Design and Simulation of a Wireless Algorithm for Lane Switching on a Drone Road

---

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science
in the
University of Canterbury
by
Zhouyu Qu

University of Canterbury
2022

Abstract

Unmanned aerial vehicles (UAV) or drones have developed very fast in recent years, and they have started to be applied in many fields such as aerial photography, military, and law enforcement. In a conceivable future, people will deploy more drones for civil and commercial uses which are not fully implemented today. One example is parcel delivery. Drones in large numbers will carry parcels to different destinations. To limit the movement of those drones. One idea is to construct a "drone road," a virtual tube-like area in the airspace. Drones should usually cruise within the tube but be able to move outside to avoid collision, or other hazards. One of the biggest problems, similar to ground self-driving cars, are collisions. Collisions will cause severe economic loss and even threaten people's lives. Therefore, collision avoidance for drones is a valuable and important topic to research.

We propose a coordination-free algorithm for drones to avoid collisions on the drone road and present the result of a simulation-based analysis to evaluate how some parameters related to wireless communications can affect performance. This study first proposes a system model to clarify the boundaries of the collision-avoiding problem we attempt to solve. In this model, drones are required to fly within a long straight drone road with the same forward direction. Drones are only equipped with a GPS sensor and wireless communication components to perceive the environment. Moreover, only position and speed information can be shared with other drones. Then we propose a criteria-based algorithm specifying when drones should switch to another lane to avoid an incoming collision. Otherwise, drones should slow down their speed to wait for a good chance. Eight criteria are put forward in this research, including two baseline criteria "do nothing" and "always slow down." Other parameters considered in simulations include beacon interval, path loss exponent for log-distance model, transmit power, and drone density. We run simulations with those parameters on OMNet++ for evaluation. We furthermore present a cost model to represent the performance to quantify simulation results.

The simulation results show that packet loss is the main reason for collisions in our scenario. All simulation parameters can severely affect the packet

loss rate and then consequent the collision rate. Four out of eight criteria can significantly reduce the total cost compared to the baseline.

List of Figures

List of Tables

Table of Contents

Chapter I

Introduction

Unmanned aerial vehicles (UAV) or drones are flying vehicles that do not have any humans on board. The history of drones traces back to the 1840s. The first drone, which is a balloon carrier, was invented for the military [7]. From World War II until now, the development of drones has been rapid. By 2013, at least 50 countries have used UAVs for their own purposes. At the same time, the civil drone market is also expanding. Drones are used for aerial photography [8, 9, 10], agriculture and forestry [11, 12], infrastructure monitoring [13, 14], and law enforcement [15, 16], amongst many other application.

1.1 Using Drones for Parcel Delivering

Drones are widely used for aerial photography and have a strong potential ability to perform complicated tasks in many fields [17, 18, 19]. In a conceivable future, one of the tasks drones could carry out is to deliver goods and parcels in a city. This has some advantages for people, such as lowering the delivering cost. One example is in [20], a delivery company said it could save about $50M if drones are used in the last step of parcel delivering. Drones (Here we restrict to rotary blade drone like quadcopters) can not only deliver commodities for people's uses, but also transport healthcare items for rescue since they can cross the busy road which ground vehicles cannot [21]. Those benefits make it worth building a drone delivery system. At the same time a big problem, drone control, needs to be considered. In detail, people need to manage drones very well to prevent any safety or other issues that can negatively affect people's life.

Since the carrying capacity of one drone is quite limited (which is typically

from 0.3 to 20 kg) [22], to deliver the current daily average packages (e.g., approximately 43 million the for USA in 2020 [23]), we would need to deploy a large number of drones. Drones flying in the same area run the risk of colliding. Currently, there are some papers introducing algorithms addressing drone collision avoidance in a free airspace area [24, 25, 26]. Their objects are to detect the position and velocity of flying obstacles and then plan a collision-free route for drones. However, so many drones flying without any specified area restriction will cause the airspace to become chaotic, the flying obstacles will be more unpredictable since every drone can be considered an obstacle to others, and they all have different speeds and directions. Hence a drone may not react to the upcoming collision quickly enough or plan a complicated route to avoid all obstacles.

1.2 Drone Road

A good way to manage drones is to construct a "drone road," a virtual, tube-like area in airspace; drones will be asked to fly in the tube but can also be out of them when necessary to avoid an incoming collision. Such a virtual road can be planned to set it in high attitude airspace by the government, which reduces the interference from ground-based wireless communication and lower the possibility that drones affect people's lives. In or around the drone road, we assume that there are non-overlapping sub-tubes or lanes to restrict the drone's position in the section area of the drone road. Drones are forced to cruise in those lanes but can switch lanes. Those lanes are pre-defined, just like lanes for ground vehicles in streets.

In this project, the scenario we will focus on is a straight and finitely long tube segment, in which all drones fly in the same direction. A collision can happen when a faster drone flies up onto a slower one in the same lane of the tube, or when they are in different lanes but one of them switches lane. The faster drone should change the lane and overtake the slower one to avoid the former case. If the current condition does not suit overtaking, the fast drone has to decrease its speed to avoid the collision. However, drones that lower the speed will increase their delivery time.

1.3 Problem Statement and Motivation

So the problems here are how we can manage drones to overtake others without any collisions and minimize the time a drone does not fly at its preferred speed or the time it spends out of the tube. Based on that, this project's main objective is firstly to define suitable problem formulation and build a cost model that assigns cost to various events, like drone collisions, deviation from preferred speed, and others. Then we want to design a collision avoidance algorithm for drones flying on a tube-like 'drone road' and implement a simulation-based performance evaluation. The algorithm should be designed for a wireless network environment and consider the effect of packet loss. We would like to understand performance impact of various parameters of the algorithm on of the underlying system, such as noise, drone density, and others. The algorithm is built on drones sharing their position and reacting to upcoming collisions.

For such a collision avoidance algorithm for drones, we intend to make it "coordination-free," which means that only limited information can be exchanged between drones, and more explanation for that is given in Chapter IV. One way to evaluate the performance of our algorithm is to compare it with some baseline algorithms that have very simple strategy such as "do nothing".

The direct benefit of researching this topic is that it can provide a common solution for collision avoidance when drones fly in a straight tube. In another aspect, this topic will provide strong support for further research on collision avoidance and drone overtaking algorithms in a more complicated drone tube network (e.g., a network consisting of many straight, non-straight drone tubes and intersections).

1.4 Research Questions

Based on our problem statement, there are three key research questions that we should be able to answer at the end.

- ***RQ1****:* Can our coordination-free algorithms perform better than the baseline? Do they have any limitations?

- ***RQ2****:* What factors (drone density, noise, etc.) have significant impact on performance, and how do they affect it? Can we explain that?

- ***RQ3****:* Can we find a "genie" solution (i.e. guarantee no collision) for the problem?

1.5 Methodology

1.5.1 Cost Model

The first step is to turn the problem into an optimization problem. Then we need a cost model. Obviously, collision count is one crucial parameter in our cost model. However, it is still not a considerable method if our algorithm achieves zero collision but double the distance each drone traveled. In fact, there are some other parameters that we need to examine, such as the time for flying out of the tube, the deviation from the preferred speed and others. Each event that is related to those parameters in our scenario will be addressed a cost, and the sum of those costs can reflect how good our algorithm is.

1.5.2 Algorithm Design

Besides the cost model provides our algorithm's quantified performance, we also need a range of baseline algorithms to compare the result. These include:

- One naive algorithm in which drones always keep the same speed and never switch lanes.

- One always slow down algorithm in which drones always choose to slow down when they detect an incoming collision.

Hence, we can learn whether our method can save a large amount of cost than do-nothing algorithm and whether it can find a good balance between collision cost and other types of cost.

1.5.3 Implement System Model and Algorithm

We will use the simulation method to observe and analyze the behavior of drones. We use OMNeT++ [27] for setting up the simulation framework, and it works based on C++ and NED. Any model used in this project, such as drones, the communication channel, the wireless module for each drone, and our algorithm, etc., will be coded in a C++ class file which describes the parameters and behaviors of that model. To combine these models to a network, we use several .ned files to describe the relationship of those (e,g. one .ned file can be used to specified that each drone has exactly one GPS device and one wireless communication device).

OMNeT++ is a widely used network simulator which comes with a large library of pre-defined components for various protocols, channel model, etc. Moreover, many basic experiment results are given by OMNeT++ (such as fast fading and free space path loss) fully comply with the theory [27, 28, 29, 30].

1.5.4 Experiment Planing/Design

Many experiments will be needed to test the performance of our algorithm. We need to know which parameters affect the result, whether our algorithm can work successfully in all cases, whether some drones take unnecessary or unpreferred actions, and many other questions. We will control a number of variables to design those experiments. The variables include:

- Drone density
- Tube length
- Transmit power
- Beacon interval
- Path loss exponent.

For those variables above, we define a set of value for each of them and generate an experiment plan that can give enough information with a feasible number of experiments.

Since some of the parameters are generated by random distributions (e.g., maximum speed for each drone, bit error, and initial drone position), we will repeat one experiment many times to ensure the accuracy of the result.

1.5.5 Data Collection and Evaluation

OMNet++ supports recording data during the experiment. Data we collect that way are:

- collision times
- speed variance for each drone
- position variance for each drone
- packet loss counts

We use that data to calculate the cost for each component of the cost model and the total cost. Those costs are the key observable for evaluating the algorithms. Other types of information, such as number of collisions happening in each second, and the average speed deviation, can also help us to analyze the results from a different perspective. A python script will summarize the data and generate several figures to show the result. The figures indicate the relationship between cost and some variables. We evaluate the performance mainly based on those figures. We are interested to know in which cases our algorithm performs badly(total cost is high) and how and why that is.

1.6 Structure of Thesis

Our thesis is structured as follows:

2. **Background and Related works**, chapter II introduces the background knowledge required in this thesis. This includes fundamentals of wireless communications and wireless channel models. Furthermore, we provide an overview of related works.

3. **System Model** clarifies the boundaries of this project. It introduces the system we are considering, and gives details about different system parts.

4. **Criteria-based Switching/Overtaking Algorithm** mainly provides an explanation of our collision avoidance algorithm. In addition, it presents the design of other contrast algorithms.

5. In **Simulations and Results** chapter, we introduce how our simulation is designed and present the simulation result, then give explanation of it. The last section shows an interesting phenomenon found during the experiment.

6. **Discussion** presents the interpretation of our results and list the limitation of this project. We also suggest some worthwhile paths for future research based on these limitations.

7. In **Conclusion**, we answer our research question and summarize our findings.

Chapter II

Background and Related works

In this chapter, we introduce the basic knowledge related to this thesis. It includes drone and drone collision, IEEE 802.11, and wireless channel model. In the last section, we also present some related works that discuss packet loss.

2.1 Drone Types

Drone has two main types [31], which are Multi-Rotor Drones and Fixed-Wing Drones. Multi-Rotor Drones are called that name since they have multiple motors [32]. The most common type of Multi-Rotor Drones is quadcopters, other types like tricopters and hexacopters can also be found in the market. The biggest advantage of Multi-Rotor Drones is that they have excellent maneuverability, which means they can move up and down vertically, move front to end, and rotate in their own axis. So they have good control of aircraft when flying. Fixed-Wing Drones work like the aeroplane, they have one rigid wing. Based on that design, they need not consume too much energy once it has been launched. However, the extent of changing the moving direction of this type is quite limited. Other types of drones exist such as Single-Rotor Drones and Fixed-Wing Hybrid VTOL, more detail see [33, 34, 35].

In this thesis, we have demand for good maneuverability of drones. Drones need to have the ability to fly forwards, backwards, and laterally. Hence we only consider Multi-Rotor Drones in this thesis.

2.2 Drone Collision

Drone technology has developed at a fast pace over the last years, and the number of drones deployed has increased. However, that causes more incidents involving drones happened, such as drones intruding airports or two drones colliding with each other [1]. According to Graham's report [36], the general process of a drone mission can be separated into four steps: take off, cruise, approach, and landing. Moreover, during those steps, there are four potential factors: human factors, environment issues, equipment problems, and organization issues, which can lead to drone collisions.

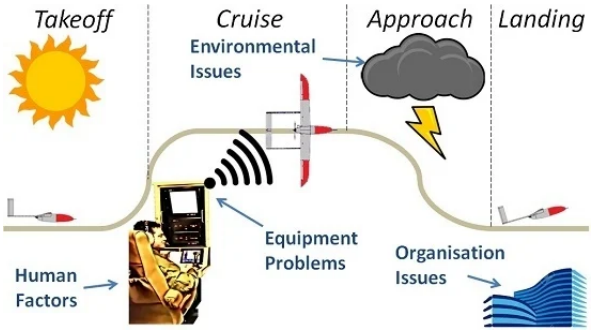

Figure 2.1: Summary of drone collisions [1]

In our scenario, where we only consider drones cruising in the drone road, organization issues are the only factor we should be concerned about. In other words, we need to organize drone behaviors or their decision-making in such a way that no collision happens.

2.3 Wireless Communications

Although we use parcel-delivery as an example of the missions that drones can take in the future, drones may have many different types of jobs. Whatever their tasks are, they should share one drone road and follow the 'traffic rule.' In addition, those drones may not be owned or controlled by one or two entities. In the worst case, each drone may belong to a different company. Hence drones may be equipped with different types of sensors, may use different controlling methods, or even have a different physical structure.

That leads to a challenge about wireless communications for them. At least, drones have to exchange the most basic safety information with each other to avoid accidents. This section will introduce a common technique for drone wireless communications.

2.3.1 Wireless Ad-hoc Networks

Wireless ad-hoc networks were firstly introduced by Chai-Keong in 1997 [37]. They are a decentralized system in which communication does not rely on pre-existing infrastructures such as routers or access points. Instead, each node in this network can forward packets to others [38]. Nodes can freely join or leave the network, which also means the network connectivity and capacity are time-varying. A new node that no one knows is able to join the network without manual configuration. Due to this, different drone companies need not agree and allow to one large network configuration, and then any new product drone can link and share information with others.

Vehicular ad hoc networks (VANETs) are an application of wireless ad-hoc networks, mainly used for vehicle-to-vehicle and vehicle-to-roadside communication [39]. The purpose of using VANETs for the ground vehicle is mainly to ensure road safety. One example are self-driving cars, VANETs allow them to build a dynamic network to receive other cars' position and velocity information [40].

Similar to the above scenario of self-driving cars, a system with multiple drones also demands exchanging key safety information, such as position and speed. A key challenge for either cars or drones is that they cruise at high speed. So the network members change very frequently and any action such as a change of direction or speed should be broadcast to others as quickly as possible.

In 2017, Khan introduced a new application of ad hoc network that works for drones, called Flying ad-hoc networks (FANETs) which can link a small group of drones in an ad-hoc manner[41]. Other than the high cruise speed problem we have talked about, lack of central control, self-organizing and ad-hoc nature between the UAVs are also the main concerns that Khan has considered.

In this thesis, we adopt a vehicular technology (11p) instead of FANETs we talked above, as this is well-developed and serves comparable applications. Furthermore, no dedicated wireless technology has merged for FANETs yet.

2.3.2 IEEE 802.11p Introduction

The IEEE 802.11 protocol [42] is a set of standards for wireless local area networks. It is designed and maintained by the Institute of Electrical and Electronics Engineers (IEEE). That protocol consists of a set of physical and media access control layer protocols.

The IEEE 802.11p amendment [43], first published in 2010, aims to add a standard for wireless access in vehicular environments (WAVE). In such an environment, network transmission should be completed between two vehicles at high speed or between a cruising vehicle and a roadside infrastructure.

In that protocol, data transmission between high-speed vehicles will no longer need to wait for association and authentication. In other words, every node in the network can simply send and receive data packets with others without establishing a basic service set (BSS). Packets can directly be sent to the destination without wasting time for other handshake packets. Hence vehicle and roadside infrastructure can acquire important safety data more quickly.

There are other features that IEEE 802.11p has introduced. One example is a new time management frame that can help devices synchronize clock time. Orthogonal frequency-division multiplexing (OFDM) is applied as the passband modulation, transmission should happen in the 5.9GHz band with a 10MHz bandwidth (typically). Then this set of standards can achieve an approximate 1000 meters' commutation range [44], which is generally enough for moving vehicles.

However, there is no specified 802.11 amendment designed for drones or FANETs. 802.11p as introduced above can be a choice for that, many references [45, 46, 47] have tested the performance of 802.11p, and the result shows 11p can be able to ensure the communication requirement for drones. Also, other 802.11 amendments like 802.11n [45, 48] or 802.11ac [49, 50] can be an option anyway.

2.3.3 Beacon

Beacon is a broadcast form of packet that each node sends periodically (around every 100ms) to their local neighborhoods. For example, in 802.11 [51] beacons are transmitted by the access point to synchronize the network members and to declare the existence of the network. In our thesis, we use the word "beacon" to represent the packet sent periodically by a drone which includes position and speed information to ensure the most basic safety of our drones, e.g., to learn about positions of neighbors and avoid collisions.

In VANETs and FANETs [39, 41], any nodes should set beacon packets to share safety information to maintain the safety coordinately. For the nodes that are not identified by others, the main purpose of sending the beacon packet is to announce its presence. Other nodes can acquire the knowledge of foreign ones and share their neighbor information with them. For the nodes that have already known each other, the beacon packet is used to update the safety message (position, velocity, etc.).

Due to the high speed of cars and drones, an accident can happen within a very short time. To ensure the timeliness of the safety information, beacon packets should be sent frequently by nodes, a reasonable and common value for that is 100ms [52]. Besides that, beacons should have a higher priority over other types of packets, which means if multiple packets are waiting for transmission, the beacon should be the first one. Moreover, to increase the communication range so that farther nodes can receive the beacon, increasing the transmit power is a general solution [53]. However, this leads to a problem: simply increasing the transmit power or decreasing the beacon period may not have a positive effect on the system since this increases interference and channel competition. Interference is the key problem, when the transmit power for all beacon packets is increased, then those packets with a high transmit power may interfere with each other, and hence the receiver will experience a high packet loss rate.

In our drone parcel delivering scenario, there is no requirement for other types of packet besides beacon, so we ignore other types of packet. However, in references [53, 54], the authors give the solutions of congestion problem, such as Adaptive beaconing schemes or even a beacon-less system. In adaptive

beaconing schemes [53], transmission rate or frequency, transmission power, and contention window size should be individually controlled to adapt to the current congestion situation. It uses some parameters such as packet loss rate, the number of neighbors, and channel quality as a reference to adjust the value of those controlled variables mentioned above. The purpose of the adaptive beaconing schemes is to reduce the congestion of beacon packets and minimize the delay of transmissions for every node. In the beacon-less system [54], the nodes do not receive the same beacon packet many times anymore. In the normal system, a beacon sent by one node will be relayed by all neighborhoods to make sure 'everyone' can be able to receive this packet. Differently, beacon-less routing chooses one 'optimal' node to relay the packet by calculating the Dynamic Forwarding Delay (DFD) through the position information for each node. When the optimal delay is chosen, other neighbors can recognize this knowledge and cancel the incoming transmission of that packet. Hence, the number of packets transmitted simultaneously will be significantly reduced. Then it is possible to increase transmit power to ensure transmission quality, and packet interference can be avoided to some extent.

2.3.4 IEEE 802.11p MAC

Media Access Control (MAC) is the layer used to control the hardware interaction with the wired, optical, or wireless transmission. This section will introduce what the MAC packet looks like and how the packet is transmitted inside the MAC layer.

Like any other layer, the MAC layer has its own packet format to exchange key parameters used in the MAC layer. A general MAC packet has the following fields [55]:

- Frame control, which is used to store several control flags. In detail, it always includes three fields they are (1) Protocol Version (2) Type and (3) subtype. Those provide the basic information of this MAC packet. And other fields (e.g., To-DS, From-DS, More-Fragments, Retry, Power Management, More Data, Protected Frame, +HTC/Order) could also be contained for fragmentation, power management, and others.

- Duration/ID is mainly used for three purposes: Virtual Carrier Sense, Legacy Power Management, and Contention-free Period. For the Virtual Carrier Sense, values in Duration fields are presented in microseconds, and it means the duration required to complete a whole frame transmission procedure. Other nodes will listen to this value and will not send packets for the reserved time. This field is set to an identifier for power management and will not be used for duration anymore. Similarly, if this field is used for the Contention-free Period in Point coordination function (PCF), the value should be a fixed value 32768 [55].

- Address fields indicate either unicast address or broadcast address. There are four fields in total, but typically just three of them are allocated in most cases, the last field only used in the presence of a Wireless Distribution System. Address fields contain the transmitter address, receiver address, basic service set identifier (BSSID), destination address, and source address, depending on circumstances.

- Sequence control has two sub-fields: sequence number and fragment number. They can together allow the receiver to detect packet duplicates.

- QoS control lists the parameters (e.g., traffic identifier, End of Service Period, and ACK Policy) used in Quality of Service for stations.

- HT Control indicates the control parameters (e.g., Link Adaptation Control, Calibration Position, and Calibration Sequence) for High Throughput that is relevant to the physical layer.

- Frame body, or named data field, is used to store the actual data the transmitter needs to send.

- FCS means the frame check sequence, just like the checksum in other types of packets, is used to check the integrity of the entire packet.

The distributed coordination function (DCF) [56] is the most fundamental channel access technique in IEEE 802.11. It enables carrier sense multiple access with collision avoidance (CSMA/CA) and binary slotted exponential back-off to provide the fundamental mechanism and access method for packet transmission in the MAC layer. In fact, other techniques have the similar purpose, such as point coordination function (PDF) and the extended version EDCA and HCCA in 802.11e.

The main procedure for DCF is shown in Figure 2.2 below. Inter-frame spaces (IFS) are the key way to enable a priority mechanism for different packets by using different waiting times. Actually, four time frames are shown in the figure, which are Slot time, SIFS, PIFS, and DIFS. In fact, ten of these time frame existed, for more details see [57]. The slot time is the unit time for the whole timeline. The value is set to 9 μs in the OFDM-PHY. Signal propagation and physical carrier sensing happen within the slot time.

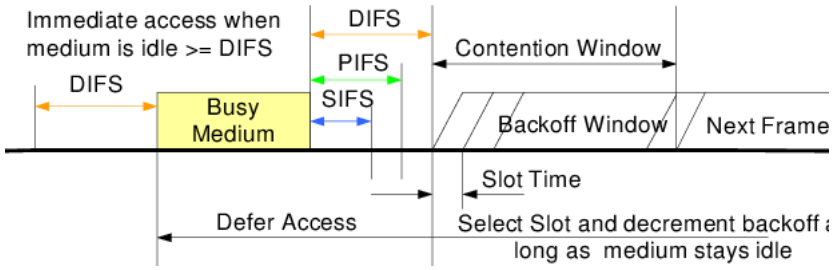

Figure 2.2: DCF timeline and parameters [2]

Carrier sensing is used to ensure the absence of other transmissions for avoiding packet collision. For physical carrier sensing [58]. Firstly the sender starts listening to any signal for a short amount of time. Then if no other signals are detected, the sender concludes that the channel is idle, otherwise it is busy. Actually, IEEE 802.11p uses not only physical carrier sensing but also virtual carrier sensing [59]. It can be done by overhearing packet transmissions in the neighborhood. If some packets are detected, it will extract the duration value to a parameter named network allocation vector (NAV). The sender then starts a counter to decrease NAV. During this process, if a new packet has a larger duration value than the sender's current NAV, then NAV should be set to the larger value. NAV is zero means there is no transmission now so

that the channel can be seen as idle, otherwise as busy. Combining physical and virtual carrier sensing, the channel is idle only if both two types of sensing result in idle.

The short inter-frame space (SIFS) is typically set to 16 μs. It is used to accommodate hardware activities such as switching mode from transmit to receive. Also, ACKs should be started to transmit within at most SIFS time. We can calculate other time frames using SIFS plus integral times of slot time. PCF inter-frame space (PIFS) exactly equals SIFS plus slot time, which is 25μs. Beacon packets use this time frame to show high priority over other normal packets. Finally, DCF inter-frame space (DIFS) is defined by using SIFS plus double slot times, so that it equals 34 μs. It is the time delay for waiting for the backoff to be completed. Backoff is a process to avoid packet collision. The value of backoff time is random, and it is likely to become larger after each retransmission. The detailed backoff procedure is:

- When someone wants to transmit a data frame or management frame and carrier sensing results in a busy channel, it starts to wait for the DIFS time.

- After that, it needs to wait additional backoff time, which equals slot time times a random integer. The domain of the random integer is $[0, CW]$, and contention window CW is initially equaled to CW_{min} which is defined in configuration.

- Retransmission will happen when CTS or ACK is not received after RTS or data frame respectively. The retransmission times are limited to a configurable value.

- After each retransmission, CW will be doubled if the doubled value is smaller than CW_{max}, or set to CW_{max} otherwise. CW_{min} is also a pre-defined parameter.

- After the transmission either succeeds or reaches the maximum retransmission times, CW will set to CW_{min} finally.

Recall that the DCF enables carrier sense multiple access with collision avoidance (CSMA/CA) and binary slotted exponential back-off. After introducing these two techniques, the whole DCF access procedure can be discussed. Firstly, there are some cases in which carrier sense is not enabled. They are:

- When an ACK packet needs to be transmitted, the sender should send the ACK within SIFS time without using any type of carrier sense
- When a CTS packet needs to be transmitted, and virtual carrier sensing indicates an idle medium. Then the sender will send the CTS packet immediately without using physical carrier sensing (in response to RTS).
- When the sender just receives a CTS addressed to itself, it will start to transmit the data packet immediately without using any type of carrier sensing.

Otherwise, the carrier sense mechanism will be used, where the sensing duration depends on the types of packets to be transmitted. This time the sender firstly starts the carrier sensing. Based on the result, there are three cases.

- First case, the carrier sensing indicates the channel is idle and the current back-off timer is zero. Then the sender will start transmission.
- Second case, the carrier sensing finds the channel is busy.
  - The back-off timer is set to a random value. And the waiting time for this stage is the back-off timer times the slot time.
  - The sender waits for the channel to become idle for the sensing duration.
  - Then the back-off timer start to decrease. And physical carrier sensing still works at the same time.
  - Each time the back-off timer is decremented, the sender checks whether the timer reaches zero. If it is and the channel is also idle,

it starts transmission. If the channel is busy, then the sender suspends the back-off procedure and waits until the medium becomes idle for the sensing duration time.

- Third case, the carrier sensing finds the channel is idle, but the back-off timer is not zero. That case is similar to the second case that the sender still needs to complete the back-off. And during this process, if the channel is not idle anymore, back-off process will also be suspended.

In this procedure, the sender must wait a random time before starting transmission. In this way, if multiple senders wait for the back-off simultaneously, the randomized mechanism can decrease the possibility of having a collision since different senders are possibly drawing a different back-off time. However, there is still a chance that two senders have the same timer, then the collision will happen in this case.

In 802.11p, an advanced version of DCF called Enhanced Distributed Channel Access (EDCA) is used. EDCA focuses on packet priorities ensuring that packets with higher priorities are more likely to have a shorter back-off time than the packets of lower priorities. To achieve this, firstly, the DIFS time is replaced by arbitration inter-frame space for category C (AIFS[C]). Furthermore, the boundary of contention window CW_{min} and CW_{max} will also depend on the access category. The Tables below list the EDCA access category and the default DCF and EDCA access parameters in 802.11p.

User priority	**Access category (AC)**	**Designation**
1	AC_BK	Background
2	AC_BK	Background
0	AC_BE	Best effort
3	AC_BE	Best effort
4	AC_VI	Video
5	AC_VI	Video
6	AC_VO	Voice
7	AC_VO	Voice

Table 2.1: Packet priorities in EDCA

Access category (AC)	CWmin[C]	CWmax[C]	AIFS[C]
AC_BK	15	1023	9
AC_BE	7	1023	6
AC_VI	3	15	3
AC_VO	3	7	2

Table 2.2: Default access parameters in 802.11p

As mentioned before, the DCF/EDCA can help decrease the probability of packet collisions. But unfortunately, packet collisions are unavoidable, especially for broadcast packets such as beacons. In 802.11p, there are three types of collisions which are internal collision, direct collision, and hidden-terminal collision[60]. Internal collisions happen when two packets with different access classes within one station would start transmission at exactly the same time. In that case, only the packet with the highest priority can be granted transmission, all other contending packets would suffer a collision. Direct collision caused by packets from two stations within mutual range starts transmission at the same time. Moreover, hidden-terminal collision occurs when two packets from two stations that are not in the mutual range start transmission simultaneously, then the signals of these two packets collide with each one in another station between them. Figure 2.3 shows an example of a hidden-terminal collision.

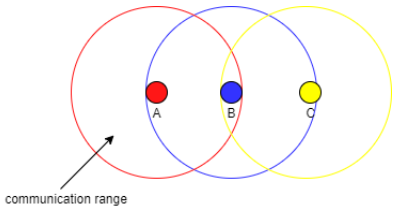

Figure 2.3: Example of hidden-terminal collision

Station A and B, B and C can communicate, but A and C cannot see each other. In that case, A will not be able to indicate Whether C is transmitting the packet or not. Hence the carrier sense mechanism does not work. Then A and C are likely to send packets simultaneously, but B can receive those packets from A and C. The signals of both packets will interfere with each

other, leading to packet corruption. And finally, no packets will be successfully received by B.

2.4 Wireless Channel Model

In the last section, we discussed some background related to the communication between drones, and in this section we will give a brief overview of the wireless channel model, which is essential for the physical layer.

2.4.1 Radio Wave Propagation

Drones rely on wireless communication instead of wired communication. Wireless communication is implemented by transmitting radio waves in free space. The radio wave is transmitted and received by antennas, which are part of our wireless devices. The process of translating a stream of bits into a radio wave is called modulation. Conversely, demodulation converts the radio wave into a stream of bits. Generally speaking, many distortions existed in the channel, such as path loss, fading, etc...

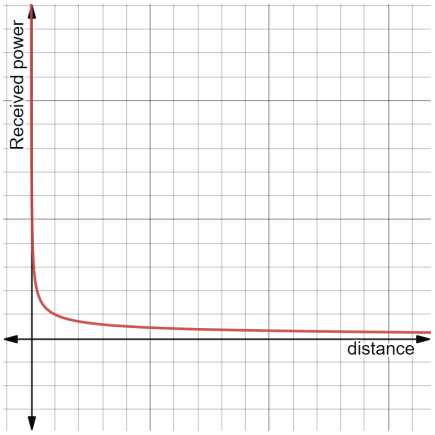

Figure 2.4: Friis Free Space Loss Equation

Path loss is caused by some radiated electrical power getting lost during its propagation, leading to the actual received power being less than the transmitted power. To ensure the demodulator can decode the received signal into the transmitted bit-stream, the transmitter should consider the power

loss and increase its radiated electrical power. Then the received power can achieve the minimal SNR for decoding. A straightforward model, the free space model, states the relationship between the transmitted power and received power over transmission distance if the environment does not affect the wave propagation. The underlying formula is called Friis Free Space Loss Equation [61], which is:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}$$

Where G_t and G_r are the transmitter and receiver antenna gain, respectively, L summarises other loss factors unrelated to antennas, λ represents the wavelength, and d is the distance between transmitter and receiver. This equation indicates a quadratic decreasing relationship between the received power P_r and transmitted power P_t when the distance between transmitter and receiver is raised while other parameters are fixed.

Path loss is the ratio of the transmitted power to the received power, i.e.:

$$PL(d) = \frac{P_t}{P_r(d)} = \frac{(4\pi)^2 d^2 L}{G_t G_r \lambda^2}$$

If we introduced another parameter $PL(d_0)$ which is the path loss at the reference distance d_0, then the path loss equation (only valid when $d \geq d_0$) can also be written as:

$$PL(d) = PL(d_0) \cdot \left(\frac{d}{d_0}\right)^2$$

2.4.2 Unit Disk Model

The nit disk model is a very simple radio model. Whether a packet can be received only depends on the distance between transmitter and receiver [62]. If the distance is less than a parameter called communication range, then the packet loss rate would be 0. Conversely, PLR would be 100%. In this model $PL(d)$ is not needed to be calculated since the effect of path loss is only considered through the communication range. Hence, Unit Disk provides an ideal communication model. If a node stays within the communication

ranges of all other nodes, then packets from others would not be lost by path loss. However, packet loss still exists when a node receives two packets from different nodes at an exactly same time. In that case, a packet collision happens.

2.4.3 Log-Distance Path Loss Model

The Log-distance model is a radio propagation model which only considers thermal noise and the path loss. Similar to free space path loss equation, the path loss PL(d) is:

$$PL(d) = PL(d_0) \cdot \left(\frac{d}{d_0}\right)^{\gamma}$$

We use γ instead of 2 as the exponent in this equation, and it is also called the path loss exponent, which will vary across over different signal propagation environments [63] (e.g., Vacuum, Office with hard partition, and Retail store). Some typical values are listed in the following table. One finding [64] related to path loss exponent for UAV networks in free space is, with the increase of height, the path loss exponent would reduce. In that paper, the simulation result shows γ would decrease from 3.7 (at ground) to 2.0 (at 120m).

Environment	γ
Vacuum, infinite space	2.0
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
In building line-of-sight	1.6 to 1.8
Obstructed in buildings	4 to 6
Obstructed in factories	2 to 3

Table 2.3: Typical values for path loss exponent [6, page 69]

Due to the random movement of the free electrons caused by heat, random electrical waveforms always exist in the signal propagation environment. Those waveforms, named thermal noise, will superimpose on the received waveforms and add some difficulties for the receiver to decode it. One com-

monly used noise model for is white Gaussian noise. If time t is fixed, noise is expressed by a Gaussian random distribution with zero-mean.

The general equation [65] to calculate bit error rate is:

$$BER = \frac{1}{2} erfc\left(\sqrt{E_b/N_0}\right)$$

where E_b is the energy per bit which is proportional to the received power, and N_0 is noise power spectral density. The function $erfc(\cdot)$ [66] is the complementary error function, given by:

$$erfc(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

This is a monotonically decreasing function so that with the increase of E_b/N_0, the bit error rate should decrease. Namely, either increasing E_b or reducing N_0 can help lower the bit error rate.

2.5 Packet Loss in VANETs

In VANETs and FANETs, safety information delivery is crucial to ensure the safety of cars or drones. However, packets containing beacons might get lost during transmission for the many reasons mentioned above. This leads to the problem of controlling the packet loss rate in VANETs and FANETs.

Reference [67] indicates that, under a high car density with a heavy communication load scenario, a large number of packet collisions will happen due to congestion. That leads to a high packet loss rate. Based on that, Rajeswar performed a simulation-based study [3] to explore the performance of 802.11p under different road scenarios, and results are shown in Figure 2.5. Obviously, with a high vehicle density (> 200 per square kilometer), the packet loss ratio for all scenarios is very close to 1. Hence, indicating approximately all local broadcast packets are expected to be lost during transmission.

Since CSMA/CA is the default medium access control mechanism in VANETs in 802.11p, the results above suggest that 802.11p cannot ensure high QoS under a high node density. The authors also give a few methods that can help to control the congestion in this situation, they are:

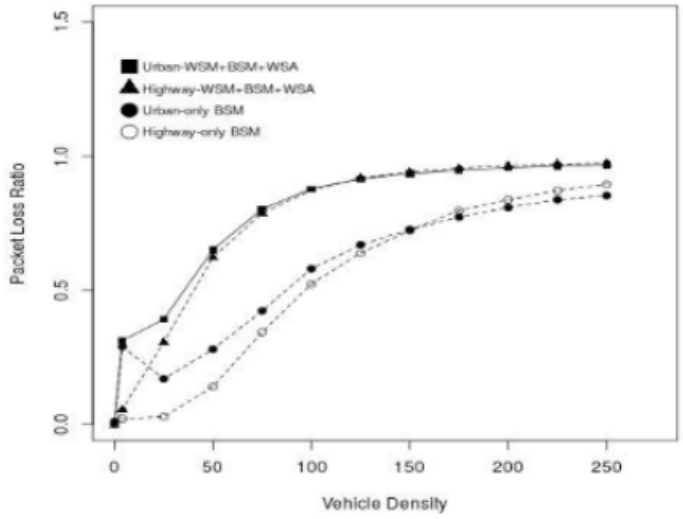

Figure 2.5: Packet loss rate vs vehicle density in different road scenario [3]

- Control transmission rate to reduce the transmission time.
- Lower beacon generation rate to lighten the communication load.
- Reduce transmission power to shrink the contention area.
- Control CW size and AIFS[C] values to decrease the probability of packet collisions.
- Optimize the prioritization scheme to give to the higher priority of safety messages.

Another reference [4] focuses on modeling the relationship between packet loss and vehicle speed. In his scenario, vehicles are assumed to drive at the same speed and the same direction on the road. Vehicles send packets periodically and want to keep a minimum safety distance with others in front. Results in figure 2.6 show that speed can significantly affect the packet loss rate. For example, the packet loss rate of vehicles travelling at 18km/h is about ten times smaller than the packet loss rate in the 90km/h situation.

With a higher speed, the received signal power of the vehicle will be smaller, and therefore, the bit error rate will be larger.

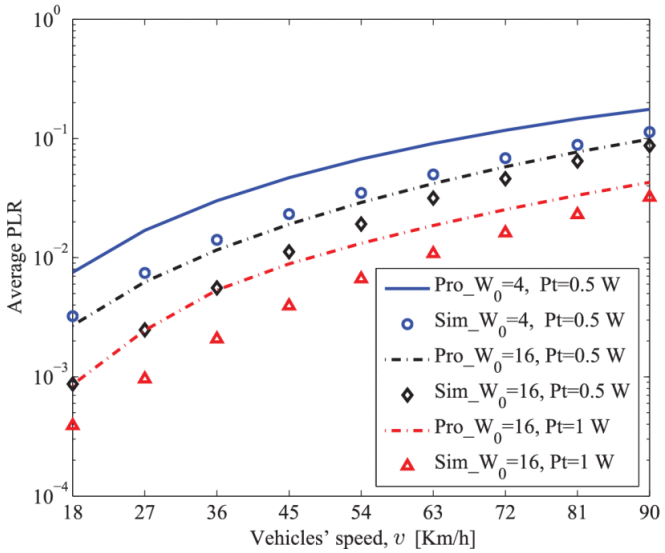

Figure 2.6: Packet loss rate vs. vehicle speed [4]

In addition, this paper also shows the influence of transmit power and contention window size. Increasing CW size or decreasing the transmit power can significantly reduce the packet loss rate. When CW size increases, the back-off timer will be longer, reducing the packet collision rate. Compared to the result presented in [4], large transmit power has a different effect on the packet loss rate in different scenarios. In some cases, increasing transmit power can expand the communication range of the vehicle. Therefore, hidden-terminal packet collisions won't occur since every node can know the existence of all others. However, increasing transmit power also causes more interference between packets. In other cases, reducing the transmit power can shrink packet reception range, leading to decreased channel contention from stations farther away, but it increases the rate of hidden-terminal collisions.

In reference [5], the author presents results showing about the contribution of every type of loss. The scenario is a simple two-way road with vehicles driving in the opposite direction. Results indicate that most packet loss is caused by hidden-terminal collisions, the second is the noise, then is the direct collision.

Based on that, increasing CW size can lower the packet loss rate. But there is a trade-off of it. On the one hand, the hidden-terminal packet collisions decrease. On the other hand, packet loss is caused by channel switching (such as CCH to SCH [68, 69]) increases. Also, the delay(the time a packet waits in the MAC layer) may increase due to the large CW size. Considering about hidden terminal is the main reason for packet loss. And it is hard to reduce the packet loss due to noise. Therefore, using a large CW size can help to increase QoS in this scenario.

Some papers have investigated about controlling congestion by parameter adaptation, such as rate adaptation, transmit power adaptation or CW size adaptation. In [70], rate adaptation is used to lower the packet generation rate when the estimated number of neighbors increases, and vice versa. Another approach is to increase the packet generation rate when neighbors' estimated packet loss rate is high enough. About transmit power adaptation, Andy presents one cross-layer approach [71], which optimizes the transmit power by using the channel state information learned from the PHY layer. For other approaches see [72, 73, 74]. For CW size adaptation, the general idea [75, 76] is to adapt the CW size according to the network load conditions and the instantaneous collision rate. A cross layer algorithm for CW size adaptation is introduced in [77], where MIMO operating parameters of physic layer control CW size.

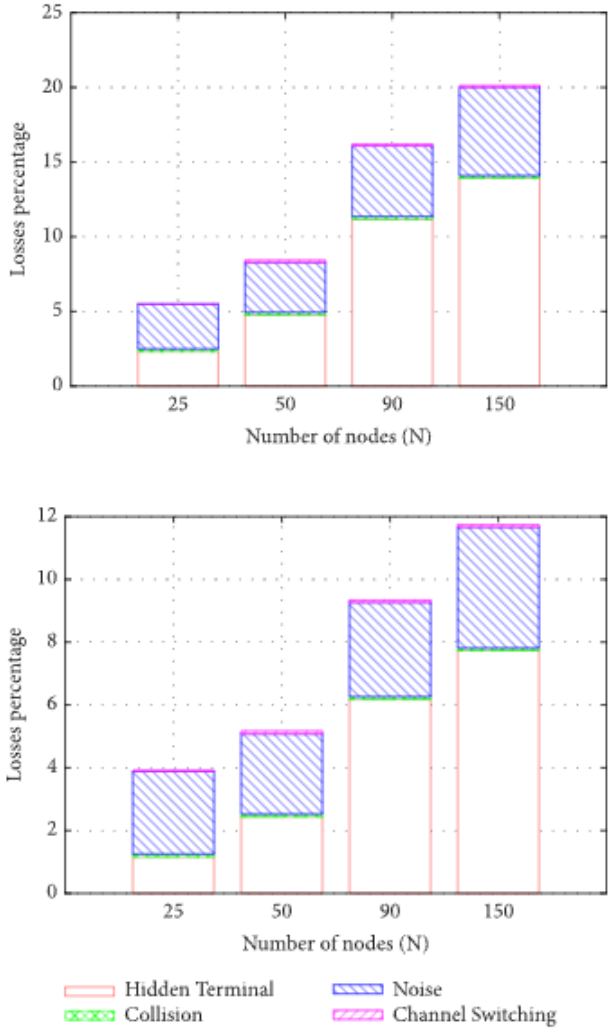

Figure 2.7: Contribution of each type of loss on the total loss when CW=512 (top), CW=1024(bottom) [5]

Chapter III

System model

To clarify the scope and boundaries of this project, we introduce the system model and provide a framework for precisely formulating our research questions in this chapter. The system model includes the drone and drone road, communication, and cost models. We will describe each model in a separate section.

3.1 Drone Road and Coordinate System

The tube is straight and its length is finite. One important assumption is that the section area's size is infinitely large. But we will partition it into several tiers (Figure 3.1), and assign a cost value to each of them by using a cost function. The innermost cycle area will have the lowest cost value of zero, and the cost will increase when the radius of the cycle becomes larger.

Further, the drone tube is partitioned into several lanes (or sub-tubes). We assume that a lane has a sufficiently large diameter to hold any of the considered drones completely inside. And two drones can fly in different lanes in parallel without any collisions. We put the center of each lane on an intersection point of an equilateral triangle network. Figure 3.2 shows the layout of lane distribution. Based on that layout, one lane would have the same distance to all its six neighbors, and the relative positions of those six neighbors to the 'center' lane are always the same. Therefore, once we know the position of a lane, we can then easily calculate the coordinates of its neighbor lanes.

We build a 3-dimensional coordinate system for the drone road. In this coordinate system, the value for the y-axis represents how far a drone has cruised from the start. If that value is larger than the tube length l, then the

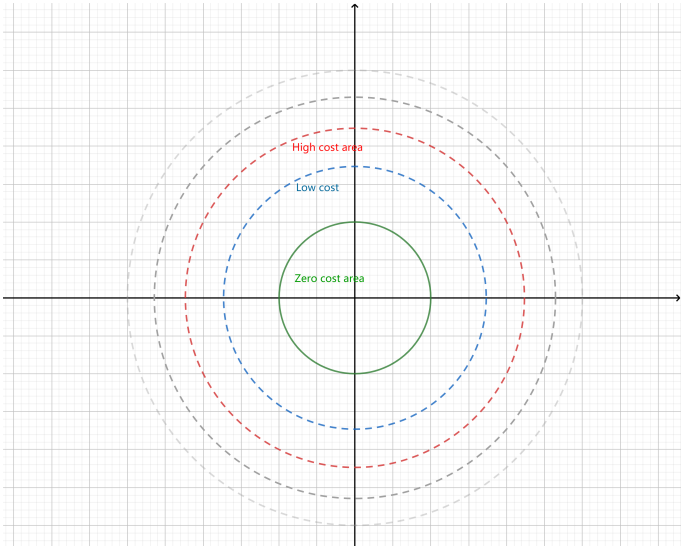

Figure 3.1: Section area of tube

drone with this y value is considered as having reached the destination. The values for the x-axis and z-axis together represent the drone's position in the section area.

Assume the side length of those equilateral triangles is r, and the 2-d coordinate of one tube is (x, z). Then neighbors' coordinates are:

- $(x + r,\ z)$ for the right neighbor
- $(x - r,\ z)$ for the left neighbor
- $(x + r/2,\ z+\sqrt{3}r/2)$ for the top-left neighbor
- $(x + r/2,\ z-\sqrt{3}r/2)$ for the bottom-left neighbor
- $(x - r/2,\ z+\sqrt{3}r/2)$ for the top-right neighbor
- $(x - r/2,\ z-\sqrt{3}r/2)$ for the bottom-right neighbor

Lanes would have different tier levels based on the distance to the center of the tube. The lane, whose center is precisely on the tube's center, is defined as on tier 0. And lanes which are the neighbors of the lane on tier 0 is on tier

1. Similarly, lanes which are not on tier 0 but are the neighbors of tier 1 lanes is on tier 2, and so on. Except tier 0, the convex hull generated by centers of all lanes on the same tier is like a hexagon, and the side length of such a hull is r times tier number minus one. Hence, we can compute how many lanes on a given tier level, the relationship between the number of lanes l_n and tier level n is shown below:

$$l_n = 6 * (n - 1)$$

Based on this layout, there is one important problem. Assume drones know their own position, represented by 3-d coordinates. And they are always on the center point of the lane. How can drones determine which lane they currently cruise in? In addition, if drones acquire the lane information (such as tier number) of one neighbor drone, how they can calculate the related 3-d coordinates. To solve this, we first need to assign a unique identifier for each lane and then provide an approach for identifier-coordinates conversion.

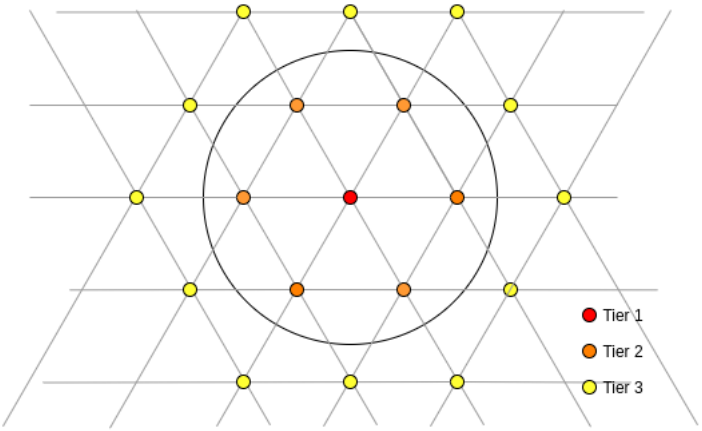

Figure 3.2: Layout of sub-tube

Recall that the center lane is at tier 0. Then we define two two basis vectors, they are:

- $e_1 = (r, 0)$
- $e_2 = (r/2, \sqrt{3} * r/2)$

Then any valid lane center-point can be uniquely represented as $i*e_1+j*e_2$ for some (signed) integers i and j. From here onward, it is much simpler to denote a lane by its (i, j) pair. Knowing (i, j), it is trivial to calculate the coordinates of the center point of the lane, namely from $i * e_1 + j * e_2$. To generate all the lanes/center points of tier n, it has to include the points:

- Corner points: (n,0),(-n,0),(0,n),(0,-n),(n,-n),(-n,n)
- The points (x,n-x) for x in 1 .. n-1
- The points (-x,-(n-x)) for x in 1..n-1
- The points (-x, n) for x in 1 .. n-1
- The points (x, -n) for x in 1 .. n-1
- The points (n, -x) for x in 1 .. n-1
- The points (-n, x) for x in 1 .. n-1

Conversely, to work out to which tier a given point (i,j) belongs, the following is used:

- If i == 0 then tier = abs(j)
- If j == 0 then tier – abs(i)
- If sign(i) == sign(j) then tier = abs (i) + abs(j)
- Otherwise tier = max (abs(i), abs(j))

This kind of layout of lanes has several benefits for designing overtaking algorithms. Based on the knowledge of identifier-coordinate conversion and neighbors' position calculation we discussed above, a drone can conveniently detect the neighbor lanes' identifier. Since lower tier number means lower cost in most cases, drones can use tier identifier as a parameter to minimize the cost value when they need to select another lane to avoid risk. Moreover, the

distance between two neighbored lanes is always the same, so when selecting a lane for overtaking, drones need not consider which lane is closer or farther.

Finally, there are another two assumptions. First, this layout is fixed in all of our simulations, and drones are assumed to know all the above calculations. And secondly, since the length of the tube is finite, any drone arriving at the end will be considered to have arrived at the destination. Hence, that drone will not be involved in the calculations and simulation anymore.

3.2 Drone

The drone in this project does not have a physical shape and size. We treat it as a point. A collision happens when the distance between two drones is smaller than a d_{min}. Drones are only equipped with GPS sensor, which means they can only identify their position information. We treat that information as x, y, and z-axis values in a 3D coordinate system, respectively.

In the figure 3.3, point A and point B represent two drones, whereas d is the euclidean distance between them (In other words, $d = ||A - B||$). If d is smaller than d_{min}, then a collision has happened between drones A and B.

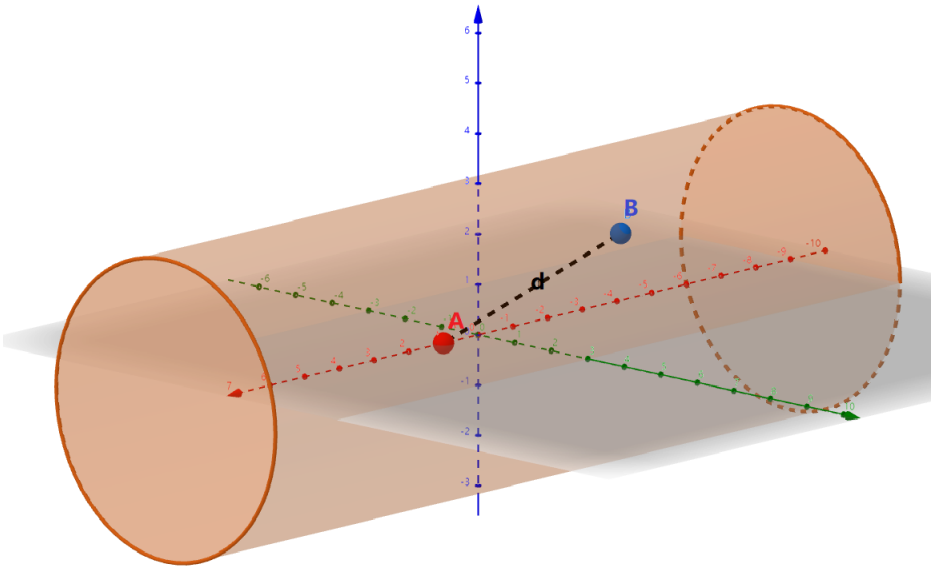

Figure 3.3: A situation when two drones are in the tube

Each drone has its own maximum speed and preferred speed. The maxi-

mum speed represents the highest speed that the drone can fly in the tube, and the preferred speed is a cruise speed which is smaller than the maximum speed. The maximum speed for each drone is a constant whose value is defined as thirty meters per second, and the preferred speed depends on the maximum speed. It follows a uniform distribution, for example, between 70% and 90% of the maximum. If the drone does not fly at the preferred speed, it incurs some cost which depends on the difference between the current speed and preferred speed.

Drone can change their speed and direction for overtaking. It carries out those actions by its own decision. No other infrastructures and devices are used to control them. When changing speed, the acceleration process will be seen as instantaneous. That means if a drone decides to increase its speed from 10m/s to 20m/s, the speed will directly jump to 20m/s. Exploring a more realistic drone dynamics model is an aspect of future work.

Recall that drones typically cruise in parallel to the y-axis. Therefore, an important assumption is that maximum speed and preferred speed parameters are only for the y-axis. For the x and z-axis, we use another parameter named $v_{overtake}$ to restrict the speed for lane switching. In other words, during a lane switching action, the speed of a drone at the x and z-axis should follow:

$$\sqrt{v_x^2 + v_z^2} = v_{overtake}$$

Then the overall speed on all axes is:

$$v = \sqrt{v_{overtake}^2 + v_y^2}$$

Drones are generated using a periodic process, and the amount depends on the drone density. In the experiment, the generation interval $t_{generate}$ and the number of drones generated each time $n_{generate}$ are two critical parameters to be varied. When a drone is newly generated, its position will be at the start of the tube, and it will be randomly placed at one lane in the section area. In addition, when generating drones, we need to ensure that they do not already collide with other drones (i.e., the generation process should generate a maximum of one drone per lane).

To generate drones for simulation, we have a drone generator. Once a drone is set up, it is asked to fly to the end of the tube in parallel to the y-axis. During this process, drones continuously send their beacon packets and receive others packets to exchange their safety information to avoid collisions. There are two cases in which a drone will be removed from the simulation environment. The first is when it reaches the end of the tube, and the other one is it suffers a collision. In both cases, the removed drones disappear immediately. They will not be able to intervene (e.g., send beacons) in our environment any further.

3.3 Communication

Communication between drones is wireless and based on the IEEE 802.11p protocol. In this project, we will introduce a coordination-free algorithm which means drones only share position and velocity information with others. Anything like an overtake plan is not transmitted. Based on that, drones need to learn information by exchanging beacon packets and make overtaking decisions based on their learned knowledge.

For the most basic safety use, the beacon packets need to contain at least:

- timestamp
- identification
- neighbors identification and safety information
- own safety information with timestamp

Since neighbor information are included in the beacon packet, one node might receive many repeated packets that include another one's information. We use the timestamp to identify the newer information to solve this problem. When a node receives a beacon from another, it must check the timestamp before updating its table. Identification is the unique mac address to specify the identity of one drone. And safety information includes the position and velocity data. Both of them are 3-d vector.

The traffic model of this project is quite simple. The beacon packet (namely, safety message) is the only type of packet being transmitted. Beacon packets are broadcast, and packet collisions may happen. Each drone will send a beacon packet periodically (but with some random jitter such as drawing from a uniform distribution). When the next beacon arrives but the previous beacon hasn't been transmitted yet, we simply drop the previous beacon and try to transmit the new beacon instead. The time interval of sending a beacon packet should be short enough to ensure other drones can get the information timely. We will vary it in our experiments. When a drone receives a beacon packet, it will add the sender and all sender's neighbors (within a certain distance) to its own neighbor list, and save or update the information in this packet. If a drone does not receive the beacon packet sent by a neighbor within a time period, that drone will not be considered a neighbor anymore and will be removed from the list. Based on the beacon packet, drones have the ability to know the information (e.g., speed and position) of others to plan its further operation.

We consider the impact of physical and MAC layers in wireless communication (introduced in Section 2.2.4 and 2.3). Packets may be lost by packet collisions or bit errors during the transmission. We detect packet loss by comparing the sequence number between the received beacon. If the gap of the sequence number is not one between the two newest received beacons for one neighbor, then it indicates that some packets have been lost. We use two radio models in our project: scalar radio with log-distance path loss model and the unit disk model. For the former one, the transmit power will encounter path loss which means a longer distance between sender and receiver leads to a higher probability of packet loss (caused by the bit errors) [78]. For the unit disk model, communications are guaranteed[1] if the distance is less than one parameter called communication range [79].

3.4 Cost Model

We define the cost during an experiment, including the collision cost, the speed cost, and the position cost. The total cost is defined by the following

[1] PLR is 0 for distance smaller than range and 100% for larger distance

equation, where τ_1 and τ_2 are the weight parameters for different types of cost.

$$C = C_c + \tau_1 C_s + \tau_2 C_p \quad (\tau_1, \tau_2 \geq 0)$$

.

The collision cost C_c stands for the total cost caused by all collisions in one experiment. It is

$$C_c = c * n_{collision}$$

,

where c is the cost for one collision and $n_{collision}$ is the number of collisions observed over the simulation time. Since we expect no collision to happen, the value of c is set to a very high value.

Speed cost is incurred by flying at a different speed than the preferred speed. Therefore, the relationship between speed and cost should be that the cost increases when the difference between the current speed and the preferred speed increases. One example is the power function, and it is shown in the Figure 3.4 below.

The parameter α is the power coefficient which equals to 2 in our project, so our speed cost function is:

$$f_s(v_{current}, v_{preferred}) = \tau_1(v_{preferred} - v_{current})^2 \quad 0 \leq v_{current} \leq v_{max}$$

Hence, larger differences between current cruise speed and preferred speed will be penalized much more, where a small range of speed change just has a small impact on the final cost.

Speed and position information are recorded periodically (every 100 milliseconds) during the simulation, and when a node is dropped, the recording action for this node will stop. Hence, the different nodes may have a different amount of the recorded speed and position values. We use t_k to represent the total amount of the recorded value for node k. In this way, the first speed information we collected for node k can be written as v_{k_1}, the second is v_{k_2}, and the last speed should be $v_{k_{t_k}}$.

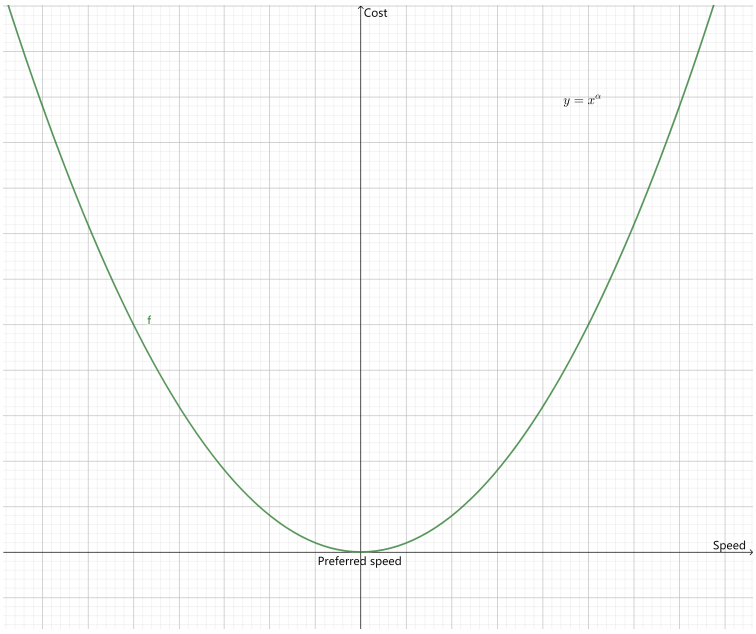

Figure 3.4: An example power function for calculating speed cost

For one point (x,y) in the graph 3.4, the value of y stands for the cost of a drone flying at x speed per second. So assume we have n drones marked as 1 to n, respectively. If we know the speed set and the preferred speed (represented by V_k for node k, for example) for each node, then the speed cost can be expressed as

$$C_s = \sum_{i=1}^{n} \sum_{j=1}^{t_i} f_s(V_i, v_{i_j})$$

Drones produce position costs when they are out of the preferred area. The position cost will be zero when drones stay within the tube (In our scenario, they should be at lanes in tiers 0 and 1). And otherwise, the cost will increase with tier number. Figure 3.5 shows an function for calculating position cost.

We have introduced the calculation for converting coordinates to tier numbers. Thus, we can infer the tier information for drones easily. Assume ti represents tier number, then our position cost equation in this project is defined by:

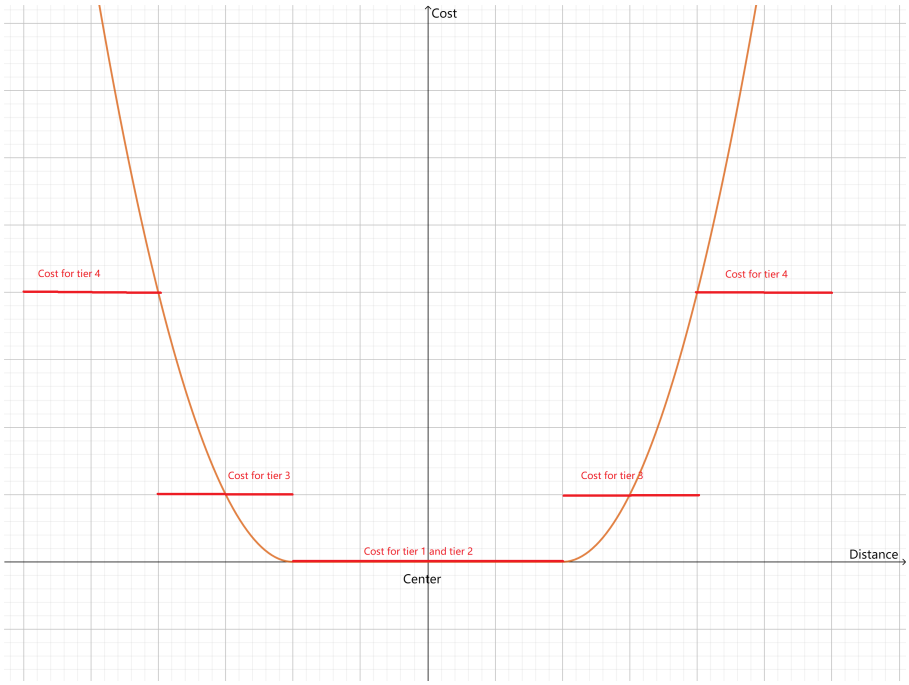

Figure 3.5: An example function for calculating position cost

$$f_p(ti) = \tau_2 * max(0, ti - 2)^3$$

In our scenario, despite the drone being allowed to be out of the tube, it pays an enormous cost for doing that. Therefore, the exponent coefficient in the position cost function is set to 3 to penalize drones that are far away from the center of our tube. So, similarly, if parameter t_{i_j} represents the j-th collected tier number for drone i, the total position cost is:

$$C_p = \sum_{i=1}^{n} \sum_{j=1}^{t_i} f_d(t_{i_j})$$

The cost model is designed to express the result of our algorithm numerically. Therefore, we need to lower the cost as much as possible to improve performance. To achieve that, we should consider lowering each part of the total cost, especially the collision cost, since we are not supposed to see any collision happen.

3.5 Key Parameters

Table 3.1 introduces key parameters and the related value we used in our project.

Parameter	**Default value**
radius of the tube for zero cost area	2.5m
minimal safety distance	1m
drone generate interval	1s
preferred speed distribution	uniform(0.7, 1)
preferred overtake speed	1m/s
maximum speed	30m/s
bitrate	12Mbps
noise power for channel models	-90 dBm
backoff times in carrier sensing	7
cwMin	31
cwMax	1023
sifsTime	10us
slotTime	20us
Collision cost coefficient	1000
Speed cost weight	1
Speed cost function coefficient	2
Position cost weight	5
Position cost function coefficient	3
length of one entry of safety message	144B
length of beacon header	8B

Table 3.1: Key parameters in system model

Chapter IV

Criteria-Based Switching/Overtaking Algorithm

In this chapter, we discuss our criteria-based algorithm. Section 4.1 introduces the difference between drone overtaking and ground vehicle overtaking and then gives the basic overview of how drones switch lanes in our scenario. Section 4.2 and 4.3 present two baseline algorithms for comparative research. Section 4.4 and 4.5 introduce the idea of our criteria-based algorithm and give several sets of criteria that can make a significant difference in drone behavior.

4.1 Overview

Overtaking for a drone is different from a vehicle. The most significant point is overtaking direction. A ground vehicle should drive on the ground, which is a plane. So the position vector is two-dimensional. But the drone is flying in the airspace, and its position vector is three-dimensional. Hence, if a ground vehicle moves forward, it will have a one-dimensional direction space (left or right) for overtaking. But drones will need to operate in two dimensions. Besides turning left or right, drones can also higher or lower their flight altitude to avoid collisions (assume the tube is straight). Then the drone theoretically should have infinite choices for choosing an overtaking direction, while the ground vehicle only has two. Our project separates the tube into several non-overlapping lanes to limit the choices and simplify the scenario, for more detail see Section 3.1. Designing the strategy to select a lane for overtaking is a crucial point for the algorithm to avoid collisions.

For the scenario of this project, one important assumption is that drones can fly out of the tube, and the actual moving space is infinite (but with penalties). Based on that assumption, the drone will sometimes face a choice

to either slow down its speed to wait or fly out of the tube to overtake the front one (a simple example is shown in Figure 4.1). This is a 2-D example. The drone flies at 30m/s, it cannot turn left to avoid the collision since another drone with a slower speed is currently in that lane. Switching to that lane will cause another collision with this slower drone. So the faster drone faces a choice to slow down or use the right lane (it is out of the tube) to respond to the incoming collision.

The different decisions will lead to different amounts of position or speed costs and will change the future topology. In addition, sometimes, the cost of avoiding a collision may be higher than allowing it to happen. The purpose of our algorithm is to try to prevent collisions as much as possible. This project does not consider predicting all future states to optimize overall cost or balancing the trade-off between collision and overtaking/switching.

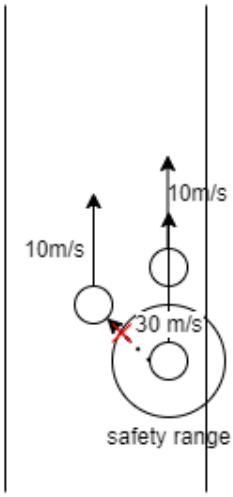

Figure 4.1: An example of a drone facing choice between slow down and move out

The main objective of our project is to design a coordination-free lane-switching/overtaking algorithm for drones. To evaluate its performance, we introduce two baseline algorithms. Then, by comparing the results of those baseline algorithms to the objective algorithm, we can quantify how our coordination-free algorithms optimize the system.

This chapter introduces the details of two baselines algorithms and our

objective coordination-free lane-switching/overtaking algorithm. In the introductions of some algorithms, we use the word "collision-free" to describe them, which is only applicable in a no packet loss environment. In addition, we will conduct experiments to test how packet loss affects the result of those collision-free algorithms.

4.2 The Naive/Do-Nothing Algorithm

This algorithm uses the simplest strategy: doing nothing when drones know they will suffer a collision. Namely, the behavior of drones is to keep the current speed and lane permanently. Then when a drone detects an incoming collision, it just ignores this information and keeps cruising as usual, until it collides or reaches the end of the tube.

Obviously, all incoming collisions will happen so that the final collision cost will be extremely high. At the same time, due to no action being taken, drones will never change their speed and lane. Hence the other two types of cost (i.e., speed and position costs) are always zero. Recalling from the cost model, we set a very high cost value for each collision, such that the cost incurred by overtaking is always less than waiting for a collision to happen. Based on that, using the naive algorithm will lead to the worst-case result in most scenarios. Therefore, the naive algorithm is the most basic algorithm that we will compare other overtaking algorithms' results against.

If other algorithms give a higher cost than the naive one, then it means those algorithms have negative effects on the system. There are two possible situations for that case. The first one is that those algorithms avoid some incoming collisions, but the corresponding speed and position cost are higher than the avoided collision cost. The other case is that algorithms prompt drones to make wrong decisions that cause additional collisions. In both cases those algorithms are not well designed.

In conclusion, the naive algorithm provides a low-level baseline to evaluate the performance of other algorithms, we can get a worst-case result from it.

4.3 "Always slow down" Algorithm

The "Always slow down" algorithm is collision-free because the drone will never change lane but always choose to slow down its speed when it detects an incoming collision with a drone in front. Based on this behavior, no drone will be out of the tube, and thus the position cost should be zero forever. In terms of speed cost, it should have a significantly high value because many drones with a high preferred speed might keep the slow down state for a long time.

If we say the naive algorithm performs the worst case across all other algorithms, then this algorithm is expected to be the worst of all collision avoidance algorithms. When facing an incoming collision, a drone has two choices: switching lanes or slowing down to avoid it. Slowing down obviously increases the speed cost. And switching lanes will sometimes lead to a lower speed cost and a high position cost.

The speed cost for switching lanes usually is much lower than the speed cost for slowing down. Recall from the system model in Section 3.2, that drones should have a parameter named $v_{overtake}$, which is a small value that indicates the speed in the two-dimensional direction space for overtaking. Hence the speed difference during the lane switching should also be very small. But for slowing down, the speed difference depends on the cruise speed of the drone in front. And based on our preferred speed distribution, that difference is possibly larger than the former. In addition, the period for lane switching is usually short (with our parameter settings, it is about one second), whereas drones may need to keep the slow speed for many seconds after slowing down. For these two reasons, we expect the speed cost for switching lanes usually is lower than for slowing down.

In our evaluations, if an algorithm's total cost is higher than this one, it will be considered a bad algorithm. In other words, the objective algorithm should perform much better than the always slow down algorithm.

4.4 Criteria-based Algorithm

The criteria-based algorithm can use several lane-switching/overtaking criteria for drones. When the current situation satisfies all requirements, the drone will either switch its lane or overtake to avoid the incoming collision. One thing to notice is that this algorithm allows for collisions since drones may not receive or update the safety information of neighbors and then make a wrong decision, or two drones may switch onto the sane lane simultaneously and collide.

This algorithm is "coordination-free," which means drones can only share their position, velocity, and neighbor information (also just speed and position information) with others. Drones react to different situations independently, they will not tell any decisions they have made and will not negotiate about them. To use a real-world analogy, it is like you drive a vehicle on a crowded road, and other drivers would not flash the turn signal, so you have to react with lane swaps to the actions of other vehicles, or proactively overtake others and hope they will not have a conflict with you. Further building on that analogy, coordination could be calling other drivers and starting a negotiation.

Figure 4.2 shows the main procedure of our algorithm. Here we only explain the "black" part of the figure which shows the basic procedure. The "red" part of the figure is related to moving back to the tube, and we will introduce it in Section 4.5.8. Our algorithm has four main states. They are ideal, criteria check, overtaking, and slow down. Roughly speaking, each drone initially stays at the ideal state, and it will turn into the criteria check stage as soon as an incoming collision is detected. If all criteria are satisfied, then the drone will start switching lanes or overtaking. If not, it has to slow down its speed and wait to check again. Finally, the drone will be back to the ideal state when the switching or overtaking action is finished. In the following section, we will introduce these states in detail.

We design many different criteria sets for this algorithm to compare results. In general, strict criteria cause less switching or overtaking, and the collision count would be low. Loose criteria are expected to have more of those actions and a higher collision rate. So one objective here is to see which sets of criteria can perform better in which cases, and to attempt to find the

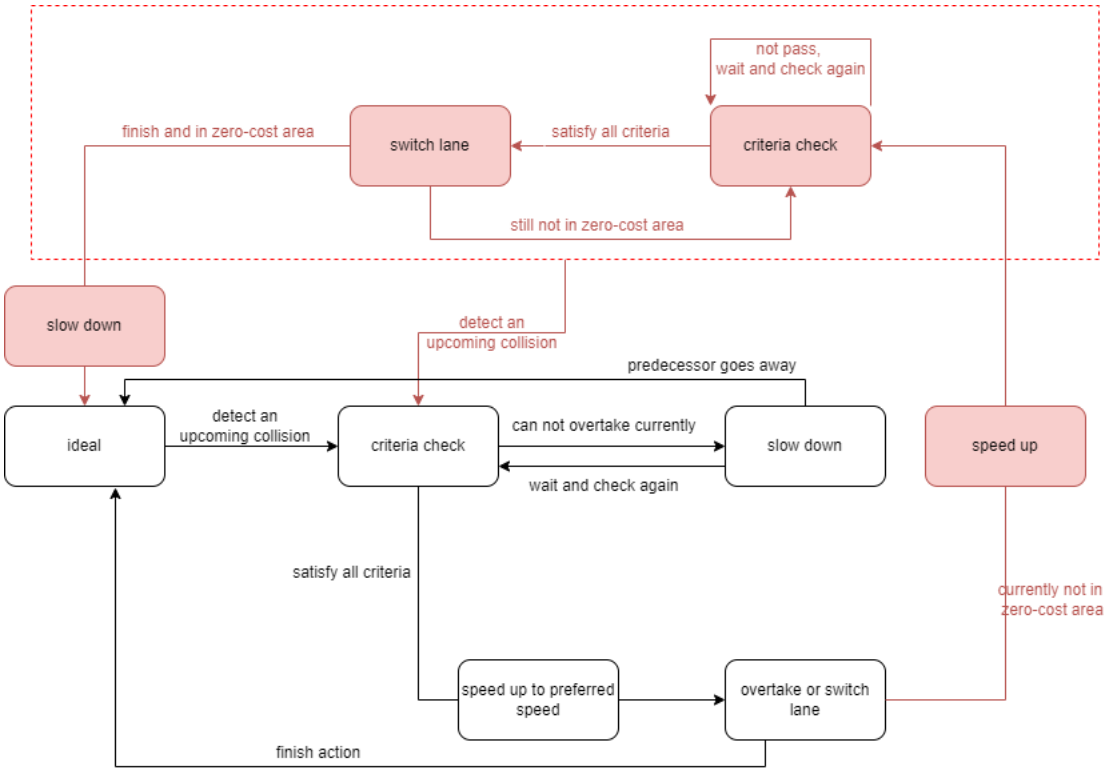

Figure 4.2: Procedure graph of criteria-based algorithms

trade-off between some simulation parameters, then give our recommendation in different scenarios.

4.5 Algorithm States

4.5.1 Ideal State

Being in the ideal state means that a drone is driving at its preferred speed, and no action needs to be taken. In this stage the drone will only have one task, which is detecting future collisions. Each time the drone receives a 'new' beacon packet from others, it will calculate an estimated collision time with the drones whose position information is updated. However, since only the drones behind need to respond to the incoming collision in our scenario, the calculated collision time is useful only for the drone behind[1]. If the

[1] For a drone, that calculation is only carried out for drones that are ahead of it

drone calculates that time for the drone behind it, it will simply ignore the calculation and do nothing for that value.

Assume (px_k, py_k, pz_k) and (vx_k, vy_k, vz_k) are the position and velocity vector of drone k respectively. Then drone k should be able to learn (px_j, py_j, pz_j) and (vx_j, vy_j, vz_j) when it receives a beacon from drone j. By performing the following calculation from solid geometry, we will get the estimated collision time.

Firstly, calculate the difference between those two types of vectors:

$$dp = (d_{px}, d_{py}, d_{pz}) = (px_k - px_j, py_k - py_j, pz_k - pz_j)$$

$$dv = (d_{vx}, d_{vy}, d_{vz}) = (vx_k - vx_j, vy_k - vy_j, vz_k - vz_j)$$

Then the collision will happen at a time t when

$$t = \min\{0 \leq t < \infty : ||dv * t + dp|| \leq d\}$$

Parameter d is the minimum safety distance of happening a collision. Based on that, we can construct a quadratic equation to compute the value of time t

$$(tdv_x + dp_x)^2 + (tdv_y + dp_y)^2 + (tdv_z + dp_z)^2 \leq d^2$$

We plug this into the general quadratic equation is:

$$at^2 + bt + c = 0$$

where a, b and c are:

$$a = {d_{vx}}^2 + {d_{vy}}^2 + {d_{vz}}^2$$

$$b = 2 * (d_{vx} * d_{px} + d_{vy} * d_{py} + d_{vz} * d_{pz})$$

$$c = {d_{px}}^2 + {d_{py}}^2 + {d_{pz}}^2 - d^2$$

After solving the equation, we should get two solutions t1 and t2, which represent a near solution and a far solution (an example is shown in Figure 4.3). A special case is when two drones move in different lanes, the quadratic

equation above will have no solution so that those two drones will not collide with each other (requires $\|dp\| > d$).

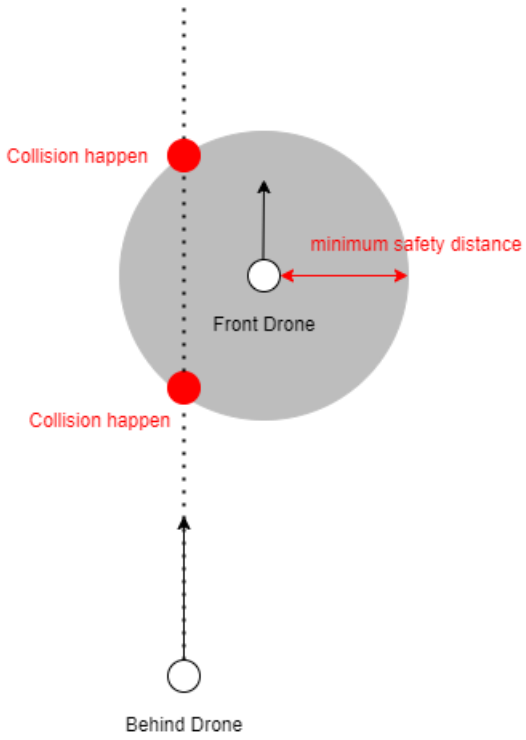

Figure 4.3: Calculating time for future collisions

Based on the sign (positive or negative) of these two solutions, there are four cases to distinguish:

- Both solutions are positive, then the estimated collision time is the smaller one or either if they are equal.
- One is positive and another is negative, which means the behind drone is already in the hazard area and should have suffered a collision with the front drone. In that case, an error should be thrown since those two drones should have been removed in the past.
- If both solutions are negative, it indicates the sender drone is behind the receiver drone, and it is not any problem.

- The last case is solutions are in the complex space, and that means these two drones will never collide with each other

If the algorithm meets the first case and the time value is less than a threshold $t_{threshold}$, it will immediately turn into the next state, criteria check. The purpose of setting $t_{threshold}$ is to avoid that drones start overtaking at a very early time (e.g., dozens of seconds before a collision happens). $t_{threshold}$ should depend on the speed difference. But for simplification, $t_{threshold}$ in our project is set to a constant value of one second. One important reason for that is to avoid the drone back to the initial lane due to another overtake. An example is shown in Figure 4.4. In the right part, we do not set $t_{threshold}$. Then the drone will immediately switch lane to avoid colliding with the drone in front. However, after switching, it faces another incoming collision with another drone. Therefore, it needs to switch lane again. One possibility is to go back to the original lane. In that case, it should have the third switching to avoid the drone in front (notice the original goal is just to avoid that drone). But with the $t_{threshold}$ in the left-hand side of the figure, the drone will avoid switching until the estimated collision time is less than $t_{threshold}$. Hence only one switching is performed in this case.

4.5.2 Criteria Check and Slow Down

In the criteria check state, lanes around drones are required to satisfy all the rules to lower the risk of collision during overtaking or lane switching. In the system model section, we have defined the tube layout in which each lane will have six neighbor lanes. Drones will check those lanes in turn to find a feasible route for collision avoidance. When a drone passes the check for one lane, it will enter the overtake/switching state and set the target lane to the one that satisfies the criteria. If several neighbor lanes meet the criteria, there is a secondary-criteria check to pick the best one. And in other cases, it needs to set its speed to the same as the front drone to avoid collision. Corresponding, it will move to the slow down state. However, with the change of surroundings, drones may pass the rule check after slowing down their speed. If the predecessor speeds up or goes to another lane, drones go back to the ideal state and cruise at their preferred speed. Therefore, we

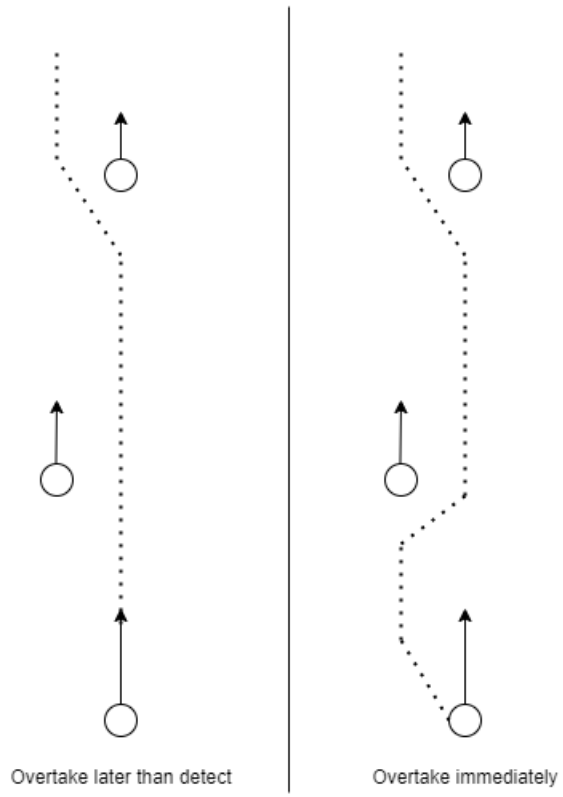

Figure 4.4: Two cases for starting overtaking at different time

implement a continuous check mechanism, in which drones check the criteria every second until it satisfies all the rules or the predecessor goes away.

In the following section, we introduce several criteria.

4.5.3 The Most Basic Criteria

The criterion has only one rule: "there is enough space between me and the drone in front of me." The expression "in front of" refers to two drones sharing the same lane, one being in front of the other. This rule will ensure a sufficiently large gap between the "prepare to overtake" drone and the "be overtaken" drone so that no collision will happen when the behind drone switches lane or overtakes. This criterion avoids the rear-end collisions caused by the starting of a switching. This kind of collision will not happen in the always slow down algorithm since drones choose to slow their speed, and the

rear-end can be avoided. But in other algorithms, we encourage the drone to switch or overtake. Hence, when a drone needs to do lane switching or overtaking, its speed should be larger than the first drone in front. Therefore, due to the acceleration, the behind drone might actively collide with the ahead drone if there is not a sufficient gap between them. The following Figure 4.5 shows how that criterion can prevent this kind of collision.

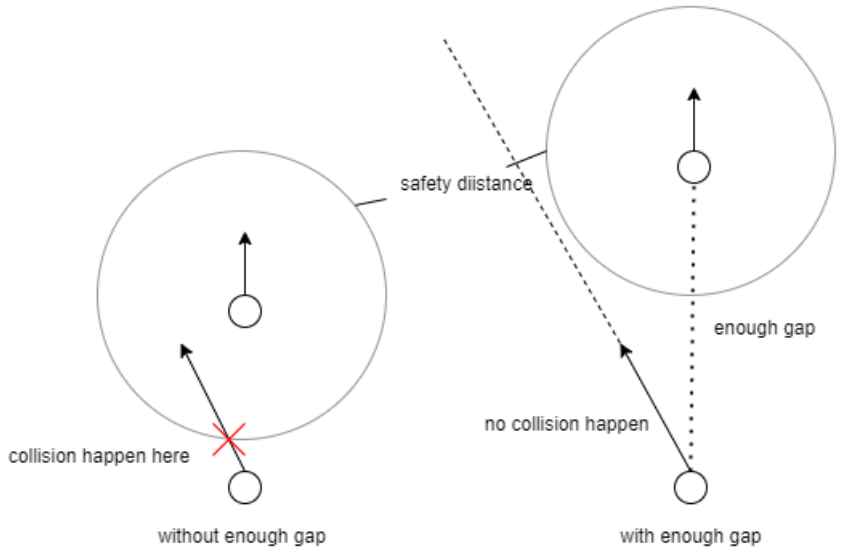

Figure 4.5: The importance of having enough gap between two drone

This set provides the lowest safety for drones since it cannot prevent collisions caused by other reasons, such as the collision with the drone in the target lane or collision with other drones also selecting the target lane for overtaking. In other words, this basic criterion cannot ensure the safety of drones in different lanes. But it also indicates that if the probability of there being a drone in other lanes is low, this criterion will likely be able to prevent most collisions. So, it is expected to perform well only the drone density is sufficiently low.

4.5.4 Target Lane Criteria

When switching lane or overtaking, the drone needs to select a target lane to start those actions. In this set, we add some criteria to restrict the eligibility

of target lanes.

The criteria are as follows:

- There is enough space between me and the ahead drone, which is the closest to me.

- There is some empty space in the target lane for me to switch lane safely.

The added rule for the target lane aims to avoid collisions with drones in the target lane. We set two different standards for the empty space. The first one requires more space but is safer in that it can ensure the drone who are planning to switch lanes will not collide with other drones in the target lane. The following equation can calculate the expected length value of this gap where v_{pmax} and v_{pmin} are the maximum and minimal preferred speeds for drones, respectively. And $t_{switching}$ is the time used for switching lane.

$$length = 2 * (v_{pmax} - v_{pmin}) * t_{switching}$$

But in reality, drones should not know some parameters such as the maximum or minimum preferred speed of other drones. In the simulation, we will replace the safety length with a fixed simulation parameter, which will make the safety length inaccurate. Sometimes that fixed parameter is smaller than the actual safety value we calculated above. And in these cases, a collision may happen in the area that we "ignored."

4.5.5 Target-Neighbor Lane Criteria

Another rule is added for neighbor lanes of the target lane; hence drones can avoid some collisions caused by concurrent overtaking from two drones select the same lane.

The criteria are listed as follows:

- There is enough space between me and the ahead drone.

- There is some empty space in the target lane.

- There is some empty space in the neighbor lanes of the target lane (except the original lane).

The actual safety length of the empty space for neighbor lanes depends on several physical considerations such as maximum acceleration or deceleration, so here we simply use another fixed simulation parameter to represent that. And similar to the target lane criteria, two versions are given, one with full value, and the second one being half that value. If the first standard is used, no collisions are expected to happen since no other drones can be too close during the overtaking or switching action.

4.5.6 Loose Target-Neighbor Lane Criteria

Instead of requiring no drones in neighbor lanes, we only check whether the speed of the current drone is the fastest for all the neighbor lanes in some range.

The criteria are listed as follows:

- There is enough space between me and the ahead drone.
- There is some empty space in the target lane.
- I have the fastest speed for all neighbor lanes in some range.

Here, the "range" value is the same as "some empty space." This criterion is also collision-free once we use the first standard of the safety length value, since only one drone can overtake in a piece of the tube, and then no collision should happen due to concurrent lane switching action.

4.5.7 Secondary Criteria Check

Due to multiple feasible lanes that might exist after applying the criteria check, we need another check to distinguish which one is the best for the current case. Hence we set secondary criteria check for those lanes that passed the first criteria check.

The procedure of this check is:

- First, select all lanes with the lowest tier number (as introduced in Section 3.1). In this step, we treat all tiers within the tube (in our scenario, they are tier one and tier two) as the same, since drones flying in these tiers will not incur any position cost.

- Then, for each filtered lane, calculate the estimated collision time with the first drone in front in the same lane.

- Finally, select the lane with the longest estimated time as the target lane. If there is more than one feasible lane, randomly select one.

The basic idea for the secondary criteria check is to prioritize lanes with a smaller tier number, as it can directly reduce the potential position cost. We then use the estimated collision time as the secondary criteria to choose the suitable lane. Note that we do not consider further things. But it can help postpone the time before the next overtaking. When this happens at the end of the tube, it might even help to avoid the next overtake. One thing that needs to be noticed is that our algorithm does not balance the first check (tier number) and the second check (estimated collision time). That is, lanes with lower tier numbers always win over lanes with longer estimated collision time.

4.5.8 Overtake / Switch lane

In the Overtake/switch lane state, drones first change their velocity to move to another lane. In our algorithm, the speed along the y-axis is not changed during the lane switching. Recall that drones are asked to fly in parallel to the y-axis, so when switching lanes the velocity value along the x-axis or z-axis or both will not be zero anymore. During the check state, the drone has found a target lane with its position vector (t_x, t_y, t_z). Note that we have a parameter $v_{overtake}$ to restrict the velocity value along the x-axis or z-axis. Hence we can calculate the desired velocity vector for lane switching.

Assuming the drone's current position is (p_x, p_y, p_z), we can first calculate the normalized vector of the expected overtake speed.

$$n = (n_x, 0, n_z) = normalized(t_x - p_x,\ 0,\ t_z - p_z)$$

Then the velocity v will be calculated by:

$$v = (v_x, v_y, v_z) = (n_x * v_{overtake},\ v_y,\ n_z * v_{overtake})$$

$v_{overtake}$ is a value of the maximum speed in the x and z axis.

Then the time t for that lane switching is:

$$t = \frac{t_x - p_x}{v_x} \quad or \quad \frac{t_z - p_z}{v_z}$$

Using which fraction depends on whether v_x or v_z is zero.

When they finish moving to another lane, drones will switch back to the ideal state or move back to the original lane, depending on which mode (switching or overtaking) they have enabled. In the overtaking mode, drones are expected to check the criteria for the origin as their target. These checks will happen periodically if drones fail to find a safe route. However, the drone may not be able to go back to the original lane due to either (1) there not being a suitable route for a prolonged amount of time, or (2) it needs another overtake to respond to an upcoming situation, or (3) it reaches the end of the tube. For all these cases, the drone will finish the periodic check, set the current lane as the original lane, and start to prepare the new overtaking if needed.

Since staying in lanes with a large tier number leads to a huge position cost, even a drone with switching mode should go back to lanes that are in a low tier. Here we introduce the "red" part of the procedure graph shows in Figure 4.2.

If a drone has moved out of the zero-cost area, it will firstly speed up and then attempt to move into the tube, but this time, the target lane may not be the original lane it has used before. Only lanes, whose tier number is smaller than its current lane are considered as potentially feasible lanes. Also, a similar periodical lane criteria check will be performed to decide whether we can switch lanes right now.

Once a successful switching has finished, drones will either go back to the ideal state or continue to move towards lower tiers, depending on whether they are in zero-cost area. If they go back to the ideal state, their speed

should be set again to the preferred value. It is possible to detect a new incoming collision during the criteria checking. If drones detect this type of collision, they will then finish the checking process and change the state to the original criteria check state to avoid the upcoming collision. Once the collision avoidance process is ended, drones should be set to the red criteria-check state again to continue to move into the zero-cost area.

Also, the recursive overtaking problem only happens when the drone chooses to overtake rather than switch. In other words, using switching instead of overtaking can avoid this kind of problem, and leads to a simple algorithm.

Chapter V

Simulations and Results

This chapter focuses on showing simulation results and explaining them. In Section 6.1, we give the basic simulation parameters to help readers better understand our simulation. We also provide the result of the preliminary study in Section 5.2 to narrow down the set of parameters to be varied in further simulations. In addition, it can also help to answer the third research question.

- ***RQ3****:* Can we find a "genie" solution (i.e. guarantee no collision) for the problem?

Based on our first and second research questions:

- ***RQ1****:* Can our coordination-free algorithms perform better than the baseline? Do they have any limitations?

- ***RQ2****:* What factors (drone density, noise, etc.) have significant impact on performance, and how do they affect it? Can we explain that?

We ran several simulations and varied key parameters to see how they affected the cost performance. Results are shown in section 5.3, help to answer those two research questions. Moreover, in section 5.4, we will show an interesting phenomenon we discovered during the research.

5.1 Basic Simulation Environment and Parameters

All simulations in this project were built and ran under OMNet++ 5.6.2 in the Linux operating system. Simulations are built based on the INET framework version 4.3.2 and a local broadcast protocol framework.

In our simulation implementation, there are two kinds of module: node and node manager. Node is the object we observe during the simulation. Each node can be seen as a drone. It is an extended version of ad-hoc host in INET framework that we add two important sub-modules: transceiver and processor. Transceiver is used for transmitting and receiving beacons to all nearby nodes by following the local broadcast protocol. Processor analyzes the received beacon and responds to the incoming collisions. Node manager acts as the administrator of the simulation network, it generates drones according to the configuration file and removes drones when it detects collisions or arrivals.

Table 5.1 shows the basic simulation parameters that are not varied across all our simulations.

Parameter	**Value**
IEEE 802.11 version	11p
Zero position cost area radius	2.5m
Lane distance	1m
Tier number in zero position cost area	2
Minimum Safety Distance	0.5m
Node generating period	1s
Max speed	30mps
Preferred Speed Distribution	uniform(0.67,1)
Mac layer channel model	log-distance model
Bitrate	12Mbps
Collision cost coefficient	1000
Speed cost weight (τ_1)	1
Position cost weight (τ_2)	5

Table 5.1: Fixed simulation parameters introduced in Chapter III

And the Table 5.2 represents the parameters of the variables that we are going to vary to explore their effect.

Note that, 30s simulation time is just for the preliminary study in Section 5.2. 2500m length and 200s, 400s time are just for Section 5.4.2 in this chapter. Furthermore, we use random inter-arrival time beacon transmission to avoid packet collision. For the results in Section 5.2, 5.3, and 5.4, our graphs show the average values of 30, 100, and 300 receptions, respectively.

Parameter	Value
Repeat	30,100,300
Tube length	1000m, 2500m
Simulation time	30s, 100s, 200s, 400s
Average number of drone generated at each epoch	0.5..7 step 0.5
Path loss exponent	2, 3
Collision avoidance algorithm	8 different algorithms
Transmit power	0.01W, 0.005W, 0.001W
Average number of beacon per second	25, 28, 33, 40 ,50

Table 5.2: Varied simulation parameters

5.2 Preliminary Study

At the early stages of this project, we had a preliminary study to access the magnitude of impact of simulation parameters with the goal of identifying less relevant ones and fixing their parameters. With this study, we can know which criteria are valuable for further simulations. In this section, we present our findings.

5.2.1 Comparing Different Criteria

Based on the criteria-based algorithm introduced in Section 4.4, we prepared eight set of rules for lane switching and then compared their overall performance to find out some better algorithms for further simulation.

The eight algorithms are:

Index	description
0	Naive
1	Always slow down
2	Loose Target-Neighbor lane criteria, using full safety distance
3	only check target lane, using full safety distance
4	Loose Target-Neighbor lane criteria, using half safety distance
5	Target-Neighbor lane criteria, using full safety distance
6	Target-Neighbor lane criteria, using half safety distance
7	only check target lane, using half safety distance

Table 5.3: Possible collision avoidance methods

Algorithm 0 was introduced in section 4.2, and algorithm 1 was in section

4.3. Algorithm 2 and 4 were explained in section 4.5.6. Algorithm 3 and 7 were in section 4.5.4. Finally, algorithms 5 and 6 were explained in section 4.5.5.

The total cost for those algorithms are shown in Figures 5.1 and 5.2. In this simulation set, the simulation time is set to 30s, and the drone generation rate (how many drones are generated each second) varies from 1 to 7, with one as the step. In Figure 5.1, the total cost increase appears to be broadly linear. There is a significant difference between the naive algorithm and the others. The naive algorithm costs approximately eight times as much as other algorithms due to its high collision cost. In addition, the cost increases with a larger slope compared to others. This result meets our expectations since the naive algorithm cannot avoid any kind of collisions. Furthermore, algorithms related to our criteria-based algorithm all have a quite small value of total cost compared to the naive algorithm, hence showing a clear advantage.

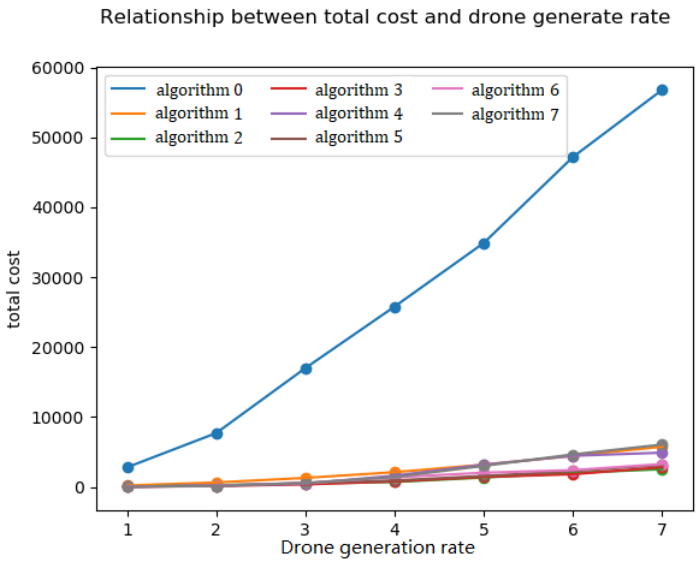

Figure 5.1: Total cost of eight algorithms

In Figure 5.2, we eliminate the line for the naive algorithm to make the difference between the other algorithms clear. Based on this result, we can separate those algorithms into two groups: algorithms 1,4,7 are in the bad performance group, and algorithms 2,3,5,6 are in the good performance group.

Except for the algorithm named "Target-Neighbor lane criteria, using half safety distance," all other algorithms which just check half the safety length are in the bad group. However, if we only consider the low-density case (i.e., when the generation rate is less than 3), algorithms 2-7 seem to perform equally well. After that, if we increase the drone generation rate, the cost for algorithms 4 and 7 increases significantly faster than the cost of other algorithms (except algorithm 1).

One thing to be noticed is that algorithm 1 is the "always slow down" algorithm, which we have expected to be the worst solution to the drone collision problem (aside from the naive algorithm), but the result for algorithm 7 is larger than the "worst case," for high generation rate surprisingly. That indicates algorithm 7 (only checking target lane, using half the safety distance) is a weak solution.

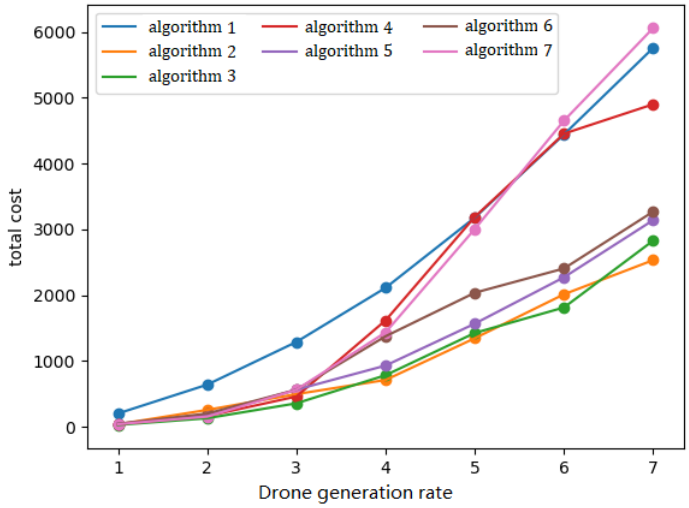

Figure 5.2: Total cost without naive algorithms

Figure 5.3 shows the collision cost for algorithms 1-7. Obviously, algorithm 1 and 5 (Always slow down, and Target-Neighbor lane criteria, using full safety distance) have zero collision cost. Algorithms 2,3,6 have a minor collision cost, but for algorithms 4 and 7 the cost is much higher suggesting that they will have a high probability of drone collisions.

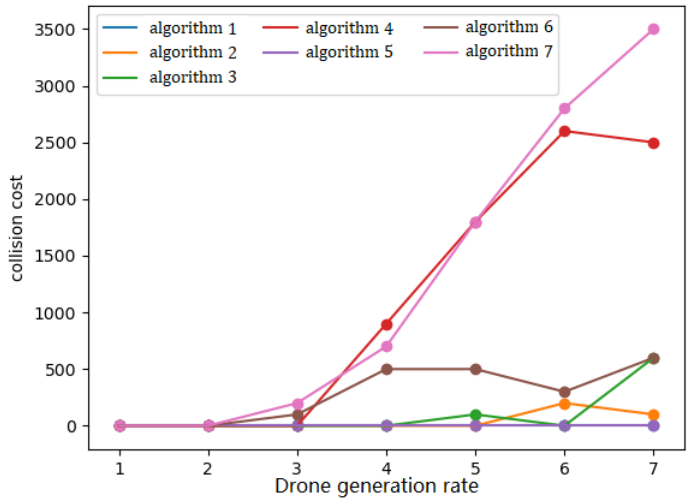

Figure 5.3: Collision cost of eight algorithms

Combining results for the total cost and collision cost, we find that algorithm 5 performs the best as it is in the good performance group in terms of total cost and has no collisions during the simulation. Algorithms 2,3,6 are also good enough that we can experiment with them further. However, for other algorithms, those preliminary results indicate that their performance is significantly worse than the algorithms above, so we are not interested in them and will not test them in more detail.

5.2.2 Switching vs Overtaking

In the algorithm introduction (Section 4.1), we distinguished two modes for collision avoidance: lane switching and overtaking. In this section we compare the performance of those two modes. Specifically, algorithms 2,3,5,6 are tested with 1, 5, and 7 as the drone generation rate, representing the low, medium, and high drone density, respectively. In this thesis, we only show the results for "drone generation rate = 5" since the results for others are very similar.

The results (shown in Figure 5.4) indicate that there is only a minor difference between switching and overtaking mode, so it is hard to say which one is better in this case. Because of the small difference, there is no need

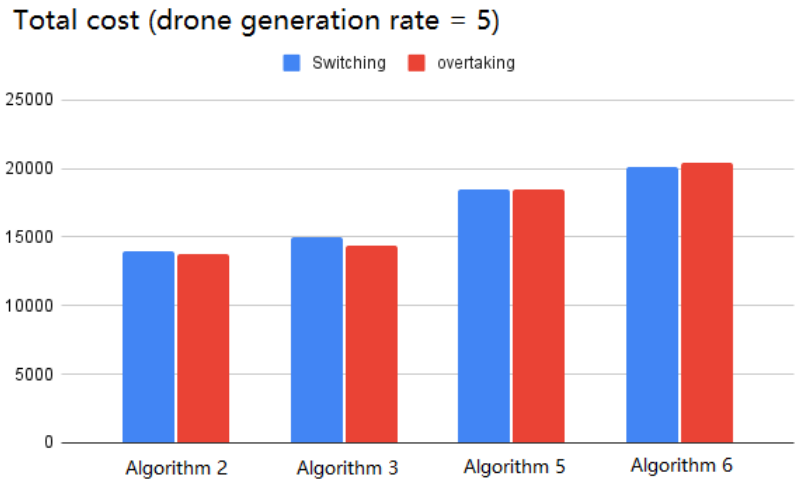

Figure 5.4: Total cost of switching and overtaking mode

to experiment with both in the following simulation. In Section 4.5.8, one important feature of lane switching is that it avoids the recursive overtaking problem. This can make the collision avoidance algorithm simpler. Hence, we finally decided to use lane switching as the only mode for further testing.

5.2.3 Effect of Packet Loss

Section 2.4 introduced that packet loss plays an important role in wireless communication. It could be the main problem for nodes to exchange information for safety use. Hence, one goal here is to see how packet loss can affect our algorithm's performance and whether it can even let collisions happen in a "collision-free" algorithm. We compared results using the unit disk model and the log distance packet loss model for the four algorithms we are interested. By setting the numbers of beacon per second to 50, the interference between packets should be serious in log-distance model, and many packets might get lost. For the unit disk model with a communication range of 200m, we disable the effect of path loss so that packet loss can only be caused by packet collisions, which leads to lower packet loss compared with the former model.

Our results are shown in Figure 5.5 and 5.6, which reveal the difference

for total cost and collision count, respectively. Notice that the beaconing rate in this simulation is high (about 50 packets each second). From now on, we will use abbreviations for the four better coordination-free algorithms, shown in the following table.

Criteria the algorithm used	**Abbreviation**
Loose Target-Neighbor lane criteria, using full safety distance	Loose T-N-full
Only check target lane, using full safety distance	T-full
Target-Neighbor lane criteria, using full safety distance	T-N-full
Target-Neighbor lane criteria, using half safety distance	T-N-half

Table 5.4: Abbreviation for the potentially good collision avoidance algorithms

Obviously, with fewer packet losses by using the unit disk model, the total cost for each algorithm is about 2.5 times lower than using log-distance model, and that difference is mainly due to a large number of collisions in the latter model. As for the collision count, there is a significant gap between those two models, only a few collisions happened in unit disk model while nearly half the drones were destroyed (around sixty collisions) using the log-distance model. This result manifests that packet loss is the main reason for collisions in our simulation environment.

The figure for collision count also indicates that no collision happened for the Loose T-N-full and T-N-full algorithms. That result conforms to our expectation in the introduction of those algorithms in Section 4.5. However, due to the limit of the number of drones (at most 300 at the same time) and the simulation time (just 100s), we are not confident enough to say that the two algorithms are collision-free by using unit disk model.

In the remaining of this chapter, we do not consider the unit disk model anymore, since log-distance model is more realistic, and we are interested to see how packet loss is affected by our simulation parameters. Also, we aim to find out at what level of packet loss rate our algorithm could have a good performance.

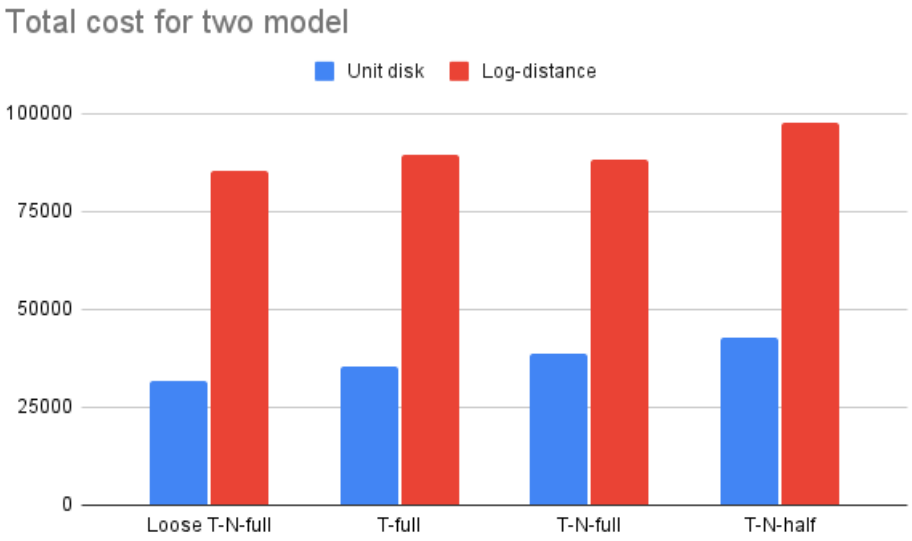

Figure 5.5: Total cost for the two mode

Figure 5.6: Collision count for the two mode

5.3 Impact of Key Parameters

In this section, we show results when varying a range of key parameters to learn how each parameter affects the performance, helping us answer RQ2. In addition, in the simulation, different algorithms are also seen as a parameter to indicate the difference between different criteria. That can help us answer RQ1.

Those parameters and the corresponding values are listed in Table 5.2, except tube length and simulation time, which are set to 1000m and 100s, respectively. Furthermore, we only test those algorithms that performed well in Section 5.2.1. Overall, we have simulated 1680 parameter combinations, and with 100 repetitions. For each combination, we have calculated the average over all repetitions. For the graphs and tables in this section, if we only specify the value of one parameter, then the result shows the average of all parameter combinations whose value of this parameter is equal to our specification.

5.3.1 Collisions

The total collision counts for each drone generation rate is shown in Figure 5.7. It clearly shows that the T-N-half algorithm has the highest collision count, and the next is T-full. T-N-full and Loose T-N-full perform similarly well, their collision counts are close to each other.

When drone generation rate is less than 2.5, the collision count for all algorithms increases with a quite small slope. In addition, there is no significant difference in cost values in this case. However, when the generation rate is more than 2.5, the collision counts for algorithms T-full and T-N-half start to increase faster. For T-N-full and Loose T-N-full, this kind of drone generation rate threshold is 4.5. Only if the generation rate reaches 4.5 can we see a significant increase of the collision count. Otherwise, the collision count is quite close to zero. Moreover, in the most crowded situation (generation rate = 7), the collision count for the best algorithm is about two times smaller than for the worst algorithm.

Figure 5.8 shows the collision counts for different beacon intervals. The values are quite small, and they grow very slowly when the beacon frequency

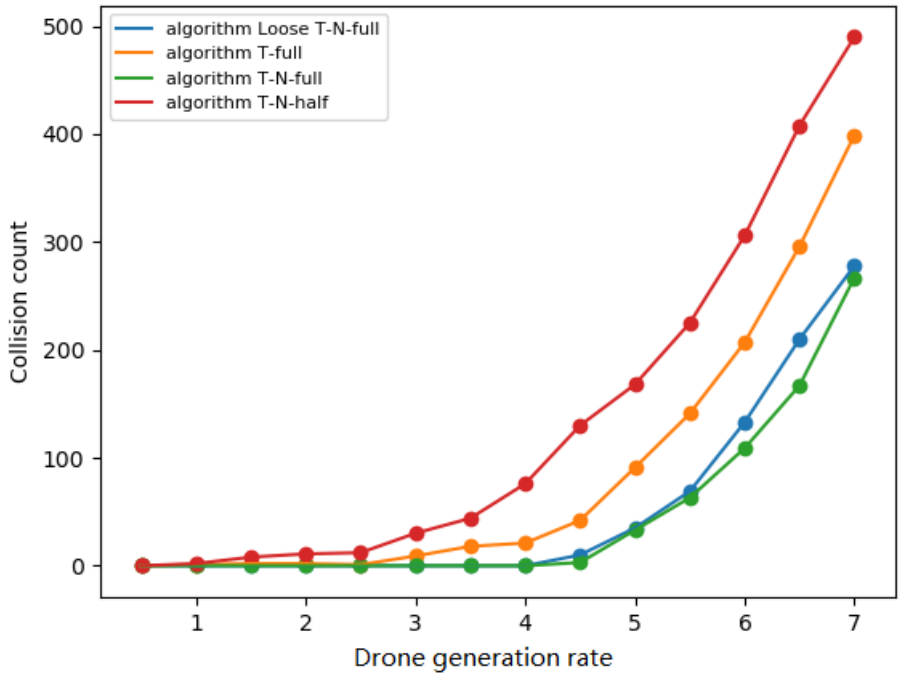

Figure 5.7: Relationship between collision count and drone generation rate

is either of 25, 28, 33 HZ. In other words, when drones send fewer than around 33 beacon packets per second, only few collisions happen. When the interval is around 0.025 seconds (i.e., 40 beacons per second), the collision count sharply increases if drone generation rate is larger than 5.5. And for the interval of 0.02 seconds, the collision count already starts to increase when the generation rate reaches 4. In the most crowded situations, 0.02s beacon interval leads to the most collisions, which is about three times than the second one (0.025s) and about twenty times than others.

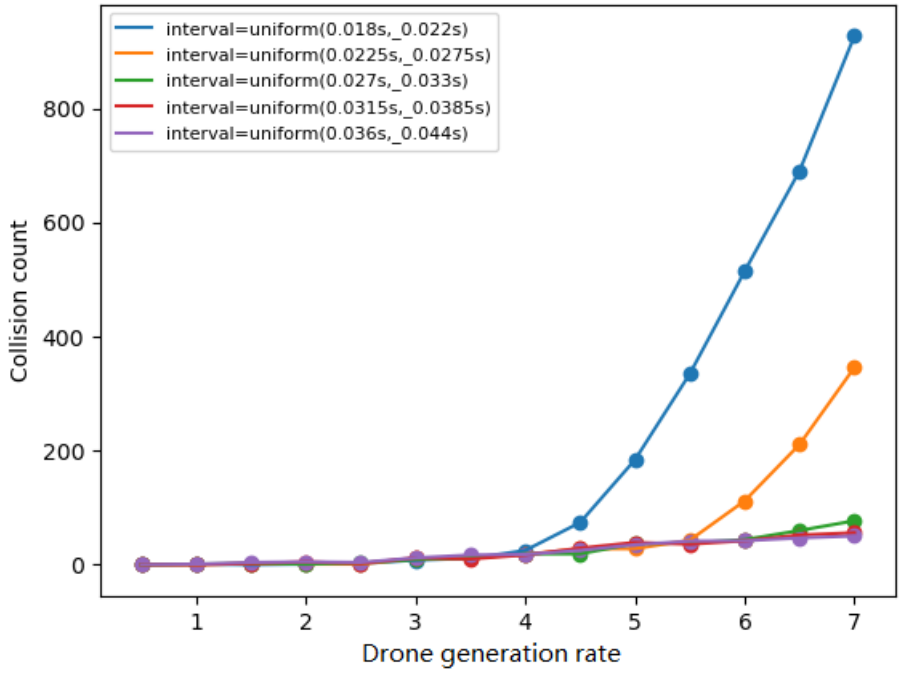

Figure 5.8: Collision counts for different beacon interval

The two figures above indicate that both drone generation rate (drone density) and beacon interval can significantly affect the collision count. More transmitted packets cause a higher probability of having packet collisions, leading to more packets being lost. Therefore, increasing the drone generation rate, or reducing the beacon interval would increase the packet loss rate. And lost packets translate into uncertainty about the position of other drones.

Hence, more collisions can happen.

Figure 5.9 shows the collision counts for different path loss exponents and transmit powers. In cases when path loss = 3 or transmit power = 0.001W, the collision count remains a small value and grows steadily and slowly with increasing drone generation rate . But in other cases, we can see a "superlinear" growth of the collision count. For example, the differences between (1) transmit power = 0.001W and 0.01W (2) Path loss = 3 and 2 are both obvious.

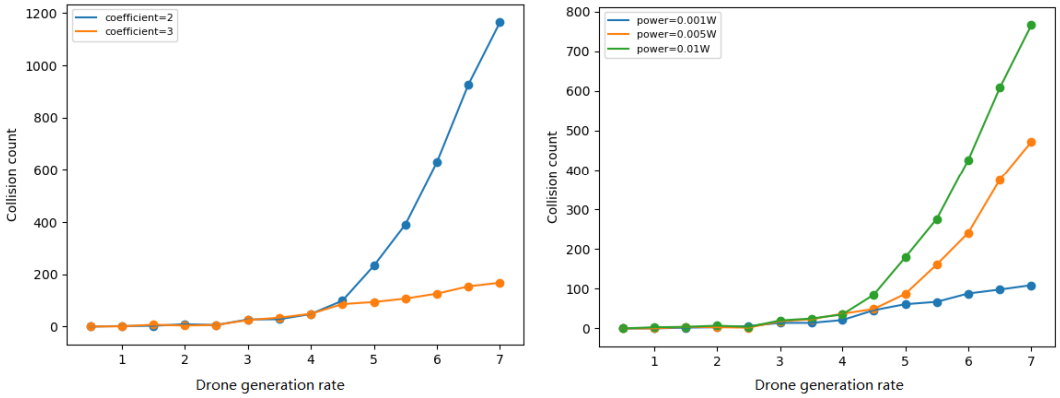

Figure 5.9: Collision counts for different path loss exponent (left) transmit power (right)

Path loss exponent and transmit power can also significantly influence the collision count in our simulation. Like the drone generation rate and beacon interval, these two parameters also impact the packet loss rate, However in a different way. A smaller path loss exponent or a larger transmit power would increase a drones' interference range (also the communication range). With larger communication and interference ranges, drones can receive beacon packets from others that are at a wider distance. This kind of effect is a double-edged sword. On the one hand, drones can know more drone safety information to avoid potential collisions with other faraway drones. But, on the other hand, the possibility of receiving multiple beacons at the same time would also increase (mostly through hidden-terminal collisions). Leading to packet loss. An additional problem is that with a large commu-

nication/interference range, a station will see more competitor when doing carrier-sensing, leading to fewer transmit opportunities. Then, drones may not be able to successfully update information for their neighbors. Finally, more collisions happen.

Our result shows that increasing transmit power or having a smaller path loss exponent can negatively affect the collision count. That indicates the communication range when (1) path loss exponent = 3 or (2) transmit power = 0.01W is large enough for drones to learn sufficient information to avoid the collision. Lowering the path loss exponent or increasing transmit power could enlarge the packet loss rate. And hence, the collision count would be boosted.

Combining those four figures, all parameters concededly can significantly affect the final results. The trends for all lines can be divided into two categories, the first is slowly growing, and the second is a superlinear increase when drone generation rate is larger than a threshold. However, one problem is that both transmit power and beacon interval should have poor performance at their extreme values (very low, very high), and the optima is somewhere in between. In our results, all simulations only show one-side effect. So we cannot find the optima of these parameters.

To see whether our parameters can affect the drone destruction rate (destroyed drones / all drones), in Figure 5.10, we show simulation results where the drone destruction rate is ≥ 0.05, since the difference is unclear in other cases. In the figure, a triangle means the beacon interval is 0.2s (square is 0.25s), and blue shapes mean the transmit power is 0.01W. So, a blue triangle represents that the beacon interval is 0.2s and the transmit power is 0.01W, similar for other cases. And the path loss exponent is always 2.

Obviously, the destruction rate only exceeds 0.05 when the drone generation rate is greater than or equal to 4.5 and path loss exponent is 2. For other values of interval (0.033, 0.035, and 0.04) and transmit power (0.001W), no simulations reach the threshold.

With the increase of generation rate, the destruction rate for the simulations with the same parameters shows approximately linear growth. Hence, increasing the drone generation rate can have a worse effect on drones since each of them would have a higher probability of being destroyed. In addi-

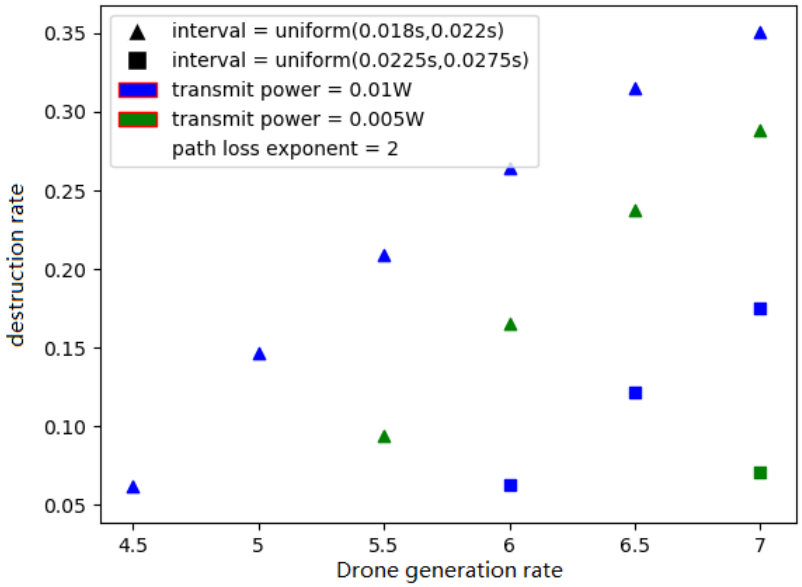

Figure 5.10: Bad destruction rate (≥ 0.05) in different situations

tion, the increasing slopes for different parameter sets are also approximately the same. This suggests that changing the drone generation rate's effect is independent of other parameters. If we fix the beacon interval and transmit power, and increase the generation rate by 0.5, the destruction rate approximately grows by 0.075 ± 0.025.

Simulations, with 0.02s beacon interval and 0.01W transmit power give the worst destruction rate, with more than 35% drones being destroyed by collisions. That is the worst case observed in our simulations.

5.3.2 Cost

Cost is the most important parameter to evaluate how good an algorithm is. Recall from Section III that our cost model includes three types of cost: position cost, speed cost, and collision cost. By combining all of them (with weights for each cost), we can calculate a total cost to show the overall performance. In this section, we only give a brief explanation of the figures and provide more detailed discussion in Section 6.1.

Figure 5.11 shows the average total cost with different drone generation rates. All four algorithms give a similar result. The total cost increases faster as the drone generation rate increases. For example, the difference between rate = 7 and rate = 6.5 is larger than the difference between rate = 6.5 and rate = 6. Algorithm T-N-half always has the highest cost for each drone generation rate, while algorithm T-N-full is the second one. The other two algorithms perform around equally well all the time and always have the lowest cost.

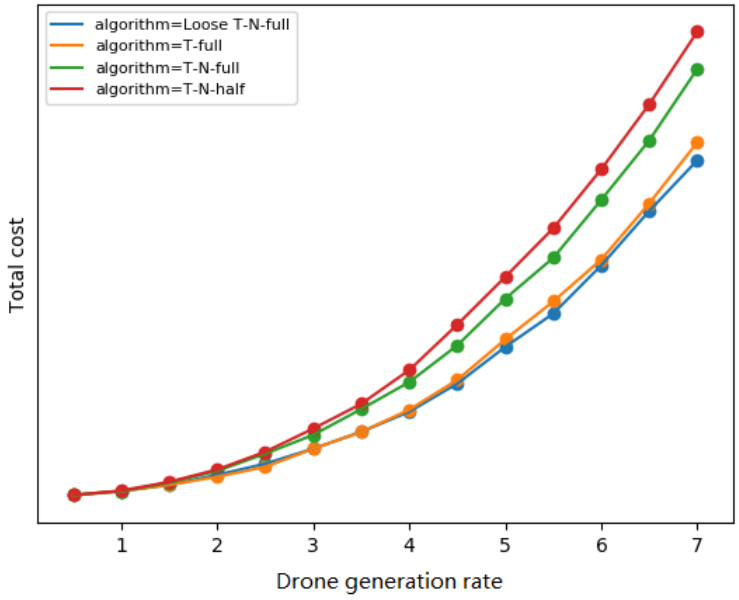

Figure 5.11: Relationship between total cost and drone generation rate for four different algorithms

One thing to be noticed here is that the order of algorithms is different in Figure 5.11 and Figure 5.7. An algorithm with a higher collision count may not have a high total cost. It indicates a phenomenon that the cost of avoiding a collision may be higher than the cost for letting it happen. Again, this is discussed in Section 6.1.

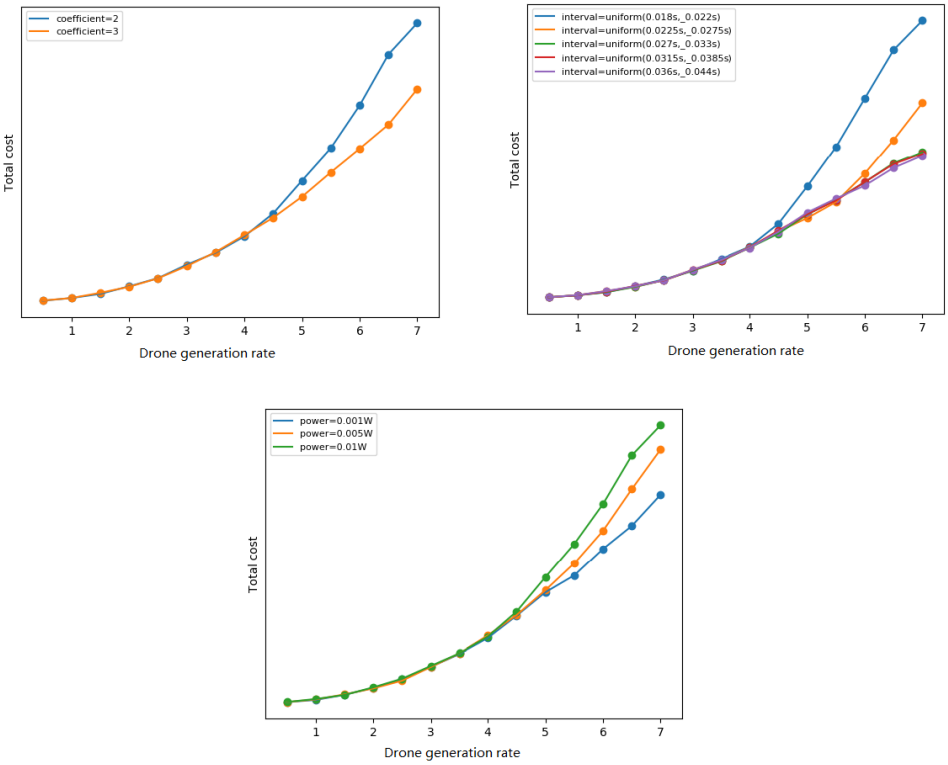

Figure 5.12: Effect of transmit power (bottom), path loss exponent (upper left), and beacon interval (upper right) on the total cost of algorithm T-N-full

Figure 5.12 represents how transmit power, beacon interval, and path loss exponent affects the total cost. Firstly, all lines shown in these three graphs have a similar trend of a faster-than-linear increase of cost as the drone generation rate increases. The results for path loss exponent of 2 are always worse than these for 3. However, when the average beacon interval equals 0.03s, 0.035s, or 0.04s, there is no significant difference on the graph. When the beacon interval is less than or equal to 0.25, the impact on cost becomes more pronounced when the drone generation rate is fixed. Finally,

for large drone generation rates, increasing the transmit power (from 0.001W to 0.01W) also substantially increase total cost.

As we have seen in the last section, changing these parameters can affect the collision count. And with an increasing number of collisions, the collision costs increase as well. Hence the total cost increases too. For example, varying the path loss exponent from 3 to 2 causes more collisions. Hence the total cost for path loss is 2 should approximately be the sum of the total cost for a path loss exponent is 3 and the collision costs for the additional collisions. That is the main reason for the difference when we change the value of parameters, but note that these parameters also affect the position and speed cost.

Our four switching algorithms have different rules for drones to start lane switching. Leading to differences in the position cost and speed cost. We record the values of each type of cost for all four criteria. Results are shown in Figure 5.13.

We see that, speed cost is always higher than position cost, and position cost is completely caused by lane switching. The speed cost is caused by either slowing down or staying out tube[1]. So, here we cannot infer which action (slowing or switching) incurs the highest cost.

Combining these two graphs, algorithm T-full is the best since it has minimal speed and position cost. The second-best is the Loose T-N-full algorithm, which shows a minor difference to the former. T-N-full seems to be the worst as it has the second-highest speed cost and the highest position cost by some meaning. However, if collision cost is considered, the actual performance and the ordering of these algorithms changes, loose T-N-full is the best, and T-N-full is now quite competitive. More discussion about this can be found in Section 6.1.

In Table 5.5, we list the ratio of different types of cost to total cost in different situations. We choose three cases for the drone generation rate, which are 1, 4, and 7, to represent the situations of a low, medium, and high drone density, respectively.

When drone density is low, Loose T-N-full has the highest ratio of speed

[1] As explained in Section 4.5.8, drones always use the maximum speed when they are out of the tube.

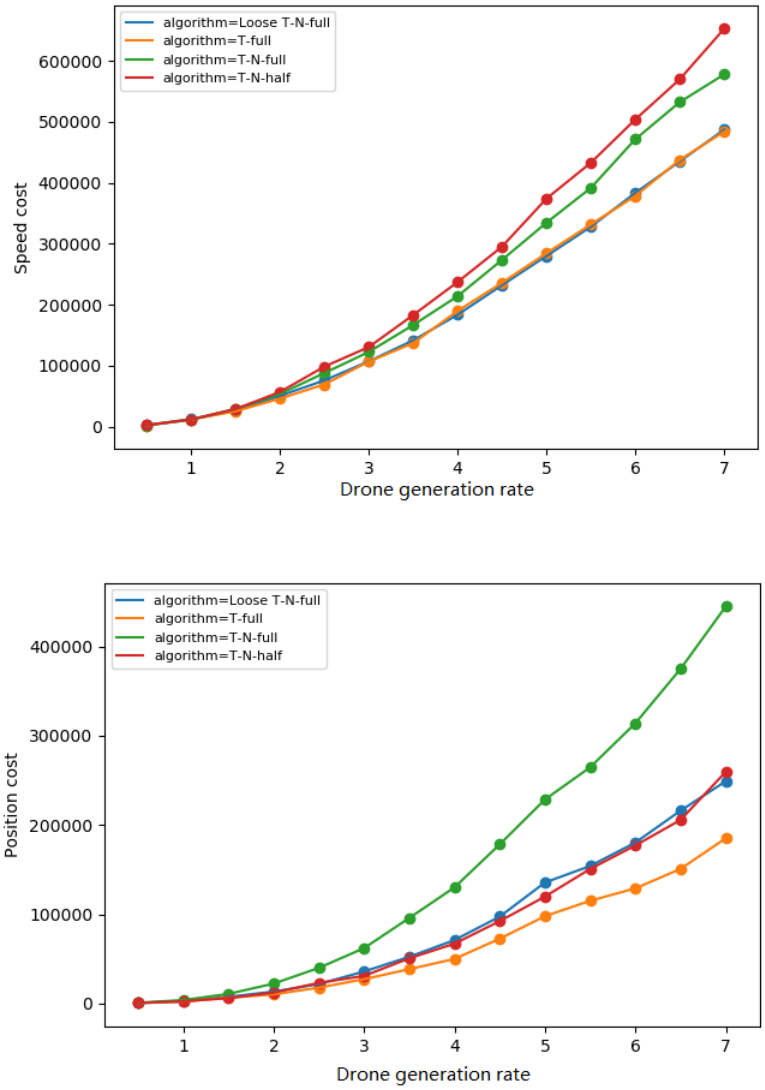

Figure 5.13: Position and speed cost for different algorithm.

	generation rate = 1				**generation rate = 4**				**generation rate = 7**			
	s	p	c	t	s	p	c	t	s	p	c	t
Loose T-N-full,	0.8474	0.1525	0.0000	14564.54	0.7208	0.2791	0.0000	254573.57	0.4802	0.2456	0.2741	1013975.63
T-full	0.7865	0.1462	0.0673	14866.45	0.7280	0.1914	0.0806	260619.20	0.4530	0.1733	0.3736	1067897.93
T-N-full	0.7514	0.2486	0.0000	15166.10	0.6217	0.3783	0.0000	344094.7450	0.4480	0.3450	0.2070	1289570.29
T-N-half	0.7314	0.1454	0.1232	16235.38	0.6239	0.1762	0.1999	380149.7426	0.4654	0.1852	0.3495	1401528.97
	s / p				**s / p**				**s / p**			
Loose T-N-full,	5.55				2.58				1.96			
T-full	5.38				3.80				2.61			
T-N-full	3.02				1.64				1.30			
T-N-half	5.03				3.54				2.51			
s = speed cost / total cost, p = position cost / total cost, c = collision cost / total cost, t = total cost												

Table 5.5: Ratio of different types of cost

cost, while T-N-full has the highest ratio of position cost. That does not change when the drone density becomes medium or high. It suggests that drones using T-N-full are most likely to fly out of the tube and stay in this state for a long time.

Increasing the drone density means that the value of s for each algorithm decreases, especially for medium to high densities, s suffers a large attitude of decrease than low to medium. The reason for the decrease of s is mainly due to the increase of c. The collision ratio grows as the drone generation rate increases, this phenomenon is more obvious when the density changes from medium to high.

However, despite p also growing from low to medium density, it shows slight reduction when density changes from medium to high. The growth of p when the density is from low to medium is mainly due to the tube becoming crowded and drones not finding an in-tube lane to switch to. Then more drones would fly out of the tube, leading to an increase in position cost. Furthermore, the slight reduction is also due to the growth of collision cost.

Considering the collision ratio c, T-N-full has the lowest value at all times. Next is Loose T-N-full, then T-full, and finally T-N-half. But for the total cost t, the owner of the lowest c does not incur the lowest cost. This time, Loose T-N-full performs the best, while T-full gives approximately the same cost as the former, despite its high value of c compared to others. One possible conclusion is that the high value of c does not mean bad performance. Again, sometimes, the cost of avoiding a collision is higher than letting it happen.

The ratio s/p is also listed in the table. Recalling the definition of position cost, it will be incurred only if a drone keeps staying out of the zero-cost area of the tube. And from the algorithm description where in Chapter IV, drones are allowed to be out of the tube when it is the only choice to avoid an incoming collision. Hence, a lower s/p value means that drones are more likely to move out of the tube to respond to a collision rather than to slow down or switch to the in-tube lanes. Furthermore, more drones are out of tube with increasing drone density.

Back to the data, Loose-T-N-full, T-full, and T-N-half have a similar behaviour of s/p when the drone density is low, while T-N-full gives the lowest value in that case. But algorithms T-full and T-N-half still have the

highest value for medium and high density. T-N-full shows a significant drop that its value is much smaller than the values of the two algorithms mentioned above. At the same time, T-N-full still stays last.

5.3.3 *Switching Count*

Continuing to understand how different algorithms can affect the switching activities, we record the actual switching count for each simulation. Results are shown in the Figure 5.14. Note that if a drone selects an out-tube lane to switch to, then based on our design, it might switch back into the tube even no upcoming collision is detected. In this case, we treat the switching count as two. Based on the graph, drones with algorithm T-full are expected to have the highest number of lane switching. Followed by Loose-T-N-full, T-N-half, and T-N-full. Not surprisingly, the switching count increases when we increase the drone density.

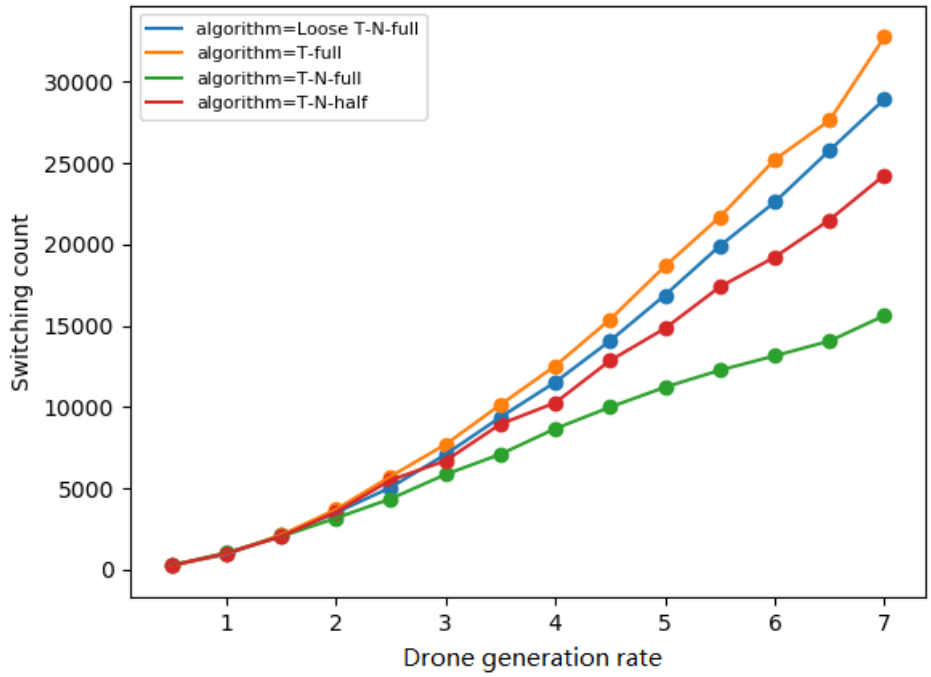

Figure 5.14: Switching count for different algorithm

The result presented in Figure 5.15, shows that when the drone generation rate is less than 4, switching count is not affected by path loss exponent, beacon interval, and transmit power. In other cases, using a lower path loss exponent, or shortening the beacon interval, or expanding the transmit power

can slightly increase the number of lane switching.

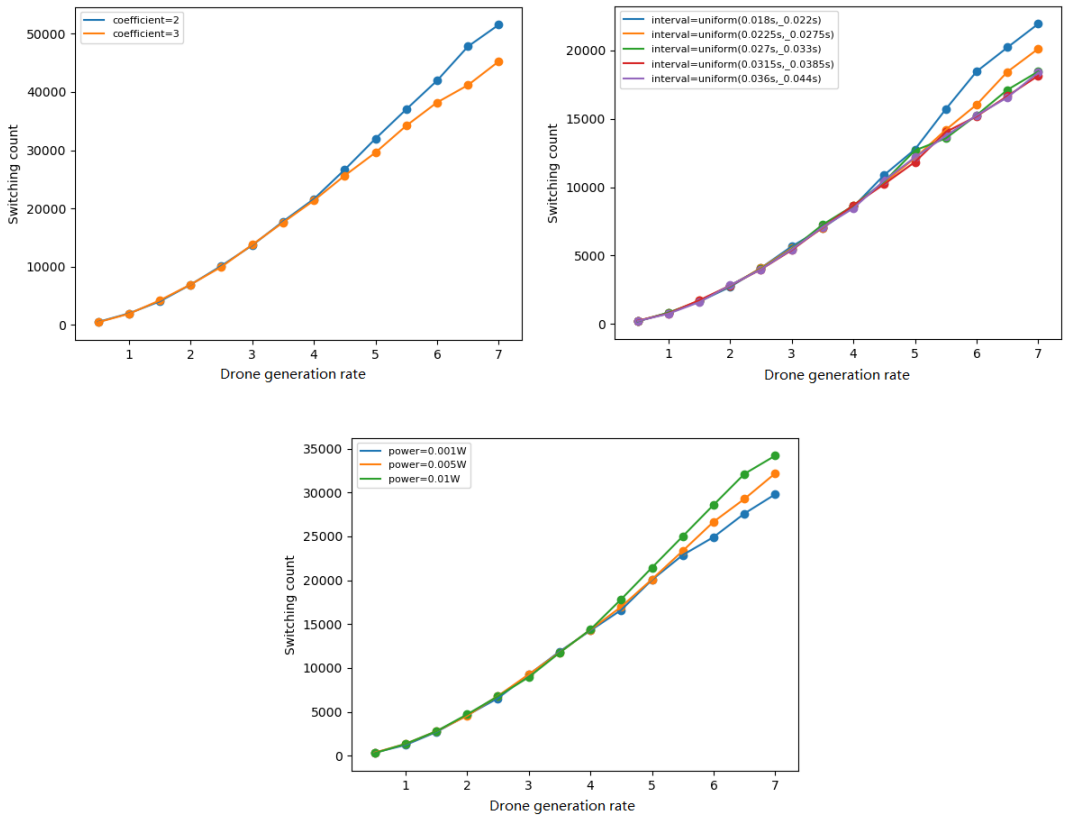

Figure 5.15: Switching count for different values of parameters for T-N-full

5.3.4 Packet Loss

Packet loss is the main reason for drone collisions in our project. Figure 5.16 shows the results for packet loss value.

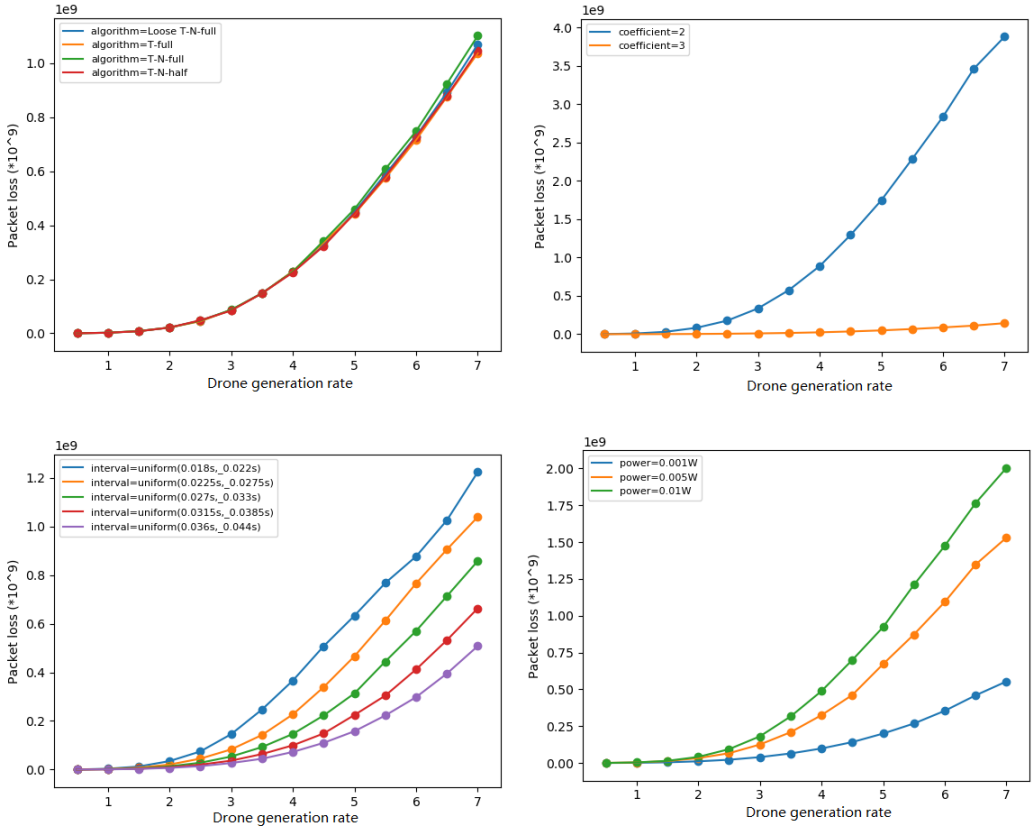

Figure 5.16: Packet loss for different values of parameters

From the figure, the algorithm can slightly affect the packet loss count. The reason is that different algorithms would cause different collision counts, then different collision counts lead to different total numbers of existing drones, which in turn affects the packet loss rate.

How these parameters affect the packet loss rate has been discussed in section 5.3.1. Here, we focus on comparing the trends between packet loss rate (Figure 5.16) and collision count (Figure 5.8, Figure 5.9).

The most obvious one is related to the beacon interval. In the packet loss rate Figure 5.16, the line trend is always superlinear. But for the collision

count, the value nearly does not change as long as drone generation rate $\leq$ 4[2]. And after that, it shows a faster growth speed than the packet loss figure. For other parameters, this phenomenon also exists.

That means that packet loss only slightly affects the collision count if the packet loss rate is lower than a threshold value. But, once the packet loss rate is higher than the threshold, it can significantly affect the collision count. The reason is that when the loss rate is low, drones can update safety information on time, the deviation between the received information and the actual situation is small, and drones make good decisions almost all the time. However, when the loss rate exceeds the threshold, drones cannot receive beacon packets from others. Hence, drones are more likely to make the wrong decision by using outdated data.

[2] In detail, when the beacon interval is 0.02s, the generation rate threshold is 4. When the beacon interval is 0.025s, the generation rate threshold is 5.5. For other cases, the collision count always grows very slowly

5.4 Runs in Extreme Conditions

Based on the last simulation, we can preliminarily infer that beacon interval, path loss exponent, and transmit power can significantly affect the total cost. At this stage, we also prefer to learn how our algorithm performs in one of the very bad situations (When beacon interval = 0.025s, path loss exponent = 2, transmit power = 0.1W).

5.4.1 Cost and Collision

In Figure 5.17, the total cost of all algorithms show a slow growth before the drone generation rate reaches 5.5, then the slopes of those lines become much larger. In addition, their performance is quite similar, but still with visible differences. Based on that, loose T-N-full is still the best algorithm in this situation.

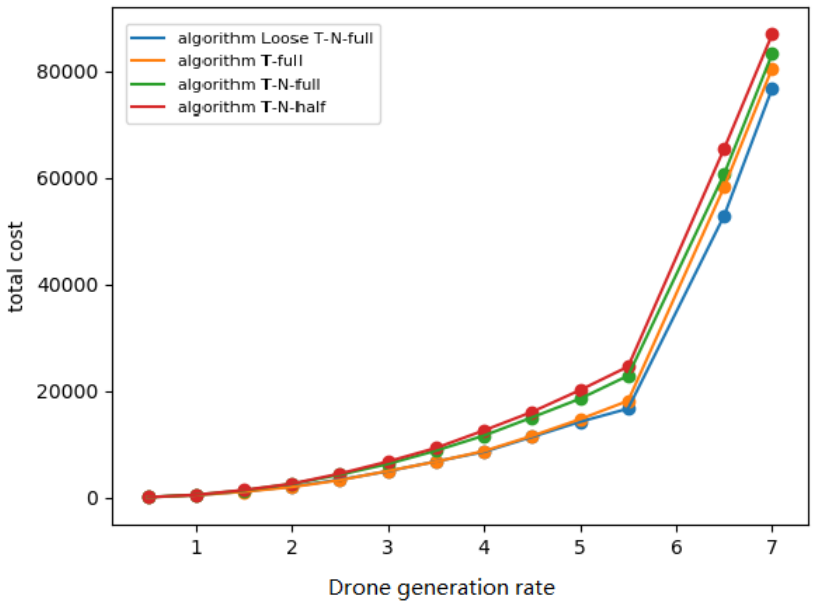

Figure 5.17: Total cost for four algorithms

In Figure 5.18, we present results for the position cost. They do not show anything surprising. Similar to Figure 5.13, T-N-full incurs the highest position cost, Loose-T-N-full and T-N-half have a similar performance, while T-full is always better than the others.

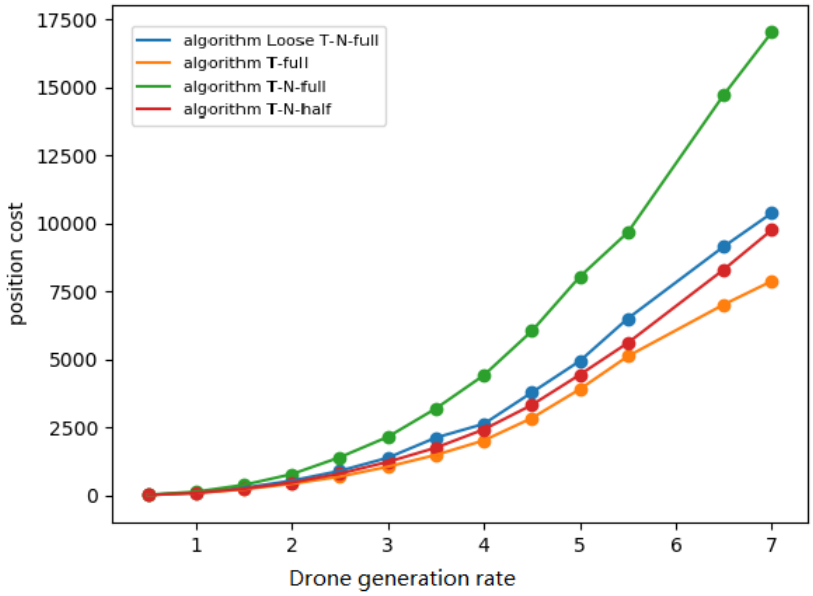

Figure 5.18: Position cost for four algorithms

In Figure 5.19, we can see that the speed cost does not increase all the time. It decreases when the drone generation rate becomes (1) 5 for T-full and Loose-T-N-full (2) 5.5 for T-N-full and T-N-half. The reduction of speed cost could due be able to (1) drones do no longer choose to slow down as they cannot "see" the drone ahead as a result of packet losses, or (2) more drones are destroyed so that the overall speed cost decreases.

In Figure 5.20, the increasing trends of lines are analogous to the lines in Figure 5.17. Collision cost has an explosive growth when the generation rate is larger than 5.5, since the packet loss rate starts to significantly influence the collision count at that time.

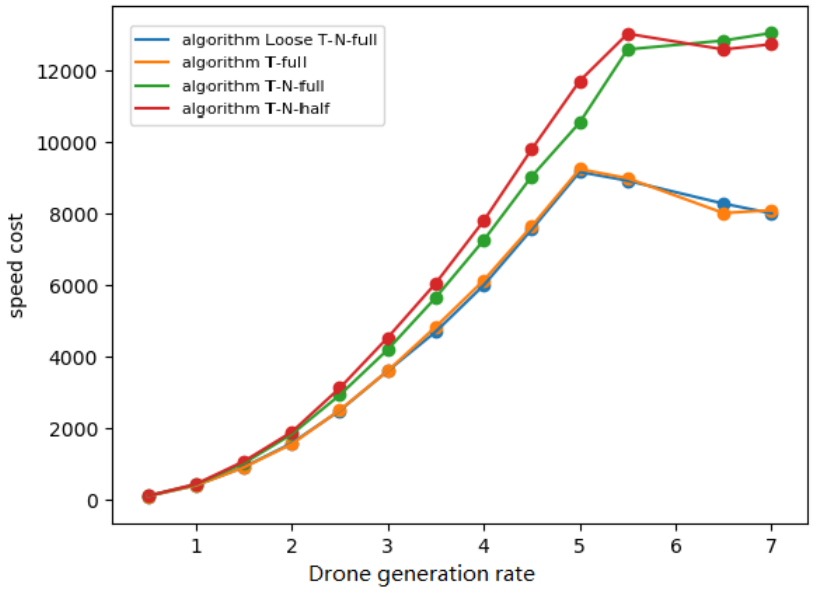

Figure 5.19: Speed cost for four algorithms

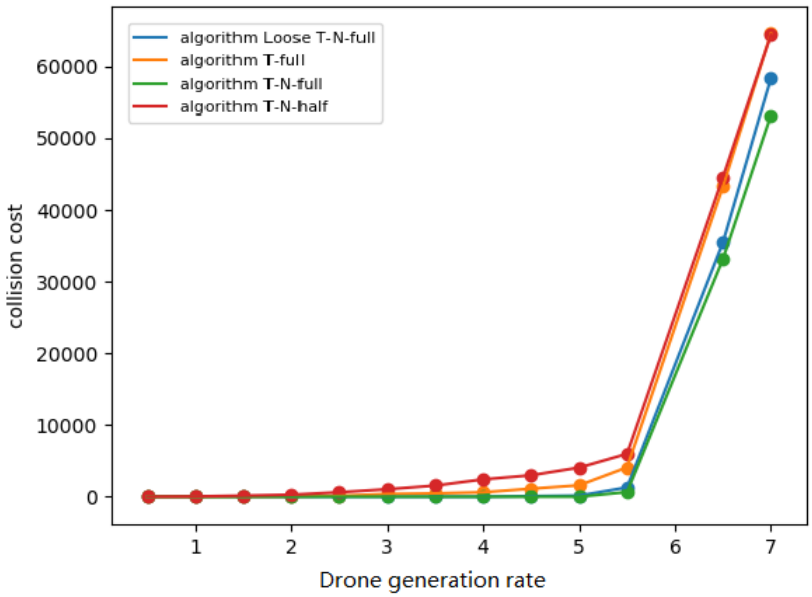

Figure 5.20: Collision cost for four algorithms

5.4.2 Oscillations

From the last section, we can find that most collisions happen when the drone tube is crowded. To learn more about this phenomenon, we run further simulations to collect data.

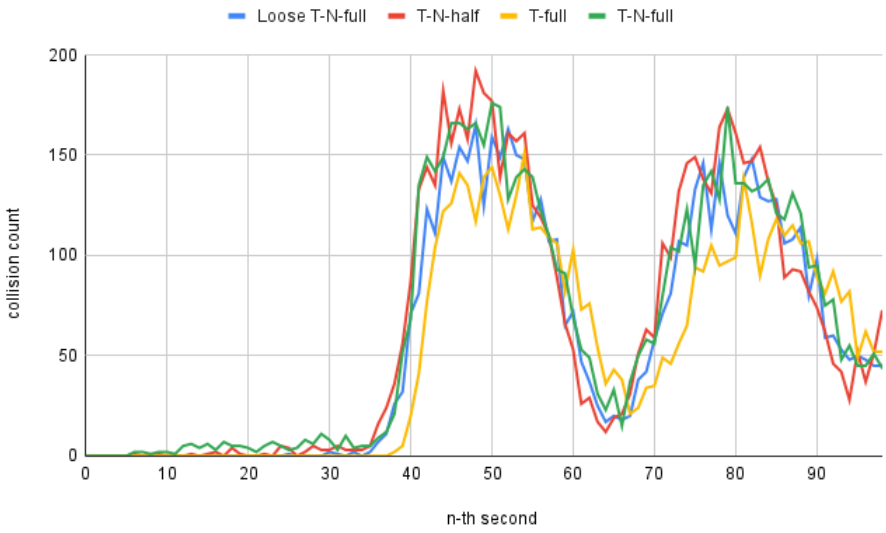

Figure 5.21: Collisions in the n-th second for four algorithms

In Figure 5.21, we count how many collisions happened in each second during the whole simulation. The values of y-axis represent the total number of collisions happened in the value of x-axis second for one hundred repetitions. Interestingly, two visible oscillations are shown between [40s,60s] and [70s,90s], and small oscillations exist everywhere in the line.

What's more, it seems that the amplitude of the second "big" oscillation is slightly smaller than the first one. To further investigate this conjecture, we extend the simulation time from 100 seconds to 400 seconds to see what the "big" oscillations look like in this situation. Since all algorithms show a similar shape, we only test Loose-T-N-full this time. The results are shown in Figure 5.22. Note that the y-axis is the average collision count for each repetition instead of the total collision count. Between 50 and 250 second, we can still recognize distinct oscillation of decreasing amplitude, and afterward things appear to become irregular that sort of stationary.

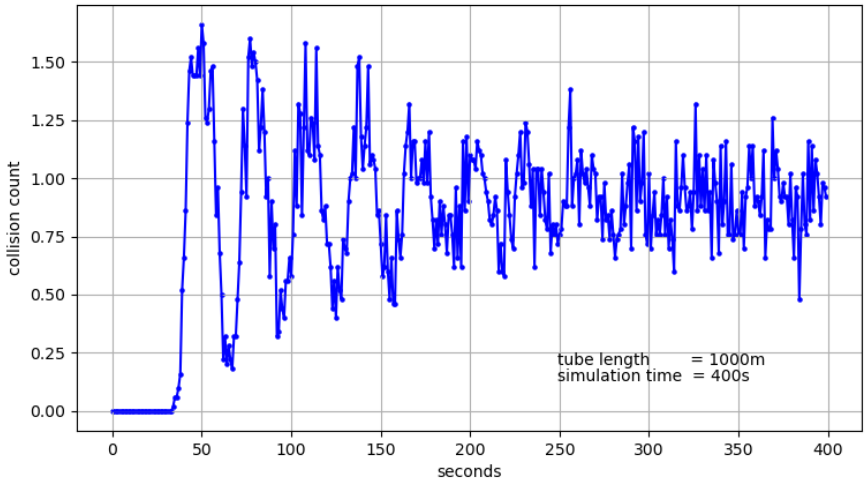

Figure 5.22: Collisions in the n-th second when simulation time = 400s

In Figures 5.23 and 5.24 show the packet loss count and node count in the n-th second, respectively. Oscillations also exist in both graphs, having a similar shape. The difference is that node counts always keep increasing between the start and the 50th second, whenever the packet loss count does not significantly grow before the 10th second.

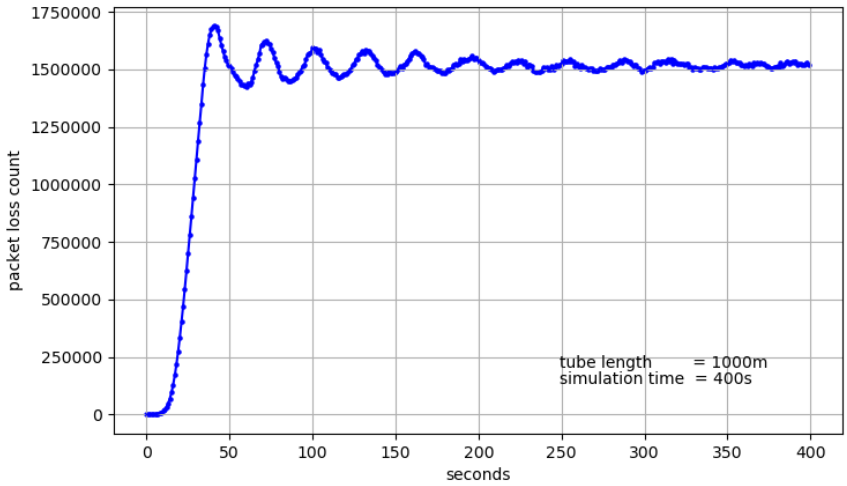

Figure 5.23: Packet loss counts in the n-th second when simulation time = 400s

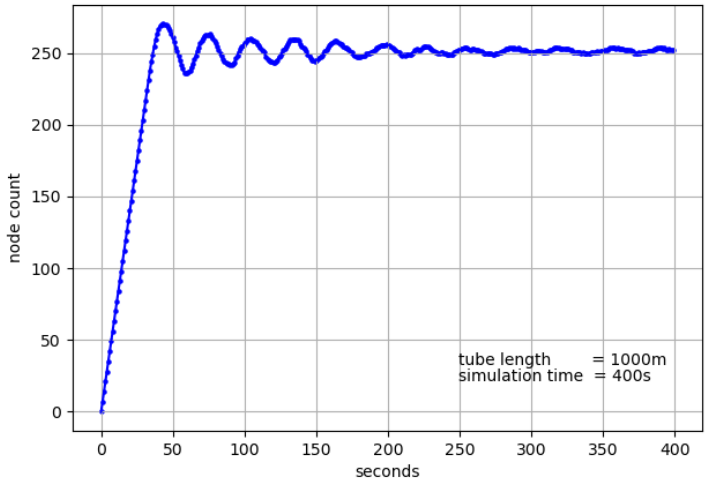

Figure 5.24: Node counts in the n-th second when simulation time = 400s

Another important parameter we have not changed before is the tube length. A longer length means that the drone tube can accommodate more drones simultaneously, which may affect the results of our simulations. Being inspired by Figures 5.22, 5.23, and 5.24, we run the simulations which increase the simulation time and tube length to see whether the oscillation still exists.

This time, oscillations are always irregular, and the amplitude is small. This shape is quite similar to the situation for the time between 250s and 400s in Figure 5.25. The node count and packet loss count show similar shapes.

In Figures 5.22, 5.23, and 5.24, we find that when the oscillation becomes irregular, the node and packet loss count do not change significant anymore. This phenomenon is also visible in Figure 5.25 from the 100th second to the end.

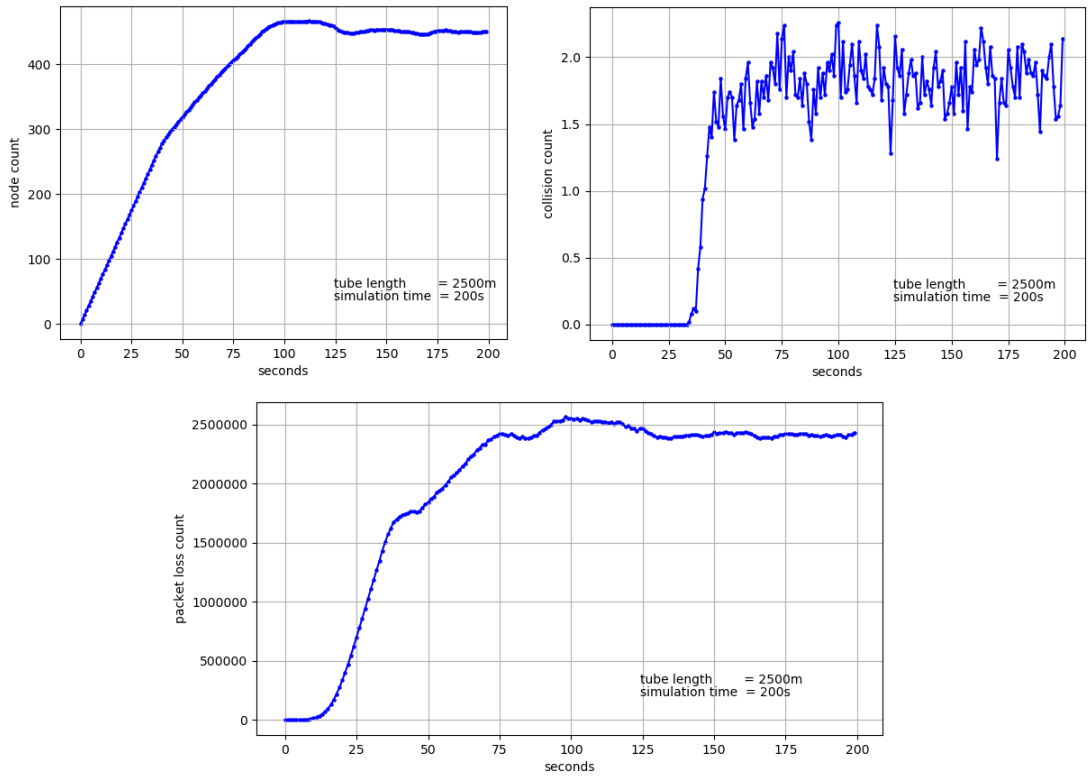

Figure 5.25: Node (upper left) Packet loss (bottom) Collisions (upper right) counts when simulation time = 200s and tube length = 2500m

Chapter VI

Discussion

In this chapter, we discuss the performance results of our algorithms in Section 6.1. We also briefly explain the oscillations observed in Section 5.4.2. In Section 6.2, we indicate the limitations of this study and suggest some future work.

6.1 Algorithm Performance

In this project, algorithm performance is measured by the value of total cost, including speed, position, and collision cost. Based on our preliminary study, our coordination-free algorithms can prevent many collisions. However, in an environment where the packet loss rate is very high, our algorithms can only avoid half of the incoming collisions. In this section, we discuss our four better algorithms and compare and summarize them.

From Figure 5.11, we cannot see a significant difference between the total costs of those algorithms when the drone generation rate is low. When we spawn more drones, we find that Loose T-N-full always has the best performance, and T-full has a slightly higher cost than Loose T-N-full. T-N-full and T-N-half are the third and fourth ones, respectively. Considering the collision count shown in Figure 5.7, one phenomenon is that algorithms with lower collision counts may not have the smaller total cost.

6.1.1 T-N-full

One case is the T-N-full algorithm. It always shows the smallest number of collisions, even avoiding all collisions when the drone generation rate is not high, but the total cost of it is surprisingly high. From Table 5.5 and Figure 5.13, we can see that its position cost and speed cost are relatively high. Especially the position cost are nearly twice as high than for other three algorithms. However, drones using this algorithm even have the smallest

number of lane switching. Based on the definition of T-N-full, being allowed to start a switching is quite hard for a drone since it must ensure no other drones in all neighbor lanes of the target lane exist in some range.

This strict criterion of the algorithm has two effects. The obvious one is that more drones are expected to slow down the speed to respond to incoming collisions. This will lead to a high speed cost of this algorithm. Another effect is that drones are more likely to choose the outer lanes for switching. That is because drones are generated and expected to stay in the zero-cost area, leading to the probability of finding a drone in a zero-cost area being much higher than finding it out of the tube. Hence the lanes which are out of the tube have more chance to satisfy the criteria. One example is shown in the following (see Figure 6.1). If the target lane is in tier 0, then all neighbor lanes are in the zero-cost area. If it is in tier 1, then three of six neighbor lanes are in the tube. When the target is in tier 2, only two of six neighbor lanes are in the tube, and these are likely more crowded. For this reason, drones are more likely to fly out of the tube. In addition, drones are also relatively hard to move into the zero-cost area since lanes in tube are quite crowded. This is why T-H-full has relatively high position cost.

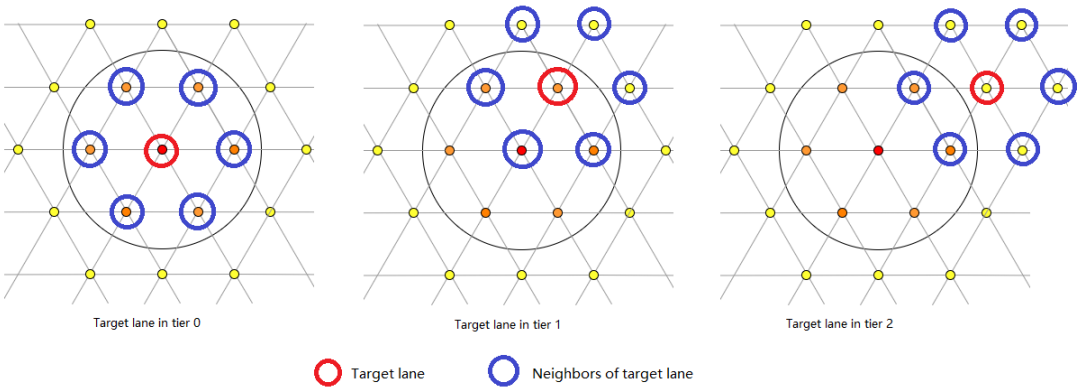

Figure 6.1: An example of why the probability of selecting outer lanes is higher

We summarize the features of T-N-full. Firstly, it is conservative in that the drone is only allowed to switch lanes when the nearby airspace totally has no other drones, and so the probability of having collisions by concurrency

should be small. But the disadvantage of the conservatism is that drones already out of the tube find it quite hard to move back into the tube. That causes the high position cost. If the coefficient/weight of position cost is small and keeping fewer collisions is more important, then this algorithm is expected to have a good performance.

6.1.2 T-full

T-full shows quite the opposite: a higher collision cost does not automatically lead to a high total cost. In Figure 5.7, the collision count of T-full is very high, but the total cost of it is nearly the lowest one. From the graph (Figure 5.14) showing the switching count, T-full shows the highest lane switching numbers. In T-full, drones only check circumstances on the target lane and do not consider any other lanes. On the one hand, the criteria for this algorithm is much looser that the drone can be encouraged to avoid collision by switching lanes, and hence leads to a low speed and position. On the other hand, simultaneous switching can happen quite often as drones do not care about who may also switch to the target lane.

From the results of position and speed cost in Figure 5.13, T-full causes the lowest position and speed cost at the same time. That is because more switching chances can shorten the total time of slowing down, and also shorten the waiting time when the drone is out of the tube. Thus both speed and position cost should be relatively low compared to other algorithms.

Considering the relationship between this algorithm's total cost and collision cost (in Figures 5.7 and 5.11), we can clearly see the trade-off between switching count, collision count, and total cost. Allowing more lane switching will lead to more collisions, but the cost of allowing those additional collisions is smaller than the cost of avoiding them. In other words, sometimes avoiding an incoming collision incurs a higher cost than letting it happen. But that conclusion only holds for our current coefficients/weights for different types of costs. If the collision cost coefficient becomes higher, this algorithm will not be considered as a good one.

6.1.3 T-N-half

The difference between T-N-half and T-N-full is that the former one checks only half of the safety range than the latter during the criteria check stage. Our purpose is to allow more lane switching while still restricting collisions caused by simultaneous lane switching. Our results show that this is a failed attempt, as this algorithm has the highest total cost, the most collisions, and the highest speed cost compared to other algorithms.

In this and the following sections, we define "simultaneous lane switching collision" as a collision type. This type of collision is caused by two drones in different lanes simultaneously selecting a same target lane to switch into. We also call it "concurrency."

We can learn an important point by considering the collision counts of those two algorithms (see Figure 5.7). The method for concurrency avoidance might be weak since the collision count of T-N-half is not lower than that of T-full (which actually did nothing for concurrency avoidance). Inspired by these, one conclusion is that having strict criteria on the target lane is a very important way to reduce the collision count. And with the success of T-full, another conclusion is that avoiding all concurrency will lead to a significant cost, and partially avoiding concurrency might not make things better. It could be an interesting topic of future work to combine those two algorithms, i.e., having strict criteria on the target lane and loose criteria on neighbor lanes.

One interesting thing is, the speed cost for this algorithm is unexpectedly higher than for the stricter version T-N-full. The speed cost would be significantly increased by either spending much time on slowing down or staying out of tube for a long time (since drones always fly with the maximum speed when it is out of tube). Considering this algorithm's low position cost, the reason for the high speed cost could be the former, as more drones may have to slow down by the sudden lane switching of others, since T-N-half has a loose check on the target lane. But with the current data, we cannot give a more precise explanation of this phenomenon.

Overall, this algorithm is the worst one, and it does not perform well in any situation. One possible optimization is to use stricter criteria on the

target lane.

6.1.4 Loose T-N-full

T-N-full is the best algorithm from our simulation results. This is because it has the lowest total cost and the lowest collision count. The difference between Loose T-N-full and T-N-full is that the former one can ensure the drone, which has the fastest speed within a certain range, can always start lane switching. So there should be more switching happening compared to the most strict algorithm T-N-full. Moreover, this loose check also ensures only one drone can change lane at the same time, which can be helpful for concurrency avoidance.

There is one potential problem with this algorithm. When the fastest drone does not need to change lane, other drones, who need, must wait for the fastest one to go far away. Based on that, one possible optimization is that the comparison pool only includes drones that potentially need to switch in the near future when comparing who is the fastest.

6.1.5 Comparison the Four Algorithms

In Table 5.5, we give additional cost information to compare the performance between the four algorithms. The collision cost of T-N-full and Loose T-N-full are zero when the generation rate is not too high. These two algorithms are good at collision avoidance when the packet loss rate is low.

The performance of our algorithms changes as the drone generation rate change. The obvious point is the ratio of speed cost to position cost. This ratio keeps decreasing when we increase the drone generation rate. If more drones exist in the simulation, they become more likely to fly out of the tube to avoid a collision. For a fixed drone generation rate such as 1, 4, and 7, the algorithms with higher s/p (in Table 5.5) tend to allow drones to select outer lanes rather than inner lanes.

With respect to the total cost value, the best algorithm Loose T-N-full is just about 11% better than the worst one (T-N-half) for a low drone density. This indicates that either loose or strict criteria will lead to a similar impact on the overall performance. This is mostly caused by the low probability of

facing concurrency problems. Drones need not wait or slow down to prevent concurrency. But when the drone generation rate is high (=7), the difference increases to 40%. This time, the ability to solve high concurrency is quite important. Loose-T-N-full performs the best in balancing the switching count and collision count. T-full is also good but it leads to a high destruction rate, as it encourages drones to switch lanes and does not care about concurrency at all. T-N-full is conservative, it is designed to avoid any simultaneous lane switching, thus leads to the smallest collision cost, and highest speed and position costs. While T-N-half is the worst algorithm, it fails to balance cost and collision by using looser criteria. However, when the values of simulation parameters are extreme (in Section 5.4), all algorithms perform very bad. That is because the packet loss rate is high enough, drones only have a very small probability of hearing information from others. In that case, our algorithms have limited uses for collision avoidance.

6.1.6 Explanation of Oscillation

The oscillation is the most interesting finding in our simulations. In our expectation, there should be only the oscillation with a small amplitude shown in collision count vs. n-th second graph (see Figure 5.22). One reason for that is that the number of repetitions is not large enough. And another is that a high collision count in the last second will slightly decrease the packet loss rate in the next second. Hence, the collision count in the next second might show a subtle change. However, our results also show big oscillations, in which the max/min ratio can even be 600% (1.5 vs. 0.25) and the period is around 25 seconds. The amplitude of these big oscillations becomes smaller as time progresses. In this section, we discuss these phenomena and give our explanation.

Before we explain the oscillation, we first discuss the dynamic balance between collisions and drone counts. We spawn drones and release them into the tube continuously based on our system model. When the environment is extreme (i.e., short beacon interval, high transmit power), many drones suffer collisions and are destroyed. During this process, we have the following loop.

Under extreme conditions, packets sent at the start of the tube can inter-

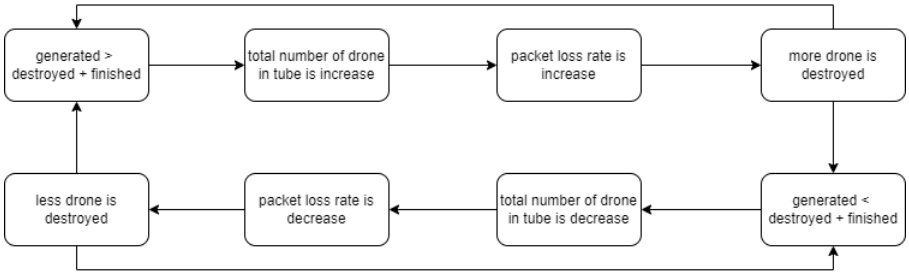

Figure 6.2: Dynamic balance between collision count and drone count

fere with the drone at the end of the tube, thus increasing the drone count should negatively affect the packet loss rate. Then, in one second, if the number of the generated drone is larger than the number of destroyed drones plus the number of drones that reached the end of the tube (we use "finished" to represent these in Figure 6.2), the total number of drones in tube would increase, leading to an increase of packet loss rate. Consequently, more drones are destroyed. If the total number of drones is still increasing, this process will continue. Oppositely, if the total number of drones decreases, it would finally cause fewer drones to be destroyed. This loop should happen continuously until the total number of drones in the tube does not change.

We hypothesize that there are three factors in our simulation related to the big oscillations, they are:

- First of all, the effect of packet loss on collisions has some delay. For a rear-end collision, the length of the delay depends on the distance (when communication is lost) between the two related drones. For a simultaneous lane switching collision, it depends on the overtaking speed (introduced in Section 3.2).

- Secondly, a threshold of packet loss rate exists in our simulation. Recall from Section 5.3.4, when the packet loss rate is lower than the threshold, it only slightly affects the collision count. But, once it is higher than the threshold, the collision count would significantly increase. Hence, before the packet loss rate reaches the threshold value, only a small number of drones are destroyed.

- Thirdly, based on our simulation setting in Table 5.1, the average time for a drone to reach the end of the tube is 40 second.

Due to the third factor, the total number of drones should keep increasing before the 40th second if no collisions happen. However, with the increase of drone count, the packet loss rate also increases. At a certain time (approximately the 35th[1] second in Figure 5.23), the packet loss rate becomes larger than the threshold and start to significantly affect the collision count. But due to the impact of delay we mentioned above, no collisions will happen immediately when the packet loss rate first grows too large. It leads to both the total number of drones and packet loss rate still keep increasing while at this stage (35th to 40th second in Figure 5.23).

At another certain time (40th second in Figures 5.21 and 5.22), many drones start to collide with others due to the high packet loss rate some seconds before, and many drones reach the end of the tube. Consequently, the total number of drones shows a sudden sharp drop. Therefore, the balancing mechanism shown in Figure 6.2 is enabled. Furthermore, since a large number of drones is destroyed at the same time, the packet loss rate will be reduced to a relatively low value sometime later. This leads to a significant decrease in collision count. Hence the oscillation has started a new cycle until the negative feedback mechanism reaches the balance.

There is another possibility for starting an oscillation. Based on our algorithm design, a drone is expected to slow down when there is no suitable lane to switch into. For the extreme parameter set, the tube is crowded, and most drones can not find an available lane. Therefore, many drones will keep slowing down and then block each other. This situation is shown in Figure 6.3 below. In such situation, a significant drop in drone numbers can also happen when this big group of drones reaches the end of the tube. And due to the sudden decrease, the adjustment mechanism will start also.

When we extend the length of our tube, the result is quite different. In this further simulation, we set the tube length from 1000m to 2500m, and this change makes drones need 100 seconds to reach the end of the tube.

[1] This is because the drone count at this time is equal the to final "converged" value in Figure 5.24

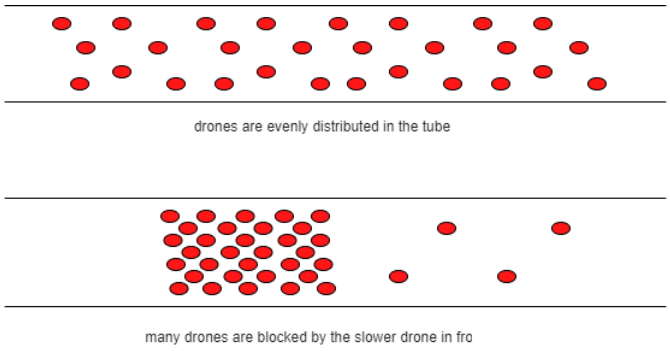

Figure 6.3: Drone jam due to the slower drones in front

Hence, at the 40th second, drones count would keep increasing (but with a smaller slope due to collisions) instead of starting to decrease. Furthermore, the interference range of a drone is smaller than the new tube length. Thus, a newly generated drone cannot interfere the firstmost generated drones after a certain time[2] we start the simulation. After that time, when more drones are generated, the packet loss rate will not increase anymore. In other words, after the drone count is larger than a threshold value, the packet loss rate would not change.

At the 100th second, drones start to reach the end of the tube, leading to the decrease of the slope of the drone count. However, the packet loss rate does not change since the drone count is still larger than the threshold we mentioned above. Based on the dynamic balancing mechanism shown in Figure 6.2, the number of collisions that happen in one second would not change if the packet loss rate is fixed. Therefore, the node count would show a "monotonically" smooth change until it is smaller than the threshold. Finally, the big oscillations would not appear as there is no sudden change of node count during the simulation, only irregular oscillations exist for some time after the simulation starts running. Those irregular oscillations either appear in Figure 5.25 or Figure 5.22, and we think they are mainly caused by random fluctuations and small number of repetitions.

Overall, oscillations only appear to exist in extreme environments. It is

[2] We strongly suspect that it is the 75th second, as the packet loss count firstly decreases after the 40th second in Figure 5.25.

important to note that based on the data we collected, we cannot fully prove the correctness of our explanation for the oscillation. But the oscillations clearly suggest that there might be a balancing mechanism between the collision count and node count in a small area. That could be an interesting topic to explore in the future.

6.2 Limitations and Future Works

In this section, we discuss the limitations of this project, and then we give our suggestions for future works.

6.2.1 Cost Normalization Problem

One hidden parameter that can significantly affect the final cost in our cost model is tube length. For the collision cost, a longer tube can increase the possibility of having more collisions. Then more collisions will lead to a higher collision cost. For speed and position cost, if we fixed other parameters, the final values for two thousand meters should be approximately two times larger than those for one thousand meters. To make cost values more easily understandable, it makes sense to normalize them, for example to one kilometer.

6.2.2 The Domain of Parameters

One purpose of our project is to learn how simulation parameters affect the result for the total cost. Due to a limitation of time available for simulations, we have restricted the range of simulation parameters. An example is the transmit power. We only tested the three values 0.001W, 0.005W, and 0.01W. But the results did not satisfy our expectations, and we are interested to see the expected negative influence of lowering the transmit power. Since sending beacon packets with a low transmit power may only lead neighbors in too small of a range. Another case is the beacon interval, our suspicion is that the collision count will become large when the interval is too long or too short. The results only showing longer intervals have this kind of impact, shortening the interval can be beneficial in optimizing the cost, up to the point where the wireless channel becomes too congested and packet loss rates rise again.

So, the domain of our parameters in this project is not sufficient to show the trade-off of those parameters. Our discussion of the effect of parameters is necessarily limited by that.

6.2.3 Simple Traffic Model

We have a simple traffic model for the communication of drones. That is, drones only send beacon packets during the whole simulation. On the one hand, this assumption can make the system model simple to design and implement. And the final total cost can clearly show the use and effect of the beacon packets.

But on the other hand, there should be many other types of packets, such as packets used for applications. This can also influence the simulation result. One obvious effect is that they add further interference for beacons, meaning that beacons will be less likely to be received by other drones, and more collisions might happen. To explore the application of our coordination-free algorithm, it needs more realistic traffic in future works.

6.2.4 Negotiation Between Drones

Our coordination-free algorithms are designed to use no negotiation at all. That means drones are not allowed to share their overtaking plan with others, nor to give others a signal when starting a lane switching. The advantage of totally coordination-free algorithms is that they still work when drones meet some problems when negotiating with others. Hence, our coordination-free algorithms can be seen as a baseline, and we expect that other coordination-based algorithms should perform better than that baseline.

In reality, in the context of the ground vehicles, there always exists some negotiation between drivers. One example is the brake light; even if the driver does not want to tell the intention of braking to cars behind, those cars can still recognize the incoming braking. Another example is the blinker. In the future, the beacon packet of drones might always include some "negotiation" information in it.

6.2.5 Future Works

Based on our discussion above, we now give some suggestions for future work.

Regarding the drone road model:

- Firstly, attempt to increase the drone density, which by necessity requires more lanes in the tube. Our current setting only allows us to generate up to seven drones each time so that the drone density is limited. Many more drones can be set in the tube with a wider zero cost area. Hence, interference should be more prominent. Future work can focus on finding the trade-off between increasing the drone density and controlling the collision rate, or focus on finding a good value set of simulation parameters for higher density environments.

- Secondly, another possible research direction is to consider removing the sub-tube or lanes in the tube. That means drones are no longer restricted to lanes, and they can stay anywhere in the section area of the tube. Drones are expected to be more disordered, and more collisions may happen. In addition, removing the concept of lanes will make it harder for drones to switch the "lane," especially finding a suitable position in the section area of the tube. Since there are only six choices in our current model, computational overhead is quite manageable. But without lanes, there are infinitely many choices, leading to the challenge of optimizing the selection. Furthermore, the concurrency switching problem will be much more complicated as all drones may select a particular position to switch into. Future studies may focus on the challenges we list above and then design an algorithm that applies to that case.

Regarding the communication model, we suggest the following:

- Expand the content of the beacon packet to include some kind of negotiation information (such as the switching plan or the turning signal). Then the lane switching algorithms can be re-designed to use the "coordination" messages to simplify the criteria check or further optimize the performance.

- Consider adding some background packets. Those packets are transmitted in addition to the beacons that drones send. With the interference generated by those packets, the number of beacon packets lost should be much higher than in the current setting. And correspondingly, the values for simulation parameters to minimize the collision rate should be different.

For the cost model, we encourage to normalize the three types of cost and use a unit like cost per kilometer instead of using the total cost. Based on this normalization, we can find a standard to tell whether a value of total cost is considered perfect, good, normal, or bad. Another suggestion about the cost model is to add additional types of cost, such as a penalty for multiple collisions in a short time.

We suggest expanding the domain of our simulation parameters for the simulations, especially for the transmit power and beacon interval. This can be helpful to further explore the effect of those parameters on performance. In addition, the "best value" that minimizes the total cost might be found within a wider domain. Those optimal values can be valuable for applying the algorithm to reality. What's more, other parameters, such as coding and modulation scheme, can also be added to the simulation.

In Section 6.2.1, we mentioned that it is hard for drones to move back to the tube when the drone density is high and the criteria are strict. Our current strategy is that a drone must choose a lane with a smaller tier than it is currently flying in to move back into the tube. That strategy is simple, but it also restricts the route. The drone cannot take a detour. So, one suggestion for future works is to explore a method for a drone to move into the tube at a lower cost.

Another suggestion for algorithm design is to implement the adaption of beaconing rate and transmit power, which means drones can be able to adapt the related parameters to lower the packet loss rate. Drones may find the optimal of those parameters during the adaption, and the collision count is like to be reduced.

A final piece of possible future work is to learn more about the oscillation phenomenon we observed in Section 6.4.2. Based on our results, this phe-

nomenon happens within a "small" area in which drones can significantly interfere with others. One aim for the future is to find the relationship between the drone density and the maximum range of the oscillation area. Another is to explore the potential effect of this phenomenon. Finally, in our discussion, we did not give a confident explanation of why big oscillation will disappear eventually. This could also be investigated in future works by collecting more data such as the number of arrived drones per second or the delay between losing communication and collision.

Chapter VII

Conclusion

This project aims to design a lane switching algorithm for drones in a tube road, which is a long straight air space. Drones are only allowed to share basic position and speed information with each other. One key problem our algorithms should be able to overcome is to avoid collisions. Based on the literature review, one important reason for drone collisions is packet loss. That is, drones fail to update the safety information for others and cannot recognize an incoming collision. A key purpose for our project is to explore how certain key parameters can affect the performance of our algorithms. We are interested in the parameters which can affect the packet loss rate by influencing the probability of direct or hidden-terminal packet collisions. These parameters include drone density, beacon packet interval, transmit power, and path loss exponent for the log-distance channel model.

To simplify the problem and be able to implement our algorithm in a programming language, we present a system model to give several assumptions about the drone, the tube road, the communication, and the cost. We have used OMNet++ as the simulation software in our project. It can simulate most events happening in wireless communications, including packet loss, which is quite important in our scenario. We have designed a "big" simulation campaign to investigate the effect of simulation parameters on the performance of our algorithms.

In the remainder of this chapter we discuss our responses to the research questions, starting with that one:

- ***RQ1***: *Can our coordination-free algorithms perform better than the baseline? Do they have any limitations?*

We designed eight algorithms and compared them to evaluate their per-

formance. Those eight algorithms include two baseline algorithms which are "do-nothing" and "always slow down." The other six algorithms are similar but with different criteria for starting lane switching. Based on a preliminary study, we have seen that all algorithms except the naive one show a significant advantage in avoiding collisions and lowering the total cost. Two of our criteria-based algorithms have similar performance compared to the "always slow down" algorithm, and the other four algorithms perform equally well in this preliminary study. So we can say that four of our coordination-free algorithms perform better than the baseline, and they can help to reduce the collision count and total cost.

In our simulation results, Loose T-N-full always performs better than all the other algorithms, whereas T-full and T-N-full can also be considered feasible solutions in some situations. T-full has the highest switching count and the lowest speed and position costs. However, it is not able to avoid collisions that are caused by simultaneously multiple lane switching (we call it "concurrency"), which gives it the second most collision count among all algorithms. T-N-full gives the lowest collision count, and if we were to increase the coefficient for the collision cost in our total cost function, it is potentially the best algorithm. For T-N-half, the results show it does not have any advantages compared with other algorithms, but is still a reasonable attempt to solve the concurrency problem. One common limitation of those algorithms is that they can prevent most collisions only if the packet loss rate is low. A potential reason is that our algorithms do not predict the trajectories of other drones well enough. When the packet loss rate is high, the knowledge of neighbored drones can be quite inaccurate. Hence drones cannot avoid incoming collisions. Another common limitation is related to the concurrency problem, our algorithms cannot solve it in a good way. T-N-full and Loose T-N-full can somewhat prevent collisions caused by concurrency, but they also limit the choice during lane switching, and slowing down becomes the only one option in many cases.

Regarding the second research question:

- ***RQ2****: What factors (drone density, transmit power, etc.) have significant impact on performance, and how do they affect it? Can we explain*

that?

Overall, all of the factors considered can significantly affect the final performance in our scenario. All parameters influence the total cost by affecting the packet loss rate. With the increase of drone density or the reduction of the beacon interval, there should be more packets transmitted at the same time. Hence, the probability of packet collisions will be higher, leading to a greater difficulty for drones to recognize an incoming collision and respond to it. Transmit power and path loss exponent can affect the interference seen by packets and thus impact the communication range and packet loss rate.

Regarding the last research question:

- ***RQ3****: Can we find a "genie" solution (i.e. no collision guarantee) for the problem?*

In reality, the answer is no since packet loss should always exist. Even though some of our algorithms are designed to avoid collisions in a low packet loss rate environment, some collisions can still happen with small probability events (e.g., key beacon packets are continuously lost). Even for the most simple algorithm, always slow down, collisions are not fully avoided in the presence of the hidden-terminal packet collisions. However, when the packet loss rate is not too high (details in Figure 5.10), most algorithms can ensure that more than 95% of drones will not be destroyed by collisions. And when the drone density is small (drone generation rate is less than 4.5), T-N-full and Loose T-N-full can almost avoid collisions.

One interesting finding in our simulations is the existence of big oscillations for the collision count. This phenomenon happens when drones at the start of the tube start to interfere with drones further down the tube. In our inference, when the interference power is very high, there might be a dynamic feedback mechanism to balance the collision count and drone count. In our view, this phenomenon is probably caused by the time leg between a surge of packet losses and resulting collisions. But this hypothesis needs more evidence and data to prove in future works.

References

[1] Na Zhang, Hu Liu, Bing Feng Ng, and Kin Huat Low. Collision probability between intruding drone and commercial aircraft in airport restricted area based on collision-course trajectory planning. *Transportation research part C: Emerging Technologies*, 120:102736, 2020.

[2] Jeonggyun Yu and Sunghyun Choi. Performance Comparison of Dual Queue and EDCA for VoIP over IEEE 802.11 WLAN. In *11th European Wireless Conference 2005 - Next Generation wireless and Mobile Communications and Services*, pages 1–7, Nicosia, Cyprus, 2006.

[3] Rajeswar Reddy G and Ramanathan R. An Empirical study on MAC layer in IEEE 802.11p/WAVE based Vehicular Ad hoc Networks. *Procedia Computer Science*, 143:720–727, 2018. 8th International Conference on Advances in Computing & Communications (ICACC-2018), Rajagiri School of Engineering & Technology, Kochi, India.

[4] Wuwen Lai, Wei Ni, Hua Wang, and Ren Ping Liu. Analysis of average packet loss rate in multi-hop broadcast for vanets. *IEEE Communications Letters*, 22(1):157–160, 2018.

[5] Salvador Gonzalez, Victor Ramos, and MoonBae Song. A simulation-based analysis of the loss process of broadcast packets in wave vehicular networks. *Wirel. Commun. Mob. Comput.*, 2018, jan 2018.

[6] S Rappaport Theodore et al. Wireless Communications: Principles and Practice. In *Upper Saddle River*. Prentice Hall, 2002.

[7] John F Keane and Stephen S Carr. A brief history of early unmanned aircraft. *Johns Hopkins APL Technical Digest*, 32(3):558–571, 2013.

[8] Eric Cheng. *Aerial photography and videography using drones*. Peachpit Press, San Francisco, USA, 2015.

[9] Alexandru-Ionut Siean, Radu-Daniel Vatavu, and Jean Vanderdonckt. Taking that perfect aerial photo: A synopsis of interactions for drone-based aerial photography and video. In *ACM International Conference on Interactive Media Experiences*, pages 275–279, New York, NY, USA, 2021.

[10] AK Puttock, AM Cunliffe, K Anderson, and Richard E Brazier. Aerial photography collected with a multirotor drone reveals impact of eurasian beaver reintroduction on ecosystem structure. *Journal of Unmanned Vehicle Systems*, 3(3):123–130, 2015.

[11] Man Liang and Daniel Delahaye. Drone fleet deployment strategy for large scale agriculture and forestry surveying. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4495–4500, Auckland, New Zealand, 2019. IEEE.

[12] Elisabetta Raparelli and Sofia Bajocco. A bibliometric analysis on the use of unmanned aerial vehicles in agricultural and forestry studies. *International Journal of Remote Sensing*, 40(24):9070–9083, 2019.

[13] Ashley Varghese, Jayavardhana Gubbi, Hrishikesh Sharma, and P Balamuralidhar. Power infrastructure monitoring and damage detection using drone captured images. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1681–1687, Anchorage, AK, USA, 2017. IEEE.

[14] Francesco Flammini, Concetta Pragliola, and Giovanni Smarra. Railway infrastructure monitoring by drones. In *2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*, pages 1–6, Toulouse, France, 2016. IEEE.

[15] Arthur P Cracknell. Uavs: regulations and law enforcement. *International Journal of Remote Sensing*, 38(8-10):3054–3067, 2017.

[16] Doina Popescu Ljungholm et al. Regulating government and private use of unmanned aerial vehicles: Drone policymaking, law enforcement deployment, and privacy concerns. *Analysis and Metaphysics*, 18:16–22, 2019.

[17] Raissa Z. B. Bravo, Adriana Leiras, and Fernando L. Cyrino Oliveira. The Use of UAVs in Humanitarian Relief: An Application of POMDP-Based Methodology for Finding Victims. *Production and operations management*, 28(2):421–440, 2019.

[18] Thomas M. Solvin, Stefano Puliti, and Arne Steffenrem. Use of UAV photogrammetric data in forest genetic trials: measuring tree height, growth, and phenology in Norway spruce (Picea abies L. Karst.). *Scandinavian journal of forest research*, 35(7):322–333, 2020.

[19] Faine Greenwood, Erica L. Nelson, and P. G. Greenough. Flying into the hurricane: A case study of UAV use in damage assessment during the 2017 hurricanes in Texas and Florida. *PloS one*, 15(2):e0227808–e0227808, 2020.

[20] Aline Karak and Khaled Abdelghany. The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, 102:427–449, 2019.

[21] Judy Scott and C. Scott. Drone delivery models for healthcare. In *Proceedings of the 50th Hawaii international conference on system sciences*, page 3297–3304, Hilton Waikoloa Village, Hawaii, 01 2017. IEEE.

[22] Youngmin Choi and Paul M Schonfeld. Optimization of multi-package drone deliveries considering battery capacity. In *Proceedings of the 96th Annual Meeting of the Transportation Research Board, Washington, DC, USA*, pages 8–12, 2017.

[23] Tim Laseter, Andrew Tipping, and FRED Duiven. The rise of the last-mile exchange. *PWC Strategy+ Business*, 92:30, 2018.

[24] J. N. Yasin, S. A. S. Mohamed, M. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila. Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches. *IEEE Access*, 8:105139–105155, 2020.

[25] Myungwhan Choi, Areeya Rubenecia, Taeshik Shon, and Hyo Hyun Choi. Velocity obstacle based 3D collision avoidance scheme for low-cost micro UAVs. *Sustainability*, 9(7):1174, 2017.

[26] Mingrui Lao and Jun Tang. Cooperative multi-UAV collision avoidance based on distributed dynamic optimization and causal analysis. *Applied Sciences*, 7(1):83, 2017.

[27] Andras Varga. Omnet++. In Wehrle Klaus, Günes Mesut, and Gross James, editors, *Modeling and tools for network simulation*, pages 35–59. Springer, Berlin, Heidelberg, 2010.

[28] Andreas Köpke, Michael Swigulski, Karl Wessel, Daniel Willkomm, PT Klein Haneveld, Tom EV Parker, Otto W Visser, Hermann S Lichte, and Stefan Valentin. Simulating wireless and mobile networks in omnet++ the mixim vision. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, Marseille France, 2008.

[29] Antonio Virdis, Giovanni Stea, and Giovanni Nardini. Simulte-a modular system-level simulator for lte/lte-a networks based on omnet++. In *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications (SIMULTECH)*, pages 59–70, Vienna Austria, 2014. IEEE.

[30] Xiaodong Xian, Weiren Shi, and He Huang. Comparison of omnet++ and other simulator for wsn simulation. In *2008 3rd IEEE Conference*

on Industrial Electronics and Applications, pages 1439–1443, Singapore, 2008. IEEE.

[31] Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. Drone technology: Types, payloads, applications, frequency spectrum issues and future developments. In *The future of drone use*, pages 21–45. Springer, The Hague, 2016.

[32] Hyunsoo Yang, Yongseok Lee, Sang-Yun Jeon, and Dongjun Lee. Multi-rotor drone tutorial: systems, mechanics, control and state estimation. *Intelligent Service Robotics*, 10(2):79–93, 2017.

[33] Haowei Gu, Ximin Lyu, Zexiang Li, Shaojie Shen, and Fu Zhang. Development and experimental verification of a hybrid vertical take-off and landing (vtol) unmanned aerial vehicle (uav). In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 160–169, Miami, FL, USA, 2017. IEEE.

[34] Ugur Ozdemir, Yucel Orkut Aktas, Aslihan Vuruskan, Yasin Dereli, Ahmed Farabi Tarhan, Karaca Demirbag, Ahmet Erdem, Ganime Duygu Kalaycioglu, Ibrahim Ozkol, and Gokhan Inalhan. Design of a commercial hybrid VTOL UAV system. *Journal of Intelligent & Robotic Systems*, 74(1):371–393, 2014.

[35] Lawrence J Fennelly and Marianna A Perry. Unmanned aerial vehicle (drone) usage in the 21st century. In *The Professional Protection Officer*, pages 183–189. Elsevier, Naples, FL, 2020.

[36] Graham Wild, John Murray, and Glenn Baxter. Exploring civil drone accidents and incidents to help prevent potential air disasters. *Aerospace*, 3(3):22, 2016.

[37] Chai-Keong Toh. Associativity-based routing for ad hoc mobile networks. *Wireless personal communications*, 4(2):103–139, 1997.

[38] Morteza M Zanjireh, Ali Shahrabi, and Hadi Larijani. Anch: A new clustering algorithm for wireless sensor networks. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 450–455, Barcelona, Spain, 2013. IEEE.

[39] Francisco J Martinez, Chai-Keong Toh, Juan-Carlos Cano, Carlos T Calafate, and Pietro Manzoni. Emergency services in future intelligent transportation systems based on vehicular communication networks. *IEEE Intelligent Transportation Systems Magazine*, 2(2):6–20, 2010.

[40] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[41] Muhammad Asghar Khan, Alamgir Safi, Ijaz Mansoor Qureshi, and Inam Ullah Khan. Flying ad-hoc networks (fanets): A review of communication architectures, and routing protocols. In *2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, pages 1–9, Karachi, Pakistan, 2017.

[42] Sourangsu Banerji and Rahul Singha Chowdhury. On ieee 802.11: wireless lan technology. *arXiv preprint arXiv:1307.2661*, 2013.

[43] Daniel Jiang and Luca Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008-IEEE Vehicular Technology Conference*, pages 2036–2040. IEEE, 2008.

[44] Abdeldime MS Abdelgader and Wu Lenan. The physical layer of the ieee 802.11 p wave communication standard: the specifications and challenges. In *Proceedings of the world congress on engineering and computer science*, volume 2, pages 22–24, San Francisco, USA, 2014.

[45] Laura Michaella B Ribeiro and Leandro Buss Becker. Performance Analysis of IEEE 802.11 p and IEEE 802.11 n based on QoS for UAV net-

works. In *Proceedings of the 9th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, pages 65–71, Miami Beach FL USA, 2019.

[46] Truong Duy Dinh, Duc Tran Le, Thi Thu Thao Tran, and Ruslan Kirichek. Flying ad-hoc network for emergency based on IEEE 802.11 p multichannel MAC protocol. In *International Conference on Distributed Computer and Communication Networks*, pages 479–494, Moscow, Russia, 2019. Springer.

[47] Hassen Redwan Hussen, Sung-Chan Choi, Jong-Hong Park, and Jaeho Kim. Performance analysis of manet routing protocols for UAV communications. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 70–72, Czech Republic, 2018. IEEE.

[48] Antonio Guillen-Perez, Ramon Sanchez-Iborra, Maria-Dolores Cano, Juan Carlos Sanchez-Aarnoutse, and Joan Garcia-Haro. Wifi networks on drones. In *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, pages 1–8, Bangkok, Thailand, 2016. IEEE.

[49] Evşen Yanmaz, Markus Quaritsch, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. Communication and coordination for drone networks. In *Ad hoc networks*, pages 79–91. Springer, 2017.

[50] Samira Hayat, Evşen Yanmaz, and Christian Bettstetter. Experimental analysis of multipoint-to-point uav communications with ieee 802.11 n and 802.11 ac. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1991–1996, Hong Kong, China, 2015. IEEE.

[51] Guido R Hiertz, Dee Denteneer, Lothar Stibor, Yunpeng Zang, Xavier Pérez Costa, and Bernhard Walke. The IEEE 802.11 universe. *IEEE Communications Magazine*, 48(1):62–70, 2010.

[52] Ozan Tonguz, Nawapom Wisitpongphan, Fan Bai, Priyantha Mudalige, and Varsha Sadekar. Broadcasting in VANET. In *2007 mobile networking for vehicular environments*, pages 7–12, Anchorage, AK, USA, 2007. IEEE.

[53] Syed Hassan Ahmed, Safdar Hussain Bouk, and Dongkyun Kim. Adaptive beaconing schemes in vanets: Hybrid approach. In *2015 International Conference on Information Networking (ICOIN)*, pages 340–345, Cambodia, 2015.

[54] Marc Heissenbüttel, Torsten Braun, Thomas Bernoulli, and Markus Wälchli. Blr: beacon-less routing algorithm for mobile ad hoc networks. *Computer communications*, 27(11):1076–1086, 2004.

[55] David D Coleman. *CWAP Certified Wireless Analysis Professional Official Study Guide: Exam PW0-270*. John Wiley & Sons, Alameda, CA, United States, 2011.

[56] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, 2016.

[57] Matthew Gast. *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.", Sebastopol, California, 2005.

[58] Xue Yang and Nitin Vaidya. On physical carrier sensing in wireless ad hoc networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, pages 2525–2535, Miami, FL, USA, 2005. IEEE.

[59] Mineo Takai, Jay Martin, Rajive Bagrodia, and Aifeng Ren. Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad*

hoc networking & computing, pages 183–193, New York, United States, 2002.

[60] R Michael Buehrer. Code division multiple access (CDMA). *Synthesis Lectures on Communications*, 1(1):1–192, 2006.

[61] Curt A. Levis. *Encyclopedia of RF and Microwave Engineering*, chapter Friis Free-Space Transmission Formula. John Wiley & Sons, Ltd, Hoboken, New Jersey, U.S., 2005.

[62] Emmanuelle Lebhar and Zvi Lotker. Unit disk graph and physical interference model: Putting pieces together. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–8, Chengdu, China, 2009.

[63] Warren L Stutzman and Gary A Thiele. *Antenna theory and design*. John Wiley & Sons, Hoboken, New Jersey, U.S., 2012.

[64] Rafhael Amorim, Huan Nguyen, Preben Mogensen, István Z Kovács, Jeroen Wigard, and Troels B Sørensen. Radio channel modeling for UAV communication over cellular networks. *IEEE Wireless Communications Letters*, 6(4):514–517, 2017.

[65] Edward A Lee and David G Messerschmitt. *Digital communication*. Springer Science & Business Media, Boston, MA, 2012.

[66] Marco Chiani and Davide Dardari. Improved exponential bounds and approximation for the q-function with application to average error probability computation. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 2, pages 1399–1402, Taipei, Taiwan, 2002. IEEE.

[67] Nasrin Taherkhani and Samuel Pierre. Improving dynamic and distributed congestion control in vehicular ad hoc networks. *Ad Hoc Networks*, 33:112–125, 2015.

[68] Jelena Mišić, Ghada Badawy, Saeed Rashwand, and Vojislav B Mišić. Tradeoff issues for CCH/SCH duty cycle for IEEE 802.11 p single channel devices. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–6, Miami, FL, USA, 2010. IEEE.

[69] Qi Chen, Daniel Jiang, and Luca Delgrossi. IEEE 1609.4 DSRC multichannel operations and its implications on vehicle safety communications. In *2009 IEEE vehicular networking conference (VNC)*, pages 1–8, Tokyo, Japan, 2009. IEEE.

[70] Xia Shen, Xiang Cheng, Rongqing Zhang, Bingli Jiao, and Yang Yang. Distributed congestion control approaches for the IEEE 802.11 p vehicular networks. *IEEE intelligent transportation systems magazine*, 5(4):50–61, 2013.

[71] Andy Triwinarko, Iyad Dayoub, Marie Zwingelstein-Colin, Mohamed Gharbi, and Basma Bouraoui. A PHY/MAC cross-layer design with transmit antenna selection and power adaptation for receiver blocking problem in dense VANETs. *Vehicular Communications*, 24:100233, 2020.

[72] Jiho Jang and Kwang Bok Lee. Transmit power adaptation for multiuser OFDM systems. *IEEE Journal on Selected Areas in Communications*, 21(2):171–178, 2003.

[73] Guoru Ding, Qihui Wu, Yulong Zou, Jinlong Wang, and Zhan Gao. Joint spectrum sensing and transmit power adaptation in interference-aware cognitive radio networks. *Transactions on Emerging Telecommunications Technologies*, 25(2):231–238, 2014.

[74] Esteban Egea-Lopez and Pablo Pavon-Marino. Fair congestion control in vehicular networks with beaconing rate adaptation at multiple transmit powers. *IEEE Transactions on Vehicular Technology*, 65(6):3888–3903, 2016.

[75] Danda B. Rawat, Dimitrie C. Popescu, Gongjun Yan, and Stephan Olariu. Enhancing vanet performance by joint adaptation of transmis-

sion power and contention window size. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1528–1535, 2011.

[76] Wan Norsyafizan, W. Muhamad, Jamil Khan, and Jason Brown. Energy efficient contention window adaptation algorithm for multi-class traffic. In *2015 IEEE International RF and Microwave Conference (RFM)*, pages 73–78, Kuching, Malaysia, 2015.

[77] Yuxia Lin and Vincent W. S. Wong. Cross-layer design of mimo-enabled wlans with network utility maximization. *IEEE Transactions on Vehicular Technology*, 58(5):2443–2456, 2009.

[78] Julius Goldhirsh and Wolfhard J Vogel. Handbook of propagation effects for vehicular and personal mobile satellite systems. *NASA Reference Publication*, 1274:40–67, 1998.

[79] Fabian Kuhn, Thomas Moscibroda, and Rogert Wattenhofer. Unit disk graph approximation. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pages 17–23, Philadelphia, PA, USA, 2004.