa \cdot (j-1)$. Trivially, we have $d_l^j = d^j$.

Now suppose that $d_l^j \leq d^j$ and consider $l-1$. We have:

$$\begin{aligned} d_{l-1}^j &\overset{\text{Def. 15}}{=} \text{TRANS}(f,p,j,l-1) - \text{TRANS}(f,p,j-1,l-1) \\ &\overset{\text{Eq. 2.39}}{=} \text{TRANS}(f,p,j,l-1) - (\text{TRANS}(f,p,j-1,m) + (l-1) - m) \\ &\overset{\text{Def. 14}}{<} \text{TRANS}(f,p,j,l) - (\text{TRANS}(f,p,j-1,m) + l - m) + 1 \\ &\overset{\text{Eq. 2.39}}{=} \text{TRANS}(f,p,j,l) - \text{TRANS}(f,p,j-1,l) + 1 \\ &\overset{\text{Def. 15}}{=} d_l^j + 1 \\ &\overset{\text{I.H.}}{\leq} d^j + 1 \end{aligned} \tag{2.43}$$

Considering that the transmission shifts are integer values we obtain $d_{l-1}^j \leq d^j$.

b) $i+1-k > \kappa \cdot (j-1)$

Note that $i+1-\kappa \leq i$. Claim 3 of the induction hypothesis yields $d_{i+1-\kappa}^j = d^j$.

Combining this result with the transmission shift definition yields the following:

$$\begin{aligned} \text{TRANS}(f,p,j,i+1-\kappa) &\overset{\text{Def. 15}}{=} \text{TRANS}(f,p,j-1,i+1-\kappa) + d_{i+1-\kappa}^j \\ &\leq \text{TRANS}(f,p,j-1,i+1-\kappa) + d^j \end{aligned} \tag{2.44}$$

We use this relation to finally prove claim 1:

$$\begin{aligned} &\text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,i+1)) \\ &\overset{\text{Eq. 2.39}}{=} \text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,i+1-\kappa)+\kappa) \\ &\overset{\text{Lemma 1}}{=} \text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,i+1-\kappa)) + n_{\mathcal{S}} \\ &\overset{\text{Def. 14}}{<} \text{TSL}(f,p,j,\text{TRANS}(f,p,j,i+1-\kappa)) + n_{\mathcal{S}} \\ &\overset{\text{Lemma 1}}{\leq} \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,i+1-\kappa)+d^j) + n_{\mathcal{S}} \\ &\overset{\text{Lemma 1}}{=} \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,i+1-\kappa)+\kappa+d^j) \\ &\overset{\text{Eq. 2.39}}{=} \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,i+1)+d^j) \end{aligned} \tag{2.45}$$

2. Let $l := \text{TRANS}(f,p,j,\kappa \cdot (j-1)) + (i+1) - j \cdot (\kappa-1)$. We have:

$$\begin{aligned} l &= \text{TRANS}(f,p,j,\kappa \cdot (j-1)) + (i+1) - \kappa \cdot (j-1) \\ &> \text{TRANS}(f,p,j,\kappa \cdot (j-1)) + i - \kappa \cdot (j-1) \\ &\overset{\text{I.H.}}{=} \text{TRANS}(f,p,j,i) \end{aligned} \tag{2.46}$$

Thus, l meets condition C_2 of Definition 14. Also note that because l is obviously the smallest integer that meets C_2, it is a lower bound for $\text{TRANS}(f,p,j,i+1)$.

Left to prove is that l also fulfills condition C_1 of Definition 14.

$$\begin{aligned} & l - \text{TRANS}(f, p, j-1, i+1) \\ & \quad \overset{\text{Eq. 2.39}}{=} \; l - (\text{TRANS}(f, p, j-1, m) + (i+1) - m) \\ & \quad = \; l - (\text{TRANS}(f, p, j-1, m) + i - m) - 1 \\ & \quad \overset{\text{Eq. 2.39}}{=} \; l - \text{TRANS}(f, p, j-1, i) - 1 \\ & \quad \overset{\text{Eq. 2.46}}{>} \; \text{TRANS}(f, p, j, i) - \text{TRANS}(f, p, j-1, i) - 1 \end{aligned} \tag{2.47}$$

Considering that we are dealing with integers:

$$\begin{aligned} l - \text{TRANS}(f, p, j-1, i+1) & \overset{\text{Eq. 2.47}}{\geq} \text{TRANS}(f, p, j, i) - \text{TRANS}(f, p, j-1, i) \\ & \overset{\text{I.H.}}{=} d^j \end{aligned} \tag{2.48}$$

Thus, $\text{TRANS}(f, p, j-1, i+1) + d^j \leq l$. This relation finally allows to verify C_1 as of Definition 14:

$$\begin{aligned} & \text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i+1)) \\ & \quad \overset{\text{claim 1}}{<} \; \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i+1) + d^j) \\ & \quad \overset{\text{Lemma 1}}{\leq} \; \text{TSL}(f, p, j, l) \end{aligned} \tag{2.49}$$

We conclude that l is the minimal transmission opportunity number that meets conditions C_1 and C_2 of Definition 14 and find:

$$\text{TRANS}(f, p, j, i+1) = l = \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + (i+1) - j \cdot (\kappa - 1) \tag{2.50}$$

3. Making use of the already proved claim 2, it is straightforward to verify claim 3:

$$\begin{aligned} d^j_{i+1} & \overset{\text{Def. 15}}{=} \text{TRANS}(f, p, j, i+1) - \text{TRANS}(f, p, j-1, i+1) \\ & \overset{\text{Eqs. 2.39, 2.50}}{=} \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + (i+1) - \kappa \cdot (j-1) \\ & \qquad - (\text{TRANS}(f, p, j-1, m) + (i+1) - m) \\ & \overset{\text{I.H., Eq. 2.39}}{=} \text{TRANS}(f, p, j, i) + 1 - (\text{TRANS}(f, p, j-1, i) + 1) \\ & \overset{\text{I.H.}}{=} d^j \end{aligned} \tag{2.51}$$

This concludes the induction step and completes the proof of the Lemma. □

Lemma 2 gives properties of the transmission opportunities used for forwarding frames at some fixed node v^p_j of some path p that hold when the premise as stated in the Lemma is true. The next step is to show that these properties are actually true for all forwarding nodes v^p_j, $j \in \{2, \ldots, |p|\}$. This is what the following Lemma is about:

Lemma 3
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario supported by $(\text{SA}, \text{FMAP}, \psi)$. Let f be a flow in $\mathcal{F}$ and $(p, \kappa) \in \psi(f)$ a path in its routing. For all $j \in \{2, \ldots, |p|\}$ and all $i \in \mathbb{N}$, $i \geq \kappa \cdot (j-1)$ the following is true:

1. *Each frame c_i arrives early enough so it can potentially be forwarded during transmission opportunity* $\text{TRANS}(f, p, j-1, i) + d^j$ *on the outgoing edge:*

$$\text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i)) < \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i) + d^j) \tag{2.52}$$

2. *Starting with frame* $c_{\kappa\cdot(j-1)}$ *all frames will be forwarded during consecutive transmission opportunities:*

$$\text{TRANS}(f, p, j, i) = \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + i - \kappa \cdot (j-1) \tag{2.53}$$

3. *The transmission shift for* c_i *is* d^j*, i.e.* $d_i^j = d^j$.

PROOF (LEMMA 3)
The proof is by induction on j, thereby making use of Lemma 2 in each step. Assume $j = 2$. Due to $\text{TRANS}(f, p, j-1, i) = \text{TRANS}(f, p, 1, i) = i$, the premise of Lemma 2 as stated in Equation 2.39 is trivially true. Applying the Lemma yields the desired results.

Now assume that for some fixed j the claims of Lemma 3 hold, i.e. for all $i \geq \kappa \cdot (j-1)$ the following statements are true:

1. $\text{TSL}(f, p, j-1, \text{TRANS}(f, p, j-1, i)) < \text{TSL}(f, p, j, \text{TRANS}(f, p, j-1, i) + d^j)$
2. $\text{TRANS}(f, p, j, i) = \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + i - \kappa \cdot (j-1)$
3. $d_i^j = d^j$

Consider node v_{j+1}^p. The precondition for Lemma 2 follows immediately from claim 2 of the above induction hypothesis since $\kappa \cdot (j-1) + 1 > \kappa \cdot (j-1)$. Thus, we can apply Lemma 2 at node $j+1$ and obtain the results that complete the induction step. □

PROOF (PROPOSITION 1)
Recall the proposition that is to be proved: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ be a scenario supported by $(\text{SA}, \text{FMAP}, \psi)$. Let $f \in \mathcal{F}$ and $(p, \kappa) \in \psi(f)$, $\mathcal{M} = \{i \in \mathbb{N} \mid i > \kappa \cdot (|p| - 1)\}$ and $j \in \{1, \ldots, |p|\}$. The equivalence relation

$$\sim_j := \{(i, l) \in \mathcal{M}^2 \mid \text{REM}(\text{TRANS}(f, p, j, i), \kappa) = \text{REM}(\text{TRANS}(f, p, j, l), \kappa)\} \tag{2.54}$$

generates a set of equivalence classes

$$[i]_j := \{l \in \mathcal{M} \mid i \sim_j l\} \tag{2.55}$$

for which the following claims are to be proved:

1. The elements of the equivalence classes are the frame numbers within distance of multiples of κ:

$$\forall i \in \mathcal{M}\ \forall j \in \{1, \ldots, |p|\} : [i]_j = \{\min [i]_j + m \cdot \kappa \mid m \in \mathbb{N}_0\} \tag{2.56}$$

2. The equivalence classes are the same at every edge in p:

$$\forall j, l \in \{1, \ldots, |p|\}\ \forall i \in \mathcal{M} : [i]_j = [i]_l \tag{2.57}$$

Thus, in particular $[i]_1 = [i]_{|p|}$ for every $i \in \mathcal{M}$ and we define $\sim := \sim_1$ and $[i] := [i]_1$.

3. There are κ equivalence classes.

The proofs for the claims stated above follow.

1. The proof is conducted by showing that each set is a subset of the other.
   a) This part of the proof establishes $\{\min [i]_j + m \cdot \kappa \mid m \in \mathbb{N}_0\} \subseteq [i]_j$. Consider these two cases:
      i. $j = 1$

      Recall that $\text{TRANS}(f, p, 1, i) = i$ for all $i \in \mathbb{N}$. By definition of $\sim_1$, we have:

      $$\sim_1 = \{(i, l) \in \mathcal{M}^2 \mid \text{REM}(i, \kappa) = \text{REM}(l, \kappa)\} \tag{2.58}$$

      Let $i \in \mathcal{M}$, $m \in \mathbb{N}_0$, $i^* = \min [i]_j$. Consider frame $i^* + \kappa \cdot m$:

      $$\text{REM}(i^* + \kappa \cdot m, \kappa) = \text{REM}(i^*, \kappa) = \text{REM}(i, \kappa) \tag{2.59}$$

      Hence, $(i^* + \kappa \cdot m) \in [i]_1$ by definition of $\sim_1$.

      ii. $j \in \{2, \ldots, |p|\}$

      By Lemma 3, we have for $l' := \kappa \cdot (j-1)$:

      $$\forall l \in \mathbb{N}, l \geq l' : \text{TRANS}(f, p, j, l) = \text{TRANS}(f, p, j, l') + l - l' \tag{2.60}$$

      Let $i \in \mathcal{M}$ (note that $i \geq l'$), $m \in \mathbb{N}$ and $i^* = \min [i]_j$.

      $$\begin{aligned} &\text{REM}(\text{TRANS}(f, p, j, i^* + \kappa \cdot m), \kappa) \\ &\overset{\text{Eq. 2.60}}{=} \text{REM}(\text{TRANS}(f, p, j, l') + (i^* + \kappa \cdot m) - l'), \kappa) \\ &= \text{REM}(\text{TRANS}(f, p, j, l') + i^* - l', \kappa) \\ &\overset{\text{Eq. 2.60}}{=} \text{REM}(\text{TRANS}(f, p, j, i^*), \kappa) \\ &= \text{REM}(\text{TRANS}(f, p, j, i), \kappa) \end{aligned} \tag{2.61}$$

      Thus, by definition of $\sim_j$, we have $(i^* + \kappa \cdot m) \in [i]_j$.

   b) Let $i \in \mathcal{M}$, $l \in [i]_j$ and $i^* = \min [i]_j$. By definition of $\sim_j$, we have:

   $$\text{REM}(\text{TRANS}(f, p, j, l), \kappa) = \chi = \text{REM}(\text{TRANS}(f, p, j, i^*), \kappa) \tag{2.62}$$

   Thus, there exist $\iota, \lambda \in \mathbb{N}_0$, such that

   $$\iota \cdot \kappa + \chi = \text{TRANS}(f, p, j, i^*) \tag{2.63}$$
   $$\lambda \cdot \kappa + \chi = \text{TRANS}(f, p, j, l) \tag{2.64}$$

   We argue that $(\lambda - \iota) \cdot \kappa = l - i^*$. There are two cases:

i. $j = 1$

Then, due to $\text{TRANS}(f, p, 1, m) = m$, we have:

$$\iota \cdot \kappa + \chi = \text{TRANS}(f, p, j, i^*) = i^* \tag{2.65}$$
$$\lambda \cdot \kappa + \chi = \text{TRANS}(f, p, j, l) = l \tag{2.66}$$

Calculating the difference of these equations yields $(\lambda - \iota) \cdot \kappa = l - i^*$.

ii. $j \in \{2, \ldots, |p|\}$

By Lemma 3, we have:

$$\begin{aligned} \iota \cdot \kappa + \chi &= \text{TRANS}(f, p, j, i^*) \\ &= \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + i^* - \kappa \cdot (j-1) \end{aligned} \tag{2.67}$$

$$\begin{aligned} \lambda \cdot \kappa + \chi &= \text{TRANS}(f, p, j, l) \\ &= \text{TRANS}(f, p, j, \kappa \cdot (j-1)) + l - \kappa \cdot (j-1) \end{aligned} \tag{2.68}$$

Again, substracting these equations yields $(\lambda - \iota) \cdot \kappa = l - i^*$.

Due to the fact that $l \geq i^*$, we also have $\lambda \geq \iota$ and thus $\lambda - \iota \geq 0$. Hence, $l = i^* + (\lambda - \iota) \cdot \kappa$ and $l \in \{\min [i]_j + m \cdot \kappa \mid m \in \mathbb{N}_0\}$.

2. It is an immediate consequence of claim 1 of the proposition that the equivalence classes are the same at each node.

3. It is easy to see that there are exactly κ equivalence classes. Consider the first κ frame numbers in $\mathcal{M}$, i.e. the frames $c_{\kappa \cdot (|p|-1)}, \ldots, c_{\kappa \cdot |p|-1}$. By Lemma 3, we have for $i \in \{\kappa \cdot (|p|-1), \ldots, \kappa \cdot |p| - 1\}$:

$$\text{REM}(\text{TRANS}(f, p, 1, i), \kappa) = \text{REM}(i, \kappa) = i - \kappa \cdot (|p| - 1) \tag{2.69}$$

Thus:

$$\forall i, l \in \{\kappa \cdot (|p|-1), \ldots, \kappa \cdot |p| - 1\} : i \neq l \rightarrow [i] \neq [l] \tag{2.70}$$

At this point we have proved that there are at least κ equivalence classes. Due to the fact that $\text{REM}(i, \kappa) \in \{0, \ldots, \kappa - 1\}$ for all $i \in \mathbb{N}$, there cannot be more than κ equivalence classes. □

Proposition 1 states that beginning with frame $c_{\kappa \cdot (|p|-1)}$, the transmission of the frames within a thread reaches a stable state. Each node handles frames with distance κ at transmission opportunities that are also in distance κ of each other. Thus, these frames are transmitted during the same micro slots. Moreover, after stabilization, all frames share the common transmission shift d^j. Obviously, the transmission shift must be large enough so that all incoming frames arrive early enough to catch their respective outgoing transmission opportunities. However, it is unclear whether d^j is actually the minimal shift that satisfies this property. The following proposition answers this question.

Proposition 2 (Minimality of the transmission shift)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario supported by $(\text{SA}, \text{FMAP}, \psi)$. *Let f be a flow in $\mathcal{F}$ and $(p, \kappa) \in \psi(f)$ a path in f's routing. For all $j \in \{2, \ldots, |p|\}$, the transmission shift at node j can be characterized as follows:*

$$d^j = \min\{m \in \mathbb{N} \mid \forall i \in \mathbb{N} : \text{TSL}(f, p, j-1, i) < \text{TSL}(f, p, j, i+m)\} \tag{2.71}$$

PROOF (PROPOSITION 2)
Let $j \in \{2, \ldots, |p|\}$, $\mathcal{M} = \{m \in \mathbb{N} \mid \forall i \in \mathbb{N} : \text{TSL}(f, p, j-1, i) < \text{TSL}(f, p, j, i+m)\}$ and $\hat{d}^j = \min \mathcal{M}$. We prove Equation 2.71 by first showing that $d^j \in \mathcal{M}$ and then $\hat{d}^j \geq d^j$.

1. Claim: $\forall i \in \mathbb{N} : \text{TSL}(f, p, j-1, i) < \text{TSL}(f, p, j, i + d^j)$

   Due to the frames using consecutive transmission opportunities after the startup phase, we observe that for each transmission opportunity, there is a frame index describing the frame that is transferred during that transmission opportunity. Formally:

$$\begin{aligned} &\forall j' \in \{1, \ldots, |p|\}\ \forall l \in \mathbb{N}, l \geq \text{TRANS}(f, p, j', \kappa \cdot (j'-1) + 1) : \\ &\qquad \exists m \in \mathbb{N} : (m \geq \kappa \cdot (j'-1)) \wedge (\text{TRANS}(f, p, j', m) = l) \end{aligned} \tag{2.72}$$

   For the proof consider these two cases:

   - If $j' = 1$, choose $m := l$. Equation 2.72 is trivially true due to Def. 14:

$$\text{TRANS}(f, p, 1, m) = m = l \tag{2.73}$$

   - In the case of $j' \in \{2, \ldots, |p|\}$, a suitable value for m is obtained by reconsidering claim 2 of Lemma 3:

$$m := l - \text{TRANS}(f, p, j', \kappa \cdot (j'-1)) + \kappa \cdot (j'-1) \tag{2.74}$$

     Note that $m \geq \kappa \cdot (j'-1)$. Thus, we obtain:

$$\begin{aligned} &\text{TRANS}(f, p, j', m) \\ &\quad \overset{\text{Lemma 3}}{=} \text{TRANS}(f, p, j', \kappa \cdot (j'-1)) + m - \kappa \cdot (j'-1) \\ &\quad \overset{\text{Def. } m}{=} \text{TRANS}(f, p, j', \kappa \cdot (j'-1)) - \kappa \cdot (j'-1) \\ &\qquad\quad + l - \text{TRANS}(f, p, j', \kappa \cdot (j'-1)) + \kappa \cdot (j'-1) \\ &\quad = \quad l \end{aligned} \tag{2.75}$$

   Now we prove the claim. The idea is that for transmission opportunities after the startup phase, the statement follows from the the above observation and claim 1 of Lemma 3. We invoke the periodicity of TSL to also proof the claim for the transmission opportunities that are part of the startup phase. Let $i \in \mathbb{N}$ be an arbitrary transmission opportunity

index and $x \in \mathbb{N}$ such that $\kappa \cdot x \geq \text{TRANS}(f, p, j-1, \kappa \cdot (j-2))$. We have:

$$\begin{aligned}
\text{TSL}(f,p,j-1,i) &= \text{TSL}(f,p,j-1,i) + x \cdot n_{\mathcal{S}} - x \cdot n_{\mathcal{S}} \\
&\overset{\text{Lemma 1}}{=} \text{TSL}(f,p,j-1,i + x \cdot \kappa) - x \cdot n_{\mathcal{S}} \\
&\overset{\text{Eq. 2.72}}{=} \text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,m)) - x \cdot n_{\mathcal{S}} \\
&\overset{\text{Lemma 3}}{<} \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,m) + d^j) - x \cdot n_{\mathcal{S}} \\
&= \text{TSL}(f,p,j,i + x \cdot \kappa + d^j) - x \cdot n_{\mathcal{S}} \\
&\overset{\text{Lemma 1}}{=} \text{TSL}(f,p,j,i + d^j) + x \cdot n_{\mathcal{S}} - x \cdot n_{\mathcal{S}} \\
&= \text{TSL}(f,p,j,i + d^j)
\end{aligned} \tag{2.76}$$

2. Claim: $d^j \leq \hat{d}^j$

   The proof is by contradiction, so assume $d^j > \hat{d}^j$.

   First, we prove by induction that $d^j_i > \hat{d}^j$ for all $i \in \{1, \ldots, \kappa \cdot (j-1)\}$. Consider $i = \kappa \cdot (j-1)$. Trivially, $d^j_i = d^j_{\kappa \cdot (j-1)} = d^j > \hat{d}^j$.

   Now suppose that $d^j_i > \hat{d}^j$ for some fixed $i \in \{2, \ldots, \kappa \cdot (j-1)\}$ and consider $i-1$. Due to the induction hypothesis, we have:

$$\begin{aligned}
\text{TRANS}(f,p,j,i) &= \text{TRANS}(f,p,j-1,i) + d^j_i \\
&> \text{TRANS}(f,p,j-1,i) + \hat{d}^j
\end{aligned} \tag{2.77}$$

   However, by definition of $\hat{d}^j$, we know that

$$\text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,i)) < \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,i) + \hat{d}^j) \tag{2.78}$$

   This potentially allows transmission of frame c_i on edge (v^p_j, v^p_{j+1}) in transmission opportunity $\text{TRANS}(f,p,j-1,i) + \hat{d}^j$. However, it is only transferred later in transmission opportunity $\text{TRANS}(f,p,j-1,i) + d^j_i$. This can only be due to the outgoing transmission of frame c_{i-1} blocking earlier transmission of c_i on (v^p_j, v^p_{j+1}). Thus, the relevant condition for the minimization in Definition 14 is C_2, so we have:

$$\text{TRANS}(f,p,j,i-1) = \text{TRANS}(f,p,j,i) - 1 \tag{2.79}$$

   Another consequence of C_2 as of Definition 14 is

$$\text{TRANS}(f,p,j-1,i-1) < \text{TRANS}(f,p,j-1,i) \tag{2.80}$$

   Considering that transmission opportunity numbers are integers, we obtain:

$$\text{TRANS}(f,p,j-1,i-1) \leq \text{TRANS}(f,p,j-1,i) - 1 \tag{2.81}$$

   Combining Equations 2.79 and 2.81 completes the induction step:

$$\begin{aligned}
d^j_{i-1} &= \text{TRANS}(f,p,j,i-1) - \text{TRANS}(f,p,j-1,i-1) \\
&\geq \text{TRANS}(f,p,j,i) - 1 - (\text{TRANS}(f,p,j-1,i) - 1) \\
&= d^j_i \\
&\overset{\text{I.H.}}{>} \hat{d}^j
\end{aligned} \tag{2.82}$$

Now consider frame c_1. Due to the above induction, we have $d_1^j > \hat{d}^j$ and obtain:

$$\begin{aligned} \text{TRANS}(f,p,j,1) &= \text{TRANS}(f,p,j-1,1) + d_1^j \\ &> \text{TRANS}(f,p,j-1,1) + \hat{d}^j \end{aligned} \tag{2.83}$$

By definition of $\hat{d}^j$, we have:

$$\text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,1)) < \text{TSL}(f,p,j,\text{TRANS}(f,p,j-1,1) + \hat{d}^j) \tag{2.84}$$

Considering Equation 2.83, this is a contradiction to Definition 14:

$$\begin{aligned} &\text{TRANS}(f,p,j,1) = \\ &\quad \min\{l \in \mathbb{N} \mid \text{TSL}(f,p,j-1,\text{TRANS}(f,p,j-1,1)) < \text{TSL}(f,p,j,l)\} \end{aligned} \tag{2.85}$$

Hence, the assumption was wrong and we find $d^j \leq \hat{d}^j$.

Due to the facts $d^j \in \mathcal{M}$, $d^j \leq \hat{d}^j$ and $\hat{d}^j = \min \mathcal{M}$, we conclude that $d^j = \hat{d}^j = \min \mathcal{M}$. □

2.8 Latency

For real-time communication, it is important that a network delivers data as fast as necessary at its destination, i.e. it introduces only delay that is predictable in a sense of having an upper bound. This delay is quantified by the concept of latency. Latency is the maximum delay experienced by a piece of data being part of a flow when being transferred from its source to its destination. In our model we define latency of a flow to be the maximum delay (in logical time) experienced by the frames that are part of this flow.

Using the TSL and TRANS functions, the expression $\text{TSL}(f,p,j,\text{TRANS}(f,p,j,i))$ gives the consecutive slot number of the micro slot during which $c_i \in \text{THREAD}(f,p)$ is transmitted via edge (v_j^p, v_{j+1}^p). The consecutive micro slot numbers of the slots during which a frame is sent by the source and received by the sink, respectively, can then be compared. Their difference gives the latency value in discrete time. Thus, the definition of the latency notions for frames, threads and flows is straightforward:

Definition 16 (Latency of frames, threads and flows)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario supported by $(\text{SA}, \text{FMAP}, \psi)$. Let $f \in \mathcal{F}$ and $(p, \kappa) = \psi(f)$.

1. *The* latency $\text{FRAMELAT}(f,p,i)$ of a frame $c_i \in \text{THREAD}(f,p)$ *is given by the difference of the consecutive slot numbers of the slots during which c_i is transferred via the last and first edge of p, respectively:*

$$\text{FRAMELAT}(f,p,i) := \text{TSL}(f,p,|p|,\text{TRANS}(f,p,|p|,i)) - \text{TSL}(f,p,1,i) + 1 \tag{2.86}$$

Note that the second term of the difference in Equation 2.86 *has already been simplified due to* $\text{TRANS}(f,p,1,i) = i$. *The constant* 1 *offset accounts for the latency caused by forwarding the frame over the first edge.*

2. *Calculating the maximum of the latency values of all frames in a thread yields the* latency of a thread. *It is given by the function* THREADLAT : $\mathcal{F} \times \mathcal{P}_{\mathcal{G}} \to \mathbb{N}$*:*

$$\text{THREADLAT}(f, p) := \max_{i \in \mathbb{N}} \text{FRAMELAT}(f, p, i) \tag{2.87}$$

3. *The* latency of a flow *is described by the function* FLOWLAT : $\mathcal{F} \to \mathbb{N}$ *and is defined to be the maximum latency of the threads in f's routing:*

$$\text{FLOWLAT}(f) := \max_{(p,\kappa) \in \psi(f)} \text{THREADLAT}(f, p) \tag{2.88}$$

In the definition of THREADLAT, a maximum over all frames that travel via the thread is calculated. This poses a problem since the model does not provide starting and ending times of flows. Having no ending time means there is no bound on the number of frames that are transferred by a flow. Thus, it must be justified that the maximum in Equation 2.87 always exists and a way of computing it without having to consider arbitrary many frames is to be derived. This is achieved by verifying that the latency notion is well defined w.r.t. the transmission equivalence classes introduced by Proposition 1. Thus, to obtain the latency of a thread, only the first $\kappa \cdot |p| - 1$ frames must be considered, since the frames coming after these will use the same micro slots and thus expose the same latency as their respective representative in the range $c_{\kappa \cdot (|p|-1)} + 1, \ldots, c_{\kappa \cdot |p|}$. Even more, the following lemma states that the latency of frames transmitted before reaching stable state will be not greater than the maximum latency experienced by frames that are transmitted once the stable state is reached.

Lemma 4 (Frame latency respects equivalence classes)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario supported by (SA, FMAP, ψ). *Let $f \in \mathcal{F}$ and $(p, \kappa) \in \psi(f)$. Let $\mathcal{M} = \{m \in \mathbb{N} \mid m > \kappa \cdot (|p| - 1)\}$.*

1. *For the frames transmitted after reaching stable forwarding state, the frame latency is the same for all members of an equivalence class:*

$$\forall i, l \in \mathcal{M} : i \sim l \to \text{FRAMELAT}(f, p, i) = \text{FRAMELAT}(f, p, l) \tag{2.89}$$

2. *The latency of all frames transmitted prior to reaching the stable state is bounded by the latency experienced by frames transmitted after reaching the stable state:*

$$\forall i \in \{1, \ldots, \kappa \cdot (|p|-1)\}\ \exists l \in \mathcal{M} : \text{FRAMELAT}(f, p, i) \leq \text{FRAMELAT}(f, p, l) \tag{2.90}$$

PROOF (LEMMA 4)

1. Let $i, l \in \mathcal{M}$ and $i \sim l$. Without loss of generality, assume $i \geq l$. Due to Proposition 1, there exist $\iota, \lambda \in \mathbb{N}$ and $\chi \in \{1, \ldots \kappa - 1\}$, such that:

$$i = \iota \cdot \kappa + \chi \tag{2.91}$$
$$l = \lambda \cdot \kappa + \chi \tag{2.92}$$

Thus, we have $(i-l) = (\iota - \lambda) \cdot \kappa$. Using this relation, we verify the claim of the proposition ($\delta := \kappa \cdot (|p|-1)$):

$$\begin{aligned}
&\textsc{framelat}(f,p,i)\\
&\quad = \quad \textsc{tsl}(f,p,|p|,\textsc{trans}(f,p,|p|,i)) - \textsc{tsl}(f,p,1,i) + 1\\
&\quad \overset{\text{Lemma 3}}{=} \quad \textsc{tsl}(f,p,|p|,\textsc{trans}(f,p,|p|,\delta) + i - \delta)\\
&\qquad\quad - \textsc{tsl}(f,p,1,i) + 1\\
&\quad = \quad \textsc{tsl}(f,p,|p|,\textsc{trans}(f,p,|p|,\delta) + l + (i-l) - \delta)\\
&\qquad\quad - \textsc{tsl}(f,p,1,l+(i-l)) + 1\\
&\quad \overset{\text{Lemma 1}}{=} \quad \textsc{tsl}(f,p,|p|,\textsc{trans}(f,p,|p|,\delta) + l - \delta)\\
&\qquad\quad + (\iota - \lambda) \cdot n_{\mathcal{S}} - (\textsc{tsl}(f,p,1,l) + (\iota - \lambda) \cdot n_{\mathcal{S}}) + 1\\
&\quad \overset{\text{Lemma 3}}{=} \quad \textsc{tsl}(f,p,|p|,\textsc{trans}(f,p,|p|,l))\\
&\qquad\quad - \textsc{tsl}(f,p,1,l) + 1\\
&\quad = \quad \textsc{framelat}(f,p,l)
\end{aligned} \tag{2.93}$$

2. First, we show by induction that the transmission shift between source node and last forwarding node on p for the frames transmitted before reaching stable state is bounded by the transmission shift in effect when the stable state has been reached, i.e.

$$\begin{aligned}
&\forall i \in \{1,\ldots,\kappa\cdot(|p|-1)\}:\\
&\quad \textsc{trans}(f,p,|p|,i) \leq \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) + i - \kappa\cdot(|p|-1)
\end{aligned} \tag{2.94}$$

Thus, let $i = \kappa \cdot (|p|-1)$. The claim is trivially true:

$$\textsc{trans}(f,p,|p|,i) \leq \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) + i - \kappa\cdot(|p|-1) \tag{2.95}$$

Now suppose that Eq. 2.95 is true for some fixed $i \in \{2,\ldots,\kappa\cdot(|p|-1)\}$ and consider $i-1$. We have:

$$\begin{aligned}
&\textsc{trans}(f,p,|p|,i-1)\\
&\quad \overset{\text{Def. 14}}{\leq} \quad \textsc{trans}(f,p,|p|,i) - 1\\
&\quad \overset{\text{I.H.}}{\leq} \quad \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) - \kappa\cdot(|p|-1) + i - 1\\
&\quad = \quad \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) - \kappa\cdot(|p|-1) + (i-1)
\end{aligned} \tag{2.96}$$

Consider a frame c_i during the startup phase, i.e. $i \in \{1,\ldots,\kappa\cdot(|p|-1)\}$. With the help of Eq. 2.94 it is easy to see that $c_{i+\kappa\cdot|p|}$ will be transferred at least $\kappa\cdot|p|$ transmission opportunities after the one that was used for c_i:

$$\begin{aligned}
&\textsc{trans}(f,p,|p|,i)\\
&\quad \overset{\text{Eq. 2.94}}{\leq} \quad \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) - \kappa\cdot(|p|-1) + i\\
&\quad = \quad \textsc{trans}(f,p,|p|,\kappa\cdot(|p|-1)) + i + \kappa\cdot|p| - \kappa\cdot(|p|-1) - \kappa\cdot|p|\\
&\quad \overset{\text{Lemma 3}}{=} \quad \textsc{trans}(f,p,|p|,i+\kappa\cdot|p|) - \kappa\cdot|p|
\end{aligned} \tag{2.97}$$

Using this result and the periodicity of TSL, it is straightforward to complete the proof:

$$
\begin{aligned}
&\text{FRAMELAT}(f,p,i)\\
&\quad= \quad \text{TSL}(f,p,|p|,\text{TRANS}(f,p,|p|,i)) - \text{TSL}(f,p,1,i)+1\\
&\quad\overset{\text{Eq. 2.97}}{\leq} \quad \text{TSL}(f,p,|p|,\text{TRANS}(f,p,|p|,i+\kappa\cdot|p|)-\kappa\cdot|p|)\\
&\qquad\quad - \text{TSL}(f,p,1,i)+1\\
&\quad= \quad \text{TSL}(f,p,|p|,\text{TRANS}(f,p,|p|,i+\kappa\cdot|p|)-\kappa\cdot|p|)\\
&\qquad\quad - \text{TSL}(f,p,1,i+\kappa\cdot|p|-\kappa\cdot|p|)+1\\
&\quad\overset{\text{Lemma 1}}{\leq} \quad \text{TSL}(f,p,|p|,\text{TRANS}(f,p,|p|,i+\kappa\cdot|p|))-n_{\mathcal{S}}\cdot|p|\\
&\qquad\quad - (\text{TSL}(f,p,1,i+\kappa\cdot|p|)-n_{\mathcal{S}}\cdot|p|)+1\\
&\quad= \quad \text{FRAMELAT}(f,p,i+\kappa\cdot|p|)
\end{aligned}
\tag{2.98}
$$

Thus, for each frame transmitted during the startup phase, there is a frame in the stable state that experiences at least as much delay. □

The consequence of Lemma 4 is that for computing the latency of a thread, it is enough to determine the delay values for a representative of each of the κ transmission equivalence classes. The maximum value obtained will be the thread's latency, since the frames being transmitted during the startup phase won't experience higher delay and the frames transmitted during stable state share the delay characteristics with their respective representative.

Note that due to the recursive definition of TRANS, the transmission opportunities that are used by the frames that are part of the startup phase must still be determined in order to compute the transmission opportunities used for the frames during stable state. However, Lemma 3 states that there is a fixed transmission shift once stable state is reached and Proposition 2 provides a way of computing this transmission shift that does not need the exact TRANS values. Recalling that $\text{TRANS}(f,p,1,i)=i$, we can compute the delay experienced by frames being transmitted during stable state without having to compute the TRANS values for the startup phase:

Proposition 3 (Calculation of frame and thread latency)
Let $\mathcal{G}=(\mathcal{V},\mathcal{E})$. Let $\mathcal{F}$ be a scenario supported by $(\text{SA},\text{FMAP},\psi)$ and $(p,\kappa)\in\psi(f)$ be the routing of a flow $f\in\mathcal{F}$. Let $\mathcal{M}=\{i\in\mathbb{N}\mid i>\kappa\cdot(|p|-1)\}$.

1. *For all frames transmitted after reaching stable state, the transmission opportunities they are forwarded during can be calculated from the transmission shifts at the forwarding nodes:*

$$\forall i\in\mathcal{M}\;\forall j\in\{1,\ldots,|p|\}:\text{TRANS}(f,p,j,i)=i+\sum_{l=2}^{j}d^l \tag{2.99}$$

2. *The frame latency values of frames transmitted during stable state depends only on the transmission shifts at the nodes that forward the frames:*

$$\forall i\in\mathcal{M}:\text{FRAMELAT}(f,p,i)=\text{TSL}(f,p,|p|,i+\sum_{l=2}^{|p|}d^l)-\text{TSL}(f,p,1,i)+1 \tag{2.100}$$

3. *Let* $\mathcal{M}' = \{\kappa \cdot (|p| - 1) + 1, \ldots, \kappa \cdot |p|\}$. *For computing the thread latency, it is sufficient to consider the frames* c_i, $i \in \mathcal{M}'$*:*

$$\text{THREADLAT}(f, p) = \max_{i \in \mathcal{M}'} \text{FRAMELAT}(f, p, i) \tag{2.101}$$

PROOF (PROPOSITION 3)

1. The proof is by induction on j. Let $i \in \mathcal{M}$. For $j = 1$, we have:

$$\text{TRANS}(f, p, 1, i) = i = i + \sum_{l=2}^{j} d^l \tag{2.102}$$

Now suppose that the claim holds for some fixed j and consider $j + 1$:

$$\begin{aligned} \text{TRANS}(f, p, j, i) &\overset{\text{Def. 14}}{=} \text{TRANS}(f, p, j-1, i) + d_i^j \\ &\overset{\text{I.H.}}{=} i + \sum_{l=2}^{j-1} d^l + d_i^j \\ &\overset{\text{Lemma 3}}{=} i + \sum_{l=2}^{j-1} d^l + d^j \\ &= i + \sum_{l=2}^{j} d^j \end{aligned} \tag{2.103}$$

2. The claim follows immediately from the definition of FRAMELAT and claim 1.

3. This property is a direct consequence of Lemma 4. It allows to compute the latency of a thread without having to consider each frame that is transmitted. □

The actual calculation of latency values is best demonstrated by an example. Reconsider the situation shown in Figure 2.6. Two flows are defined. Flow f_1's routing specifies two paths: $p_1^1 = (a, b, c, e, g)$ and $p_2^1 = (a, c, e, g)$. The routing of f_2 consists of a single path $p_1^2 = (f, e, c, d)$. The calculation of the flow latency values is as follows:

- Flow f_1:
  - Path p_1^1: The transmission shift values are computed according to Proposition 2 to $d^2 = d^3 = 0$ and $d^4 = 1$. The latency values for representatives of the transmission equivalence classes are calculated using the sum of transmission shifts representation presented in Proposition 3. In this case $\kappa = 1$, so c_4 is the only frame to consider:

$$\begin{aligned} \text{FRAMELAT}(f_1, p_1^1, 4) &= \text{TSL}(f_1, p_1^1, 4, 5) - \text{TSL}(f_1, p_1^1, 1, 4) + 1 \\ &= 43 - 31 + 1 = 13 \end{aligned} \tag{2.104}$$

  Since there is only one equivalence class, we obtain $\text{THREADLAT}(f_1, p_1^1) = 13$ for the thread latency.

– Path p_2^1: Here, we have $d^2 = 0$ and $d^3 = 1$. The frame latency of the representative of the only transmission equivalence class is:

$$\begin{aligned} \text{FRAMELAT}(f_1, p_2^1, 3) &= \text{TSL}(f_1, p_2^1, 3, 4) - \text{TSL}(f_1, p_2^1, 1, 3) + 1 \\ &= 35 - 22 + 1 = 14 \end{aligned} \tag{2.105}$$

Thus, the thread latency is $\text{THREADLAT}(f_1, p_2^1) = 14$.

According to Definition 16, the flow latency is the maximum of the thread latency computed above, i.e. $\text{FLOWLAT}(f_1) = \max\{\text{THREADLAT}(f_1, p_1^1), \text{THREADLAT}(f_1, p_2^1)\} = \max\{13, 14\} = 14$.

- Flow f_2:

  In this case, we have only one path in the routing of f_2, but the path p_1^2 provides a capacity of $\kappa = 2$. The transmission shifts are $d^2 = 0$ and $d^3 = 2$, respectively. Since $\kappa = 2$ there are 2 transmission equivalence classes and 2 representatives must be considered:

$$\begin{aligned} \text{FRAMELAT}(f_2, p_1^2, 5) &= \text{TSL}(f_2, p_1^2, 3, 7) - \text{TSL}(f_2, p_1^2, 1, 5) + 1 \\ &= 33 - 21 + 1 = 13 \end{aligned} \tag{2.106}$$

$$\begin{aligned} \text{FRAMELAT}(f_2, p_1^2, 6) &= \text{TSL}(f_2, p_1^2, 3, 8) - \text{TSL}(f_2, p_1^2, 1, 6) + 1 \\ &= 35 - 22 + 1 = 14 \end{aligned} \tag{2.107}$$

The thread latency is defined as the maximum frame latency values, thus we find

$$\begin{aligned} \text{THREADLAT}(f_2, p_1^2) &= \max\{\text{FRAMELAT}(f_2, p_1^2, 5), \text{FRAMELAT}(f_2, p_1^2, 6)\} \\ &= \max\{13, 14\} \\ &= 14 \end{aligned} \tag{2.108}$$

For the flow latency we also have $\text{FLOWLAT}(f_2) = 14$ because p_1^2 is the only path in f_2's routing.

2.9 Quality-of-Service requirements and solutions

So far, the only information known about a flow are the source and sink nodes. This suffices to discuss possible paths through the network that handle the flow. However, applications that transfer data on the flows typically have Quality-of-Service (QoS) requirements such as a minimum data rate the network must be able to provide to the flow or a limit on the latency that is acceptable to the application. For instance, a voice communication application will probably require a constant data rate of a few kilobits per second as well as a maximum latency in the order of a second. These restrictions are captured by the notion of QoS requirements that are attached to a flow.

Definition 17 (QoS requirements)
Let $\mathcal{F}$ be a scenario. Each flow is associated with a number of Quality-of-Service requirements:

Capacity *The capacity requirement of flow is denoted by the function* $\text{FCAP} : \mathcal{F} \rightarrow \mathbb{R}^+$*, which gives the data rate a flow requires in bytes per second.*

Latency *A flow is assigned a latency requirement by the function* $\text{FLAT} : \mathcal{F} \rightarrow \mathbb{N} \cup \{\infty\}$*, which indicates the maximum latency tolerable for a flow in discrete time, i.e. the maximum number of consecutive micro slots after which a frame must have arrived at the sink.*

Of course, the QoS requirements should be met by a combination of slot assignment, flow mapping and flow routing $(\text{SA}, \text{FMAP}, \psi)$. This is captured by the following definition, which defines the notion of a solution to a given scenario:

Definition 18 (Solution to a scenario)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *be the connectivity graph and* $\mathcal{F}$ *a scenario supported by* $(\text{SA}, \text{FMAP}, \psi)$*. The triple* $(\text{SA}, \text{FMAP}, \psi)$ *is called a* solution to $\mathcal{F}$ *if it meets the QoS requirements of the flows in* $\mathcal{F}$*:*

Capacity *For each flow, the capacity provided by the flow routing must be large enough to handle the flow's capacity requirement (converted to frames per macro slot representation):*

$$\forall f \in \mathcal{F} : \sum_{(p,\kappa)\in\psi(f)} \kappa \cdot \frac{n_F}{d_{ms}} \geq \text{FCAP}(f) \tag{2.109}$$

Latency *The latency of a flow as derived in Section 2.8 should be not larger than the latency requirement stated for the flow:*

$$\forall f \in \mathcal{F} : (\text{FLAT}(f) \neq \infty) \rightarrow (\text{FLOWLAT}(f) \leq \text{FLAT}(f)) \tag{2.110}$$

Note that the QoS requirements defined above are just two possible choices that fit well with the model. The model can be extended easily be defining new requirements and the conditions that a solution must fulfill to meet the requirement.

2.10 Metrics

Normally, wireless networks should be designed such that they provide enough capacity to carry the flows that are to be transferred through the network. Thus, scenarios are expected to always be solvable. However, solutions are not unique in general. Therefore, it is reasonable to define performance metrics that allow to evaluate and compare different solutions. Properties that are usually considered when defining metrics include latency, capacity usage and energy efficiency, which is particularly important in the context of wireless sensor networks.

Note that the metrics defined in the following paragraphs only represent some examples of metrics that can be considered in the model. It is of course possible to define further metrics or even use combined metrics that grade a solution w.r.t. different criteria. A possible way to do so is to consider weighted sums of the metric values obtained from individual metrics.

2.10.1 Capacity usage

Capacity usage is the sum of capacity allocated by a solution. Depending on the paths on which data flows through the network, the number of nodes that need to forward frames varies. At each edge, capacity is required to forward frames to the next hop. Thus, the capacity usage notion gives an overall indication on network load. Intuitively, it is favorable to use less capacity since that suggests the unused capacity could be used for other transmissions. It is expected that shorter paths result in lower capacity usage and should thus be preferred. Furthermore, shorter paths require less nodes to transmit a frame, so the total energy consumed by a data flow is expected to be lower.

Definition 19 (Capacity usage metric)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and $(\text{SA}, \text{FMAP}, \psi)$ a solution to $\mathcal{F}$. The total capacity allocated by FMAP *is called its* capacity usage *and can be calculated as follows:*

$$\sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_\mathcal{G}} \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} \text{FMAP}(f, p, e, s) \tag{2.111}$$

2.10.2 Latency

Using the latency notion defined in Section 2.8, we can define latency metrics. One obvious possibility is to define a metric that favors solutions minimizing the maximum latency frames encounter:

Definition 20 (Maximum latency metric)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and $(\text{SA}, \text{FMAP}, \psi)$ a solution to $\mathcal{F}$. The maximum latency metric *considers the maximum latency experienced by any frame in the network, i.e. the maximum latency of all flows:*

$$\max_{f \in \mathcal{F}} \text{FLOWLAT}(f) \tag{2.112}$$

Note that solutions that perform well in this metric will have to give priority to flows for which the hop-distance between source and sink is large, because their latency will likely dominate the latency values of flows that connect nearby nodes. The following metric that considers the latency sum of all flows alleviates this problem:

Definition 21 (Latency sum metric)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and $(\text{SA}, \text{FMAP}, \psi)$ a solution to $\mathcal{F}$. The sum of the flow latency values over all flows defines the latency sum metric*:*

$$\sum_{f \in \mathcal{F}} \text{FLOWLAT}(f) \tag{2.113}$$

2.10.3 Energy consumption

While energy efficiency of computing hardware has recently received attention in a broad range of contexts, it has always been particularly important for WSNs. Wireless sensors are only equipped with very limited energy resources, thus a WSN is only useful as long as there are enough sensor nodes alive to perform the networks' task.

Different energy consumption metrics were put forward over the years. Some examples (taken from [42]) include: Energy consumption by frame, which is based on the idea to minimize the sum of energy consumed when transferring a single frame from source to sink. Furthermore, it can be beneficial to keep the network in a connected state, which is achieved by using the time until the network is partitioned as a metric.

Note that some metrics can be difficult to implement in actual networks without further assumptions. This happens when the situation can arise that a metric-relevant decision must be made that is influenced by events which will happen in the future. For example, the optimal decision about which paths to use for a flow in order to maximize the time to network partition depends on what flows need to be handled during the remaining lifetime of the network.

For the definition of energy consumption metrics a model is needed that allows to calculate how much energy is used by the operations that can take place in the network. The energy consumption of a single node typically depends on what it is currently doing [50, 11]. The wireless communication hardware of a node can be in different states:

Sleeping: The node's transceiver hardware is powered down, no messages can be transmitted or received. This is the state of smallest energy consumption.

Idle: The wireless network device is powered up and monitoring the channel for incoming transmissions, but nothing is transmitted or received at the moment.

Receiving: A message is received, decoded by the hardware and made available to the sensor node for further processing of the message.

Discarding: The wireless medium is carrying a message, but the node decided to discard this message instead of receiving it. Depending on the hardware design, the wireless network device can switch to a low power state similar to the sleeping state for the duration of the message.

Transmitting: A message is currently being transmitted by the node.

In [11], measurements of the actual energy consumption of an 802.11 network device are described. According to the results, the amount of energy consumed when transmitting and receiving even depends on whether a point-to-point or a broadcast message is being transmitted or received. For our model, this is irrelevant since we only use point-to-point communication. The measurements also show that actual energy consumed when receiving or transmitting can be modeled with sufficient accuracy by a linear expression [11, 10] that comprises a constant term and a proportional cost that depends on the number of bytes carried by the message. The constant term results from operations that are needed for all messages, e.g. transmitting or synchronizing to a preamble that is transmitted at the very beginning of every message. Based on these results, we introduce an energy consumption model that is linear, but uses the amount of time the hardware spends in a certain state as the linear parameter instead of the number of bytes of a message.

Definition 22 (Energy consumption model)
Let $\mathcal{Z} = \{\mathrm{S}, \mathrm{I}, \mathrm{R}, \mathrm{D}, \mathrm{T}\}$ be the set of the possible hardware states as mentioned above, i.e. sleeping, idle, receiving, discarding and transmitting. The energy consumed by a node depends on its current operating state $z \in \mathcal{Z}$.

1. *If the node is sleeping or idle ($z \in \{\mathrm{S}, \mathrm{I}\}$), the energy consumption is described by the function $E_z : \mathcal{V} \times \mathbb{R}^+ \to \mathbb{R}^+$ which maps the duration of time t in which node a is in a particular state z to the corresponding energy cost:*

$$E_z(a,t) = m_z^a \cdot t \tag{2.114}$$

2. *For the states receiving, discarding and transmitting, the energy consumption also depends on the communication partner b and the slot during which the communication takes place. The function $E_z : \mathcal{V} \times \mathbb{R}^+ \times \mathcal{V} \times \mathcal{S} \to \mathcal{R}^+$ gives the energy consumed by node a that is in state $z \in \{\mathrm{R}, \mathrm{D}, \mathrm{T}\}$ and communicates with node b during slot s:*

$$E_z(a,t,b,s) = c_{z,b,s}^a + m_{z,b,s}^a \cdot t \tag{2.115}$$

The coefficients m_z^a, $c_{z,b,s}^a$ and $m_{z,b,s}^a$ are system parameters. $c_{z,b,s}^a$ represents a fixed energy cost that is caused by bringing the hardware into the desired state, m_z^a and $m_{z,b,s}^a$ account for the energy that is consumed per time unit when node a is in state z. Note that for the idle and sleeping states, there is no constant cost.

Now consider the energy spent by a node during a single macro slot. Each micro slot must be considered. Depending on the slot assignment and flow mapping functions, a node can be doing nothing, receive or discard messages from other nodes or transmit own messages in the time interval corresponding to a micro slot. For calculating the energy cost, we will assume the system is in stable forwarding state. Due to Proposition 1, all transmission opportunities a node has will be used, thus we can base the energy calculation on the capacity allocated by the flow mapping. Of course, the results will not be valid for the startup phase. However, the energy spent by a node during a macro slot of the startup phase is expected to be lower than the energy consumption per macro slot in the stable forwarding state, since less transmission opportunities are actually carrying frames in the startup phase. Furthermore, the number of macro slots that are part of the startup phase is bounded by the maximum number of hops in the flows' paths. Thus, for flows that are not too short lived we expect the relevant share of energy to be spent once stable forwarding state has been reached such that the error term due to inaccurate treatment of the startup phase can safely be ignored.

Definition 23 (Energy consumption of a node)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario, $(\mathrm{SA}, \mathrm{FMAP}, \psi)$ a solution to the scenario and $a \in \mathcal{V}$ a node.

- *Let $s \in \mathcal{S}$ be a micro slot. The energy a spends during s can be computed as follows:*

$$\mathrm{NRG}_a(s) = \begin{cases} E_\mathrm{S}(a, d_s) & \nexists b \in \mathcal{V} : (a,b) \in \mathrm{SA}(s) \vee (b,a) \in \mathrm{SA}(s) \\ e_1 & \exists b \in \mathcal{V} : (b,a) \in \mathrm{SA}(s) \\ e_2 & \exists b \in \mathcal{V} : (a,b) \in \mathrm{SA}(s) \end{cases} \tag{2.116}$$

  The terms e_1 and e_2 describe the energy that is spent when a is potentially receiving or transmitting. Let $n^e = \mathrm{FMAP}(f,p,e,s)$, i.e. the capacity (in number of frames) allocated on edge e during slot s. Let $\mathcal{E}' = \{(v,w) \in \mathrm{SA}(s) \mid (v,a) \in \mathrm{SA}(s) \wedge w \neq a\}$.

When a is receiving, it is idle unless it actually receives or discards a frame from another node:

$$e_1 = E_{\mathrm{I}}(a, d_s) + \sum_{b\in\mathcal{V}}\sum_{f\in\mathcal{F}}\sum_{p\in\mathcal{P}_\mathcal{G}}\sum_{i=1}^{n^{(b,a)}} E_{\mathrm{R}}(a, \mathcal{T}((b,a),s), b, s) - E_{\mathrm{I}}(a, \mathcal{T}((b,a),s)) + \sum_{b\in\mathcal{V}}\sum_{f\in\mathcal{F}}\sum_{p\in\mathcal{P}_\mathcal{G}}\sum_{e\in\mathcal{E}'}\sum_{i=1}^{n^e} E_{\mathrm{D}}(a, \mathcal{T}(e,s), b, s) - E_{\mathrm{I}}(a, \mathcal{T}(e,s)) \tag{2.117}$$

When a is supposed to transmit during slot s it does so until all transmissions scheduled by the flow mapping are completed, after which the node goes to sleep:

$$e_2 = E_{\mathrm{S}}(a, d_s) + \sum_{b\in\mathcal{V}}\sum_{f\in\mathcal{F}}\sum_{p\in\mathcal{P}_\mathcal{G}}\sum_{i=1}^{n^{(a,b)}} E_{\mathrm{T}}(a, \mathcal{T}((a,b),s)) - E_{\mathrm{S}}(a, \mathcal{T}((a,b),s)) \tag{2.118}$$

- *The total energy spent by a node in the course of a macro slot is described by the function* $\mathrm{NRG} : \mathcal{V} \to \mathbb{R}^+$ *and is calculated by summing up the energy consumption during each micro slot:*

$$\mathrm{NRG}(a) = \sum_{s\in\mathcal{S}} \mathrm{NRG}_a(s) \tag{2.119}$$

The energy consumption per node a and slot s as defined above accurately models the amount of energy spent when the node is sleeping or transmitting during s. However, there is a slight inaccuracy for the case when SA determines that a is supposed to receive. e_1 correctly accounts for the energy consumption during the time intervals while a is actually handling messages (i.e. receiving or discarding) on the edges in its immediate neighborhood. The first sum expression considers all messages that a can hear which are actually destined for it. Other transmissions by nodes that are allowed to transmit messages to a are accounted for in the second sum term of e_1. It is assumed a picks up these messages, which is a simplification because that depends on the transmission energy used by the sending node. Furthermore, the nodes allowed to transmit to a may decide to keep quiet during subintervals of s or even for the entire time interval associated with s. If so, a might be able to pick up transmissions from nodes farther away that it will discard. For simplicity, we ignore this situation in e_1.

There is another aspect about the energy consumption that is worth pointing out here. As explained in Section 2.5, the model allows time intervals during the macro slots that are not under control of the TDMA mechanism. During these time intervals, the nodes will obviously also consume energy but this is not reflected in the definition above. However, the purpose here is to devise a metric that can be used to evaluate different solutions to a given combination of network and scenario. The solutions only influence the energy consumption during the time intervals that are controlled by the TDMA mechanism. Thus it is quite safe to assume that the energy spent in the remaining time intervals is the same for all possible solutions, so it is not necessary to consider it in the metrics.

Building upon the energy consumption per node and macro slot, it is relatively straightforward to define energy consumption metrics. The first obvious metric is to just sum up the absolute energy consumption imposed by a solution:

Definition 24 (Absolute energy consumption metric)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and $(\text{SA}, \text{FMAP}, \psi)$ a solution to $\mathcal{F}$. The absolute energy consumption *of the solution is given by:*

$$\sum_{v \in \mathcal{V}} \text{NRG}(v) \tag{2.120}$$

Obviously, the absolute energy consumption metric has the drawback that energy is treated as a system resource instead of considering the energy resources at individual nodes. Thus, a solution that is good according to the absolute energy consumption metric may overutilize individual nodes such that they fail early while the other nodes still have considerable remaining energy resources. To deal with this, it is reasonable to also consider the energy resources local to a node and to define a metric that favors solutions which maximize the time until the first node fails:

Definition 25 (Node-local energy consumption metric)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario and $(\text{SA}, \text{FMAP}, \psi)$ a solution to $\mathcal{F}$. Let NRGR_v be the initial energy resources at node v. The node-local energy consumption metric *uses the time at which the first node is expected to fail as metric criteria:*

$$\min_{v \in \mathcal{V}} \frac{\text{NRGR}_v}{\text{NRG}(v)} \tag{2.121}$$

Thus, only solutions that ensure none of the nodes fail early will be considered good by the node-local energy consumption metric. However, there are situations in which this metric does not yield the desired result. For example, if there are one or more nodes that do not have much initial energy resources available, these will probably fail early even if they are not utilized by a solution. Thus, the metric value is only determined by the early-failing node and the node-local energy consumption metric will not provide any indication about relative performance of the good solutions.

3 Analysis and evaluation

This chapter discusses scenarios and their solutions as defined in Chapter 2. The focus of this chapter is on the question whether using different paths to handle the threads of a flow is beneficial over using only a single path and thread. To answer this question, optimal solutions are considered, both for the original scenario that allows multiple paths per flow and also for the scenario modified by adding the condition of using only a single path per flow. After motivating the use of multi-path routing and discussing potential benefits over the single path approach, a selection of example scenarios is presented from which some of the characteristics of the two approaches of solving the scenarios can be derived. We proceed to investigate the relative performance of the single-path and multi-path approaches by examining the possible multi-path performance gain in a generic setting. Then, algorithmic approaches for finding an optimal solution given an arbitrary scenario are discussed. Finally, the results obtained by applying a problem solving program to a large number of randomly generated scenarios are used to compare the performance of the multi-path and single-path approaches, respectively.

3.1 Single-Path versus Multi-Path Routing

Routing is the problem of finding paths through the network that can be used to transfer the frames of a data flow from its source to the sink node. While single-path routing means that all data belonging to the flow will travel on the same path, multi-path routing allows a flow to be handled by different paths. There are various motivations for using multi-path routing. The following examines some of them.

Load balancing The use of multiple paths for a flow instead of a single one is expected to distribute the load caused by the flow more uniformly over the network. The set of nodes that handle the flow is increased, thereby reducing the average load experienced by individual nodes. This is desirable for a number of reasons. Less load means less energy consumption, which can prolong the network lifetime in the context of WSNs. Furthermore, the more uniform distribution of load should help to alleviate hot spot situations in which certain nodes or regions in the network are fully utilized thereby preventing any additional flows to be handled even if the remaining network still has unused capacity available.

Reliability Multi-path routing can increase reliability. The problem addressed is the unreliability that is inherent to wireless networks: Link quality and stability is subject to environmental factors such as foreign interference, weather conditions, remaining energy resources etc. Similarly, nodes can fail or move. Thus, links in wireless networks can break and disrupt flows if the broken link is part of the path that was used to handle the flow. Numerous approaches have been proposed in order to increase reliability (e.g. [28, 36]. Multiple node- and link-disjoint paths allow to transmit data even if a single

node or link fails, as long as at least one path is still available. Reliability is achieved either by redundancy, i.e. sending a copy of each data frame on each available path, or by switching to a different path if one path fails. Information about alternative paths can either be kept at the source node or at the forwarding nodes. With the former approach, path breakage must be reported to the source node which can then switch to a different path. The latter allows a forwarding node to switch to a different path segment to the sink node when the node detects its forwarding link has failed. Another possibility to use multiple paths for redundancy is to employ a diversity coding scheme that encodes the data to be transferred in a way such that if a given fraction of the data frames arrive at the destination, the original data can be recovered [46].

Note that many of the reliability enhancing techniques discussed above do not fit well with the reservation based scheme that is used with TDMA networks. In our network model, a path is only available when the resources required to handle that path have been allocated. Thus, making reservations for paths that are only used when the primary path fails is not a good idea, because a lot of the reserved capacity is actually not used. Hence, if multi-path techniques are used to enhance reliability in a reservation based network, additional reserved capacity should be used to transfer redundant information that helps to recover data in case a path breaks and only a fraction of the data frames arrive at the sink.

Throughput In a reservation-based network employing single-path routing the situation may arise that a new flow f can not be handled because there is not enough capacity left to handle the flow, i.e. no path is available that could provide the requested throughput. This can be due to f's capacity requirement being too large to be handled by a single path (e.g. when there are bottleneck regions in the network) or due to other flows that are already present taking up too much resources such that no single path to provide the requested capacity is available. In contrast, a multi-path routing scheme may split up a flow's large capacity requirement into several smaller ones for which suitable paths can possibly be found such that the aggregate capacity allows to handle the flow.

Latency Increasing the data rate on a single path will likely increase latency. This is because frames in transit might queue at intermediate nodes until they can be forwarded. In our formal model, this effect is due to the following observation: If there are only few unused micro slots available at a forwarding node, an incoming frame will have to wait longer on average for its outgoing slot than in a situation where the node has many unused micro slots available. Hence, heavily used paths are likely to be inferior in terms of latency. A single-path solution is expected to put heavier load on its single path than each of the paths in a multi-path solutions are required to handle. Thus, the multi-path solution is likely to yield better latency.

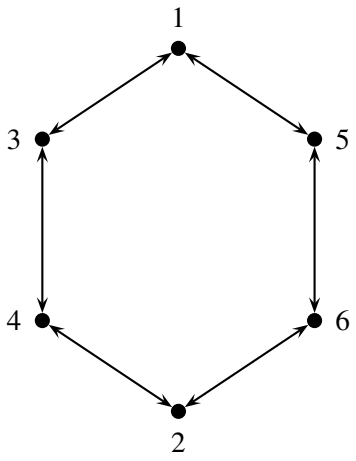
This discussion promises that multi-path approaches are indeed beneficial over those that only use single paths. However, discovering and managing multiple paths can also add additional cost, which must be considered when evaluating the benefits of multi-path routing. Additional overhead is caused e.g. by an increase in the number of control messages that a route discovery protocol may need to exchange to establish multiple routes.

The critical aspect that determines whether multi-path routing techniques perform well or not in wireless networks is the system-inherent problem of interference. Results of work

that studies multi-path routing in wired networks [4, 6] cannot be easily adapted to wireless networks because the interference problem between packets traveling on node-disjoint paths does not occur in wired networks. In contrast, node-disjoint paths in wireless networks can possibly interact due to mutual interference.

3.2 Selected scenarios

This section identifies some scenarios that demonstrate situations for which the multi-path approach yields better results than the single-path approach. Consider the network depicted in Figure 3.1a. The figure shows a realistic network, i.e. the situation as depicted can easily be reconstructed in reality by placing six identical nodes at the vertices of a hexagon such that the nodes on adjacent vertices are on the verge of each other's transmission distance.



(a)

s	SA(s)
1	$\{(1,3),(6,2)\}$
2	$\{(3,4),(1,5)\}$
3	$\{(4,2),(5,6)\}$

(b)

Figure 3.1: A hexagon network in the plane. (a) shows the positions of the nodes in the plane and also illustrates the edges of the connectivity graph. (b) gives a possible consistent slot allocation.

The scenario to be handled by the network is a single data flow that starts at node 1 and has node 2 as destination. Suppose there are three micro slots per macro slot and that each node can transmit exactly one frame during a single micro slot. This number is obviously chosen very small. Note that this is only for simplicity; the following considerations are also valid for a scaled version of the scenario that uses appropriately scaled numbers of micro slots and capacity requirements for the flows. The question that is to be discussed is how big the capacity requirements may be chosen such that there is a solution to the scenario and whether there is a difference between the single-path and the multi-path approach.

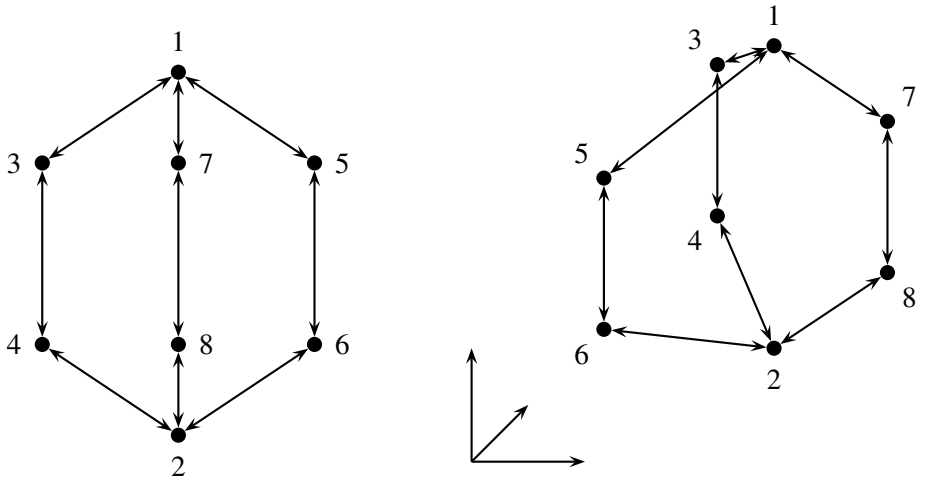
For this example, the graph-based noninterference model is used to determine transmission conflicts. There are exactly two cycle-free paths that connect node 1 to node 2, namely $p_1 = (1,3,4,2)$ and $p_2 = (1,5,6,2)$. Both paths have three edges for which one can easily check that only one edge in a path can be scheduled in a micro slot as otherwise the transmissions would interfere with each other. Thus, the slot allocation must allocate a different slot to each of the edges in the path. However, for each edge, the opposite in the other path can be used concurrently without causing interference. A possible consistent slot allocation is shown in

Figure 3.1b.

Mapping the allocated capacity to flows is straightforward. In the single-path case, we have to pick one path and can transmit exactly one frame per macro slot. Thus, the maximum capacity requirement is that of the frame size divided by the macro slot duration. Using multi-path routing, we can do better by also using the second path that is available, thus capacity requirements up to twice the frame size per macro slot duration are feasible.

By running the scenario solving tool that is described in Section 3.7 it can be verified that the solutions just discussed are the optimal solutions for the single-path and multi-path approaches, respectively. Moreover, the tool proved that the capacity requirement bounds given are correct, i.e. increasing the capacity requirements stated before and trying to find a solution yields the result that no such solution exists.

By means of the example, we have established that there are scenarios for which multi-path routing can yield twice the performance that is possible with single-path routing in terms of throughput. The performance gain can be improved by adding a third path. This is considered in the example given in Figure 3.2.



s	SA(s)
1	{ (1, 3), (5, 6), (8, 2) }
2	{ (3, 4), (1, 7), (6, 2) }
3	{ (4, 2), (7, 8), (1, 5) }

(c)

Figure 3.2: Another example that exhibits differences between the single-path and multi-path routing approaches. (a) shows the connectivity graph, the node locations in 3D space are indicated in (b). (c) gives a possible consistent slot allocation under the graph-based noninterference model.

Note that the connectivity graph given in Figure 3.2a is one that cannot be easily reconstructed by a physical setup in the plane unless special arrangements are made such as directional antennas or objects that block reception. However, this connectivity graph can be quite easily obtained in a physical network by placing the nodes in 3D space. A possible setup

places the intermediate nodes on the corners of a triangle each and the source and sink nodes slightly above or below the respective triangle. An illustration of this setup is given in Figure 3.2b.

Again, the scenario consists of a single flow starting at node 1 and flowing to node 2. A suitable consistent slot allocation function is given in Figure 3.2c. The results for the single-path routing approach are the same as for the previous example, i.e. the maximum capacity requirement that can be met is one frame size per macro slot duration. In the case of multi-path routing, there is now a third path available that increases the maximum feasible capacity requirement to three times the frame size per macro slot duration. Again, these findings have been verified using the problem solving tool described in Section 3.7.

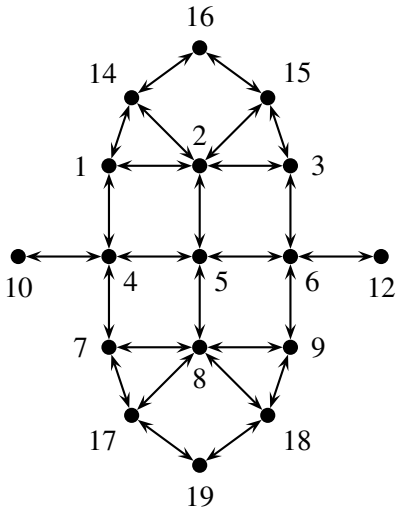
At this point, the node setup could still be changed by adding a further paths that do not introduce interference with the existing 3 paths at the forwarding nodes. However, this will not increase the maximum feasible capacity requirement of the flow, because the source and sink nodes do not have any more slots available during which they could push data through the additional path [1].

Figure 3.3 shows another scenario that benefits from the multi-path approach. The scenario has two flows, one flowing from node 16 to node 19 and a second one that crosses the first from node 10 to node 12. The setup provides for 10 micro slots per macro slot, each of which allows a single frame of 100 bytes to be transferred by a node. The macro slot duration is 1 second and the micro slot duration 0.1 seconds. The capacity requirements are 5 frames per macro slot for the first flow and 1 frame per macro slot for the second flow.

The slot allocation and flow mapping of an optimal multi-path solution w.r.t. the maximum latency metric are given in the Figures 3.3b and 3.3c, respectively. The latency value achieved by the solution is 7. Note that the optimal latency is 6 if the cross flow is removed from the scenario, which is expected because the minimum hop distance between nodes 16 and 19 is 6. It is interesting to note that the crossing flow in this case does not reduce the maximum feasible capacity requirement of the first flow, but degrades the latency that is achievable.

Note that there is no single-path solution for the example scenario of Figure 3.3. This is due to the following observation: Consider a path of minimum length 3 that is part of a flow routing as illustrated in Figure 3.4. Let a, b, c and d be four adjacent nodes on the path. Each frame that is transferred via this path must be processed by every node on the path. Consider a single frame at node b. It receives the frame from node a during some micro slot k. During this micro slot, node b must not transmit any messages, otherwise it would miss the incoming frame from a. Thus, node b needs to forward the frame to node c during some different micro slot l. Eventually c forwards the frame to d. The micro slots k and l cannot be used for that, because c is listening in micro slot l and c may not send in micro slot k, since it would interfere with a's transmission to b (assuming a symmetric connectivity relation and the graph-based noninterference model). Thus, a micro slot m different from k and l must be used for the outgoing transmission at c. Recalling that only one frame per micro slot can be transferred by each node, this means that only up to a third of the available micro slots can actually carry outgoing frames at each node on the path. This gives the following bound on

[1] However, if the number of available micro slots is increased to e.g. 4, another independent path will help to increase the feasible capacity requirement.



(a)

s	$\mathrm{SA}(s)$
1	$\{(16,15),(6,9),(1,4)\}$
2	$\{(15,2),(9,18),(4,7),(6,12)\}$
3	$\{(2,5),(18,19),(7,17)\}$
4	$\{(5,8),(16,15)\}$
5	$\{(8,18),(15,3),(16,14),(17,19)\}$
6	$\{(18,19),(3,6),(14,1),(10,4)\}$
7	$\{(6,9),(1,4),(16,15)\}$
8	$\{(9,18),(4,7),(15,3)\}$
9	$\{(18,19),(7,17),(3,6),(16,14),(4,5)\}$
10	$\{(17,19),(14,1),(5,6)\}$

(b)

	Mapping of flows to slots									
Edge	1	2	3	4	5	6	7	8	9	10
$(1,4)$	1.5						1.3			
$(2,5)$			1.1							
$(3,6)$						1.2			1.4	
$(4,5)$									2.1	
$(4,7)$		1.5						1.3		
$(5,6)$										2.1
$(5,8)$				1.1						
$(6,9)$	1.4						1.2			
$(6,12)$		2.1								
$(7,17)$			1.5						1.3	
$(8,18)$					1.1					
$(9,18)$		1.4						1.2		
$(10,4)$						2.1				
$(14,1)$						1.3				1.5
$(15,2)$		1.1								
$(15,3)$					1.2			1.4		
$(16,14)$					1.3				1.5	
$(16,15)$	1.1			1.2			1.4			
$(17,19)$					1.5					1.3
$(18,19)$			1.4			1.1			1.2	

(c)

Figure 3.3: A scenario with two crossing flows, one flowing from node 16 to node 19 and the other from node 10 to node 12. (a) shows the node locations and connectivity. A consistent slot allocation function for the graph under the graph-based noninterference model is given in (b). The flow mapping shown in (c) in conjunction with the slot allocation in (b) gives a solution to the scenario.

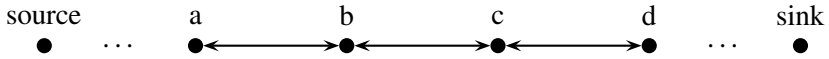


Figure 3.4: Four nodes, a, b, c and d, being part of a path.

the capacity c in bytes per second that a path p can provide:

$$c \leq \left\lfloor \frac{n_{\mathcal{S}}}{3} \right\rfloor \cdot \frac{n_{\mathrm{F}}}{d_{ms}} \tag{3.1}$$

The first term is due to the observation discussed above; only up to a third of the micro slots that are available can actually be used by a node to transfer data. For the rest of the time it has to keep quiet such that adjacent nodes can transmit successfully. The second term converts the bound representation from frames per macro slot to bytes per second.

For the single-path approach, the bound given in Equation 3.1 is also a bound on the maximum satisfiable capacity bound per flow, because there is only one path per flow. In contrast, multi-path routing does not have a corresponding capacity bound per flow, because the capacity of multiple paths can be combined up to the situation in which the source node uses all the outgoing capacity for transmissions as we have seen in the example of Figure 3.2. Applying the single-path bound to the first flow of the example given in Figure 3.3 yields a capacity bound of 300 bytes per second, which is not enough to satisfy the bandwidth requirement of 5 frames per macro slot, i.e. 500 bytes per second. Thus, there is no single-path solution to this scenario.

3.3 Relative performance of single-path and multi-path solutions

This section continues the comparison of single-path and multi-path solutions in a more systematic way. Obviously, a single-path solution also represents a correct multi-path solution. Thus, the objective discussed is the potential performance gain that can be achieved by employing the multi-path approach. In this context, the notion of performance refers to the optimal metric values achievable by any solution under the single-path and multi-path approaches, respectively. Additionally, the maximum feasible capacity requirement for which there is a solution is considered as a performance indicator. As will be shown, the performance gain that is achievable by multi-path routing over single-path routing is not bounded, i.e. there exist networks that exhibit arbitrarily large performance gains. The discussion in this section is restricted to the graph-based noninterference model. However, the networks constructed below are designed with actual node locations in mind, such that they are expected to yield similar results under the other noninterference models.

In the discussion of the example network described in Figure 3.2, an upper bound on the maximum feasible capacity requirement for single-path solutions has already been described. The multi-path performance gain can be captured by the quotient $\frac{\beta_m}{\beta_s}$ of maximum feasible capacity requirements achievable by optimal solutions in the single-path (β_s) and multi-path

(β_m) approaches. In the following, a family of networks that allows arbitrarily large performance gains is presented.

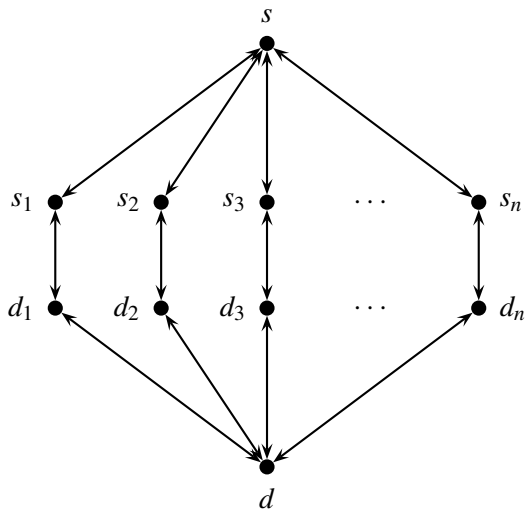


Figure 3.5: Construction of a network that exhibits a performance gain of n w.r.t. maximum feasible capacity requirement when comparing the multi-path approach to the single-path approach.

The construction of the network (also illustrated in Figure 3.5) is as follows: Let $n \in \mathbb{N}$ such that $n \geq 5$. The connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given by:

$$\mathcal{V} := \{s, d\} \cup \{s_i \mid i \in \{1, \ldots, n\}\} \cup \{d_i \mid i \in \{1, \ldots, n\}\} \tag{3.2}$$

$$\mathcal{E}' := \{(s, s_i) \mid i \in \{1, \ldots, n\}\} \cup \{(s_i, d_i) \mid i \in \{1, \ldots, n\}\} \cup \{(d_i, d) \mid i \in \{1, \ldots, n\}\} \tag{3.3}$$

$$\mathcal{E} := \{(v, w) \in \mathcal{V}^2 \mid (v, w) \in \mathcal{E}' \vee (w, v) \in \mathcal{E}'\} \tag{3.4}$$

For the TDMA model, we define the number of micro slots as $n_{\mathcal{S}} := n$ and the frame size n_{F} and timing parameters d_s, d_{ms} in a way that allows each node to transmit exactly one frame per micro slot. The scenario to be solved is given by $n+1$ flows $f_1, \ldots, f_n, f'$ with the following parameters:

$$\forall i \in \{1, \ldots, n\} : \textsc{fsource}(f_i) := s_i \tag{3.5}$$

$$\forall i \in \{1, \ldots, n\} : \textsc{fsink}(f_i) := d_i \tag{3.6}$$

$$\forall i \in \{1, \ldots, n\} : \textsc{fcap}(f_i) := (n-3) \cdot \frac{n_{\mathrm{F}}}{d_{ms}} \tag{3.7}$$

$$\textsc{fsource}(f') := s \tag{3.8}$$

$$\textsc{fsink}(f') := d \tag{3.9}$$

$$\tag{3.10}$$

Now consider the maximum achievable capacity requirement $\textsc{fcap}(f')$ for flow f'. In the single-path approach, the flows f_i must be handled by paths formed by the direct edge (s_i, d_i). Any other paths cannot provide the necessary capacity due to Equation 3.1. This takes up $n-3$

slots and leaves 3 slots available on each of the branches in the network that connect node s to node d. For flow f', only one of the branches can be used and the 3 transmissions that are necessary to transfer a frame from node s to node d use the remaining 3 micro slots. Thus, the maximum feasible capacity requirement in the single-path approach is $\text{FCAP}(f')_s = \frac{n_\text{F}}{d_{ms}}$. For the multi-path solution, the flows f_i ($i \in \{1, \ldots, n\}$) can be handled as in the single-path solution. However, the capacity available on the n branches can now be used in conjunction, obtaining a feasible capacity requirement of $\text{FCAP}(f')_m = n \cdot \frac{n_\text{F}}{d_{ms}}$. This is obviously the maximum feasible value, because nodes s and d are already at the limit of the capacity they can send into or receive from the network, respectively. Thus, the performance gain of the multi-path approach over the single-path approach is $\frac{\text{FCAP}(f')_m}{\text{FCAP}(f')_s} = n$. We conclude that there is no upper bound on the performance gain achievable due to multi-path routing if arbitrary networks are considered. However, if the number of slots is fixed in advance, there is the trivial upper bound of $n_\mathcal{S}$ (with the exception of scenarios that are infeasible for the single path approach), which is due to the maximum capacity the source and sink nodes can handle.

For the metrics discussed in Chapter 2, i.e. capacity usage, latency and energy consumption, the length of the paths forming the solution directly influence the metric value. Thus, when comparing the single-path and multi-path approaches, an interesting question to ask is how the maximum path lengths of single-path and multi-path solutions relate to each other. It turns out that there are networks for which scenarios can be constructed that yield arbitrarily worse metric values under the single-path approach than under the multi-path approach. The construction of such a network is illustrated in Figure 3.6.

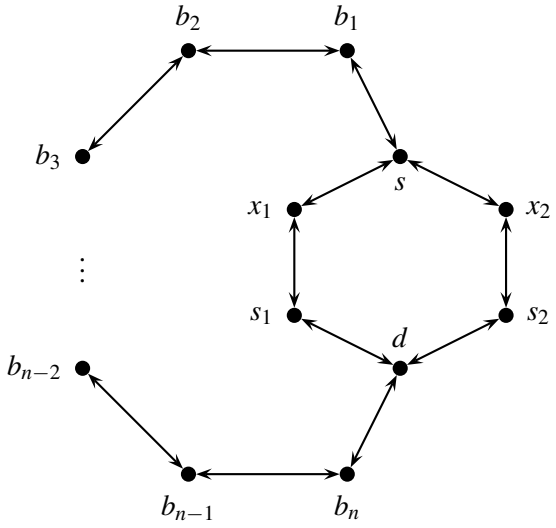


Figure 3.6: A network that potentially forces a single-path solution to use the detour via the b_i nodes for a flow from node s to node d while the multi-path approach can split the flow to use the shorter (s, x_i, s_i, d) paths.

The network in the figure is composed of a hexagon with a data flow that emerges at node s and flows to node d. There is another path connecting nodes s and d formed by n nodes whose transmissions do not interfere with those originating from the hexagon nodes, except for the edges adjacent to s or d. Again, the TDMA parameters n_F, d_s and d_{ms} are chosen such that each node can transfer exactly one frame per macro slot and the number of available

micro slots is fixed to $n_{\mathcal{S}} := 6$. Consider a scenario $\mathcal{F} := \{f_1, f_2, f'\}$ with the following parameters:

$$\forall i \in \{1,2\} : \text{FSOURCE}(f_i) := s_i \tag{3.11}$$

$$\forall i \in \{1,2\} : \text{FSINK}(f_i) := d \tag{3.12}$$

$$\forall i \in \{1,2\} : \text{FCAP}(f_i) := \frac{n_{\text{F}}}{d_{ms}} \tag{3.13}$$

$$\text{FSOURCE}(f') := s \tag{3.14}$$

$$\text{FSINK}(f') := d \tag{3.15}$$

$$\text{FCAP}(f') := 2 \cdot \frac{n_{\text{F}}}{d_{ms}} \tag{3.16}$$

Solutions to the single-path approach are subject to the following observation: Assume flow f' is handled via one of the paths $p_i = (s, x_i, s_i, d)$, $i \in \{1,2\}$. There are only 6 slots available, thus each of the nodes s, x_i and s_i transmit in exactly 2 of the slots. However, under the graph-based noninterference model, these transmissions block any outgoing transmission of flow f_i at s_i. Since there are no more unused slots, flow f_i cannot be handled. We conclude that f' cannot take path p_i but must be routed via the detour path $(s, b_1, \ldots, b_n, d)$ under the single-path approach. A multi-path solution can split flow f' such that both paths p_1 and p_2 can be used in parallel. Each of the 6 transmissions can be handled in its own slot, the single transmissions that are required to handle f_1 and f_2 can share the slot with the respective transmission of flow f' on edges (s, x_2) and (s, x_1), respectively.

Now consider the metric values corresponding to the optimal single-path and multi-path solutions, respectively. For the capacity usage metric, the metric value of the optimal single-path solution is $2(n+1)+2$, while the optimal multi-path solution yields a value of 8. The optimal latency-maximum metric value is $n+1$ in the single-path case and 3 in the multi-path case. A similar effect also applies to the energy consumption metric, i.e. the optimal single-path metric value is variable in n, while for the multi-path approach it is constant. Thus, the metric value for the optimal single-path solution can be increased arbitrarily by inserting additional nodes on the detour path. This means that the performance gain realized by the multi-path approach when compared to the single-path approach has no upper bound.

Although the previous paragraphs show that there are scenarios in which the performance gain of multi-path solutions can be arbitrarily large, these scenarios are constructed cases and it is not expected they show up very often in practical networks. In the following, we look into topological connectivity graph features that represent bottleneck situations and thus prevent arbitrary performance gains of multi-path solutions over their single-path counterparts, provided all paths that connect the source node to the sink node have to traverse the critical region. For simplicity, we restrict the discussion to symmetric connectivity graphs, only consider the graph-based noninterference model and assume that each node can transmit exactly one frame per macro slot to one of its neighbors. Furthermore, interactions between different flows are ignored, i.e. only one flow in an otherwise idle network is considered.

An obvious bottleneck feature if present in a connectivity graph is a common edge $(v, w) \in \mathcal{E}$ that is part of all paths connecting source node a to destination node b (cf. Figure 3.7). If so, the upper bound on the capacity requirement for the flow is the capacity that can be transferred via (v, w). In the case that $a = v$ and $b = w$ (cf. Figure 3.7a), the maximum feasible capacity requirement is obviously $n_{\mathcal{S}} \cdot \frac{n_{\text{F}}}{d_{ms}}$, i.e. $n_{\mathcal{S}}$ frames per macro slot. In this

situation, the single-path solution is optimal, thus there is no performance gain if multiple paths are used. The case of either $a = v$ or $b = w$ is treated below in the paragraph that discusses the effect of a node that is common to all paths. The only case left is that of (v, w) being an intermediate edge of the paths connecting node a to node b (cf. Figure 3.7b).[2] In this case the transmissions that are required to transfer frames on the incoming edges of v and the outgoing edges of w further reduce the feasible capacity requirement for the flow. Observe that each frame travelling through (v, w) takes up three micro slots: One when arriving at v, one for the transmission from v to w and another one for the outgoing transmission at w. During any of these three transmissions other transmissions can be performed neither on (v, w) itself, on incoming edges of v, nor on outgoing edges of w, even if these edges belong to different paths. Thus, each transmitted frame requires three micro slots in the considered area of the network, limiting the maximum feasible capacity requirement for a flow to $\left\lfloor \frac{n_S}{3} \right\rfloor \cdot \frac{n_F}{d_{ms}}$. Again, using multiple paths to handle the flow does not improve performance over the single-path solution due to the assumption that all paths have to traverse (v, w).

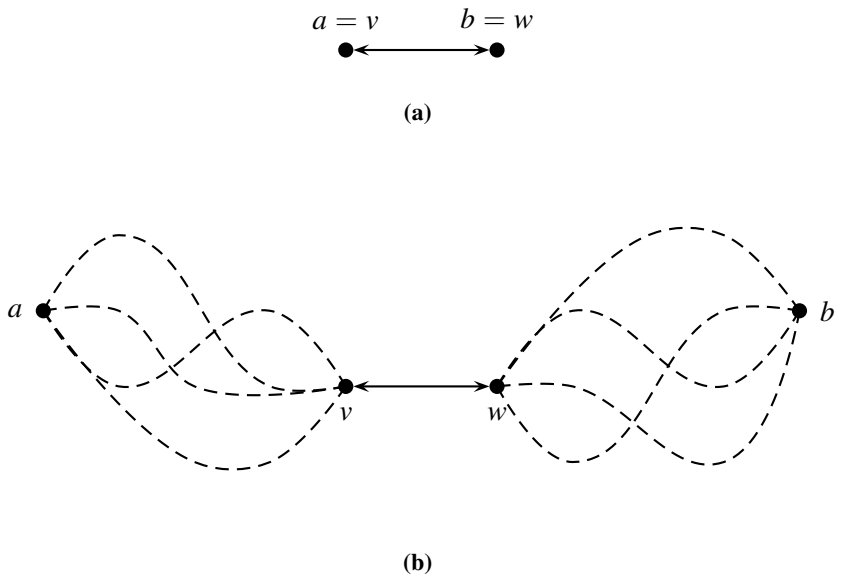


Figure 3.7: Possible situations for which a single edge (v, w) limits the maximum feasible capacity requirement for a flow f that has a as its source and b as its sink node. We consider the case (a) of the edge being the only one in the path and (b) (v, w) being an intermediate edge.

A second bottleneck situation is that of a single node v being member of every path that can be used to handle the flow. The cases $v = a$ and $v = b$ need not be considered here, since these do not restrict network topology. Hence, v is assumed to be an intermediate node of the paths that can potentially be used to handle the flow, i.e. it has one incoming and one outgoing edge on each path (cf. Figure 3.8). Observe that each frame traversing the bottleneck node v claims two micro slots, one for the incoming transmission into v from one of the neighbours $l \in \{l_1, \ldots, l_n\}$ and another one for the outgoing transmission from

[2] An example for a setup in which this situation arises is that of two groups of nodes that are only interconnected via a single link.

v to one of nodes $r \in \{r_1, \ldots, r_m\}$. Hence, the maximum capacity that node v can handle is $c_b = \left\lfloor \frac{n_S}{2} \right\rfloor \cdot \frac{n_F}{d_{ms}}$. For single-path solutions, the capacity bound of $c_p = \left\lfloor \frac{n_S}{3} \right\rfloor \cdot \frac{n_F}{d_{ms}}$ of Equation 3.1 applies as long as the path is of minimum length 3. In this case, the capacity gain achievable when using multi-path routing is $\frac{c_b}{c_p} \approx 1.5$, depending on the particular value of n_S and the corresponding floor function results.

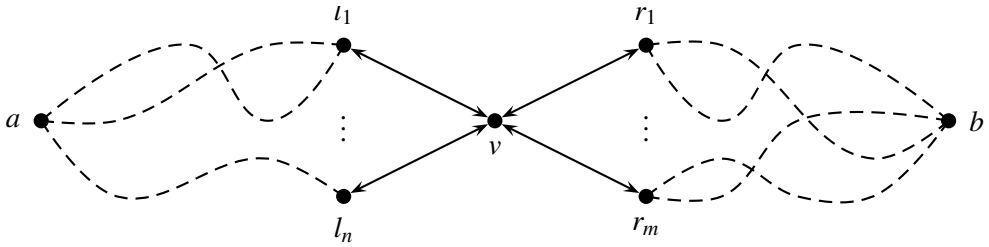


Figure 3.8: A single node v that is part of every path from source node a to sink node b acts as a bottleneck.

The multi-path gain can only be realized if the multi-path solution is capable of using the total available capacity at v. In order to do so, $\left\lfloor \frac{n_S}{2} \right\rfloor$ frames must pass through v during the n_S micro slots. Thus, roughly every second slot a frame must be available for an incoming transmission and likewise the receiving nodes $r_1, \ldots, r_m$ need to be able to accept a frame every second micro slot. If one of the neighbour nodes is actually the source or sink node, i.e. $l = a$ or $r = b$, this node can transmit or receive frames at the necessary rate. Otherwise, the multi-path solution needs at least two neighbor nodes connecting the incoming or outgoing section of the paths passing through v, i.e. $m, n \geq 2$. This is because a single neighbor node (i.e. $m = 1$ or $n = 1$) would turn the bottleneck into the single edge situation described above, which exhibits a smaller capacity bound.

A third class of bottleneck situations is caused by cliques in the graph with which all paths that connect the source to the sink node share an edge; see Figure 3.9 for an illustration. Due to the definition of the graph-based noninterference model and the assumption of a symmetric connectivity relation, only one of the clique nodes can transmit successfully at a time. Moreover, when one node of the clique is transmitting, this transmission will be heard by all other nodes in the clique, thus no other incoming transmission can be performed, even by nodes that are not clique members.

For the capacity that the clique can handle this means each frame passing through the clique claims its own micro slot. Unless the source or the sink node are also clique nodes, more slots are needed for incoming and outgoing transmissions for frames that arrive at and leave the clique. The exact number of slots that are needed depends on the number of clique nodes that handle incoming and outgoing transmissions. Let n_c denote the number of clique nodes, $n_i = k$ the number of clique nodes that handle incoming transmissions. Then, at most $n_o = n_c - k$ nodes are available to handle the outgoing transmissions. With these parameters, the following bound on the number of frames per macro slot n that can be handled by the clique applies:

$$\left\lceil \frac{n}{n_i} \right\rceil + \left\lceil \frac{n}{n_o} \right\rceil + n \leq n_S \tag{3.17}$$

The first two terms of Equation 3.17 state the number of micro slots that are required for the

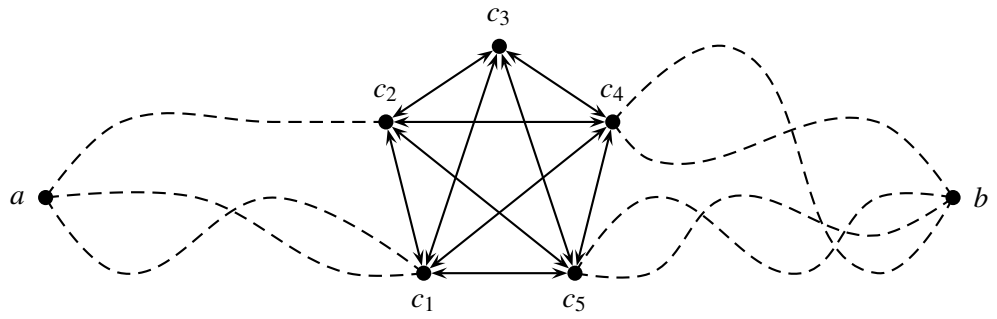


Figure 3.9: The nodes c_1, c_2, c_3, c_4 and c_5 form a clique and act as a bottleneck if all possible paths that connect source node a to sink node b traverse the clique.

incoming and outgoing transmissions, respectively. The third accounts for the transmission of the frames between clique members. Replacing the n_i and n_o variables with their definition, weakening the condition by dropping the ceiling functions and rearranging yields:

$$n \leq \frac{n_{\mathcal{S}}}{\frac{1}{k} + \frac{1}{n_c - k} + 1} \tag{3.18}$$

For computing the bound, a value for the parameter k is needed that describes how many input and output clique nodes, respectively, are available. However, interpreting Equation 3.18 as a function in k with $k \in [0, n_c]$, it is straightforward to calculate the value for k that maximizes the right hand side of Equation 3.18. As expected, that value is $k^* = \frac{n_c}{2}$. Substituting k^* for k and noting that only integers are allowed for the result, we arrive at the following generic bound for the maximum number of frames a clique can handle per macro slot:

$$n \leq \left\lfloor \frac{n_{\mathcal{S}}}{\frac{4}{n_c} + 1} \right\rfloor \tag{3.19}$$

Thus, an upper bound on the maximum feasible capacity requirement in the presence of a clique bottleneck with n_c nodes is

$$c_c = \left\lfloor \frac{n_{\mathcal{S}}}{\frac{4}{n_c} + 1} \right\rfloor \cdot \frac{n_{\mathrm{F}}}{d_{ms}} \tag{3.20}$$

Note that for the trivial clique consisting of only a single edge, we have $n_c = 2$ and consequently $n \leq \left\lfloor \frac{n_{\mathcal{S}}}{3} \right\rfloor$. This coincides with our discussion of single-edge bottleneck situations above.

Compared to the trivial maximum capacity requirement of $n_{\mathcal{S}} \cdot \frac{n_{\mathrm{F}}}{d_{ms}}$ due to the limited outgoing and incoming transmission opportunities at the source and sink node, respectively, the bounds discussed above seem rather slack. However, if the bottleneck situation does not only affect one flow but several, the capacity bounds imposed grow more important, because the flows have to share the available capacity. Of course, the list of bottleneck situations discussed above is by no means exhaustive. When a particular network under consideration does not exhibit any of the topological features discussed above, one can combine them: The edge bottleneck situation can be applied to each edge of a network cut, the combined capacity bound

is obviously bounded by the sum of the individual capacity bounds. Likewise, it may possible to decompose the network into two subnetworks such that their connection is covered by a set of cliques. Again the sum of the individual clique capacity bounds is an upper bound on the total capacity.

The results presented above suggest that there is potential for the multi-path approach to deliver better solutions. However, actual protocols that implement multi-path routing will be more complex to their single-path counterparts, because more than one route must be discovered and maintained. Whether it is sensible to employ multi-path routing depends on the magnitude of the performance gain realized in typical scenarios. Experiments have been conducted in order to answer this question; the results can be found in Section 3.9.

3.4 Solving scenarios

As discussed in the modeling chapter, solving a scenario involves allocating slots in a way that satisfies the noninterference condition, then mapping the allocated slots to flows such that each flow is given enough capacity to satisfy its capacity requirement (cf. Definition 18). While the two-stage construction of a solution from separate slot allocation and flow mapping functions has the benefit of separating the noninterference constraints from the assignment of capacity to flows, this formulation suggests to first find the slot allocation and then the flow mapping and is thus not very suitable for actually constructing solutions incrementally. A better approach when looking for solutions is to use a problem formulation that is centered around single allocations of transmission opportunities. The various conditions such as noninterference and satisfying capacity requirements are then formulated as constraints that must be met when making allocations. The following definition introduces this formulation of the problem:

Definition 26 (Allocation-centric problem formulation)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph and $\mathcal{F}$ a scenario. A set $\mathcal{A} \subseteq \mathcal{F} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$ describes a set of allocations, i.e. $(f, n, (a, b), s) \in \mathcal{A}$ indicates there is capacity allocated for transferring a single frame of path number n of flow f from node a to node b during slot s. $\mathcal{A}$ is called an allocation set *for scenario $\mathcal{F}$ if it meets the following conditions:*

Timing constraint *The time it takes a node to transmit frames for all allocations in a slot may not exceed the micro slot duration:*

$$\forall v \in \mathcal{V}\ \forall s \in \mathcal{S} : \sum_{(w,f,n) \in \mathcal{A}^T(v,s)} \mathcal{T}((v,w),s) \leq d_s \tag{3.21}$$

where $\mathcal{A}^T(v,s) := \{(w,f,n) \in \mathcal{V} \times \mathcal{F} \times \mathbb{N} \mid (f,n,(v,w),s) \in \mathcal{A}\}$.

Noninterference constraint *None of the allocations may interfere with any other allocation. Let Φ be the noninterference model. Then the allocation set must meet the following condition:*

$$\forall((v,w),s) \in \mathcal{A}^I : \Phi(v,w,s) \tag{3.22}$$

For the noninterference check only the edge and slot is relevant, thus the set of allocations to be considered is $\mathcal{A}^I := \{(e,s) \in \mathcal{E} \times \mathcal{S} \mid \exists f \in \mathcal{F}\ \exists n \in \mathbb{N} : (f,n,e,s) \in \mathcal{A}\}$.

For the noninterference condition specified by the graph-based and protocol noninterference models to be properly defined, the transmission predicate TX *must be defined on a per slot basis:*

$$\text{TX}(a,s) := \begin{cases} 1 & \text{if } \exists b \in \mathcal{V}\ \exists f \in \mathcal{F}\ \exists n \in \mathbb{N} : (f,n,(a,b),s) \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{3.23}$$

If the physical noninterference model is used, the definition of the transmission power function TXP *is required:*

$$\text{TXP}(a,b,s) := \begin{cases} P_{a,b,s} & \text{if } \exists f \in \mathcal{F}\ \exists n \in \mathbb{N} : (f,n,(a,b),s) \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

The constant $P_{a,b,s}$ is a parameter that specifies the transmission power node a uses for transmitting to node b in slot s. In the scenarios that are considered in this thesis, the transmission power is assumed to be constant for all nodes and slots.

Capacity constraint *The number of paths allocated for a flow must match the flow's capacity requirement. Given a flow $f \in \mathcal{F}$, the number of paths to allocate is:*

$$n^f = \left\lceil \frac{\text{FCAP}(f) \cdot d_{ms}}{n_F} \right\rceil \tag{3.25}$$

Thus, reservations should only be made for these n^f paths:

$$\forall f \in \mathcal{F}\ \forall i \in \{n^f + 1, \ldots\} : |\mathcal{A}^P(f,i)| = 0 \tag{3.26}$$

The set $\mathcal{A}^P(f,i) := \{(f',n,e,s) \in \mathcal{A} \mid f' = f \wedge i = n\}$ is the subset of $\mathcal{A}$ that contains the allocations for path i of flow f.

Path constraint *For each path index exists a path that starts at the source node of the corresponding flow.*

$$\forall f \in \mathcal{F}\ \forall i \in \{1, \ldots, n^f\} : \exists p \in \mathcal{P}_\mathcal{G} : \\ v_1^p = \text{FSOURCE}(f) \wedge \forall e \in \mathcal{E} : |A^N(f,i,e)| = 1 \leftrightarrow e \wr p \tag{3.27}$$

The requirement for the cardinal number of the set $A^N(f,i,e) := \{(f',n,e',s) \in \mathcal{A} \mid f = f' \wedge e = e' \wedge n = i\}$ to match 1 guarantees that there is exactly 1 allocation for each hop on the path.

Note that compared to the original problem formulation, the above formulation is different in the way paths are used. In the original formulation, a path is associated with a number of frames it can transfer per macro slot. Here, the capacity is first translated into a frames per macro slot requirement n^f and consequently there must be n^f paths to handle these frames. Whereas in the original formulation frames taking the same route through the network would be handled by a single path of appropriate capacity, the revised formulation is designed to assign a path per frame per macro slot, but allowing the paths to be identical. An important property of the allocation-centric formulation is that the paths that are considered in the path constraint are uniquely determined:

Proposition 4 (Uniqueness of paths in an allocation set)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ be a scenario and $\mathcal{A} \subseteq \mathcal{F} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$ be an allocation set. Let $f \in \mathcal{F}$ be a flow and $i \in \{1, \ldots, n^f\}$ be a path index. Let $p_1, p_2 \in \mathcal{P}_\mathcal{G}$. If both $p \in \{p_1, p_2\}$ satisfy the path constraint of Definition 26

$$v_1^p = \text{FSOURCE}(f) \wedge \forall e \in \mathcal{E} : |A^N(f, i, e)| = 1 \leftrightarrow e \wr p \tag{3.28}$$

then the paths are equal, i.e. $p_1 = p_2$. Thus the path that is considered in the path constraint is uniquely determined. In the following, we refer to this path by $\mathcal{P}_{f,i}^{\mathcal{A}}$.

PROOF (PROPOSITION 4)
Let $p_1, p_2 \in \mathcal{P}_\mathcal{G}$ be two paths that both satisfy the path constraint of Equation 3.28. Assume that $p_1 \neq p_2$. By constructing a contradiction, we will prove the assumption wrong.

Let $p' \in \mathcal{P}_\mathcal{G}$ be the largest common prefix of p_1 and p_2. Without loss of generality assume $|p_2| < |p_1|$. Thus, there is an edge $e = (v_{|p'|+1}^{p_1}, v_{|p'|+2}^{p_1})$ that is not part of p_2. Due to Equation 3.28, we have $|A^N(f, i, e)| = 1$, but this directly contradicts the path constraint for p_2, since $\neg(e \wr p_2)$. Thus the assumption was wrong, p_1 cannot be different from p_2, i.e. $p_1 = p_2$. □

Based on the uniqueness of the allocated paths, solutions are characterized as follows:

Definition 27 (Solutions in the allocation-centric problem formulation)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ a scenario, and $\mathcal{A} \subseteq \mathcal{F} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$ an allocation set. $\mathcal{A}$ is a solution to *$\mathcal{F}$ if the paths in $\mathcal{A}$ are complete and meet the latency requirements:*

$$\forall f \in \mathcal{F}\, \forall i \in \{1, \ldots, n^f\} : v_{|\mathcal{P}_{f,i}^{\mathcal{A}}|+1}^{\mathcal{P}_{f,i}^{\mathcal{A}}} = \text{FSINK}(f) \tag{3.29}$$

$$\forall f \in \mathcal{F} \; : (\text{FLAT}(f) \neq \infty) \rightarrow \left(\max_{i \in \{1,\ldots,n^f\}} \text{APLAT}(\mathcal{A}, f, i) \leq \text{FLAT}(f) \right) \tag{3.30}$$

The allocation-centric formulation allows to construct solutions incrementally in a relatively straightforward way. Starting with the empty allocation set (which trivially meets Definition 26), single reservations are made, thereby constructing the paths. The timing and noninterference constraints are checked after each added hop in order to verify that the configurations considered stay valid. Since the number of paths and their sources and sinks are known in advance from the scenario parameters, the task is to try and construct appropriate paths without violating the timing and noninterference constraints. This approach has been taken for the branch and bound solver, which is described in Section 3.7.

3.4.1 Metrics

In order to calculate metric values according to the metrics defined in the modeling chapter, solutions in the allocation-centric problem formulation can be converted to the formulation used in the modeling chapter. It is also possible to device corresponding metric definitions for solutions given in the allocation-centric problem formulation. The capacity usage metric is just the number of allocations. Given a solution $\mathcal{A}$ in the allocation-centric formulation the metric value is the cardinal number $|\mathcal{A}|$ of the allocation set.

Providing a logically separate path per frame per macro slot simplifies the calculation of the latency value, because there is only a single allocation at each hop for that particular

path. Recall that the latency of a flow is defined to be the maximum latency of all paths that belong to the flow. The latency of a path in the allocation-centric formulation can easily be calculated by comparing the slot numbers that are used for each hop. The slot numbers are uniquely determined due to the path constraint because each path is allocated exactly one transmission opportunity per macro slot for each of its hops. Thus, the latency for path i of flow f in a solution $\mathcal{A}$ can be expressed as:

$$\text{APLAT}(\mathcal{A}, f, i) := 1 + \sum_{j=2}^{|\mathcal{P}^{\mathcal{A}}_{f,i}|} \delta(s_{A,f,i,j}, s_{A,f,i,j-1}) \tag{3.31}$$

$$\delta(i,j) = \begin{cases} i-j & i > j \\ i-j+n_{\mathcal{S}} & i \leq j \end{cases} \tag{3.32}$$

The term $s_{\mathcal{A},f,i,j}$ denotes the slot in which the allocation for the j-th hop of path i of flow f is made. Note that $s_{\mathcal{A},f,i,j}$ is uniquely determined due to the path constraint of Definition 26. The δ function computes latency caused by a frame waiting for its outgoing slot at a node. Thus, the maximum latency metric expressed in terms of the allocation-centric model is:

$$\text{ALAT}(\mathcal{A}) := \max_{f \in \mathcal{F}} \max_{i \in \{1,\ldots,n^f\}} \text{APLAT}(\mathcal{A}, f, i) \tag{3.33}$$

3.5 Complexity of the slot allocation problem

In this section, we consider the slot allocation problem, i.e. finding a solution to a scenario in the allocation-centric formulation. Given the combinatorial characteristics of the slot allocation problem, especially w.r.t. the choice of slots for a given flow, it is natural to ask whether the problem is NP-complete. As will be proved in this section, this is indeed the case. The consequence is that there is no algorithm that solves the problem in polynomial time unless P = NP. Thus, we should expect that only small instances of the problem are solvable within reasonable time and memory limits. The NP-completeness proof is by reduction of the graph coloring problem: Given an undirected graph and a number of colors, the question is whether there is a mapping of nodes to colors such that no pair of adjacent nodes shares the same color. This problem appears in Karp's collection of NP-complete problems [25] as chromatic number problem. For simplicity, we will only consider the graph-based noninterference model.

Proposition 5 (NP-completeness of the slot allocation problem)
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph, $\mathcal{F}$ be a scenario. The problem of deciding whether there is an allocation set $\mathcal{A} \subseteq \mathcal{F} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$ according to Definition 27 is in the class of NP-complete problems.

PROOF (PROPOSITION 5)
The proof is by reduction of the graph coloring problem. First, a polynomial-time transformation that maps instances of the graph coloring problem to corresponding instances of the slot allocation problem is given. Then, it is established that the answer to an instance of the graph coloring problem is yes if and only if the corresponding instance of the slot allocation problem is solvable. For the proof to be complete, an argument must be given that establishes

the slot allocation problem's membership in NP. This is obviously the case, because a randomly guessed allocation set can be checked for correctness in polynomial time by testing the constraints listed in Definitions 26 and 27.

Let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be the undirected graph and $\{1, \ldots, k\}$ the set of available colors given by the graph coloring problem instance. For the construction of the connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the slot allocation problem instance, new sets of nodes and edges are introduced:

$$\mathcal{V} := \{p_v \mid v \in \mathcal{V}'\} \cup \{q_v \mid v \in \mathcal{V}'\} \tag{3.34}$$

$$\mathcal{E} := \{(p_{v_1}, q_{v_2}) \mid (v_1, v_2) \in \mathcal{E}'\} \tag{3.35}$$

As mentioned above, the problem is considered under the graph-based noninterference model. The number of slots available is set to k. The micro slot and macro slot durations are 1 s and k s, respectively. The common transmission rate of the nodes is fixed such that exactly one frame can be transmitted during a single micro slot, i.e. $\mathcal{T}((v, w), s) = 1\,\text{s}$ for each $(v, w) \in \mathcal{E}$ and $s \in \mathcal{S}$. Finally, the scenario that is to be handled by the network is defined as follows:

$$\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{V}'|} := \{f_v \mid v \in \mathcal{V}'\} \tag{3.36}$$

$$\text{FSOURCE}(f_v) := p_v \tag{3.37}$$

$$\text{FSINK}(f_v) := q_v \tag{3.38}$$

$$\text{FCAP}(f_v) := \frac{n_\text{F}}{d_{ms}} \tag{3.39}$$

$$\text{FLAT}(f_v) := \infty \tag{3.40}$$

It is obvious that the transformation as specified above can be performed in polynomial time w.r.t. the number of nodes $|\mathcal{V}'|$ and number of edges $|\mathcal{E}'|$ of the graph given by the graph coloring problem instance.

Now we need to verify that a graph coloring problem instance I_C is solvable if and only if the corresponding slot allocation problem instance I_S has a solution. Assume that a given graph coloring instance given by the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ and number of colors k has a solution. Thus, there is a mapping $c : \mathcal{V}' \to \{1, \ldots, k\}$ that assigns colors to nodes such that:

$$\forall (v_1, v_2) \in \mathcal{E}' : c(v_1) \neq c(v_2) \tag{3.41}$$

Consider the following allocation set:

$$\mathcal{A} := \{(f_v, 1, (p_v, q_v), c(v)) \in \mathcal{F} \times \mathcal{N} \times \mathcal{E} \times \mathcal{S} \mid v \in \mathcal{V}'\} \tag{3.42}$$

To prove that $\mathcal{A}$ is indeed a solution to I_S, it must be verified that $\mathcal{A}$ meets the allocation set constraints and that all paths are complete:

Timing constraint: There is only one allocation in $\mathcal{A}$ for each sending node p_v, $v \in \mathcal{V}'$. Each frame takes 1 s to transmit and the micro slot duration is 1 s, thus the timing constraint holds.

Noninterference constraint: The set of reservations that must be checked for noninterference according to Equation 3.22 is:

$$\mathcal{A}^I = \{((p_v, q_v), c(v)) \in \mathcal{E} \times \mathcal{S} \mid v \in \mathcal{V}'\} \tag{3.43}$$

Recall the noninterference condition for a transmission from node a to node b during slot s as stated by the graph-based noninterference model:

$$\Phi(a,b,s) \equiv \neg \text{TX}(b,s) \wedge \forall v \in \mathcal{G}_b^i \setminus \{a\} : \neg \text{TX}(v,s) \tag{3.44}$$

Thus, let $((p_v, q_v), f(v)) \in \mathcal{A}^I$ be a reservation for some $v \in \mathcal{V}'$. Due to definition of $\mathcal{A}$ (i.e. only p_x nodes are senders) and the transmission predicate for the graph-based noninterference model (cf. Equation 3.23) we have $\neg \text{TX}(q_v, s)$. Let $w \in \mathcal{G}_{q_v}^i \setminus \{p_v\}$. Due to Definition of $\mathcal{G}$, there must be some $v' \in \mathcal{V}'$ such that $p_{v'} = w$. Since $(p_{v'}, q_v) \in \mathcal{E}$, by construction we have $(v', v) \in \mathcal{E}'$ and thus $c(v) \neq c(v')$. Then, by definition of $\mathcal{A}$, the only slot during which node $p_{v'}$ is active is $c(v')$ and we have $\neg \text{TX}(p_{v'}, c(v))$. We conclude that the noninterference condition holds.

Capacity constraint: By construction, the number of paths to allocate is $n^f = 1$ for each flow $f \in \mathcal{F}$. Thus, $\mathcal{A}$ meets the constraint.

Path constraint: For each flow, $\mathcal{A}$ allocates exactly one edge that connects the source with the sink node. Thus, the path constraint holds.

It is obvious that all paths allocated in $\mathcal{A}$ are complete. For each flow, there is exactly one allocation along the edge between source and sink.

Vice versa, assume there is an allocation set $\mathcal{A} \subseteq \mathcal{F} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$ that meets the constraints of Definitions 26 and 27. As will be shown, then there is a color assignment function that solves I_C. Consider an arbitrary node $v \in \mathcal{V}'$. Due to the path constraint, there is a path that connects p_v to q_v for which $\mathcal{A}$ provides exactly one allocation along the edges being part of the path. By construction, the only path from p_v to q_v is the direct edge (p_v, q_v). This is because there are only edges from p_x nodes to q_y nodes but no edges between pairs of p_x nodes or q_y nodes, respectively. Thus, there is a unique slot number $s_v \in \{1, \ldots, k\}$, such that $(f_v, 1, (p_v, q_v), s_v) \in \mathcal{A}$. We define the coloring function $c : \mathcal{V}' \to \{1, \ldots, k\}$ as $c(v) := s_v$.

Now we need to verify that for this definition of c we have $c(v_1) \neq c(v_2)$ for each edge $(v_2, v_1) \in \mathcal{E}'$. The proof is by contradiction, thus assume there is an edge $(v_2, v_1) \in \mathcal{E}'$ such that $c(v_1) = c(v_2)$.

By definition of c, there is a slot $s \in \{1, \ldots, k\}$ such that $c(v_i) = s$ for both $i \in \{1, 2\}$:

$$(f_{v_i}, 1, (p_{v_i}, q_{v_i}), s) \in \mathcal{A} \tag{3.45}$$

Thus, by Equation 3.23 we have $\text{TX}(p_{v_i}, s)$ for both $i \in \{1, 2\}$. Consider the noninterference condition $\Phi(p_{v_1}, q_{v_1}, s)$ for the transmission from from p_{v_1} to q_{v_1} during slot s. Due to $(v_2, v_1) \in \mathcal{E}'$, by construction there is an edge $(p_{v_2}, q_{v_1}) \in \mathcal{E}$. Thus, we have $p_{v_i} \in \mathcal{G}_{q_{v_1}}^i$ and p_{v_i} transmitting during slot s for $i \in \{1, 2\}$, i.e. the transmissions interfere at q_{v_1}. This is a contradiction to the fact that $\mathcal{A}$ meets the noninterference constraint. Therefore, the assumption was wrong and the opposite holds: For each edge $(v_1, v_2) \in \mathcal{E}'$ the colors assigned by c are different. □

Note that from a formal point of view the NP-completeness of the slot allocation problem does at first not provide further insight about the difficulty of finding an optimal solution to the slot allocation problem w.r.t. a given metric. However, for example for the capacity usage metric and the latency metrics there are upper bounds on the metric value that any solution

will meet. Thus, by giving the decision version of the optimization problem an upper bound for the metric value to achieve, the decision version of the optimization problem has been reduced to the problem of finding any feasible slot allocation, which has just been proved NP-complete. Thus, also the optimization problems that allow construction of an upper bound for the metric value are NP-hard.

3.6 Mixed Integer Programming Models

An advantage of the allocation-centric problem formulation is that it easily lends itself to a transformation into a standard Mixed Integer optimization problem. The idea is to introduce a binary variable for every possible hop allocation that could be made. What is left is to formulate the constraints and the metric that is to be optimized as linear expressions. Finally, the problem can be solved using standard optimization methodology.

Recall the standard model of linear optimization known as a linear program: Given a coefficient matrix $A \in \mathbb{R}^{m \times n}$, a bound vector $b \in \mathbb{R}^m$ and an objective function coefficient vector $c \in \mathbb{R}^n$, the goal is to find a variable vector $x \in \mathbb{R}^n$ that optimizes the objective function under the constraints given by A and b:

$$\text{minimize or maximize} \sum_i c_i x_i \tag{3.46}$$

$$\text{subject to } Ax \leq b \tag{3.47}$$

There are algorithms that solve the linear programming problem in polynomial time. However, the standard method for solving linear programs is the simplex algorithm, which is efficient for most practical applications although its running time is exponential in problem size in the worst case. Mixed Integer Programs are Linear Programs with the additional constraint for some variables to be natural numbers. Programs whose variables are further restricted to binary values are referred to as 0-1 Integer Programs or Binary Integer Programs. Both solving Mixed Integer Programs and Binary Integer Programs is NP-hard, in fact the decision version of the Binary Integer Programming problem is part of Karp's collection of NP-complete problems [25].

As mentioned before, the Mixed Integer Program formulation is based on the idea to introduce a decision variable for each capacity allocation that can be made. Thus, for each tuple of flow f, path index n, edge e and slot s the binary variable $x_{e,s}^{f,n}$ decides whether there is capacity allocated to that flow and path during the corresponding slot on the edge. Note that the number of variables can be cut down quite significantly by only using a subset of all edges specific to each flow, for example only the edges that belong to paths that are within a certain distance with the shortest path in terms of path length. In order to be able to formulate the constraints, another set of variables is introduced that indicate edge usage. For each pair of edge e and slot s, the binary variable $x_{e,s}$ determines whether the edge is in use during the slot by any of the paths. Of course, the edge usage variables $x_{e,s}$ need to be bound to the edge allocation variables $x_{e,s}^{f,n}$. This is subject of the first set of constraints that ensures an edge usage variable is set to one if any of the corresponding allocation variables is non-zero:

$$\forall e \in \mathcal{E}\ \forall s \in \mathcal{S} : \sum_{f \in \mathcal{F}} \sum_{n=1}^{n^f} (x_{e,s}^{f,n} - x_{e,s}) \leq 0 \tag{3.48}$$

Using these variables, we need to formulate the constraints stated in Definitions 26 and 27 as well as an objective function, for which the metric that is to be optimized is used. The timing constraint is captured by the following set of conditions:

$$\forall v \in \mathcal{V}\, \forall s \in \mathcal{S} : \sum_{w \in \mathcal{G}_v^o} \sum_{f \in \mathcal{F}} \sum_{n=1}^{n^f} \mathcal{T}((v,w),s) \cdot x_{(v,w),s}^{f,n} \leq d_s \tag{3.49}$$

For the noninterference constraint, the model constraints to be added are specific to the noninterference model that is considered. The basic idea is to write a condition for each edge variable x_{e_s} that holds if the variable indicates the edge is not used or if it is used and the noninterference condition is met. For the graph-based noninterference model, a suitable set of conditions is given below. When an edge usage variable is 1, all the variables for the interfering edges must be 0 or the condition will be false.

$$\forall (v,w) \in \mathcal{E}\, \forall s \in \mathcal{S} : x_{(v,w),s} + \frac{1}{|\mathcal{G}_w^i|} \sum_{q \in (\mathcal{G}_w^i \setminus \{v\})} x_{(q,w),s} + \frac{1}{|\mathcal{G}_w^o|} \sum_{q \in \mathcal{G}_w^o} x_{(w,q),s} \leq 1 \tag{3.50}$$

For the physical noninterference model, the condition for successful transmission of a message is given by Equation 2.6. The maximum of the transmission power received from other nodes can be captured by introducing continuous helper variables $y_{(v,r),s}$ that give an upper bound on the power potentially received at node r due to nodes v's transmissions in time slot s:

$$\forall r \in \mathcal{V}\, \forall (v,w) \in \mathcal{E}\, \forall s \in \mathcal{S} : \rho(v,w,r,s) \cdot x_{(v,w),s} - y_{(v,r),s} \leq 0 \tag{3.51}$$

A linear formulation of the physical noninterference constraint for use with the Mixed Integer Program is then obtained by rearranging Equation 2.6:

$$\forall (v,w) \in \mathcal{E}\, \forall s \in \mathcal{S} : \rho(v,w,w,s) \cdot x_{(v,w),s} - \gamma \sum_{p \in (\mathcal{V} \setminus \{v\})} y_{(p,w),s} \geq \gamma \cdot N \tag{3.52}$$

Note that this formulation is only correct for the case $x_{(v,w),s} = 1$, i.e. the edge (v,w) is used during slot s. If it is unused ($x_{(v,w),s} = 0$), the inequality in Equation 3.52 is trivially false, which is incorrect, since the noninterference constraint for unused edges must always evaluate to true. This can be fixed by a compensation constant K that must be chosen s.t. $K - \gamma \cdot N$ is large enough to compensate the negative term on the left hand side of Equation 3.52:

$$\forall (v,w) \in \mathcal{E}\, \forall s \in \mathcal{S} : (\rho(v,w,w,s) - K) \cdot x_{(v,w),s} - \gamma \sum_{p \in (\mathcal{V} \setminus \{v\})} y_{(p,w),s} \geq \gamma \cdot N - K \tag{3.53}$$

Note that if $x_{(v,w),s} = 1$, Equation 3.53 is equivalent to Equation 3.52. Otherwise, the compensation term is only active on the right hand side, thus ensuring the inequality is trivially true if the edge that is considered is not in use.

The capacity and path constraints of Definition 26 are formulated in the Mixed Integer Problem as a set of conditions that are inspired by the observation that each path that must be allocated resembles a network flow problem. Thus, the incoming and outgoing allocations at each node must be balanced, except for the source and the sink node that must emit and absorb

a single frame, respectively. Let $f \in \mathcal{F}$ be a flow, $a = \text{FSOURCE}(f)$ and $b = \text{FSINK}(f)$. A set of conditions that ensure there is a complete path from node a to node b is:

$$\forall i \in \{1,\ldots,n^f\} : \sum_{s\in\mathcal{S}} \left(\sum_{w\in\mathcal{G}_a^i} x^{f,i}_{(w,a),s} - \sum_{w\in\mathcal{G}_a^o} x^{f,i}_{(a,w),s} \right) = -1 \tag{3.54}$$

$$\forall i \in \{1,\ldots,n^f\} : \sum_{s\in\mathcal{S}} \left(\sum_{w\in\mathcal{G}_b^i} x^{f,i}_{(w,b),s} - \sum_{w\in\mathcal{G}_b^o} x^{f,i}_{(b,w),s} \right) = 1 \tag{3.55}$$

$$\forall i \in \{1,\ldots,n^f\}\ \forall v \in (\mathcal{V} \setminus \{a,b\}) : \sum_{s\in\mathcal{S}} \left(\sum_{w\in\mathcal{G}_v^i} x^{f,i}_{(w,v),s} - \sum_{w\in\mathcal{G}_v^o} x^{f,i}_{(v,w),s} \right) = 0 \tag{3.56}$$

Note that these constraints allow cycles in the paths and even cycles that are not connected to the path. However, these artifacts normally do not show up, because allocating more capacity typically degrades the metric value. Even if there are unnecessary allocations due to cycles in the solution obtained by solving the Mixed Integer Program, a clean solution can be obtained by removing these cycles.

Finally, an objective function should be added to the Mixed Integer Program that reflects the metric that should be optimized. A linear expression formulation of the capacity usage metric value is:

$$\sum_{f\in\mathcal{F}} \sum_{i=1}^{n^f} \sum_{e\in\mathcal{E}} \sum_{s\in\mathcal{S}} x^{f,i}_{e,s} \tag{3.57}$$

The latency metrics are somewhat problematic. This is because the latency that is caused at a node depends on both the incoming and outgoing allocations. Thus, the value depends on two allocation variables, which suggests a quadratic term that is not possible in the linear model. While it is possible to work around this problem, this does not provide further insight and is therefore not discussed here.

A Mixed Integer Program formulation of the hexagon scenario that was introduced in Figure 3.1 can be found in Appendix A. The formulation listed in the appendix was created manually. The linear program as listed there does not pose a problem for current optimization software packages such as ILOG's CPlex [20]. However, Mixed Integer formulations for larger examples such as the crossing flows scenario of Figure 3.3 are already intractable; both the CPlex [20] and the GLPK [12] solvers were far off the optimal solutions even after hours of computation. In contrast, the program implementing the branch and bound technique described in the next section finds the optimal solution to the crossing flows example within a few seconds.

3.7 Branch and Bound Technique

For obtaining statistics about the relative performance of the multi-path and single-path routing approaches in a large number of randomly generated scenarios, a method for finding the optimal solutions to the scenarios for the multi-path and single-path cases, respectively, is useful. As has just been discussed, the Mixed Integer Program formulation does not perform

well enough to be an option for solving random scenarios. This section introduces a technique based on a branch and bound approach which performs reasonably well given the complexity of the problem.

The branch and bound approach incrementally constructs solutions by adding reservations step by step. In each step, a lower bound is computed for the metric value that can be achieved by completing the current configuration to a solution for the scenario. Once a solution is known, configurations that yield a worse lower metric bound than the metric value of known solution can be discarded.

Figure 3.10 lists the branch and bound algorithm in pseudo code. The input for the algorithm are:

- Connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Noninterference model Φ
- Number of slots $n_{\mathcal{S}}$
- Macro slot duration d_{ms}
- Micro slot duration d_s
- Frame size n_{F}
- Frame duration function $\mathcal{T} : \mathcal{V} \times \mathcal{V} \times \mathcal{S} \to \mathcal{R}$
- Scenario as a sequence of flows $f_1, \ldots, f_k$

The output is the allocation set C^* that describes the optimal solution to the slot allocation problem specified by the input parameters or an empty set if there is no such solution. The basic approach taken by the algorithm is to allocate capacity for the paths of the flows in order, starting at the first path of the first flow. The algorithm works with *configurations* which are partially completed allocation sets that meet the timing and noninterference constraints but still miss reservations to handle all paths of all flows. The configurations the algorithm needs to consider further are managed in the set Q. Along with the configuration itself, the entries in Q also contain further information: The flow index that gives the flow the path that is currently constructed belongs to, the index of this path and the node at which the path under construction currently ends. During each iteration of the loop, a configuration is picked from Q (cf. lines 10–11). Then there are two cases: Either the configuration represents a solution, which is the case when all paths of all flows have been allocated capacity, or the configuration is not complete yet. In the first case (cf. lines 25–29) the metric value of the solution is evaluated and the solution saved if it is the best solution so far. If the paths are not yet complete, additional reservations must be made. After determining the path that needs to be extended and the node v at which this path currently ends, the current configuration is branched by considering all possible extensions to the current path (cf. lines 33–48). This results in new configurations that emerge by adding a reservation from v to a neighbor node. Not all of these new configurations are feasible, those that fail to meet the timing or noninterference constraints are rejected. Furthermore, the lower metric bound of the generated configurations is checked, such that configurations that cannot result in a better solution than the best currently known solution are also discarded. Configurations that meet all the conditions are then added to Q in order to be further considered by the algorithm.

The METRIC : $2^{\mathbb{N}\times\mathbb{N}\times\mathcal{E}\times\mathcal{S}} \to \mathbb{R}$ and LMB : $2^{\mathbb{N}\times\mathbb{N}\times\mathcal{E}\times\mathcal{S}} \to \mathbb{R}$ functions used by the algorithm depend on the metric that is to be optimized. The METRIC function calculates the

1: $Q \leftarrow \emptyset$ *Initialize configurations queue*
2: $C^* \leftarrow \emptyset$ *Best solution found so far*
3: $m^* \leftarrow \infty$ *Metric value of best solution*
4: $a \leftarrow \text{FSOURCE}(f_1)$
5: **for** $(v,s) \in \mathcal{G}_a^o \times \mathcal{S}$ **do** *Create initial configurations*
6: $C \leftarrow \{(1,1,(a,v),s)\}$
7: $Q \leftarrow Q \cup \{(1,1,C,v)\}$
8: **end for**
9: **while** $Q \neq \emptyset$ **do** *Main loop, branches one configuration per iteration*
10: $(j,i,C,v) \leftarrow \min_{B \in Q} \text{PREC}(B)$
11: $Q \leftarrow Q \setminus \{(j,i,C,v)\}$
12: **if** $\text{LMB}(C) \geq m^*$ **then**
13: next iteration
14: **end if**
15: **if** $v = \text{FSINK}(f_j)$ **then** *Current path finished*
16: **if** $i < n^{f_j}$ **then** *Build next path of current flow*
17: $v \leftarrow \text{FSOURCE}(f_j)$
18: $i \leftarrow i+1$
19: **else**
20: **if** $j < |\mathcal{F}|$ **then** *Switch to next flow*
21: $j \leftarrow j+1$
22: $i \leftarrow 1$
23: $v \leftarrow \text{FSOURCE}(f_j)$
24: **else** *All flows handled, configuration is a solution*
25: **if** $\text{METRIC}(C) < m^*$ **then** *Best solution so far*
26: $C^* \leftarrow C$
27: $m^* \leftarrow \text{METRIC}(C)$
28: **end if**
29: next iteration
30: **end if**
31: **end if**
32: **end if**
33: **for** $(w,s) \in \mathcal{G}_v^o \times \mathcal{S}$ **do** *Branch the configuration*
34: $C' = C \cup \{(j,i,(v,w),s)\}$
35: $C^I = \{(e,s) \in \mathcal{E} \times \mathcal{S} \mid \exists i' \in \mathbb{N}\, \exists j' \in \mathbb{N} : (j,i,e,s) \in C'\}$
36: **if** $\exists((v',w'),s') \in C^I : \neg\Phi(v',w',s')$ **then** *Check noninterference constraint*
37: next iteration
38: **end if**
39: $C^T \leftarrow \{(v',j',i') \in \mathcal{V} \times \mathbb{N} \times \mathbb{N} \mid (j',i',(v,v'),s) \in C'\}$
40: $d \leftarrow \sum_{(v',j',i') \in C^T} \mathcal{T}((v,v'),s)$
41: **if** $d > d_s$ **then** *Check timing constraint*
42: next iteration
43: **end if**
44: **if** $\text{LMB}(C') \geq m^*$ **then**
45: next iteration
46: **end if**
47: $Q \leftarrow Q \cup \{(j,i,C',w)\}$
48: **end for**
49: **end while**

Figure 3.10: Branch and bound method for solving the slot allocation problem.

metric value of a solution, definitions for the capacity usage and maximum latency metrics are given in Section 3.4.1. Given a configuration $C \subseteq \mathbb{N} \times \mathbb{N} \times \mathcal{E} \times \mathcal{S}$, $\text{LMB}(C)$ determines a lower bound of the metric value that is achievable when completing configuration C to a solution. For the capacity usage and latency metrics, a reasonable lower bound function can be constructed from the minimum hop distance between the source and destination nodes of the paths that do not have reservations yet. A suitable definition of LMB for use with the capacity usage metric is:

$$|C| + \sum_{f \in \mathcal{F}} \sum_{i=1}^{n^f} \text{DIST}_{\mathcal{G}}(v_{|\mathcal{P}^C_{f,i}|+1}^{\mathcal{P}^C_{f,i}}, \text{FSINK}(f)) \tag{3.58}$$

For the maximum latency metric, a lower bound for the metric value can be computed as:

$$\max_{f \in \mathcal{F}} \max_{i \in \{1,\ldots,n^f\}} \text{APLAT}(C, f, i) + \text{DIST}_{\mathcal{G}}(v_{|\mathcal{P}^C_{f,i}|+1}^{\mathcal{P}^C_{f,i}}, \text{FSINK}(f)) \tag{3.59}$$

When selecting the next configuration to consider from the set of available configurations in Q, the algorithm should pick one that is likely to yield a good solution when completed. The choice is made by the $\text{PREC} : 2^{\mathbb{N}\times\mathbb{N}\times\mathcal{E}\times\mathcal{S}} \to \mathbb{R}$ function that computes a precedence value for a given configuration. The solution with the lowest value in Q is picked to be considered next. In the following definition of PREC, the precedence value is computed from the lower metric bound of a given configuration C and the minimum number of allocations that are missing to complete the configuration to a solution is used as a secondary criterion to distinguish configurations with equal metric bounds:

$$\text{PREC}(C) := \text{LMB}(C) + \frac{\sum_{f \in \mathcal{F}} \sum_{i=1}^{n^f} \text{DIST}_{\mathcal{G}}(v_{|\mathcal{P}^C_{f,i}|+1}^{\mathcal{P}^C_{f,i}}, \text{FSINK}(f))}{|\mathcal{V}|^2 \cdot n_{\mathcal{S}}} \tag{3.60}$$

A C++ implementation of the branch and bound method described above has been constructed in the course of preparing this thesis. This implementation incorporates some additional refinements that improve efficiency. The following observation helps to reduce the number of configurations that are considered when looking for a solution: All paths p_i within a flow start at the same node and have their first hop allocated during some slot s_i. Because the order of these paths is irrelevant, it is not necessary to consider all ${n_{\mathcal{S}}}^i$ possible ordered tuples of starting slots, but only the $\binom{n_{\mathcal{S}}}{i}$ different sets of starting slots. This is easily implemented by only considering starting slots whose slot numbers are larger or equal than the slot numbers of the starting slots of any already allocated paths belonging to the same flow. A similar optimization is possible for the first path of the first flow. Given an arbitrary solution, another solution can be obtained by cyclically shifting the slot numbers. The metrics that we consider are invariant under this transformation, i.e. they yield the same metric value if the slot numbers are cyclically shifted. Thus, for the first hop of the first flow's first path p, only configurations that add a reservation for the first slot need to be considered. This is because all solutions that can be created for different starting slots of p also have a cyclically shifted version that assigns the first hop of p to the first slot.

Another efficiency enhancing measure concerns the noninterference checks. Implemented naïvely, each noninterference check triggers a computation that considers all the reservations

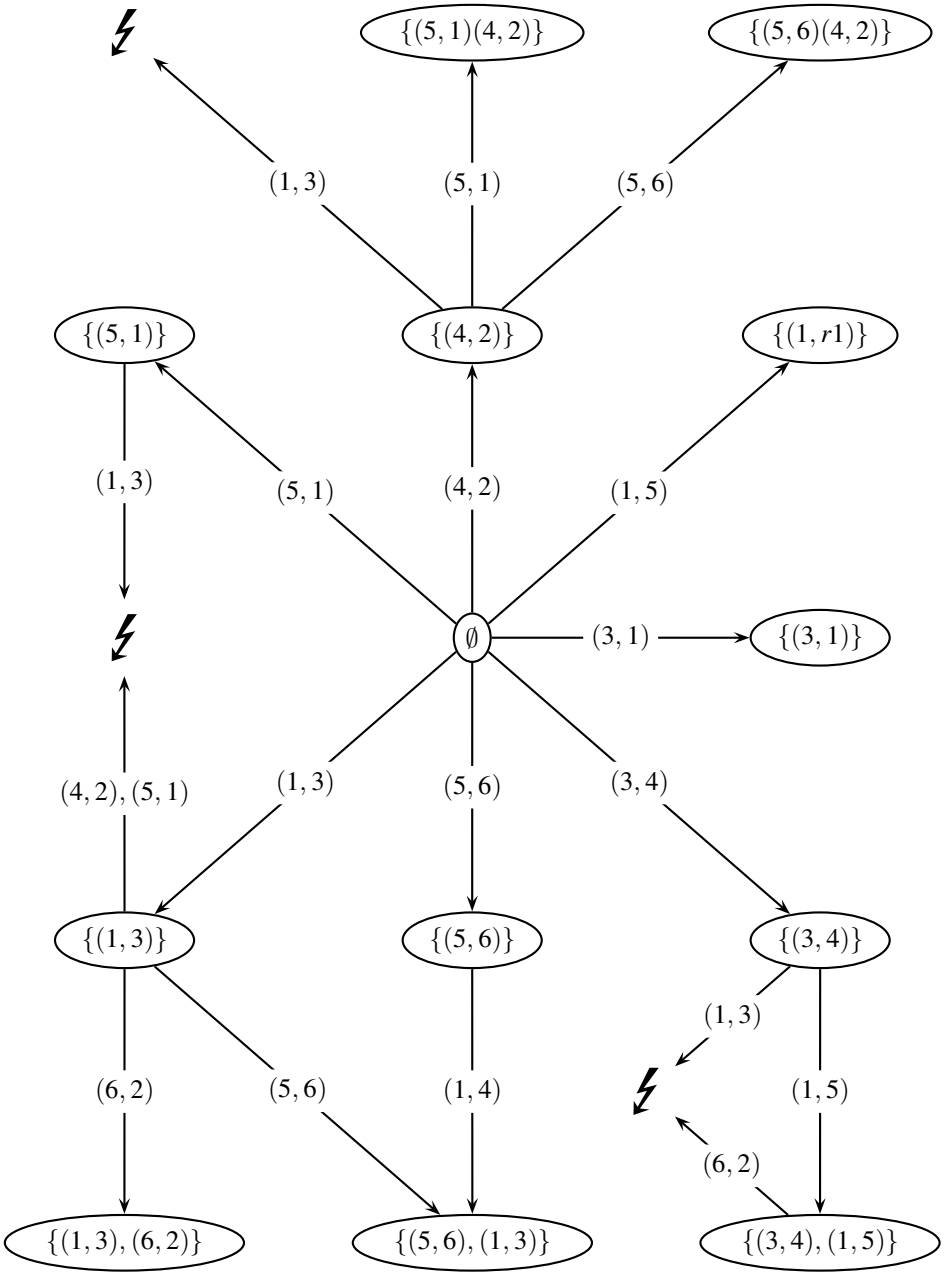


Figure 3.11: A noninterference checking graph for the hexagon scenario. Each node represents a set of connectivity graph edges that are successfully checked for noninterference. Every edge (p, q) in the noninterference checking graph is labeled with a connectivity graph edge e such that $p \cup \{e\} = q$, as long as the noninterference check for q succeeds. If the check fails this is represented by an edge pointing to the invalid edge activation set marker shown as $\lightning$.

that have already been made for a slot. Furthermore, the same set of active edges can arise during different slots, thus the same noninterference calculations will be carried out more than once. In order to speed up the noninterference checks, the already computed noninterference checks are cached such that for any situation that arises more than once the result of the check can be reused. To do so, a data structure is introduced that keeps information about the noninterference checks that have already been performed. This data structure is organized as a directed graph. Each node in the graph represents a set of edges that have been checked for noninterference. The edges in the graph are marked with some edge of the connectivity graph and represent additions to the sets of concurrently transmitting edges that are represented by the nodes. Thus, when considering an additional edge reservation, the branch and bound algorithm follows the corresponding edge in the noninterference checking graph to see whether it points to another noninterference checking graph node or to the invalid marker that is placed when the noninterference check fails for the resulting set of edges. Since the noninterference checking graph is constructed on the fly, it may also happen that an edge is missing. The edge set under consideration might already be present in another node of the noninterference checking graph, in which case the missing edge is inserted. Edges can also be missing if the edge activation set that the edge should point to has not been encountered yet. In this case the noninterference check is performed once and a corresponding node added to the noninterference checking graph. An example of the noninterference checking graph that results from running the branch and bound algorithm on the hexagon scenario is given in Figure 3.11.

The branch and bound solver implementation featuring all the techniques described above is able to process more than 10^6 configurations per second. Solving the examples given in Figures 3.1, 3.2 and 3.3 takes time in the order of seconds on current hardware. [3]

The solver interfaces with the environment by reading and writing XML-formatted data. Both the scenario descriptions that the program reads and the solution it produces is in XML format. This enables convenient manual editing, but also the creation of utility programs for scenario generation and result evaluation. A self-explanatory example of a scenario description and the solution produced by the solver can be found in Appendix B.

3.8 Parameterized random scenario generation

Experimental results about the performance of single-path and multi-path routing approaches, respectively, are presented later in this chapter. These results have been obtained empirically by calculating the optimal solution for a large number of families of scenarios. This section details the process of creating these scenarios. The scenario generation process follows a two-stage approach. First, nodes are placed and the connectivity graph is computed. In a second step, the scenario is created by selecting nodes from the graph to act as source and sink nodes for the flows.

For the graph generation step, the idea is to place a fixed number of nodes at random using a uniform distribution in a plane or within a box in 3D space. A transmission range parameter specifies the maximum distance within which nodes can communicate with each other. Each pair of nodes that is within this distance will be assigned a bidirectional edge

[3] Tests were performed on a machine equipped with an Intel Core Duo 6420 processor running at 2.13 GHz and 2 gigabytes of system memory.

in the connectivity graph. For the node placement, some constraints can be enforced by the graph generation algorithm. These are a minimum acceptable node distance value as well as a bound on the maximum node degree. The algorithm works by picking a random node position and checking whether the constraints hold. If this is not the case, the position is discarded and a new random position is generated. Another constraint that is usually imposed on the graphs that describe networks is for them to be strongly connected, i.e. there should be a directed path between each pair of nodes in the network. The generation process ensures this by generating a whole graph and checking whether it is strongly connected. If not, the graph is discarded and the generation process starts from the beginning. This rationale behind this approach is to uniformly pick one of the strongly connected random graphs from all graphs meeting the parameters. Adding additional constraints to the node placement that guarantee connectivity are likely to interfere with the uniform random selection of a graph within the subclass of all connected graphs and are thus avoided. A visual impression of a number of graphs that have been generated by the described process is given in Figure 3.12.

After the connectivity graph has been generated, data flows are added. Source and sink nodes are chosen randomly according to a uniform distribution. The capacity requirement of the generated flows is not chosen randomly, but fixed to a specified value. The choice of the sink node can be constrained w.r.t. the distance between source and sink node. This works by giving lower and upper bounds for the length of the shortest path between the nodes and only accepting chosen sink nodes if they respect these bounds. Another restriction for choosing nodes that has been applied in many of the experiments is the restriction for a node to be either source or sink in only one of the flows. A node that is acting as source or sink for more than one flow represents a hotspot in the system and is likely to represent a bottleneck.

3.9 Empirical performance evaluation

This section presents results from an empirical performance comparison of the single-path and multi-path approaches in randomly generated scenarios. For the generation of the scenarios, the technique described above has been employed. Solutions were obtained by running the branch and bound solver implementation on these scenarios twice to find an optimal single-path solutions and an optimal multi-path solution, respectively. In order to acquire statistically significant results, a large number of scenarios were considered. First, a number of connectivity graphs were generated subject to a fixed set of parameters such as number of nodes, minimum node distance, transmission range etc. (cf. Section 3.8). Then, for each of these graphs, a number of scenarios was chosen randomly by defining flows subject to parameters such as minimum hop distance etc. The exact parameter values and number of connectivity graphs and scenarios that make up an experiment are given in the scenario descriptions below.

Due to the complexity of the slot allocation problem, there are some caveats concerning the results: There is a practical bound on the number of flows and paths that the solver can handle. Whether the scenario is tractable heavily depends on the number of allocations that are required to construct a solution. This is due to combinatorial explosion, i.e. there typically is a considerable number of choices for each allocation that makes up the solution. For the experiments, this means that only scenarios that require less than roughly 15 allocations for the solution are tractable. Thus, only scenarios that are comprised of a small number of flows

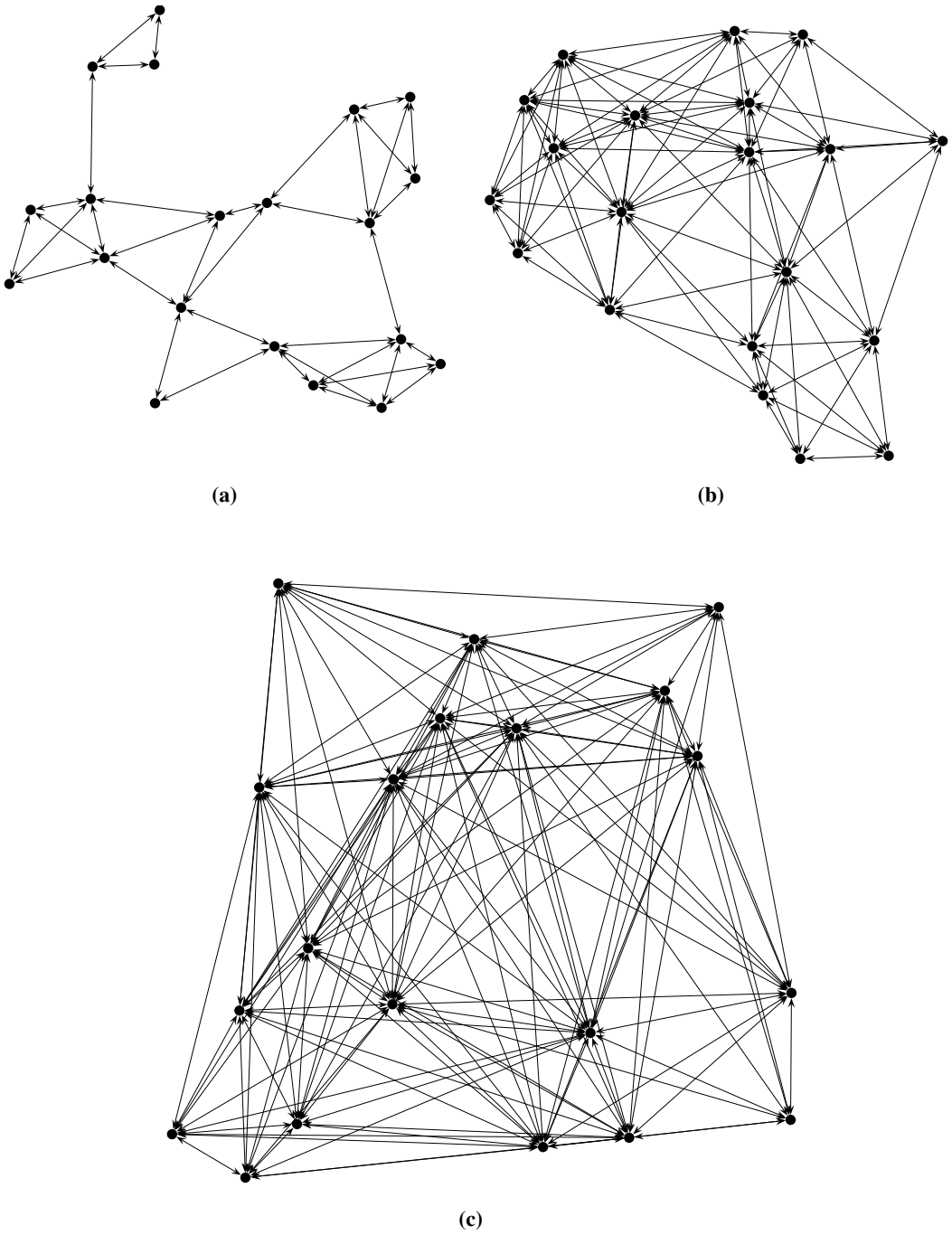


Figure 3.12: Three randomly generated graphs. 20 nodes have been placed randomly in a unit square with the additional constraints of a minimum node distance of 0.1 and the resulting graph being connected. The transmission range has been varied, it is 0.3 in Figure (a), 0.5 in Figure (b) and 0.7 in Figure (c).

Parameter	range 0.3	range 0.4	range 0.5	range 0.6	range 0.7
Number of nodes	20	20	20	20	20
Minimum number of edges	50	96	130	198	226
Mean number of edges	76.44	123.44	176.32	230.2	277.28
Maximum number of edges	100	154	218	276	318
Minimum node degree	1	1	2	3	5
Mean node degree	3.82	6.17	8.82	11.51	13.86
Maximum node degree	9	14	17	19	19
Minimum network diameter	5	3	3	2	2
Mean network diameter	6.78	4.3	3.14	2.76	2.1
Maximum network diameter	12	7	4	3	3

Figure 3.13: Properties of the connectivity graphs generated for the first experiment.

and paths can be examined. Nevertheless the results presented below give valuable insights.

Another issue is that there is a certain fraction of scenarios that the solver is unable to find solutions for. This can be due to the scenario being unsolvable, a situation that the solver can theoretically identify. However, to do so typically requires to also consider a large number of configurations with bad metric bounds, thus imposing excessive time and memory requirements. The same is true for scenarios for which there are solutions, but only with bad metric values. The solver only finds these after proving there are no solutions that have a better metric value. For the experiments below, these cases have been dealt with by introducing a metric value threshold and ignoring any configurations with a lower metric bound value larger than this threshold. The solver then either finds a solution with a metric value below the threshold or comes to the conclusion that there is no solution that would meet the threshold. This means that either there is a solution with a larger metric value than the threshold or the scenario is unsolvable. These situations are indicated by “larger value or unsolvable” in the results below.

Despite the measures described above, there are still scenarios arising in the experiments that the solver fails to handle within reasonable time or memory limits. In order to cope with these scenarios, the experiments were run in an environment that restricted time and memory consumption to values chosen in advance. Scenarios that could not be handled within the limits are marked below as “out of time” and “out of memory”, respectively.

3.9.1 Experiment 1: Unloaded networks

The first experiment examines two flows with capacity requirement of 3 frames per macro slot each. For the multi-path approach this means that the flows can possibly be split up to be handled by 3 different paths. The connectivity graphs that have been used for the experiment are made up of 20 nodes that have been randomly placed in a unit square with a minimum distance of 0.1 between the nodes. Different transmission range values of 0.3, 0.4, 0.5, 0.6 and 0.7 have been considered in different runs of the experiment. Some example graphs from different runs are shown in Figure 3.12. Summary information about the graphs is given in Figure 3.13.

A total of 50 graphs have been generated for each run of the experiment. On each of the graphs, 50 scenarios were generated by randomly selecting nodes for the two flows with the

additional constraints of a hop distance of 2 between the corresponding source and sink nodes and no node being source and sink in different flows at the same time. This resulted in 2500 scenarios, each of which was run through the solver to find both optimal single-path and multi-path solutions under the capacity usage metric. For the noninterference calculations, the graph-based model was used and the number of micro slots available was fixed to 10. The TDMA timing parameters and node transmission rates were chosen such that each node could transfer exactly one frame per macro slot.

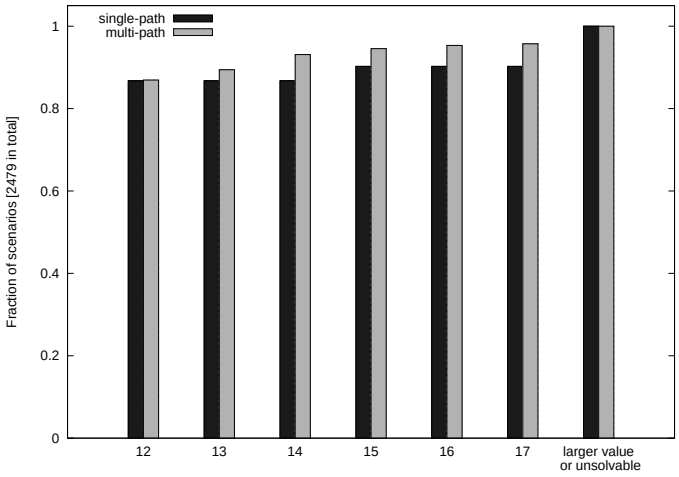
The results obtained are summarized in Figure 3.14. Note that for the single-path solutions there can only be metric values in increments of 3. This is due to the definition of the capacity usage metric: A single path solution that uses a path p of length n needs to push all 3 frames of the flow that must be transferred per macro slot through p, thus requiring $3n$ transmissions. Recall that by definition the capacity usage metric is the number of reservations a solution makes. In contrast, the multi-path approach allows to use paths of different lengths for each of the 3 frames to be handled per macro slot.

The fraction of scenarios that the solver could not find a solution for because it hit the limit of 600 seconds computation time or the 2 gigabyte memory restriction is less than 5% for all the runs of the experiment, for many it is well below. Hence, the error due to unsolved scenarios is quite small and can be ignored.

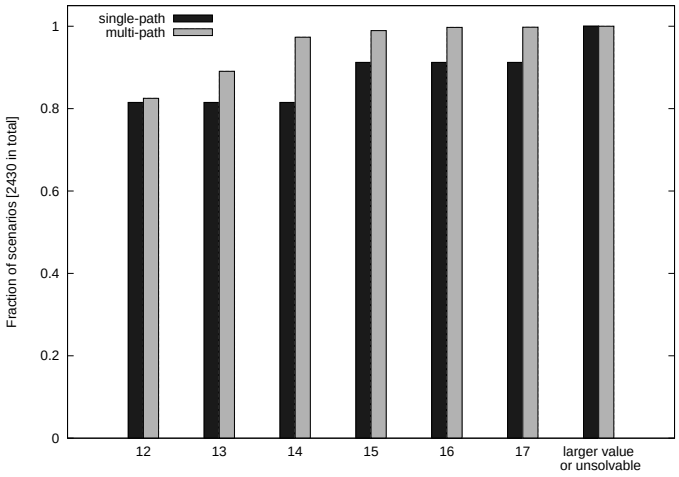
A quite large number of scenarios can be solved with metric value 12, i.e. there is a solution that only makes 12 reservations, both for the single-path and the multi-path approach. Observe that every solution will have to make at least 12 reservations, since it has to push a total of 6 frames per macro slot along paths of a length of at least 2. The large fraction of around 80% of scenarios that are solvable making only 12 allocations is expected, since the only reason for using paths that are longer than 2 is all shorter paths cannot be used, which is only the case if the allocations for the flows interfere with each other. However, the 10 available micro slots allow 10 noninterfering transmissions per macro slot, thus the scenario can be solved optimally if the setup allows for concurrent transmissions e.g. of two pairs of frames in two of the micro slots.

Range	Variant	12	13	14	15	16	17	L/U	T	M
0.3	single	2150	-	-	87	-	-	263	0	0
0.3	multi	2155	62	91	36	19	10	106	15	6
0.4	single	1980	-	-	236	-	-	270	13	1
0.4	multi	2005	160	211	40	19	1	6	20	38
0.5	single	1897	-	-	391	-	-	159	46	7
0.5	multi	1979	304	163	23	4	0	0	0	27
0.6	single	1918	-	-	392	-	-	79	85	26
0.6	multi	2070	319	85	3	2	0	0	0	21
0.7	single	2141	-	-	198	-	-	35	111	15
0.7	multi	2276	144	45	0	0	0	0	10	25

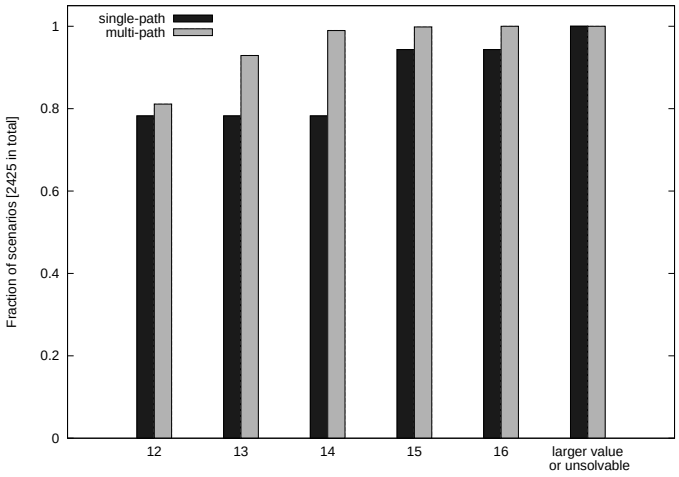
Figure 3.14: Resulting capacity-usage metric values for an experiment involving two flows on an otherwise idle network. Each line gives the number of scenarios that resulted in the metric value given in the column header. “L/U”, “T” and “M” indicate situations with larger metric value or unsolvable scenario, or the solver running out of time or memory, respectively.



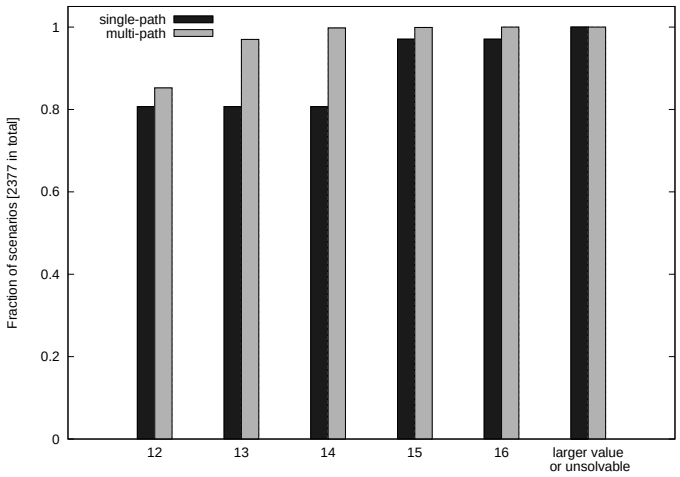
(a) Transmission range 0.3



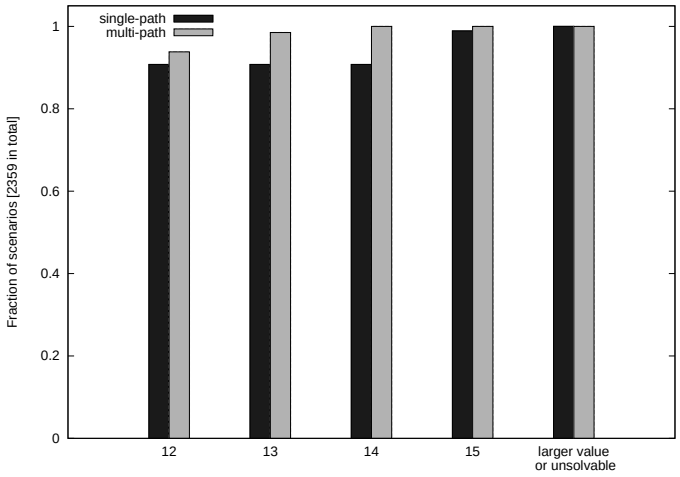
(b) Transmission range 0.4



(c) Transmission range 0.5



(d) Transmission range 0.6



(e) Transmission range 0.7

Figure 3.15: Comparison of the fraction of scenarios that can be optimally solved with at most the given metric value by the single-path and multi-path solutions. Each diagrams shows the results for a different transmission range value.

Figure 3.15 illustrates the fraction of scenarios that can be solved with at most a given metric value both for single-path and multi-path solutions. The data used for generating the diagrams excludes all scenarios that resulted in the solver aborting due to time or memory constraints when looking for a single-path or multi-path solution. The total number of scenarios that have been considered when calculating the fractions is noted in the vertical axis label. Recall that the single-path solutions can only have metric values in steps of 3, which explains constant fractions for metric values 12–14 and 15–17, respectively.

An interesting feature of the diagrams is that the multi-path approach does not yield significantly more solutions with metric value 12 than the single-path approach. However, a fair number of scenarios can be solved by only using longer paths for one or two of the 6 frames

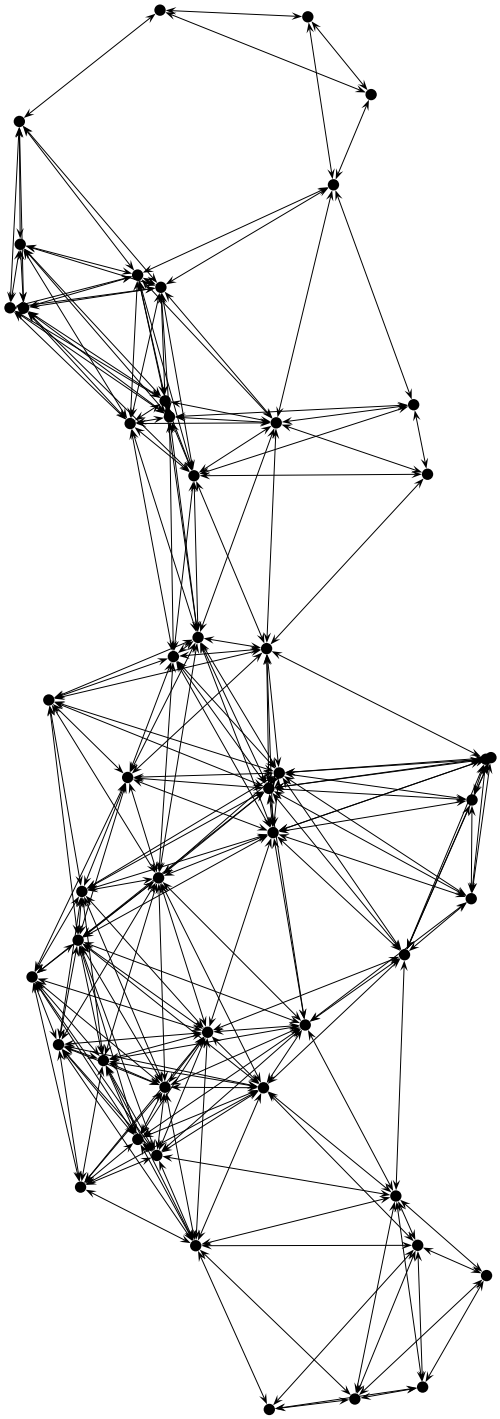


Figure 3.16: One of the connectivity graphs used for the second experiment. 50 nodes have been distributed randomly in a rectangular area of size 1.5 by 0.5, enforcing a minimum node distance of 0.0025. The transmission range is 0.25.

Parameter	3 hops	4 hops	5 hops	6 hops
Number of nodes	50	50	50	50
Minimum number of edges	368	398	400	384
Mean number of edges	465.32	471.24	467.12	470.6
Maximum number of edges	516	544	564	546
Minimum node degree	1	1	1	1
Mean node degree	8.8	8.88	9.34	8.96
Maximum node degree	20	20	21	20
Minimum network diameter	7	7	7	7
Mean network diameter	8.12	8.2	8.44	8.28
Maximum network diameter	10	9	12	10

Figure 3.17: Statistics characterizing the connectivity graphs used for the second experiment.

that must be handled per macro slot. These roughly correspond to metric values of 13 and 14 in the multi-path approach and the fraction of scenarios that can be solved with at most metric value 14 approaches 100% for the larger transmission ranges. The relatively slow increase of the fraction of scenarios solvable in the multi-path approach that is evident in Figure 3.15a is probably due to the small node degree which results in less choices for the paths that can be used to handle a flow.

3.9.2 Experiment 2: Background traffic

This experiment examines the optimal latency values achievable by the single-path and multi-path approaches, respectively, in a network that has been put under load by preallocating a number of background flows. The connectivity graphs generated for the experiment are connected graphs made up of 50 nodes in a rectangular area of size 1.5 by 0.5. The node transmission range has been fixed to 0.25 while a node distance minimum of 0.0025 was enforced. 50 random connectivity graphs have been generated; an example graph is shown in Figure 3.16. A summary of the parameters characterizing the sets of graphs that have been used for the 4 runs of the experiment is given in Figure 3.17.

The background traffic is divided into 4 classes of 4 flows each. Each flow only requires a capacity of one frame per macro slot. The distance in the connectivity graph between source and sink nodes of a flow is 2 hops for the first two classes and 3 hops for classes 3 and 4. Within a class, nodes are used only once, i.e. a graph node can at most participate as source or sink node in 4 of the total of 16 background flows. Given a total number of 20 slots, capacity was allocated incrementally for the background flows: A capacity-usage optimal solution was generated for each flow in turn, thereby regarding the allocations for previous flows as fixed and adding allocations only for the flow currently under consideration.

This resulted in 50 graphs with some capacity blocked by preexisting allocations as described above. For each of these graphs, 50 scenarios subject to the following parameters were considered: Each scenario consists of a single flow that requires two frames per macro slot of capacity, thus the flow could be handled using two different paths under the multi-path approach. Hop distances between the source and sink nodes of 3, 4, 5 and 6 have been considered in different runs of the experiment. In each run of the experiment, 2500 scenarios were

Distance	Variant	3	4	5	6	7	8	9	L/U	T	M
3	single	1872	118	73	47	39	35	37	276	3	0
3	multi	1986	144	57	29	39	23	16	158	48	0
4	single	-	1649	116	86	71	60	60	422	36	0
4	multi	-	1735	142	110	54	27	17	258	157	0
5	single	-	-	1079	203	126	179	117	759	37	0
5	multi	-	-	1200	257	153	127	62	406	295	0
6	single	-	-	-	832	255	135	105	1105	68	0
6	multi	-	-	-	886	308	78	38	529	661	0

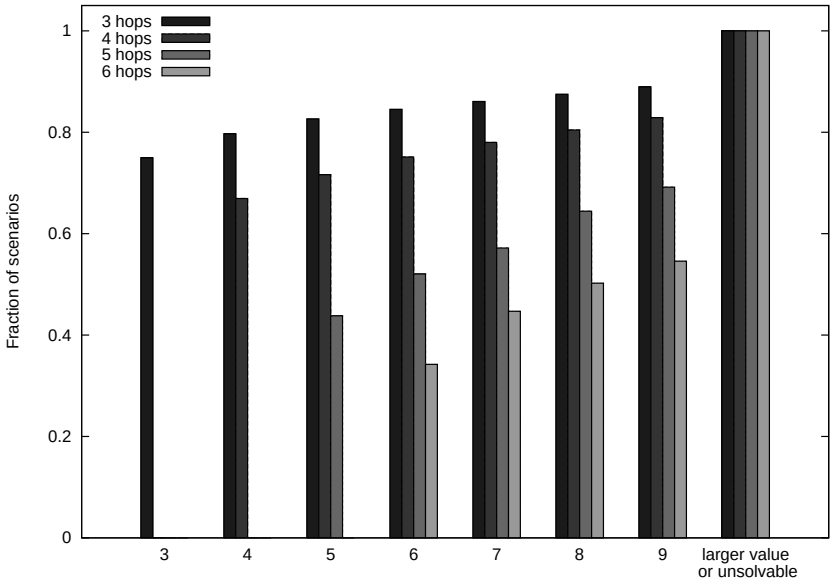
Figure 3.18: A flow made up of two paths was routed through a network that also had to handle some background traffic. The hop distance between the source and the sink nodes of the flow under observation was varied from 3 to 6. The table shows the number of scenarios that could be optimally solved with the indicated latency value. The last 3 columns refer to scenarios that have larger metric values or are unsolvable (L/U), or could not be handled by the solver due to time (T) or memory (M) restrictions.

examined in total and the solver was requested to calculate optimal solutions w.r.t. the maximum latency metric for both the single-path and multi-path approaches. All noninterference calculations were performed under the graph-based noninterference model.

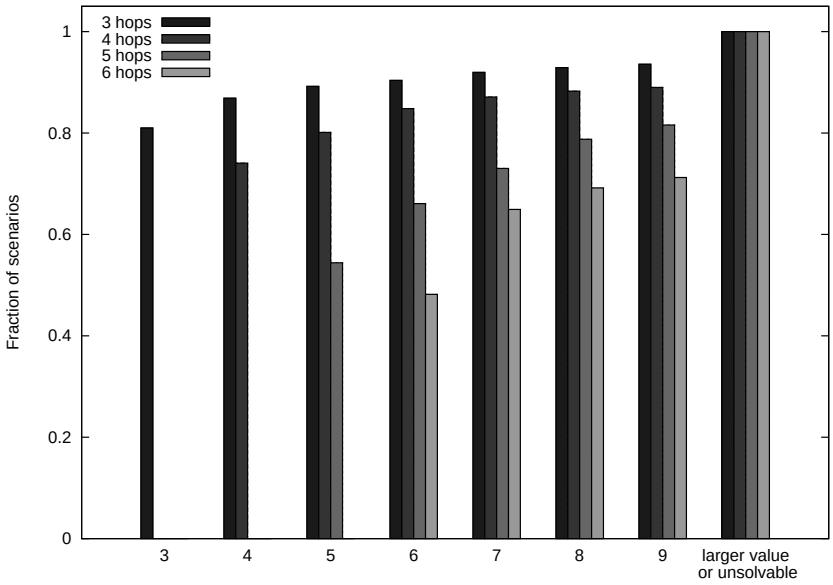
The results of the experiment are shown in Figure 3.18. It gives the number of scenarios whose optimal solutions achieved the given latency value in the single-path and multi-path approaches, respectively. The number of scenarios for which the solver could not compute a solution within the given time bound of 300 seconds or the memory bound of 2 gigabytes of system memory make up only up to 2.7% for the single-path approach and can thus be neglected. However, in the multi-path approach, in the 6-hop run of the experiment there are 661 scenarios unsolved due to time restrictions which represents more than 25% of the total of 2500 scenarios. This will inevitably influence the results. However, it is expected that mostly scenarios that yield higher latency values are affected, because those typically take more time due to the larger number of configurations that must be considered. Thus, the number of scenarios for latency values 6 and 7 are anticipated to be accurate.

Figure 3.19 shows a graphical display of the fraction of scenarios that have optimal solutions of a given latency value or better. Note that scenarios that could not be solved due to exceeded time or memory limits were removed from the data set before generating the diagram. As expected, the fraction of scenarios that can be solved increases with the latency value. Note that especially in the single-path case, e.g. the fraction of scenarios solved in the 3-hop run of the experiment seems to hit a bound at 0.9 as the latency value increases, suggesting that 10% of the scenarios considered are indeed unsolvable. As the hop distance increases, the fraction of unsolvable scenarios is expected to increase, which is one reason for the other runs of the experiment to show overall worse ratios of solved scenarios. Comparing the two diagrams of Figure 3.19 indicates that the multi-path approach performs better, i.e. with multi-path routing a larger fraction of scenarios have solutions that meet a given latency value than in the single-path approach.

The diagrams in Figure 3.20 further confirm this observation. Again, before generating these plots, the data has been preprocessed to remove all the scenarios from the statistics for which the solver failed to compute a solution due to the time or memory restrictions

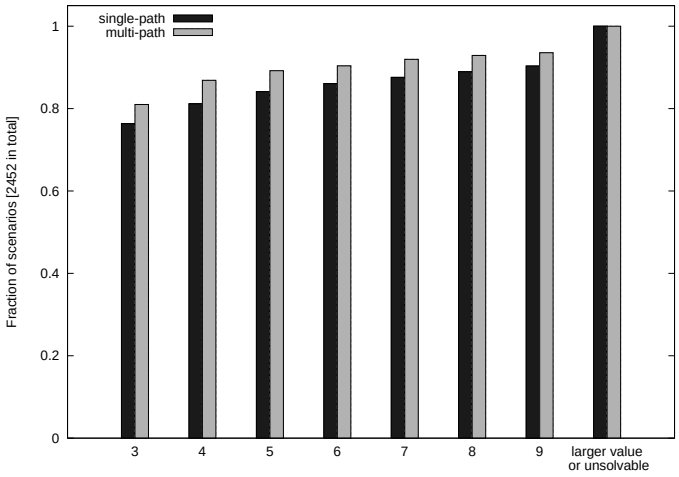


(a) Single-path solutions

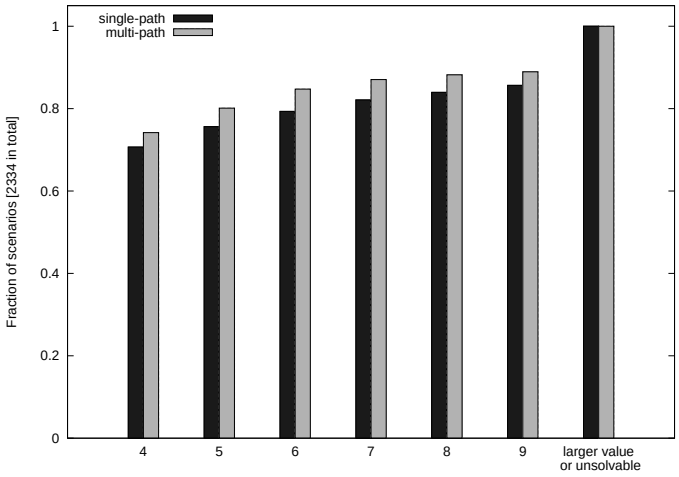


(b) Multi-path solutions

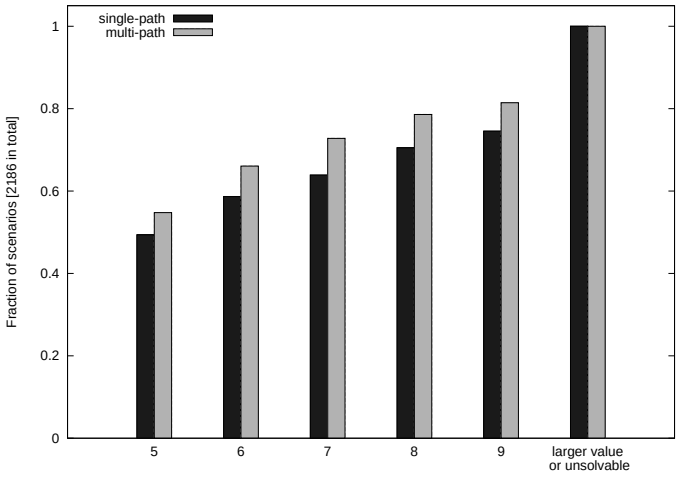
Figure 3.19: Latency values versus the fraction of scenarios that have optimal single-path (a) or multi-path (b) solutions of at most the given latency.



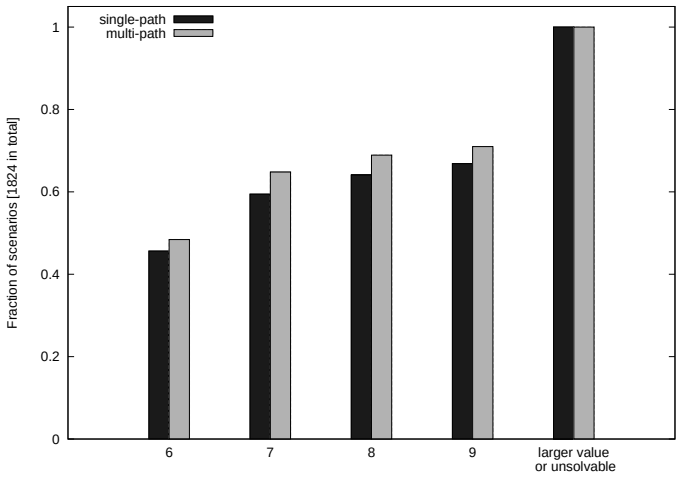
(a) Hop distance 3



(b) Hop distance 4



(c) Hop distance 5



(d) Hop distance 6

Figure 3.20: The fraction of scenarios that can be solved with a latency value at least as good as a threshold in the single-path and multi-path approaches, respectively. As the hop distance value increases, the fraction of scenarios that have solutions with the optimal latency value decreases.

for the single-path or the multi-path approach. Depending on the hop distance and latency value, the multi-path approach has up to 9% more optimal solutions with a given latency value or better than the single-path approach. Note that for the 3-hop run of the experiment, the margin of the multi-path approach stays approximately the same as the latency value increases, while for the runs with larger hop distances, the advantage even increases with larger latency values. This effect is presumably caused by the fact that the increase in the number of usable paths is expected to be larger in the multi-path approach as the latency value becomes larger, because the multi-path approach can use any path that allows to handle one frame per macro slot with a good metric value, but the single-path approach needs two frames per macro slot. An interesting follow-up question is whether the multi-path advantage significantly increases when the capacity requirement allows the flow to be split up into more than two paths. However, due to the complexity of the slot allocation problem, this experiment is intractable with our solver.

3.9.3 Experiment 3: Physical noninterference model

While the physical noninterference model more closely resembles actual wireless networks than the graph-based one, the latter is more convenient when reasoning on an abstract level. The main reason for this is that the graph-based noninterference model definition is independent from actual node locations, i.e. reduces the network to its topological features. As discussed in Chapter 2, not all possible topologies correspond to actual networks, so care must be taken when considering arbitrary topologies. Nevertheless, the abstraction has proved usable in the introductory sections of this chapter, so it is important to assess the influence on the results due to the use of this simplified model.

In order to do so, a third series of experiments has been run. The parameters are based upon those used for the second experiment. The network creation process is as follows: 50

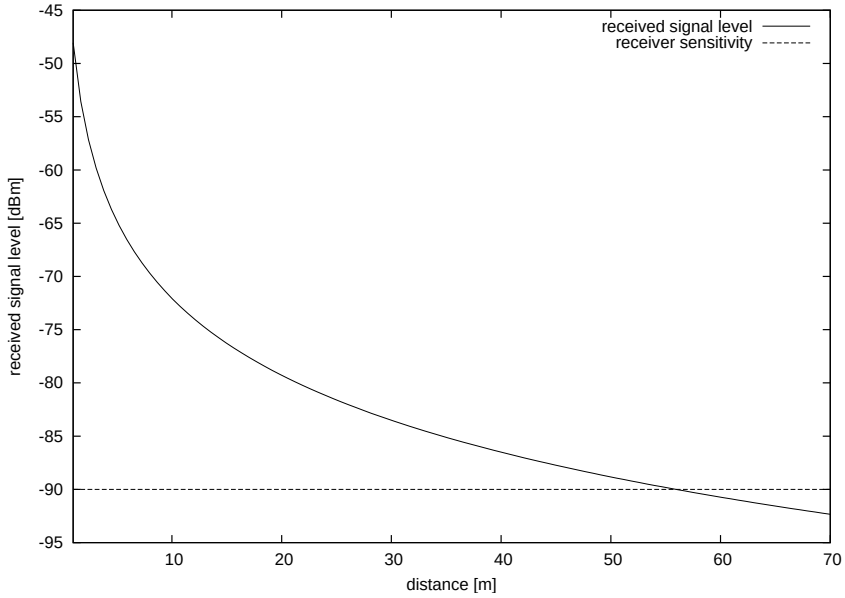


Figure 3.21: Received signal level characteristic used for the physical noninterference model in the experiment.

nodes are randomly placed in a rectangular area, the dimensions have been scaled by a factor of 220 m to 330 m by 110 m. The minimum node distance has been set accordingly to 0.55 m and the transmission range to 55 m. The TDMA parameters are taken unchanged from the second experiment and allow each node to transmit exactly one frame per micro slot, of which there are 20 available in a macro slot.

The parameter values for the physical noninterference model have been chosen to roughly correspond to the characteristics of the Texas Instruments CC2420 [21] RF transceiver based on information from the data sheet and measurements [43, 3]. The Friis equation is a simple model for radio propagation that relates the received power P_r at distance d to the emitted power P_t:

$$P_r = P_t \cdot G_r \cdot G_t \cdot \left(\frac{\lambda}{4 \cdot \pi}\right)^{\alpha} \tag{3.61}$$

The parameters G_r and G_t and λ describe the sender and receiver antenna gains and the wavelength, respectively. The transmission power of 1 mW the transceiver feeds to the antenna, antenna gain coefficients of 1 and an attenuation exponent of $\alpha = 2.4$ yield the characteristic depicted in Figure 3.21, which agrees with the measurements [43, 3]. Corresponding values for the TXP and α parameters of the physical noninterference model that yield this characteristic have been used for the experiment. For the ambient noise level N, a value of 10^{-10} mW has been chosen, corresponding to -100 dBm. The data sheet specifies a receive sensitivity of -90 dBm, yielding a transmission range of approximately 56 m with the propagation characteristic used. The SINR threshold value was fixed to $\gamma = 6$ dB. The behavior of the simulation has been found to be very sensitive to the SINR threshold parameter. Larger values quickly reduce the number of concurrent transmissions that are possible, even if the distance between the transmitting nodes is large.

Similar to the situation in the second experiment, the network has been put under load but

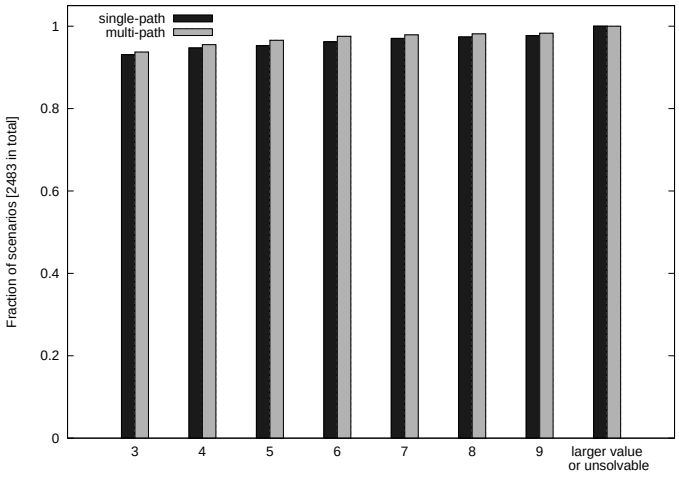
Distance	Variant	3	4	5	6	7	8	9	L/U	T	M
3	single	2311	41	14	23	20	14	11	64	2	0
3	multi	2327	45	26	24	9	6	4	42	17	0
4	single	-	2093	124	52	25	31	53	97	25	0
4	multi	-	2116	168	46	15	10	10	26	106	3
5	single	-	-	1572	184	98	109	89	407	41	0
5	multi	-	-	1619	247	96	53	22	32	422	9
6	single	-	-	-	1395	231	179	138	518	39	0
6	multi	-	-	-	1474	297	93	33	155	439	9

Figure 3.22: Results of the third experiment that considered 2500 randomly generated scenarios with a flow connecting two nodes of varying distance. The number of scenarios for which an optimal solution of the given value w.r.t. latency was found is shown. The columns "L/U", "T" and "M" hold the numbers of scenarios that only have solutions with higher latency values or are unsolvable, could not be solved due to the time limit or were aborted due to exceeding the memory limit, respectively.

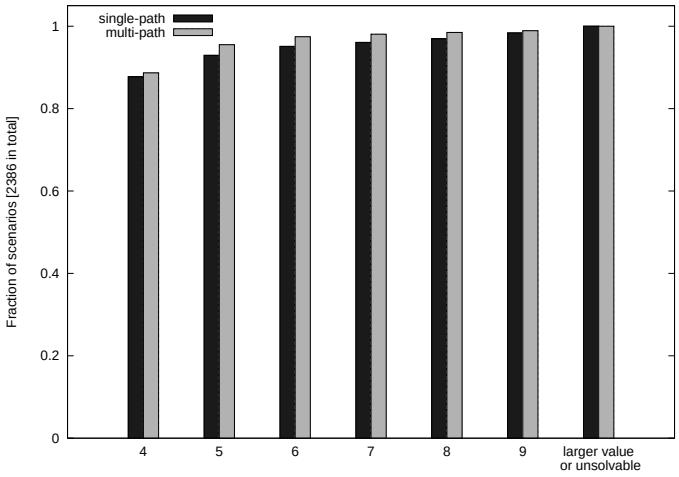
preallocating capacity for some background flows. However, the number of background flows was cut down to a total of 8 instead of 16 that were used in the second experiment. Otherwise, there was not enough network capacity to handle even only the background flows. This is because in the physical noninterference model, a transmission not only prevents communication in the direct connectivity graph neighborhood of the transmitting node, but may also hinder communication in a larger region due to the addition of energy levels and the slowly decaying received signal level characteristic. The load was due to two classes of 4 background flows each and a capacity requirement of one frame per macro slot each. The hop distance between source and sink nodes was 2 hops for the first class and 3 hops for the second. The source and sink nodes were chosen such that no node would be source or sink node more than once within the flows belonging to a background traffic class.

The parameters of the flow used as subject of the experiment were not changed from the second experiment: The capacity requirement was 2 frames per macro slot and hop distance values of 3, 4, 5 and 6 between source and sink node were examined in different runs of the experiment. For each run, 50 graphs were randomly generated subject to the parameters discussed above. Background traffic was added to the graphs and a total of 50 scenarios were generated based on each graph by randomly setting up the flow under consideration. The resulting 2500 scenarios were then run through the branch and bound solver, once requesting a single-path solution and once also allowing multi-path solutions. The time limit for a single solver run was fixed to 300 seconds and a memory limit of 2 gigabytes was enforced.

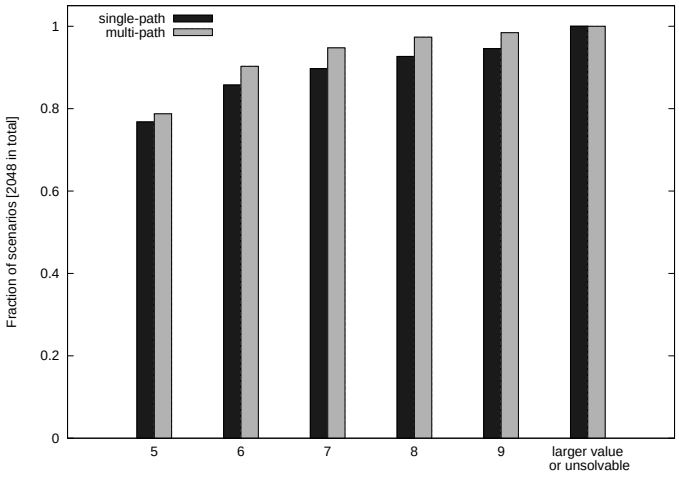
The results of the different runs of the experiment are shown in Figure 3.22. Note that the number of experiments that were aborted due to the time or memory limits are approximately below 20% even for the 5- and 6-hop runs when allowing multi-path solutions. Thus, the bias due to unsolvable scenarios is in the same order as for the second experiment. An overall increase in the fraction of solvable scenarios when compared to the second experiment is clearly evident. This is however not relevant due to the lighter background traffic. Other than that, the numbers show a feature that has already been identified in the second experiment. With the hop distance increasing, the number of scenarios that can be solved with the optimal latency value decreases. This is expected, since it is harder to find paths that can provide



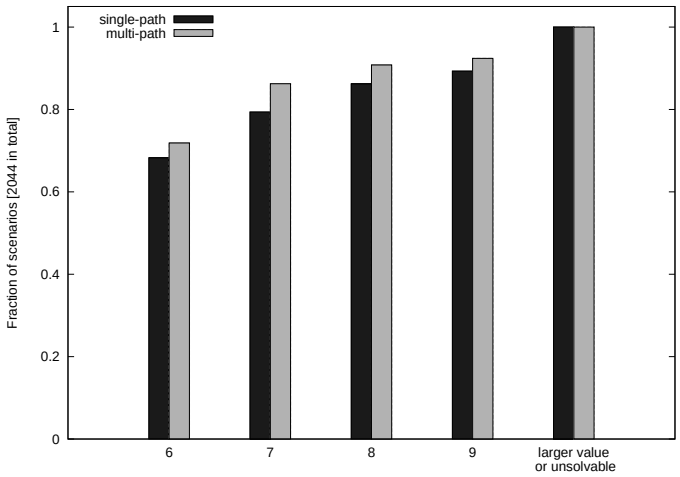
(a) Hop distance 3



(b) Hop distance 4



(c) Hop distance 5



(d) Hop distance 6

Figure 3.23: Fractions of the total number of solved scenarios for which solutions with at least a given latency value in the single-path and multi-path approaches, respectively, were found.

consecutive micro slots for the hops as the paths grow longer, considering that some of the slots are not usable due to the background traffic.

The direct comparison between fractions of the total numbers of scenarios solvable with at most a given latency value in the single-path and multi-path approaches is presented in Figure 3.23. Similar to Figure 3.20 there is a multi-path gain evident in the diagrams. However, the margin of scenarios solvable with at most a given latency value in the multi-path approach but not in the single-path approach is only about 7% at maximum. This is less than what was observed in the second experiment. We presume this is due to the fact that transmissions can also interfere with other transmissions in a larger region than the direct connectivity graph neighborhood that is considered under the graph-based noninterference model. Comparing Figures 3.20a and 3.23a yields a similar result. The multi-path margin is less than half as large under the physical noninterference model than it was with graph-based one. However, the results of the experiments show a surprisingly close match from a qualitative point of view, thus justifying the use of the graph-based noninterference model in qualitative analysis.

4 Literature review

This chapter reviews some relevant literature and points out which aspects of the reviewed material are relevant for wireless multi-hop TDMA networks. There are several research areas in which previous work relevant for multi-hop time-slotted networks is found. These are time synchronization techniques which are required to synchronize a node's transmissions to the TDMA slot structure, ad hoc routing research that addresses the problem of discovering routes between source and sink nodes, and TDMA scheduling, which considers micro slot allocation under the constraint of avoiding interference.

4.1 Time synchronization

Time synchronization, i.e. a mechanism that provides all nodes in a network with a common understanding of time, is an important aspect in wireless networks. Traditional time synchronization approaches [35] typically adjust the local node's system clock in order to sync it with a global reference time. Alternatively, one can collect the necessary information to be able to convert time values from adjacent nodes to the timescale as defined by the local clock [7].

Time synchronization mechanisms typically work by comparing local clocks against a well defined reference point in time. For example, this can can be a node clock that is used as reference. The current time is read from this reference clock and communicated to other nodes that wish to synchronize their clocks against the reference time. The problem with this approach is the aging of the reference time value: Receivers can only compare their clocks against the reference time value when they receive it, but in the meantime the reference clock will have advanced. These communication delay effects can be mitigated by measuring the delay and compensating for it; this is what NTP [35] does. However, since transmission times cannot be measured directly, the round trip time is used for estimating single way delay, which can be inaccurate. Another approach is to try and reduce the total communication delay, so that it can be ignored [15, 7].

In our network model, we have assumed accurate time synchronization of all nodes in the network. This is needed for the nodes to agree on the starting time of each TDMA macro slot and the micro slots it is comprised of. In practice, perfect time synchronization cannot be achieved, thus there is some uncertainty w.r.t. when the individual micro slots begin and end. In order to avoid collisions between transmissions in adjacent micro slots, the slots are spaced by short guard time intervals of length twice the maximum synchronization error.

The following sections review some time synchronization protocols that have suitable characteristics for use in WSNs. There are many others, a good overview of clock synchronization protocols for WSNs can be found in [44].

"Reference Broadcast Synchronization" (RBS) proposed by Elson et al. [7] exploits the broadcast nature of wireless channels for time synchronization. Instead of using a node clock to obtain reference time, the authors make use of the fact that a broadcast message sent by

an arbitrary node is received by all listening nodes approximately at the same time. Thus, this point in time can be used as a reference to synchronize the clocks of all nodes that received the broadcast message. In the RBS implementation described by the authors, the nodes periodically exchange messages containing the arrival times at which recent reference broadcasts arrived in the timescale defined by the sending node's clock. This information can be used by the other nodes to derive the clock offset and its drift rate such that clock values can be converted back and forth between the nodes' clocks.

Compared to the traditional sender–receiver synchronization approach used e.g. in NTP, the receiver–receiver synchronization used by RBS has some advantages. This stems from the fact that many of the currently used medium access control protocols (e.g. CSMA/CA as used in IEEE 802.11) can introduce a non-deterministic delay when waiting for a clear channel. This imposes a problem for sender-receiver synchronization protocols such as NTP that assume a constant delay for both the transmission from sender to receiver and the transmission in the opposite direction. In RBS, the medium access delay does not contribute to the communication delay because the reference point in time is only established when the broadcast message is actually sent.

Ganeriwal et al. propose a "Timing-sync Protocol for Sensor Networks" (TSPN) [13] that builds on the sender-receiver approach. Just like NTP, it synchronizes a sender with a receiver by transmitting a message from the sender to the receiver and an answer in the opposite direction. Then the communication delay and clock offset is estimated from the timestamps included in the messages.

The main difference to NTP is how the timestamps are generated. Traditional NTP implementations run as user programs and obtain the timestamps when their messages leave from or arrive at the application level. Due to varying operating system load and task scheduling, this introduces a lot of non-deterministic delay in the transmission path. However, exploiting the software-controlled design of typical low cost wireless network interfaces for sensor network nodes, time stamping of frames can be done at the MAC level. This not only eliminates the processing delay for message construction and its transfer to the MAC layer but also the medium access delay. Moreover, incoming messages are time stamped at the MAC layer directly after they have been received.

"Black Burst Synchronization" (BBS) as described by Gotzhein and Kuhn in [15] uses periodic frames sent at well defined reference points in time for synchronization. A special frame encoding based on so called *black bursts* is used for these synchronization tick frames, so they are protected against collisions when sent approximately at the same time. The idea is to use only the energy level for encoding. 0 bits are encoded as no transmission, 1 bits are encoded as transmission. By using the clear channel assessment (CCA) function of the receiver hardware, receivers can detect whether the medium is clear which is interpreted as a 0 bit. A detected signal is regarded as a 1 bit. Moreover, in case of multiple senders, 1 bits dominate 0 bits. Thus, receivers will hear the bitwise-or of all the transmissions sent by stations in transmission range.

In BBS, the period at which tick frames are sent is fixed. Nodes that receive a tick frame record the time it arrived and thus know when the next tick frame is expected to arrive. For recording tick frame arrival time, the CCA logic triggers an interrupt and the arrival time is recorded in the interrupt handler. This ensures that apart from CCA jitter effects the arrival time can be recorded very accurately. At the sender side, communication delay is minimized by creating the tick frames well in advance of the next tick, so they can be passed to the

network interface at the exact time the tick is scheduled. Furthermore, a central property of black bursts is that they are transmitted immediately without waiting for a clear channel, so there effectively is no medium access delay. These measures keep the global time difference between sending a tick and receiving it small (in the order of *ms* for experiments with MICAz WSN nodes described in [15]), so the authors disregard it. Thus the nodes use the arrival time of the last tick as reference point in time. Events in the interval bounded by ticks i and $i+1$ are then timed relatively to tick i in terms of the node's local clock. Note that clock drift is ignored, since it is expected to be negligible as long as the tick period is reasonably short.

For distributing tick frames in the network, the protocol described in the paper uses both a decentralized and a centralized scheme in parallel. Each interval bounded by ticks i and $i+1$ starts with a synchronization phase in which tick frames are exchanged. The synchronization phase consists of a predefined number n of rounds in each of which there is room for transmission of a single tick frame. The first part of the tick frame is a marker that starts the frame and is dedicated to the decentralized part of the protocol. It is always transmitted by every node participating in the synchronization protocol. Unsynchronized nodes that overhear n tick frames can therefore deduce the time tick i has occurred to be the time at which the first tick frame arrived. The optional second part of the tick frame is called master tick frame and is used for centralized synchronization. A master node transmits a master tick frame in the first round. All nodes that overhear the master tick frame will retransmit it in the next round. The round number is embedded in the master tick frame and will be increased by each node retransmitting it. Thus, since the round duration is fixed, the arrival time of a tick frame that carries round number k is enough to derive the time at which tick i occurred.

A virtue of BBS is that the authors derive a deterministic bound for the synchronization accuracy. The synchronization accuracy bound is linear in the network diameter and a bound on the CCA jitter when centralized synchronization is in effect. For decentralized synchronization, the transceiver switching time, i.e. the time needed to switch from receive mode to transmit mode adds to the centralized synchronization accuracy bound. Moreover, synchronizing all nodes in the network can be done during a single synchronization phase, thus yielding a low deterministic bound for convergence time. Experiments with BBS on MICAz WSN nodes and IEEE 802.11 hardware described in [15] achieved synchronization accuracy in the order of *ms* and μs, respectively.

The protocols discussed above all achieve tight time synchronization between the nodes in the network. However, their suitability for providing time synchronization in a TDMA-based system also depends on other aspects. RBS requires regular broadcast transmissions in order to synchronize nodes. It seems possible to use regular data frames for that purpose, but this does not help in network regions that do not handle any data transmissions for a longer time interval. Furthermore, RBS does not specify how to coordinate the nodes when exchanging information about their observations. While a competition-based approach would certainly work, this probably requires a significant portion of transmission time available in a macro slot. Another problem is that there is no guaranteed bound on the synchronziation error, such that longer guard intervals are required. An upper synchronization error bound is also missing in TSPN. However, the hierarchical structure used in TSPN that synchronizes all nodes starting at a root node integrates better with the TDMA slot structure. Synchronization for the micro slots can be achieved by resynchronizing the nodes at the beginning of each macro slot. Still, messages must be exchanged between each pair of synchronizing node, which will introduce significant overhead. In contrast, BBS only needs a number of rounds in the order

of the network diameter to synchronize the whole network. Again, this can be done at the beginning of each macro slot in order to provide accurate time synchronization for the following micro slots. BBS also provides a guaranteed upper bound on the synchronization error and a very short convergence time. Thus, it seems a very intruiging choice for synchronizing the nodes in a wireless TDMA network.

4.2 Routing in ad-hoc networks

Due to node mobility and the dynamic nature of the wireless medium, well established routing techniques that are used with wired networks are not suitable for wireless networks. In the wired world, two very popular approaches are link-state and distance-vector protocols, respectively. They are both based on the idea to disseminate information about the network topology to all nodes in the network so these can decide to which of their neighbors to forward frames to for a given destination node. In link-state routing protocols, each node keeps a complete view of the network connectivity graph and uses that information to compute a shortest path to any other node. The connectivity information is communicated by periodically flooding link state information local to a node to the whole network. Each node can then independently construct the connectivity graph and perform the shortest path calculations. In contrast, the distance-vector approach implements a distributed shortest path search algorithm that calculates the shortest paths between any two nodes in the network. Each node periodically broadcasts the shortest-path distance it knows to any other node in the network. Initially, this information will only contain distance values for local neighbors. After receiving distance vectors from its neighbors, a node selects the next-hop node for a destination as the neighbor that announces the minimum distance to the destination node. It can then announce this destination in its own distance vector, thereby accounting for the additional hop to the neighbor. Eventually, the routing information has diffused through the whole network and each node knows the shortest path length and the next-hop neighbor for each destination.

As mentioned above, the traditional routing techniques are not suitable for highly dynamic wireless networks. In case of frequent topology changes, the routing information must be updated very often, causing considerable load on the network. Furthermore, problems such as routing loops that can occur with some of the traditional approaches when the topology changes must be addressed. The fact that routing information for a particular destination node is only needed if there actually is data to be sent to that node means the overhead of maintaining unused route information can be avoided. Consequently, research has shifted from *proactive* routing techniques that always try to maintain current routing information for all possible destination nodes to *reactive* routing schemes that construct routes only on demand, i.e. when there is data to be transferred to a given destination node. Two prominent members of this class, Dynamic Source Routing [23] and Ad-Hoc On-demand Distance Vector Routing [38] are discussed below.

Note that most ad hoc routing research is not targeted at reservation-based wireless networks which are the focus of this thesis. Thus, many authors assume a competition-based medium access mechanism in the evaluation of their routing protocols. In this setting, metrics such as latency and throughput behave very differently. Due to reduced queuing, latency decreases when the load on a node is decreased. This is in sharp contrast to reservation-based systems for which latency can be considered as constant once capacity for a flow is reserved.

Nevertheless, reservation-based networks also need some form of route discovery. Therefore, it is worthwhile to study general ad hoc routing protocols.

4.2.1 Destination-Sequenced Distance-Vector Routing (DSDV)

In [37], Perkins and Bhagwat describe a distance vector based routing protocol that has been specifically adapted for the use in wireless ad hoc networks. The most significant modification is the addition of a sequence number that is attached to each distance value. Increasing sequence numbers are generated by the node the corresponding distance value corresponds to and represent "freshness" of the routing information. For the routing decisions, only fresh information is used, which avoids well-known problems with distance-vector protocols such as routing loops and the count-to-infinity phenomenon.

Every time a node broadcasts its distance vector, it generates a new even sequence number larger than the last one and tags the distance to itself with this sequence number. When updating their local distance vectors, the sequence number received from the selected next-hop neighbor is copied. Due to the asynchronous distance vector propagation scheme, it may happen that a node receives fresh routing information about a particular destination from different neighbors in a pattern such that information about a path with a worse metric arrives before the optimal path for that destination. Because only paths that are tagged with the maximum sequence number known for the particular destination node are considered when making the routing decision, this scenario could lead to nodes frequently switching the next-hop neighbor for a destination. To address this issue, the authors suggest a node should not immediately announce its changed routing decision upon receiving the new information but wait a certain time for the arrival of fresh information from the other neighbors.

When a node detects a broken link, it broadcasts a new distance vector for the affected paths. Paths that have been broken are tagged with an odd sequence number obtained by incrementing the previously used good sequence number by one. This ensures that the broken link information supersedes any previously announced information due to the sequence number. In order to cope with the highly dynamic nature of wireless networks, the authors also propose that important information such as broken links is reported immediately by incremental updates to the last distance vector information instead of only incorporating the new information in the next periodic broadcast.

4.2.2 Dynamic Source Routing in Ad-Hoc Wireless Networks (DSR)

A reactive routing protocol that uses source-controlled routes is described in [23]. In this protocol the source node is responsible for including the route a frame should take within the frame header. Intermediate nodes on the path simply examine the header to find the next-hop node they should send the frame to. For this to work, the source node must have information about a working path to the destination node available. This information is discovered by a *route discovery* process. In its simplest incarnation it works by flooding the network with route request messages that carry the destination node identifier for which the route must be found. A node receiving route request messages will rebroadcast the first route request message it receives, unless the node is the destination node. Every node that forwards the request will also add its own address to the request message in order to record the path that the message took. If the destination node receives the request message, it will answer with

a route reply message containing all the forwarding nodes that have been collected in the route request message received. When the reply reaches the source node, it can start to use the route.

If a link breaks, this will cause all routes using this link to fail. The authors assume that failures of local links can be detected. For example, this is the case when acknowledgment messages are used at the link layer protocol. If a node detects that a frame could not be forwarded to the next-hop neighbor, it will send a route error message back to the source node from which the failed frame originated. Upon receiving the error message, the source node may find a different route by restarting the route discovery process.

The inclusion of full routing information in both route reply and data messages allows for some sophisticated optimizations in the route discovery process. The basic idea is to have all nodes cache the routing information they have gathered in order to reduce cost for handling later route requests. Nodes learn about available routes they happen to be part of when they see a route reply or data message. Moreover, the wireless transceivers may be operated in promiscuous mode such that they process also frames destined to other nodes. This way, even more routes may be learned from frames sent by nearby nodes. This additional information can not only be used to help with routes requested by a node itself, but can also be used to answer route request messages coming from other nodes, thereby avoiding the need for the request message to travel all the way to the destination node.

4.2.3 Multi-path extensions to DSR

Several multi-path variations of DSR have been proposed in the literature [48, 36, 28, 49, 27]. The original DSR protocol can actually discover multiple paths between source and sink node, e.g. if multiple copies of a route request reach the destination node. Even if the destination node only returns one route reply to the source node, the source node may still receive several route replies if other nodes answer the request from their route cache. A popular idea found in the literature is to use the additional paths discovered by a route request to improve reliability without incurring additional cost.

The idea of the multi-path extension to DSR described by Nasipuri and Das in [36] is to have the destination node consider not only the first route request it receives, but also those that arrive later. The authors propose that the destination node should check whether these paths are link-disjoint from the first route discovered. Such paths are then provided to the source node or intermediate nodes who can use the alternative paths if the primary path breaks. Note that when an intermediate node n detects that its path to the destination node broke, this information is not known at the source, so the path stored in the messages will not work. When switching to an alternate path, node n is therefore responsible to rewrite the routing information in the data messages on the fly such they are handled by the alternate path. The authors also derive an analytical model describing the behavior of the alternate-path switching technique. They show that avoiding to rediscover routes when a path breaks by switching to an alternate path increases the average time between route discoveries, thus reducing the load caused by the route request floods.

Another technique that uses multiple routes potentially discovered by an unmodified DSR route discovery is described in [48]. Instead of using only a single path at a time and switching paths on failure of the current path, the authors propose to use the available paths in parallel. The mechanism described in the paper assumes a constant bandwidth-delay product

for the wireless network. By measuring the delay, the bandwidth of a particular path can be calculated. The data is then distributed onto the different paths according to their bandwidth estimates.

The MP-DSR protocol [28] due to Leung et al. uses a modified route discovery procedure. The objective they consider is to discover a set of paths with a given end-to-end reliability characteristic. The end-to-end reliability is defined to be the probability that at least one of a set of disjoint paths used for a flow is still available after a given time increment t. They compute this probability based on the path reliabilities, which is defined as the probability of a path surviving an interval of length t. The path reliability in turn is obtained by multiplying the link availabilities, i.e. the probability of a link staying usable in the interval.

Given a desired end-to-end reliability, the protocol first decides how many paths to discover and then sends that many route request messages to the neighbors with the highest link availability. In addition to the standard DSR route request information, the route request message also contains the minimum path reliability that the path being constructed should meet and the actual reliability for the path formed by the hops the request message has already visited. Before forwarding the request, a node checks whether the requested reliability is still being met by the accumulated reliability for the hop sequence constructed so far. It then updates the path reliability value in the request message and forwards the request. Eventually, requests arrive at the destination node that can provide the required reliability. The destination then sorts the paths by the reliability values and tries to construct a set of disjoint paths that meets the initial desired end-to-end reliability. The procedure might fail to find a suitable set of paths. In this case, the source node needs to restart the route discovery with more relaxed reliability parameters. One way to do this is to try and discover more paths. Another option is decreasing the time increment since paths are more likely to stay usable during shorter time intervals.

Route maintenance activity is required if all paths in the selected set of paths break or when the time window expires. In the latter case, the source node sends route check messages along the existing paths to gather their reliability values. The destination node reflects the path reliability values back to the source node which can then decide whether the overall reliability still meets the desired value or whether new routes must be discovered.

The extended DSR protocol [49] described by Wu modifies the original DSR route discovery process in order to construct two node-disjoint routes. This is achieved by starting two logically separate node discovery processes that are distinguished by adding a color of either black or white to the route request. Before forwarding requests, intermediate nodes decide which color they assume for the request and only forward request messages of the matching color. If both the black and the white route request messages arrive at the destination, two node-disjoint paths have been found. A problem with this technique is that paths completed from the route cache of an intermediate node are not guaranteed to be node-disjoint. Although the routes are kept with coloring information in the cache, it may happen that the white and black requests are completed using information obtained from different previous requests. These routes are not guaranteed to be node-disjoint because the property only holds for the pairs of paths that have been found by two requests that traveled all the way to the destination node.

4.2.4 Ad-hoc On-Demand Distance Vector Routing (AODV)

In [38], Perkins and Royer present a reactive version of the distance-vector approach. Similar to [23], routes are only established on demand, but in contrast to DSR, the AODV protocol keeps routing information only locally at the nodes instead of including it within the transmitted messages. Again, the protocol is comprised of route discovery and route maintenance procedures. For route discovery, a route request message is broadcast by the node that wants to discover a new route. In addition to information identifying the originating node and the particular route request, the message carries the destination node identifier, the hop count as well as a sequence number that indicates the required freshness of the route to be established. A node that receives a route request message checks its local routing table to see whether it has a route to the destination that is fresh enough (i.e. tagged with a sequence number not smaller than the one received as part of the request message). If not, the message is rebroadcast in the local neighborhood. Otherwise, if there is a route available, a route reply message is generated that is sent back to the node from which the original route request was received. For this purpose, all nodes that receive a route request must temporarily store information about the request (amongst others, destination node, request identification and the node from which the request was received), in order to match incoming route replies and to forward them to the source node. If no fresh route is known, the request message will eventually reach the destination node, which will then construct a reply message with a fresh sequence number. When handling route reply messages, the nodes check the sequence numbers and hop counts received with the reply and discard replies that are either not tagged with the newest destination sequence number, or carry a larger hop number than the current minimum. As in DSDV, this ensures that routing loops cannot form. Once a reply reaches the source node, the route has been established and the source node can start sending data messages.

For link breakage detection, the authors note that a link-level acknowledgment mechanism can be used if available. If not, they propose to periodically broadcast short hello messages in the one-hop neighborhood to enable the nodes to detect which incoming links are still usable. If a link breakage occurs, the node detecting it will notify all upstream neighbors being part of affected routes by sending an unsolicited route reply with an infinite metric value. This message will then propagate back to the source which can initiate a new route discovery process.

Note that during route discovery, the intermediate nodes unconditionally store some information in order to be able to forward replies and data should the route be established along a path they are part of. This information is subject to expiration, i.e. will be deleted after a timeout. A suitable expiration time must be chosen such that there is enough time for the request message to travel to the destination and a reply message back to source node. Routes that already have been established are aged out of the system in a similar fashion if they are not used anymore, but with a larger timeout.

4.2.5 Multi-path extensions to AODV

Several authors propose modified versions of AODV that make use of multiple paths to a destination node [33, 26]. Adding multi-path capabilities to AODV requires the nodes to keep information about several next-hop neighbors for a single destination node. This can be a problem, because the nodes must report the number of hops for a path they can provide to a

destination node in their route advertisement messages. However, the different paths known to a node are not necessarily of the same length.

The approach described by Lee and Gerla in [26] solves this problem by only advertising the primary (i.e. shortest) route to its neighbors. In addition to the primary route table, each node keeps an alternate route table into which it inserts next-hop entries for routes that it is not part of but overheard a route reply message for. If a node on the primary path later detects that its next-hop link is broken, it will broadcast the data message in the local neighborhood. If there is some node that has information in its alternate route table, this node will try to salvage the data by forwarding the data message to the next-hop neighbor learned earlier. Thus, the scheme tries to repair paths that locally break with the help of other nodes in proximity.

A more sophisticated multi-path extension called AOMDV (Ad hoc On-demand Multipath Distance Vector) is proposed in [33]. The authors propose to keep a list of next-hop neighbors for a single destination in the routing tables at each node. When advertising routes, a node specifies the maximum hop count of all the routes it has in the route reply message. This is required in order to conserve the loop-free property of AODV. For data messages, any of the routes available can be used.

AOMDV uses a modified route discovery scheme in order to discover multiple link-disjoint paths to a destination. The basic idea is to allow nodes to reply to more than just the first route request message it receives. The multiple replies will then travel back to the node that originated the request and set up multiple paths. The link-disjointness property of the paths is established by only replying to route requests that left the originating node via different links. This guarantees link-disjoint paths since every node forwards a single route request only once. In order to enable the answering node to decide whether the paths are different, the originating node stores the next-hop neighbor it sends the message to in the route request. Before answering an incoming request, a node makes sure it has not already answered this request for the specified neighbor of the originating node yet. The authors also introduce a parameter that restricts the maximum number of replies a node may generate for a request in order to contain the additional load.

4.2.6 Assessment

The routing protocols described in the previous sections almost exclusively consider networks using competition-based medium access mechanisms. However, some form of route discovery is also needed in TDMA networks. Therefore, the basic ideas found in DSR and AODV are useful in combined route discovery and slot reservation protocols for TDMA networks, some of which are described below in Section 4.4. In reservation based systems, route discovery also needs to take resource availability into consideration, which the protocols described above do not account for, since they are meant for systems providing only best-effort service. Resource availability restrictions are also a problem for route caching mechanisms as used with DSR and AODV. Once a discovered path is used, some of the available capacity will be claimed. Thus, the available capacity from the discovery phase will be invalid and should not be cached.

Most of the literature discussed above uses multi-path techniques for improving reliability. Since path capacity bounds are rarely available, it is unclear how to decide what load a path can actually handle. This makes it hard to split up a flow appropriately so that it can take advantage of using different available capacity shares on the links. The approach described

in [48] uses the bandwidth-delay product to estimate the relative capacity of a path and tries to balance the load. However, it cannot prevent the network from becoming overloaded.

Many of the protocols that use multi-path techniques to improve reliability consider node-disjoint or link-disjoint paths. The rationale is that single failing link or node should only affect at most one of the paths available to a flow. In case of a failing path, the protocol switches to a backup path. For reservation-based systems, keeping unused backup paths is very problematic. The allocated capacity cannot be used for other flows if the reservations are made in advance. A scheme that discovers feasible backup paths but only reserves capacity when it actually needs them might be a better choice.

4.3 TDMA scheduling

In realistic networks, several nodes can be allowed to transmit at the same time without interfering with each other as long as each node is sufficiently far away from every other transmitting node. This technique is referred to as *spatial reuse* and requires a solution to the problem of TDMA scheduling, i.e. deciding which transmissions are allowed to proceed in which micro slot. Two different variants of TDMA scheduling can be considered [41]: In the *broadcast scheduling* variant, all the neighbors of a transmitting node must be able to receive the message correctly. In our model we consider *link scheduling*, which requires that all scheduled links must be able to perform a successful transmission. TDMA scheduling schemes can further be categorized w.r.t. when the schedule is constructed. *Static* schedules are determined before the network is activated and are fixed during the operation of the network, while *dynamic* scheduling computes and changes the schedules dynamically during operation.

In systems that use static TDMA scheduling, schedules should be constructed in a way such that each node or link has the opportunity to transmit a message in at least one micro slot. Otherwise, it might happen that the network cannot provide service to some data flows in case they require to use a link or node that has not been assigned capacity in the schedule. In the static setting, different notions of optimal schedules are considered. They can refer to minimum schedule length [41], or to throughput metrics [24]. In the early literature, the networks are almost exclusively considered at the graph abstraction level, i.e. presuming a graph-based noninterference model. In this model, the problems are very similar to the graph-coloring problem [25], hence it is not surprising that many of the problems are indeed NP-hard [2, 8, 9].

Dynamic TDMA scheduling protocols such as [5, 31, 32, 34, 45, 51] that manage micro slot assignment to links or nodes work by running a coordination protocol in special time intervals that are separate from the time intervals controlled by the TDMA schedule. Often, the control message exchange time interval is situated at the start of the macro frame (e.g. [31, 34, 51]) but it may also precede each data slot [45]. In the latter case, the control message exchange before a micro slot determines the reservation of that slot. The former approach is more flexible, because it does not enforce a one-to-one correspondence between micro slots and control message intervals, which allows to reduce the time overhead caused by the reservation protocol. This is useful when reservations are expected to stay valid for a large number of macro slots, such that only few resources are required to make new reservations. In this case, a common control message time interval in which reservations for any micro slot

can be made is useful.

The CATA [45], RBRP [34] and FPRP [51] protocols solve the TDMA scheduling problem in the local neighborhood. The basic mode of operation is that whenever a node wants to send a flow of data messages to one of its neighbors, it can make reservations for an appropriate number of micro slots. In these protocols, a node is only granted a reservation after successfully contending for a micro slot during the control message exchange interval. Because the reservation protocol is contention-based, it needs to address similar problems as pure contention-based medium access mechanisms. One is the hidden terminal problem, i.e. the situation that messages transmitted concurrently by two senders that cannot hear each other may still collide at a node that hears both of them. In the reservation protocols, this can be addressed by making the intended receivers acknowledge whether a control message has been received correctly or not. Another problem in the context of broadcast TDMA scheduling is that of so-called *deadlocks*. Due to accurate time synchronization as required for TDMA, all nodes in a local neighborhood may transmit their reservation messages concurrently. If their is no common receiver, they cannot detect the concurrent reservation. The protocols described below address this by randomized detection schemes.

The Five-Phase Reservation Protocol (FPRP) [51] by Zhu and Corson reserves slots for broadcast transmissions by means of a control message exchange during a reservation time interval at the beginning of the macro slot. For each data micro slot, there is a corresponding reservation slot in this interval. A reservation message exchange consists of five phases: The first is the reservation request phase during which a node that wants to make a reservation sends a request message. In the second phase, neighbors report potential collisions in case concurrent reservation requests have been detected during the first phase. Note that the nodes must be able to detect collisions, both for the collisions of the request message and the collision reports sent by neighbors. A reservation requesting node that has not received an indication of a collision in the second phase sends a confirmation message during the third phase, which is acknowledged by all neighbors in the fourth phase. The fifth phase can be used by requesting nodes to break deadlocks probabilistically: A requesting node sends a message with a probability of 0.5. Other requesting nodes that choose to keep quiet can thus learn that their reservation failed.

Another broadcast reservation protocol put forward by Marina et al. is RBRP [34]. It uses a number of so-called reservation slots for the control message exchange. Any micro slot can be reserved in any reservation slot by stating the number of the requested micro slot in the reservation message. If a node wants to make a reservation, it participates in each of the reservation slots with a certain probability. Each reservation slot further consists of m minislots and negative acknowledgment and confirmation intervals. The requesting node randomly picks k of the m minislots and transmits reservation requests during these minislots. At the end of the reservation slot, nodes that detected a conflict will send a negative acknowledgment. If a requesting node does not receive a negative acknowledgment, it will consider the reservation as successful and announce this by a confirmation message. The randomization both in the decision in which reservation slots to participate and the choice of minislots helps to avoid deadlock situations. If a requesting node hears another request in a minislot, it learns that another node is trying to make a reservation concurrently and will abort its attempt to claim the slot. A deadlock probability analysis described in the paper shows that the deadlock probability is kept very low by these measures. RBRP employs another robustness-enhancing technique. At the beginning of each micro slot, there is a short reconfirmation phase in which

the sending node announces its intent to send and listening nodes again have the possibility to issue negative acknowledgments. The purpose of this is to be able to detect schedule conflicts that result from environment changes such as node mobility. After detecting the conflict, a node will drop its reservation and contend for a new micro slot in the next reservation phase.

Although the protocols discussed above only consider single-hop reservations, a similar mechanism is required multi-hop reservation systems. At each hop, the sending node needs to run a protocol in order to agree on reservations with the local neighborhood. Multi-hop reservations can then be made in a hop-by-hop fashion. Note that both in FPRP and RBRP only nodes that have bidirectional links between each other will participate in the reservation protocol. Therefore, problems will arise in the case of unidirectional links, e.g. caused by variations in background noise w.r.t. the different node locations. Furthermore, effects such as interference from nodes that are not in transmission distance but close enough to increase the local background noise due to their transmissions such that local communication is hindered, are not considered. We have seen in the evaluation in Chapter 3 that under the physical noninterference model, a smaller number of nodes could transmit concurrently than under the graph-based model. Thus, resource reservation protocols should probably consider more than the local one-hop neighborhood.

4.4 QoS-aware multi-hop slot reservation

Applications that have strong QoS requirements regarding the service provided by the network such as voice or video streams can only be adequately handled by making resource guarantees on a per flow basis. Resource reservation schemes allow end-to-end QoS guarantees such as guaranteed minimum throughput and maximum latency. In dynamically scheduled TDMA networks, this situation requires protocols that discover paths and allocate micro slots for multi-hop flows such that the flows' demands are met. This also includes a form of admission control. A flow is only accepted to be handled if enough resources for it can be reserved, otherwise it is rejected. References [5, 22, 30, 31, 32] describe some protocols for QoS routing and reservation in dynamic TDMA networks.

The protocol due to Liao et al. described in [30] uses an approach similar to DSR [23] to jointly discover paths capable of providing a requested transfer rate and make appropriate reservations in a TDMA network. When a node is requested to handle a new flow, it will flood a route request message into the network. In addition to the DSR information, the message contains the required capacity in form of the number of slots per macro slot that must be allocated on each hop. When a node receives the message, it checks whether it has enough free slots to any of its neighbors to handle the flow. If so, it rebroadcasts the message, including information about the possible next-hop neighbors. Each node retransmitting the message also adds a vector of slots it is going to allocate for the flow to the message. When it arrives at the destination, a suitable path has been found. The destination then returns a reply message to the source on the reverse path. Intermediate nodes make the reservations upon receiving the reply message.

For the decision about which slots are usable for a transmission, a set of conditions resembling the graph-based noninterference model discussed in Section 2.3 is used. Note that in order to make sure a reservation does not interfere with any other already scheduled transmissions, a node needs information about the allocations made by nodes in its two-hop neighbor-

hood. The authors propose to distribute this data by having each node periodically broadcast the reservation it knows about to the neighborhood.

The authors note that allocation schemes are sensible to which slots are reserved for a hop if there are multiple choices. A hop-by-hop allocation scheme as employed by the protocol described in the paper may lead to situations where downstream nodes cannot find any possible slots to allocate due to the allocation pattern used by upstream nodes and thus decide to reject the route request. However, it is possible that another allocation pattern would have led to a successful route establishment.

Some shortcomings of the protocol are addressed in [22]. The authors identify two situations that can possibly lead to the same slot incorrectly being reserved for several flows. These issues are caused by reservation requests handled at a node while other requests affecting this node have not yet been confirmed or canceled. Consider that a node v handles a second reservation request before the reply message of the first request has arrived. If v does not keep track of the half-reserved slots it has decided to use for the earlier request, these slots can be used again. Later, when the reply messages matching the two requests arrive, v would reserve the same slot twice. This can also happen if two nodes situated in the direct neighborhood of each other offer to reserve the same slot to two different reservation requests they process concurrently. When the respective reply arrives, they both reserve the slot, thereby violating the noninterference condition. Jawhar and Wu remedy this situation by introducing an allocated-but-not-reserved state with which every slot that a node has offered in a request message is tagged. The noninterference calculations can then consider these slots as being used, thus avoiding multiple reservations.

The protocols described in [5, 31, 32] are targeted at a different network model that considerably relaxes the noninterference conditions. The authors assume the network to use a CDMA technique on top of the TDMA network. Every node is assumed to be assigned its own CDMA code. In this model, adjacent nodes may share a slot without interfering and the hidden terminal problem is also avoided. The only restriction for slot reservation is that a node cannot send and receive at the same time. In [32], the authors note that given the sets of free slots at each node along a path, deciding whether there is a suitable allocation pattern to establish a connection is equivalent to the satisfiability problem, i.e. the problem is NP-complete.

The protocol described in [32] uses a hop-by-hop scheme to calculate the number of available slots for a route between the source and the destination node. The mechanism extends the DSDV ad hoc routing protocol [37] to also include the set of slots that a route to a given destination d uses at the next hop destination. Based on this information, the local node can decide which slots it wants to use for the route to d and what the maximum supportable number of allocations per macro slot is. When a node wants to use a particular route to handle a flow, it sends a reservation message along this route. The intermediate nodes will reserve the slots as determined by the pre-calculated information in the routing tables and remove the allocated slots from any other routes. When the reservation message reaches the destination, it will return a reply message to the source indicating that the path has been reserved. Due to the dynamic nature of the routing tables, it may happen that an intermediate node finds it does not have enough free slots to satisfy the reservation, in which case it replies with a reset message. A reactive variant of the protocol by the same author is described in [31].

A similar, but more sophisticated reactive QoS slot allocation protocol due to Chen et al. is proposed in [5]. When a route to a destination node must be discovered, the source node

floods a request message that specifies the capacity requirement into the network. Each node that forwards the request will add its set of free slots to the message and rebroadcast it. Eventually, different copies of the request that have traveled via different paths will arrive at the destination. The destination extracts the free slots list from the message and uses it to decide which slots to allocate at each node such that the flow's capacity requirement is met. As mentioned above, this problem is NP-complete. The authors propose to have the destination node apply a heuristic approach to construct suitable slot allocation patterns for all the paths that have been discovered. When the set of potential paths and their capacity is known, the destination node selects a set of paths to handle the flow and sends reservation messages back along these paths to reserve the capacity as indicated in the message. Compared to the hop-by-hop approach in [31, 32], the availability of free slot information for all nodes on the paths potentially allows for better slot allocation decisions.

While [30] addresses the multi-hop reservation problem in TDMA networks and [22] extends the protocol to guard against concurrent reservation, the authors do not elaborate on how to ensure mutually exclusive resource allocation in the local neighborhood. [22] proposes to have a node broadcast its reservations to the two-hop neighborhood. It is unclear how these broadcasts are coordinated such that they do not collide. Furthermore, it is still possible that two adjacent nodes reserve the same slot at the same time without having heard each other's broadcasts. Mechanisms such as employed in RBRP [34] and FPRP [51] are necessary to handle these situations gracefully. The CDMA-over-TDMA approach proposed in [5, 31, 32] eliminates the concurrent allocation problems. However, the CDMA mechanism needs additional infrastructure for assigning pairwise orthogonal codes to the nodes. Additionally, the combination of CDMA over TDMA typically needs more complex hardware, resulting in both higher cost and increased energy consumption.

4.5 Load balancing and optimal routing

As mentioned earlier, the interference effects due to the shared wireless medium significantly limit the number of concurrent transmissions that are possible. In [17], Gupta and Kumar show that in a setting of n nodes arbitrarily placed in a disk in the plane, the throughput available to each node is in the order of $\Theta\left(\frac{W}{\bar{L}} \cdot \sqrt{\frac{A}{n}}\right)$ with W being the common transmission rate of the nodes, A the area of the plane and $\bar{L}$ the mean distance between source and sink nodes of a flow. The result is also proved to be true for networks in which the node placement in the unit disk and the traffic pattern is not arbitrary, but random. The implication is that the capacity available to a node depends on the node density, i.e. the number of nodes per area. If the node density and the transmission rate are presumed to be fixed, the only way to increase capacity per node is to reduce the mean distance between source and sink node. Thus, large wireless networks with non-local communication patterns face system inherent capacity problems, suggesting different design alternatives should be considered. The follow-up paper [16] extends the work to networks in which the node locations are not restricted to planes or surfaces but can be placed in 3-dimensional space. For this setting, similar results are obtained.

It should be noted that the lower capacity bound is derived in [17] in a constructive way, thereby explicitly specifying a way to compute routes for the frames. This scheme uses single-

path routing. The resulting bound is asymptotically equivalent to the upper bound, demonstrating that single-path routing is equivalent to multi-path routing in terms of asymptotic performance.

The paper by Hyytiä and Virtamo [18] also arrives at the conclusion that single-path routing schemes can yield optimal results. They model the network nodes as a continuum in the shape of a disk. In the continuum model, a packet flow is described by a flux in analogy to the flux notion in physics. A route through the network is simply a curve within the disk that connects the source node to the sink node. The class of routings considered in the paper has the property that the routing decision for a packet at position x only depends on the destination d and is given by the direction in which the packet is forwarded. The objective to be minimized is maximum network load over the whole network, i.e. the maximum flux density at any point in the disk.

The authors consider the setting of a uniform traffic load, i.e. the same constant flow of data from each possible source to each possible sink. It is shown that for each multi-path routing, which is conceived as a combination of a possibly infinite number of single-path routings, a single path routing can be constructed exhibiting a maximum load not larger than that of the multi-path routing. The shortest-path routing creates a high load hot spot in the center of the network. Using a combination of several routing schemes, the authors are able to construct a routing that cuts the load almost in half when compared to the shortest-path routing. The continuum model is indeed a very attractive choice at least for massively dense wireless networks. However, more research is needed to determine how these results relate to actual networks that consist of finite numbers of nodes.

Building upon earlier work [39], Ganjali and Keshavarzian describe a geometric method for estimating the load caused by multi-path routings comprised of a given number of paths [14]. The setting they examine is a network of uniformly distributed nodes in a disk and a multi-path routing scheme that uses paths close to the straight line connecting source and sink nodes. Their approach is to approximate the region in which the paths making up the multi-path routing are situated by a rectangle. Increasing the number of paths is expected to lead to a linear increase of the rectangle's width. For any source node position a and forwarding node position f, they derive the area in which a sink node b must be located such that f is within the routing rectangle for a flow from a to b. The sum of these area sizes for all possible source node locations is presumed to correspond to the load experienced at f. Simulations are used to empirically determine the rectangle width parameter and to validate the model, which is then used to compute the expected load for a node in a given distance from the center of the network. The results indicate that for practical values for the number of paths up to 50, the load experienced by the nodes is almost invariant w.r.t. the number of paths.

The results discussed above agree very well with our findings in Section 3.9. In terms of performance metrics such as capacity usage (corresponding to network load) and latency, the multi-path approach does not show a significant advantage over single-path routing in the setting of random networks. While the literature mainly discusses large and dense wireless networks on an abstract level, our results indicate that similar multi-path performance behavior is also found in smaller networks under a more realistic system model that considers effects such as physical radio propagation characteristics.

5 Conclusion

In this chapter, we reflect on the previous chapters, discuss and interpret the results, elaborate on conclusions to draw and also give some ideas for follow-up work.

In Chapter 2, a formal model of wireless networks has been developed. This model captures many of the important aspects of actual wireless networks in an accurate way. The model can accommodate different noninterference models. This makes it convenient to tailor the model for use on an abstract level by means of the graph-based noninterference model that only considers topological features of the underlying connectivity graph but also allows to use physical noninterference models coming closer to the physical effects that determine whether a transmission can be successfully received.

On this foundation, the model has been extended with the notion of consistent slot allocations that model the set of transmissions that can be allowed to proceed concurrently at the same time in a slot. This allows to consider network activity independent of actual load offered to the network. We have modeled multi-hop data transmissions as data flows. The network handles the flows by forwarding data frames belonging to a flow along one or more paths. Whether a network can support a flow depends on whether sufficient interference free resources are available that can be mapped to the paths handling the flow. Finally, we have introduced QoS parameters on a per flow basis and defined conditions that must be met for the network to solve this scenario.

Metrics have been defined in order to assess the quality of a solution to a scenario. The reduced indeterminism over competition based medium access control mechanisms resulting from the use of a network-wide TDMA approach allows us to capture important characteristics such as network load, end-to-end latency and energy consumption in a very intuitive and precise way.

Optimal solutions to scenarios in the model have been discussed in Chapter 3. The slot allocation problem was introduced which captures the conditions for solving a scenario from the point of view of allocating individual slots. We have found the slot allocation problem to be NP-complete. Therefore, solving realistic examples optimally in reasonable time is only possible for scenarios of restricted size, i.e. with a very limited number of flows. For solving the problem, we have both developed a mixed integer program formulation and a specialized branch and bound approach that allows to obtain and examine optimal solutions to scenarios of a size sufficient to compare multi-path and single-path solutions.

The results of our comparative study of single-path and multi-path solutions in random wireless networks of different characteristics indicate that the reduction of network load and improvement in latency achievable with multi-path routing approaches is actually quite marginal. This is a surprising result given that multi-path solutions are intuitively anticipated to better distribute network load and to consequently alleviate hot spots that deteriorate performance. However, this result agrees with and complements other studies [14, 18] that examined the effect of multi-path approaches on network load distribution in very abstract models. We confirm the findings for smaller, but physically more accurately modeled networks.

An interesting result with implications for modeling wireless networks has been obtained by comparing solutions under different noninterference models. It turns out that under the physical noninterference model, the network capacity is much smaller than under the graph-based noninterference model due to the increased region of nodes that is affected by interference caused by a single transmission. This means that network models which consider only graph topologies tend to idealize the interference problem. Consequently, results obtained in this model should be carefully validated against physical networks or more accurate models. However, it is worth noting that while the network model behaves subtly different depending on the noninterference model, the results we obtained match very closely from a qualitative point of view. This legitimates graph-based models as useful, though not always accurate abstraction.

In the evaluation, only optimal solutions to the random scenarios have been considered. This has the advantage of being able to quantify the expected multi-path performance gain independent of a particular algorithm employed by a slot allocation protocol as long as the protocol performs reasonably well, i.e. it is able to compute solutions that are close to optimal. If so, our results show that multi-path routing is probably not worth the effort when used to try and improve network load and end-to-end delay characteristics.

However, it has also been shown that the slot allocation problem is inherently difficult. Thus, it is unclear what performance figures are actually achievable by practical protocols when compared to an optimal solution. In order to save message transmissions, slot allocation protocols are often designed as distributed algorithms. The unavailability of global information thus further reduces the chance of finding an optimal solution. Hence, it is unclear how well actual protocols like the ones described in [22, 30] actually perform compared to an optimal solution. Therefore, it would be very interesting to create an implementation of these protocols within the framework developed for this thesis such that the protocol's solutions can be directly compared to the optimal solution obtained by the solver.

Another area that deserves attention in follow-up work is the dynamic nature of many wireless networks, i.e. mobile nodes and changes in medium characteristics over time. In this thesis, we have only considered snapshots of the wireless network at an instant in time and evaluated the optimal scenario solutions for this particular situation. Of course, dynamically changing networks can be captured in this framework by reconsidering the whole scenario when the situation has changed. However, this approach is not practical in actual protocols. Instead, they should react and repair their current solution if this is required by environmental changes. An interesting approach would be to try and evaluate the robustness of solutions in dynamically changing networks and see whether solutions optimal w.r.t. network load, throughput or latency achieve similar robustness as non-optimal solutions.

An additional issue in this context is the fact that flows typically must be allocated capacity for online, i.e. at the time the network is requested to handle a flow. Parameters of future flows are unknown at this time. Even if a slot allocation algorithm manages to establish an optimal solution for the current scenario, constructing an optimal solution after adding an additional flow might require to change the reservations of current flows. In the worst case, the additional flow can only be handled if the existing reservations are changed. Whether such a reallocation-capable slot allocation protocol is practical is an open question, as well as the benefit a reallocation scheme can provide in general.

Although the evaluation of multi-path routing performance gains show very little improvement in realistic scenarios with 50 nodes, larger scenarios are difficult to investigate in our

framework due to the combinatorial properties of the problem. Either by further improving the scenario solving algorithm or by developing approximation techniques of guaranteed performance, our methodology could be applied to larger problem instances. These includes a number of different interesting settings:

- Increase the hop distance between source and sink node for the flows under consideration to values larger than 7. This might lead to more independent paths and enlarge the multi-path performance gain.
- Consider more realistic networks that provide more than 20 micro slots in their TDMA schedule. Investigate whether also scaling the other relevant system parameters such as capacity requirements appropriately yields the same results as before.
- Study the effect of increasing the number of paths a flow can potentially use to values in the order of 10–100.

Despite these open questions, some conclusions can be drawn from our results. The most obvious is that multi-path routing should not be considered as a possibility to increase system performance in categories such as network load or end-to-end delay. This is particularly true in situations where discovering and maintaining multiple paths in the network adds additional overhead to the respective network protocols, as the overhead will likely void the little multi-path performance gain. However, we did not consider reliability characteristics of solutions in this thesis. Multi-path approaches are expected to help improve reliability in highly-dynamic networks. Research that considers this aspect can be found e.g. in [28, 36].

It is also worthwhile to consider the question *why* multi-path routing does not yield significant performance gains. Compared to wired systems in which multi-path techniques have been successfully used to improve throughput and latency [6, 47], wireless networks differ primarily in the characteristics of the shared medium. The resulting interference problems are known to reduce network capacity significantly [17] and should not be underestimated. Thus, it is worth considering techniques that try to reduce the interference problem. One possibility is to use antennas that have directional characteristics. This will reduce the interference experienced at other adjacent nodes and thus possibly increase the multi-path performance gain. However, directional antennas often require additional setup work or additional hardware and can therefore not be used in applications such as WSNs.

When designing networks with the goal of taking advantage of multi-path routing, several points should be observed. Obviously, the bottleneck situations discussed in Section 3.3 should be avoided. This includes the source and sink nodes. If they are to be capable of using the total transmission rate a single link can provide, it is important to have several neighbors at the source and sink nodes that do not interfere with each other. Because forwarding neighbor nodes need time to push incoming data to their respective neighbors, the source node must spread its outgoing transmissions to different neighbors if it wants to send at full rate. Similar considerations apply for the sink node. Furthermore, if providing multiple paths in a network, one should make sure these paths interfere as little as possible. As soon as a node in one path blocks transmission in a neighbor path, the multi-path performance gain is likely to be lost, because the transmissions on both paths must be serialized.

Bibliography

[1] N. Abramson. The ALOHA System - Another Alternative for Computer Communications. In *Proceedings of Fall Joint Computer Conference*, pages 281–285, 1970.

[2] Erdal Arikan. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory*, 30(4):681–685, 1984.

[3] Philipp Becker, Reinhard Gotzhein, and Thomas Kuhn. Model-driven Performance Simulation of Self-organizing Systems with PartsSim. *Praxis der Informationsverarbeitung und Kommunikation*, 31(1):45–50, 2008.

[4] Johnny Chen, Peter Druschel, and Devika Subramanian. An Efficient Multipath Forwarding Method. In *IEEE INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1418–1425, March 1998.

[5] Yuh-Shyan Chen, Yu-Chee Tseng, Jang-Ping Sheu, and Po-Hsuen Kuo. An on-demand, link-state, multi-path QoS routing in a wireless mobile ad-hoc network. *Computer Communications*, 27(1):27–40, January 2004.

[6] Israel Cidon, Raphael Rom, and Yuval Shavitt. Analysis of Multi-Path Routing. *IEEE/ACM Transactions on Networking*, 7(6):885–896, December 1999.

[7] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. *SIGOPS Operating Systems Review*, 36(special issue):147–163, 2002.

[8] A. Ephemerides and T.V. Truong. Scheduling Broadcasts in Multihop Radio Networks. *IEEE Transactions on Communications*, 38(4):456–460, April 1990.

[9] S. Even, O. Goldreich, S. Moran, and P. Tong. On the NP-Completeness of Certain Network Testing Problems. *Networks*, 14:1–24, 1984.

[10] Laura Marie Feeney. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 6 (3):239–249, June 2001.

[11] Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *INFOCOM*, pages 1548–1557, April 2001.

[12] Free Software Foundaction, Inc. GNU Linear Programming Kit, 2008. URL `http://www.gnu.org/software/glpk/`.

[13] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync Protocol for Sensor Networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, New York, NY, USA, 2003. ACM.

[14] Yashar Ganjali and Abtin Keshavarzian. Load Balancing in Ad Hoc Networks: Single-path Routing vs. Multi-path Routing. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1120–1125, 2004.

[15] Reinhard Gotzhein and Thomas Kuhn. Decentralized Tick Synchronization for Multi-hop Medium Slotting in Wireless Ad Hoc Networks using Black Bursts. In *SECON '08. 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 422–431, June 2008.

[16] Piyush Gupta and P. R. Kumar. Internets in the Sky: Capacity of 3-D Wireless Networks. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 3, pages 2290–2295, 2000.

[17] Piyush Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[18] Esa Hyytiä and Jorma Virtamo. On Optimality of Single-Path Routes in Massively Dense Wireless Multi-Hop Networks. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 28–35, New York, NY, USA, 2007. ACM.

[19] IEEE 802.16-2004. IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems, 2004.

[20] ILOG, Inc. ILOG 9.1 – User's and Reference manuals, 2005. URL `http://www.ilog.com`.

[21] Texas Instruments. 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver, 2007. URL `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[22] Imad Jawhar and Jie Wu. A Race-Free Bandwidth Reservation Protocol for QoS Routing in Mobile Ad Hoc Networks. In *Proceedings of the 37th Annual Hawaii Internation Conference on System Sciences*, January 2004.

[23] David B Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 353:153–181, 1996.

[24] Ji-Her Ju and Victor O. K. Li. An Optimal Topology-Transparent Scheduling Method in Multihop Packet Radio Networks. *IEEE/ACM Transactions on Networking*, 6(3): 298–306, 1998.

[25] Richard M. Karp. Reducibility Among Combinatorial Problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations (Proceedings of a Symposium on the Complexity of Computer Computations)*, pages 85–103. Plenum Press, New York, 1972.

[26] Sung-Ju Lee and Mario Gerla. AODV-BR: Backup Routing in Ad hoc Networks. *Wireless Communications and Networking Conference*, 3:1311–1316, 2000.

[27] Sung-Ju Lee and Mario Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. *IEEE International Conference on Communications*, 10:3201–3205, June 2001.

[28] Roy Leung, Jilei Liu, Edmond Poon, Ah-Lot Charles Chan, and Baochun Li. MP-DSR: A QoS-Aware Multi-Path Dynamic Source Routing Protocol for Wireless Ad-Hoc Networks. In *26th Annual IEEE International Conference on Local Computer Networks*, page 132. IEEE Computer Society, 2001.

[29] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of Ad Hoc Wireless Networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 61–69, New York, NY, USA, 2001. ACM.

[30] Wen-Hwa Liao, Yu-Chee Tseng, and Kuei-Ping Shih. A TDMA-based Bandwidth Reservation Protocol for QoS Routing in a Wireless Mobile Ad Hoc Network. In *ICC 2002. IEEE International Conference on Communications*, volume 5, pages 3186–3190, 2002.

[31] Chunhung Richard Lin. On-Demand QoS Routing in Multihop Mobile Networks. In *IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1735–1744, April 2001.

[32] Chunhung Richard Lin and Jain-Shing Liu. QoS Routing in Ad Hoc Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1426–1438, August 1999.

[33] Mahesh K. Marina and Samir R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of the Ninth International Conference on Network Protocols*, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[34] Mahesh. K. Marina, George D. Kondylis, and Ulaş C. Kozat. RBRP: A Robust Broadcast Reservation Protocol for Mobile Ad Hoc Networks. In *ICCC 2001. IEEE International Conference on Communications*, volume 3, pages 878–885, June 2001.

[35] D. L. Mills. Internet Time Synchronization: the Network Time Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.

[36] Asis Nasipuri and Samir R. Das. On-Demand Multipath Routing for Mobile Ad Hoc Networks. In *Eight International Conference on Computer Communications and Networks*, pages 64–70, 1999.

[37] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *SIGCOMM Computer Communication Review*, 24(4):234–244, 1994.

[38] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the Second Workshop on Mobile Computing Systems and Applications*, pages 90–100. IEEE press, 1999.

[39] Peter P. Pham and Sylvie Perreau. Performance Analysis of Reactive Shortest Path and Multi-path Routing Mechanism With Load Balance. *INFOCOM 2003: Twenty-Secod Annual Joint Conference of the IEEE Computer and Communications Societies*, 1:251–259, 2003.

[40] Subramanian Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5(2):81–94, March 1999.

[41] Subramanian Ramanathan and Errol L. Lloyd. Scheduling Algorithms for Multihop Radio Networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.

[42] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, New York, NY, USA, 1998. ACM.

[43] Tsenka Stoyanova, Fotis Kerasiotis, Aggeliki Prayati, and George Papadopoulos. Evaluation of impact factors on RSS accuracy for localization and tracking applications. In *Proceedings of the 5th ACM international workshop on Mobility management and wireless access*, pages 9–16, New York, NY, USA, 2007. ACM.

[44] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock Synchronization for Wireless Sensor Networks: A Survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[45] Zhenyu Tang and J.J. Garcia-Luna-Aceves. A Protocol for Topology-Dependent Transmission Scheduling in Wireless Networks. In *WCNC. 1999 IEEE Wireless Communications and Networking Conference*, volume 3, pages 1333–1337, September 1999.

[46] Aristotelis Tsirigos and Zygmunt J. Haas. Multipath Routing in the Presence of Frequenct Topological Changes. *IEEE Communications Magazine*, 39(11):132–138, November 2001.

[47] Srinivas Vutukury and J. J. Garcia-Luna-Aceves. A Simple Approximation to Minimum-Delay Routing. *ACM SIGCOMM Computer Communication Review*, 29(4): 227–238, October 1999.

[48] Lei Wang, Yantai Shr, Miao Dong, and Lianfang Zhang. Adaptive Multipath Source Routing in Ad Hoc Networks. In *IEEE International Conference on Communications*, volume 3, pages 867–871, 2001.

[49] Jie Wu. An Extended Dynamic Source Routing Scheme in Ad Hoc Wireless Networks. *Telecommunication Systems*, 22(1–4):61–75, January 2003.

[50] Qing Zhao and Lang Tong. Energy Efficiency of Large-Scale Wireless Networks: Proactive Versus Reactive Networking. *IEEE Journal on Selected Areas in Communications*, 23(5):1100–1112, May 2005.

[51] Chenxi Zhu and M. S. Corson. A Five-Phase Reservation Protocol (FPRP) for Mobile Ad Hoc Networks. *Wireless Networks*, 7(4):371–384, July 2001.

A Mixed Integer Program for the hexagon scenario

The following is a Mixed Integer Program formulation of the scenario introduced in the example given in Figure 3.1. The syntax used here follows the LP format that can be used with CPlex solver by ILOG [20] and that is also understood by many other optimization software packages. The variables named `a_pN_eXY_sS` represent the allocation variables that decide whether the edge (`X`, `Y`) is allocated during slot `S` to path index `N`. The edge usage variables are named `e_sS_eXY` and determine whether edge (`X`, `Y`) is in use during slot `s` or not. The version shown is a hand-crafted formulation of the Mixed Integer Program. A tool that converts problem descriptions in XML format into a Mixed Integer Program is part of the software developed while preparing this thesis.

```
\ Mixed integer programming model for the hexagon scenario

MINIMIZE
\ objective: minimize capacity usage
capacity_usage :
+ a_p1_e13_s0 + a_p1_e13_s1 + a_p1_e13_s2
+ a_p1_e34_s0 + a_p1_e34_s1 + a_p1_e34_s2
+ a_p1_e42_s0 + a_p1_e42_s1 + a_p1_e42_s2
+ a_p1_e15_s0 + a_p1_e15_s1 + a_p1_e15_s2
+ a_p1_e56_s0 + a_p1_e56_s1 + a_p1_e56_s2
+ a_p1_e62_s0 + a_p1_e62_s1 + a_p1_e62_s2
+ a_p2_e13_s0 + a_p2_e13_s1 + a_p2_e13_s2
+ a_p2_e34_s0 + a_p2_e34_s1 + a_p2_e34_s2
+ a_p2_e42_s0 + a_p2_e42_s1 + a_p2_e42_s2
+ a_p2_e15_s0 + a_p2_e15_s1 + a_p2_e15_s2
+ a_p2_e56_s0 + a_p2_e56_s1 + a_p2_e56_s2
+ a_p2_e62_s0 + a_p2_e62_s1 + a_p2_e62_s2

SUBJECT TO
\ path 1
+ a_p1_e13_s0 + a_p1_e13_s1 + a_p1_e13_s2
+ a_p1_e15_s0 + a_p1_e15_s1 + a_p1_e15_s2
= 1

- a_p1_e13_s0 - a_p1_e13_s1 - a_p1_e13_s2
+ a_p1_e34_s0 + a_p1_e34_s1 + a_p1_e34_s2
= 0

- a_p1_e34_s0 - a_p1_e34_s1 - a_p1_e34_s2
+ a_p1_e42_s0 + a_p1_e42_s1 + a_p1_e42_s2
= 0
```

```
- a_p1_e42_s0 - a_p1_e42_s1 - a_p1_e42_s2
- a_p1_e62_s0 - a_p1_e62_s1 - a_p1_e62_s2
= -1

- a_p1_e15_s0 - a_p1_e15_s1 - a_p1_e15_s2
+ a_p1_e56_s0 + a_p1_e56_s1 + a_p1_e56_s2
= 0

- a_p1_e56_s0 - a_p1_e56_s1 - a_p1_e56_s2
+ a_p1_e62_s0 + a_p1_e62_s1 + a_p1_e62_s2
= 0

+ a_p1_e13_s0 + a_p1_e13_s1 + a_p1_e13_s2
+ a_p1_e15_s0 + a_p1_e15_s1 + a_p1_e15_s2
= 1

\ path 2
- a_p2_e13_s0 - a_p2_e13_s1 - a_p2_e13_s2
+ a_p2_e34_s0 + a_p2_e34_s1 + a_p2_e34_s2
= 0

- a_p2_e34_s0 - a_p2_e34_s1 - a_p2_e34_s2
+ a_p2_e42_s0 + a_p2_e42_s1 + a_p2_e42_s2
= 0

- a_p2_e42_s0 - a_p2_e42_s1 - a_p2_e42_s2
- a_p2_e62_s0 - a_p2_e62_s1 - a_p2_e62_s2
= -1

- a_p2_e15_s0 - a_p2_e15_s1 - a_p2_e15_s2
+ a_p2_e56_s0 + a_p2_e56_s1 + a_p2_e56_s2
= 0

- a_p2_e56_s0 - a_p2_e56_s1 - a_p2_e56_s2
+ a_p2_e62_s0 + a_p2_e62_s1 + a_p2_e62_s2
= 0

\ fix binary variables that determine the active edges per slot
+ a_p1_e13_s0 + a_p2_e13_s0 - 2 e_s0_e13 <= 0

+ a_p1_e13_s1 + a_p2_e13_s1 - 2 e_s1_e13 <= 0

+ a_p1_e13_s2 + a_p2_e13_s2 - 2 e_s2_e13 <= 0

+ a_p1_e34_s0 + a_p2_e34_s0 - 2 e_s0_e34 <= 0

+ a_p1_e34_s1 + a_p2_e34_s1 - 2 e_s1_e34 <= 0

+ a_p1_e34_s2 + a_p2_e34_s2 - 2 e_s2_e34 <= 0

+ a_p1_e42_s0 + a_p2_e42_s0 - 2 e_s0_e42 <= 0
```

```
+ a_p1_e42_s1 + a_p2_e42_s1 - 2 e_s1_e42 <= 0

+ a_p1_e42_s2 + a_p2_e42_s2 - 2 e_s2_e42 <= 0

+ a_p1_e15_s0 + a_p2_e15_s0 - 2 e_s0_e15 <= 0

+ a_p1_e15_s1 + a_p2_e15_s1 - 2 e_s1_e15 <= 0

+ a_p1_e15_s2 + a_p2_e15_s2 - 2 e_s2_e15 <= 0

+ a_p1_e56_s0 + a_p2_e56_s0 - 2 e_s0_e56 <= 0

+ a_p1_e56_s1 + a_p2_e56_s1 - 2 e_s1_e56 <= 0

+ a_p1_e56_s2 + a_p2_e56_s2 - 2 e_s2_e56 <= 0

+ a_p1_e62_s0 + a_p2_e62_s0 - 2 e_s0_e62 <= 0

+ a_p1_e62_s1 + a_p2_e62_s1 - 2 e_s1_e62 <= 0

+ a_p1_e62_s2 + a_p2_e62_s2 - 2 e_s2_e62 <= 0

\ noninterference conditions
2 e_s0_e13 + e_s0_e34 + e_s0_e42 <= 2

2 e_s1_e13 + e_s1_e34 + e_s2_e42 <= 2

2 e_s2_e13 + e_s2_e34 + e_s2_e42 <= 2

e_s0_e34 + e_s0_e42 <= 1

e_s1_e34 + e_s1_e42 <= 1

e_s2_e34 + e_s2_e42 <= 1

e_s0_e42 + e_s0_e62 <= 1

e_s1_e42 + e_s1_e62 <= 1

e_s2_e42 + e_s2_e62 <= 1

2 e_s0_e15 + e_s0_e56 + e_s0_e62 <= 2

2 e_s1_e15 + e_s1_e56 + e_s1_e62 <= 2

2 e_s2_e15 + e_s2_e56 + e_s2_e62 <= 2

e_s0_e56 + e_s0_e62 <= 1

e_s1_e56 + e_s1_e62 <= 1
```

```
e_s2_e56 + e_s2_e62 <= 1

e_s0_e62 + e_s0_e42 <= 1

e_s1_e62 + e_s1_e42 <= 1

e_s2_e62 + e_s2_e42 <= 1

\ timing constraints
+ a_p1_e13_s0 + a_p2_e13_s0 + a_p1_e15_s0 + a_p2_e15_s0 <= 1

+ a_p1_e13_s1 + a_p2_e13_s1 + a_p1_e15_s1 + a_p2_e15_s1 <= 1

+ a_p1_e13_s2 + a_p2_e13_s2 + a_p1_e15_s2 + a_p2_e15_s2 <= 1

+ a_p1_e34_s0 + a_p2_e34_s0 <= 1

+ a_p1_e34_s1 + a_p2_e34_s1 <= 1

+ a_p1_e34_s2 + a_p2_e34_s2 <= 1

+ a_p1_e42_s0 + a_p2_e42_s0 <= 1

+ a_p1_e42_s1 + a_p2_e42_s1 <= 1

+ a_p1_e42_s2 + a_p2_e42_s2 <= 1

+ a_p1_e56_s0 + a_p2_e56_s0 <= 1

+ a_p1_e56_s1 + a_p2_e56_s1 <= 1

+ a_p1_e56_s2 + a_p2_e56_s2 <= 1

+ a_p1_e62_s0 + a_p2_e62_s0 <= 1

+ a_p1_e62_s1 + a_p2_e62_s1 <= 1

+ a_p1_e62_s2 + a_p2_e62_s2 <= 1

BINARY
\ allocations are binary
a_p1_e13_s0
a_p1_e13_s1
a_p1_e13_s2

a_p1_e34_s0
a_p1_e34_s1
a_p1_e34_s2

a_p1_e42_s0
a_p1_e42_s1
a_p1_e42_s2
```

```
a_p1_e15_s0
a_p1_e15_s1
a_p1_e15_s2

a_p1_e56_s0
a_p1_e56_s1
a_p1_e56_s2

a_p1_e62_s0
a_p1_e62_s1
a_p1_e62_s2

a_p2_e13_s0
a_p2_e13_s1
a_p2_e13_s2

a_p2_e34_s0
a_p2_e34_s1
a_p2_e34_s2

a_p2_e42_s0
a_p2_e42_s1
a_p2_e42_s2

a_p2_e15_s0
a_p2_e15_s1
a_p2_e15_s2

a_p2_e56_s0
a_p2_e56_s1
a_p2_e56_s2

a_p2_e62_s0
a_p2_e62_s1
a_p2_e62_s2

\ binary variables for edge activity
e_s0_e13
e_s1_e13
e_s2_e13
e_s0_e34
e_s1_e34
e_s2_e34
e_s0_e42
e_s1_e42
e_s2_e42
e_s0_e15
e_s1_e15
e_s2_e15
e_s0_e56
e_s1_e56
```

```
e_s2_e56
e_s0_e62
e_s1_e62
e_s2_e62

END
```

An optimal solution generated by the GLPK solver [12] is given below. Its metric value is 6, i.e. there are 6 allocations in the solution.

Variable	Value
a_p1_e13_s0	0
a_p1_e13_s1	0
a_p1_e13_s2	0
a_p1_e34_s0	0
a_p1_e34_s1	0
a_p1_e34_s2	0
a_p1_e42_s0	0
a_p1_e42_s1	0
a_p1_e42_s2	0
a_p1_e15_s0	0
a_p1_e15_s1	1
a_p1_e15_s2	0
a_p1_e56_s0	0
a_p1_e56_s1	0
a_p1_e56_s2	1
a_p1_e62_s0	1
a_p1_e62_s1	0
a_p1_e62_s2	0
a_p2_e13_s0	1
a_p2_e13_s1	0
a_p2_e13_s2	0
a_p2_e34_s0	0
a_p2_e34_s1	1
a_p2_e34_s2	0
a_p2_e42_s0	0
a_p2_e42_s1	0
a_p2_e42_s2	1
a_p2_e15_s0	0
a_p2_e15_s1	0
a_p2_e15_s2	0
a_p2_e56_s0	0
a_p2_e56_s1	0
a_p2_e56_s2	0
a_p2_e62_s0	0
a_p2_e62_s1	0
a_p2_e62_s2	0

Variable	Value
e_s0_e13	1
e_s1_e13	0
e_s2_e13	0
e_s0_e34	0
e_s1_e34	1
e_s2_e34	0
e_s0_e42	0
e_s1_e42	0
e_s2_e42	1
e_s0_e15	0
e_s1_e15	1
e_s2_e15	0
e_s0_e56	0
e_s1_e56	0
e_s2_e56	1
e_s0_e62	1
e_s1_e62	0
e_s2_e62	0

B Scenario and solution formats used by the branch and bound solver

The branch and bound solver implementation reads scenario definitions in XML format. A rendition of the hexagon scenario in this format is given below. Information contained in this description are the connectivity graph, the flows that make up the scenario, parameters describing the TDMA model in use and the noninterference model that is to be used when solving the scenario.

```
<?xml version="1.0" encoding="UTF-8"?>
<problem>
 <cgraph>
  <nodes>
   <node id="5" name="r1" x="0.8" y="0.7" z="0" tx_power="10"
    tx_rate="1000"/>
   <node id="6" name="r2" x="0.8" y="0.3" z="0" tx_power="10"
    tx_rate="1000"/>
   <node id="3" name="l1" x="0.2" y="0.7" z="0" tx_power="10"
    tx_rate="1000"/>
   <node id="4" name="l2" x="0.2" y="0.3" z="0" tx_power="10"
    tx_rate="1000"/>
   <node id="1" name="t" x="0.5" y="0.9" z="0" tx_power="10"
    tx_rate="1000"/>
   <node id="2" name="b" x="0.5" y="0.1" z="0" tx_power="10"
    tx_rate="1000"/>
  </nodes>
  <edges>
   <edge first="5" second="6"/>
   <edge first="5" second="1"/>
   <edge first="6" second="5"/>
   <edge first="6" second="2"/>
   <edge first="3" second="4"/>
   <edge first="3" second="1"/>
   <edge first="4" second="3"/>
   <edge first="4" second="2"/>
   <edge first="1" second="5"/>
   <edge first="1" second="3"/>
   <edge first="2" second="6"/>
   <edge first="2" second="4"/>
  </edges>
 </cgraph>
 <scenario>
  <flow id="1" name="flow1" source="1" sink="2" capacity="200"/>
 </scenario>
 <tdma_parameters number_of_slots="3" micro_slot_duration="0.1"
```

```
  macro_slot_duration="1" frame_size="100"/>
 <ni_model name="graph-based"/>
</problem>
```

The solutions output by the solver are also in XML format. For each flow and path index, allocations are given that form a path connecting the source to the sink node. The solver output obtained by running the solver on the scenario listed above follows. The metric the solver was requested to optimize is maximum latency.

```
<?xml version="1.0" encoding="UTF-8"?>
<solution>
 <routing flow="1">
  <path>
   <atom sender="1" receiver="5" slot="0" hop="0"/>
   <atom sender="5" receiver="6" slot="1" hop="1"/>
   <atom sender="6" receiver="2" slot="2" hop="2"/>
  </path>
  <path>
   <atom sender="1" receiver="3" slot="1" hop="0"/>
   <atom sender="3" receiver="4" slot="2" hop="1"/>
   <atom sender="4" receiver="2" slot="0" hop="2"/>
  </path>
 </routing>
</solution>
```