



# Drone meshnetwerk simulatie

Software Design Document

HAN Arnhem

561399

MWJ.Berentsen@student.han.nl

Versie 1

Alten Nederland B.V.

Docent: J. Visch, MSc

Assessor: ir. C.G.R. van Uffelen

M.W.J. Berentsen

2 mei 2019

# Inhoudsopgave

<b>1</b>	<b>Inleiding</b>	<b>2</b>
1.1	Begrippenlijst . . . . .	2
<b>2</b>	<b>Architectural Overview</b>	<b>3</b>
2.1	Componenten . . . . .	5
2.2	Interfaces . . . . .	9
2.3	Ros topics en services . . . . .	10
<b>3</b>	<b>Detailed Design Description</b>	<b>12</b>
3.1	Deployment Diagram . . . . .	12
3.2	Design Sub-Systeem Communicatie . . . . .	12
3.3	Design Sub-Systeem gazebo . . . . .	19
3.4	Design Sub-System ros . . . . .	20
3.5	Design Sub-System NRF24 . . . . .	21
	<b>Literatuur</b>	<b>22</b>
<b>A</b>	<b>Appendix 1</b>	<b>23</b>

# 1 — Inleiding

Het volgende verslag betreft de Software Requirements Specification voor de afstudeerstage van Maurice Berentsen (hierna: student). Dit document volgt het document: *"Software Design Description Template"* (Van Heesch, 2016)

Het beschrijft de hoe de uiteindelijke applicatie eruit zal zien en wat de functionaliteit hiervan zal zijn. Op de manier is het voor de student maar ook betrokken partijen duidelijk wat er gerealiseerd zal worden. Het verduidelijkt de werking van de sub-componenten en de onderlinge relaties.

## 1.0.1 leeswijzer

Eerst zal een korte beschrijving gegeven worden van het doel van dit document. Daarna wordt er door middel van een tabel een begrippenlijst toegelicht. In het tweede hoofdstuk is als eerste een component diagram te vinden waarin de functionaliteit van de verschillende componenten te vinden is en de manier waarop deze componenten met elkaar communiceren. Vervolgens is de algemene flow van het programma te zien, hierin wordt duidelijk welke stappen er onder water worden genomen als er een bepaalde actie wordt uitgevoerd. In hoofdstuk drie worden alle subsystemen duidelijk uitgewerkt en worden de ontwerpen van deze systemen uitgewerkt.

## 1.1 Begrippenlijst

Term	Omschrijving
term	Omschrijving

Tabel 1.1: Begrippenlijst

## 2 — Architectural Overview

In het component diagram [Figuur 2.1](#) op de volgende pagina is te zien dat bepaalde componenten voorzien zijn van een andere kleur. De groene kleur betekend dat het component voorzien is door gazebo. De rode en blauw kleur zijn gegeneerd door ROS waarbij de rode een rosservice zijn en de blauwe een rostopic. Op het hoogste niveau is het diagram te verdelen in vier groepen.

**Communication** is het component die de beslissingen neemt voor de communicatie. Het heeft de intelligentie om het netwerk in kaart te brengen en te routeren. Daarnaast is het in staat om een beslissing te nemen om een verzoek te sturen om de drone te verplaatsen. Tenslotte bevat het meerdere berichten die gebruikt om het netwerk te kunnen onderhouden en op te bouwen.

**Drone** is op dit moment een leeg component waar alleen een high level interface aanwezig is die de mogelijkheid biedt om de huidige locatie van een drone terug te geven of een nieuwe locatie als doel te geven aan de drone.

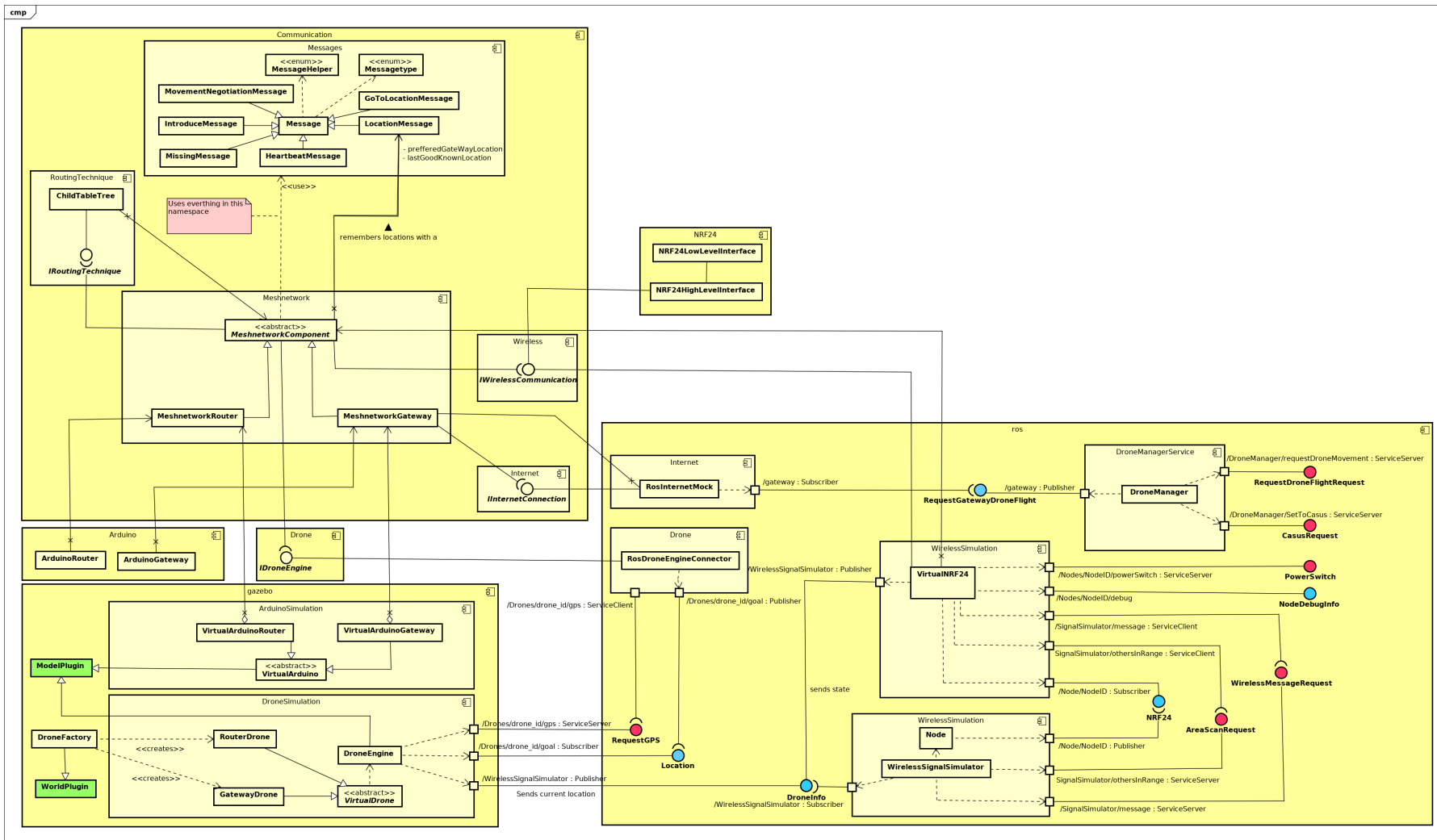
**Gazebo** is de plek waar de virtualisatie van de drone plaats vindt. Er kunnen virtuele router of gateway drones gemaakt worden die voorzien zijn van een virtuele arduino ingeladen met de juiste software en een virtuele drone motor. Deze worden als model plug-in gekoppeld aan de drone in een gesimuleerde wereld waar Gazebo vervolgens physics op de drones kan uitvoeren.

**ROS** is het component waar de simulatie van de communicatie plaats vindt. Er is een virtuele NRF24 om te communiceren met andere nodes in het netwerk. De aanwezige WirelessSignalSimulator zorgt dat dit realistisch gebeurt door alleen communicatie uit te voeren als dit volgens de voorwaarden mag. Dit bepaald de simulator op basis van informatie die het krijgt van Gazebo en de NRF. Tenslotte is er een DroneManager aanwezig die een interface aanbiedt via ros voor de ontwikkelaar om verbinding te maken met de gateways uit het netwerk.

**Arduino** is het component die zich onderscheid in twee rollen, een Arduino kan de rol van gateway hebben of de rol van router in de het meshnetwerk.

**NRF24** is het gebruikte component om draadloze communicatie mogelijk te maken. Het is voorzien van een driver in de vorm van een low level interface en bied zich aan via een high level interface voor communicatie.

Na het component diagram zal elke subcomponent kort toegelicht worden



Figuur 2.1: Globale component diagram

## 2.1 Componenten

### 2.1.1 Internet

Dit component bevat alleen een high level interface om verbinding te maken en te verbreken met het internet

### 2.1.2 Meshnetwork

Dit component bevat de basis componenten voor het opbouwen van een meshnetwork.

### 2.1.3 MeshnetworkComponent

Een MeshnetworkComponent is de basis van elk component in het mesh netwerk. Het vereist routingstechniek via de [IRoutingTechnique](#), een aansluiting naar een drone via [IDroneEngine](#) en een vorm van draadloze communicatie via [IWirelessCommunication](#). Hij maakt gebruik van het component [Messages](#) om te communiceren met andere MeshnetworkComponenten.

### 2.1.4 MeshnetworkRouter

Een meshnetwork router is een [MeshnetworkComponent](#) die in staat is verbinding met andere nodes op te bouwen. Zijn doel is om altijd verbinding te hebben met een [MeshnetworkGateway](#). Als hij dit te lang niet heeft kan hij zich verplaatsen door aanspraak te maken op de DroneEngine. Wanneer hij zich in een groep van andere Routers bevindt zal hij eerst onderling onderhandelen wie er zich moet verplaatsen.

### 2.1.5 MeshnetworkGateway

De gateway is een [MeshnetworkComponent](#) die in staat is een verbinding op te bouwen naar een punt buiten het meshnetwork. In de huidige situatie kan dit via een internetverbinding die loopt via de interface [IInternetConnection](#).

### 2.1.6 Messages

Dit component betreft een verzameling van bericht samenstellingen die gebruikt worden voor de communicatie van het meshnetwork.

### 2.1.7 Message

Een Message is de basis van elke bericht en bevat tenminste de volgende informatie: Maker, zender, berichttype, ontvanger, geadresseerde.

### 2.1.8 GoToLocationMessage

Dit bericht bevat een locatie waar een drone zich naartoe moet verplaatsen.

### 2.1.9 HeartbeatMessage

Dit bericht wordt gebruikt om de verbinding met anderen te onderhouden. Dit bericht maakt een hop per keer dat deze doorgestuurd wordt.

### 2.1.10 IntroduceMessage

Een introductie bericht wordt gebruikt door een node om schik voor te stellen aan alle andere nodes die dichtbij zijn.

### 2.1.11 LocationMessage

Dit bericht wordt gebruikt om de huidige locatie van een node door te sturen naar een ander.

### 2.1.12 MissingMessage

Zodra een [MeshnetworkComponent](#) de verbinding verliest met een ander gebruikt hij dit bericht om andere daarover te informeren.

### 2.1.13 MovementNegotiationMessage

Om onderling te onderhandelen tussen de nodes wie er actie moet ondernemen wordt dit bericht gebruikt.

### 2.1.14 RoutingTechnique

Dit component voorziet het meshnetwerk van een routing techniek wat dus inhoudt dat dit het component is die de communicatieroutes opbouwt naar andere punten in het netwerk.

### 2.1.15 ChildTableTree

Deze techniek is een hybride meshnetwerk routing techniek waarbij een node de burens die deze heeft ziet als een kind. Als een node een nieuw kind heeft vertelt hij dit aan zijn burens. Omdat een node door zijn burens ook gezien wordt als kind registreren deze zijn nieuwe kind dus als kleinkind. Hierdoor hoeft een node alleen maar te zoeken aan wie welk kind hij een bericht hoeft door te geven. De complexiteit van de routingtechniek neemt per stap in het netwerk af bij nodes met meerdere kinderen. Als een drone zich verplaatst heeft wist deze het de opgebouwde geheugen van kinderen en kleinkinderen.

### 2.1.16 Wireless

Dit component voorziet een [MeshnetworkComponent](#) van een high level interface voor draadloze communicatie. Als er een draadloos component wordt aangesloten moet deze hierop aangesloten worden.

### 2.1.17 Drone

Dit component representeert een drone. Omdat er op dit moment geen fysieke drone aanwezig is bezit deze component alleen een high level interface. Deze interface is geschikt voor het ophalen van de huidige locatie en het ontvangen van een doellocatie

### 2.1.18 ArduinoRouter

Dit component is een Arduino die voorzien is van een sketch om zich te gedragen als een router in het meshnetwerk

### 2.1.19 ArduinoGateway

Dit component is een Arduino die voorzien is van een sketch om zich te gedragen als een gateway in het meshnetwerk

### 2.1.20 ArduinoSimulation

In de dit component wordt een arduino gesimuleerd, deze simulatie is puur functioneel en betreft alleen een gelijk gedrag met betrekking tot de Arduino sketch. Het heeft geen effect op de clock snelheid deze is gelijk aan de computer waarop de software draait.

### 2.1.21 VirtualArduino

Dit abstracte component verplicht elke virtuele Arduino tot het aanmaken van een setup en loop functie zoals dit ook is in Arduino sketches. Omdat deze Arduino altijd wordt toegevoegd aan een ander model erft dit component over van modelplugin.

### 2.1.22 VirtualArduinoRouter

Deze [VirtualArduino](#) is voorzien van een sketch die de arduino zich laat gedragen als in router in het meshnetwerk. Dit component representeert het component [ArduinoRouter](#) in gazebo.

### 2.1.23 VirtualArduinoGateway

Deze [VirtualArduino](#) is voorzien van een sketch die de arduino zich laat gedragen als in gateway in het meshnetwerk. Dit component representeert het component [ArduinoGateway](#) in gazebo.

### 2.1.24 DroneSimulation

Dit component is verantwoordelijk voor het simuleren van drones.



### 2.1.25 DroneEngine

Elke [VirtualDrone](#) is voorzien deze drone engine. Dit component maakt het mogelijk voor een drone om zich in een rechte lijn door lucht zich te verplaatsen waarbij het opstijgen en de landing verticaal wordt uitgevoerd. Daarnaast biedt dit component via een rosservice het deel van de interface [IDroneEngine](#) aan om de huidige locatie op te vragen. Via een rostopic kan er een doel gestuurd gestuurd worden om het andere deel van de net genoemde interface te voorzien. Tenslotte stuurt dit component bij elke verplaatsing de huidige locatie door naar de [WirelessSignalSimulator](#).

### 2.1.26 VirtualDrone

Een virtuele drone is een abstract component die alle variabelen bevat die nodig zijn om een drone in de wereld te injecteren. Deze variabelen betreffen een drone id, locatie en verwijzing naar de gazebo wereld.

### 2.1.27 GatewayDrone

Deze drone is een [VirtualDrone](#) die in de sdf omschrijving wordt voorzien van een [VirtualArduinGateway](#) plugin.

### 2.1.28 RouterDrone

Deze drone is een [VirtualDrone](#) die in de sdf omschrijving wordt voorzien van een [VirtualArduinRouter](#) plugin.

### 2.1.29 DroneFactory

Dit component is verantwoordelijk voor het produceren van drones. Het is een WorldPlugin die aan de hand van meegegeven parameters gateway en router drones aanmaakt.

### 2.1.30 RosDroneEngineConnector

Dit component is een adapter tussen de [DroneEngine](#) en de interface [IDroneEngine](#). Die het mogelijk maakt om ros te gebruiken om aanspraak te maken op de virtuele drone.

### 2.1.31 DroneManagerService

Dit component is de toegangspoort voor de ontwikkelaar tot de [MeshnetworkGateway](#). Op dit moment is het component alleen geschikt om verplaatsingverzoeken te versturen via de gateway naar de Drones. Hiervoor biedt het twee interfaces in de vorm van rosservices aan waarbij er een voor locaties is en de ander voor casussen.

### 2.1.32 DroneManager

Dit subcomponent realiseert de rosservices en publiceert verplaatsingverzoeken naar de gateways.

### 2.1.33 RosInternetMock

Deze internetmock laat de [DroneManager](#) zich voordoen als internetpunt zodat er geen daadwerkelijke TCP/IP implementatie hoeft worden gemaakt

### 2.1.34 WirelessSimulation

Het WirelessSimulation component is verantwoordelijk voor het simuleren van de NRF24 en het draadloze signaal hiervan.

### 2.1.35 WirelessSignalSimulator

Deze simulator bepaald of twee nodes met elkaar mogen communiceren. Het doet dit op basis van informatie die het continue ontvangt via het drone informatie topic.

### 2.1.36 Node

Dit component wordt alleen gebruikt door de [WirelessSignalSimulator](#) om te registreren welke Nodes bestaan, waar ze zijn en naar welk topic ze luisteren.

### 2.1.37 VirtualNRF24

Dit component is de virtuele versie van de NRF24, het is in staat om payloads te verwerken van 32 byte. Deze kan de NRF24 naar een direct adres versturen of zenden naar alle NRF24 nodes binnen bereik.

## 2.2 Interfaces

### 2.2.1 IInternetConnection

Deze high level interface wordt gebruikt voor het leggen van een verbinding naar een extern punt buiten het netwerk. Omdat de gateway zich alleen hoeft te verbinden met een punt bestaat deze interface alleen uit een connect en een disconnect.

### 2.2.2 IWirelessCommunication

In deze interface wordt de functionaliteit gesteld waar een draadloos communicatiemiddel aan moet voldoen. Het bevat functies om een antenne te starten en te stoppen. Het versturen van een bericht naar een specifiek punt of het zenden naar elke punt in de buurt. Er is een functie aanwezig om wanneer mogelijk debugging te gebruiken. Tenslotte moet een aansluitend component met deze interface moeten kunnen teruggeven of deze aan of uit staat.

### 2.2.3 NRF24HighLevelInterface

Deze high level interface is de aangeboden interface van de NRF24 en voldoet aan de interface [IWirelessCommunication](#). Het praat direct met de driver van de NRF24.

### 2.2.4 NRF24LowLevelInterface

Deze low level interface is de driver van de NRF24

### 2.2.5 IDroneEngine

Een drone engine interface representeert de aansluiting met een drone. Hierin moet het mogelijk zijn om een doel coördinaat te sturen naar de drone om zich naartoe te verplaatsen. Daarnaast moet het mogelijk zijn voor een drone om de huidige positie terug te geven.

### 2.2.6 IRoutingTechnique

Deze interface bevat functies voor het uitvoeren van een routingstechniek. Het heeft functies voor het starten en onderhouden van het netwerk. Daarnaast zijn er functies hoe gereageerd moet worden op het vinden en verliezen van andere netwerkpunten. Er zijn functies aanwezig om een adres op te halen waar naartoe gezonden moet worden om een punt te bereiken. Ook kan er opgehaald worden welke punten zijn aangesloten, hoeveel punten dit zijn en hoeveel directe aansluitingen er zijn. Tenslotte is er functie aanwezig die aangeroepen wordt zodra een drone zich verplaatst zodat hier adequaat op gereageerd kan worden.

## 2.3 Ros topics en services

Zoals al eerder benoemd wordt er gebruik gemaakt van de transportlaag van ROS. Dit wordt gedaan in de vorm van Ros topics en services. Hieronder worden de messages en services kort toegelicht.

### 2.3.1 RequestGatewayDroneFlight

Via deze aangeboden service is het mogelijk om een verzoek te sturen naar de gateway voor een verplaatsing. In het verzoek moet het nodeID en locatie zitten.

### 2.3.2 CasusRequest

Deze service maakt het mogelijk om een casus posities door te sturen naar de drones.

### 2.3.3 PowerSwitch

Via deze simpele service kan een [VirtualNRF24](#) aan of uit gezet worden.

### 2.3.4 WirelessMessageRequest

De [VirtualNRF24](#) gebruikt deze service om een NRF24 bericht te versturen naar een andere NRF24. De response geeft aan of het zenden lukte.

### 2.3.5 AreaScanRequest

Deze service kan aangeroepen worden om de id's van alle nodes binnen het bereik van een node terug te krijgen. Dit is nodig om een algemene zending naar iedereen binnen bereik mogelijk te maken.

### 2.3.6 RequestGPS

Deze service wordt aangeboden door de [DroneEngine](#) en biedt de mogelijkheid om de huidige locatie van een drone op te vragen.

### 2.3.7 RequestGatewayDroneFlight

Dit topic wordt gebruikt om locatieverplaatsingverzoeken op te publiceren.

### 2.3.8 NodeDebugInfo

Elke [VirtualNRF24](#) maakt wanneer dit verzocht wordt een Debug topic aan. Hierop wordt informatie gepubliceerd wat van toegevoegde waarde is voor ontwikkelaars.

### 2.3.9 NRF24

Een [VirtualNRF24](#) luistert naar dit topic om zo berichten te kunnen ontvangen.

### 2.3.10 Location

Het location topic representeert de verbinding die een [MeshnetworkComponent](#) zou hebben met een drone op doelen op te versturen waar de Drone heen moet vliegen.

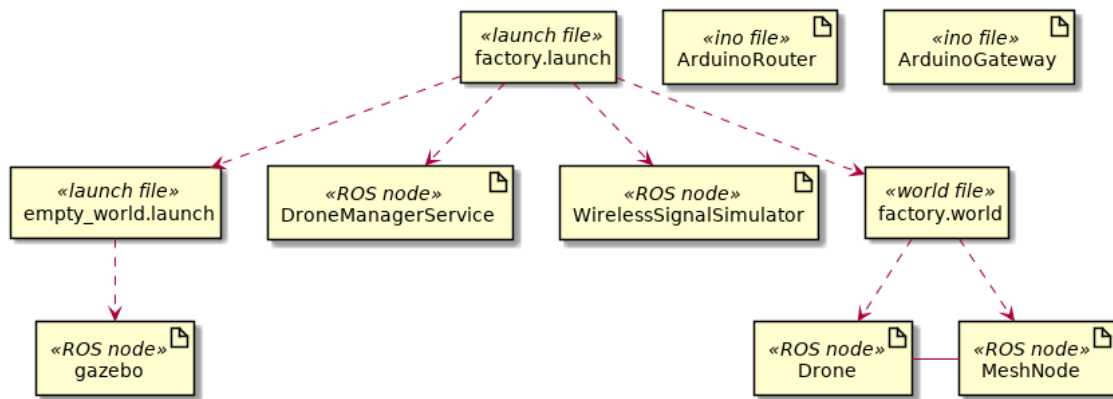
### 2.3.11 DroneInfo

Dit topic is essentieel voor de [WirelessSignalSimulator](#). De informatie die hierop gepubliceerd wordt omvat informatie over de locatie van een [VirtualNRF24](#) of die aan staat en naar welk [NRF24](#) topic deze luistert.

## 3 — Detailed Design Description

### 3.1 Deployment Diagram

In dit deployment diagram staan de executables (ROS nodes) die opgestart worden door de verschillende launch files, daarnaast zijn ook de twee ino files meegenomen die gebruikt worden bij de drones



Figuur 3.1: deployment diagram drone meshnetwerk

#### 3.1.1 Ontwerpkeuzes met betrekking tot deployment

##### Geen aangeboden mogelijkheid om drones los op te starten

Er is voor gekozen om alleen via het factory.launch bestand de drones te kunnen starten. De factory.launch moet gebruikt worden omdat deze factory.world aanroept welke op zijn beurt de plugin start die de drones aanmaakt. In factory.world is het configureerbaar hoeveel gateway en router drones aangemaakt moeten worden. Deze keuze is gemaakt omdat elke drone een uniek id moet hebben in de simulatie omdat elke motor zijn aansluiting publiceert aan de hand van dit id.

Om de gebruiker ervan te ontmoedigen is er daarom ook geen SDF bestand aanwezig waarmee normaal objecten in gazebo geladen worden.

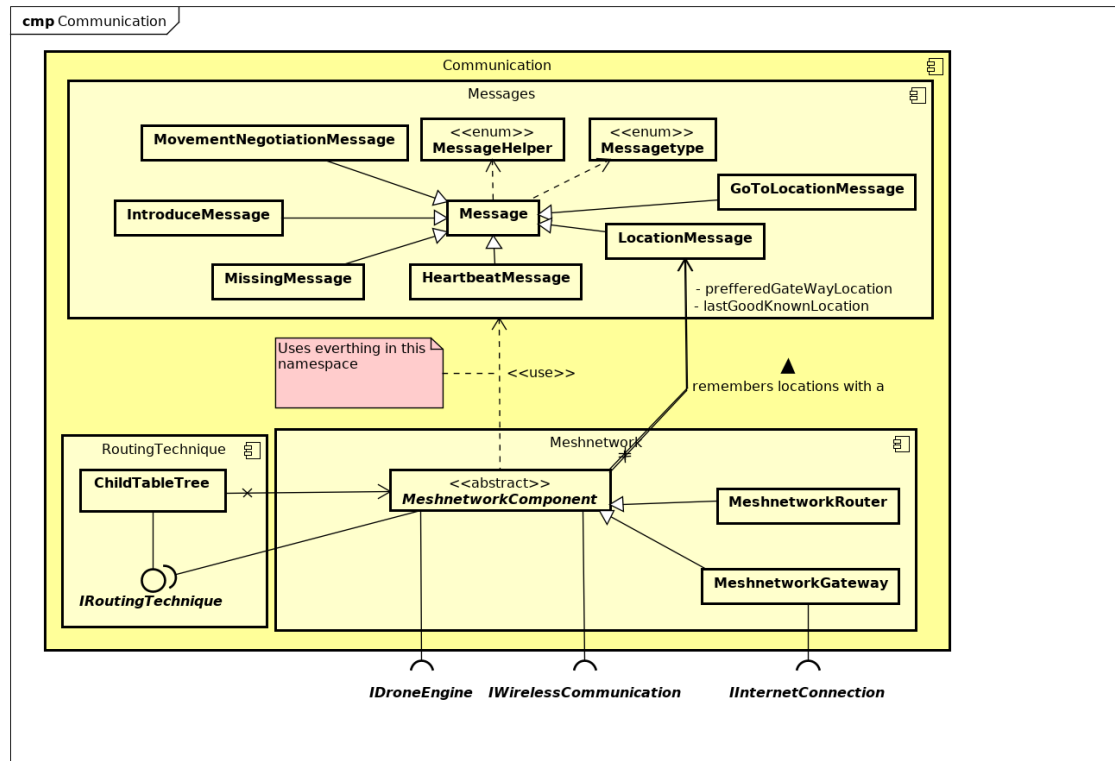
### 3.2 Design Sub-Systeem Communicatie

Het sub-systeem communicatie is verantwoordelijk voor de communicatie door het gebruik van mesh technologie. Het subsysteem kan dit niet alleen en werkt daarom samen met

andere subsystemen, welke zichtbaar zijn in het [Globale component diagram](#). Het component communicatie bestaat ook weer uit kleinere componenten welke weergegeven staan in het [Component diagram communicatie](#)

### 3.2.1 Component Diagram

In de onderstaande afbeelding [3.2](#) staat het component diagram van de communicatie weergegeven. Het is een extractie uit het [Globale component diagram](#)



Figuur 3.2: Component diagram communicatie

#### Interface beschrijving

Het communicatie component heeft geen aangeboden interfaces. Wel wordt er intern een interface aangeboden door het subcomponent RoutingTechnique deze wordt hier behandeld.

**IRoutingTechnique** De volgende functies worden aangeboden door de interface IRoutingTechnique.

```
uint8_t getDirectionToNode( const uint8_t node );

void startRouting( );

void maintainRouting( );

void NodeMovedLocation( );
```

```

uint8_t cantCommunicateWithNode(const uint8_t node);

uint8_t OtherCantCommunicateWithNode(const uint8_t other, const uint8_t node);

void canCommunicateWithNode(const uint8_t node);

void OtherCanCommunicateWithNode(const uint8_t other, const uint8_t node);

const uint16_t getTableSize( );

const uint16_t getAmountOfChildren( );

const std::set< uint8_t > getSetOfChildren( );

const bool empty( );

```

### 3.2.2 getDirectionToNode

Deze functie is de functie waar de aanvragende interface het meeste belang bij heeft. Deze functie verwacht een id van de geadresseerde node als parameter. De routing techniek zal vervolgens uitzoeken naar welke node een bericht doorgegeven moet worden om het bericht aan te laten komen bij de geadresseerde. Dit adres wordt terug gegeven als een return waarde.

**preconditie:** Geadresseerde is bekend bij de routing techniek.

**postconditie:** Routing techniek geeft terug welk adres het vervolg adres is voor het bericht.

### 3.2.3 startRouting

De functie start routing wordt aangeroepen zodra de routing techniek moet beginnen met het opbouwen van communicatie routes.

**preconditie:** Het communicatie waarover de routering gebeurd is beschikbaar

**postconditie:** De node zich aangemeld bij andere nodes en kan adressen opslaan.

### 3.2.4 maintainRouting

Deze functie wordt herhaaldelijk aangeroepen in de software. Het roept de routing techniek aan om een onderhoud te plegen aan de opgebouwde routing informatie. Zo zal vaak het geval zijn dat de routing techniek controleert of de aansluitende nodes nog bestaan.

**preconditie:** De routing techniek is gestart met routeren en heeft een lijst van beschikbare nodes.

**postconditie:** Er is onderhoud uitgevoerd aan de lijst van beschikbare nodes.

### 3.2.5 NodeMovedLocation

Deze functie wordt aangeroepen zodra de drone zich verplaatst heeft. Het is aan routing techniek om hier adequaat op te reageren. In de huidige implementatie worden alle routes als ongeldig gezien waardoor de tabel geleegd wordt.

*preconditie:* Er is geen preconditionie.

*postconditie:* De routingstechniek heeft de tabel bij adequaat bijgewerkt.

### 3.2.6 cantCommunicateWithNode

Deze functie wordt aangeroepen als er geen connectie gelegd kan worden met een node. Deze functie geeft terug op hoeveel routes de node had in het netwerk.

*preconditie:* De node is aanwezig in de route tabel.

*postconditie:* De route wordt verwijderd of als niet beschikbaar beschouwd.

### 3.2.7 OtherCantCommunicateWithNode

Deze functie wordt aangeroepen als een andere node aangeeft dat hij geen connectie kan maken met een node. Deze functie geeft terug op hoeveel routes de niet bereikbare node voorkomt.

*preconditie:* De node is aanwezig in de route tabel.

*postconditie:* De route wordt verwijderd of als niet beschikbaar beschouwd.

### 3.2.8 canCommunicateWithNode

Zodra een node bevestiging heeft dat deze verbinding kan maken wordt deze functie aangeroepen. Als parameter wordt het adres meegegeven.

*preconditie:* Er is bevestiging dat een node een directe verbinding heeft met een andere node.

*postconditie:* De verbonden node wordt toegevoegd als direct route punt.

### 3.2.9 OtherCanCommunicateWithNode

De functie wordt aangeroepen als een andere node doorstuurt dat hij bevestiging heeft dat hij verbinding heeft met een nieuwe node in het netwerk. Als parameter wordt het adres meegegeven van zowel de node die aangeeft dat hij een verbinding heeft gevonden als het adres die hij gevonden heeft.

*preconditie:* Er is bevestiging dat een node een verbinding heeft met een andere node.

*postconditie:* De gevonden node wordt toegevoegd als route punt in het netwerk.

### 3.2.10 getTableSize

Deze functie geeft terug hoeveel routes er zich bevinden in de opgebouwde routing tabel.

*preconditie:* Geen pre conditie.

*postconditie:* Alle routes zijn geteld en de functie geeft dit aantal terug



### 3.2.11 getAmountOfChildren

In deze functie wordt het aantal gevonden direct aansluitende nodes geteld en terug gegeven.

**preconditie:** Geen preconditie

**postconditie:** Het aantal direct aansluitende nodes is geteld en en dit aantal is terug gegeven.

### 3.2.12 getSetOfChildren

Hier wordt een set gemaakt met id's van direct aansluitende nodes **preconditie:** Geen preconditie

**postconditie:** Er is een set met de id's van alle nodes waar een directe verbinding mee gevonden is

### 3.2.13 empty

Deze functie geeft een boolean terug welke aangeeft of de tabel leeg is. Dit is een aparte functie omdat het scheelt in complexiteit ten opzichte van het tellen van een tabel en die vergelijken met 0.

**preconditie:** Geen preconditie

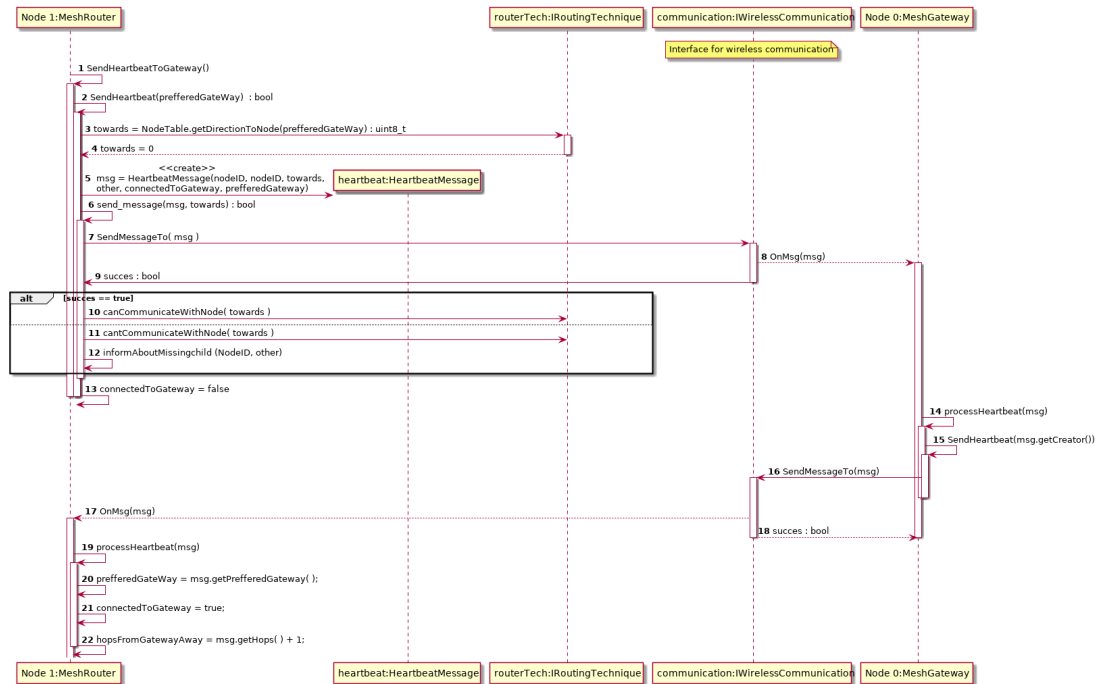
**postconditie:** Er is een check gedaan of de lijst leeg is en dit wordt terug gegeven.

### 3.2.14 Sequence Diagrams

Hier worden aan de hand van sequence diagrammen communicatie wegen omschreven die zich binnen het component afspelen.

#### Heartbeat van router naar gateway met directe verbinding

Om duidelijk te maken hoe een bericht van een node naar een gateway komt wordt het scenario van een heartbeat uitgewerkt in een direct verbinden. Deze weg van communicatie wordt voor alle directe berichten het zelfde uitgevoerd. Dat maakt het overbodig om voor elk type bericht een sequence diagram uit te werken



Figuur 3.3: Sequence diagram van een heartbeat bericht van een router naar een gateway met een directe onderlinge verbinding.

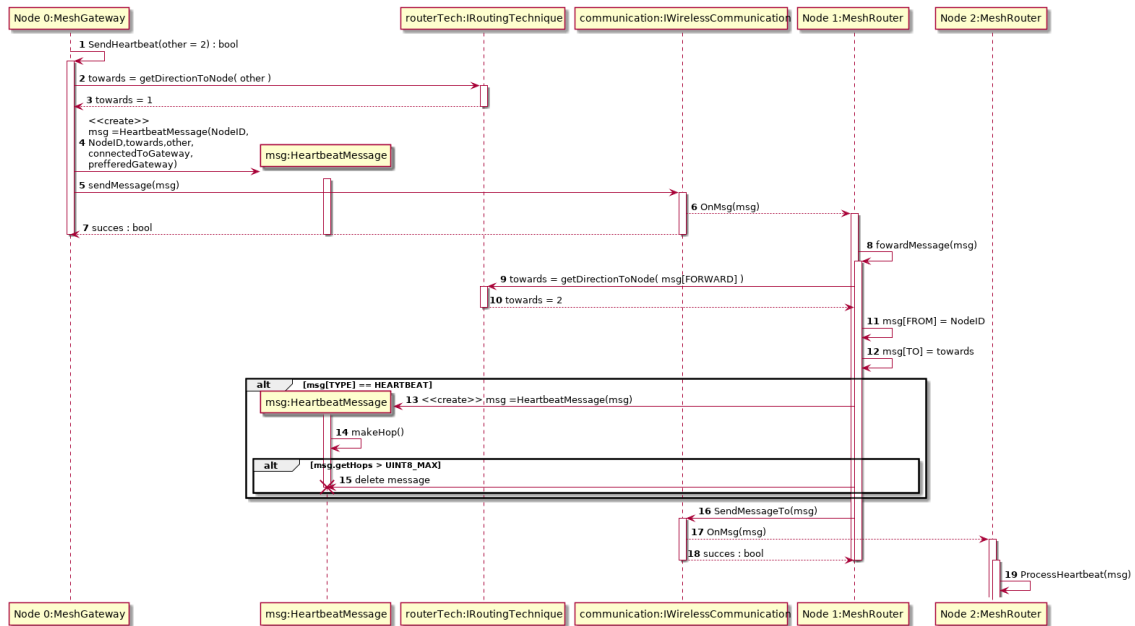
In diagram 3.3 is de flow zichtbaar van een router die een heartbeat bericht verstuurd naar een gateway en daar ook weer response op krijgt. Op het moment dat een router het bericht om een heartbeat te versturen naar de gateway zal hij dit altijd doen naar de gateway waar hij een voorkeursverbinding (preferredGateway) mee heeft. Hij zal aan de aangesloten routeringstechniek vragen of er een route mogelijk is naar de router een aan wie hij zijn bericht dan moet afgegeven. In het geval van deze sequence diagram heeft de router een directe verbinding met de gateway en is in stap 4 te zien dat het teruggegeven adres 0 is het adres van de gateway.

Vervolgens maakt de router een heartbeat bericht aan die het via de [WirelessCommunication](#) interface verstuurd naar de Gateway. De router geeft door aan de routeringstechniek in stap 10 of 11 of het zenden wel of niet gelukt is op basis van de succes feedback uit stap 9. Als het niet lukt geeft hij dit ook door aan zijn aangesloten punten in stap 12 Als laatste voordat de router zijn functie verlaat zet hij in stap 13 de boolean die bijhoudt of verbinding is met de gateway (connectedToGateway) op false omdat hij verwacht dat er reactie komt van de gateway die het weer op true zet.

Bij stap 8 is het gelukt om het bericht te versturen naar de gateway die daar via zijn eigen [WirelessCommunication](#) interface het bericht ontvangt. Er is voor gekozen om deze interface maar één keer weer te geven in de tabel om het overzichtelijk te houden. De reactie van een gateway op een heartbeat is altijd dat er een heartbeat wordt teruggestuurd naar de zender. De functie SendHeartbeat(uint8\_t) in stap 15 werkt gelijk aan die van stap 2 en is daarom niet opnieuw uitgewerkt.

Stap 17 is het moment dat de router weer response heeft van de gateway. Hij stelt op basis van de inhoud van het bericht het ID van de gateway in als preferredGateway, zet de boolean connectedToGateway weer op true en registreert hoeveel hops het bericht nodig had om aan te komen.

## Heartbeat van gateway naar een router via een andere router



Figuur 3.4: Sequence diagram van een heartbeat bericht van een gateway naar een router via een andere router.

Diagram 3.4 laat zien hoe een heartbeat verstuurd wordt vanaf een gateway naar een router via een andere router. Er is gekozen om de flow van een heartbeat bericht te laten zien omdat deze bij een forward ook een hop extra krijgt en daarmee het meest complex is. Elke node heeft een eigen instantie van een routingstechniek en draadloze communicatie, in deze diagram is gekozen om dezelfde te gebruiken om de leesbaarheid goed te houden.

Bij stap 3 is zichtbaar dat de routingstechniek bij aanvraag voor een route naar node 2 een adres teruggegeven wordt van node 1 omdat via deze weg het bericht doorgegeven moet worden. Vanaf stap 8 is zichtbaar wat er gebeurd als een bericht doorgestuurd moet worden. Bij stap 9 wordt er eerst gekeken of er een route is naar de geadresseerde en aan wie het bericht dan doorgegeven moet worden. Node 1 heeft een directe verbinding met node 2 dit wordt dan ook als adres doorgegeven in stap 10. Vervolgens past de meshrouter het bericht aan dat zodat de zender en ontvanger weer juist staan. De waardes waar in staat wie de geadresseerde en maker van het bericht zijn blijven onaangepast. Als het bericht van het type heartbeat is wordt deze apart genomen om er in stap 14 een hop bij op te tellen. Als een heartbeat bericht meer hops heeft gemaakt dan dat er in de waarde van hop past wordt het bericht verwijderd en de flow gestopt.

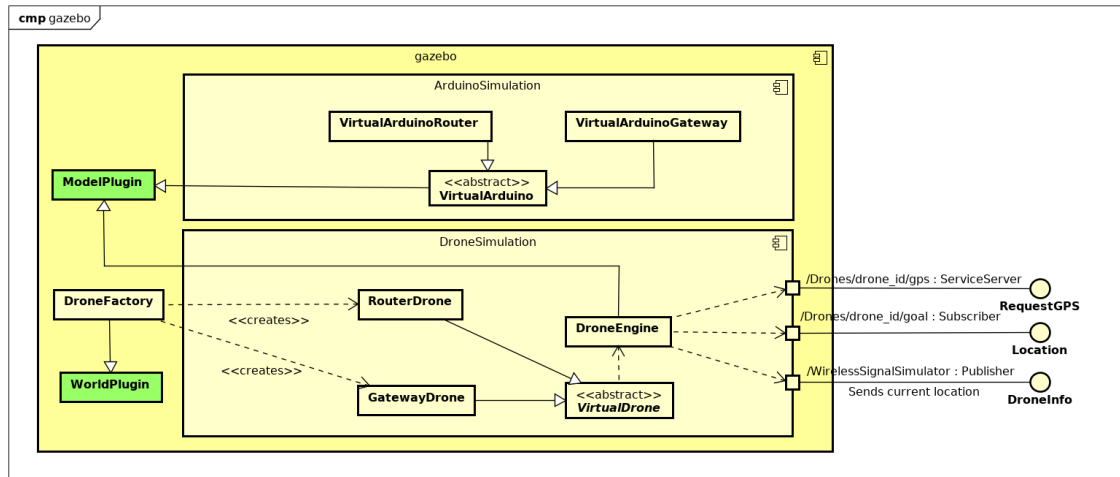
In stap 16 wordt het bericht doorgestuurd naar de geadresseerde. Als het bericht een heartbeat was wordt deze doorgestuurd anders wordt het aangepaste bericht verstuurd.

### 3.2.15 Activity and State Diagrams

### 3.2.16 Design decisions made for the sub-system

## 3.3 Design Sub-System gazebo

### 3.3.1 Component Diagram



Figuur 3.5: component diagram gazebo

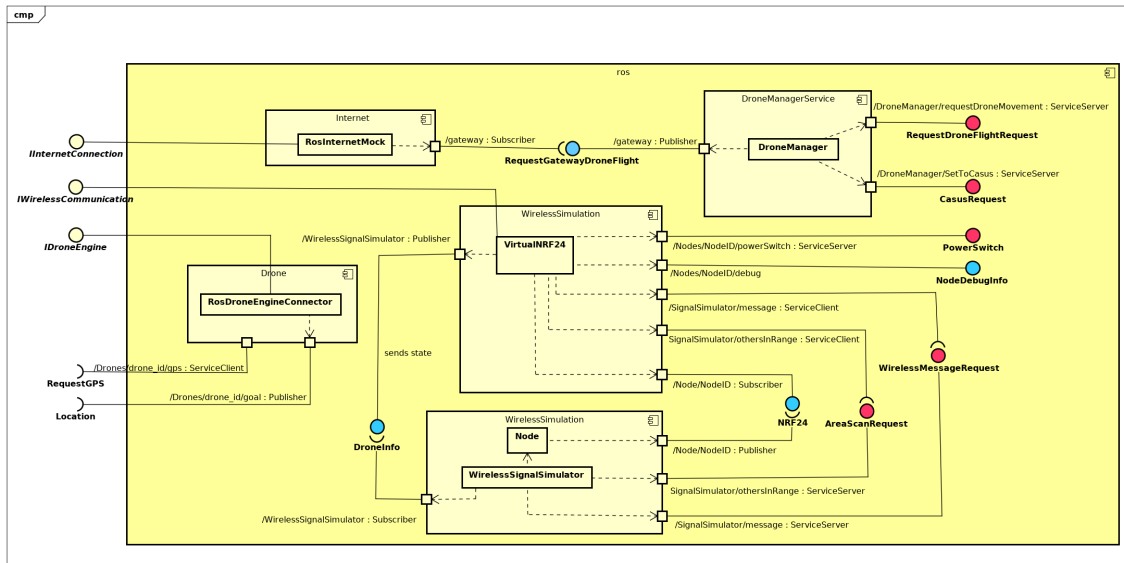
### 3.3.2 Sequence Diagrams

### 3.3.3 Activity and State Diagrams

### 3.3.4 Design decisions made for the sub-system

## 3.4 Design Sub-System ros

### 3.4.1 Component Diagram



Figuur 3.6: component diagram ros

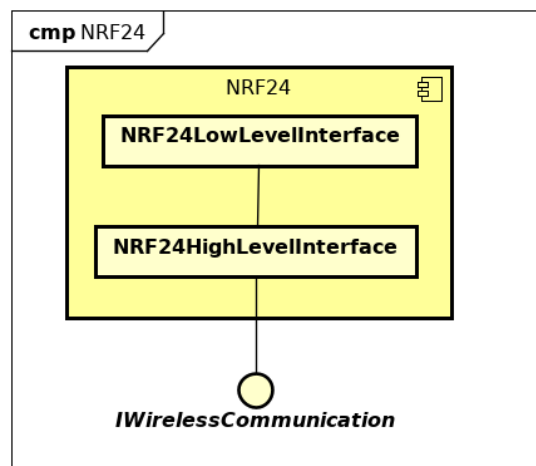
### 3.4.2 Sequence Diagrams

### 3.4.3 Activity and State Diagrams

### 3.4.4 Design decisions made for the sub-system

## 3.5 Design Sub-System NRF24

### 3.5.1 Component Diagram



Figuur 3.7: component diagram ros

### 3.5.2 Sequence Diagrams

### 3.5.3 Activity and State Diagrams

### 3.5.4 Design decisions made for the sub-system

## Literatuur

Van Heesch, U. (2016, 23 september). *Software design description template*.

## A — Appendix 1