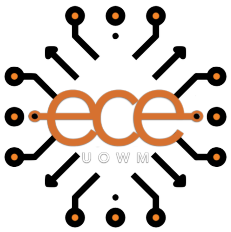


Πανεπιστήμιο Δυτικής Μακεδονίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προσομοίωση Πρωτοκόλλων Δρομολόγησης σε Ιπτάμενα Αυτοοργανωμένα Ασύρματα Δίκτυα

Ανδρέας Μανίτσας

Επιβλέπων: Παναγιώτης Σαρηγιαννίδης
Αναπληρωτής Καθηγητής Π.Δ.Μ.

Κοζάνη, Οκτώβριος 2020

Περίληψη

Η παρούσα εργασία αποτελεί μια εισαγωγή στα συστήματα πολλαπλών ΜΕΑ και στα εργαλεία που χρησιμοποιούνται για την προσομοίωση των αυτοοργανωμένων ασύρματων δικτύων και τις εφαρμογές τους. Στην αρχή αναλύονται οι αρχές των ΜΕΑ, παρουσιάζονται τα χαρακτηριστικά τους και αναφέρονται τα διαφορετικά σενάρια χρήσης. Επιπλέον γίνεται η εισαγωγή στα συστήματα πολλαπλών ΜΕΑ, τα πλεονεκτήματα και τα μειονεκτήματα που έχουν καθώς και η πρώτη αναφορά στα δίκτυα ΙΑΔ. Περνώντας στο πρακτικό μέρος, παρουσιάζεται ο προσομοιωτής δικτύων NS3, αναλύεται ο τρόπος λειτουργίας του και εκτελούνται δύο σενάρια (ένα για ΑΔΟ, ένα για ΙΑΔ) με σκοπό την σύγκριση τριών διαφορετικών πρωτοκόλλων δρομολόγησης. Τέλος, εξάγονται ορισμένα συμπεράσματα για τα διαφορετικά πρωτόκολλα δρομολόγησης και αναλύονται οι προκλήσεις που υπήρξαν.

Λέξεις Κλειδία: Μη Επανδρωμένα Αεροσκάφη, Πρωτόκολλα Δρομολόγησης, Ιπτάμενα Αυτοοργανωμένα Δίκτυα

Abstract

This thesis is an introduction to the multiple UAV systems and the tools used for the simulation of the wireless ad-hoc networks used for the communication of the UAV's. In the beginning, the various aspects of a UAV are analyzed and some use cases are presented. Moreover, there is an introduction to the systems of multiple UAV's, their pros and cons are presented along with a first mention of FANETs. In the practical section of the thesis, the network simulator NS3 is presented, its operating principles are analyzed and two scenarios (one for VANET, one for FANET) are simulated in order to compare three different routing protocols. Finally, some conclusions are drawn for the various routing protocols and challenges faced during the writing of this thesis are analyzed.

Keywords: NS3, UAV, Routing, FANET

Πνευματικά Δικαιώματα

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο

"Προσομοίωση Πρωτοκόλλων Δρομολόγησης σε Ιπτάμενα Αυτοοργανωμένα Ασύρματα Δίκτυα

Simulation of Routing Protocols in Wireless Flying Ad-hoc Networks"

καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Λαμαγιώτη Σαρηγιαννίδη

αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Copyright (C) Ονοματεπώνυμο Φοιτητή & Επιβλέποντα/ες, Έτος, Πόλη

Copyright (C) Ανδρέας Μανίτσας, Λαμαγιώτης Σαρηγιαννίδης, 2020, Κοζάνη

Υπογραφή Φοιτητή:

21/10/2020

Ευχαριστίες

Η παρούσα εργασία είναι αποτέλεσμα μιας επίπονης και εκτενούς έρευνας. Στα πλαίσια ολοκλήρωσης της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Παναγιώτη Σαρηγιαννίδη, Αναπληρωτή Καθηγητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Δυτικής Μακεδονίας για ενθάρρυνση του και στην υποστήριξη του να αναλάβω αυτή την διπλωματική εργασία με βάση τα ενδιαφέροντα μου, καθώς και τον κ. Γιώργο Κακαμούκα, Υποψήφιο Διδάκτορα για την υποστήριξη και την πολύτιμη βοήθεια που προσέφερε κατά την επίβλεψη της διπλωματικής εργασίας.

Τέλος, ευχαριστώ όλους εκείνους που με στήριξαν και έκαναν υπομονή αυτά τα χρόνια για να πραγματοποιήσω τις σπουδές μου, κυρίως την οικογένεια μου, τους φίλους μου και όσα άτομα ήταν όλα αυτά τα χρόνια δίπλα μου. Χρωστάω σε όλους ένα μεγάλο ευχαριστώ.

Αποτέλεσμα Ελέγχου Turnitin

ORIGINALITY REPORT

22%	21%	4%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.nsnam.org Internet Source	7%
2	Submitted to Bournemouth University Student Paper	6%
3	Submitted to Arab Open University Student Paper	1%
4	www.scribd.com Internet Source	1%
5	codegist.net Internet Source	<1%
6	www.mckinsey.com Internet Source	<1%
7	www.ijetch.org Internet Source	<1%
8	en.wikipedia.org Internet Source	<1%
9	Submitted to University of Strathclyde Student Paper	<1%

Προσομοίωση Πρωτοκόλλων Δρομολόγησης σε Ιπτάμενα Αυτοοργανωμένα Ασύρματα Δίκτυα

Ανδρέας Μανίτσας

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή στα Συστήματα ΜΕΑ

1.1 Ορισμός: Τι είναι ένα ΜΕΑ

UAV (Unmanned Aerial Vehicle), στα Ελληνικά ΜΕΑ (Μη Επανδρωμένο Αεροσκάφος)

ΕΛ: Ένα αεροσκάφος το οποίο μπορεί να πλοηγείται και να λειτουργεί χωρίς την παρουσία πιλότου εντός της ατράκτου.

ΕΝ: An aircraft that can navigate and operate without a human pilot onboard.

Εν συντομία ΜΕΑ (ή UAV) ονομάζονται τα ιπτάμενα οχήματα που δεν φέρουν χειριστή εντός της ατράκτου τους, αλλά πραγματοποιούν πτήσεις αυτόνομα ή μέσω τηλεκατεύθυνσης. Τα ΜΕΑ συνήθως έχουν την μορφή μικρού αεροπλάνου η ελικοπτέρου με έναν η περισσότερους κινητήρες και επιφάνειες ελέγχου για ελεγχόμενη πτήση από ειδικό πρόγραμμα ή χειριστήριο εδάφους.

1.2 Υλικό ενός ΜΕΑ

1.2.1 Άτρακτος και Επιφάνειες Ελέγχου

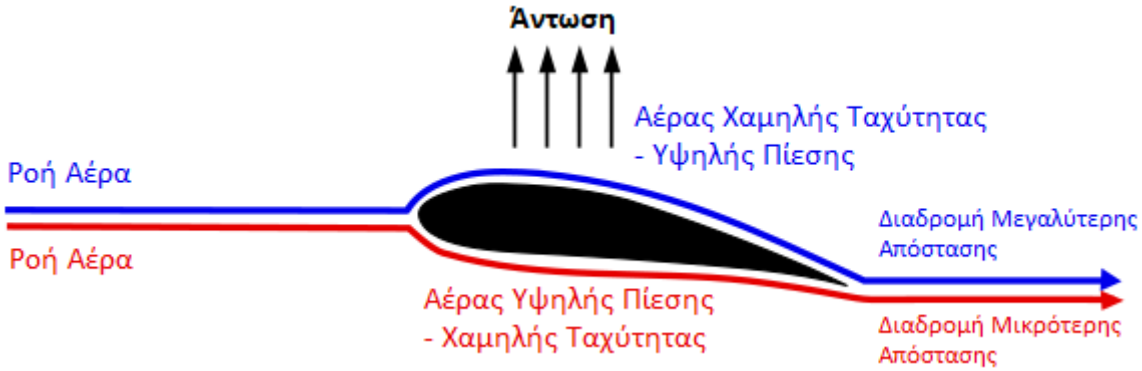
Χωρίζονται σε δύο μεγάλες κατηγορίες ανάλογα με τον τύπο της ατράκτου, που κατά συνέπεια επηρεάζει και τον τρόπο πτήσης. Τα ΜΕΑ Σταθερής Πτέρυγας και τα Πολυκόπτερα.

Σταθερής Πτέρυγας

Τα ΜΕΑ Σταθερής Πτέρυγας είναι αυτά που ομοιάζουν περισσότερο με επιβατικά αεροσκάφη. Διαθέτουν στενόμακρη άτρακτο σωληνοειδούς σχήματος εγκάρσια στα οποία υπάρχουν δύο πτερύγια. Αυτά τα πτερύγια έχουν διπλό ρόλο.

Σχήμα 1.1: ΜΕΑ Σταθερής Πτέρυγας General Atomics MQ-9 Reaper
[1]

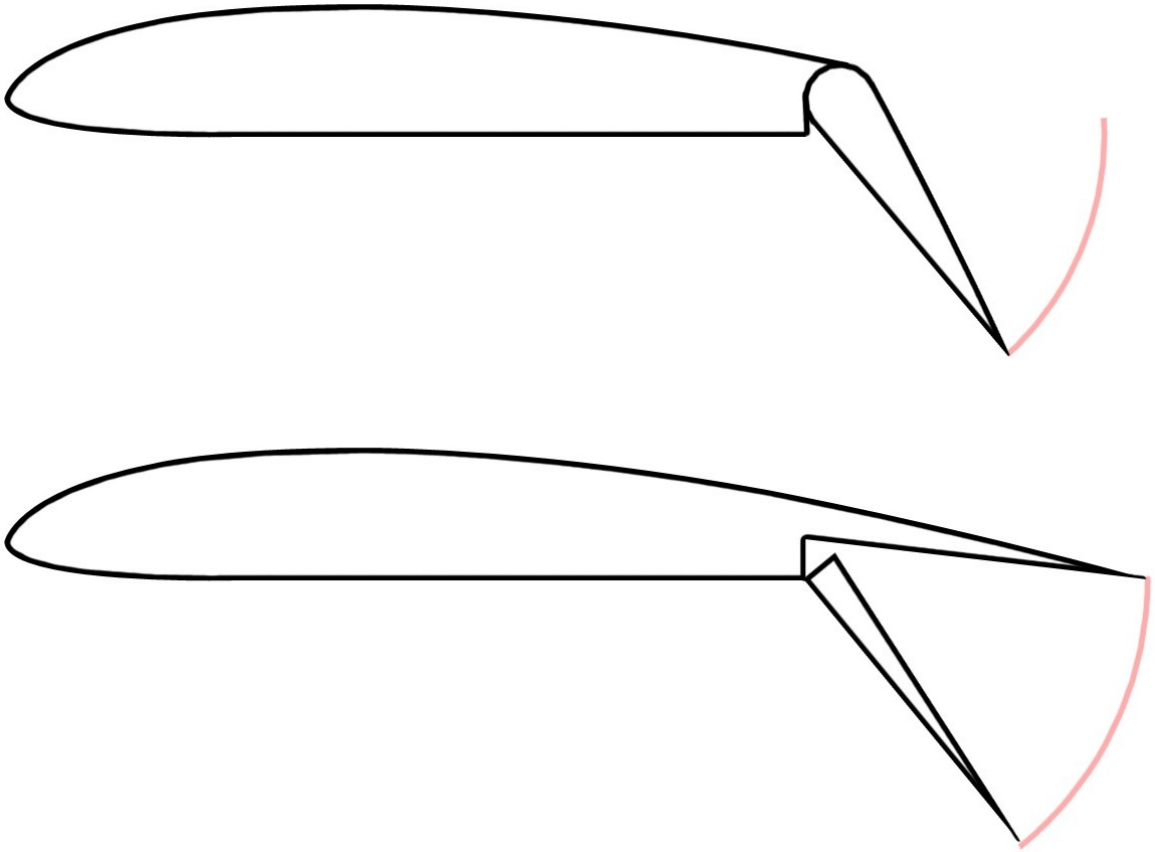
Ο πρώτος ρόλος τους είναι η παραγωγή άντωσης. Αυτό επιτυγχάνεται μέσω του σχήματος της διατομής της πτέρυγας η οποία καθώς ρέει ο αέρας δημιουργεί μια ζώνη χαμηλής πίεσης στο άνω μέρος και μια ζώνη υψηλής πίεσης στο κάτω μέρος. Αυτή η διαφορά πίεσης είναι υπεύθυνη για την άντωση που παράγει το αεροσκάφος και εξαρτάται άμεσα από τα γεωμετρικά χαρακτηριστικά της πτέρυγας.



Σχήμα 1.2: Παραγωγή Άντωσης στη διατομή της Πτέρυγας

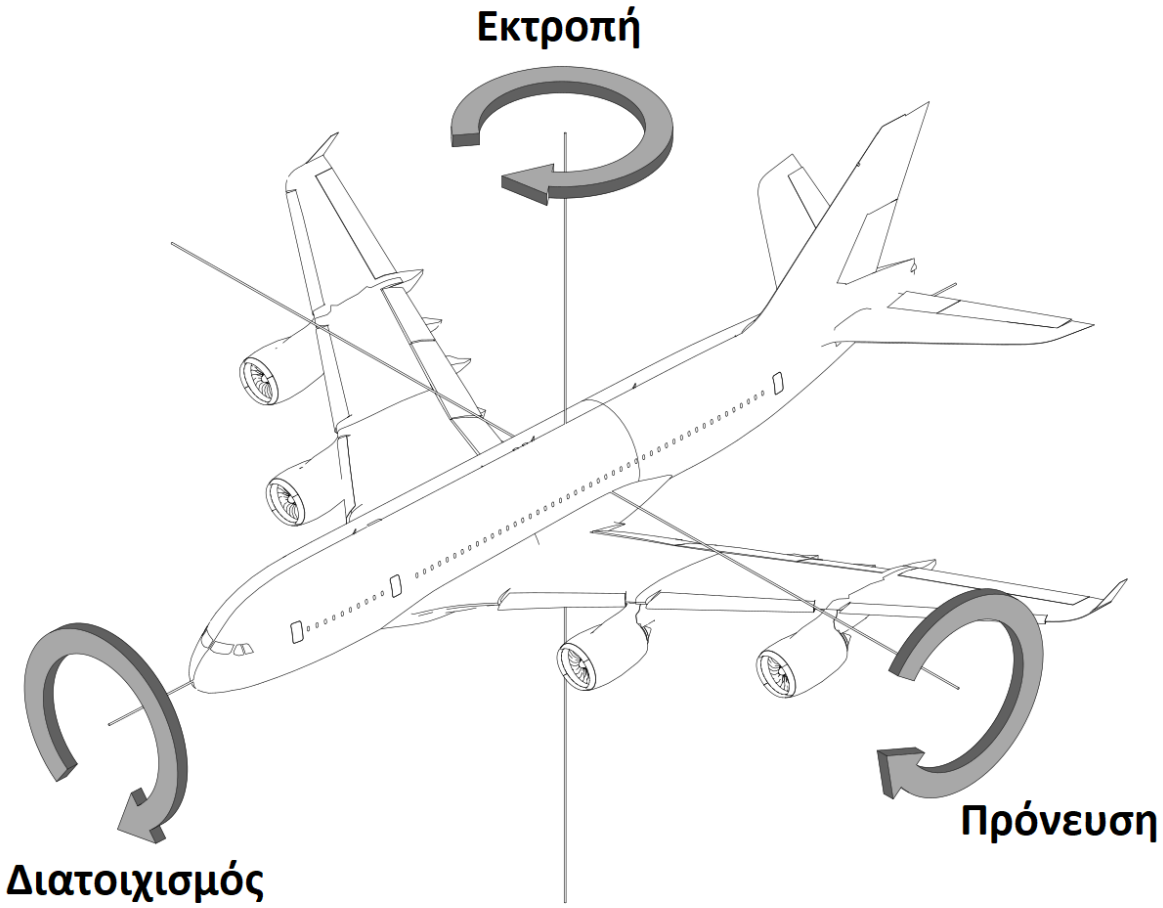
Ο δεύτερος ρόλος τους είναι ο έλεγχος της πτήσης. Κατά μήκος του πίσω άκρου τους υπάρχουν κάποιες κινούμενες επιφάνειες – πηδάλια που έχουν και αυτές διπλό ρόλο. Ο ένας είναι η αύξηση της επιφάνειας του πτερυγίου και συνεπώς την παραγωγή περισσότερης άντωσης σε μικρότερη ταχύτητα, πράγμα χρήσιμο για πιο σύντομες απογει-

ώσεις, προσγειώσεις σε μικρότερη ταχύτητα και την δυνατότητα απογείωσης με μεγαλύτερη φορτίο.



Σχήμα 1.3: Κινούμενο Πηδάλιο στο πίσω άκρο Πτέρυγας (Διατομή)
[2]

Ο δεύτερος είναι η εκτροπή της ροής του αέρα, δημιουργώντας ροπή περιστροφής κατά τους άξονες περιστροφής του αεροσκάφους, επιτρέποντας έτσι τον έλεγχο της κατεύθυνσης. Οι άξονες περιστροφής του αεροσκάφους είναι οι Εκτροπή, Πρόνευση, Διατοιχισμός (Yaw, Pitch, Roll) και εικονίζονται στην παρακάτω εικόνα.



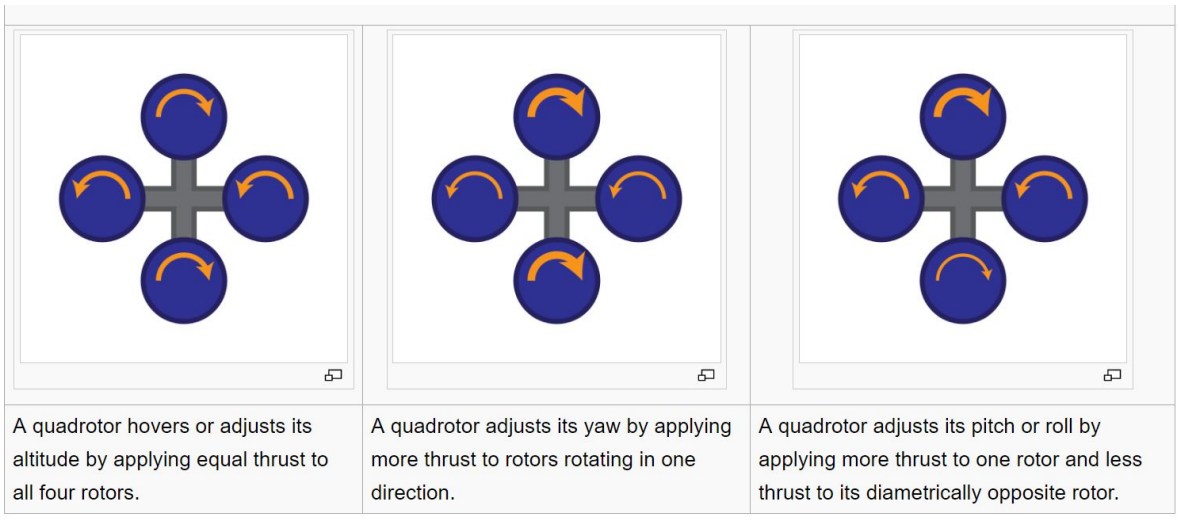
Σχήμα 1.4: Άξονες Περιστροφής Αεροσκάφους
[3]

Πολυκόπτερα

Τα Πολυκόπτερα είναι τα ΜΕΑ που χρησιμοποιούν 3, 4, 6 ή 8 ελικοφόρους κινητήρες, κατά κανόνα ηλεκτρικούς, για την παροχή της κατάλληλης ώσης και ελέγχου της πτήσης χωρίς την χρήση αεροδυναμικών επιφανειών και πτερυγίων. Το πιο συνηθισμένο στυλ είναι αυτό των Τετρακόπτερων, δηλαδή των Πολυκοπτέρων με 4 κινητήρες.

Σχήμα 1.5: Τετρακόπτερο DJI Spark
[4]

Ο έλεγχος της πτήσης γίνεται αποκλειστικά με τον έλεγχο της ώσης που παράγει ο κάθε κινητήρας. Όταν και οι 4 κινητήρες παράγουν την ίδια ώση, το ΜΕΑ ισορροπεί στον αέρα και μένει σταθερό. Για την περιστροφή κατά τον άξονα Εκτροπής, ρυθμίζεται η ώση των κινητήρων σε απέναντι ζευγάρια με την ίδια φορά περιστροφή. Για την περιστροφή κατά τον άξονα Διατοιχισμού ή Πρόνευσης αυξάνεται η ώση κατά την κατεύθυνση κίνησης.



Σχήμα 1.6: Χειρισμός Τετρακόπτερου μέσω μεταβολής ώσης
[5]

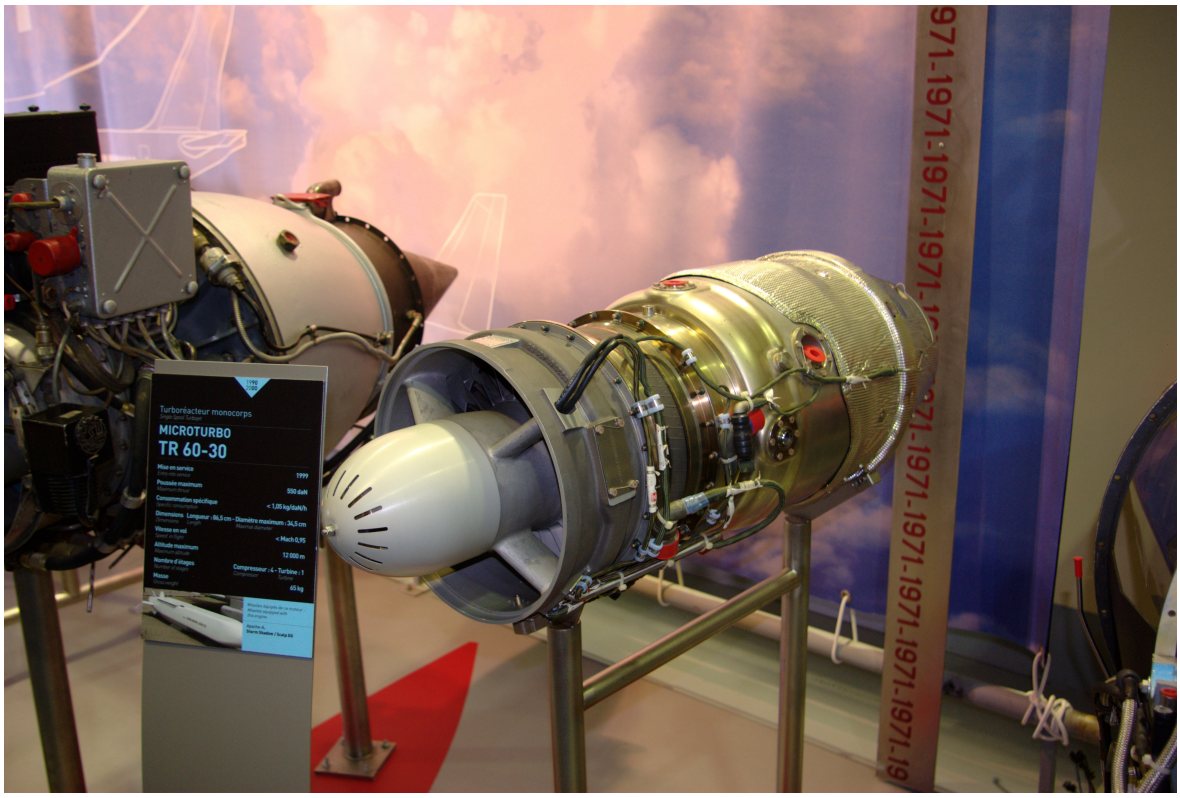
1.2.2 Πηγή Ισχύος και Προώθησης

Σε μικρότερα ΜΕΑ μάζας μέχρι μερικών δεκάδων κιλών, συναντούμε σχεδόν εξ' ολοκλήρου ηλεκτρικά συστήματα πρόωσης. Ηλεκτροκινητήρες κινούν τις προπέλες και η ενέργεια είναι αποθηκευμένη σε κάποια μπαταρία, κατά κανόνα Ιόντων Λιθίου λόγο της εξαιρετικής της ενεργειακής πυκνότητας. Ένα ακόμα πλεονέκτημα των ηλεκτρικών ΜΕΑ είναι η πιο απλή κατασκευή, η μικρότερη απαιτούμενη συντήρηση και η πιο αθόρυβη λειτουργία.

Σχήμα 1.7: Ηλεκτροκινητήρας Xoar Titan T5000 για ΜΕΑ
[6]

Καθώς όμως τα ΜΕΑ γίνονται μεγαλύτερα και βαρύτερα σε μεγαλύτερες κατηγορίες, η επιλογή θερμικών κινητήρων αεροσκαφών αποτελεί μονόδρομο, ιδίως αν οι απαιτήσεις σε διάρκεια πτήσης και υψόμετρο ξεπερνούν τα όρια της ηλεκτρικής πρόωσης. Η μεγαλύτερη ενεργειακή πυκνότητα των υγρών καυσίμων επιτρέπει την πτήση μεγαλύτερων και βαρύτερων ΜΕΑ με πολύ περισσότερο ωφέλιμο φορτίο σε σχέση με τα

ηλεκτροκίνητα που επηρεάζονται σημαντικά από την μάζα των μπαταριών που πρέπει να κουβαλούν.



Σχήμα 1.8: Safran Microturbo TR 60 Turbojet κινητήρας για ΜΕΑ
[7]

Υπάρχουν ηλεκτρικά ΜΕΑ σε πειραματικό στάδιο που μπορούν να πετάξουν σε μεγάλα υψόμετρα για ώρες, αλλά μέχρι να βελτιωθεί η ενεργειακή πυκνότητα των μπαταριών δεν προβλέπεται να υπάρξει κάποια πρακτική εφαρμογή. Στον πίνακα 1.1 παρουσιάζεται η ενεργειακή πυκνότητα της κηροζίνης και των δύο πιο συνηθισμένων τύπων μπαταριών.

Τύπος Καυσίμου	Ενεργειακή Πυκνότητα
Κηροζίνη	46.4 MJ/kg
Μπαταρίες Ιόντων Λιθίου	0.36-0.88 MJ/kg
Μπαταρίες Μολύβδου Οξέως	0.17 MJ/kg

Πίνακας 1.1: Ενεργειακή Πυκνότητα Καυσίμων

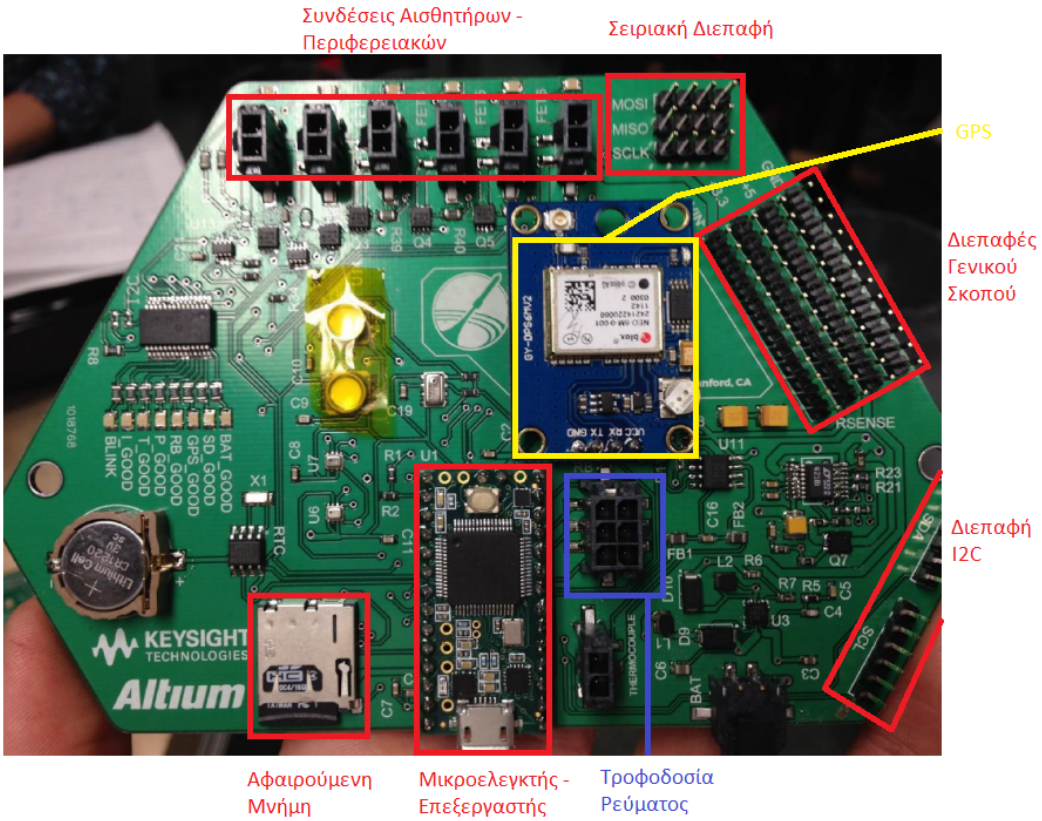
Ακόμα και η καλύτερες μπαταρίες Ιόντων Λιθίου έχουν μόλις το 5% της ενεργειακής πυκνότητας της κηροζίνης. Με τις μέγιστες τιμές για την κηροζίνη και τις μπαταρίες Ιόντων Λιθίου, δηλαδή 46.4 MJ/kg και 0.88 MJ/kg, θεωρώντας ότι ένας σύγχρονος στροβιλοκινητήρας [8, 9] έχει απόδοση 35% και ένας τυπικός ηλεκτροκινητήρας 90%. Ακόμα και έτσι η κηροζίνη αποδίδει 16.24 MJ/kg σε κινητική ενέργεια ενώ οι καλύτερες μπαταρίες Ιόντων Λιθίου 0.79 MJ/kg. Με απλά λόγια για να εκτελέσουμε την ίδια πτήση χρειαζόμαστε περίπου 20 φορές μεγαλύτερη μάζα σε μπαταρίες απ' ότι σε

κηροζίνη για να πάρουμε το ίδιο ποσό κινητικής ενέργειας. Αυτό είναι και το μεγαλύτερο πρόβλημα των ηλεκτρικών ΜΕΑ, ιδίως όσο μεγαλώνει το μέγεθος και η μάζα του αεροσκάφους.

1.2.3 Ηλεκτρονικά Υποσυστήματα και Αισθητήρες

Τα ηλεκτρονικά συστήματα των ΜΕΑ, στο μεγαλύτερο μέρος τους είναι όμοια με τα αντίστοιχα συστήματα που έχει αναπτύξει η βιομηχανία της αεροπλοΐας [10]. Οι όποιες διαφορές τους αφορούν τα συστήματα τηλεπικοινωνιών, μετάδοσης δεδομένων, αυτόνομης πλοήγησης και αναγνώρισης, που στην περίπτωση των ΜΕΑ πρέπει να λειτουργούν αυτόνομα.

Τα ηλεκτρονικά συστήματα των ΜΕΑ θυμίζουν έντονα αυτά των συμβατικών αεροσκαφών, όμως κατά κανόνα είναι ελαφρύτερα, πιο συμπαγή και με λιγότερες δικλείδες ασφαλείας σε περίπτωση σφάλματος. Καθώς δεν εξαρτάται κάποια ανθρώπινη ζωή από την αξιοπιστία των εν λόγω συστημάτων, είναι δυνατόν να γίνουν κάποιες παραλήψεις κατά τον σχεδιασμό με αποτέλεσμα την δραστική μείωση του κόστους.



Σχήμα 1.9: Πλακέτα ηλεκτρονικών από ΜΕΑ με επεξήγηση των επιμέρους εξαρτημάτων

Έτσι όπως είναι αναμενόμενο, η εξέλιξη των ηλεκτρονικών συστημάτων των ΜΕΑ ακολουθεί τις ευρύτερες τάσεις στον χώρο της αεροπλοΐας και πληροφορικής. Η επεξεργαστική ισχύς γίνεται μεγαλύτερη και φθηνότερη με το πέρασμα του χρόνου, και η αυτοματοποίηση αναλαμβάνει όλο και περισσότερους ρόλους στον χειρισμό του αεροσκάφους.

Σε αυτό το σημείο θα πρέπει να σημειωθεί ότι η διαφορά ενός ΜΕΑ με ένα συμβατικό αεροσκάφος είναι εάν εντός της ατράκτου του αεροσκάφους υπάρχει πιλότος και όχι κατά πόσο το εν λόγο αεροσκάφος έχει την δυνατότητα να πετάει αυτόνομα. Για παράδειγμα ένα σύγχρονο επιβατικό αεροσκάφος όπως το Airbus A350 [11] έχει την δυνατότητα να εκτελέσει το σύνολο της πτήσης (μετακίνηση εντός του αεροδρομίου, απογείωση, πτήση, προσγείωση, παρκάρισμα στην σωστή πύλη προορισμού) απολύτως αυτόνομα χωρίς οι πιλότοι να επέμβουν, κι ας βρίσκονται στο πιλοτήριο. Αντίστοιχα υπάρχουν ΜΕΑ τα οποία δεν διαθέτουν καμία δυνατότητα αυτονομίας και όλοι οι χειρισμοί γίνονται από κάποιο πιλότο στο έδαφος και μεταδίδονται στο αεροσκάφος.

Από πλευράς αισθητήρων, τα ΜΕΑ κουβαλούν μια πληθώρα για πολλές χρήσεις. Οι πιο συνηθισμένοι είναι αυτού που έχουν να κάνουν με τον έλεγχο της πτήσης όπως για παράδειγμα Επιταχυνσιόμετρα, Γυροσκόπια, Πυξίδες και ΣΓΕ. Επιπλέον διαθέτουν τον κατάλληλο τηλεπικοινωνιακό εξοπλισμό όπως αναμεταδότες και κεραίες για την επικοινωνία με τους χειριστές. Τέλος διαθέτουν εξειδικευμένους αισθητήρες όπως Κάμερες, Radar κ.α. ανάλογα με τις ανάγκες της αποστολής που εκτελούν.

Όσων αφορά τους αισθητήρες που έχουν να κάνουν με την μέτρηση κίνησης, χρησιμοποιείται ένα μετρικό, ο λεγόμενος Βαθμός Ελευθερίας. Στα μηχανικά συστήματα [12], οι βαθμοί ελευθερίας είναι ο αριθμός των ανεξάρτητων παραμέτρων που μπορούν να ορίσουν τη κατάσταση ενός συστήματος. Στην περίπτωση των αεροσκαφών και των ΜΕΑ, οι βαθμοί ελευθερίας αναφέρονται στο πλήθος των παραμέτρων που μπορούν να καθορίσουν την θέση και την κίνηση του αεροσκάφους. Στον πίνακα 1.2 φαίνεται πως ορίζονται οι βαθμοί ελευθερίας στην αεροπλοΐα.

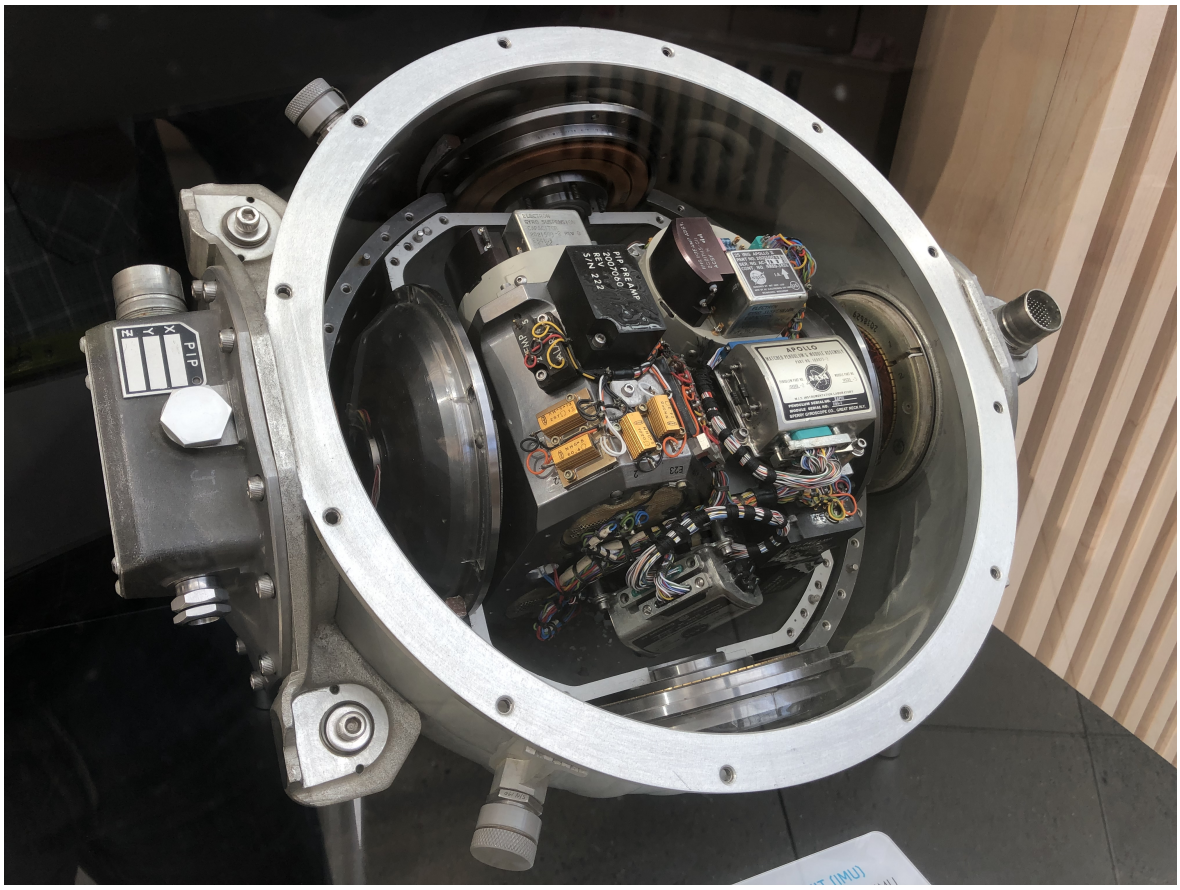
Βαθμοί Ελευθερίας	Επεξήγηση
6-DOF	$3^{ων}$ Αξόνων Επιταχυνσιόμετρο + Γυροσκόπιο (ΜΑΜ)
9-DOF	ΜΑΜ + Πυξίδα
10-DOF	ΜΑΜ + Πυξίδα + Βαρόμετρο
11-DOF	ΜΑΜ + Πυξίδα + Βαρόμετρο + ΣΓΕ

Πίνακας 1.2: Βαθμοί Ελευθερίας

Προσδιορισμός Θέσης και Κίνησης

Το σημαντικότερο πράγμα σε ένα αεροσκάφος είναι ο προσδιορισμός της θέσης και της κίνησης του στο χώρο κατά την πτήση. Αυτό επιτυγχάνεται με την συνδυασμό 2 αισθητήριων οργάνων. Του επιταχυνσιόμετρου και του γυροσκοπίου. Το επιταχυνσιόμετρο μετράει την επιτάχυνση σε κάθε έναν από τους 3 άξονες στο χώρο, ενώ το γυροσκόπιο μετράει την απόκλιση της γωνίας αυτών κατά την περιστροφή. Με τον συνδυασμό αυτών των δεδομένων, μπορούμε να εξάγουμε την θέση και την κίνηση του αεροσκάφους.

Στην αεροπλοΐα συχνά ο συνδυασμός επιταχυνσιόμετρου και γυροσκοπίου συναντάται σε ένα ενιαίο όργανο, την λεγόμενη Μονάδα Αδρανειακής Μέτρησης. [13]



Σχήμα 1.10: ΜΑΜ Αεροσκάφους
[13]

Το μειονέκτημα της ΜΑΜ είναι ότι βασίζεται σε γυροσκόπιο, τα οποία λόγω ατελούς κατασκευής με την πάροδο του χρόνου εισάγουν ένα σφάλμα στην μέτρηση. Αυτό το φαινόμενο ονομάζεται γυροσκοπική μετατόπιση και οφείλεται στις εσωτερικές τριβές του γυροσκοπίου ο οποίες μετριούνται ως περιστροφή. Έτσι ανεπαίσθητα μετριέται μια μη υπαρκτή περιστροφή.

Ο πιο απλός τρόπος να εξαλειφθεί το σφάλμα του γυροσκοπίου είναι, ανά τακτά περιοδικά διαστήματα να επαναφέρεται στο σημείο αναφοράς λαμβάνοντας υπόψιν κι' άλλους αισθητήρες όπως το ΣΓΕ, μαγνητική πυξίδα ακόμα και βαρόμετρο για τον προσδιορισμό της πραγματικής θέσης και κίνησης του αεροσκάφους. Ο συνδυασμός όλων των παραπάνω προσφέρει μεγάλη ακρίβεια και αξιοπιστία καθώς διαφορετικοί τρόποι μέτρησης διαφορετικών δεδομένων οδηγούν στο ίδιο εξαγόμενο συμπέρασμα.

Επικοινωνία και Τηλεμετρία

Η συντριπτική πλειοψηφία των ΜΕΑ διαθέτουν κάποιο είδος επικοινωνίας για την αποστολή διαγνωστικών δεδομένων στο έδαφος και λήψη εντολών από τους χειριστές τους. Επιπλέον, σχεδόν όλα τα ΜΕΑ σήμερα διαθέτουν τουλάχιστον και μια κάμερα, οπότε απαιτείται και η αποστολή της εν λόγο ροής βίντεο.

Σχήμα 1.11: Πομποδέκτες Τηλεμετρίας 433MHz από ΜΕΑ
[14]

Τα πρώτα ΜΕΑ διέθεταν ξεχωριστές ραδιοξεύξεις για αποστολή και λήψη δεδομένων, ενώ σήμερα είναι πιο συνηθισμένο να υπάρχει μια αμφίδρομη ζεύξη αρκετά μεγάλου εύρους ζώνης. Επιπλέον χρησιμοποιούνται τεχνικές ποιότητας υπηρεσιών για να δίνεται προτεραιότητα στα πακέτα που αφορούν την τηλεμετρία και τον χειρισμό του αεροσκάφους.

Τα περισσότερα ΜΕΑ επικοινωνούν με τους χειριστές τους μέσω επίγειων σταθμών βάσης. Ορισμένα που είναι για να καλύπτουν μεγαλύτερες αποστάσεις αλλά σε χαμηλό υψόμετρο μπορεί επίσης να επικοινωνούν μέσω δικτύων κινητής τηλεφωνίας. Τα ΜΕΑ στρατιωτικής χρήσης συνήθως επικοινωνούν μέσω δορυφόρων για να μην έχουν καμία εξάρτηση από υποδομές εδάφους και να είναι πιο δύσκολο να υποκλαπεί η επικοινωνία.

1.3 Λογισμικό ενός ΜΕΑ

Το λογισμικό των ΜΕΑ συχνά αποκαλείται flight stack ή autopilot. Είναι συστήματα πραγματικού χρόνου τα οποία απαιτούν ταχύτατη αντίδραση στα δεδομένα των αισθητήρων. Έχουν ένα εύρος από ιδιοκατασκευές με αισθητήρες και υπολογιστικές πλακέτες του εμπορίου έως σχεδιασμένα από το μηδέν ενσωματωμένα συστήματα με λειτουργικό σύστημα πραγματικού χρόνου. Τα πιο δημοφιλή ανοιχτού κώδικα λογισμικά πτήσης είναι τα ArduPilot, PX4, iNav.

Λειτουργικό Σύστημα

Τα πιο πολλά ΜΕΑ του εμπορίου δεν είναι αρκετά πολύπλοκα για να απαιτούν λειτουργικό σύστημα. Ένα λογισμικό που έχει γραφεί ακριβώς για το υλικό του ΜΕΑ αρκεί για να καλύψει όλες τις απαιτούμενες λειτουργίες.

Σε πιο μεγάλα και περίπλοκα ΜΕΑ όμως, ένα λειτουργικό σύστημα πραγματικού χρόνου είναι απαραίτητο. Σε πολύπλοκα αεροσκάφη, υπάρχουν δεκάδες ανεξάρτητα υποσυστήματα που επικοινωνούν μεταξύ τους και το κάθε ένα ελέγχει μια διαφορετική λειτουργία. Εκεί ο συντονισμός των διαφόρων υποσυστημάτων εκτελείται από ένα λειτουργικό σύστημα πραγματικού χρόνου. Είναι πιο εύκολο από πλευράς προγραμματισμού και ελέγχου να υπάρχει ένα κεντρικό σημείο που ελέγχει όλες τις λειτουργίες. Η απαίτηση να είναι πραγματικού χρόνου προκύπτει από το γεγονός ότι η απόκριση του ΜΕΑ πρέπει να είναι προβλέψιμη. Σε ένα λειτουργικό σύστημα με δυναμικό προγραμματισμό διεργασιών, δεν είναι δυνατές προβλέψεις για παράδειγμα: πότε θα εκτελεστεί τι και πόσο χρόνο θα διαρκέσει. Αυτό το αναλαμβάνει το λειτουργικό σύστημα αυτόματα και δυναμικά κατά τον χρόνο εκτέλεσης. Ενώ στα λειτουργικά συστήματα πραγματικού χρόνου, η σειρά και διάρκεια εκτέλεσης της κάθε διεργασίας είναι αυστηρά καθορισμένη από πριν, κάνοντας το σύστημα προβλέψιμο.

1.3.1 Σύστημα ΜΕΑ – Σταθμού Βάσης

Ένα σημαντικό κομμάτι των ΜΕΑ είναι η άμεση μετάδοση δεδομένων, και γι' αυτό έχουμε σταθμό βάσης (είτε στο έδαφος είτε μέσω δορυφορικής ζεύξης) με τον οποίο επικοινωνούμε αμφίδρομα. Μέσω αυτής της επικοινωνίας μπορούμε να ελέγχουμε το ΜΕΑ, να λαμβάνουμε πίσω τα δεδομένα τηλεμετρίας του καθώς και τα δεδομένα από τους διάφορους αισθητήρες που χρειαζόμαστε για την εκάστοτε αποστολή του.

1.3.2 Περιπτώσεις Χρήσης των ΜΕΑ

Ο τομέας των ΜΕΑ, αν και είναι αρκετά νέος και μετρά δύο δεκαετίες ζωής, έχει ήδη διεισδύσει σε πολλά σημεία της καθημερινής μας ζωής και μετρά δεκάδες εφαρμογές, άλλες πιο εμφανείς και άλλες όχι τόσο.

Αμυντικές Εφαρμογές

Σήμερα υπάρχουν αρκετές εκατοντάδες στρατιωτικού τύπου ΜΕΑ εν ενεργεία στις αεροπορίες ανά τον κόσμο. Οι πιο μεγάλες κατασκευάστριες εταιρίες είναι γνωστά ονόματα και παραδοσιακές δυνάμεις στο χώρο της αεροδιαστημικής [15] που προέρχονται από χώρες με ισχυρή αμυντική βιομηχανία. Ενδεικτικά αναφέρουμε ότι οι 7 μεγαλύτεροι

κατασκευαστές στρατιωτικών ΜΕΑ (με βάση το ύψος των πωλήσεων) είναι οι παρακάτω.

1. General Atomics (ΗΠΑ)
2. Northrop Grumman (ΗΠΑ)
3. Boeing (ΗΠΑ)
4. BAE Systems (Ηνωμένο Βασίλειο)
5. IAI (Ισραήλ)
6. Dassault Aviation (Γαλλία)
7. Nexter (Γαλλία)

Η πιο συνηθισμένη στρατιωτική εφαρμογή των ΜΕΑ ήταν αυτή της συλλογής πληροφοριών. Το πλεονέκτημα της χρήσης ΜΕΑ για την συλλογή πληροφοριών είναι η ταχύτητα κίνησης του, ιδίως για παρακολούθηση στόχων εδάφους, και η δυνατότητα τους να εισέρχονται σε εχθρικό εναέριο χώρο που θα ήταν κίνηση υψηλού κινδύνου για επανδρωμένο αεροσκάφος [16].

Ότι δεδομένα συλλέγονται [17] μεταδίδονται σε πραγματικό χρόνο στους χειριστές του ΜΕΑ. Η συλλογή των δεδομένων από τους διάφορους αισθητήρες του ΜΕΑ μπορεί να είναι είτε πλήρως χειροκίνητη, είτε αυτοματοποιημένη με βάση διάφορες παραμέτρους (π.χ. ανίχνευση κίνησης).

Τυπικό παράδειγμα ενός ΜΕΑ αυτής της κατηγορίας είναι το Heron της IAI που εικονίζεται παρακάτω.

Σχήμα 1.12: IAI Heron - ΜΕΑ Στρατιωτικής Χρήσης
[18]

Το συγκεκριμένο ΜΕΑ το χρησιμοποιεί και η Ελλάδα [19] για την επιτήρηση των συ-

νόρων του εναέριου και θαλάσσιου χώρου.

Είναι ένα ΜΕΑ μέσου υψομέτρου και μακράς διάρκειας πτήσης. Μπορεί να πετάει στα 35,000ft (10,5km) για έως και 52 ώρες. Το Heron μπορεί να κουβαλάει φορτίο έως 250kg σε αισθητήρες με τα πιο συνηθισμένα να είναι κάμερες για διάφορα φάσματα (υπέρυθρο, ορατό) και Radar. Επιπλέον για την επικοινωνία διαθέτει και πανκατευθυντικούς πομποδέκτες στο κάτω μέρος της ατράκτου για σύνδεση με σταθμούς εδάφους, και δορυφορική ζεύξη στο άνω μέρος της ατράκτου για επικοινωνία μέσω δορυφόρου. Το σύστημα πλοήγησης είναι ένας συνδυασμός ΜΑΜ και ΣΓΕ. Μπορεί είτε πλήρως αυτόνομα να ακολουθήσει ένα προγραμματισμένο σχέδιο πτήσης είτε να είναι πλήρως τηλεχειριζόμενο από χειριστή στο έδαφος.

Το μέγιστο βάρος κατά την απογείωση είναι 1150kg. Κινείται από έναν 4κύλινδρο βενζινοκινητήρα απόδοσης 86kW (115hp) οποίος περιστρέφει μια προπέλα 3 στοιχείων στο πίσω μέρος της ατράκτου. Η μέγιστη ταχύτητα που μπορεί να αναπτύξει σε σταθερή πτήση είναι 207km/h.

Έρευνα και Διάσωση

Μέχρι πριν λίγα χρόνια, τα κλιμάκια έρευνας και διάσωσης ανά τον κόσμο όταν επιχειρούσαν βασίζονταν σε πεζοπόρες και μηχανοκίνητες μονάδες για την πλειάδα των αποστολών τους, αφήνονταν τα εναέρια μέσα (κατά κύριο λόγο επανδρωμένα ελικόπτερα) μόνο για εξαιρετικά σοβαρά περιστατικά. Ο λόγος είναι ότι τα επανδρωμένα ελικόπτερα είναι ακριβά στην απόκτηση και χρήση, και σχεδόν ποτέ αρκετά. Οπότε είναι η έσχατη λύση όταν δεν υπάρχει εναλλακτική.

Σε περιπτώσεις όμως που πρέπει να αντιμετωπιστούν μεγάλες καταστροφές (φυσικές ή μη) ή η περιοχή έρευνας είναι πολύ μεγάλη, το να υπάρχει τέτοια έλλειψη σε εναέρια μέσα μπορεί να αποβεί μοιραίο σε ανθρώπους που βρίσκονται σε κίνδυνο και ο χρόνος μετράει εναντίων τους. [20]

Με την πρόοδο όμως της τεχνολογίας των ΜΕΑ, βλέπουμε όλο και πιο συχνά να χρησιμοποιούνται για τέτοιου είδους αποστολές καλύπτοντας αυτό το κενό. Από μικρά ηλεκτροκίνητα ΜΕΑ για ταχύτητα κάλυψη μικρών περιοχών (π.χ. κατάρρευση κτηρίου μετά από σεισμό) μέχρι μεγάλα στρατιωτικού τύπου ΜΕΑ που πετούν σε μεγάλο υψόμετρο για ώρες και μπορούν να σαρώσουν πολύ μεγάλες περιοχές (π.χ. εύρεση χαμένου σκάφους σε κακοκαιρία).

Χαρακτηριστικό παράδειγμα μικρού ΜΕΑ της πρώτης κατηγορίας είναι το Matrice 210 της DJI στην εικόνα 1.13.

Σχήμα 1.13: DJI Matrice 210 - ΜΕΑ Έρευνας και Διάσωσης
[21]

Είναι ένα μικρό φορητό (το ΜΕΑ και ο σταθμός βάσης του χωράνε σε μια βαλίτσα) ηλεκτρικό τετρακόπτερο σχεδιασμένο για εφαρμογές έρευνας και διάσωσης σε αστικά περιβάλλοντα. Διαθέτει δύο κάμερες (μία έγχρωμη και μία θερμική) οι οποίες διαθέτουν γυροσκοπική σταθεροποίηση 2 αξόνων, συν σημεία πρόσδεσης επιπλέον εξαρτημάτων όπως φώτα, επιπλέον κάμερες κλπ.

Μπορεί να πετάξει σε μια ακτίνα 8km γύρω από τον σταθμό βάσης του κουβαλώντας μέγιστο φορτίο 1,4kg. Επιπλέον διαθέτει πιστοποίηση IP43 για αντοχή σε νερό και σκόνη επιτρέποντας το να πετάει και υπό τέτοιες συνθήκες. Και το σημαντικό είναι ότι το κιτ με το ΜΕΑ και τον σταθμό βάσης κοστίζει 6500$ και τα λειτουργικά του έξοδα είναι πολύ χαμηλά, σε αντίθεση με ένα επανδρωμένο ελικόπτερο που κοστίζει εκατομμύρια για την αγορά του και πολλές χιλιάδες ανά ώρα πτήσης σε καύσιμα και συντήρηση.

Στον αντίποδα, ένα παράδειγμα στρατιωτικού ΜΕΑ που χρησιμοποιείται και σε περιπτώσεις Έρευνας και Διάσωσης είναι το RQ-4 Global Hawk της Northrop Grumman. Είναι ένα από τα κορυφαία στρατιωτικά ΜΕΑ Συλλογής Πληροφοριών αλλά συχνά βοηθάει και σε αποστολές έρευνας διάσωσης και επιτήρησης συνόρων.

Σχήμα 1.14: Northrop Grumman RQ-4 - MEA Μικτής Χρήσης
[22]

Το RQ-4 είναι ένα από τα μεγαλύτερα MEA που υπάρχουν στον κόσμο με μήκος 14,5m, άνοιγμα φτερών 39,9m και βάρος πλήρες υγρών 14,628kg. Πηγή ισχύος του είναι ένας turbofan κινητήρας F137-RR-100 της Rolls Royce με μέγιστη παραγόμενη ώση 34kN.

Από πλευράς επιδόσεων μπορεί να πετάει στα 60,000ft (18,000m) για 32 ώρες με ταχύτητα 570km/h. Η εμβέλεια του είναι της τάξης των 22,780km. Στον βασικό του εξοπλισμό περιλαμβάνει ένα υψηλής ανάλυσης ραντάρ συνθετικού διαφράγματος [23] καθώς και κάμερες για το υπέρυθρο και το ορατό φάσμα. Με αυτούς τους αισθητήρες και τις επιδόσεις που διαθέτει μπορεί να σαρώνει μια περιοχή 100,000km^2 σε 24 ώρες, δηλαδή μια έκταση όσο η Ισλανδία. [24]

Γεωργία

Άλλη μια μεγάλη εφαρμογή των MEA είναι αυτή της παρακολούθησης και της διαχείρισης καλλιεργειών. Μέχρι πρότινος αυτή η δουλεία γινόταν κυρίως με δορυφορικές φωτογραφίες. Το πρόβλημα με αυτές όμως είναι διπλό. Πρώτον η απόκτηση τους είναι ακριβή και δεν γίνεται άμεσα, καθώς ο δορυφόρος θα πρέπει να περάσει πάνω από την περιοχή ενδιαφέροντος την κατάλληλη χρονική στιγμή. Και δεύτερον, εάν υπάρχει συννεφοκάλυψη η λήψη φωτογραφιών είναι σχεδόν αδύνατη.

Η χρήση MEA μας επιτρέπει γρήγορα να δούμε οποιαδήποτε περιοχή μας ενδιαφέρει και μάλιστα με καλύτερη ευκρίνεια από τις εμπορικά διαθέσιμες υπηρεσίες δορυφορικών φωτογραφιών. Ιδίως αν έχουμε να κάνουμε με μικρές εκτάσεις, η χρήση MEA είναι σημαντικά πιο οικονομική. Επιπλέον, λόγο της φύσης των MEA μπορούμε να

πάρουμε λήψεις από διάφορες οπτικές γωνίες και να εξετάσουμε με μεγαλύτερη λεπτομέρεια. [25, 26]

Πρόβλεψη Καιρού

Μετά την χρήση μετεωρολογικών σταθμών και δορυφορικών εικόνων, και τα μη επανδρωμένα αεροσκάφη μπαίνουν στην επιστήμη της μετεωρολογίας. Ένα ΜΕΑ που μεταφέρει έναν φορητό μετεωρολογικό σταθμό μπορεί να πάει να πάρει μετρήσεις από πολλαπλά σημεία και άμεσα να μεταδώσει τις εν λόγω μετρήσεις. Επίσης μπορεί να πετάξει κατά τη διάρκεια ακραίων καιρικών φαινομένων χωρίς να θέσει σε κίνδυνο ανθρώπινες ζωές.

Ιδίως στη δεύτερη περίπτωση της μελέτης ακραίων καιρικών φαινομένων βλέπουμε την μεγαλύτερη αύξηση στην χρήση ΜΕΑ από πλευράς μετεωρολογίας. Σε τέτοιες συνθήκες, συνήθως έχουμε έντονη συννεφοκάλυψη, πράγμα που αχρηστεύει τους περισσότερους τύπους δορυφορικών εικόνων. Ταυτόχρονα είναι πιο επικίνδυνο για κάποιο άνθρωπο να βγει και να πάρει τις απαιτούμενες μετρήσεις. [27]

Ερασιτεχνική Χρήση

Η τελευταία και πιο μεγάλη εφαρμογή των ΜΕΑ είναι αυτών της ερασιτεχνικής χρήσης. Από μικρά ΜΕΑ-παιχνίδια για παιδιά, εμπορικά ΜΕΑ για λήψη φωτογραφικών και βίντεο μέχρι μεγάλες ιδιοκατασκευές αερομοντελιστών για προσωπική χρήση.

Ιδίως τα εμπορικά ΜΕΑ όπως το εικονιζόμενο DJI Phantom 4 είναι αυτά που έχουν πιάσει το μεγαλύτερο κομμάτι της αγοράς.

Σχήμα 1.15: DJI Phantom 4 Pro - ΜΕΑ Ερασιτεχνικής Χρήσης
[28]

Αρκετά οικονομικά στην αγορά, εύκολα στο χειρισμό και αρκετά μικρά για της δυ-

νατότητες που προσφέρουν. Χρησιμοποιούνται συνήθως είτε σαν παιχνίδι είτε σαν συσκευή ερασιτεχνικής κινηματογράφησης.

Κεφάλαιο 2

Συστήματα Πολλαπλών ΜΕΑ

2.1 Συνοπτική Περιγραφή Συστημάτων Πολλαπλών ΜΕΑ

Ένα Σύστημα Πολλαπλών ΜΕΑ αποτελείται από πολλαπλά ΜΕΑ ή/και σταθμούς βάσης και χειριστές που συνεργάζονται μεταξύ τους για την εκτέλεση μιας αποστολής.

Στο προηγούμενο κεφάλαιο αναλύθηκαν διάφορες εφαρμογές των ΜΕΑ. Στα πιο πολλά παραδείγματα αναφέρεται ότι χρησιμοποιείται ένα ΜΕΑ για την εκάστοτε αποστολή. Όμως, στην πλειοψηφία των περιπτώσεων είναι πιο συμφέρον να χρησιμοποιηθεί ένας στόλος από ΜΕΑ και αυτό φέρνει πολλά πλεονεκτήματα.

2.2 Πλεονεκτήματα

2.2.1 Εξοικονόμηση Χρόνου

Σε εφαρμογές όπως χαρτογράφηση που πρέπει να καλυφθεί μεγάλη έκταση γης, είναι σημαντικά πιο γρήγορο να χρησιμοποιηθούν πολλαπλά ΜΕΑ που θα μοιράσουν την περιοχή ενδιαφέροντος μεταξύ τους παρά να υπάρχει ένα ΜΕΑ το οποίο θα σαρώνει ένα κομμάτι, θα επιστρέφει στην βάση για φόρτιση/ανεφοδιασμό και θα απογειώνεται πάλι για να συνεχίσει από εκεί που έμεινε. Ιδίως σε αγροτικές εφαρμογές αυτό μπορεί να έχει μεγάλη επίδραση στην ταχύτητα λήψης των δεδομένων και συνεπώς την πιο γρήγορη λήψη αποφάσεων. [29, 30]

2.2.2 Κόστος

Σε ορισμένες περιπτώσεις η χρήση ενός μεγάλου ΜΕΑ που θα μπορεί να καλύψει όλες τις ανάγκες μας ενδέχεται να είναι πιο ακριβή από την χρήση ενός στόλου πολλαπλών μικρότερων ΜΕΑ. Τέτοιες εφαρμογές όπως η παρακολούθηση καλλιεργειών, η τηλεπισκόπιση και γενικά οι εφαρμογές που απαιτούν τη σάρωση μεγάλων εκτάσεων συνήθως επωφελούνται από την χρήση συστημάτων πολλαπλών ΜΕΑ. Ένα μεγάλο ΜΕΑ που θα μπορεί να καλύψει μονομιάς δεκάδες τετραγωνικά χιλιόμετρα είναι πολύ ακριβό, σε αντίθεση με μικρά ηλεκτροκίνητα ΜΕΑ που ναι μεν θα καλύπτουν πολύ μικρότερη

έκταση σε κάθε πτήση, αλλά λόγο της συνεργασίας μεταξύ τους θα παρέχουν την ίδια ή και καλύτερη κάλυψη. [30]

2.2.3 Ταυτόχρονη Εκτέλεση Ενεργειών

Ένας στόλος από ΜΕΑ μπορεί να εκτελέσει διαφορετικές ενέργειες ταυτόχρονα σε διαφορετικές τοποθεσίες, σε αντίθεση με ένα μόνο ΜΕΑ που θα πρέπει να αλλάξει τοποθεσία, ή και να επιστρέψει στην βάση σου για να αλλάξει εξοπλισμό (π.χ. κάμερες, αισθητήρες) εάν αυτοί που διαθέτει εκείνη την στιγμή δεν είναι οι κατάλληλοι. [30]

2.2.4 Συμπληρωματικότητα

Σε ορισμένες αποστολές που απαιτούνται να χρησιμοποιηθούν πολλαπλοί αισθητήρες και εξοπλισμός, μπορεί να είναι δύσκολο να φορτωθούν όλα σε ένα και μόνο ΜΕΑ. Σε αυτή την περίπτωση είτε πρέπει να χρησιμοποιηθεί κάποιο μεγάλο και ακριβό ΜΕΑ (συνήθως στρατιωτικού τύπου), είτε η άλλη λύση είναι όλος αυτός ο εξοπλισμός να μοιραστεί σε μια ομάδα ΜΕΑ και αυτή να συνεργαστεί προκειμένου να εκτελέσει την αποστολή. Για παράδειγμα σε μια αποστολή έρευνας και διάσωσης μπορεί να υπάρχουν πολλαπλά εμπορικά ΜΕΑ που το καθένα να κουβαλάει από ένα κομμάτι εξοπλισμού (έγχρωμη κάμερα, θερμική κάμερα, ραντάρ, ηχοεντοπιστικό σύστημα σόναρ, κιτ πρώτων βοηθειών, τροφή και νερό για παράδοση σε δυσπρόσιτα μέρη κλπ.). Όλα αυτά εάν έπρεπε να χωρέσουν σε ένα ΜΕΑ ενδέχεται να χρειαζόταν κάποιο μεγάλο αεροσκάφος με σημαντικά μεγαλύτερο κόστος κτήσης και χρήσης. [30, 31]

2.2.5 Αξιοπιστία

Όταν χρησιμοποιείται ένα ΜΕΑ, εάν υπάρξει κάποια μηχανική βλάβη ή αστοχία, η συνέχιση της αποστολής θα είναι αδύνατη. Όμως όταν υπάρχουν πολλαπλά ΜΕΑ, είναι δυνατόν με δυναμικό διαχωρισμό των εργασιών, η απώλεια ενός η περισσότερων αεροσκαφών να μπορεί να αντισταθμιστεί χωρίς να επηρεάσει την έκβαση της αποστολής. Ιδίως σε περιπτώσεις όπως έρευνα και διάσωση αυτό μπορεί να είναι ζωτικής σημασίας. [30, 32]

2.2.6 Ευελιξία Εργασιών

Ένα ΜΕΑ μπορεί να εκτελέσει μια εργασία ανά πάσα στιγμή. Μια ομάδα από ΜΕΑ όμως μπορεί να αναθέτει στα διάφορα ΜΕΑ της διάφορες εργασίες και αυτός ο διαχωρισμός να γίνεται δυναμικά βάσει τον αναγκών. [30]

2.3 Μειονεκτήματα

2.3.1 Νομικοί Περιορισμοί

Πολλές περιοχές του κόσμου έχουν νομικούς περιορισμούς σχετικά με τη χρήση ενός συστήματος πολλαπλών ΜΕΑ και μέχρι σήμερα δεν υπάρχει κάποιο παγκόσμιο στάνταρ που να καθορίζει την λειτουργία τους. Για παράδειγμα, στις ΗΠΑ [33] η χρήση των εν λόγω συστημάτων απαγορεύεται για ιδιωτική και εμπορική χρήση.

2.3.2 Πολυπλοκότητα Χειρισμού

Τα σημερινά συστήματα ΜΕΑ πέρα από τις όποιες δυνατότητες αυτονομίας διαθέτουν, μπορούν ανά πάσα στιγμή να είναι και τηλεχειριζόμενα από κάποιο πιλότο. Όμως όταν υπάρχουν ταυτόχρονα πολλά ΜΕΑ εν πτήση, ο χειρισμός τους από ένα και μόνο άτομο γίνεται πολύ δύσκολος. Ένας πιλότος μπορεί να εστιάσει την προσοχή του σε ένα ΜΕΑ την φορά και είναι εξαιρετικά δύσκολο να παρακολουθεί τι κάνουν τα υπόλοιπα. Έτσι είναι απαραίτητο να υπάρχουν συστήματα που βοηθούν στο χειρισμό των συστημάτων πολλαπλών ΜΕΑ από ένα άτομο. [30]

Άλλο ένα σημαντικό πρόβλημα είναι αυτό τις αποφυγής συγκρούσεων μεταξύ των ΜΕΑ που πετούν στην ίδια περιοχή. Θα πρέπει οι χειριστές στο έδαφος αλλά και τα ΜΕΑ μεταξύ τους να διαμοιράζονται συνεχώς τις θέσεις τους και να μπορούν να αποφύγουν αυτόνομα μια σύγκρουση χωρίς να χρειάζεται ανθρώπινη παρέμβαση.

2.4 Προκλήσεις

2.4.1 Διαχείριση Ενέργειας

Όταν έχουμε να διαχειριστούμε πολλαπλά ΜΕΑ ταυτόχρονα, πρέπει να λάβουμε υπόψιν πως θα χρησιμοποιήσουμε την διαθέσιμη ενέργεια του στόλου. Η πιο απλή επιλογή είναι να πετάει όλος ο στόλος ταυτόχρονα σαν να είχαμε ένα ΜΕΑ και απλά να διαχωρίσουμε τις εργασίες μεταξύ τους. Μπορούμε όμως να υλοποιήσουμε και πιο σύνθετα σχήματα όπου κάποια ΜΕΑ θα εκτελούν τις εργασίες, κάποια άλλα θα επιστρέφουν στη βάση τους για ανεφοδιασμό και κάποια μπορούν να βρίσκονται σε κοντινή περιοχή για να αντικαταστήσουν τα ΜΕΑ που αποσύρονται. Τέτοια σύνθετα σχήματα απαιτούν αρκετά περίπλοκο λογισμικό διαχείρισης και τεχνικές εξοικονόμησης ενέργειας σε κάθε ΜΕΑ. [33]

2.4.2 Αποφυγή Συγκρούσεων και Έλεγχος Σμήνους

Όπως αναφέρθηκε και στα μειονεκτήματα των συστημάτων πολλαπλών ΜΕΑ, ο ασφαλής και αξιόπιστος έλεγχος του σμήνους είναι μια αρκετά δύσκολη διαδικασία. Από την μία είναι σχεδόν αδύνατον για ένα πιλότο να ελέγξει πολλαπλά ΜΕΑ, έτσι απαιτείται μεγάλη αυτονομία από τα αεροσκάφη για να εκτελούν ενέργειες χωρίς την επέμβαση του πιλότου. Από την άλλη, με την ύπαρξη ενός συστήματος πολλαπλών ΜΕΑ σε περιορισμένο εναέριο χώρο, αυξάνεται η πιθανότητα για εναέριες συγκρούσεις. Είτε οι αποφάσεις λαμβάνονται κεντρικά από κάποιο σταθμό βάσης, είτε από κάθε ΜΕΑ ξεχωριστά για την διαδρομή που θα ακολουθήσει το κάθε αεροσκάφος, θα πρέπει να λαμβάνεται υπόψιν η θέση του κάθε ΜΕΑ ανά πάσα στιγμή. Αυτό αυξάνει την επεξεργαστική ισχύ που απαιτείται για τους υπολογισμούς των διαδρομών καθώς και τις ανάγκες για μεταφορά δεδομένων μεταξύ των ΜΕΑ και τον σταθμών βάσης. Επιπλέον, σε περίπτωση που κάποια ΜΕΑ τύχει να πλησιάσουν πολύ κοντά μεταξύ τους θα πρέπει αυτόνομα να μπορούν να αποφύγουν την σύγκρουση χωρίς την επέμβαση πιλότου. [33]

2.4.3 Επικοινωνία Μεταξύ ΜΕΑ και Σταθμών Βάσης

Καθώς υπάρχουν πολλαπλά ΜΕΑ που κινούνται συνεχώς στον χώρο, η μη σταθερή τοπολογία του δικτύου, οι γρήγορες και απρόβλεπτες αλλαγές δυσκολεύουν την επι-

κοινωνία τόσο των ΜΕΑ με τους σταθμούς βάσης όσο και των ΜΕΑ μεταξύ τους. Για το ιδιαίτερο αυτό περιβάλλον, έχουν προταθεί τα δίκτυα ΙΑΔ (Ιπτάμενα Αυτοοργανωμένα Δίκτυα), τα οποία αναλύονται παρακάτω.

2.5 Ιπτάμενα Αυτοοργανωμένα Δίκτυα (ΙΑΔ)

2.5.1 Ορισμός: Τι είναι ένα ΙΑΔ

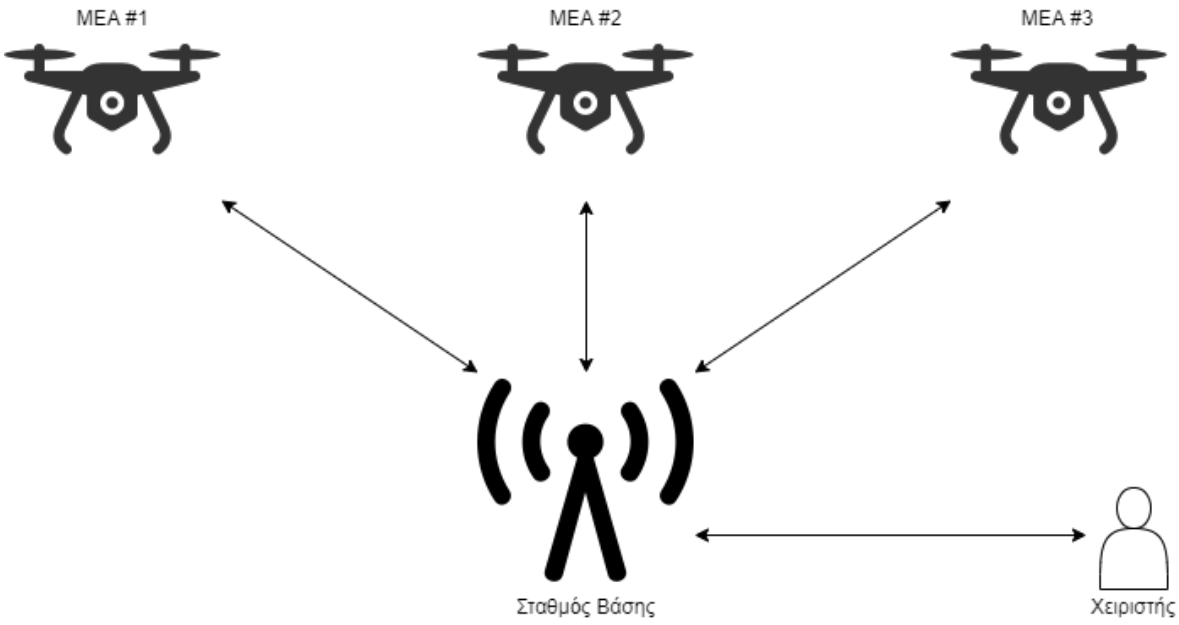
Τα ΙΑΔ είναι υποκατηγορία των ασύρματων αυτοοργανωμένων δικτύων. Αυτοοργανωμένα σημαίνει ότι δεν βασίζονται σε κάποια υφιστάμενη υποδομή όπως δρομολογητές η συγκεκριμένα σημεία πρόσβασης, αλλά ο κάθε κόμβος δρομολογεί τα δεδομένα στους υπόλοιπους κόμβους με τους οποίους έχει επικοινωνία. Η διαδρομή που θα ακολουθήσουν τα πακέτα μέσα στο δίκτυο καθορίζεται δυναμικά με βάση των αλγόριθμό δρομολόγησης που χρησιμοποιείται. Τα ΙΑΔ βασίζονται στην ίδια αρχή όπου τα ΜΕΑ είναι οι κόμβοι του δικτύου και δρομολόγηση των πακέτων για την επικοινωνία ΜΕΑ-ΜΕΑ και ΜΕΑ-Σταθμού Βάσης γίνεται δυναμικά. [34]

2.5.2 Οργάνωση των ΙΑΔ

Ένα δίκτυο ΙΑΔ μπορεί να οργανωθεί με τρεις τρόπους. Μπορεί να έχει κεντρική οργάνωση, οργάνωση ενός συστήματος πολλαπλών ομάδων ή κυψελωτή οργάνωση. [35]

Κεντρική Οργάνωση

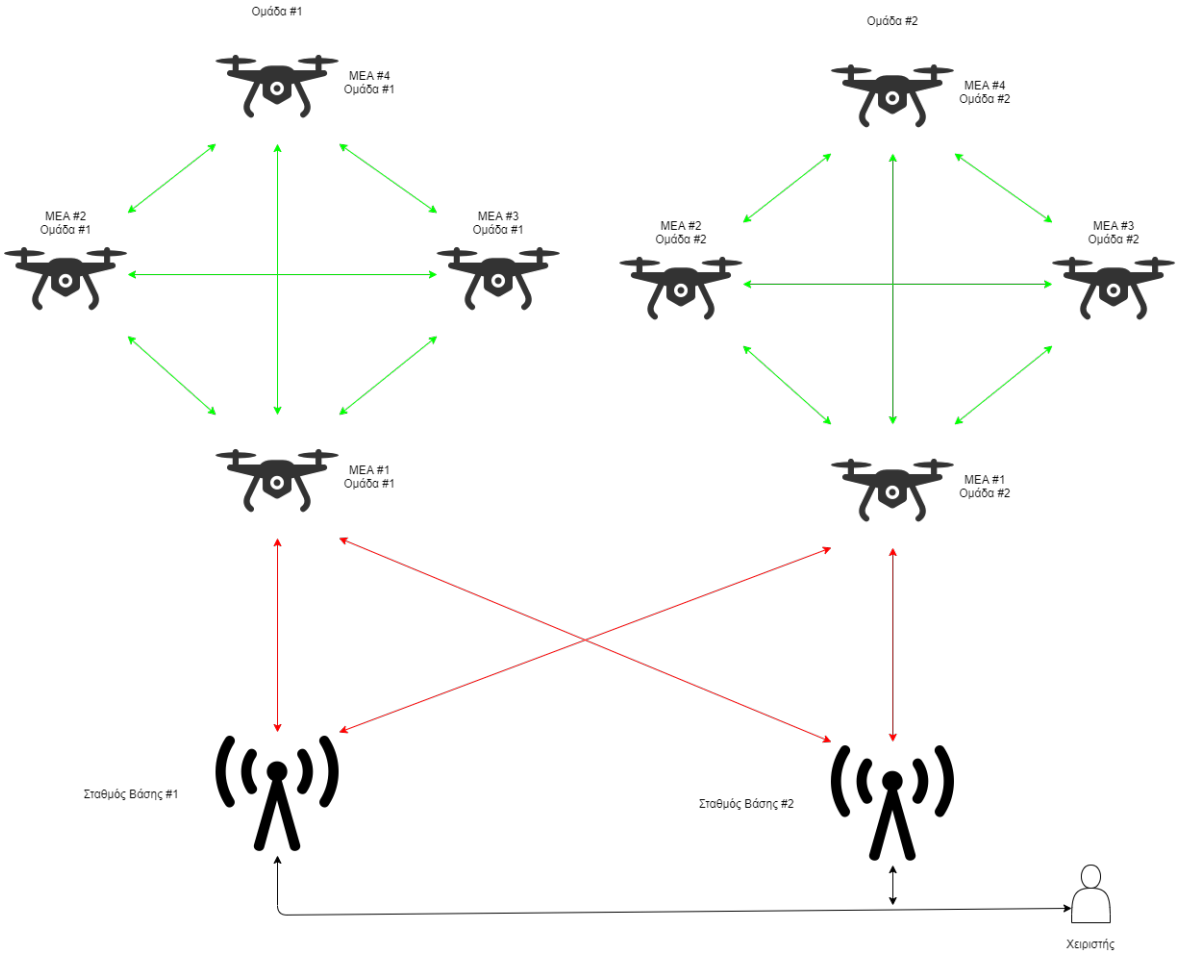
Στα ΙΑΔ με κεντρική οργάνωση, όλα τα ΜΕΑ επικοινωνούν απευθείας με έναν η περισσότερους σταθμούς βάσης. Η επικοινωνία μεταξύ των ΜΕΑ απευθείας δεν υφίσταται και γίνεται διαμέσου των σταθμών βάσης. Αυτή ο οργάνωση έχει το πλεονέκτημα ότι απλοποιεί την κατασκευή των ΜΕΑ καθώς πρέπει να επικοινωνούν μόνο με τους σταθμούς βάσης που είναι σταθεροί και πεπερασμένοι σε αριθμό καθώς και ότι αυξάνει την αξιοπιστία καθώς δεν εξαρτάται η δρομολόγηση των πακέτων από αναξιόπιστους ιπτάμενους κόμβους. Έχει όμως και το μειονέκτημα της υψηλής καθυστέρησης για την επικοινωνία μεταξύ των ΜΕΑ καθώς η κίνηση πρέπει να δρομολογηθεί μέσω των σταθμών βάσης. Επιπλέον εισάγεται ένα σημείο αναξιοπιστίας, ο σταθμός βάσης. Αν για οποιοδήποτε λόγο δεν λειτουργεί σωστά τότε χάνεται η επικοινωνία όλων των ΜΕΑ στην περιοχή του.



Σχήμα 2.1: Κεντρική Οργάνωση ΙΑΔ

Οργάνωση Ενός Συστήματος Πολλαπλών Ομάδων

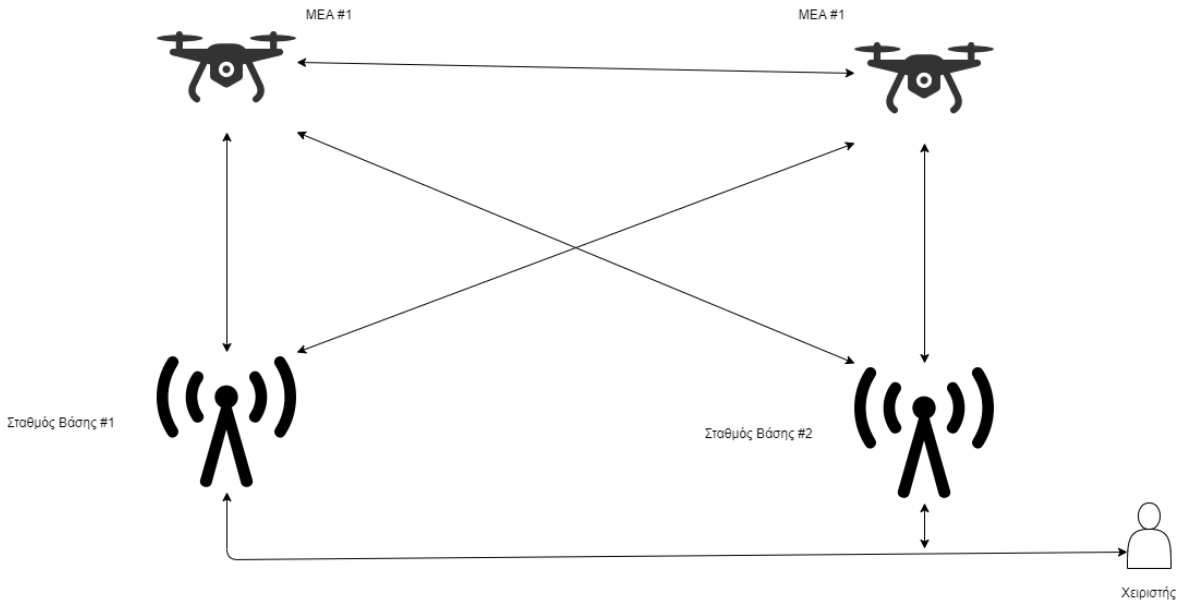
Στα ΙΑΔ με οργάνωση ενός συστήματος πολλαπλών ομάδων, διατηρείται η κεντρική οργάνωση της προηγούμενης κατηγορίας με την διαφορά ότι τώρα έχουμε πολλαπλές τέτοιες ομάδες στον αέρα και τα μέλη της κάθε ομάδας μπορούν να επικοινωνούν με μεταξύ τους. Όμως η επικοινωνία μεταξύ των ομάδων γίνεται μέσω των σταθμών βάσης. Αυτού του είδους η οργάνωση παρέχει καλύτερη απόδοση από την προηγούμενη κατηγορία αν και πάλι υπάρχει ένα μοναδικό σημείο αποτυχίας, ο σταθμός βάσης καθώς αν πέσει να κοπεί και η επικοινωνία μεταξύ των ομάδων. Εντός της κάθε ομάδας όμως τα ΜΕΑ συνεχίζουν να επικοινωνούν μεταξύ τους.



Σχήμα 2.2: Οργάνωση ΙΑΔ Πολλαπλών Ομάδων

Κυψελωτή Οργάνωση

Τελευταίο είδος είναι αυτό της κυψελωτής οργάνωσης. Αυτή μοιάζει πολύ με την οργάνωση των δικτύων κινητής τηλεφωνία όπου υπάρχουν σταθμοί βάσης στην περιοχή ενδιαφέροντος και τα ΜΕΑ που πετάνε σε αυτή συνδέονται κάθε φορά σε κάποιο από αυτούς με βάσης κριτήρια απόστασης, ποιότητας σύνδεσης κλπ. Επιπλέον τα ΜΕΑ μπορούν να επικοινωνούν και μεταξύ τους χωρίς την διαμεσολάβηση των σταθμών βάσης. Τέλος, αν κάποιο ΜΕΑ είναι εκτός εμβέλεια των σταθμών βάσης αλλά εντός εμβέλειας άλλων ΜΕΑ, μπορεί να δρομολογήσει τα πακέτα μέσω άλλων ΜΕΑ μέχρι να φτάσουν στους σταθμούς βάσης.



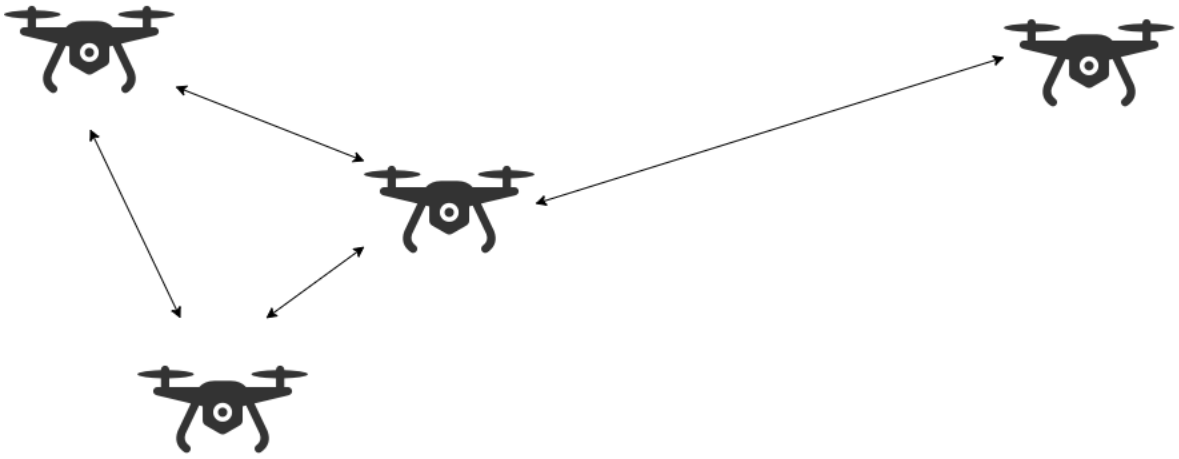
Σχήμα 2.3: Κυψελωτή Οργάνωση ΙΑΔ

2.5.3 Τύποι Σύνδεσης σε ΙΑΔ

Οι κόμβοι σε ένα σύστημα μπορούν να λειτουργήσουν ως αναμεταδότες προκειμένου να ανταπεξέλθουν στην συνεχώς μεταβαλλόμενη τοπολογία του δικτύου και να επιτευχθεί η επικοινωνία μεταξύ διάφορων μελών του δικτύου που ειδάλλως δεν θα είχαν την δυνατότητα να επικοινωνήσουν. Αυτή η επικοινωνία γίνεται με 3 τρόπους στα ΙΑΔ. [35]

Επικοινωνία ΜΕΑ-ΜΕΑ

Η πιο συνηθισμένη μορφή επικοινωνίας σε ένα σμήνος ΜΕΑ είναι των αεροσκαφών μεταξύ τους. Κάθε ΜΕΑ μπορεί να επικοινωνήσει με οποιοδήποτε άλλο ΜΕΑ είτε απευθείας είτε χρησιμοποιώντας τα υπόλοιπα ως αναμεταδότες για να προωθήσουν τα πακέτα στον τελικό τους προορισμό. Αυτή η τεχνική λειτουργεί πολύ καλά όταν πετούν σε ανοιχτό χώρο χωρίς εμπόδια και παρεμβολές, αλλά δυσκολεύονται σε αστικά περιβάλλοντα ιδίως όταν χάνεται η οπτική επαφή. Αυτή η συνεχής μεταβολή της κατάστασης των συνδέσεων μεταξύ των ΜΕΑ είναι που κάνει την δρομολόγηση πακέτων στα δίκτυα ΙΑΔ δύσκολη καθώς δεν υπάρχει σταθερή τοπολογία του δικτύου.

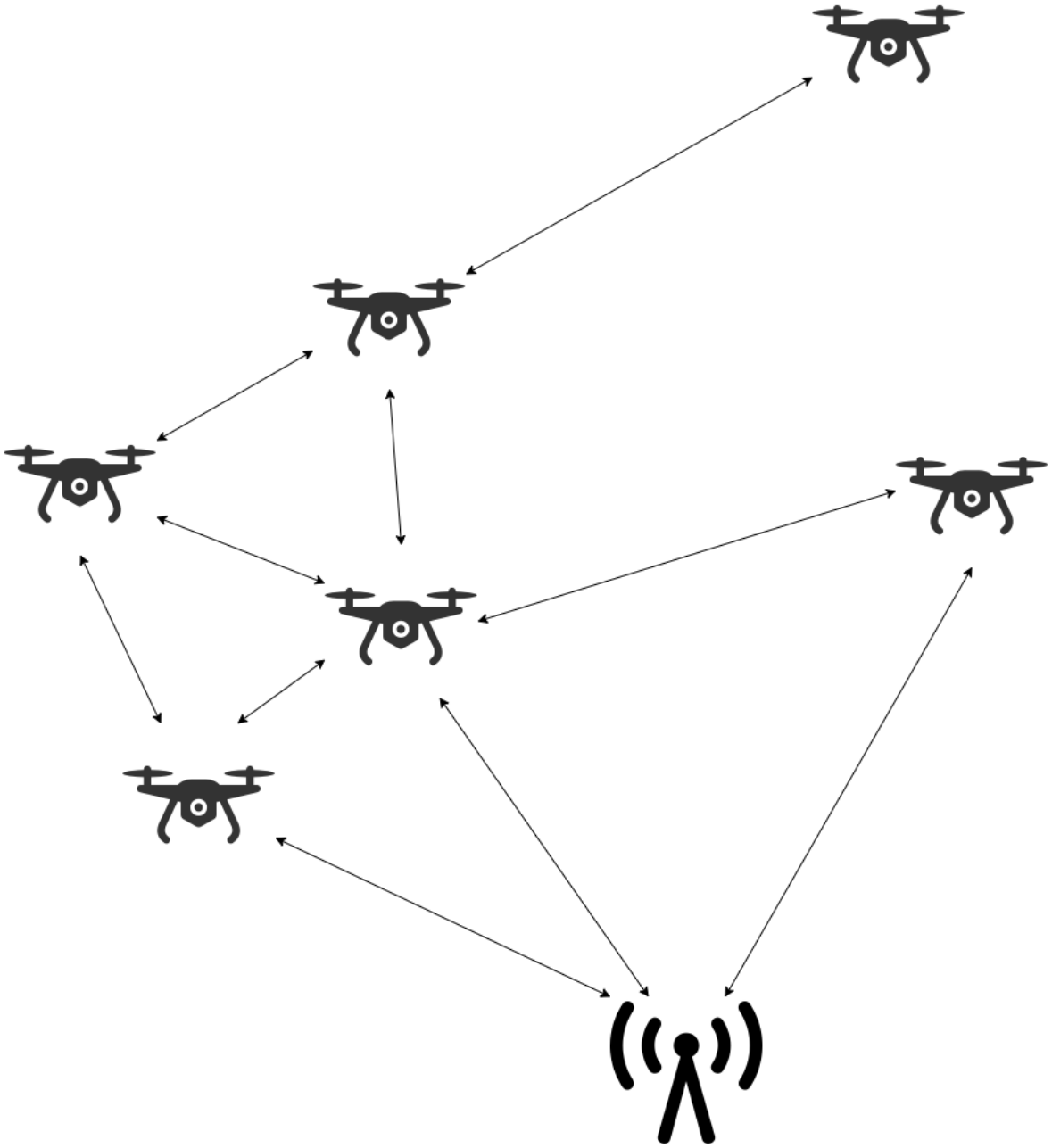


Σχήμα 2.4: Επικοινωνία ΜΕΑ - ΜΕΑ

Επικοινωνία ΜΕΑ-Σταθμού Βάσης

Όπως στα συστήματα με ένα ΜΕΑ, έτσι και στα συστήματα με πολλαπλά ΜΕΑ υπάρχουν ένας η περισσότεροι σταθμοί βάσης για την επικοινωνία των χειριστών με τα αεροσκάφη. Μέσω αυτών των σταθμών βάσης που μπορεί είτε να είναι μόνιμοι σε μια περιοχή είτε φορητοί και να στήνονται με βάση τις ανάγκες της κάθε αποστολής είναι δυνατός ο έλεγχος των ΜΕΑ. Επιπλέον δύναται τα ΜΕΑ να είναι χωρισμένα σε ομάδες οι οποίες για να επικοινωνήσουν να χρειάζονται τους σταθμούς βάσης καθώς δεν έχουν την δυνατότητα να επικοινωνήσουν απευθείας μεταξύ τους. Συνήθως σε κάθε ομάδα υπάρχουν μόνο μερικά ΜΕΑ με την δυνατότητα να συνδεθούν στους σταθμούς βάσης, και μέσω αυτών συνδέονται και τα υπόλοιπα.

Το πρόβλημα με τους σταθμούς βάσης είναι ότι είναι ακίνητες υποδομές στο έδαφος και τα ΜΕΑ μπορούν αρκετά γρήγορα να βρεθούν εκτός εμβέλειας ή να παρεμβάλλονται από ακίνητα εμπόδια. Οπότε χρειάζονται αρκετοί σταθμοί βάσης διασκορπισμένοι σε μεγάλη έκταση για να καλύψουν την περιοχή πτήσης. Και σε ΜΕΑ που πετούν σε μεγάλα υψόμετρα είναι ακόμα πιο δύσκολα τα πράγματα λόγο της απόστασης από το έδαφος.



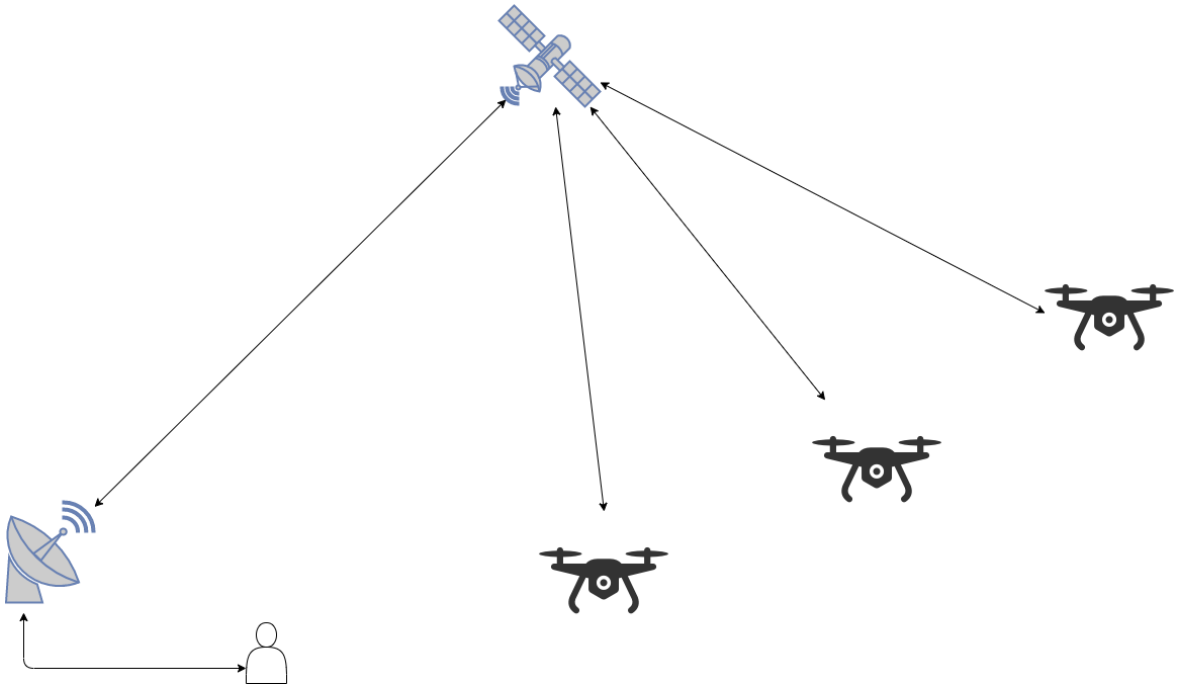
Σχήμα 2.5: Επικοινωνία ΜΕΑ - Σταθμού Βάσης

Δορυφορική Επικοινωνία

Τελευταίος τύπος επικοινωνίας και ο πιο σπάνια χρησιμοποιούμενος είναι αυτός της δορυφορικής ζεύξης. Χρησιμοποιείται από μεγάλα ΜΕΑ, συνήθως στρατιωτικών εφαρμογών, καθώς η δορυφορική σύνδεση δικαιολογεί το υψηλό της κόστος μόνο όταν δεν υπάρχει άλλη επιλογή για την επίτευξη επικοινωνίας. Εφαρμογές όπως πτήσεις πάνω από θαλάσσιες περιοχές, σε υψηλά υψόμετρα η σε περιβάλλοντα όπου το ρίσκο υποκλοπής της επικοινωνίας είναι μεγάλο είναι πιο συνηθισμένες εφαρμογές για δορυφορική σύνδεση των ΜΕΑ.

Στα θετικά της δορυφορικής επικοινωνίας είναι η σχεδόν καθολική κάλυψη της περιοχής ενδιαφέροντος όταν ο δορυφόρος βρίσκεται στην κατάλληλη τροχιακή θέση και

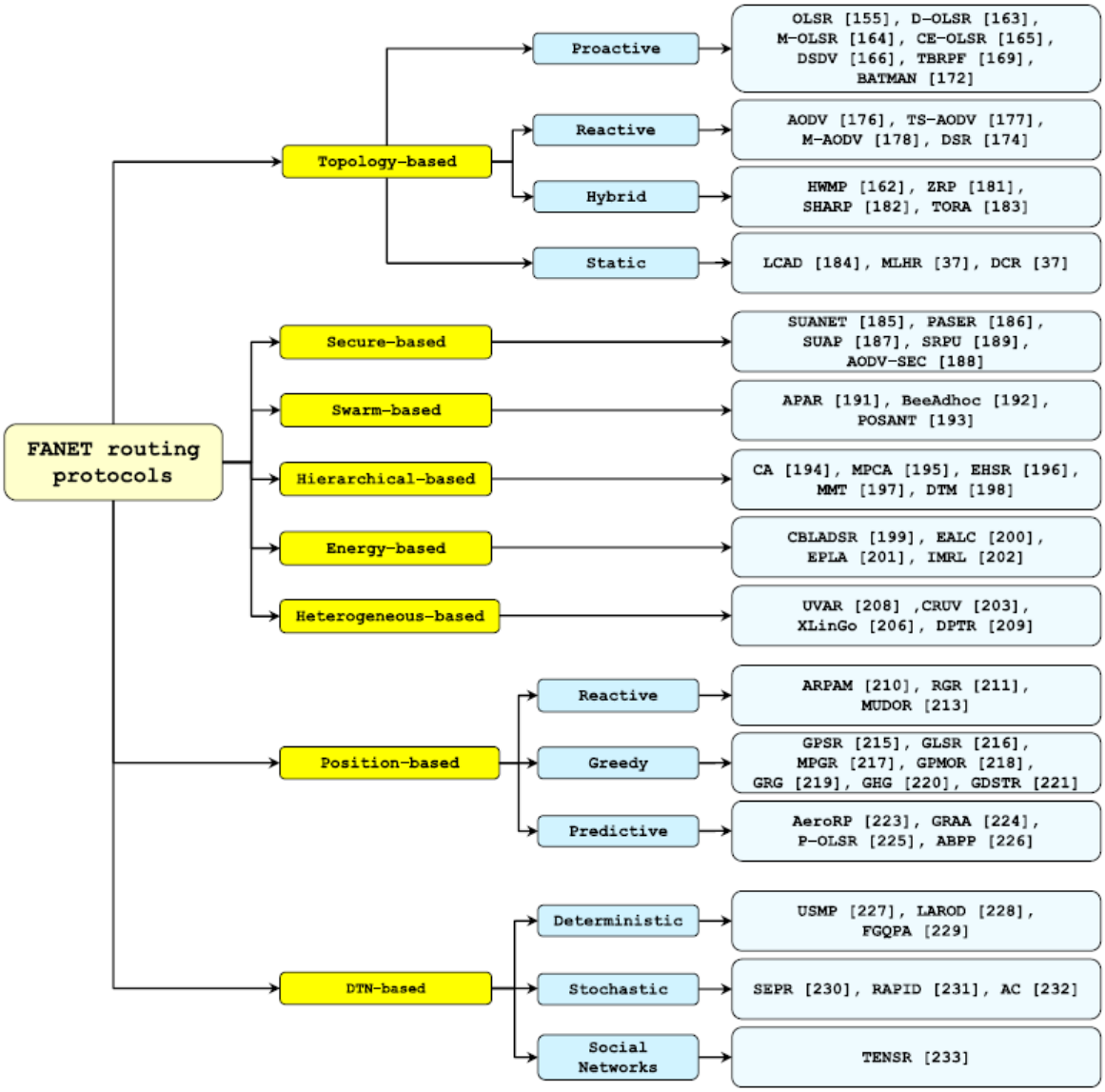
η αξιοπιστία των δορυφόρων καθώς δεν βασίζονται σε επίγειες υποδομές. Το μεγαλύτερο αρνητικό όμως των δορυφορικών ζεύξεων είναι το μεγάλο κόστος χρήσης. Η αγορά των δορυφορικών επικοινωνιών είναι αρκετά κλειστή με λίγες εταιρίες να δραστηριοποιούνται σε αυτήν λόγο του μεγάλου κόστους εισόδου σε αυτήν. Έτσι τα κόστη ενοικίασης φάσματος σε υπάρχοντες δορυφόρους είναι αρκετά υψηλά. Επιπλέον η κατασκευή και η εκτόξευση ενός δορυφόρου στην τροχιακή του θέση ανεβάζει το κόστος χρήσης κατακόρυφα καθώς πρέπει η διαχειρίστρια εταιρία να κάνει απόσβεση της επένδυσης. [36–38]



Σχήμα 2.6: Επικοινωνία ΜΕΑ Μέσω Δορυφόρου

2.5.4 Δρομολόγηση στα ΙΑΔ

Υπάρχουν πολλά πρωτόκολλα για την δρομολόγηση σε δίκτυα ΙΑΔ, όπως φαίνεται και στον ακόλουθο πίνακα, και μερικά από αυτά αναλύονται παρακάτω.



Σχήμα 2.7: Πρωτόκολλα Δρομολόγησης σε ΙΑΔ
[35]

AODV (Ad-hoc On-demand Distance Vector)

Το AODV είναι ένα αντιδραστικό πρωτόκολλο δρομολόγησης, δηλαδή αντιδράει και εξετάζει την τοπολογία του δικτύου μόνο όταν χρειάζεται να υπάρξει επικοινωνία.

Ανά τακτά χρονικά διαστήματα, όλοι οι κόμβοι εκπέμπουν μηνύματα “HELLO”. Όταν ένας κόμβος λάβει μήνυμα “HELLO” από κάποιον άλλο κόμβο, τότε υπάρχει επικοινωνία μεταξύ τους. Έτσι όλοι οι κόμβοι γνωρίζουν ποιοι είναι οι γείτονες τους στο δίκτυο. Αυτή η διαδικασία γίνεται ανά τακτά χρονικά διαστήματα για να αντικατοπτρίζει τις συνεχείς αλλαγές στην τοπολογία του δικτύου.

Όταν ένας κόμβος θέλει να μεταδώσει κάτι προς έναν άλλο κόμβο, πρέπει πρώτα να καθοριστεί η διαδρομή που θα ακολουθηθεί στο δίκτυο. Ο κόμβος εκπομπής στέλνει στους γείτονες του, που έχουν καθοριστεί με τα τακτικά μηνύματα “HELLO”, ένα μήνυμα “RREQ – Route Request” για τον επιθυμητό παραλήπτη. Κάθε κόμβος που λαμβάνει αυτό το μήνυμα, ελέγχει αν είναι ο ίδιος ο παραλήπτης ή έχει άμεση σύνδεση με τον

παραλήπτη, και αν όχι καταγράφεται η διαδρομή μέχρι εκείνο το σημείο και το μήνυμα προωθείται στους επόμενους γείτονες. Εάν όμως έχει βρεθεί ο παραλήπτης, τότε δημιουργείται το μήνυμα "RREP – Route Reply" και προωθείται στην διαδρομή που έχει καταγραφεί μέχρι να φτάσει στον αποστολέα. Εάν ο αποστολέα λάβει πίσω πολλαπλά "RREP" τότε διαλέγει την πιο σύντομη από τις πολλαπλές διαθέσιμες διαδρομές.

Κατά την διάρκεια της μετάδοσης δεδομένων, κάθε κόμβος κατά μήκος της διαδρομής ελέγχει αν οι γείτονες του μέσω των οποίων γίνεται η δρομολόγηση είναι ακόμα ενεργοί και εντός εμβέλειας. Εάν εντοπιστεί διακοπή, τότε ο κόμβος που έχασε την σύνδεση με τον γείτονα του δημιουργεί ένα μήνυμα "RERR – Route Error" και το προωθεί σε όλους τους γείτονες του με στόχο τον αποστολέα. Κάθε κόμβος που λαμβάνει το μήνυμα "RERR" ακυρώνει την σχετική διαδρομή από τον πίνακα δρομολόγησης του και προωθεί το μήνυμα παρακάτω. Όταν ο αποστολέας λάβει το μήνυμα, σταματάει την μετάδοση, ακυρώνει την διαδρομή και ξεκινάει πάλι την διαδικασία εύρεσης διαδρομής με τον επιθυμητό κόμβο. [39]

DSR (Dynamic Source Routing)

Το DSR, όπως και το AODV που αναλύθηκε στην προηγούμενη παράγραφο είναι επίσης ένα αντιδραστικό πρωτόκολλο δρομολόγησης. Ομοιάζει με το AODV καθώς οι κόμβοι εντοπίζουν τους γείτονες με τον ίδιο τρόπο. Αλλά αυτή η διαδικασία δεν γίνεται ανά τακτά χρονικά διαστήματα, αλλά μόνο όταν έχει τροποποιηθεί η τοπολογία του δικτύου. Θεωρητικά, εάν οι κόμβοι ήταν σταθεροί στον χώρο, αυτή η διαδικασία ανακάλυψης γειτόνων θα γίνει μόνο μια φορά κατά την ενεργοποίηση του δικτύου. Αυτό μειώνει δραματικά την επιβάρυνση του δικτύου από την συνεχή επικοινωνία των κόμβων μεταξύ τους για την επιβεβαίωση της επικοινωνίας τους.

Αφού έχει καθοριστεί η τοπολογία του δικτύου, αυτή γίνεται γνωστή σε όλους τους κόμβους. Οι κόμβοι εσωτερικά διατηρούν πίνακες δρομολόγησης για κάθε πιθανό συνδυασμό κόμβων, οι οποίοι είναι έγκυροι μέχρι την επόμενη φορά που θα εκτελεστεί η διαδικασία την ανακάλυψης γειτόνων.

Όταν πρόκειται να μεταδοθεί ένα πακέτο στο δίκτυο, στην κεφαλίδα του πακέτου αναγράφονται οι κόμβοι αποστολής και λήψης. Κάθε κόμβος που παραλαμβάνει το πακέτο, το προωθεί στον κατάλληλο γείτονα του με βάση τον πίνακα δρομολόγησης που διατηρεί. [40–42]

OLSR (Optimized Link State Routing)

Το OLSR, σε αντίθεση με τα δύο προηγούμενα πρωτόκολλα που αναφέρθηκαν, ανήκει στην κατηγορία των προληπτικών πρωτοκόλλων δρομολόγησης. Βασίζεται σε πίνακες δρομολόγησης τους οποίους ανταλλάζει τακτικά με τους υπόλοιπους κόμβους του δικτύου. Κάθε κόμβος, διαλέγει κάποιους από τους γείτονες τους ως multipoint relay (MPR). Αυτοί οι κόμβοι είναι και οι μόνοι υπεύθυνοι για την προώθηση της κίνηση που προορίζεται για την διάδοση στο δίκτυο.

Οι κόμβοι που έχουν επιλεγεί ως MPR είναι υπεύθυνοι για τον καθορισμό των συνδέσεων μεταξύ οποιονδήποτε κόμβων του δικτύου μέσω αυτών. Έτσι αν ένας κόμβος A θέλει να επικοινωνήσει με τον κόμβο B, η κίνηση θα δρομολογηθεί μέσω διαφόρων MPR, πρώτα τον γειτονικό MPR του A, μετά μέσω άλλων MPR και τέλος στον γειτονικό MPR του B.

Ένα αρνητικό που OLSR είναι ότι επειδή αντιμετωπίζει τις συνδέσεις σαν ενεργές ή όχι, πάντα επιλέγει την συντομότερη διαδρομή. Στα ασύρματα δίκτυα όμως, μια σύνδεση ενδέχεται και ενδιάμεσες καταστάσεις όπου υπάρχει επικοινωνία αλλά είναι χαμηλής ποιότητας. Ταυτόχρονα μια διαφορετική διαδρομή που μπορεί να μην είναι η πιο σύντομη, μπορεί να προσφέρει καλύτερη ποιότητα σύνδεσης. [43]

Κεφάλαιο 3

Προσομοίωση με τον NS3

3.1 Τι είναι ο NS3

Ο NS3 είναι ένας προσομοιωτής δικτύων διακριτών γεγονότων με ευρεία χρήση κυρίως σε εκπαιδευτικές και ερευνητικές εφαρμογές. Η άδεια χρήσης του λογισμικού είναι GPLv2 [44] και ο πηγαίος κώδικας είναι ελευθέρα διαθέσιμος για έρευνα, ανάπτυξη και χρήση. Την συντήρηση και την περαιτέρω ανάπτυξη έχει αναλάβει ομάδα εθελοντών.

Στόχος του πρότζεκτ του NS3 είναι η ανάπτυξη ενός δωρεάν, ανοιχτού κώδικα περιβάλλοντος προσομοίωσης δικτύων. Έχει ευρεία υποστήριξη για όλους του σύγχρονους τύπους ενσύρματων, ασύρματων και υβριδικών δικτύων καθώς και την δυνατότητα επέκτασης των δυνατοτήτων του μέσω βιβλιοθηκών. Ο κύριος κορμός του προγράμματος και όλες οι βιβλιοθήκες είναι γραμμένες σε C++ ενώ οι οπτικοποιήσεις των προσομοιώσεων καθώς και το σχετικό πρόγραμμα γραφικής διεπαφής, ονόματι NetAnim, σε Python. [45]

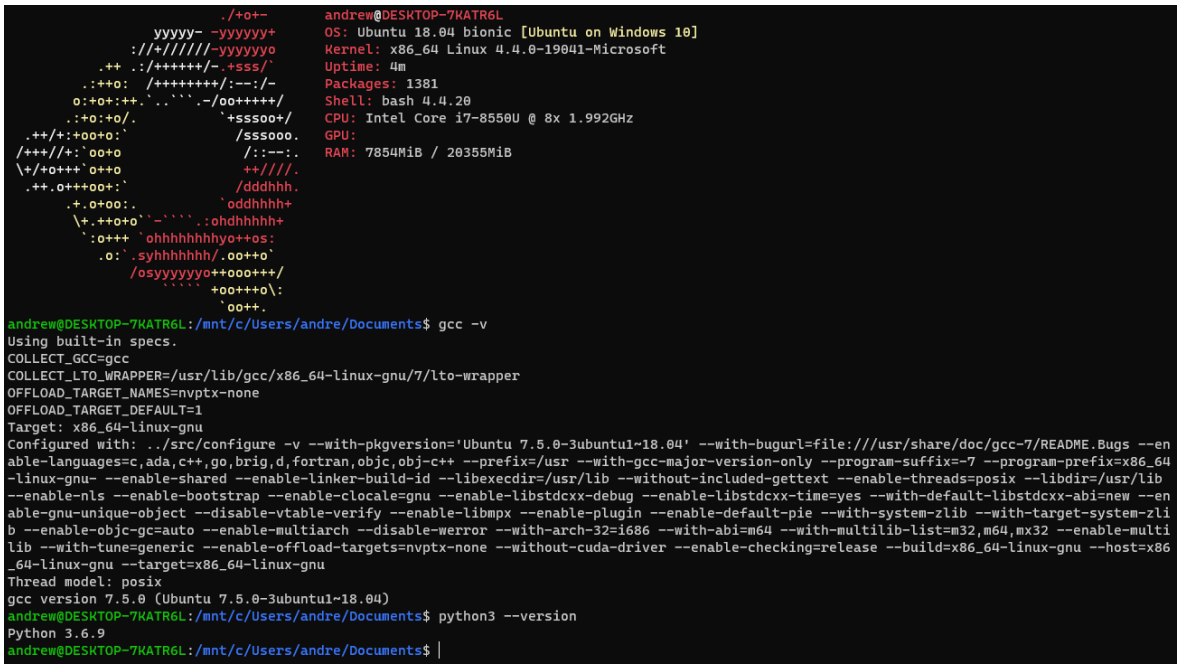
3.2 Εγκατάσταση του NS3

Ο NS3 είναι συμβατός με Linux και macOS, όπου υπάρχει GCC compiler έκδοσης 4.9 και νεότερης (ή άλλος εξίσου συμβατός compiler για C++) καθώς και Python 3 έκδοσης 3.5 και νεότερης.

Κάποιες από τι εκδόσεις του NS3 έχουν τροποποιηθεί κατάλληλα έτσι ώστε με την χρήση του Cygwin να μπορούν να εγκατασταθούν σε περιβάλλον Windows, αλλά αυτό δεν είναι επίσημα υποστηριζόμενη δυνατότητα και προτείνεται να χρησιμοποιηθεί περιβάλλον Linux ή macOS.

3.2.1 Προετοιμασία και Προαπαιτούμενα

Για την εγκατάσταση του NS3 για την ανάγκες της παρούσας εργασίας χρησιμοποιήθηκε υπολογιστής με Windows 10 2004 όπου έτρεχε σε περιβάλλον WSL το λειτουργικό σύστημα Ubuntu 18.04 LTS. Εντός του Ubuntu χρησιμοποιήθηκε GCC 7.5.0 και Python 3.6.9. Και εγκαταστάθηκε η έκδοση 3.31 του NS3.



Σχήμα 3.1: Εκδόσεις Λογισμικού

Το NS3, όντως ένα αρκετά περίπλοκο πρόγραμμα, βασίζεται στην ύπαρξη πολλών άλλων προγραμμάτων για να εκτελέσει όλες τις λειτουργίες του. Αυτά τα προγράμματα, υπό την μορφή πακέτων, δίνονται αναλυτικά στις οδηγίες εγκατάστασης στην παράγραφο των προαπαιτούμενων. Οπότε είτε μπορούμε να τα εγκαταστήσουμε χειροκίνητα, είτε να χρησιμοποιήσουμε το παρεχόμενο script. [Β′.1]

Για να εκτελέσουμε script, ανοίγουμε τερματικό και το εκτελούμε με δικαιώματα διαχειριστή ως εξής:

sudo ./ns3_dependencies.sh

Αφού τελειώσει επιτυχώς η εγκατάσταση των προαπαιτούμενων πακέτων (μπορεί να διαρκέσει αρκετά λεπτά), προχωράμε στην εγκατάσταση του NS3.

3.2.2 Λήψη και Εγκατάσταση NS3

Ο πιο εύκολος τρόπος είναι να χρησιμοποιήσουμε το εργαλείο Bake που έχει φτιάξει η ίδια ομάδα που έφτιαξε και το NS3. Αλλιώς μπορούμε και να κατεβάσουμε το συμπιεσμένο repository από την ιστοσελίδα του NS3 και να τα κάνουμε όλα χειροκίνητα.

Το Bake είναι ένα εργαλείο γραμμένο σε Python που αυτοματοποιεί αυτή την διαδικασία της εγκατάστασης και ρύθμισης του NS3.

Κατεβάζουμε από το repository το εργαλείο Bake ως εξής:

git clone https://gitlab.com/nsnam/bake

Για δική μας ευκολία, μπορούμε να εισάγουμε το εκτελέσιμο του Bake στις μεταβλητές περιβάλλοντος για να έχουμε πρόσβαση σε αυτό από παντού ως εξής:

export BAKE_HOME=‘pwd‘/bake

export PATH=$PATH:$BAKE_HOME

export PYTHONPATH=$PYTHONPATH:$BAKE_HOME

Έπειτα ελέγχουμε να δούμε αν όλα τα απαιτούμενα πακέτα (από το προηγούμενο βήμα) έχουν εγκατασταθεί και είναι προσβάσιμα από το σύστημα μας. Αυτό γίνεται με την εντολή:

bake.py check

Εάν όλα εμφανίζονται με την ένδειξη "OK", τότε προχωράμε στο επόμενο βήμα, την λήψη και μεταγλώττιση του NS3. Η λήψη του γίνεται ως εξής:

bake.py configure -e ns-3.31

Για να δούμε ποια μοδυλε θα εγκατασταθούν καθώς και τις λεπτομέρειες του τρέχοντος configuration πριν γίνει το τελικό compile μπορούμε να το κάνουμε με:

bake.py show

Τέλος, το τελευταίο και πιο χρονοβόρο βήμα είναι η μεταγλώττιση του NS3 για την παραγωγή του εκτελέσιμου προγράμματος στο μηχάνημα μας. Αυτό γίνεται ως εξής:

bake.py deploy

Όταν τελειώσει η παραπάνω διαδικασία, και αν δεν υπάρχουν σφάλματα στην πορεία, θα δούμε να έχει παραχθεί το φάκελος source δίπλα στο φάκελο bake. Εκεί μέσα και συγκεκριμένα στην διαδρομή /source/ns-3.31 βρίσκεται η εγκατάσταση του NS3.

Κατευθυνόμαστε στον κεντρικό φάκελο του NS3 (/source/ns-3.31). Δίνουμε την εξής εντολή:

./waf

Για να μεταγλωττιστούν τα επιμέρους κομμάτια του NS3. Αυτή η διαδικασία κρατάει αρκετή ώρα μόνο την $1^{η}$ φορά που έχει να κάνει compile όλες τις βιβλιοθήκες.

Έπειτα για να δούμε ότι όλα πήγαν καλά, μπορούμε να τρέξουμε το ενσωματωμένο δοκιμαστικό πρόγραμμα του NS3 που δοκιμάζει όλες τις λειτουργίες του. Μπορούμε να το εκτελέσουμε με:

./test.py

Στο τέλος τις εκτέλεσης του test θα μας αναφέρει αναλυτικά αν έχει αποτύχει κάποιο επιμέρους module ποιο είναι αυτό. [46]

3.3 Χρήση του NS3

Όλα τα σενάρια προσομοίωσης στον NS3 γράφονται σε γλώσσα C++. Τα πηγαία αρχεία του κώδικα τοποθετούνται πάντα στο φάκελο ns3/scratch. Έστω ότι έχουμε το πηγαίο αρχείο ονόματι test.cc. Για να γίνει compile και εκτέλεση του αρχείου, όσο είμαστε στη κεντρικό φάκελο του NS3, χρησιμοποιούμε το εργαλείο waf ως εξής:

./waf -run "scratch/test"

Έτσι γίνεται compile και εκτέλεση του σεναρίου. Εάν θέλουμε να προβάλλουμε και την οπτικοποιήση της προσομοίωσης, μπορούμε να την καλέσουμε με την παράμετρο –vis ως εξής:

./waf –run "scratch/test" –vis

Εάν αντιμετωπίσουμε προβλήματα κατά την εκτέλεση της προσομοίωσης, μπορούμε να κάνουμε αποσφαλμάτωση όπως και σε κάθε άλλο πρόγραμμα C++ με τον GDB καλώντας την παράμετρο –gdb ως εξής:

./waf –run "scratch/test" –gdb

Όσον αφορά τα παραγόμενα αρχεία με τα δεδομένα της προσομοίωσης, αυτά και οι παραμέτροί τους ρυθμίζονται εντός του πηγαίου κώδικα του σεναρίου μας και είναι σε μορφή XML.

Κεφάλαιο 4

Χρήση του NS3 για Προσομοίωση Δικτύων σε Χώρο 2 Διαστάσεων

Ο NS3 είναι, όπως αναφέρθηκε παραπάνω, διαθέτει πολλές επιμέρους βιβλιοθήκες που του προσθέτουν διαφορετικές λειτουργίες και δυνατότητες. Μεταξύ της πληθώρας δυνατοτήτων που έχει στην βασική του έκδοση, είναι και αυτή της προσομοίωσης ασύρματων δικτύων ΦΑΔ/ΑΔΟ/ΙΑΔ σε χώρο 2 διαστάσεων, καθώς και υποστήριξη για τα πρωτόκολλα δρομολόγησης AODV, DSR και OLSR που επίσης αναλύσαμε παραπάνω.

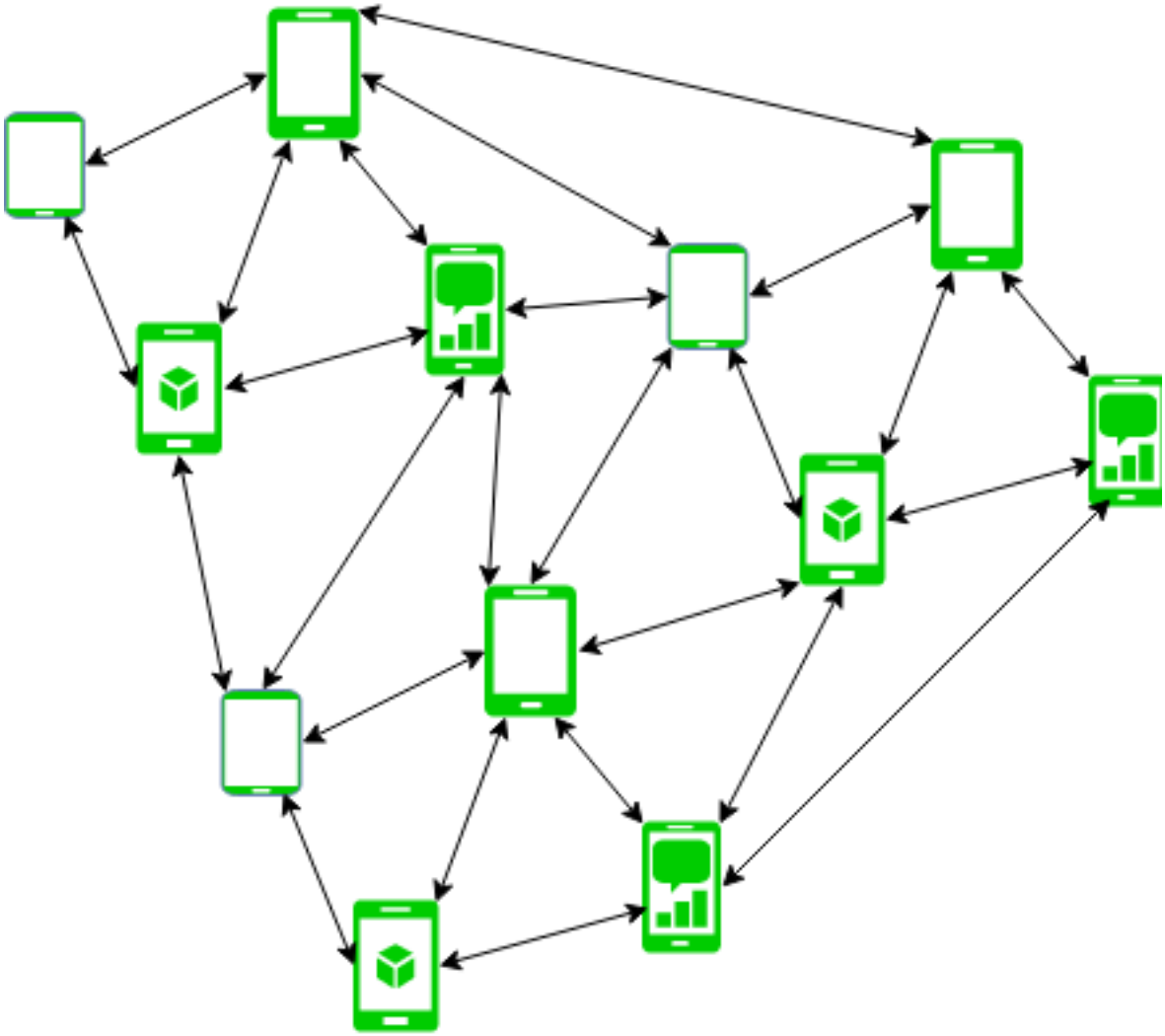
Πριν όμως προχωρήσουμε στις λεπτομέρειες της προσομοίωσης, θα γίνει μια αναφορά στα δίκτυα ΙΑΔ και ΑΔΟ.

4.1 Φορητά Αυτοοργανωμένα Δίκτυα (ΦΑΔ)

Τα δίκτυα ΦΑΔ (Φορητά Αυτοοργανωμένα Δίκτυα) είναι υποκατηγορία των αυτοοργανωμένων δικτύων και αναφέρονται σε δίκτυα με κινούμενους κόμβους και τοπολογία που συνεχώς μεταβάλλεται. [47]

Τα δίκτυα ΦΑΔ αποτελούνται από κινούμενους ασύρματους κόμβους χωρίς συγκεκριμένη τοπολογία δικτύου οι οποίοι είτε μπορούν να λειτουργούν αυτόνομα είτε να είναι μέρος ενός μεγαλύτερου δικτύου. Ο κάθε κόμβος εκτελεί και χρέη δρομολογητή για την δρομολόγηση της κίνησης στους γειτονικούς του κόμβους με τους οποίους έχει επικοινωνία.

Η μεγαλύτερη πρόκληση των ΦΑΔ είναι η δυνατότητα του κάθε κόμβου να αντιλαμβάνεται τις αλλαγές στην τοπολογία τους δικτύου γύρω του και να δρομολογεί σωστά την κίνηση σε ένα συνεχώς μεταβαλλόμενο περιβάλλον.

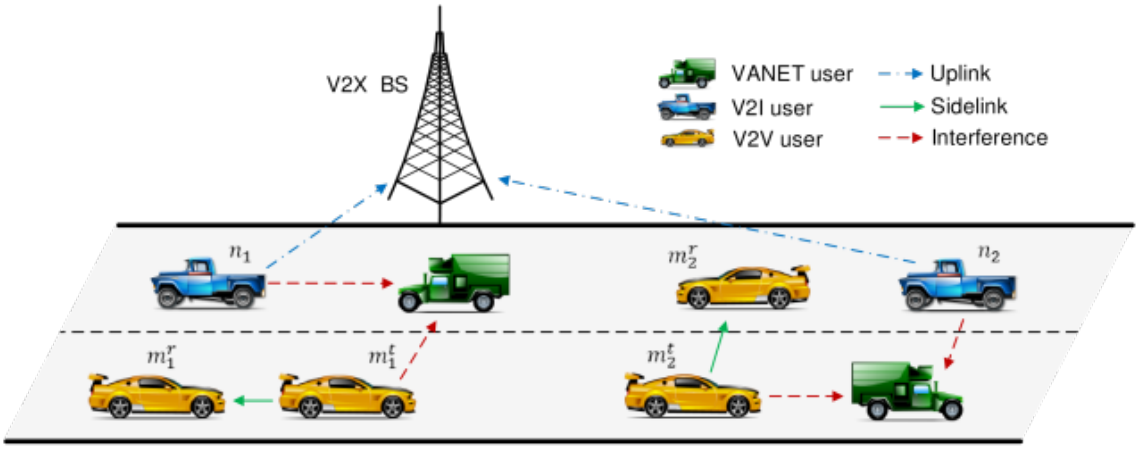


Σχήμα 4.1: Τυπικό Δίκτυο ΦΑΔ

4.2 Αυτοοργανωμένα Δίκτυα Οχημάτων (ΑΔΟ)

Τα δίκτυα ΑΔΟ (Αυτοοργανωμένα Δίκτυα Οχημάτων), ακολουθούν τις βασικές αρχές που διέπουν τα δίκτυα ΦΑΔ αλλά οι κόμβοι του δικτύου είναι αποκλειστικά κινούμεα οχήματα. Είναι συνώνυμα και με τον πιο γενικό όρο ΔΟΕ (Διαοχηματική Επικοινωνία) και τα τελευταία χρόνια με την εξέλιξη των αυτόνομων οχημάτων έχουν τραβήξει την προσοχή της ερευνητικής κοινότητας. [48]

Οι κύριες προκλήσεις των ΑΔΟ είναι η γρήγορη εναλλαγή των κόμβων από τον έναν σταθμό βάσης στον επόμενο, καθώς κινούνται αρκετά γρήγορα, και η ελαχιστοποίηση της καθυστέρησης του δικτύου. Ιδίως σε εφαρμογές αυτόνομων οχημάτων, η ταχύτητα απόκρισης χρειάζεται να είναι όσο το δυνατόν πιο άμεση.



Σχήμα 4.2: Τυπικό Δίκτυο ΑΔΟ

4.3 Προσομοίωση Δικτύων ΑΔΟ με τον NS3

Στα πλαίσια της διπλωματικής εργασίας, και για την αναγνώριση των δυνατοτήτων του NS3, δημιουργήθηκε ένα σενάριο προσομοίωσης με βάση τις αρχές των ΑΔΟ. [Β΄.2]

Βάση για την δημιουργία του σεναρίου, αποτέλεσε ο κώδικας ενός εκ των παραδειγμάτων που περιλαμβάνονται στον NS3, και πιο συγκεκριμένα του παραδείγματος της σύγκρισης διαφόρων πρωτοκόλλων δρομολόγησης για δίκτυα ΑΔΟ. [49], καθώς και η σχετική εργασία του Dr. Pradeep Kumar ονόματι “MANET Routing Protocols Using NS3”. [50]

Συνδυάζοντας τις παραπάνω γνώσεις, δημιουργήσαμε ένα ρεαλιστικό σενάριο δικτύου ΑΔΟ σε χώρο 2 διαστάσεων, το οποίο εκτελέστηκε για διαφορετικούς αριθμούς κόμβων και για διαφορετικά πρωτόκολλα δρομολόγησης προκειμένου να μελετηθεί η διαφορετική συμπεριφορά των πρωτοκόλλων δρομολόγησης με μεταβλητό αριθμό κόμβων στην ίδια περιοχή και με σταθερές τις υπόλοιπες παραμέτρους. Οι παράμετροι της προσομοίωσης ήταν οι εξής:

Πρωτόκολλο Δρομολόγησης	AODV / DSR / OLSR
Μοντέλο Κίνησης	Random Waypoint
Χώρος Προσομοίωσης	2000x2000 m
Αριθμός Κόμβων	10 / 15 / 20 / 25
Χρόνος Προσομοίωσης	60 sec
Μέγεθος Πακέτου UDP	1000 byte
Wireless Standard	802.11g
Μοντέλο Απωλειών	Friis
Ταχύτητα Κόμβων	10 m/s
Ακινησία Κόμβων	1 sec
Εύρος Ζώνης	1 Mbps
Ισχύς Εκπομπής	27 dBm (500 mW)

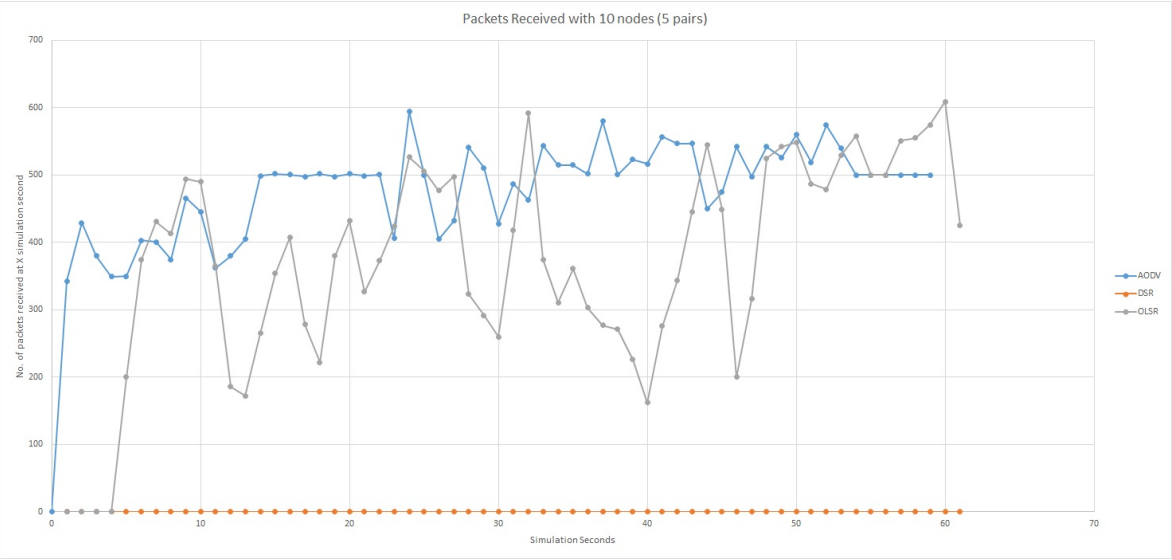
Πίνακας 4.1: Παράμετροι Προσομοίωσης Δικτύου ΑΔΟ

Οι παραπάνω παράμετροι δεν καθορίστηκαν στην τύχη, αλλά ήταν προϊόν σκέψης και καθορισμού ρεαλιστικών συνθηκών για της ανάγκες τις προσομοίωσης. Ο χώρος καθορίσαμε ότι είναι μια περιοχή 2 km^2 που θα μπορούσε να αναπαριστά οποιαδήποτε περιοχή με κίνηση οχημάτων. Ο αριθμός των κόμβων εντός της περιοχής ήταν κυμαινόμενος από 10 έως 25. Ο χρόνος της προσομοίωσης ορίστηκε στο 1 λεπτό. Η ταχύτητα κίνησης των κόμβων ορίστηκε συντηρητικά στα 10 m/s (36 km/h) [51]. Ο χρόνος ακινησίας, δηλαδή για πόση ώρα μένει ακίνητος ο κόμβος πριν αλλάξει κατεύθυνση ορίστηκε στο 1 sec. Το μοντέλο κίνησης που επιλέχθηκε είναι το πλήρως τυχαίο. Το εύρος ζώνης της κάθε σύνδεσης ορίστηκε το 1 Mbps ενώ η ισχύ εκπομπής του κάθε κόμβου ορίστηκε στα 27 dBm (500 mW) [52], όσο περίπου είναι η μέγιστη ισχύ εκπομπής ενός κινητού τηλεφώνου.

Επιπλέον, ορίσαμε στο σενάριο μας να καταγράφονται μετρικά περί της κίνησης των δεδομένων και τις κίνησης των κόμβων, και να εξάγονται σε αρχεία.

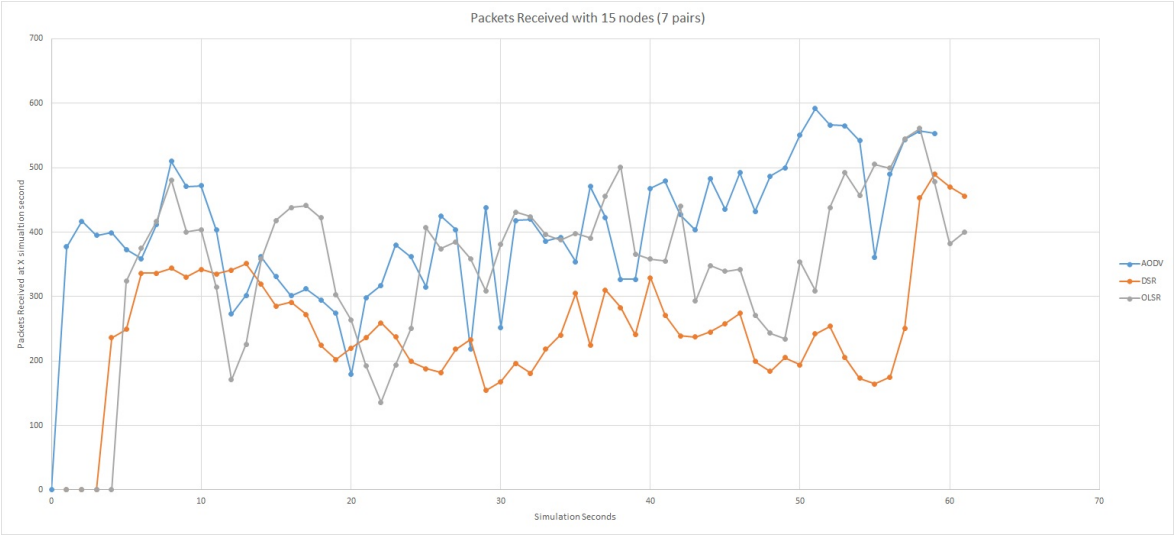
Με το πέρας της προσομοίωσης, παράγονται 4 αρχεία που έχουν καταγράψει διάφορες πληροφορίες. Ένα αρχείο με κατάληξη “csv” όπου για κάθε ένα δευτερόλεπτο του χρόνου προσομοίωσης καταγράφεται ο συνολικός αριθμός πακέτων που έχει περάσει επιτυχώς από το δίκτυο. Ένα αρχείο με κατάληξη “flowmon” όπου καταγράφονται οι λεπτομέρειες των επιμέρους συνδέσεων όπως ο χρόνος αποστολής, ο χρόνος άφιξης, η καθυστέρηση, jitter, ο αριθμός προωθήσεων κ.α. Ένα αρχείο με κατάληξη “mob” όπου καταγράφεται για κάθε χρονική στιγμή η θέση και η ταχύτητα του κάθε κόμβου. Και τέλος, ένα αρχείο με κατάληξη “xml” όπου επίσης περιλαμβάνει πληροφορίες κίνησης και μπορεί να χρησιμοποιηθεί από το πρόγραμμα NetAnim για την οπτικοποίηση της προσομοίωσης.

Παρακάτω ακολουθούν μερικά γραφήματα που δείχνουν τα επιτυχώς μεταδιδόμενα πακέτα σε κάθε δευτερόλεπτο της προσομοίωσης για κάθε ένα από τα 3 πρωτόκολλα δρομολόγησης (AODV, DSR, OLSR) για αυξανόμενο αριθμό κόμβων (10, 15, 20, 25) και με όλες τις υπόλοιπες παραμέτρους της προσομοίωσης να παραμένουν σταθερές.

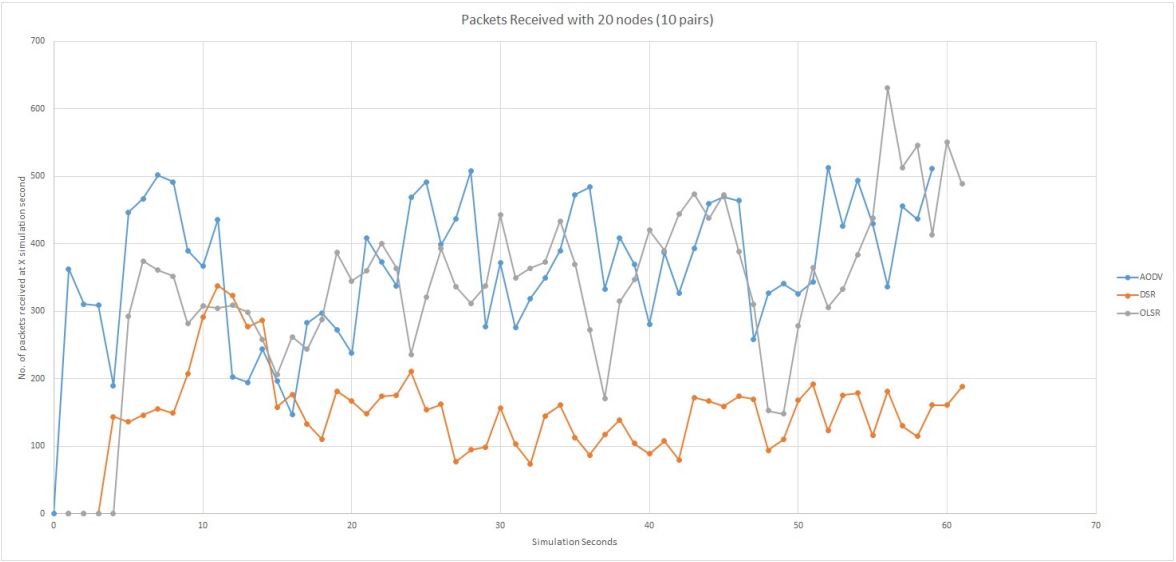


Σχήμα 4.3: ΑΔΟ με 10 Κόμβους

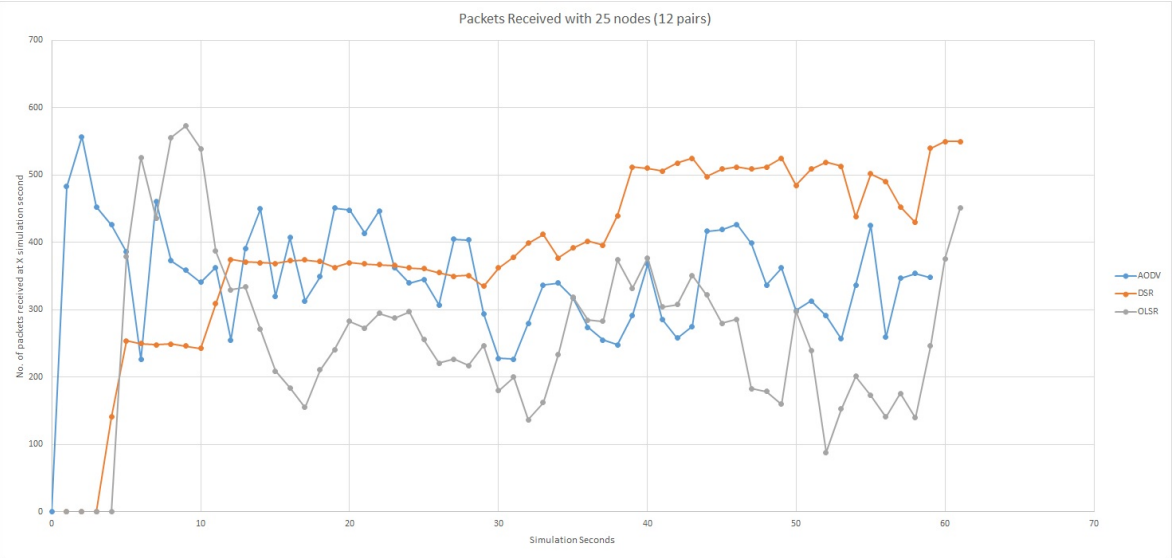
Για 10 κόμβους βλέπουμε ότι το AODV καταφέρνει να μεταδώσει επιτυχημένα τα περισσότερα πακέτα. Λόγο του ότι εκτιμάει την τοπολογία του δικτύου κάθε φορά που είναι να μεταδώσει, καταφέρνει να κρατήσει τις συνδέσεις σε καλύτερη κατάσταση και συνεπώς να μεταδώσει πιο πολλά πακέτα. Σε αντίθεση με το OLSR που αντιμετωπίζει τις συνδέσεις μόνο σαν ενεργές ή όχι και τις ελέγχει ανά τακτά χρονικά διαστήματα και δρομολογεί τα πακέτα πάντα από την πιο σύντομη διαδρομή, ακόμη και αν δεν είναι η βέλτιστη. Επειδή έχουμε να κάνουμε με ασύρματες συνδέσεις, η πιο σύντομη από πλευράς απόστασης διαδρομή, υπάρχει περίπτωση να μην είναι και η βέλτιστη. Μια άλλη διαδρομή μεγαλύτερης απόστασης αλλά με περισσότερους ενδιάμεσους κόμβους και καλύτερα χαρακτηριστικά σύνδεσης είναι δυνατόν να προσφέρει καλύτερα αποτελέσματα. Μεταξύ των διαστημάτων ελέγχου της τοπολογίας, επειδή η θέση των κόμβων δεδομένων μεταβάλλεται συνέχεια, το OLSR κάνει το λάθος να επιλέγει την διαδρομή που ¨θυμάται¨ ως την βέλτιστη και όχι αυτή που είναι πραγματικά βέλτιστη τη δεδομένη χρονική στιγμή. Όταν ανανεώνει τους εσωτερικούς πίνακες δρομολόγησης, βλέπουμε ότι η απόδοση του φτάνει αυτή του AODV, αλλά μετά από λίγη ώρα η απόδοση ακολουθεί πτωτική τάση.



Σχήμα 4.4: ΑΔΟ με 15 Κόμβους



Σχήμα 4.5: ΑΔΟ με 20 Κόμβους



Σχήμα 4.6: ΑΔΟ με 25 Κόμβους

Όσον αφορά τις 3 επόμενες περιπτώσεις, για τους AODV και OLSR βλέπουμε το ίδιο μοτίβο. Ο AODV είναι ελαφρώς καλύτερος και πιο σταθερός από τον OLSR. Ο OLSR όταν ενημερώνει τους πίνακες δρομολόγησης του, φτάνει στα νούμερα του AODV αλλά μετά από λίγη ώρα ακολουθεί πτωτική τάση. Η μεγάλη αλλαγή όμως στην προσομοίωση με τους 20 κόμβους, είναι η μεγάλη βελτίωση του DSR, ιδίως μετά την πάροδο του μισού χρόνου της προσομοίωσης. Η εξήγηση είναι ότι ο DSR ενημερώνει τους πίνακες δρομολόγησης κάθε φορά που γίνεται αλλαγή στην τοπολογία του δικτύου. Όταν έχουμε 20 κόμβους, έχουμε και περισσότερες συνδέσεις μεταξύ των κόμβων. Και αφού όλοι μαζί κινούνται, οι αλλαγές στην τοπολογία του δικτύου είναι πολύ περισσότερες. Συνεπώς ο DSR ενημερώνει τους πίνακες δρομολόγησης και επιλέγει την βέλτιστη διαδρομή πιο συχνά από τα άλλα 2 πρωτόκολλα δρομολόγησης όσο αυξάνεται ο αριθμός των κόμβων.

Κεφάλαιο 5

Χρήση του NS3 για Προσομοίωση Δικτύων σε Χώρο 3 Διαστάσεων

Με βάση το προηγούμενο σενάριο σε χώρο 2 διαστάσεων, το επόμενο βήμα ήταν η εκτέλεση αντίστοιχου σεναρίου σε χώρο 3 διαστάσεων για να ομοιάζει με δίκτυο ΙΑΔ που οι κόμβοι του είναι ιπτάμενα αεροσκάφη. Έτσι ξεκίνησε η έρευνα για τον τρόπο με τον οποίο θα μπορούσε ο NS3 να υποστηρίξει χώρο 3 διαστάσεων.

5.1 Προσομοίωση Δικτύου ΙΑΔ με τον NS3

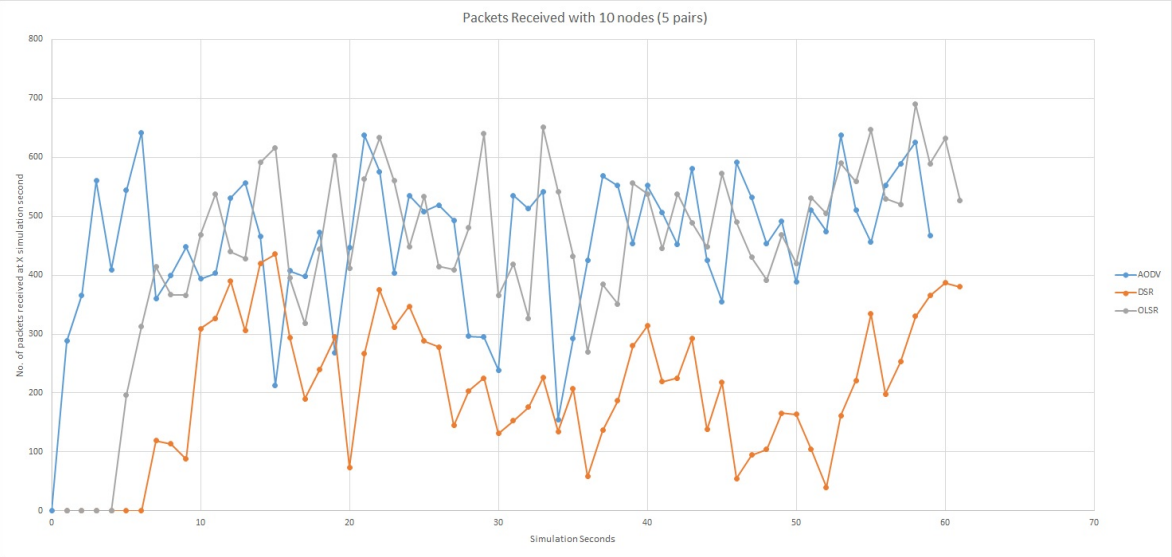
Σε αυτό το σημείο, βασική πηγή αποτέλεσε η εργασία των Sunil Kr. Maakar et al [53] οι οποίοι επιτυχώς έχουν προσομοιώσει δίκτυο ΙΑΔ στον NS3. Με βάση το προηγούμενο λοιπόν σενάριο [Β΄.2], έγιναν οι κατάλληλες τροποποιήσεις έτσι ώστε να έχουμε σενάριο ΙΑΔ [Β΄.3]. Συγκεκριμένα οι δύο αλλαγές που έγιναν είναι ότι άλλαξε εντός του κώδικα του σεναρίου ο τύπος του αντικειμένου ObjectFactory από "ns3::RandomRectanglePositionAllocator" σε "ns3::RandomBoxPositionAllocator" καθώς και το μοντέλο κίνησης άλλαξε από “Random Waypoint” σε “Gauss Markov”.

Η πρώτη αλλαγή μας επέτρεψε να έχουμε χώρο 3 αντί για 2 διαστάσεων, ενώ με την δεύτερη αλλαγή αλλάξαμε το μοντέλο κίνησης από εντελώς τυχαίο στο μοντέλο Gauss Markov, που χρησιμοποιείται πολύ συχνά στη βιβλιογραφία για τα δίκτυα ΙΑΔ. Οι υπόλοιποι παράμετροι της προσομοίωσης παραμένουν σταθερές όπως και στο προηγούμενο σενάριο.

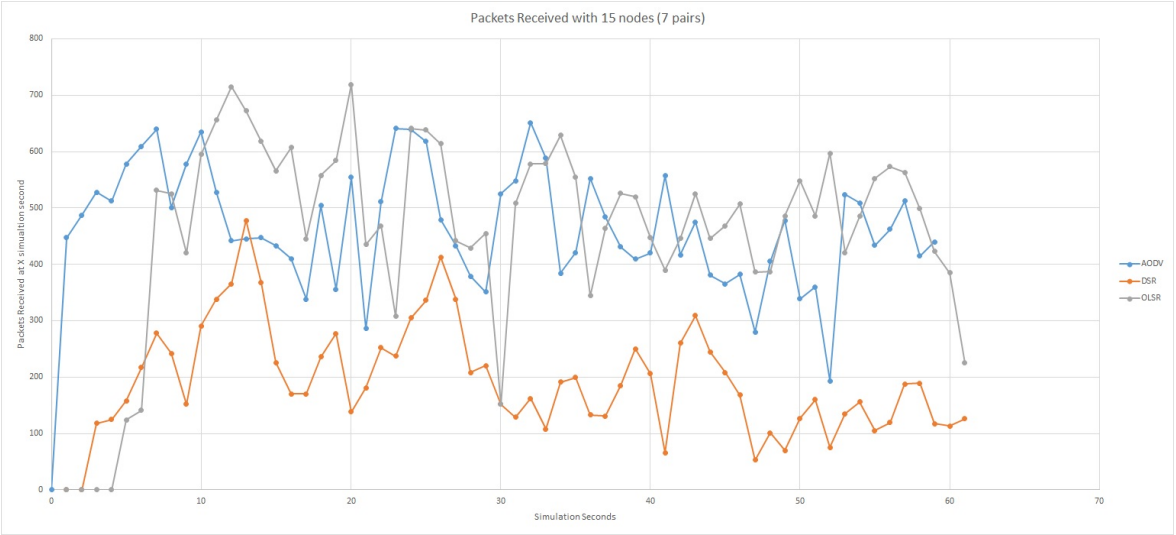
Πρωτόκολλο Δρομολόγησης	AODV / DSR / OLSR
Μοντέλο Κίνησης	Gauss Markov
Χώρος Προσομοίωσης	2000x2000x150 m
Αριθμός Κόμβων	10 / 15 / 20 / 25
Χρόνος Προσομοίωσης	60 sec
Μέγεθος Πακέτου UDP	1000 byte
Wireless Standard	802.11g
Μοντέλο Απωλειών	Friis
Ταχύτητα Κόμβων	10 m/s
Ακινησία Κόμβων	1 sec
Εύρος Ζώνης	1 Mbps
Ισχύς Εκπομπής	27 dBm (500 mW)

Πίνακας 5.1: Παράμετροι Προσομοίωσης Δικτύου ΙΑΔ

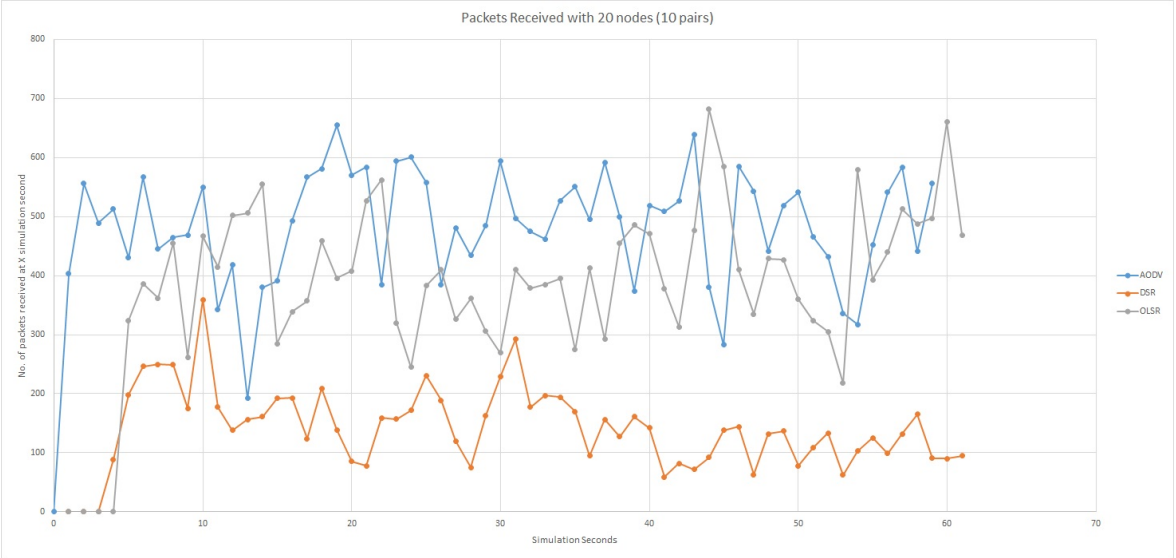
Παρακάτω ακολουθούν μερικά γραφήματα που δείχνουν τα επιτυχώς μεταδιδόμενα πακέτα σε κάθε δευτερόλεπτο της προσομοίωσης για κάθε ένα από τα 3 πρωτόκολλα δρομολόγησης (AODV, DSR, OLSR) για αυξανόμενο αριθμό κόμβων (10, 15, 20, 25) με όλες τις υπόλοιπες παραμέτρους της προσομοίωσης σταθερές, όπως και στο σενάριο των ΑΔΟ.



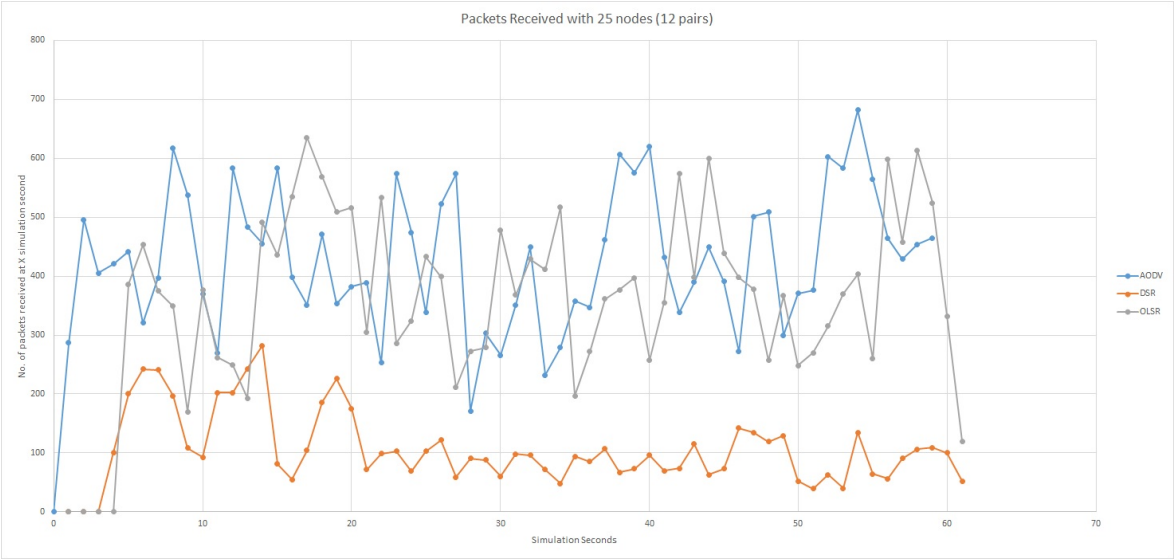
Σχήμα 5.1: ΙΑΔ με 10 Κόμβους



Σχήμα 5.2: ΙΑΔ με 15 Κόμβους



Σχήμα 5.3: ΙΑΔ με 20 Κόμβους



Σχήμα 5.4: ΙΑΔ με 25 Κόμβους

Τα συμπεράσματα, όπως και στο δίκτυο 2 διαστάσεων είναι παρόμοια. Σε γενικές γραμμές υπερτερεί ο AODV, με τον OLSR να βρίσκεται πολύ κοντά του, και σε αρκετές περιπτώσεις να τον ξεπερνά κιόλας. Όμως ο DSR, σε αντίθεση με το σενάριο ΑΔΟ, ούτε στους 25 κόμβους κατάφερε να ξεχωρίσει, όντας και στις 4 περιπτώσεις σημαντικά κατώτερος. Μόνη εξαίρεση στην ανωτερότητα του AODV αποτελεί η περίπτωση των 15 κόμβων, όπου ο OLSR είναι στο μεγαλύτερο μέρος του χρόνου λίγο πιο αποδοτικός από τον AODV στην αποδοτική μετάδοση πακέτων.

5.2 Συμπεράσματα και Προκλήσεις

Το κυριότερο συμπέρασμα είναι ότι εν τέλη, υπάρχει η δυνατότητα της προσομοίωσης δικτύων σε χώρο 3 διαστάσεων με τον NS3. Επιπλέον, στα πλαίσια αυτής της εργασίας καταφέραμε να εκτελέσουμε προσομοιώσεις για τα γνωστότερα και αποδοτικότερα πρωτόκολλα δρομολόγησης τόσο σε δίκτυα ΑΔΟ όσο και σε δίκτυα ΙΑΔ. Τέλος επιχειρήσαμε να ερμηνεύσουμε και να αξιολογήσουμε τα αποτελέσματα των προσομοιώσεων.

Όσον αφορά τις προκλήσεις, υπήρξαν πολλές κατά την διάρκεια εκπόνησης αυτής της εργασίας. Ξεκινώντας, η εγκατάσταση του NS3 ήταν αρκετά δύσκολη, κυρίως λόγο περιορισμένης τεκμηρίωσης. Οι επίσημες οδηγίες εγκατάστασης αναφέρονται σε πιο παλιές εκδόσεις του NS3 και του λειτουργικού συστήματος στο οποίο γίνεται η εγκατάσταση. Για να αντιμετωπιστεί αυτό χρειάστηκε εκτενής έρευνα και αναζήτηση λύσεων σε φόρουμ με χρήστες του NS3 και πειραματισμούς μέχρι να βρεθεί μια λύση που στην περίπτωση μας λειτούργησε. Γι' αυτό δίνεται και το βοηθητικό πρόγραμμα με τα προαπαιτούμενα της εγκατάστασης και αναλυτικές οδηγίες πως να εγκατασταθεί σωστή η συγκεκριμένη έκδοση του NS3 (3.31) στο συγκεκριμένο λειτουργικό σύστημα (Ubuntu 18.04 x64 LTS).

Στην υλοποίηση των σεναρίων, έγινε χρήση της επίσημης τεκμηρίωσης του NS3 για την κατανόηση της λειτουργίας του προσομοιωτή και την αξιοποίηση όλων των δυνατοτήτων του. Παρόλα αυτά, η επίσημη τεκμηρίωση έχει τα εξής δύο προβλήματα. Πρώτον στην πλειοψηφία των θεμάτων υπάρχουν μόνο επιγραμματικές αναφορές των διαφόρων

λειτουργιών χωρίς επεξήγηση και αμέσως μετά ακολουθεί παραπομπή στο αντίστοιχο σημείο στον πηγαίο κώδικα του NS3. Δεύτερον, μαζί με τον NS3 δίνονται και κάποια έτοιμα σενάρια ως παραδείγματα που μπορεί να εκτελέσει ο χρήστης. Το πρόβλημα όμως τόσο με τα δοθέντα παραδείγματα όσο και με τον πηγαίο κώδικα του NS3 είναι ότι δεν συνοδεύονται από σχόλια. Έτσι είναι αρκετά δύσκολο να γίνει κατανοητή η ροή του προγράμματος και πως υλοποιείται η κάθε λειτουργία. Να σημειωθεί επίσης ότι οι πιο πολλές σελίδες στην επίσημη τεκμηρίωση αφορούσαν παλαιότερες εκδόσεις του NS3 και δεν έχουν ενημερωθεί.

Τέλος, στην δική μας περίπτωση η προσομοίωση έχει καταγράψει πολλά περισσότερα δεδομένα από αυτά που παρουσιάστηκαν στα δύο προηγούμενα κεφάλαια. Τα δεδομένα αυτά είναι μέσα στα αρχεία xml, flowmon και mob που αναφέρθηκαν προηγουμένως, χωρίς ωστόσο να υπάρχει εύκολος τρόπος να εξαχθούν, να οπτικοποιηθούν και να αναλυθούν. Θα διευκόλυνε σημαντικά την έρευνητική κοινότητα η ανάπτυξη ενός προγράμματος που θα εξάγει τα δεδομένα σε csv ή excel και να επιτρέψει με αυτό τον τρόπο την επεξεργασία και την οπτικοποίησή τους. Προς το παρόν, αυτή τη διαδικασία την αναλαμβάνει αυτός που θα ασχοληθεί με τον NS3, καθώς όσοι ισχυρίζονται ότι έχουν αναπτύξει τέτοιου είδους εργαλεία δεν τα δημοσιεύουν.

Παράρτημα Α΄

Ακρωνύμια και συντομογραφίες

ΜΕΑ Μη Επανδρωμένο Αεροσκάφος

ΙΑΔ Ιπτάμενο Αυτοοργανωμένο Δίκτυο

ΑΔΟ Αυτοοργανωμένο Δίκτυο Οχημάτων

ΣΓΕ Σύστημα Γεωγραφικού Εντοπισμού

ΒΕ Βαθμός Ελευθερίας

ΜΑΜ Μονάδα Αδρανειακής Μέτρησης

ΡΣΔ Ραντάρ Συνθετικού Διαφράγματος

ΦΑΔ Φορητά Αυτοοργανωμένα Δίκτυα

ΔΟΕ Διαοχηματική Επικοινωνία

UAV Unmanned Aerial Vehicle

AODV Ad-hoc On-demand Distance Vector

RREQ Route Request

RREP Route Reply

RERR Route Error

DSR Dynamic Source Routing

OLSR Optimized Link State Routing

MPR Multipoint Relay

NS3 Network Simulator 3

GPLv2 General Public License version 2

GCC GNU C Compiler

WSL Windows Subsystem for Linux

LTS Long Term Release

GDB GNU Debugger

MANET Mobile Ad-Hoc Network

VANET Vehicle Ad-Hoc Network

IVC Inter Vehicle Communication

CSV Comma Separated Values

Παράρτημα Β΄

Πηγαίος Κώδικας

Β΄.1 Βοηθητικό Πρόγραμμα Εγκατάστασης Προαπαιτούμενων NS3

```
# Written by Andreas Manitsas
# Used for the Thesis "Supporting Real Time Data Processing
   in an Unmanned Aerial Vehicle Platform"
# University of Western Macedonia - Department of Electrical
   and Computer Engineering - 2020

# This script contains all the dependencies required for
   running NS3 3.31 on Ubuntu 18.04 LTS x64.
# Current as of 09/10/2020. Tested on Windows 10 version 2004
    with Ubuntu 18.04 LTS x64 running on WSL2.

CYAN='\033[0;36m'
NOCOLOR='\033[0m'

echo "${CYAN}--01/20-- Minimal requirements for C++ (release)
   : This is the minimal set of packages needed to run ns-3
   from a released tarball${NOCOLOR}"
apt -y install gcc g++ python

echo "${CYAN}--02/20-- Minimal requirements for Python (
   release): This is the minimal set of packages needed to
   work with Python bindings from a released tarball.${
   NOCOLOR}"
apt -y install gcc g++ python python-dev

echo "${CYAN}--03/20-- Minimal requirements for Python (
   development): For use of ns-3-allinone repository (cloned
   from Mercurial), additional packages are needed to fetch
   and successfully install pybindgen:${NOCOLOR}"
apt -y install mercurial python-setuptools git
```

```
echo "${CYAN}--04/20-- Netanim animator: qt5 development
   tools are needed for Netanim animator; qt4 will also work
   but we have migrated to qt5${NOCOLOR}"
apt -y install qt5-default
echo "${CYAN}--05/20-- For Ubuntu 20.04, python-pygoocanvas
   is no longer provided. The ns-3.29 release and later
   upgrades the support to GTK+ version 3, and requires these
    packages:${NOCOLOR}"
apt -y install gir1.2-goocanvas-2.0 python-gi python-gi-cairo
    python-pygraphviz python3-gi python3-gi-cairo python3-
   pygraphviz gir1.2-gtk-3.0 ipython ipython3

echo "${CYAN}--06/20-- Support for MPI-based distributed
   emulation:${NOCOLOR}"
apt -y install openmpi-bin openmpi-common openmpi-doc
   libopenmpi-dev

echo "${CYAN}--07/20-- Support for bake build tool:${NOCOLOR}
   "
apt -y install autoconf cvs bzr unrar

echo "${CYAN}--08/20-- Debugging:${NOCOLOR}"
apt -y install gdb valgrind

echo "${CYAN}--09/20-- Support for utils/check-style.py code
   style check program:${NOCOLOR}"
apt -y install uNOCOLORrustify

echo "${CYAN}--10/20-- Doxygen and related inline
   documentation:${NOCOLOR}"
apt -y install doxygen graphviz imagemagick
apt -y install texlive texlive-extra-utils texlive-latex-
   extra texlive-font-utils texlive-lang-portuguese dvipng
   latexmk

echo "${CYAN}--11/20-- The ns-3 manual and tutorial are
   written in reStructuCYANText for Sphinx (doc/tutorial, doc
   /manual, doc/models), and figures typically in dia (also
   needs the texlive packages above):${NOCOLOR}"
echo "${CYAN}Note: Sphinx version >= 1.12 requiCYAN for ns
   -3.15. To check your version, type (sphinx-build). To
   fetch this package alone, outside of the Ubuntu package
   system, try (sudo easy_install -U Sphinx).${NOCOLOR}"
apt -y install python-sphinx dia

echo "${CYAN}--12/20-- GNU Scientific Library (GSL) support
   for more accurate WiFi error models${NOCOLOR}"
```

```
apt -y install gsl-bin libgsl23 libgsl-dev

echo "${CYAN}--13/20-- The Network Simulation Cradle (nsc)
   requires the flex lexical analyzer and bison parser
   generator:${NOCOLOR}"
apt -y install flex bison libfl-dev

echo "${CYAN}--14/20-- To read pcap packet traces${NOCOLOR}"
apt -y install tcpdump

echo "${CYAN}--15/20-- Database support for statistics
   framework${NOCOLOR}"
apt -y install sqlite sqlite3 libsqlite3-dev

echo "${CYAN}--16/20-- XML-based version of the config store
   (requires libxml2 >= version 2.7)${NOCOLOR}"
apt -y install libxml2 libxml2-dev

echo "${CYAN}--17/20-- Support for generating modified python
    bindings${NOCOLOR}"
apt -y install cmake libc6-dev libc6-dev-i386 libclang-dev
   llvm-dev automake
pip3 install cxxfilt

echo "${CYAN}--18/20-- A GTK-based configuration system${
   NOCOLOR}"
apt -y install libgtk2.0-0 libgtk2.0-dev

echo "${CYAN}--19/20-- To experiment with virtual machines
   and ns-3${NOCOLOR}"
apt -y install vtun lxc

echo "${CYAN}--20/20-- Support for openflow module (requires
   some boost libraries)${NOCOLOR}"
apt -y install libboost-signals-dev libboost-filesystem-dev
```

Β΄.2 Σενάριο Προσομοίωσης ΑΔΟ

```
/*
* Modified and Commented by Andreas Manitsas
* Based on the NS3 example manet-routing-compare.cc and the
   work of Dr. Pradeep Kumar (https://www.nsnam.com/2019/05/
   comparison-of-adhoc-routing-protocols.html)
* Used for the Thesis "Supporting Real Time Data Processing
   in an Unmanned Aerial Vehicle Platform"
* University of Western Macedonia - Department of Electrical
   and Computer Engineering - 2020
*/
```

```
/*
* Current Configuration is:
* -------------------------
* Routing Protocol: AODV/DSR/OLSR
* Mobility Model: Random Waypoint
* Simulation Area: 2000x2000 m
* Number of nodes: 10/15/20/25
* Simulation Time: 60 sec
* UDP Packet Size: 1000 byte
* Wireless Standard: 802.11g
* Loss Model: Friis
* Node Speed: 10 m/s
* Time Node is stationary: 1 sec
* Bandwidth: 1Mbps
* Transmission Power: 27 dBm (500 mW)
*/

#define VERSION 0.8

//C++ Libraries
#include <fstream>
#include <iostream>

//NS3 Libraries
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/position-allocator.h"
#include "ns3/animation-interface.h"

using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE("routingProtocolsMANET");

class RoutingExperiment
{
        public:
                RoutingExperiment();
```

```
                void Run(int nSinks, double txp, std::string
                   CSVfileName);
                //static void SetMACParam (ns3::
                   NetDeviceContainer & devices,
                //                                  int
                   slotDistance);
                std::string CommandSetup(int argc, char **
                   argv);

        private:
                Ptr<Socket> SetupPacketReceive(Ipv4Address
                   addr, Ptr<Node> node);
                void ReceivePacket(Ptr<Socket> socket);
                void CheckThroughput();

                uint32_t port;
                uint32_t bytesTotal; //Bytes received counter
                uint32_t packetsReceived; //Packets received
                   coutner

                std::string m_CSVfileName; //Output filename
                int m_nSinks; //Number of receivers
                std::string m_protocolName; //Routing
                   protocol used (string)
                double m_txp; //Transmit power (dBm)
                bool m_traceMobility; //Enable-Disable
                   mobility tracing
                uint32_t m_protocol; //Routing protocol
                   selector (number)
};

//Constuctor with default values. those can be overwritten
   with cmd arguments
RoutingExperiment::RoutingExperiment()
{
        port = 9;                                       //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        bytesTotal = 0;                                 //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        packetsReceived = 0;                            //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_CSVfileName = "routingProtocolsMANET.csv";    //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_traceMobility = false;                        //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_protocol = 2; // AODV
           //<<<--- MODIFY THIS OR USE CMD ARGUMENTS
}
```

```
//Print when each packet is received, on which port and from
   which sender
static inline std::string PrintReceivedPacket(Ptr<Socket>
   socket, Ptr<Packet> packet, Address senderAddress)
{
        std::ostringstream oss; //Output String Stream

        oss << Simulator::Now().GetSeconds() << " " << socket
           ->GetNode()->GetId();

        if (InetSocketAddress::IsMatchingType(senderAddress))
        {
                InetSocketAddress addr = InetSocketAddress::
                   ConvertFrom(senderAddress);
                oss << " received one packet from " << addr.
                   GetIpv4();
        }
        else
        {
                oss << " received one packet!";
        }

        return oss.str();
}

//Count how packets are received from each sender
void RoutingExperiment::ReceivePacket(Ptr<Socket> socket) //
   Count how packets are received from each sender
{
        Ptr<Packet> packet;
        Address senderAddress;
        while(packet = socket->RecvFrom(senderAddress))
        {
                bytesTotal += packet->GetSize();
                packetsReceived += 1;
                NS_LOG_UNCOND(PrintReceivedPacket(socket,
                   packet, senderAddress));
        }
}

//Write simulation data at regular interval to the file
void RoutingExperiment::CheckThroughput()
{
        double intervalTime = 1.0; //How often to write data
           to file (seconds)   <<<--- MODIFY THIS
        double kbs = (bytesTotal * 8.0) / 1000;
        bytesTotal = 0;
```

```
        std::ofstream out(m_CSVfileName.c_str(), std::ios::
           app);

        out << (Simulator::Now()).GetSeconds() << "," << kbs
           << "," << packetsReceived << "," << m_nSinks << ",
           " << m_protocolName << "," << m_txp << "" << std::
           endl;

        out.close();
        packetsReceived = 0;
        Simulator::Schedule(Seconds(intervalTime), &
           RoutingExperiment::CheckThroughput, this); //
           Schedule to run this function every X seconds
}

Ptr<Socket> RoutingExperiment::SetupPacketReceive(Ipv4Address
    addr, Ptr<Node> node)
{
        TypeId tid = TypeId::LookupByName("ns3::
           UdpSocketFactory");
        Ptr<Socket> sink = Socket::CreateSocket(node, tid);
        InetSocketAddress local = InetSocketAddress(addr,
           port);
        sink->Bind(local);
        sink->SetRecvCallback(MakeCallback(&RoutingExperiment
           ::ReceivePacket, this));

        return sink;
}

//CMD arguments
std::string RoutingExperiment::CommandSetup(int argc, char **
   argv)
{
        CommandLine cmd(__FILE__);
        cmd.AddValue("CSVfileName", "The name of the CSV
           output file name", m_CSVfileName);
        cmd.AddValue("traceMobility", "Enable mobility
           tracing", m_traceMobility);
        cmd.AddValue("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR"
           , m_protocol);
        cmd.Parse(argc, argv);
        return m_CSVfileName;
}

void RoutingExperiment::Run(int nSinks, double txp, std::
   string CSVfileName)
```

```
{
        Packet::EnablePrinting();
        m_nSinks = nSinks;
        m_txp = txp;
        m_CSVfileName = CSVfileName;

        int nWifis = 25; //Number of nodes in the simulation
                         <<<--- MODIFY THIS

        double TotalTime = 60.0; //Total simulation time (sec
           )             <<<--- MODIFY THIS
        std::string rate("1000000bps"); //Data rate of
           wireless link (bps)   <<<--- MODIFY THIS
        std::string phyMode("DsssRate11Mbps");
        std::string tr_name("routingProtocolsFANET");
        int nodeSpeed = 10; //Speed of a node's movement (m/s
           )             <<<--- MODIFY THIS
        int nodePause = 1; //Time a node can stay stationary
           (sec)         <<<--- MODIFY THIS
        m_protocolName = "protocol";

        Config::SetDefault("ns3::OnOffApplication::PacketSize
           ", StringValue("1000")); //Packet size   <<<---
           MODIFY THIS
        Config::SetDefault("ns3::OnOffApplication::DataRate",
             StringValue(rate));

        //Set Non-unicastMode rate to unicast mode
        Config::SetDefault("ns3::WifiRemoteStationManager::
           NonUnicastMode", StringValue(phyMode));

        NodeContainer adhocNodes;
        adhocNodes.Create(nWifis);

        // setting up wifi phy and channel using helpers
        WifiHelper wifi;
        wifi.SetStandard(WIFI_PHY_STANDARD_80211b); //WiFi
           standard. In this case, 802.11b

        YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::
           Default();
        YansWifiChannelHelper wifiChannel;
        wifiChannel.SetPropagationDelay("ns3::
           ConstantSpeedPropagationDelayModel");
        wifiChannel.AddPropagationLoss("ns3::
           FriisPropagationLossModel"); //Friis Loss Model
        wifiPhy.SetChannel(wifiChannel.Create());
```

```
// Add a mac and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager("ns3::
   ConstantRateWifiManager", "DataMode",StringValue(
   phyMode), "ControlMode",StringValue(phyMode));

wifiPhy.Set("TxPowerStart",DoubleValue(txp));
wifiPhy.Set("TxPowerEnd", DoubleValue(txp));

wifiMac.SetType("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices = wifi.Install(
   wifiPhy, wifiMac, adhocNodes);

MobilityHelper mobilityAdhoc;
int64_t streamIndex = 0; // used to get consistent
   mobility across scenarios

ObjectFactory pos;
pos.SetTypeId("ns3::RandomRectanglePositionAllocator"
   );
pos.Set("X", StringValue("ns3::UniformRandomVariable[
   Min=0.0|Max=2000.0]")); //Grid limit on X axis
   <<<--- MODIFY THIS
pos.Set("Y", StringValue("ns3::UniformRandomVariable[
   Min=0.0|Max=2000.0]")); //Grid limit on Y axis
   <<<--- MODIFY THIS

Ptr<PositionAllocator> taPositionAlloc = pos.Create()
   ->GetObject<PositionAllocator>();
streamIndex += taPositionAlloc->AssignStreams(
   streamIndex);

std::stringstream ssSpeed;
ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max="
   << nodeSpeed << "]";
std::stringstream ssPause;
ssPause << "ns3::ConstantRandomVariable[Constant=" <<
    nodePause << "]";

mobilityAdhoc.SetMobilityModel("ns3::
   RandomWaypointMobilityModel", "Speed", StringValue
   (ssSpeed.str()), "Pause", StringValue(ssPause.str
   ()), "PositionAllocator", PointerValue(
   taPositionAlloc));

mobilityAdhoc.SetPositionAllocator(taPositionAlloc);
mobilityAdhoc.Install(adhocNodes);
streamIndex += mobilityAdhoc.AssignStreams(adhocNodes
```

```
   , streamIndex);
NS_UNUSED(streamIndex); //From this point,
   streamIndex is unused

AodvHelper aodv;
OlsrHelper olsr;
DsdvHelper dsdv;
DsrHelper dsr;
DsrMainHelper dsrMain;
Ipv4ListRoutingHelper list;
InternetStackHelper internet;

switch(m_protocol)
{
        case 1:
                list.Add(olsr, 100);
                m_protocolName = "OLSR";
                break;
        case 2:
                list.Add(aodv, 100);
                m_protocolName = "AODV";
                break;
        case 3:
                list.Add(dsdv, 100);
                m_protocolName = "DSDV";
                break;
        case 4:
                m_protocolName = "DSR";
                break;
        default:
                NS_FATAL_ERROR("No such protocol:" <<
                    m_protocol);
}

if(m_protocol < 4)
{
        internet.SetRoutingHelper(list);
        internet.Install(adhocNodes);
}
else if(m_protocol == 4)
{
        internet.Install(adhocNodes);
        dsrMain.Install(dsr, adhocNodes);
}

NS_LOG_INFO("assigning ip address");

Ipv4AddressHelper addressAdhoc;
```

```
addressAdhoc.SetBase("10.1.1.0", "255.255.255.0"); //
   IP adress range and subnet mask
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign(adhocDevices);

OnOffHelper onoff1("ns3::UdpSocketFactory",Address())
   ;
onoff1.SetAttribute("OnTime", StringValue("ns3::
   ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute("OffTime", StringValue("ns3::
   ConstantRandomVariable[Constant=0.0]"));

for(int i = 0; i < nSinks; i++)
{
        Ptr<Socket> sink = SetupPacketReceive(
           adhocInterfaces.GetAddress(i), adhocNodes.
           Get(i));

        AddressValue remoteAddress(InetSocketAddress(
           adhocInterfaces.GetAddress(i), port));
        onoff1.SetAttribute("Remote", remoteAddress);

        Ptr<UniformRandomVariable> var = CreateObject
           <UniformRandomVariable>();
        ApplicationContainer temp = onoff1.Install(
           adhocNodes.Get(i + nSinks));
        //temp.Start(Seconds(var->GetValue
           (100.0,101.0))); //Delay to start at 100
           sec
        temp.Start(Seconds(0.0));
        temp.Stop(Seconds(TotalTime));
}

std::stringstream ss;
ss << nWifis;
std::string nodes = ss.str();

std::stringstream ss2;
ss2 << nodeSpeed;
std::string sNodeSpeed = ss2.str();

std::stringstream ss3;
ss3 << nodePause;
std::string sNodePause = ss3.str();

std::stringstream ss4;
ss4 << rate;
std::string sRate = ss4.str();
```

```
        AsciiTraceHelper ascii;
        MobilityHelper::EnableAsciiAll(ascii.CreateFileStream
           (tr_name + ".mob"));

        Ptr<FlowMonitor> flowmon; //Flowmonitor tracks the
           flow of data packets and outputs them in XML file.
            Then we use a python tool to analyze the XML.
        FlowMonitorHelper flowmonHelper;
        flowmon = flowmonHelper.InstallAll(); //Install the
           flowmonitor probe to all the nodes


        NS_LOG_INFO("Run Simulation.");

        CheckThroughput();
        std::cout << "Creating XML Animation File: " <<
           m_CSVfileName << " ...\n";
        AnimationInterface anim("routingProtocolsMANET.xml");
            //Create XML file for NetAnim visualisation
        Simulator::Stop(Seconds(TotalTime));
        Simulator::Run();

        flowmon->SerializeToXmlFile((tr_name + ".flowmon").
           c_str(), false, false); //Name of the XML file
           storing the Flowmonitor data

        Simulator::Destroy();
}

//
   ------------------------------------------------------------------

int main (int argc, char *argv[])
{
        RoutingExperiment experiment;
        std::string CSVfileName = experiment.CommandSetup(
           argc,argv);

        //blank out the last output file and write the column
            headers
        std::ofstream out(CSVfileName.c_str());
        out << "SimulationSecond," << "ReceiveRate," << "
           PacketsReceived," << "NumberOfSinks," << "
           RoutingProtocol," << "TransmissionPower" << std::
           endl;
        out.close();
```

```
        int nSinks = 12; //Number of receivers       <<<---
           MODIFY THIS
        double txp = 27.0; //Transmitt power (dBm)   <<<---
           MODIFY THIS

        experiment.Run(nSinks, txp, CSVfileName);
}
```

Β΄.3 Σενάριο Προσομοίωσης ΙΑΔ

```
/*
* Modified and Commented by Andreas Manitsas
* Based on the NS3 example manet-routing-compare.cc and the
   work of Dr. Pradeep Kumar (https://www.nsnam.com/2019/05/
   comparison-of-adhoc-routing-protocols.html)
* Used for the Thesis "Supporting Real Time Data Processing
   in an Unmanned Aerial Vehicle Platform"
* University of Western Macedonia - Department of Electrical
   and Computer Engineering - 2020
*/

/*
* Current Configuration is:
* -------------------------
* Routing Protocol: AODV/DSR/OLSR
* Mobility Model: Gauss Markov
* Simulation Area: 2000x2000x150 m
* Number of nodes: 10/15/20/25
* Simulation Time: 60 sec
* UDP Packet Size: 1000 byte
* Wireless Standard: 802.11g
* Loss Model: Friis
* Node Speed: 10 m/s
* Time Node is stationary: 1 sec
* Bandwidth: 1Mbps
* Transmission Power: 27 dBm (500 mW)
*/

#define VERSION 0.13

//C++ Libraries
#include <fstream>
#include <iostream>

//NS3 Libraries
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
```

```
#include "ns3/mobility-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/position-allocator.h"
#include "ns3/animation-interface.h"

using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE("routingProtocolsFANET");

class RoutingExperiment
{
        public:
                RoutingExperiment();
                void Run(int nSinks, double txp, std::string
                   CSVfileName);
                //static void SetMACParam (ns3::
                   NetDeviceContainer & devices,
                //                                  int
                   slotDistance);
                std::string CommandSetup(int argc, char **
                   argv);

        private:
                Ptr<Socket> SetupPacketReceive(Ipv4Address
                   addr, Ptr<Node> node);
                void ReceivePacket(Ptr<Socket> socket);
                void CheckThroughput();

                uint32_t port;
                uint32_t bytesTotal; //Bytes received counter
                uint32_t packetsReceived; //Packets received
                   coutner

                std::string m_CSVfileName; //Output filename
                int m_nSinks; //Number of receivers
                std::string m_protocolName; //Routing
                   protocol used (string)
                double m_txp; //Transmit power (dBm)
                bool m_traceMobility; //Enable-Disable
                   mobility tracing
                uint32_t m_protocol; //Routing protocol
```

```
                    selector (number)
};

//Constuctor with default values. those can be overwritten
   with cmd arguments
RoutingExperiment::RoutingExperiment()
{
        port = 9;                                          //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        bytesTotal = 0;                                    //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        packetsReceived = 0;                               //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_CSVfileName = "routingProtocolsFANET.csv";       //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_traceMobility = false;                           //
           <<<--- MODIFY THIS OR USE CMD ARGUMENTS
        m_protocol = 2; // AODV
           //<<<--- MODIFY THIS OR USE CMD ARGUMENTS
}

//Print when each packet is received, on which port and from
   which sender
static inline std::string PrintReceivedPacket(Ptr<Socket>
   socket, Ptr<Packet> packet, Address senderAddress) //Print
    when each packet is received, on which port and from
   which sender
{
        std::ostringstream oss; //Output String Stream

        oss << Simulator::Now().GetSeconds() << " " << socket
           ->GetNode()->GetId();

        if (InetSocketAddress::IsMatchingType(senderAddress))
        {
                InetSocketAddress addr = InetSocketAddress::
                   ConvertFrom(senderAddress);
                oss << " received one packet from " << addr.
                   GetIpv4();
        }
        else
        {
                oss << " received one packet!";
        }

        return oss.str();
}
```

```
//Count how packets are received from each sender
void RoutingExperiment::ReceivePacket(Ptr<Socket> socket) //
   Count how packets are received from each sender
{
        Ptr<Packet> packet;
        Address senderAddress;
        while(packet = socket->RecvFrom(senderAddress))
        {
                bytesTotal += packet->GetSize();
                packetsReceived += 1;
                NS_LOG_UNCOND(PrintReceivedPacket(socket,
                   packet, senderAddress));
        }
}

//Write simulation data at regular interval to the file
void RoutingExperiment::CheckThroughput()
{
        double intervalTime = 1.0; //How often to write data
           to file (seconds)   <<<--- MODIFY THIS
        double kbs = (bytesTotal * 8.0) / 1000;
        bytesTotal = 0;

        std::ofstream out(m_CSVfileName.c_str(), std::ios::
           app);

        out << (Simulator::Now()).GetSeconds() << "," << kbs
           << "," << packetsReceived << "," << m_nSinks << ",
           " << m_protocolName << "," << m_txp << "" << std::
           endl;

        out.close();
        packetsReceived = 0;
        Simulator::Schedule(Seconds(intervalTime), &
           RoutingExperiment::CheckThroughput, this); //
           Schedule to run this function every X seconds
}

Ptr<Socket> RoutingExperiment::SetupPacketReceive(Ipv4Address
    addr, Ptr<Node> node)
{
        TypeId tid = TypeId::LookupByName("ns3::
           UdpSocketFactory");
        Ptr<Socket> sink = Socket::CreateSocket(node, tid);
        InetSocketAddress local = InetSocketAddress(addr,
           port);
        sink->Bind(local);
        sink->SetRecvCallback(MakeCallback(&RoutingExperiment
```

```
            ::ReceivePacket, this));

        return sink;
}

//CMD arguments
std::string RoutingExperiment::CommandSetup(int argc, char **
   argv)
{
        CommandLine cmd(__FILE__);
        cmd.AddValue("CSVfileName", "The name of the CSV
           output file name", m_CSVfileName);
        cmd.AddValue("traceMobility", "Enable mobility
           tracing", m_traceMobility);
        cmd.AddValue("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR"
           , m_protocol);
        cmd.Parse(argc, argv);
        return m_CSVfileName;
}

void RoutingExperiment::Run(int nSinks, double txp, std::
   string CSVfileName)
{
        Packet::EnablePrinting();
        m_nSinks = nSinks;
        m_txp = txp;
        m_CSVfileName = CSVfileName;

        int nWifis = 25; //Number of nodes in the simulation
                         <<<--- MODIFY THIS

        double TotalTime = 60.0; //Total simulation time (sec
           )             <<<--- MODIFY THIS
        std::string rate("1000000bps"); //Data rate of
           wireless link (bps)   <<<--- MODIFY THIS
        std::string phyMode("DsssRate11Mbps");
        std::string tr_name("routingProtocolsFANET");
        int nodeSpeed = 10; //Speed of a node's movement (m/s
           )             <<<--- MODIFY THIS
        int nodePause = 1; //Time a node can stay stationary
           (sec)         <<<--- MODIFY THIS
        m_protocolName = "protocol";

        Config::SetDefault("ns3::OnOffApplication::PacketSize
           ", StringValue("1000")); //Packet size   <<<---
           MODIFY THIS
        Config::SetDefault("ns3::OnOffApplication::DataRate",
             StringValue(rate));
```

```
//Set Non-unicastMode rate to unicast mode
Config::SetDefault("ns3::WifiRemoteStationManager::
   NonUnicastMode", StringValue(phyMode));

NodeContainer adhocNodes;
adhocNodes.Create(nWifis);

// setting up wifi phy and channel using helpers
WifiHelper wifi;
wifi.SetStandard(WIFI_PHY_STANDARD_80211b); //WiFi
   standard. In this case, 802.11b

YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::
   Default();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay("ns3::
   ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss("ns3::
   FriisPropagationLossModel"); //Friis Loss Model
wifiPhy.SetChannel(wifiChannel.Create());

// Add a mac and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager("ns3::
   ConstantRateWifiManager", "DataMode",StringValue(
   phyMode), "ControlMode",StringValue(phyMode));

wifiPhy.Set("TxPowerStart",DoubleValue(txp));
wifiPhy.Set("TxPowerEnd", DoubleValue(txp));

wifiMac.SetType("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices = wifi.Install(
   wifiPhy, wifiMac, adhocNodes);

MobilityHelper mobilityAdhoc;
int64_t streamIndex = 0; // used to get consistent
   mobility across scenarios

ObjectFactory pos;
pos.SetTypeId("ns3::RandomBoxPositionAllocator");
pos.Set("X", StringValue("ns3::UniformRandomVariable[
   Min=0.0|Max=2000.0]")); //Grid limit on X axis
   <<<--- MODIFY THIS
pos.Set("Y", StringValue("ns3::UniformRandomVariable[
   Min=0.0|Max=2000.0]")); //Grid limit on Y axis
   <<<--- MODIFY THIS
pos.Set("Z", StringValue("ns3::UniformRandomVariable[
```

```
   Min=0.0|Max=100.0]")); //Grid limit on Z axis
   <<<--- MODIFY THIS

Ptr<PositionAllocator> taPositionAlloc = pos.Create()
   ->GetObject<PositionAllocator>();
streamIndex += taPositionAlloc->AssignStreams(
   streamIndex);

std::stringstream ssSpeed;
ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max="
   << nodeSpeed << "]";
std::stringstream ssPause;
ssPause << "ns3::ConstantRandomVariable[Constant=" <<
    nodePause << "]";

mobilityAdhoc.SetMobilityModel("ns3::
   GaussMarkovMobilityModel", "Bounds", BoxValue(Box
   (0, 2000, 0, 2000, 0, 100)), "TimeStep", TimeValue
   (Seconds(0.5)), "Alpha", DoubleValue(0.85), "
   MeanVelocity", StringValue("ns3::
   UniformRandomVariable[Min=800|Max=1200]"), "
   MeanDirection", StringValue("ns3::
   UniformRandomVariable[Min=0|Max=6.283185307]"), "
   MeanPitch", StringValue("ns3::
   UniformRandomVariable[Min=0.05|Max=0.05]"), "
   NormalVelocity", StringValue("ns3::
   NormalRandomVariable[Mean=0.0|Variance=0.0|Bound
   =0.0]"), "NormalDirection", StringValue("ns3::
   NormalRandomVariable[Mean=0.0|Variance=0.2|Bound
   =0.4]"), "NormalPitch", StringValue("ns3::
   NormalRandomVariable[Mean=0.0|Variance=0.02|Bound
   =0.04]"));

mobilityAdhoc.SetPositionAllocator(taPositionAlloc);
mobilityAdhoc.Install(adhocNodes);
streamIndex += mobilityAdhoc.AssignStreams(adhocNodes
   , streamIndex);
NS_UNUSED(streamIndex); //From this point,
   streamIndex is unused

AodvHelper aodv;
OlsrHelper olsr;
DsdvHelper dsdv;
DsrHelper dsr;
DsrMainHelper dsrMain;
Ipv4ListRoutingHelper list;
InternetStackHelper internet;
```

```
switch(m_protocol)
{
        case 1:
                list.Add(olsr, 100);
                m_protocolName = "OLSR";
                break;
        case 2:
                list.Add(aodv, 100);
                m_protocolName = "AODV";
                break;
        case 3:
                list.Add(dsdv, 100);
                m_protocolName = "DSDV";
                break;
        case 4:
                m_protocolName = "DSR";
                break;
        default:
                NS_FATAL_ERROR("No such protocol:" <<
                    m_protocol);
}

if(m_protocol < 4)
{
        internet.SetRoutingHelper(list);
        internet.Install(adhocNodes);
}
else if(m_protocol == 4)
{
        internet.Install(adhocNodes);
        dsrMain.Install(dsr, adhocNodes);
}

NS_LOG_INFO("assigning ip address");

Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase("10.1.1.0", "255.255.255.0"); //
   IP adress range and subnet mask
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign(adhocDevices);

OnOffHelper onoff1("ns3::UdpSocketFactory",Address())
   ;
onoff1.SetAttribute("OnTime", StringValue("ns3::
   ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute("OffTime", StringValue("ns3::
   ConstantRandomVariable[Constant=0.0]"));
```

```
for(int i = 0; i < nSinks; i++)
{
        Ptr<Socket> sink = SetupPacketReceive(
           adhocInterfaces.GetAddress(i), adhocNodes.
           Get(i));

        AddressValue remoteAddress(InetSocketAddress(
           adhocInterfaces.GetAddress(i), port));
        onoff1.SetAttribute("Remote", remoteAddress);

        Ptr<UniformRandomVariable> var = CreateObject
           <UniformRandomVariable>();
        ApplicationContainer temp = onoff1.Install(
           adhocNodes.Get(i + nSinks));
        //temp.Start(Seconds(var->GetValue
           (100.0,101.0))); //Delay to start at 100
           sec
        temp.Start(Seconds(0.0));
        temp.Stop(Seconds(TotalTime));
}

std::stringstream ss;
ss << nWifis;
std::string nodes = ss.str();

std::stringstream ss2;
ss2 << nodeSpeed;
std::string sNodeSpeed = ss2.str();

std::stringstream ss3;
ss3 << nodePause;
std::string sNodePause = ss3.str();

std::stringstream ss4;
ss4 << rate;
std::string sRate = ss4.str();

AsciiTraceHelper ascii;
MobilityHelper::EnableAsciiAll(ascii.CreateFileStream
   (tr_name + ".mob"));

Ptr<FlowMonitor> flowmon; //Flowmonitor tracks the
   flow of data packets and outputs them in XML file.
    Then we use a python tool to analyze the XML.
FlowMonitorHelper flowmonHelper;
flowmon = flowmonHelper.InstallAll(); //Install the
   flowmonitor probe to all the nodes
```

```
        NS_LOG_INFO("Run Simulation.");

        CheckThroughput();
        std::cout << "Creating XML Animation File: " <<
           m_CSVfileName << " ...\n";
        AnimationInterface anim("routingProtocolsFANET.xml");
            //Create XML file for NetAnim visualisation
        Simulator::Stop(Seconds(TotalTime));
        Simulator::Run();

        flowmon->SerializeToXmlFile((tr_name + ".flowmon").
           c_str(), false, false); //Name of the XML file
           storing the Flowmonitor data

        Simulator::Destroy();
}

//
   ------------------------------------------------------------------------

int main (int argc, char *argv[])
{
        RoutingExperiment experiment;
        std::string CSVfileName = experiment.CommandSetup(
           argc,argv);

        //blank out the last output file and write the column
            headers
        std::ofstream out(CSVfileName.c_str());
        out << "SimulationSecond," << "ReceiveRate," << "
           PacketsReceived," << "NumberOfSinks," << "
           RoutingProtocol," << "TransmissionPower" << std::
           endl;
        out.close();

        int nSinks = 12; //Number of receivers       <<<---
           MODIFY THIS
        double txp = 27.0; //Transmitt power (dBm)   <<<---
           MODIFY THIS

        experiment.Run(nSinks, txp, CSVfileName);
}
```

Βιβλιογραφία

[1] Wikipedia, "Mq-9 reaper image." [Online]. Available: https://en.wikipedia.org/wiki/File:MQ-9_Reaper_UAV_(cropped).jpg

[2] ——, "Flaps image." [Online]. Available: https://en.wikipedia.org/wiki/Flap_(aeronautics)#/media/File:Airfoil_lift_improvement_devices_(flaps).png

[3] ——, "Aircraft principal axis." [Online]. Available: https://en.wikipedia.org/wiki/Aircraft_principal_axes#/media/File:Flight_dynamics_with_text_ortho.svg

[4] "Dji spark image." [Online]. Available: https://www.germanos.gr/product/dji-drone-spark-leuko/20365081/

[5] Wikipedia, "Quadcopter," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Quadcopter

[6] "Xoar titan t5000 uav motor." [Online]. Available: https://www.xoarintl.com/brushless-electric-motors/titan/titan-T5000-heavy-lifting-series/

[7] "Safran tr 60 turbojet." [Online]. Available: https://www.safran-power-units.com/propulsion-systems/turbojet-engines/tr-60-5

[8] Sharanabasaweshwara A. Asundi and Syed Firasat Ali, "Parametric study of a turbofan engine with an auxiliary high-pressure bypass," *International Journal of Turbomachinery Propulsion and Power*, pp. 1–13, 2019.

[9] Yasin Sohret, Onder Turan, and T. Hikmet Karakoc, "Analysis of combustion efficiency for turbofan engine combustor using matlab," *IACSIT International Journal of Engineering and Technology*, vol. 7, no. 2, pp. 86–90, 2015.

[10] C. Szczepanski, "Uavs and their avionic systems: Development trends and their influence on polish research and market," *Aviation*, vol. 19, no. 1, pp. 49–57, 2015.

[11] Airbus, "Airbus demonstrates first fully automatic vision-based take-off," 2020. [Online]. Available: https://www.airbus.com/newsroom/press-releases/en/2020/01/airbus-demonstrates-first-fully-automatic-visionbased-takeoff.html

[12] Wikipedia, "Degrees of freedom (mechanics)," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)

[13] ——, "Inertial measurement unit." [Online]. Available: https://en.wikipedia.org/wiki/Inertial_measurement_unit

[14] “433mhz telemetry for pixhawk.” [Online]. Available: https://drones.altigator.com/radio-telemetry-kit-a-for-pixhawk-433mhz-p-42334.html

[15] S. Chakravarty, “World’s top military uav manufacturers and global market insight,” 2019. [Online]. Available: https://www.marketresearchreports.com/blog/2019/10/31/worlds-top-military-uav-manufacturers-and-global-market-insight

[16] Abayomi O. Agbeyangi, Joseph O. Odiete, Adam B. Olorunlomerue, “Review of uavs used for aerial surveillance,” *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, vol. 3, pp. 5713–5719, 2016.

[17] S. Conditt, “Arizona fpv’s home surveillance drones,” 2014. [Online]. Available: https://www.realworldsurvivor.com/2014/07/22/arizona-fpvs-home-surveillance-drones/

[18] “Iai heron.” [Online]. Available: https://en.wikipedia.org/wiki/IAI_Heron

[19] IAI, “Israel will lease iai heron uav’s to greece,” 2020. [Online]. Available: https://www.iai.co.il/israel-will-lease-iai-heron-to-greece

[20] Sonia Waharte, Niki Trigoni, “Supporting search and rescue operations with uavs,” 2010.

[21] “Dji matrice 210.” [Online]. Available: https://www.dji.com/gr/matrice-200-series

[22] “Northrop grumman rq-4 global hawk.” [Online]. Available: https://en.wikipedia.org/wiki/Northrop_Grumman_RQ-4_Global_Hawk

[23] Wikipedia, “Synthetic aperture radar,” 2020. [Online]. Available: https://en.wikipedia.org/wiki/Synthetic-aperture_radar

[24] ——, “Northrop grumman rq-4 global hawk,” 2020. [Online]. Available: https://en.wikipedia.org/wiki/Northrop_Grumman_RQ-4_Global_Hawk

[25] ICTWorks.org, “Which is better for farm size mapping: Uav drones or satellites?” 2020. [Online]. Available: https://www.ictworks.org/farm-size-mapping-uav-drones-satellites/

[26] Anand Nayyar, Linesh Raja, “Agriculture drones: A modern breakthrough in precision agriculture,” *Journal of Statistics and Management Systems*, vol. 20, pp. 507–518, 2017.

[27] S. Fecht, “How drone swarms could help protect us from tornadoes,” 2017. [Online]. Available: https://www.popsci.com/how-drone-swarms-could-make-tornado-predictions-better/

[28] “Dji phanton 4.” [Online]. Available: https://www.dji.com/gr/phantom-4-pro

[29] Chanyoung Ju, Hyoung Il Son, “Multiple uav systems for agricultural applications: Control, implementation, and evaluation,” *Electronics*, vol. 7, pp. 1–19, 2018.

[30] Georgy Skorobogatov, Cristina Barrado, Esther Salamí, “Multiple uav systems: A survey,” *Unmanned Systems*, vol. 8, pp. 149–169, 2020.

[31] Luis Merino, José Ramiro Martínez-de Dios, Aníbal Ollero, "Cooperative unmanned aerial systems for fire detection, monitoring, and extinguishing," *Handbook of Unmanned Aerial Vehicles*, pp. 2693–2722, 2014.

[32] Jurgen Scherer, Saeed Yahyanejad, Samira Hayat et al, "An autonomous multi-uav system for search and rescue," 2015.

[33] Hazim Shakhatreh, Ahmad H. Sawalmeh et al, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[34] Wikipedia, "Wireless ad hoc network," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Wireless_ad_hoc_network

[35] Omar Sami Oubbati, Mohammed Atiquzzaman, Pascal Lorenz et al, "Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives," *IEEE Access*, vol. 7, pp. 81 057–81 105, 2019.

[36] Gary Brown, William Harris, "How satellites work," 2020. [Online]. Available: https://science.howstuffworks.com/satellite.htm

[37] Wikipedia, "Space launch market competition," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Space_launch_market_competition

[38] Chris Daehnick, Isabelle Klinghoffer, Ben Maritz, and Bill Wiseman, "Large leo satellite constellations: Will it be different this time?" 2020. [Online]. Available: https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/large-leo-satellite-constellations-will-it-be-different-this-time

[39] Ian D. Chakeres, Elizabeth M. Belding-Royer, "Aodv routing protocol implementation design."

[40] David B. Johnson, David A. Maltz, Josh Broch, "Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks."

[41] Yufei Cheng, Egemen K. Cetinkaya, and James P.G. Sterbenz, "Dynamic source routing (dsr) protocol implementation in ns-3," pp. 367–374, 2012.

[42] Ammar Odeh, Eman AbdelFattah and Muneer Alshowkan, "Performance evaluation of aodv and dsr routing protocols in manet networks," *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 3, pp. 13–22, 2012.

[43] Thomas Clausen, Philippe Jacquet et al, "Optimized link state routing protocol (olsr)," *NetworkWorking Group*, 2003.

[44] F. S. Foundation, "Gnu general public license, version 2," 1991. [Online]. Available: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[45] NSNAM, "What is ns3," 2020. [Online]. Available: https://www.nsnam.org/about/what-is-ns-3/

[46] ——, "Ns3 installation," 2020. [Online]. Available: https://www.nsnam.org/wiki/Installation

[47] O. A. A. Elahe Fazeldehkordi, "Mobile ad-hoc network," 2016. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/mobile-ad-hoc-network

[48] Wikipedia, "Vehicular ad-hoc network," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Vehicular_ad_hoc_network

[49] NSNAM, "Manet routing comparison," 2011. [Online]. Available: https://www.nsnam.org/doxygen/manet-routing-compare_8cc_source.html

[50] P. Kumar, "Manet routing protocols using ns3," 2019. [Online]. Available: https://www.nsnam.com/2019/05/comparison-of-adhoc-routing-protocols.html

[51] F. Corrigan, "Dji mavic air 2 review of features, specs and faqs answered," 2020. [Online]. Available: https://www.dronezon.com/drone-reviews/dji-mavic-air-2-review-includes-features-specs-faqs/

[52] Wikipedia, "dbm," 2020. [Online]. Available: https://en.wikipedia.org/wiki/DBm

[53] R. S. Sunil Makaar, Yudhvir Singh, "Performance investigation of olsr and aodv routing protocols for 3d fanet environment using ns3," *Journal of Communication Engineering and Systems*, vol. 8, pp. 1–10, 2018.