ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη του VEINS framework και υλοποίηση αλγορίθμων ομαδοποίησης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΚΟΛΤΣΙΑΚΗ ΠΑΥΛΟΥ

(ΑΕΜ:3056)

Επιβλέπων : Νικόλαος Δημόκας

Επίκουρος Καθηγητής

Καστοριά Δεκέμβριος-2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη του VEINS framework και υλοποίηση αλγορίθμων ομαδοποίησης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΚΟΛΤΣΙΑΚΗ ΠΑΥΛΟΥ

(ΑΕΜ:3056)

Επιβλέπων : Νικόλαος Δημόκας

Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18/12/2024

Νικόλαος Δημόκας	Ιωάννης Βαρδάκας	Ιωάννης Τουλόπουλος
Επίκουρος Καθηγητής	Αναπλ. Καθηγητής	Επίκουρος Καθηγητής

Καστοριά Δεκέμβριος-2024

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω πρώτα τους γονείς μου, για την αμέτρητη υποστήριξη όλα αυτά τα χρόνια φοίτησης μου και για την συνεχή προσπάθεια τους να με βοηθήσουν να πορευτώ στην ζωή μου και να χτίσω θεμέλια για το μέλλον μου.

Θα ήθελα επίσης να πω ένα μεγάλο ευχαριστώ στον κ. Νικόλαο Δημόκα όπου ως επιβλέπων καθηγητής για την εργασία μου , ήταν πάντα διαθέσιμος για ό,τι χρειάστηκα κατά την εκπόνηση της πτυχιακής εργασίας.

Περίληψη

Τα τελευταία χρόνια, παρατηρείται όλο και περισσότερο η «ενσάρκωση» του οράματος ενός τεχνολογικά ανεπτυγμένου αυτοκινήτου , όπου αντικατοπτρίζει τη φαντασίωση που είχαν όλοι ως παιδιά. Ένα αυτοκίνητο δηλαδή, με εφαρμογές και ιδιότητες παρόμοιες με αυτές ενός υπολογιστή, όπως το να επικοινωνεί με άλλα οχήματα μέσω μηνυμάτων και ειδοποιήσεων ή να επιδιώκει την πιο άμεση διάδραση με τον χρήστη-οδηγό. Έτσι, σήμερα βλέπουμε τις αυτοκινητοβιομηχανίες να φτάνουν όλο και περισσότερο σε αυτόν το στόχο , με οχήματα που προβλέπουν την επόμενη κίνηση ενός άλλου οδηγού στον αυτοκινητόδρομο, διορθώνουν τον οδηγό σε περίπτωση λανθασμένης κίνησης, παράγουν έναν πλήρες οδηγό για οποιοδήποτε ταξίδι ή ακόμα και οχήματα που οδηγούν μόνα τους, χωρίς την παρέμβαση του ίδιο του οδηγού.

Η εργασία αυτή, «αγγίζει» ένα τέτοιο όραμα, ένα αυτοκίνητο που βρίσκεται σε άμεση επικοινωνία με τα υπόλοιπα αμάξια σε μια δεδομένη ακτίνα, αλλά και με τον οδηγό του. Πολλοί ερευνητές έχουν ήδη ξεκινήσει να δημιουργούν προσομοιώσεις σε υπολογιστικό περιβάλλον, στη προσπάθεια να διατυπώσουν αλγορίθμους που επιτυγχάνουν την επικοινωνία που θέλουμε. Το όνομα που χαρακτηρίζει την υλοποίηση αυτής της τεχνολογίας είναι VANETs ή Vehicular ad hoc networks. Τα VANETs στηρίζονται πάνω στα ήδη υπάρχοντα ad hoc networks και συγκεκριμένα στα ασύρματα (MANETs).

Ένα εργαλείο που μπορεί να ενσαρκώσει μια τέτοια ιδέα σε πειραματικό τουλάχιστον επίπεδο και αποτελεί και το πρωτεύων θέμα της εργασίας είναι το VEINS (Vehicles in Network Simulation) framework για το οποίο ακολουθεί μία εκτενής ανάλυση του αλλά και παρουσιάσεις παραδειγμάτων προσομοιώσεων. Παράλληλα θα αναπτυχθεί αλγόριθμος ομαδοποίησης (clustering) βασισμένο πάνω στο framework δίνοντας έτσι μια ολοκληρωμένη ιδέα για το πως αυτές οι δύο θεματικές της εργασίας μπορούν να φανούν χρήσιμες σε ένα υποθετικό σενάριο.

***Λέξεις κλειδιά*: VEINS, framework, VANETs, ad hoc, car, cluster**

Abstract

In recent years, there is an increasing "embodiment" of the vision of a technologically advanced car, which reflects the fantasy that everyone had as a child. That is, a car, with applications and properties similar to those of a computer, such as communicating with other vehicles through messages and notifications or the most direct interaction with the user-driver. So today we see the car industry increasingly reaching this goal, with vehicles that predict the next move of another driver on the highway, correct the driver in case of wrong movement, generate a complete guide for any trip or even vehicles that drive on their own, without the intervention of the driver himself.

This work approaches such a vision, a car that is in direct contact with other cars in a given radius, but also with its driver. Many researchers have already begun to create simulations in a computing environment, in an attempt to formulate algorithms that achieve the communication we want. The name that characterizes the implementation of this technology is VANETs or Vehicular ad hoc networks. VANETs rely on existing ad hoc networks and specifically on wireless (MANETs).

One of the tools used to depicture hypothetical scenarios that correspond to the forementioned topics and is the main topic of the current thesis is the VEINS (Vehicles in Network Simulation) framework for which an extensive analysis follows together with presentations of simulation examples. At the same time, a clustering algorithm based on the framework will be developed trying to prove how these two topics can merge into a useful project.

***Key words*: VEINS, framework, VANETs, ad hoc, car, cluster**

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Λίστα τμημάτων κώδικα

1. Εισαγωγή

Η εργασία παρουσιάζει την εφαρμογή της τεχνολογίας των τυχαίων δικτύων οχημάτων (VANETs) στην πραγματική ζωή. Ασχολείται κυρίως με το ευρύ περιεχόμενο που αφορά την αρχή των δικτύων, το πώς αυτά εξελίχθηκαν και πως η ιδέα των δικτύων οχημάτων υλοποιείται στην πραγματικότητα. Επίσης, ένα μεγάλο μέρος της εργασίας αποτελείται από την λεπτομερή ανάλυση του VEINS framework, το οποίο είναι και το πρωτεύον στοιχείο και θέμα που ασχολείται η εργασία. Γίνεται επεξήγηση για την πλήρη κατανόηση του framework και για την διευκόλυνση πλοήγησης του χρήστη στο περιβάλλον του. Στην εργασία εμπεριέχεται και υλοποίηση παραδείγματος με αλγόριθμους ομαδοποίησης κάνοντας χρήση του VEINS αλλά και άλλων εφαρμογών όπως το OMNET++, το οποίο απεικονίζει μέσω προσομοίωσης την επικοινωνία που έχουν τυχαία οχήματα μέσα σε ένα οποιοδήποτε οδικό δίκτυο. Αναλυτικά, το δεύτερο κεφάλαιο της εργασίας περιλαμβάνει μια εκτενή αναφορά στα Ad hoc δίκτυα, στους τύπους δικτύων που συναντάμε, αναφορές σε περιπτώσεις εφαρμογής των δικτύων όπως επίσης και στα πρωτόκολλα που συνθέτουν αυτά τα δίκτυα. Στο τρίτο κεφάλαιο, γίνεται η εισαγωγή συγκεκριμένα για τον τύπου Ad hoc δικτύων με όνομα VANET, όπου ακολουθεί λεπτομερής ανάλυση για την αρχή των δικτύων αυτών, για τους τρόπους που αυτή η τεχνολογία πήρε μορφή και για τις περιπτώσεις όπου αυτά τα δίκτυα θα μπορούσαν να είναι μέρος της καθημερινότητας μας. Το τέταρτο κεφάλαιο, αφορά καθαρά το κυριότερο θέμα της εργασίας, το VEINS. Αναγράφονται πληροφορίες που καλύπτουν πλήρως τη χρησιμότητα του framework, τον τρόπο που υλοποιείται τεχνικά και την σύνδεση του με άλλα προγράμματα όπως το OMNET++, όπως και την πλοήγηση σε αυτά τα προγράμματα από την μεριά του χρήστη. Τέλος, στο πέμπτο και τελευταίο κεφάλαιο, γίνεται μια σύντομη αναφορά για τους αλγόριθμους ομαδοποίησης και ακολουθεί η υλοποίηση παραδείγματος βάση αυτών και του VEINS.

2. Ad hoc δίκτυα

Η φράση ad hoc είναι λατινικής προέλευσης και σημαίνει κυριολεκτικά «σε αυτό». Στην αγγλική διάλεκτο, χρησιμοποιείται συνήθως για να περιγράψει μια λύση για ένα συγκεκριμένο ζήτημα και όχι για μια γενικευμένη λύση βασισμένη σε παράπλευρους παράγοντες [2].

Σχετικά με τα ad hoc δίκτυα, ή αλλιώς αυτοοργανωμένα δίκτυα ή δίκτυα κατ' απαίτηση, πρόκειται για έναν αποκεντρωμένο τύπο ασύρματου δικτύου. Αυτό σημαίνει πως δεν βασίζεται σε κάποια προϋπάρχουσα υποδομή, δηλαδή δεν είναι αναγκαία η ύπαρξη δρομολογητών ή άλλων παραγόντων. Αντιθέτως , κάθε κόμβος λειτουργεί ανεξάρτητα , προωθεί πακέτα προς όλους τους άλλους κόμβους και ο καθορισμός των κόμβων που θα προωθούν τα δεδομένα γίνεται δυναμικά με βάση τη συνδεσιμότητα του δικτύου [1].

Εικόνα 1 Προεπισκόπηση ad hoc δικτύων

Πηγή: https://www.researchgate.net/figure/Wireless-Ad-Hoc-Network_fig3_220487867

2.1 Ανασκόπηση

Το παλαιότερο ασύρματο δίκτυο δεδομένων ονομαζόταν PRNET ή ραδιοδίκτυο πακέτων, και χρηματοδοτήθηκε από την Defense Advanced Research Projects Agency (DARPA) στις αρχές της δεκαετίας του 1970. Οι Bolt, Beranek and Newman Inc. (BBN) και η SRI International σχεδίασαν, κατασκεύασαν και πειραματίστηκαν με αυτά τα πρώτα συστήματα. Η ομάδα περιλάμβανε τους Robert Kahn, Jerry Burchfiel και Ray Tomlinson. Παρόμοια πειράματα έγιναν και στην ραδιοερασιτεχνική κοινότητα με το πρωτόκολλο

x25. Αυτά τα πρώιμα συστήματα ραδιοφώνου πακέτων προϋπήρχαν του Διαδικτύου και πράγματι ήταν μέρος του κινήτρου της αρχικής σουίτας πρωτοκόλλου Διαδικτύου. Αργότερα πειράματα της DARPA(Υπηρεσία Έρευνας Προηγμένων Αμυντικών Προγραμμάτων) περιελάμβαναν το έργο Survivable Radio Network (SURAN), που έλαβε χώρα τη δεκαετία του 1980. Ένας διάδοχος αυτών των συστημάτων παρουσιάστηκε στα μέσα της δεκαετίας του 1990 για τον αμερικανικό στρατό, και αργότερα για περισσότερα έθνη, ως αρχικό πλαίσιο για το ψηφιακό ραδιόφωνο [3].

2.2 Τύποι Ad hoc δικτύων

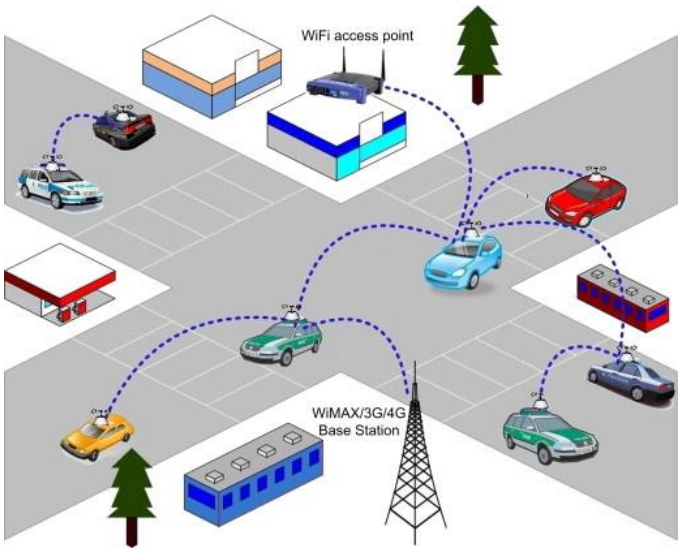
Με την πάροδο του χρόνου, τα ad hoc δίκτυα άρχισαν να εξαπλώνονται και να εφαρμόζονται σε περισσότερους τομείς της τεχνολογίας. Το γεγονός αυτό μας οδήγησε στην δημιουργία διαφορετικών ομάδων δικτύων που εξυπηρετούν αναλόγως τον εκάστοτε τομέα αλλά διατηρούν την εύχρηστη και πρακτική φύση των ad hoc δικτύων σε οποιασδήποτε μορφής περίσταση.

2.2.1 Ασύρματο Ad hoc δίκτυο

Ίσως η πιο γνωστή κατηγορία χρήσης δικτύων ad hoc , είναι σε κινητές συσκευές (smartphones, tablets κλπ.). Τα ασύρματα ad hoc δίκτυα (MANETs) , είναι πρακτικά «άυλα», κοινώς δεν χρειάζονται στοιχεία όπως καλώδια και δρομολογητές για να λειτουργήσουν.
Η κατηγορία αυτή , αποτελεί την πιο βασική μεταξύ των άλλων, κοινώς κάθε άλλη κατηγορία βασίζεται σε ένα ασύρματο ad hoc δίκτυο. Οποιοδήποτε παράδειγμα υποθεί περαιτέρω λοιπόν αποτελεί ένα ασύρματο ad hoc δίκτυο.

2.2.2 Vehicular Ad hoc network

Χρησιμοποιούνται για την επικοινωνία μεταξύ τον οχημάτων σε οποιοδήποτε οδικό περιβάλλον. Τα έξυπνα Vehicular ad hoc networks ή Intelligent VANETs προσφέρουν πληροφόρηση σχετικά με την κατάσταση που μπορεί να επικρατεί σε ένα οδικό δίκτυο. Έτσι, μπορούν να αποφευχθούν λάθη , συμφορήσεις και ατυχήματα [4].

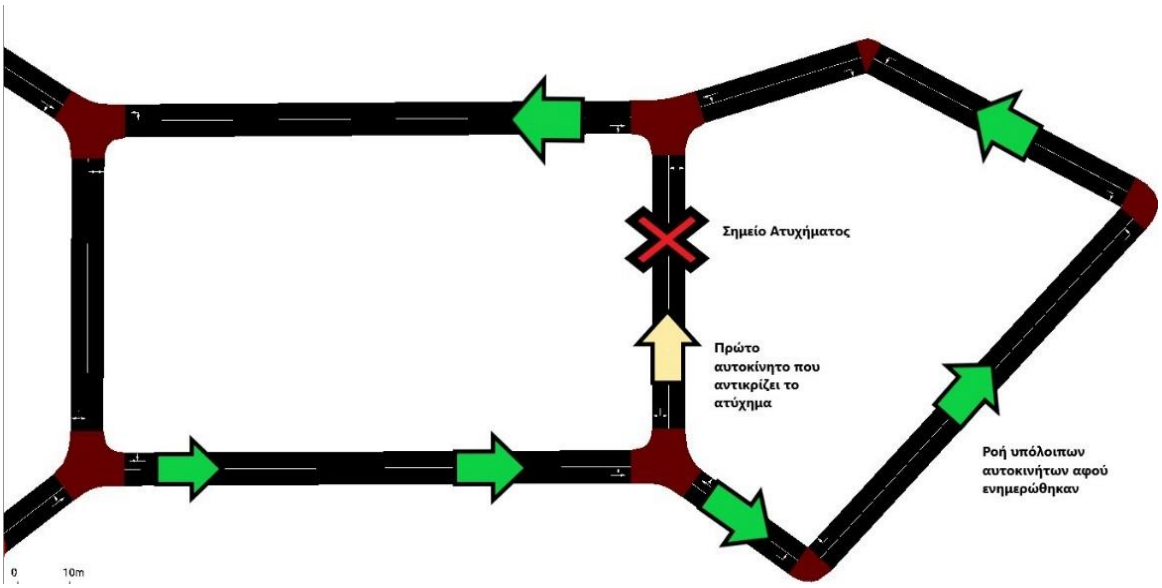


Εικόνα 2 Πρότυπο δίκτυο οχημάτων

Πηγή: https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network

Παράδειγμα:

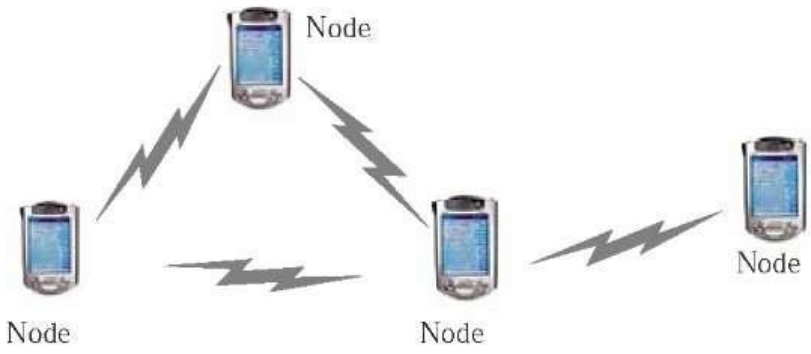
Μια κοινότυπη εφαρμογή αυτού του δικτύου μπορεί να είναι ένα σύνολο αυτοκινήτων σε ένα οδικό δίκτυο, τα οποία επικοινωνούν ασύρματα μεταξύ τους. Υποθετικά, βρίσκονται σε ένα τετράγωνο δρόμων μιας πόλης. Σε ένα συγκεκριμένο σημείο του τετραγώνου έχει συμβεί ένα ατύχημα που περιλαμβάνει δύο αυτοκίνητα, με αποτέλεσμα η ροή του αναφερόμενου δρόμου να είναι μπλοκαρισμένη. Με την χρήση ενός δικτύου επικοινωνίας , το αμάξι που θα αντικρίσει πρώτο τον μπλοκαρισμένο δρόμο, θα μπορέσει να αποστείλει αυτόματα μήνυμα-ειδοποίηση, στα υπόλοιπα αυτοκίνητα που θεωρούνται υποψήφια να διασχίσουν τον δρόμο, πως υπάρχει κάποιο εμπόδιο, ώστε να προλάβουν να αλλάξουν σύντομα την διαδρομή τους. Έτσι θα αποφευχθεί σε μεγάλο βαθμό η συμφόρησή των δρόμων και η ταλαιπωρία όλων τον οδηγών. Προαιρετικά, το ασύρματο αυτό δίκτυο μπορεί να ανταλλάζει μηνύματα και με το δίκτυο που παρέχει πληροφορίες στον οδηγό σχετικά με την βέλτιστη διαδρομή προς έναν προορισμό. Ως αποτέλεσμα , με την λήψη του προειδοποιητικού μηνύματος αλλαγής κατεύθυνσης, αμέσως μπορούν να αλλάζουν και οι οδηγίες για την επόμενη βέλτιστη διαδρομή του προεπιλεγμένου προορισμού που έχει ορίσει ο οδηγός. (βλ. Κεφ. 3 για περισσότερες πληροφορίες)



Εικόνα 3 Παράδειγμα στο περιβάλλον του SUMO

2.2.3 Smartphone Ad hoc network

Μία ακόμη κατηγορία που είναι ευρέως γνωστή , είναι τα ad hoc δίκτυα για smartphones ή αλλιώς SPANs. Ένα απλό παράδειγμα αποτελεί η ασύρματη επικοινωνία μεταξύ συσκευών με τεχνολογίες όπως Bluetooth και Wi-Fi [4].



Εικόνα 4 SPAN δίκτυο

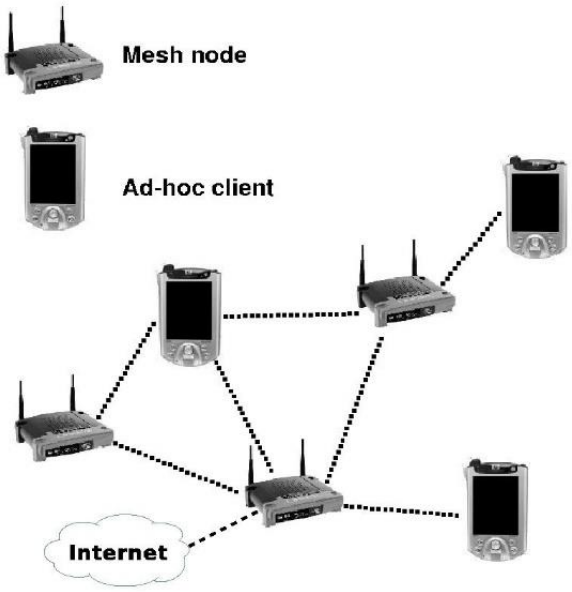
Πηγή:https://what-when-how.com/information-science-and-technology/wireless-ad-hoc-networking-information-science/

Παράδειγμα:

Μια συσκευή που λειτουργεί αμιγώς ασύρματα ή ακόμα και αν έχει την επιλογή της ασύρματης λειτουργίας, είναι σύνηθες να χρησιμοποιεί ένα εκ των 2 τεχνολογιών τουλάχιστον , Bluetooth ή Wi-Fi. Μια αντίστοιχη συσκευή μπορεί να είναι ένα σετ ασύρματων ακουστικών για παράδειγμα. Τα ακουστικά για να συνδεθούν με ένα κινητό χρησιμοποιούν το Bluetooth. Η σύνδεση γίνεται μέσα σε λίγα δευτερόλεπτα και η ανταλλαγή δεδομένων είναι άμεση, ωστόσο η απόσταση που μπορεί να απέχουν μεταξύ τους οι δύο συσκευές δεν μπορεί να είναι μεγάλη (παραπάνω από 10 μέτρα), διότι η εμβέλεια που καλύπτει ένα σήμα Bluetooth είναι περιορισμένη. Βέβαια, μια τόσο διαχρονική τεχνολογία δεν παύει να λαμβάνει συνεχής υποστήριξη και ενημερώσεις, οπότε ενδέχεται η κάθε νέα έκδοση να περιλαμβάνει παραπάνω ικανότητες. Αναλόγως μπορεί να λειτουργήσει και η συνδεσιμότητα δύο συσκευών με την χρήση Wi-Fi. Όμως, υπάρχουν 2 σημαντικές διαφορές μεταξύ των 2 τεχνολογιών. Για αρχή , όπως αναφέρθηκε και πριν, το Bluetooth μπορεί να εκπέμπει σε ένα αρκετά περιορισμένο μήκος, αντίθετα με το Wi-FI που το εύρος αποστολής ραδιοκυμάτων είναι σαφέστατα πολύ μεγαλύτερο. Δεύτερη μεγάλη διαφορά, είναι το πλήθος συσκευών που μπορούν να συνδεθούν σε μια κεντρική συσκευή-κόμβο. Το Bluetooth επιτρέπει αυστηρά μια και μόνο συσκευή την φορά, ενώ το Wi-Fi μπορεί εύκολα να διαχειριστεί παραπάνω από μία συσκευή. Σε γενικές γραμμές , η εμπειρία παραμένει ίδια και στις δύο περιπτώσεις.

2.2.4 Wireless mesh network

Ένα Wireless mesh network ή Ασύρματο Δίκτυο Πλέγματος, αποτελεί έναν τύπο ad hoc δικτύου , όπου όλες οι συσκευές επικοινωνούν άμεσα μεταξύ τους και έπειτα αποστέλλουν πακέτα προς το Διαδίκτυο [4]. Σε ένα δίκτυο πλέγματος, ο αριθμός των δρομολογητών μπορεί να είναι περισσότερο από έναν (κάθε συσκευή που είναι συνδεδεμένη στο δίκτυο μπορεί να έχει και τον ρόλο του δρομολογητή), εξασφαλίζοντας την ακεραιότητα των πακέτων και τη βεβαιότητα αποστολής τους.



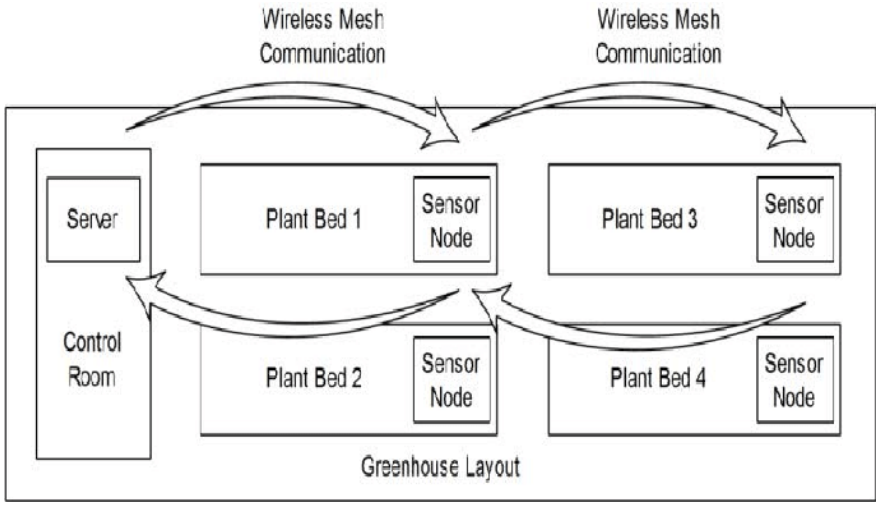
Εικόνα 5 Δίκτυο Mesh

Πηγή:https://www.researchgate.net/figure/An-overview-of-a-network-consisting-of-mesh-nodes-and-mobile-ad-hoc-clients_fig1_38104476

Παράδειγμα:

Τα ασύρματα δίκτυα πλέγματος τελευταία συσχετίζονται πάρα πολύ με τον όρο IoT (Internet of Things). Ο λόγος είναι επειδή μπορούν να διευκολύνουν τις συσκευές που εμπλέκονται στο δίκτυο να μεταφέρουν πακέτα η μία στην άλλη και έπειτα στην βάση. Σε ένα υποθετικό σενάριο υπάρχει ένα δίκτυο πλέγματος το οποίο ελέγχει πλήρως τα επίπεδα θερμοκρασίας και υγρασίας σε ένα θερμοκήπιο αλλά ακόμη, ειδοποιεί τους διαχειριστές για τις ανάγκες των φυτών μέσα σε αυτό. Αυτό επιτυγχάνεται μέσω μεμονωμένων αισθητήρων που είναι συνδεδεμένοι μεταξύ τους αλλά και με μία βάση δεδομένων που συλλέγει τις τιμές από την οποία οι εργαζόμενοι μπορούν να κρίνουν κατάλληλα τις ανάγκες του θερμοκηπίου. Ο κάθε αισθητήρας είναι εξοπλισμένος με τις απαραίτητες τεχνολογίες που τον καθιστούν ανεξάρτητο από τους υπόλοιπους έχοντας έτσι και τον ρόλο ενός κόμβου. Συνεπώς, αφού ο κάθε αισθητήρας μπορεί να πραγματοποιήσει τις δικές του μετρήσεις , έπειτα μπορεί να στείλει τα αποτελέσματα στους υπόλοιπους κόμβους-αισθητήρες ώστε να συγκριθούν οι τιμές και να προκύψει ένα μέσο νούμερο (πχ. θερμοκρασίας) το οποίο μετά θα επιστρέψει στην βάση για να μελετηθεί από τους εργαζόμενους. Όλες αυτές οι συναλλαγές πακέτων μεταξύ κόμβων καθορίζουν ένα δίκτυο πλέγματος. Η

παρακάτω απεικόνιση αναπαριστά ένα μικρού μεγέθους αλλά ολοκληρωμένο σύστημα που εξηγήθηκε παραπάνω.



Εικόνα 6 Παράδειγμα Mesh

Πηγή: https://www.researchgate.net/figure/Wireless-mesh-network-implementation_fig1_316907438

2.3 Πρωτόκολλα

Τα ad hoc δίκτυα είναι ένα είδος δικτύων που παρατηρείται πως παρέχει αρκετά προνόμια ως προς την εγκατάσταση ενός πλήρους δικτύου. Στην ενότητα 2.2 αποδεικνύεται πως ένα δίκτυο ad hoc μπορεί να είναι ταχύ, άμεσο, ασφαλές και ίσως φτηνό για χρήση. Ακόμα, φαίνεται πως ένα δίκτυο ad hoc κατέχει το προνόμιο της φορητότητας και της ελαστικότητας όταν στήνεται εκ νέου, με λιγότερες παραμέτρους υπόψιν και λιγότερες ανάγκες που αφορούν την δόμηση του, αν σκεφτούμε πως μπορεί να μην χρειαστεί τίποτε άλλο παρά κόμβους επικοινωνίας για να λειτουργήσει. Αυτό όμως που παραμένει σημαντικό είναι οι ανάγκες για σωστή δρομολόγηση και πρακτικότητα. Χρειάζονται λοιπόν τα κατάλληλα πρωτόκολλα προκειμένου να διασφαλιστεί η ποιοτικότερη και σαφώς η πιο άμεση δρομολόγηση για κάθε ένα σενάριο ξεχωριστά. Έτσι, πλέον έχουν παρουσιαστεί ολόκληρες ομάδες από πρωτόκολλα, κάθε μία με συγκεκριμένη λειτουργία, που επικεντρώνονται στην ορθή δρομολόγηση που θα χρειαστούν τα πακέτα σε ένα ad hoc δίκτυο. Περιληπτικά ακολουθούν μερικές κατηγορίες:

Reactive (on-demand): Τα πρωτόκολλα επιτρέπουν στον κόμβο να αναζητήσει δρομολόγιο αποστολής πακέτων όταν υπάρχει διαθέσιμη πληροφορία για αποστολή. Π.χ. DSR, AODV [5].

Proactive (table-driven): Τα πρωτόκολλα επιτρέπουν στους κόμβους να κατέχουν από την αρχή προσχεδιασμένα δρομολόγια για όλο το δίκτυο. Π.χ. DSDV, OLSR.

Hybrid: Σε αυτή την κατηγορία, οι ιδιότητες των δύο προηγούμενων κατηγοριών πρωτόκολλων συνδυάζονται σε μία ολοκληρωμένη ομάδα. Π.χ. ZRP, ZHLS.

Hierarchical: Η δόμηση των κόμβων χωρίζεται σε clusters μέσω ιεραρχικού αλγόριθμου ομαδοποίησης, με σκοπό να ρυθμιστεί με ανάλογο τρόπο η τοπολογία ενός δικτύου. Π.χ. CEDAR, HSR.

Multipath: Τα πρωτόκολλα στοχεύουν κυρίως στην εξασφάλιση μιας άριστης ποιότητας υλικού και λογισμικού που αφορούν την δρομολόγηση. Επιλύουν ζητήματα που αφορούν τον πλεονασμό κατά την εύρεση ταχύτερου δρομολογίου επικοινωνίας. Π.χ. CHAMP, secMR.

2.4 Κίνδυνοι-Ασφάλεια

Η αναγνώριση και κατανόηση των κινδύνων που υπάρχουν πλέον, μας φέρνουν πιο κοντά στην αντιμετώπιση τους. Οι επιθέσεις ή αλλιώς κυβερνοεπιθέσεις αποτελούν μείζων ζήτημα για τις υποδομές δικτύων που μπορεί να καλύπτουν μια εταιρεία, έναν οργανισμό ακόμα και το σπίτι μας. Παράλληλα, οι ειδικευόμενοι μηχανικοί-τεχνικοί που αφιερώνουν την επαγγελματική τους καριέρα στην αντιμετώπιση και εξάλειψη των απειλών αυτών, πληθαίνουν, δημιουργώντας έτσι μια αλυσιδωτή αντίδραση μεταξύ της εμφάνισης νέων απειλών και πιο έμπειρων τεχνικών ασφάλειας. Πιο συγκεκριμένα όμως στο πλαίσιο ανάπτυξης ενός απόρθητου δικτύου ad hoc , οι απειλές δεν διαφέρουν από αυτές που υπάρχουν συνήθως σε ένα πιο γνώριμο δίκτυο, όπως για παράδειγμα η υποκλοπή πληροφοριών που στέλνονται μεταξύ ενός συνόλου από κόμβους. Οι επιθέσεις μπορούν να χωριστούν σε δύο βασικές κατηγορίες, τις παθητικές και τις ενεργητικές. Ακολουθούν μερικές από τις περιπτώσεις τις κάθε κατηγορίας.

2.4.1 Passive Attack

Η παθητική επίθεση αφορά έναν εισβολέα που πραγματοποιεί μόνο ενέργειες παρακολούθησης ώστε να αποκτήσει πληροφορίες που επιθυμεί. Κατά την διεκπεραίωση, δεν «προσβάλλει», έναν δρομολογητή για παράδειγμα, εισάγοντας ψευδής πληροφορίες για να αποκτήσει πρόσβαση [6].

- **Eavesdropping:** Η συγκεκριμένη τεχνική υποδηλώνει την εισβολή ενός κακόβουλου χρήστη σε ένα δίκτυο με ελλιπής ασφάλεια. Ο χρήστης αποκτά πρόσβαση στην ανταλλαγή πακέτων μεταξύ κόμβων και

αποκτά πληροφορίες όπως κλειδιά ασφαλείας, κωδικούς κ.λπ. Όπως μαρτυρά και η ονομασία του όρου, η επίθεση είναι αντίστοιχη ενός πραγματικού σεναρίου όπου ένας τρίτος ακούει την συνομιλία δύο ατόμων χωρίς την έγκρισή τους.

- **Traffic Analysis:** Αυτή η τεχνική παραπέμπει στην προαναφερόμενη (Eavesdropping) μόνο που αυτή τη φορά πρόκειται για κάτι πιο εξειδικευμένο. Με την εκτέλεση της επίθεσης, ο κακόβουλος χρήστης έχει πρόσβαση πάλι σε όλα τα μηνύματα που ανταλλάσσονται, ακόμα και αν αυτά είναι κρυπτογραφημένα. Επιπλέον, μπορεί να εξετάσει μηνύματα που παρουσιάζουν ένα συγκεκριμένο μοτίβο ώστε να συμπεράνει ένα αποτέλεσμα για ένα απόρρητο θέμα.

2.4.2 Active Attack

Αντίθετα με μια παθητική επίθεση που δεν αλλοιώνει την επίδοση ενός δικτύου, μια ενεργητική επίθεση προσβάλλει άμεσα την ροή δεδομένων κατά την μεταφορά αλλά και την λειτουργία των κόμβων.

- **Flooding attack:** Σε ένα Flooding attack, ο θύτης εξαντλεί τους πόρους ενός δικτύου χρησιμοποιώντας το bandwidth ή την επεξεργαστική ισχύ, με αποτέλεσμα να μειώνεται αισθητά η απόδοση του. Για παράδειγμα, στο πρωτόκολλο AODV, ένας κακόβουλος κόμβος στέλνει πολλαπλά RREQs (Route Request messages) σε έναν άλλο που δεν υπάρχει στο δίκτυο. Συνεπώς, από την στιγμή που δεν θα επιστρέψει κάποιο RREP (Route Reply message), τα RREQs θα σταλούν σε όλους τους υπόλοιπους κόμβους. Έτσι, προκαλείται συμφόρηση με την εκμετάλλευση των πόρων που μπορεί να οδηγήσει και σε DoS [6].

- **Black Hole Attack:** Η διαδικασία εύρεσης διαδρομής στο πρωτόκολλο AODV ξεκινά με την εκλογή ενός κόμβου, ο οποίος ανταποκρίνεται σε ένα RREQ, που υπόσχεται την μικρότερη διαδρομή για τον προορισμό που θέλουμε. Όμως αυτός ο μηχανισμός μπορεί να αλλοτριωθεί από έναν ψεύτικο κακόβουλο κόμβο. Ο κόμβος αυτός θα υποσχεθεί πως έχει το μικρότερο routing delay στο δίκτυο, ακόμα και αν η διαδρομή είναι ανύπαρκτη, με συνέπεια να ανακατευθυνθούν λανθασμένα τα πακέτα προς αυτόν.

- **Wormhole attack:** Η επίθεση αυτή αποτελεί μεγάλη απειλή για ένα δίκτυο, διότι μπορεί να το βλάψει χωρίς καν να έχει αναγνωρίσει την τοπολογία. Εν συντομία, ένας κακόβουλος χρήστης που έχει πρόσβαση στο δίκτυο, λαμβάνει πακέτα σε έναν κόμβο του δικτύου, τα κατευθύνει σε έναν άλλο, ο οποίος πολλές φορές δεν ανήκει στο δίκτυο, και τα αναπαράγει από αυτόν.

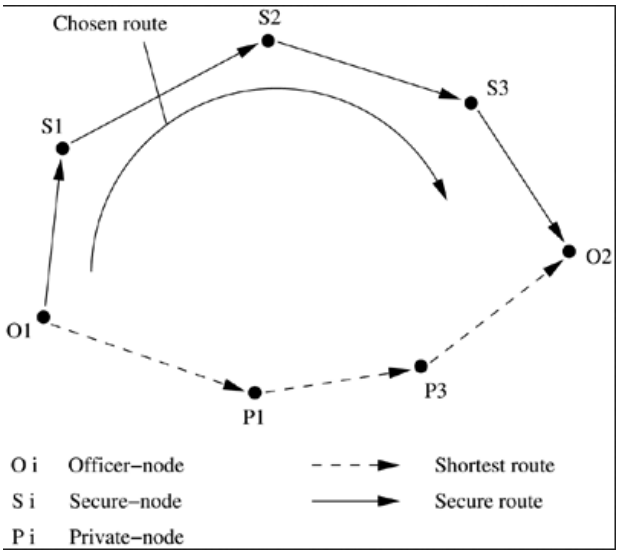
2.4.3 Θωράκιση των Ad hoc

Περιληπτικά, μπορούμε να βασιστούμε πάνω σε πέντε παράγοντες με τους οποίους μπορεί να επιτευχθεί η άρτια ασφάλεια των δικτύων ad hoc.

- ✓ **Διαθεσιμότητα:** Διασφάλιση του δικτύου από επιθέσεις DoS σε όλα τα επίπεδα [7].
- ✓ **Εμπιστευτικότητα:** Η μεταφορά πληροφοριών πρέπει να έχει ως προορισμό μόνο τους επικυρωμένους χρήστες.
- ✓ **Ακεραιότητα:** Τα μηνύματα που μεταφέρονται πρέπει να καταλήγουν στον προορισμό τους αναλλοίωτα.
- ✓ **Αυθεντικοποίηση:** Οι χρήστες που ανταλλάζουν δεδομένα πρέπει να είναι πιστοποιημένοι από το σύστημα ασφαλείας προς αποφυγήν εξαπατήσεων.
- ✓ **Αναγνώριση:** Η πηγή από την οποία προήλθε ένα μήνυμα πρέπει να είναι πάντα έγκυρη.

Ωστόσο, τα παραπάνω μέτρα δεν μπορούν να είναι απόλυτα επαρκή, για αυτό και εφαρμόζονται και κάποιες επιπλέον αρχές. Η πρώτη είναι η τεχνική της εφεδρείας, με σκοπό να υπάρχει δίοδος αποφυγής για τυχόν καταστροφές κατά την δρομολόγηση, πχ παραπάνω διαδρομές ή εφεδρικοί δρομολογητές. Η δεύτερη αρχή είναι η κατανομή της εμπιστοσύνης. Σε ένα υποθετικό σύνολο από κόμβους, είναι προτιμότερο να διαμοιράσουμε τα ποσοστά εμπιστοσύνης σε όλους τους κόμβους, παρά σε ένα μεμονωμένα, αν είναι δεδομένο πως έστω ένας κόμβος αξίζει την εμπιστοσύνη μας.

Παράλληλα, ένας άλλος τρόπος επίτευξης υψηλής ασφάλειας σε ένα δίκτυο είναι η ασφαλής δρομολόγηση. Σχετικά με την διαθεσιμότητα που προαναφέρθηκε, χρειάζονται κάποια πρωτόκολλα δρομολόγησης, τα οποία πρέπει να είναι ικανά να ανταπεξέλθουν στις εναλλαγές της δρομολόγησης αλλά και σε πιθανές επιθέσεις. Βέβαια, τα πρωτόκολλα αυτά βρίσκονται ακόμα σε πλαίσιο έρευνας οπότε δεν υπάρχει εγγύηση στην αντιμετώπιση κακόβουλων εισβολών.



Εικόνα 7 Ασφαλής Δρομολόγηση

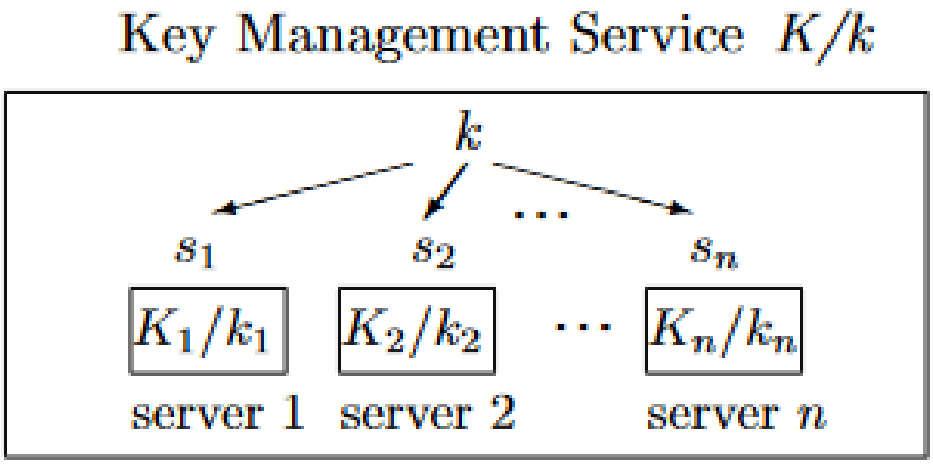
Πηγή: https://www.informit.com/articles/article.aspx?p=361984&seqNum=12

Επιπροσθέτως, ένα εξίσου σημαντικό μέτρο ασφαλείας είναι και η οργάνωση μια υπηρεσίας διαχείρισης κλειδιών ασφαλείας. Αυτό το σύστημα χρησιμοποιεί δύο βασικά στοιχεία πιστοποίησης, ένα δημόσιο κλειδί και ένα ιδιωτικό, με το ιδιωτικό να είναι γνωστό μόνο στον κόμβο που ανήκει ενώ το δημόσιο σε όλους τους κόμβους του δικτύου. Για την διαχείριση αυτών των κλειδιών έχει παρουσιαστεί μια οντότητα με όνομα Αρχή Πιστοποίησης. Η ΑΠ δεσμεύει τον κάθε κόμβο με ένα δημόσιο κλειδί υπογράφοντας ένα πιστοποιητικό που εγκαθιδρύει τις συνδέσεις μεταξύ των κόμβων. Η ΑΠ επίσης, πρέπει υποχρεωτικά να βρίσκεται διαθέσιμη, σε περίπτωση που συμβεί κάποια αλλαγή στους κόμβους. Σε περίπτωση που ένας κόμβος αποδειχθεί εχθρικός ή αποχωρήσει από το δίκτυο, η ΑΠ πρέπει να πάρει πίσω το κλειδί που δέσμευσε σε αυτόν. Μια επιπόλαια χρήση αυτής της οντότητας, είναι ικανή να οδηγήσει σε καταστροφικές επιπτώσεις με τον επιτιθέμενο να έχει πρόσβαση σε όλα τα κλειδιά του δικτύου.

Ακολουθεί παράδειγμα εφαρμογής του συστήματος:

Η υπηρεσία διαχείρισης κλειδιών εφαρμόζεται σε ένα ασύγχρονο δίκτυο ad hoc. Αυτό σημαίνει πως έχουμε ένα δίκτυο χωρίς δεσμεύσεις στην αποστολή μηνυμάτων αλλά και στην επεξεργασία τους. Επιπλέον, γνωρίζουμε πως το δίκτυο κατέχει έγκυρες διασυνδέσεις μεταξύ κόμβων. Η υπηρεσία, ως σύνολο, έχει ένα ζευγάρι με δημόσιο/ιδιωτικό κλειδί. Όλοι οι κόμβοι στο σύστημα γνωρίζουν το δημόσιο κλειδί της υπηρεσίας και εμπιστεύονται οποιοδήποτε υπογεγραμμένο πιστοποιητικό με το ανάλογο ιδιωτικό κλειδί. Οι κόμβοι λειτουργούν ως χρήστες, μπορούν να υποβάλλουν ερωτήματα (queries) για την

απόκτηση δημοσίων κλειδιών ή για ενημέρωση των δικών τους κλειδιών. Εσωτερικά στο δίκτυο υπάρχει και ένας αριθμός κόμβων που ονομάζονται εξυπηρετητές (servers). Ο κάθε εξυπηρετητής κατέχει το δικό του ζευγάρι κλειδιών και αποθηκεύουν όλα τα δημόσια κλειδιά των υπόλοιπων κόμβων, αλλά ακόμη γνωρίζουν και τα κλειδιά όλων των υπόλοιπων εξυπηρετητών δημιουργώντας ασφαλή συνδέσεις. Υποθετικά, ένας αντίπαλος μπορεί να επιτεθεί σε ένα αριθμό εξυπηρετητών σε τυχαία στιγμή για συγκεκριμένη χρονική περίοδο. Αν ένας εξυπηρετητής έχει δεχθεί επίθεση, τότε ο αντίπαλος έχει πρόσβαση σε κρυφές πληροφορίες του εξυπηρετητή. Συνήθως ένας εξυπηρετητής που έχει δεχθεί επίθεση, βρίσκεται εκτός λειτουργίας ή έχει απαρνηθεί όλα τα πρωτόκολλα που χρησιμοποιεί. Επίσης, υποθέτουμε πως ο αντίπαλος δεν έχει την απαραίτητη υπολογιστική ισχύ για να διαπεράσει τους κρυπτογραφικούς συνδυασμούς που έχουν εφαρμοστεί.



Εικόνα 8 Διαχείριση Κλειδιών

Πηγή: https://www.researchgate.net/publication/3282731_Securing_Ad_Hoc_Networks

Η ανάθεση των κλειδιών στους εξυπηρετητές γίνεται σε ζευγάρια καθολικά από το ενιαίο σύστημα διαχείρισης κλειδιών. Κάθε ένας από το πλήθος των εξυπηρετητών αποκτά ένα δημόσιο και ένα ιδιωτικό κλειδί, όπου σαφώς το δημόσιο είναι γνωστό και στους υπόλοιπους του συστήματος ενώ το ιδιωτικό όχι. Αυτή η τεχνική είναι θεμελιώδες προνόμιο για κάθε εφαρμογή μικρής ή μεγάλης κλίμακας, κυρίως σε εφαρμογές που αποκτούν επαγγελματικό ενδιαφέρον και εξυπηρετεί πολλούς πελάτες.

3. VANETs

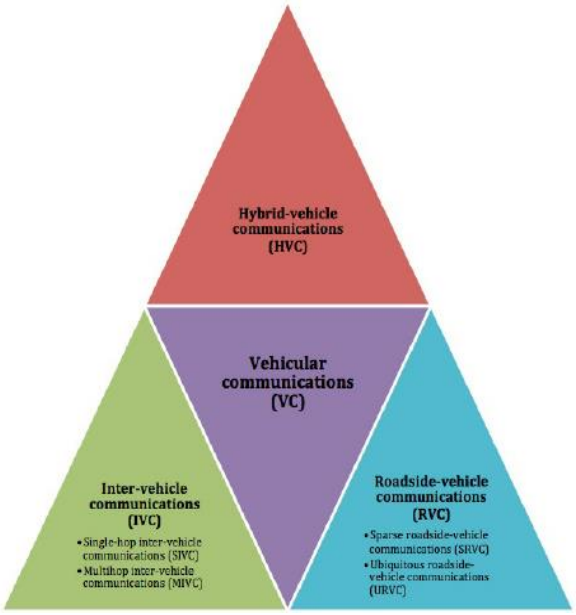
Τα οχήματα πλέον είναι εξοπλισμένα με μια ποικιλία από τεχνολογικά ευρήματα που βελτιστοποιούν σε μεγάλο βαθμό την εμπειρία της οδήγησης, χωρίς όμως να αποτελούν απρόσιτα αγαθά προς το ευρύ κοινό. Συσκευές που αναζητούν την βέλτιστη διαδρομή προς έναν προορισμό για εμάς, αισθητήρες που ανιχνεύουν τυχόν εμπόδια για λόγους ασφάλειας, περιβάλλοντα πολυμέσων για ψυχαγωγία και άλλα, είναι μόνο κάποια από τα χαρακτηριστικά που φιλοξενούν πλέον όλα τα οχήματα νεότερης εποχής. Οι συσκευές αυτές κρύβουν από πίσω τους ένα μεγάλο πλήθος αλγορίθμων που χρειάστηκαν χρόνια έρευνας και δοκιμών από μηχανικούς για να φτάσουν σε ένα ικανοποιητικό επίπεδο που θα τα κάνει έτοιμα για την αγορά.

Επιπροσθέτως, το λογισμικό αποτελεί πλέον αναπόσπαστο παράρτημα ενός οχήματος μιας και στο εσωτερικό τους δουλεύει παράλληλα μια πληθώρα από συστήματα που λειτουργούν ως βοηθοί κατά την οδήγηση. Ταυτοχρόνως, αποτελούν και σημαντικό πάτημα για όλους τους νέους χρήστες που δεν κατέχουν την απαραίτητη εξοικείωση με το όχημα τους ακόμη, διότι συμβάλλουν δραστικά στην διόρθωση λαθών που μπορούν να αποβούν μοιραία για τους ίδιους αλλά και για τρίτους. Λειτουργίες όπως το ESP (Electronic Stability Program) που διατηρεί την ισορροπία του αυτοκινήτου στον δρόμο ή το Hill Assist που κρατάει σταθερό το όχημα σε ανηφορικό επίπεδο κατά την εκκίνηση, ακόμα και ενδείξεις στο αυτοκίνητο που αφορούν την κατάσταση της μηχανής και του σασί του αυτοκινήτου μέσω επεξηγητικών γραφικών είναι ένα τμήμα μόνο από τα συστήματα που υπάρχουν στην καρδιά των εμπορικών οχημάτων.

Πολλές φορές όμως, παρουσιάζονται προβλήματα στο θέμα της απόδοσης αυτών των τεχνολογιών που εμποδίζουν τις συσκευές να αυξήσουν το επίπεδο τους. Για παράδειγμα, σε ένα GPS (Global Positioning System), μπορούμε να εμφανίσουμε γρήγορα μια τυχαία βέλτιστη διαδρομή προς έναν προορισμό σε πραγματικό χρόνο αλλά δεν έχουμε την δυνατότητα να γνωρίζουμε αν μέσα σε όλη αυτή τη διαδρομή βρίσκονται τυχόν εμπόδια που θα διακόψουν την ομαλή πορεία μας, όπως υπερβολική συμφόρηση, ή κάποιο ατύχημα που απαγορεύει να διασχίσουμε τον δρόμο. Για αυτό και πια έχει μπει στο προσκήνιο η ιδέα επικοινωνίας των ίδιων των οχημάτων μεταξύ τους σε πραγματικό χρόνο, με σκοπό την άμεση ανταλλαγή πληροφοριών που δεν μπορούν να παρέχουν άλλες ήδη υπάρχουσες συσκευές. Αυτή η εισαγωγή αφορά την υλοποίηση των Επικοινωνιών Οχημάτων ή Vehicular Communications (VC) που βασίζεται στην διατήρηση μιας άριστης επικοινωνίας σε ένα συνεχές μεταβαλλόμενο περιβάλλον δικτύου

όπου κόμβοι διαρκώς εισάγονται και διαγράφονται από την ακτίνα του σήματος μέσω της τεχνολογίας WiMAX που στηρίζεται στο πασίγνωστο Wi-Fi με την σημαντική διαφορά πως το WiMAX παρέχει τεράστια εμβέλεια σε σχέση με το Wi-Fi [27].

Οι επικοινωνίες οχημάτων βασίζεται σε δύο βασικούς πυλώνες vehicle to vehicle (IVC) και vehicle to roadside (RVC). Το IVC μπορεί να χρησιμοποιήσει είτε στρατηγική επικοινωνίας μονής μετάβασης (SICV) είτε πολλαπλής μετάβασης (MIVC). Από την άλλη, η vehicle to roadside μπορεί να χρησιμοποιεί στρατηγική επικοινωνίας URVC (Ubiquitous RVC) και SRVC (scarce RVC). Επίσης, μπορεί να υπάρχει συνδυασμός και των δύο με την ονομασία HVC (Hybrid).



Εικόνα 9 Πυραμίδα Συσχέτισης Επικοινωνιών Οχημάτων

Πηγή: https://www.researchgate.net/figure/fig1_271550130

3.1 IVC-RVC-HVC

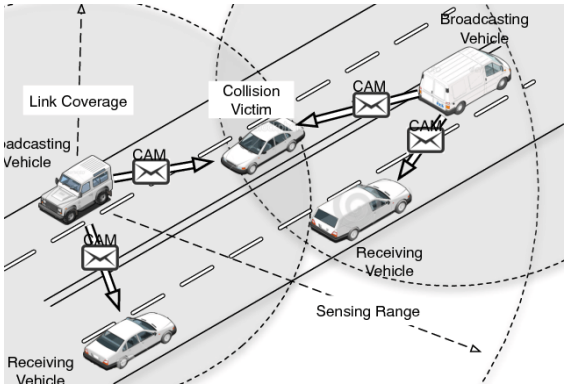
Η τεχνική IVC εφαρμόζεται απευθείας μεταξύ των οχημάτων στον δρόμο. Αυτό την καθιστά μια ασύρματη σύνδεση χωρίς υποδομές και παράλληλα είναι πολύ πιο προσιτή από μια σύνδεση RVC. Η επικοινωνία μεταξύ των κόμβων πραγματοποιείται μέσω πολλαπλής μετάβασης (multi-hop), ακόμα και με οχήματα τα οποία βρίσκονται πέραν της ακτίνας κάλυψης του σήματος. Ωστόσο, μια IVC σύνδεση μπορεί να παρέχει έναν περιορισμένο αριθμό εφαρμογών, όμως είναι προτιμότερη η χρήσης της σε εφαρμογές ασφάλειας διότι τα οχήματα μπορούν να ανιχνεύσουν άμεσα κάποιο σενάριο σύγκρουσης ή συμφόρησης

μέσα στην δεδομένη ακτίνα ισχύος. Μέσω της πολλαπλής μετάβασης, τα πορευμένα αμάξια ειδοποιούν τα υπόλοιπα που ακολουθούν να αλλάξουν διαδρομή προκειμένου να αποφύγουν κάποια ταραχή.

Αντιθέτως, το RVC προσφέρει μια μεγαλύτερη ποικιλία από εφαρμογές γιατί η πρόσβαση του στο διαδίκτυο είναι πιο σταθερή και έχει πρόσβαση σε πολύ σύντομο χρόνο σε πληροφορίες. Βέβαια, η RVC έχει δύο σημαντικά μειονεκτήματα σε περίπτωση χρήσης για λόγους ασφαλείας.

- Το κόστος αυξάνεται ραγδαία, όταν πρέπει να καλύψουμε μια μεγάλη έκταση με βάσεις, οι οποίες πρέπει εξυπηρετούν τεράστιο αριθμό αυτοκινήτων που εισέρχονται/εξέρχονται από μια δεδομένη ακτίνα ισχύος.
- Όσο αναφορά την ασφάλεια, η καθυστέρηση που θα υπάρχει για την μεταφορά πακέτων μεταξύ βάσεων, μπορεί να οδηγήσει σε καταστροφικές καταστάσεις όταν τα οχήματα δεν λαμβάνουν τα προειδοποιητικά μηνύματα στην ώρα τους.

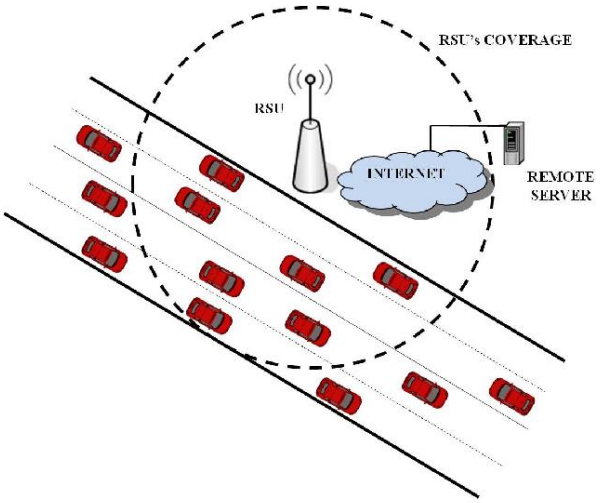
Τόσο όμως το IVC αλλά και το RVC έχουν επιθυμητά οφέλη. Αν, με την IVC οι κόμβοι μπορούν να σχηματίζουν ευέλικτες ομάδες οπουδήποτε και με την RVC, αυτοί οι κόμβοι μπορούν να έχουν γρήγορη πρόσβαση στο διαδίκτυο, τότε ο συνδυασμός αυτών των δύο σε ένα υβριδικό μοντέλο λειτουργίας μπορεί να αποφέρει μοναδικά αποτελέσματα. Το υβριδικό μοντέλο, ή αλλιώς HVC, ενώ μεγιστοποιεί τα αποτελέσματα της δικτύωσης, έχει μεγαλύτερη πολυπλοκότητα σε διάφορες πτυχές. Περιληπτικά, υπάρχει μεγάλη ανάγκη για χρήση πιο περίπλοκων πρωτοκόλλων δρομολόγησης, για ένα ισχυρό physical layer και ένα medium access layer που λειτουργεί δυναμικά για να εκμεταλλευτεί στο έπακρο τις συνδέσεις μικρής διάρκειας και να είναι ορθά οργανωμένο για να ελαχιστοποιεί τις παρεμβολές.



Εικόνα 10 Τεχνική IVC

Πηγή:https://www.researchgate.net/figure/System-model-of-Vehicle-to-Vehicle-communication-for-road-safety_fig1_262451544

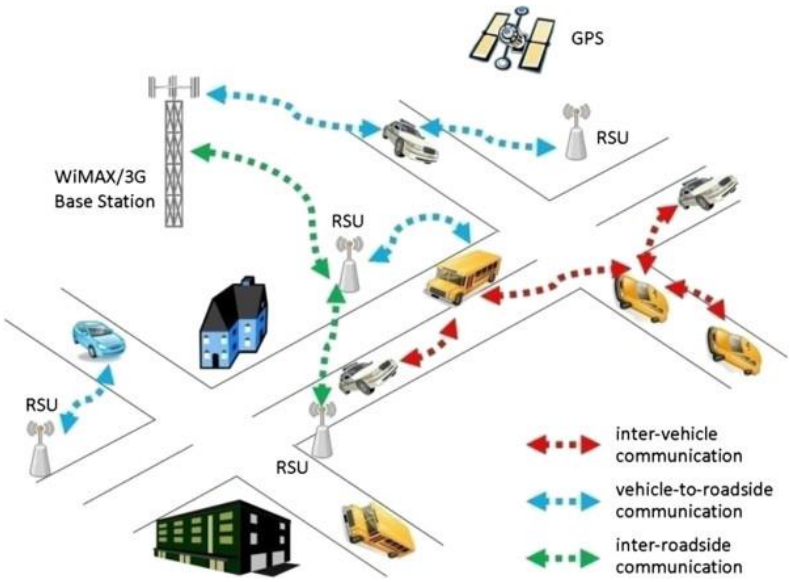
Όπως παρατηρούμε παραπάνω, το σχετικό παράδειγμα απεικονίζει ένα απλό σύστημα επικοινωνίας μερικών οχημάτων σε έναν δρόμο ταχείας κυκλοφορίας. Το κάθε όχημα εκπέμπει σήμα δεδομένης εμβέλειας που του επιτρέπει να στείλει μηνύματα στα υπόλοιπα οχήματα του δρόμου. Θεωρητικά, κάθε ένα όχημα λαμβάνει σήμα από το προπορευόμενο για τυχόν εμπόδια ή οτιδήποτε μπορεί να προξενήσει πρόβλημα στην κυκλοφορία. Το ίδιο όχημα με την σειρά του, εκπέμπει σε άλλα οχήματα.



Εικόνα 11 Τεχνική RVC

Πηγή:https://www.semanticscholar.org/paper/On-vehicle-to-roadside-communications-in-802.11p-Campolo-Molinaro/98505f29aee51302fdd2d747854544d2de169c84

Το παραπάνω παράδειγμα, παρουσιάζει ένα βασικό δίκτυο RVC. Σε έναν αυτοκινητόδρομο ταχείας κυκλοφορίας με τέσσερις λωρίδες κατεύθυνσης, δύο για κάθε κατεύθυνση, υπάρχει σε ένα κομβικό σημείο ένα σημείο πρόσβασης στο δίκτυο. Συγκεκριμένα, έχει τοποθετηθεί ένα RSU (Roadside Unit) που καλύπτει ένα μέρος του αυτοκινητοδρόμου με το σήμα του. Κάθε όχημα που διασχίζει το σημείο που καλύπτεται από το RSU έχει πρόσβαση σε μηνύματα και προειδοποιήσεις που προέρχονται από άλλα πιθανά RSU του δικτύου. Σε συνεργασία με τα RSU, τα αυτοκινούμενα μεταφέρουν τα μηνύματα τους σε υπόλοιπα οχήματα. Παράλληλα, κάθε σημείο πρόσβασης είναι συνδεδεμένο με έναν απομακρυσμένο server, ο οποίος έχει πρόσβαση στο διαδίκτυο. Βάσει της περιγραφής, καταλαβαίνουμε πως ένα δίκτυο RVC είναι όντως αρκετά αργό στην μετάδοση μηνυμάτων για αποφυγή ατυχημάτων.



Εικόνα 12 Τεχνική HVC

Πηγή: https://www.researchgate.net/figure/The-architecture-of-vehicular-networks-13_fig11_257877942

Το τρέχων παράδειγμα αποτελεί έναν συνδυασμό και της IVC και της RVC αλλά και του υβριδικού HVC. Σε ένα πιθανό δρόμο που αριστερά και δεξιά του βρίσκονται κτίρια, είναι τοποθετημένα και μερικά RSU διάσπαρτα για το δίκτυο RVC. Τα RSU είναι συνδεδεμένα με μία κεντρική κεραία που μεταφέρει τα μηνύματα, η οποία χρησιμοποιεί την τεχνολογία WiMAX. Επιπλέον, οχήματα όπως σχολικά λεωφορεία, ταξί και λοιπά αυτοκινούμενα επικοινωνούν άμεσα μεταξύ τους διαμορφώνοντας και ένα δίκτυο IVC.
Ο συνδυασμός και των δύο τεχνολογιών, δημιουργεί ένα ολοκληρωτικό νέο υβριδικό δίκτυο HVC, όπου ένα αυτοκίνητο συνδέεται με ένα άλλο και ταυτοχρόνως το άλλο όχημα χρησιμοποιεί ένα RSU για να επικοινωνεί με άλλα οχήματα σε μεγαλύτερη απόσταση. Αυτό το είδος επικοινωνίας οχημάτων είναι το πλέον αποδοτικότερο διότι θεωρητικά, μπορεί να καλυφτεί μία τεράστια απόσταση για όλα τα οδικά δίκτυα, με ένα σημαντικό μειονέκτημα που είναι προφανώς το κόστος κατασκευής και συντήρησης ενός τόσο απαιτητικού δικτύου.

3.2 Σημερινή εποχή-VANETs

Φτάνοντας στο σήμερα, οι ερευνητές έχουν έρθει αντιμέτωποι με την πλέον σύγχρονη τεχνολογία δικτύων με όνομα Vehicular Ad Hoc Networks (VANETs), τα οποία όπως αναφέρθηκε και νωρίτερα, αποτελούν προέκταση των MANET. Τα VANET είναι το επόμενο βήμα στην επικοινωνία των οχημάτων στους δρόμους

και η υλοποίηση τους είναι παράρτημα της εφαρμογής των ευφυών συστημάτων μεταφορών. Το λεγόμενο ευφυές σύστημα μεταφορών θα είναι υπεύθυνο για την ασφάλεια των δρόμων, για τις τυχόν αναμονές ακόμα και για πιο οικονομική οδήγηση [28].

Τα VANET έχουν μερικά αποκλειστικά χαρακτηριστικά που τα διαφοροποιούν πλήρως από τα υπόλοιπα MANETs.

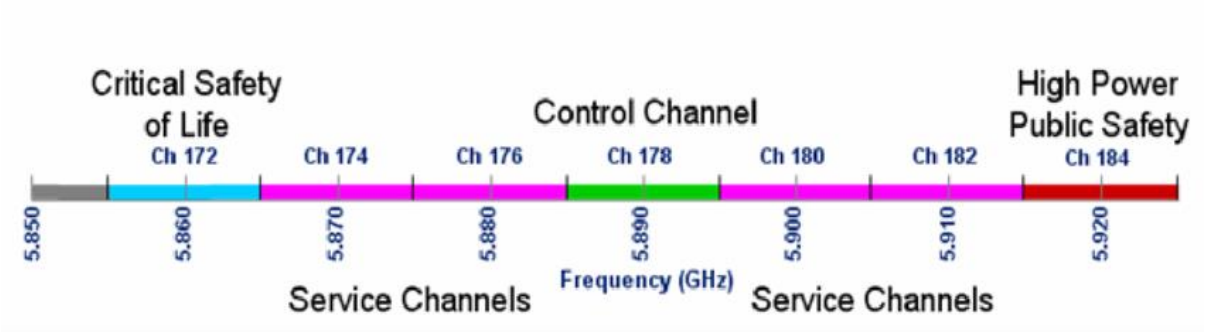
- **Υψηλή Κινητικότητα:** Οι κόμβοι σε ένα δίκτυο VANET παρέχουν μεγάλη ταχύτητα και ορίζουν ένα δυναμικό περιβάλλον δικτύου.
- **Προβλέψιμα και περιορισμένης κινητικότητας μοτίβα:** Οι κινήσεις των κόμβων σε ένα VANET ελέγχονται από ένα σύνολο αυστηρών κανόνων, που τις καθιστά προβλέψιμες.
- **Ταχεία αλλαγή τοπολογίας:** Λόγω της μεγάλης ταχύτητας κίνησης των κόμβων, η τοπολογία του δικτύου αλλάζει διαρκώς.
- **Μηδαμινοί περιορισμοί ενέργειας:** Κάθε όχημα είναι εξοπλισμένο με μια μπαταρία που χρησιμοποιείται ως πηγή ενέργειας για όλες τις λειτουργίες.
- **Εντοπισμός:** Τα οχήματα χρησιμοποιούν GPS για τον άμεσο εντοπισμό της ακριβής τοποθεσίας τους.
- **Άφθονοι κόμβοι δικτύου:** Αντίθετα με τα απλά MANET, τα VANET μπορούν να έχουν τεράστιο εύρος σήματος λόγω των οχημάτων με μεγάλο μέγεθος.
- **Αυστηροί περιορισμοί καθυστερήσεων:** Τα VANET είναι υπεύθυνα για την μεταφορά μηνυμάτων ασφάλειας. Στα μηνύματα αυτά πρέπει να δίνεται η υψηλότερη προτεραιότητα γιατί πρέπει να αποστέλλονται στην ώρα τους.

3.2.1 DSRC

Σκοπός των ερευνητών είναι να εφαρμόσουν ως τεχνολογία σήματος το DSRC (Dedicated Short Range Communication) που αποτελεί εξελιγμένη μορφή της τεχνολογίας Wi-Fi που είναι κατάλληλη για τα VANET. Η τεχνολογία θα χρησιμοποιηθεί στον τομέα ITS (Intelligent Transportation System) για την παροχή ασφαλούς και αξιόπιστης επικοινωνίας συνδέσεων μεταξύ οχημάτων και υποδομών. Αυτοί οι σύνδεσμοι επικοινωνίας επιτρέπουν τη μεταφορά δεδομένων που είναι απαραίτητη για τη λειτουργία διαφορετικών εφαρμογών ITS. Το DSRC έχει αναπτυχθεί για να λειτουργεί σε δίκτυα με πολύ υψηλή δυναμική για την υποστήριξη γρήγορης σύνδεσης εγκαταστάσεων και να ελαχιστοποιηθεί η καθυστέρηση επικοινωνίας. Κυρίως, το DSRC έχει σχεδιαστεί για να διασφαλίζει την αξιοπιστία της

υπηρεσίας για εφαρμογές ασφαλείας λαμβάνοντας υπόψη τους χρονικούς περιορισμούς για αυτού του είδους τις εφαρμογές. Μπορεί επίσης να υποστηρίξει άλλες μη ασφαλείς εφαρμογές που απαιτούν εγγύηση ποιότητας υπηρεσίας (QoS). Το DSRC έχει αναπτυχθεί για περιβάλλοντα όπου είναι μικρά. Απαιτούνται χρονική απόκριση (λιγότερο από 50 msec.) ή/και υψηλοί ρυθμοί δεδομένων σε υψηλά δυναμικά δίκτυα [28].

Για το DSRC έχει αφιερωθεί πλήρως η μπάντα συχνότητας 5.9GHz και έχει χωριστεί σε 7 διαφορετικά κανάλια των 10 MHz το καθένα χωρίς να επικαλύπτει το ένα το άλλο. Το πρώτο κανάλι είναι το κανάλι ελέγχου, τα υπόλοιπα έξι ονομάζονται κανάλια υπηρεσίας. Επίσης, τα δύο κανάλια που βρίσκονται στις άκρες είναι για μελλοντική χρήση. Για αρχή, το κανάλι ελέγχου χρησιμοποιείται για την μετάδοση προειδοποιητικών μηνυμάτων προς τους οδηγούς. Επιπλέον, μπορεί να ανακοινώσει τη χρήση άλλων υπηρεσιών μέσω των καναλιών. Τα κανάλια υπηρεσιών από την άλλη μεριά, είναι υπεύθυνα για την αναμετάδοση των προειδοποιητικών μηνυμάτων μέσω των οδηγών, προς άλλες μονάδες κόμβων σε άλλες περιοχές. Το DSRC υποστηρίζει διαφορετικούς ρυθμούς μετάδοσης δεδομένων όπως, 6,9,12,18,24 και 27 Mbps με ένα κανάλι 10 MHz. Προαιρετικά, ο ρυθμός μετάδοσης μπορεί να αυξηθεί σε 54 Mbps με ένα κανάλι 20 MHz. Η εναλλαγή μεταξύ των ρυθμών μετάδοσης επιτυγχάνεται με την αλλαγή του σχήματος του σήματος και του ρυθμού κωδικοποίησης καναλιού.



Εικόνα 13 Κανάλια και συχνότητες DSRC

Πηγή: https://www.researchgate.net/figure/DSRC-spectrum-band-and-channels_fig1_258841686

Αναφορικά κάποια παραδείγματα εφαρμογών με βάση το DSRC στην καθημερινότητα:

- **Cooperative Collision Avoidance-Συνεργατική Αποφυγή Σύγκρουσης:** Σύστημα αποφυγής ατυχημάτων στους δρόμους μέσω αισθητήρων που θα ανιχνεύουν πιθανή σύγκρουση. Το σύστημα μπορεί να αποστείλει ειδοποίηση στον οδηγό ή να λάβει μόνο του απόφαση και να διακόψει την κίνηση του οχήματος.
- **Emergency Warning Messages-Προειδοποιητικά Μηνύματα Έκτακτης Ανάγκης:** Παρόμοιο σύστημα με το CCA. Το EWM στέλνει μήνυμα στον οδηγό για πιθανό ατύχημα, όμως ανάλογα με το εμπόδιο κάθε φορά, το μήνυμα αυτό μπορεί να αποσταλεί και σε άλλους οδηγούς σε κοντινούς δρόμους για να είναι προετοιμασμένοι για το εμπόδιο.
- **Cooperative Intersection Collision Avoidance-Συνεργατική Αποφυγή Σύγκρουσης σε Σταυροδρόμια:** Αποτελεί προέκταση του συστήματος CCA όμως αφορά συγκεκριμένα τα ατυχήματα που μπορεί να συμβούν σε σταυροδρόμια. Ένα RSU είναι τοποθετημένο στο σταυροδρόμι και επικοινωνεί με ερχόμενα οχήματα προς αποφυγήν λανθασμένων κινήσεων από τους οδηγούς.
- **Electronic Toll Collection-Ηλεκτρονική Είσπραξη Διοδίων:** Μέσω δικτύου, οι πληρωμές πλέον στους σταθμούς διοδίων θα πραγματοποιούνται αυτόματα.

3.2.2 Διάδοση Δεδομένων

Στα δίκτυα VANET χρησιμοποιούνται κάποιες συγκεκριμένες μέθοδοι για την αποστολή μηνυμάτων. Ανάλογα με τις υποδομές του δικτύου και τις υπηρεσίες που είναι καθορισμένο να προσφέρει, μπορεί να αντιστοιχεί και μία συγκεκριμένη μέθοδος:

- **Flooding-based method:** Η μέθοδος Flooding (Πλημμύρας) αφορά έναν κόμβο που αποστέλλει το προειδοποιητικό μήνυμα προς άλλους κόμβους στην ακτίνα που εκπέμπει. Η αποστολή του μηνύματος είναι τύπου broadcast, δηλαδή το μήνυμα λαμβάνεται από όλους τους κόμβους που συμμετέχουν χωρίς διακρίσεις. Ένα από τα θετικά αυτής της μεθόδου είναι η στιγμιαία αποστολή του μηνύματος σε όλους που αποσκοπεί στην αποφυγή τυχόν καθυστερήσεων. Αντιθέτως, είναι σημαντικά αρνητικό στοιχείο είναι η έλλειψη ικανότητας διαχείρισης των κόμβων, όταν το δίκτυο είναι πυκνό. Συνεπώς, προκαλείται συμφόρηση με την broadcast αποστολή του μηνύματος. Επίσης, η μέθοδος πλημμύρας δεν είναι κατάλληλη για την αντιμετώπιση εισαγωγής και αφαίρεσης κόμβων από το δίκτυο [28].
- **Relay-based method:** Η μέθοδος αυτή, χρησιμοποιεί έναν εξυπνότερο αλγόριθμο πλημμύρας. Σε αυτήν την περίπτωση, χρησιμοποιείται ένας συγκεκριμένος κόμβος ή μία ομάδα κόμβων για να στείλουν τα μηνύματα, με σκοπό την αποστολή των πακέτων σε μεγαλύτερη απόσταση. Έτσι,

δημιουργείται ένας μηχανισμός «σκυταλοδρομίας» που ένας κόμβος παραδίδει σε έναν άλλο το μήνυμα και εκείνος με την σειρά του σε άλλον και ούτω καθεξής. Η μέθοδος αντιμετωπίζει εύκολα το θέμα της επεκτασιμότητας των κόμβων, αλλά παρουσιάζει δυσκολία στην επιλογή του κατάλληλου κόμβου για τον προαναφερόμενο μηχανισμό «σκυταλοδρομίας».

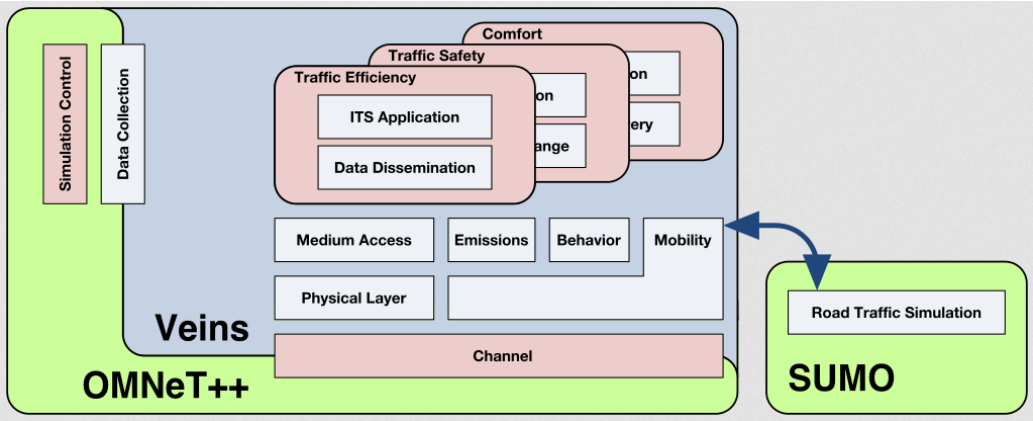
- **Time-based algorithms:** Αυτή η μέθοδος διάδοσης, αποσκοπεί στην εξάλειψη των άσκοπων μεταφορών πακέτων. Ο αλγόριθμος επιλέγει τους κόμβους που καλύπτουν την μεγαλύτερη ακτίνα για να στείλουν το μήνυμα στην αρχή. Όταν ένας κόμβος λαμβάνει ένα μήνυμα, ένα χρονόμετρο ξεκινάει να μετρά αντίστροφα έως την λήξη του ή έως όταν αποσταλεί το ίδιο μήνυμα ξανά. Η αρχική τιμή του χρονομέτρου αντιστοιχεί στην απόσταση που βρίσκεται ο αποστολέας από τον παραλήπτη. Επομένως, ο κόμβος στέλνει το μήνυμα σε επόμενο κόμβο όταν λήξει το χρονόμετρο του. Οι υπόλοιποι κόμβοι που θα λάβουν ξανά το ίδιο μήνυμα σταματάνε και αυτοί τα χρονόμετρα τους. Με την λειτουργία αυτή μειώνεται ταυτοχρόνως και η χρονική διάρκεια αποστολής ενός μηνύματος.

4. VEINS Framework

Το Veins framework είναι ένας εξομοιωτής δικτύων οχημάτων ανοιχτού κώδικα, που λειτουργεί ως σουίτα μοντέλων εξομοίωσης δικτύων για οχήματα σε πραγματικό χρόνο. Το κάθε μοντέλο παίρνει εντολές από το OMNET++, το οποίο είναι προσομοιωτής δικτύων που βασίζεται σε γεγονότα, και την ίδια στιγμή συνεργάζεται με το SUMO, ένα προσομοιωτή κίνησης σε αυτοκινητόδρομους. Το Veins λειτουργεί παράλληλα με τα υπόλοιπα προγράμματα και συνθέτει μια ολοκληρωμένη εμπειρία για διεκπεραίωση πειραμάτων για δίκτυα οχημάτων με πραγματικά δεδομένα. Κοινώς, το Veins είναι ένα framework προσομοίωσης που χρησιμοποιείται για την σύνταξη κώδικα προσομοίωσης για συγκεκριμένη εφαρμογή. Το Veins μπορεί να χρησιμοποιηθεί και χωρίς κάποια επιπρόσθετη παράμετρο εγκατεστημένη, ως ένα πολύ πιο περιορισμένο εργαλείο προσομοίωσης. Τυπικά, ο χρήστης θα γράψει κώδικα και στην συνέχεια αυτός θα αξιολογηθεί μέσω της προσομοίωσης που θα παράγει. Παράλληλα, το framework θα λειτουργεί ως ρυθμιστής για παραμέτρους όπως τα μοντέλα πρωτοκόλλων που θα χρησιμοποιηθούν στο εικονικό δίκτυο, για την κίνηση των μοντέλων μέσα στο πλαίσιο (canvas) που θα ορίσει το framework, αλλά και η συλλογή δεδομένων και αποτελεσμάτων κατά την εκτέλεση της προσομοίωσης [9].

Το Veins παρέχει μια μεγάλη συλλογή από μοντέλα προσομοίωσης που ανήκουν σε ένα ευρύ φάσμα δικτύων οχημάτων. Σαφώς, ο χρήστης επιλέγει ποια μοντέλα θα χρησιμοποιήσει σε κάθε προσομοίωση που σημαίνει πως μερικά μοντέλα μπορεί να φανούν αχρείαστα. Κάποια μοντέλα που διαθέτει το framework είναι χρήσιμα για ένα μόνο συγκεκριμένο τύπο δικτύου προσομοίωσης.

Το Veins είναι framework ανοιχτού κώδικα, που σημαίνει πως κάθε στοιχείο του είναι ελεύθερο για κάθε χρήστη, για χρήση αλλά και για μελέτη. Αυτό μπορεί να φανεί εξαιρετικά χρήσιμο για τους ερευνητές που θέλουν να κοινοποιήσουν το έργο τους μέσω του framework σε τρίτους χωρίς να σκέφτονται αν οι υπόλοιποι ερευνητές που θα ενδιαφερθούν, θα έχουν πρόσβαση στην προσομοίωση. Παράλληλα, οι ερευνητές είναι ελεύθεροι να προσθέσουν νέο υλικό στο framework όπως μοντέλα που σχεδίασαν οι ίδιοι ή και κάποιο καινούριο πρωτόκολλο δικτύωσης. Αυτό το προνόμιο, έχει φέρει σε θέση το framework να μπορεί να εξυπηρετήσει έως και πολύ εξειδικευμένα πειράματα προσομοίωσης χωρίς ελλιπή στοιχεία για την πλήρης ολοκλήρωση τους.



Εικόνα 14 Συσχετισμός λογισμικών προσομοίωσης

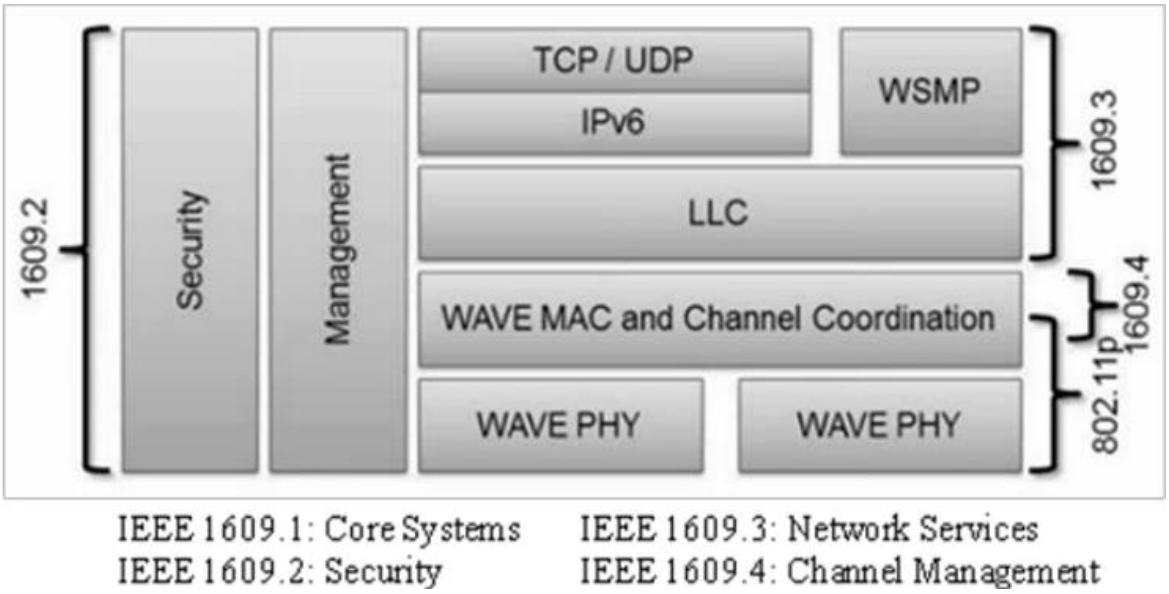
Πηγή: https://veins.car2x.org/documentation/

Το παραπάνω μοντέλο αναπαριστά το ρόλο και τις αρμοδιότητες που έχουν τα τρία συνεργαζόμενα προγράμματα και πως αυτά συνδέονται μεταξύ τους. Το Veins και το OMNET++ συνδέονται μέσω θύρας TCP. Το πρωτόκολλο επικοινωνίας που έχει καθοριστεί για αυτή τη σύνδεση λέγεται Traffic Control Interface ή TraCI. Το πρωτόκολλο επιτρέπει την αμφίδρομη σύζευξη της κίνησης στο δρόμο με την κίνηση ενός δικτύου. Επίσης, η κίνηση των μοντέλων σε μια προσομοίωση δρόμου γίνεται μέσω του SUMO και αντιστοιχίζεται σε κίνηση κόμβων στο περιβάλλον του OMNET++. Για να επιτευχθεί αυτή η σύζευξη πρέπει πρώτα να δημιουργηθεί μία μέθοδος με όνομα TraCIScenarioManager η οποία είναι αρμόδια για την

συνεργασία του SUMO και του OMNET++. Όμως, για κάθε μία προσομοίωση στο περιβάλλον του OMNET++, ο χρήστης πρέπει χειροκίνητα να εκτελέσει μία προσομοίωση στο περιβάλλον του SUMO. Εναλλακτικά, το Veins framework μπορεί να αυτοματοποιήσει την διασύνδεση των δύο προσομοιωτών με την χρήση μιας υποκλάσης του TraCIScenarioManager που ονομάζεται TraCIScenarioManagerLaunchd. Το μόνο που χρειάζεται να κάνει ο χρήστης είναι να εκτελέσει σε περιβάλλον γραμμής εντολών το command sumo-launchd.py το οποίο αναμένει για εισερχόμενες συνδέσεις προσομοίωσης OMNET++. Ακόμη, με την χρήση μιας άλλης υποκλάσης με όνομα TraCIScenarioManagerForker μπορούμε να κάνουμε απευθείας εκκίνηση μιας προσομοίωσης στο SUMO αν χρειαστεί [10].

Ιδιότητες VEINS

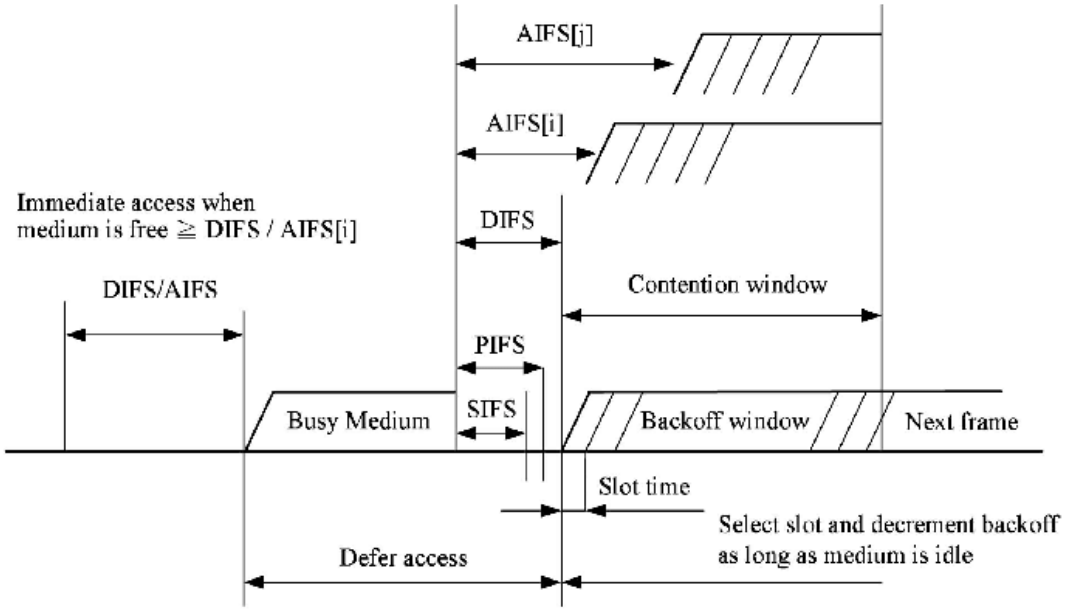
Ένα από τα βασικά προτερήματα του Veins είναι η λεπτομερής μοντελοποίηση των χαμηλότερων επιπέδων των επικοινωνιών IVC. Για την αξιοποίηση των περισσότερων εφαρμογών με επικοινωνίες IVC, χρειάζεται μία εκτενής προσομοίωση σε packet-level που να χρησιμοποιεί τα ακριβή μοντέλα της τεχνολογίας. Όσον αφορά τα δίκτυα οχημάτων, η τεχνολογία αυτή ονομάζεται IEEE WAVE ή ETSI ITS-G5, η οποία ακολουθεί τα πρότυπα της πολυκάναλης λειτουργίας IEEE 1609.4 και κάνει χρήση των IEEE 802.11p MAC και PHY επιπέδων. Το Veins επικεντρώνεται στα κατώτερα στρώματα, διότι αυτά παίζουν καθοριστικό ρόλο για την πρόσβαση στο κανάλι μετάδοσης και τη μεταφορά δεδομένων.



Εικόνα 15 IEEE 1609/WAVE protocol stack

Πηγή:https://www.researchgate.net/figure/The-IEEE-1609-WAVE-protocol-stack-adapted-from-4_fig2_265251552

Το Veins παρέχει τις τεχνολογίες IEEE 802.11b και IEEE 1609.4 του επιπέδου της MAC που υποστηρίζουν πολυκάναλη λειτουργία, εναλλαγή καναλιών, μεταφορά μηνυμάτων με unicast/broadcast και μια υλοποίηση EDCA του IEEE 802.11e με τέσσερις διαφορετικές κατηγορίες πρόσβασης. Η EDCA παρέχει τις περισσότερες πιθανότητες αποστολής για μια μεταφορά μηνύματος υψηλής προτεραιότητας σε σχέση με μία οποιαδήποτε άλλη μεταφορά με χαμηλότερη προτεραιότητα. Αυτό επιτυγχάνεται μέσω ενός πρωτοκόλλου με όνομα TCMA, το οποίο αποτελεί μία παραλλαγή του CSMA/CA, όπου χρησιμοποιεί ένα μικρότερο πλαίσιο ελέγχου για πακέτα υψηλότερης προτεραιότητας (AIFS). Επίσης, η EDCA προσφέρει πρόσβαση χωρίς συγκρούσεις σε ένα κανάλι, με την Ευκαιρία Μετάδοσης (TXOP). Η TXOP προσφέρει ένα περιορισμένο χρονικό διάστημα κατά το οποίο ένα σημείο πρόσβασης μπορεί να στείλει όσο το δυνατό γίνεται περισσότερα frames, όταν η διάρκεια των αποστολών δεν υπερβαίνει τη μέγιστη διάρκεια της TXOP. Αν ένα πακέτο έχει πολύ μεγάλο μέγεθος για να μεταδοθεί με μια TXOP, πρέπει να κατακερματιστεί (hashing) σε μικρότερα πακέτα [11].



Εικόνα 16 EDCA

Πηγή: https://www.researchgate.net/figure/EDCA-of-IEEE-80211e_fig1_220465978

4.1 Επιπρόσθετα χαρακτηριστικά του VEINS

ARIB STD-T109

Το Veins κατέχει το μοντέλο προσομοίωσης του ιαπωνικού ARIB T109 πρότυπου επικοινωνίας για το ITS στη μπάντα συχνότητας 700 MHz. Το πρότυπο ARIB ορίζει μια διεπαφή μεταξύ των ασύρματων τμημάτων του “Radio Equipment of 700 MHz Band Intelligent Transport Systems” το οποίο έχει καθοριστεί από το Υπουργείο Εσωτερικών και Επικοινωνιών μέσω σχετικού διατάγματος που ρυθμίζει τον ραδιοεξοπλισμό, μεταξύ επίγειων κινητών σταθμών και σταθμών βάσης. Η μπάντα 700 MHz αναμένεται να προσφέρει στους οδηγούς εξαιρετικά αξιόπιστες πληροφορίες ασφάλειας από φορείς διαχείρισης της κυκλοφορίας μέσω επικοινωνιών καθ’ οδόν με όχημα, καθώς και επί τόπου επικοινωνία οχημάτων σε κοντινή απόσταση με πολύ χαμηλό λανθάνοντα χρόνο. Το σύστημα αναμένεται να λειτουργεί στη συχνότητα των 700 MHz και αποτελείται από σταθμούς βάσης που αναπτύσσονται κατά μήκος των δρόμων και από κινητούς σταθμούς που θα είναι εγκατεστημένοι στα κινούμενα οχήματα. Πρωτεύον στόχος του συστήματος είναι η μείωση του αριθμού των τροχαίων ατυχημάτων βάση ενημερώσεων για τις συνθήκες κυκλοφορίας που επικρατούν στους δρόμους [12].

Η επικοινωνία που επιτυγχάνει το σύστημα βασίζεται σε ραδιοκύματα μεγάλης ταχύτητας και μικρής εμβέλειας, για την μέγιστη αποδοτικότητα. Επιπλέον, οι σταθμοί παρέχουν επαναχρησιμοποίηση υψηλού επιπέδου συχνοτήτων σε ζώνες μικρής εμβέλειας, επικοινωνίες με υψηλή χωρητικότητα, με μεγάλες ταχύτητες και πολύ χαμηλές καθυστερήσεις, όπως επίσης και μετάδοση broadcast και από τα δύο είδη σταθμών. Επιπροσθέτως, κάποιες περαιτέρω λειτουργίες προορίζονται για το μοντέλο, που όμως δεν έχουν καθοριστεί στο συγκεκριμένο πρότυπο. Αρχικά, προορίζεται συνδεσιμότητα των σταθμών με συσκευή GPS που θα είναι εγκατεστημένη στο αυτοκίνητο και θα συλλέγει πληροφορίες για διευκόλυνση του οδηγού. Δεύτερον, σύνδεση με εγκατεστημένη οθόνη στο αυτοκίνητο η οποία θα απεικονίζει πληροφορίες στα υπόλοιπα οχήματα. Τέλος, σύνδεση με συσκευή που θα είναι αρμόδια για την συλλογή δεδομένων που αφορούν την κατάσταση στους δρόμους. Οι απαιτήσεις που έχουν καθοριστεί για την εφαρμογή του συστήματος είναι αρχικά οι δρόμοι να έχουν εξοπλιστεί κατάλληλα με τους ανάλογους σταθμούς με άριστη επικοινωνία μεταξύ τους. Στη συνέχεια, οι σταθμοί πρέπει να είναι ρυθμισμένοι ώστε να χρησιμοποιούν σύνδεση 700 MHz με μονό κανάλι, και να είναι έτοιμοι να εξυπηρετήσουν επικοινωνίες IVC-RVC. Παράλληλα, για την σύνδεση IVC πρέπει να υπάρχει άνεση σε ταχύτητες που ακουμπούν τα 140 χλμ./ώρα, ενώ για επικοινωνίες RVC ταχύτητες έως 70χλμ./ώρα.

Καθώς αυτό το σύστημα είναι υποχρεωμένο να εξυπηρετεί ταυτόχρονα επικοινωνίες IVC-RVC όταν την ίδια στιγμή κάνει χρήση ενός μόνο καναλιού, πρέπει να ορίσει διαφορετικές χρονικές περιόδους μεταφοράς πακέτων και για σταθμούς βάσης και για κινητούς, ώστε να συνυπάρχουν αρμονικά. Οι σταθμοί βάσης έχουν προκαθορισμένη χρονική περίοδο εκπομπής, ενώ οι κινητοί σταθμοί ακολουθούν το πρωτόκολλο Carrier Sense Multiple Access/Collision Avoidance εντός της ορισμένης χρονικής περιόδου. Στο επίπεδο του Veins, το πρότυπο μοντελοποιεί τα χαρακτηριστικά και του PHY και του MAC επιπέδου.

Εικόνα 17 ARIB-STD T109

Πηγή: https://veins.car2x.org/documentation/modules/

Το επίπεδο της MAC αποτελείται από δύο διαφορετικές υλοποιήσεις που διακρίνονται σε σταθμούς βάσης και κινητούς σταθμούς. Και τα δύο μοιράζονται μια κοινή αφηρημένη κλάση που περιέχει λειτουργικότητα και για τις δύο υλοποιήσεις. Το επίπεδο της MAC συνδυάζει το CSMA/CA από το IEEE 802.11p και το επίπεδο IVC-RVC του ARIB T109 που κάνει χρήση του μοντέλου TDMA [31].

Two-Ray Interference Model

Τα μοντέλα απώλειας διαδρομής αποτελούν κεντρικό παράγοντα για την ακέραια μοντελοποίηση πληροφοριών διάδοσης δεδομένων σε ένα δίκτυο οχημάτων. Η απώλεια διαδρομής συνήθως υπολογίζεται έχοντας υπόψιν την διάδοση στον ελεύθερο χώρο μαζί με την απόσταση μεταξύ των δύο σταθμών και το μήκος κύματος που παράγεται [13].

Η εξίσωση είναι: $L[dB] = 20lg*(4\pi*d/\lambda)$

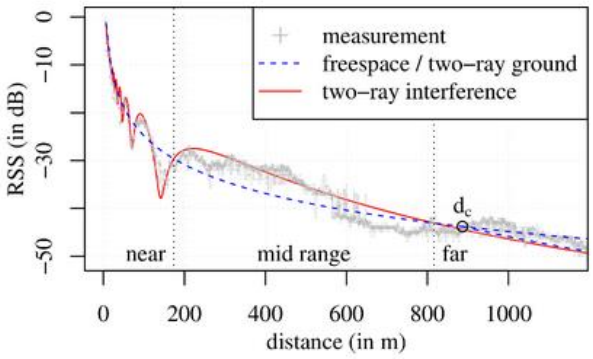
Παρόλα αυτά, ένας πιο ρεαλιστικός τρόπος για την διαχείριση της απώλειας διαδρομής είναι να αναλογιστούμε πως η διάδοση σήματος θα συναντήσει σχεδόν σίγουρα κάποια αισθητή μορφή εμποδίου. Επομένως, μια πιο σωστή προσέγγιση της απώλειας διαδρομής θα είναι η διαφορά φάσης-συντελεστή ανάκλασης το οποίο μας οδηγεί στον μοντέλο παρεμβολής δύο ακτινών (two ray interference model).

Βέβαια, ο υπολογισμός τους είναι σαφέστατα πιο περίπλοκος από τον υπολογισμό της απώλειας διαδρομής σύμφωνα με το μοντέλο του ελεύθερου χώρου. Για αυτό, παρουσιάστηκε η ιδέα πως για μεγάλες αποστάσεις d και με την υπόθεση πως υπάρχει τέλεια πόλωση και αντανάκλαση, ο υπολογισμός των παρεμβολών μεταξύ της οπτικής επαφής (Line-of-sight) και των ανακλώμενων ακτινών μπορεί να απλοποιηθεί μέσω της εξίσωσης επίγειας απώλειας διαδρομής δύο ακτινών (two ray ground reflection model).

Η εξίσωση είναι: L[dB] = 20lg*(d^2/ht*hr)

Το γεγονός αυτό οδήγησε πολλούς προσομοιωτές δικτύων να προσθέσουν στο υλικό τους αυτή την μέθοδο ως επιλογή για τον υπολογισμό απώλειας διαδρομής στην μετάδοση σήματος.

Συγκεκριμένα το Veins περιλαμβάνει ένα μοντέλο παρεμβολής δύο ακτινών το οποίο καταφέρνει να απεικονίσει τα φαινόμενα ανάκλασης επί εδάφους όπως φαίνεται παρακάτω:



Εικόνα 18 Two-Ray Interference Model

Πηγή: https://veins.car2x.org/documentation/modules/

Obstacle Shadowing

Στα VANET τα μηνύματα που μεταφέρονται μεταξύ των οχημάτων μπορεί να χαθούν λόγω διακοπτόμενων συνδέσεων που προκαλείται από φαινόμενα σκίασης εμποδίων (obstacle shadowing) μέσω ψηλών κτηρίων, ουρανοξυστών και άλλων εμποδίων σε πολλές περιοχές του κόσμου με μεγάλη πυκνότητα κίνησης οχημάτων. Οι συνθήκες ενός καναλιού που μεταφέρει πακέτα μεταξύ των δύο κόμβων μπορεί πολύ γρήγορα να υποστεί αλλαγές μεταξύ μιας σχεδόν άριστης ποιότητας , σε μια κατάσταση μέτριας απόδοσης έως και την αισθητά έντονη σκίαση εμποδίων. Έτσι, είχε εδραιωθεί η ιδέα πως τα αληθινά γεγονότα που αφορούν την σκίαση εμποδίων αποτελούσαν μείζον ζήτημα για την ποιότητα και την απόδοση των VANETs. Ωστόσο, παρουσιάστηκε η ευκαιρία για την εισαγωγή του ray-tracing ως μέσω αντιμετώπισης των εμποδίων. Βέβαια, αυτού του είδους η προσέγγιση έδειξε πολύ γρήγορα πως το

σενάριο ανάπτυξης αυτής της ιδέας είναι πανάκριβη από την άποψη του κόστους, έως και σχεδόν απαγορευτική. Ως εναλλακτική περίπτωση, δόθηκε το βήμα για την ανάπτυξη στοχαστικών μοντέλων που χρησιμοποιούνται για την περιγραφή των χαρακτηριστικών ενός ασύρματου καναλιού με αρκετή ακρίβεια από μια απομακρυσμένη σκοπιά. Όμως, η σχεδίαση του καναλιού από μια στοχαστική οπτική μπορεί να οδηγήσει σε σοβαρή απόκλιση από την πραγματική συμπεριφορά μιας μονής σύνδεσης [16].
Το Veins περιλαμβάνει ένα απλό μοντέλο σκίασης εμποδίων το οποίο έχει ρυθμιστεί και επικυρωθεί για μετρήσεις του πραγματικού κόσμου. Το μοντέλο μπορεί να ενεργοποιηθεί αν προσθέσουμε μια ενότητα τύπου ObstacleControl στην προσομοίωση.

Antenna Patterns

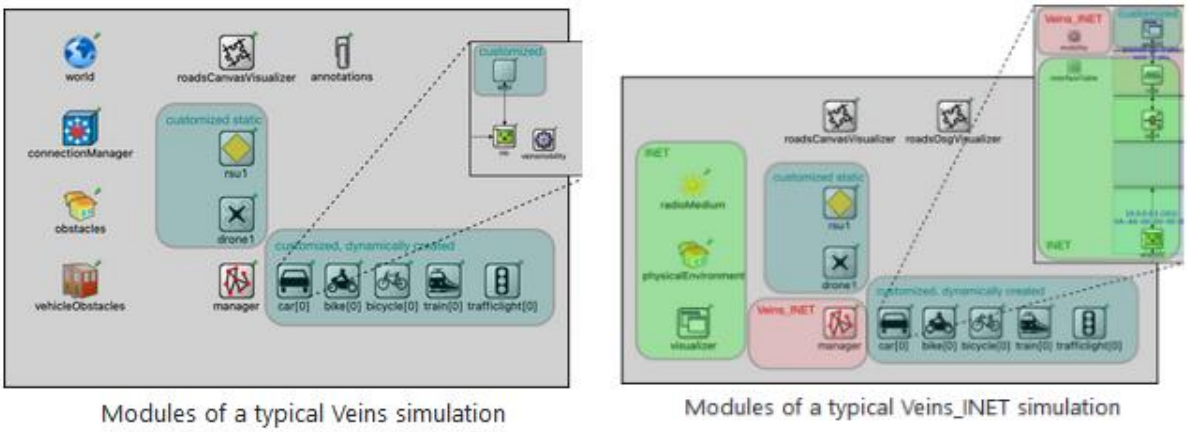
Η σημασία για τις εκπομπές των κεραιών για τα δίκτυα οχημάτων ήταν κάτι που οι ερευνητές είχαν κατανοήσει από την αρχή. Είχε γίνει γνωστό ήδη από την δεκαετία του 1980 πως η απόδοση του σήματος μια κεραίας δεν εξαρτάται μόνο από την ίδια την συσκευή της κεραίας, αλλά και από εξωτερικούς παράγοντες όπως η θέση στην οποία θα τοποθετηθεί. Παράλληλα, εξετάστηκε και η ιδέα της αμφίδρομης επικοινωνίας δύο οχημάτων σε πραγματικό χρόνο και εντοπίστηκε πως πράγματι η ικανότητα μιας κεραίας να λαμβάνει ένα σήμα με καλή ποιότητα βασιζόταν στο σημείο που έχει τοποθετηθεί η κεραία επάνω στο όχημα. Ύστερα, παρουσιάστηκε η ιδέα του συνδυασμού κεραιών τοποθετημένα σε ένα όχημα. Η κίνηση αυτή αποδείχθηκε αποδοτική μιας και όχι μόνο το σήμα της κεραίας αλλά και το μοτίβο εγκατάστασης των κεραιών επάνω στο όχημα αποδίδουν πολύ περισσότερο σε θέματα ισχύος [17].
Σχετικά με το Veins, το μοντέλο προσομοίωσης που αφορά την εξάρτηση των κεραιών από τις γωνίες που εκπέμπουν είναι διαθέσιμο προς χρήση. Περιλαμβάνει παραμέτρους από διαφορετικούς τύπους κεραιών που χρησιμοποιούνται σε πραγματικά εγχειρήματα τα οποία βασίζονται στον τύπο τους και στο σημείο τοποθέτησης τους. Ένα υπόδειγμα κεραίας μπορεί να διαμορφωθεί θέτοντας μια παράμετρο τύπου **antenna** σε ένα configuration αρχείο της προσομοίωσης του Veins ρυθμίζοντας παράλληλα τον τύπο της, και τις υπόλοιπες παραμέτρους από τις οποίες θα εξαρτάται η απόδοση της. Αν δεν έχει οριστεί κάποιο υπόδειγμα κεραίας στην προσομοίωση, τότε το φυσικό επίπεδο του Veins ορίζεται ως ισοτροπικό με 0 dBi κέρδος.
Το παράδειγμα που ακολουθεί είναι ένα τμήμα κώδικα γραμμένο σε XML που διαμορφώνει τη κεραία με τέτοιο τρόπο ώστε να έχει δείγματα κέρδους 2 dBi μπροστά στο αυτοκίνητο, 1.1 dBi στα δεξιά , -4.0 dBi στην πισινή μεριά και 0.9 dBi στην αριστερή πλευρά του οχήματος. Επιπρόσθετα, ο κώδικας εφαρμόζει

τυχαίο κέρδος σε dBi με τιμές +/- 1 σε κάθε κεραία , και περιστρέφει το υπόδειγμα της εκπομπής της κεραίας με τυχαίες τιμές μοιρών ίσες με +/- 1 dBi.

```
<Antenna type”SampledAntenna1D” id=”patch”>
        <parameter name=”samples” type=”string” value=”2.0 1.1 -4.0 0.9”/>
        <parameter name=”random-offsets” type=”string” value=”uniform -1 1”/>
        <parameter name=”random-rotation” type=”string” value=”uniform -1 1”/>
</Antenna>
```

Veins_INET Subproject και INET Framework

Το Veins_INET περιλαμβάνεται δωρεάν για χρήση μέσω του Veins συμπεριλαμβάνοντας χαρακτηριστικά όπως τις πλήρεις στοίβες των IPv4/IPv6, ενσύρματες επικοινωνίες, πρωτόκολλα πολλών θεματικών πλαισίων κ.α. (Περισσότερα για το INET παρακάτω). Η βασική έκδοση του Veins χρησιμοποιεί μια σχετικά περιορισμένη συλλογή από μοντέλα που αφορούν στοιχειώδη χαρακτηριστικά που μπορεί να αφορούν ένα δίκτυο σε δρόμο με οχήματα. Ωστόσο, η παρουσία του Veins_INET μπορεί να διαμορφώσει έναν συνδυασμό των βιβλιοθηκών του Veins με αυτή του OMNET++ (βλ. παρακάτω).



Εικόνα 19 VEINS & Συνεργασία με INET

Πηγή: https://veins.car2x.org/documentation/modules/

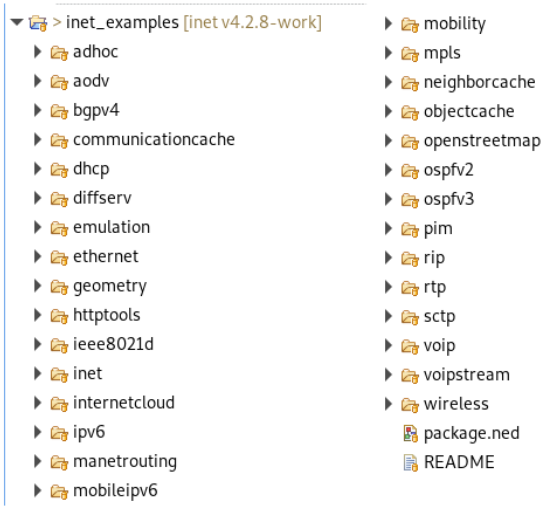
Η σχέση μεταξύ των δύο βιβλιοθηκών αποδίδει τον καλύτερο συνδυασμό για την κατασκευή προσομοίωσης δικτύων.

Το Veins_INET μπορεί να αξιοποιηθεί μέσω του περιβάλλοντος του OMNET++ με την μεταγωγή του φακέλου subprojects/veins-inet στο folder tab του interface.

Γενικότερα, το INET είναι ένα ολοκληρωμένο framework που έχει σχεδιαστεί για χρήση μέσω του περιβάλλοντος προσομοίωσης του OMNET++. Το framework είναι ανοιχτού κώδικα που σημαίνει ότι είναι δωρεάν για όλους και μπορεί να δεχτεί οποιαδήποτε προσθήκη ελεύθερα από χρήστες. Παρέχει πολλών ειδών πρωτόκολλα, μέσα αυτοματισμού και πλήθος από μοντέλα για εφαρμογή σε προσομοιώσεις δικτύων επικοινωνίας από ερευνητές και μαθητές. Το INET ειδικεύεται στον σχεδιασμό και την επικύρωση νέων πρωτοκόλλων και στην εξερεύνηση νέων περίπλοκων σεναρίων προσομοίωσης [18].

Το INET υποστηρίζει ένα ευρύ φάσμα δικτύων επικοινωνίας που μπορούν να χωριστούν σε ενσύρματα, ασύρματα, κινητά, ad hoc, και δίκτυα αισθητήρων. Συμπεριλαμβάνονται μοντέλα που αφορούν στοίβες του Internet όπως TCP, UDP, IPv4, IPv6, OSPF, BGP, πρωτόκολλα link layer σαν το Ethernet, το PPP(Point-to-Point protocol, το IEEE 802.11 κ.α. Επίσης, προσφέρει εξευγενισμένη για το φυσικό επίπεδο ασύρματων δικτύων, πρωτόκολλα δρομολόγησης για δίκτυα MANET, αρχιτεκτονική δικτύωσης διαφοροποιημένων υπηρεσιών, τεχνική δρομολόγησης MPLS (Multiprotocol Label Switching) και μια ανεξάντλητη πληθώρα άλλων υπηρεσιών. Το INET μπορεί να χρησιμοποιηθεί και ως βάση για άλλα frameworks για την προσομοίωση δικτύων οχημάτων.

Παρακάτω απεικονίζονται κάποιες από τις υπηρεσίες του INET που παρέχονται αφού εγκατασταθεί στο περιβάλλον προσομοίωσης του OMNET++



Εικόνα 20 Στιγμιότυπο από το περιβάλλον χρήσης του OMNET++ με το περιεχόμενο του INET

Το INET χρησιμοποιεί ένα API πακέτων για να διευκολύνει την υλοποίηση των πρωτοκόλλων επικοινωνίας και εφαρμογών. Η αναπαράσταση των πακέτων είναι σημαντική για τις προσομοιώσεις δικτύων επικοινωνίας. Πολλές εφαρμογές και πρωτόκολλα επικοινωνιών δομούν, αποδομούν, ενθυλακώνουν, τεμαχίζουν , συνδυάζουν και χειρίζονται πακέτα προς αξιοποίηση τους. Για την επίτευξη αυτών των τεχνικών, το INET παρέχει μια πλούσια σε δυνατότητες κλάση, την Packet. Η κλάση Packet είναι ικανή για εφαρμογές όπως για παράδειγμα η αναπαράσταση δεδομένων και πακέτων αλλά και το φιλτράρισμα αυτών [19].

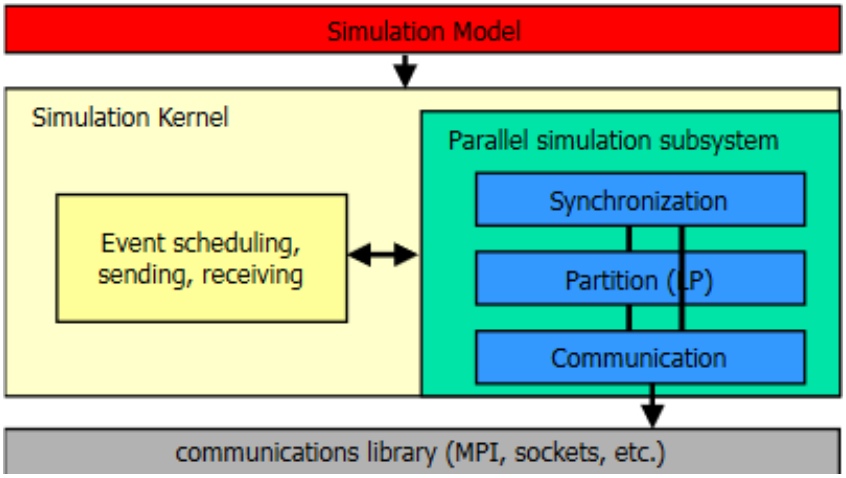
Παράλληλα, το INET κάνει χρήση και ενός διαφορετικού API, socket API, με σκοπό τον εμπλουτισμό των τυπικών διεπαφών διέλευσης μηνυμάτων για πρωτόκολλα επικοινωνίας. Τα sockets χρησιμοποιούνται για την δημιουργία διεπαφής μεταξύ εφαρμογών και πρωτοκόλλων με υπηρεσίες άλλων πρωτοκόλλων. Τα sockets διαμορφώνουν μια αμφίδρομη επικοινωνία μεταξύ των πρωτοκόλλων. Επιπλέον, μπορούν να δομούν και να στέλνουν αιτήματα υπηρεσιών και πακέτα και προφανώς να τα λαμβάνουν αντίστοιχα.
Εντολές όπως οι bind(), connect(), send(), close() ανήκουν στην γενικευμένη κλάση με όνομα **Socket**, και πραγματοποιούν λειτουργίες όπως δημιουργία και ρύθμιση socket, αποστολή και λήψη πακέτων, σε ένα ή περισσότερα socket ταυτοχρόνως [20].
Οι λειτουργίες των sockets χωρίζονται σε ομάδες ανάλογα με το πρωτόκολλο επικοινωνίας που θα εφαρμοστεί στην επικοινωνία όπως UdpSocket, TcpSocket, Ipv4socket και Ipv6socket.

4.2 OMNET++

Το OMNET++ είναι ένα αντικειμενοστραφή framework προσομοίωσης δικτύου διακριτών συμβάντων. Η αρχιτεκτονική του είναι γενικής φύσης και για αυτό μπορεί να παρουσιαστεί σε πολλές διαφορετικές περιπτώσεις. Μερικές από αυτές μπορεί να είναι ο σχεδιασμός ενσύρματων και ασύρματων δικτύων επικοινωνίας, μοντελοποίηση πρωτοκόλλων, διαμόρφωση δικτύων ουράς, κατασκευή και επικύρωση πολυεπεξεργαστών και άλλων εξαρτημάτων, αξιολόγηση απόδοσης περίπλοκων συστημάτων λογισμικού κ.α. [21].
Το OMNET++ δεν αποτελεί έναν προσομοιωτή για κάτι συγκεκριμένο, αλλά ένα ολοκληρωμένο περιβάλλον που παρέχει εργαλεία και υποδομές για την κατασκευή προσομοιώσεων. Ένα από τους κύριους παράγοντες της δομής του είναι η αρχιτεκτονική συστατικών για τα μοντέλα προσομοίωσης. Τα μοντέλα αποτελούν έναν συνδυασμό από επαναχρησιμοποιημένα στοιχεία που ονομάζονται ενότητες. Μια σωστά διαμορφωμένη ενότητα μπορεί να επαναχρησιμοποιηθεί σε μια πληθώρα περιπτώσεων.

Οι ενότητες μπορούν να συνδεθούν με άλλες μέσω κάποιων οντοτήτων που χαρακτηρίζονται ως πύλες, ώστε να χτίσουν πολυπαραγοντικές ενότητες. Δεν υπάρχει πραγματικό όριο στα σετ ενοτήτων που μπορούν να δημιουργηθούν. Οι ενότητες επικοινωνούν μεταξύ τους με την αμφίδρομη αποστολή μηνυμάτων, τα οποία περιέχουν αυθαίρετες δομές δεδομένων. Η αποστολή των μηνυμάτων μπορεί να γίνεται από προκαθορισμένες διαδρομές μέσω των πυλών ή με την απευθείας αποστολή στον παραλήπτη. Επίσης, χρησιμοποιούν παραμέτρους με στόχο τον καθορισμό της συμπεριφοράς των ενοτήτων και της τοπολογίας ενός μοντέλου. Η ενότητες του χαμηλότερου επιπέδου στην ιεραρχία χαρακτηρίζονται ως απλές ενότητες. Δουλειά τους είναι η ενθυλάκωση της συμπεριφοράς ενός μοντέλου. Το OMNET++ μπορεί να προσομοιώσει μια κατάσταση με διάφορους τρόπους, αλλά η πιο διαδεδομένη και χρήσιμη είναι με την παρουσία γραφικών μοντέλων. Διευκολύνουν την κατανόηση του πειράματος και την αποφυγή σφαλμάτων.

Ακόμη, το OMNET++ υποστηρίζει παράλληλη κατανεμημένη προσομοίωση, με την χρήση MPI ή κάποιων pipelines που επιτρέπουν την επικοινωνία μεταξύ των τμημάτων των κατανεμημένων προσομοιώσεων. Ο αλγόριθμος που συγκρατεί την παράλληλη προσομοίωση μπορεί αρκετά εύκολα είτε να επεκταθεί είτε να αντικατασταθεί με κάποιον άλλον. Τα μοντέλα επίσης, δεν έχουν την ανάγκη να ενορχηστρωθούν αντίστοιχα με την παράλληλη προσομοίωση, παρά μόνο να διαμορφωθούν κατάλληλα.

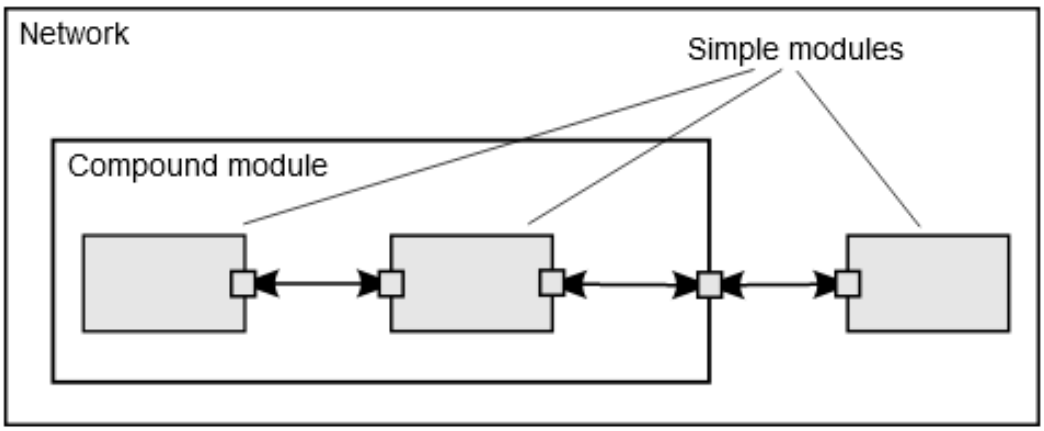


Εικόνα 21 Αρχιτεκτονική μοντέλου παράλληλης προσομοίωσης

Πηγή: https://doc.omnetpp.org/workshop2008/omnetpp40-paper.pdf σελ.4.

Η δημιουργία του OMNET++ βασίζεται στην δημιουργία και την υποστήριξη προσομοιώσεων δικτύων επικοινωνίας, πράγμα που σημαίνει πως προαπαιτείται η εκπλήρωση κάποιον προϋποθέσεων για την διασφάλιση της απόδοσης του. Οι προσομοιώσεις πρέπει να είναι πολυπαραγοντικές, οπότε η διαχείριση

των μοντέλων που συμμετέχουν στο πείραμα πρέπει να στηρίζεται σε μια ιεραρχία που κατατάσσει τα μοντέλα με τρόπο τέτοιο ώστε να αποφευχθεί οποιαδήποτε σύγκρουση προτεραιότητας μεταξύ τους. Επίσης, ένα πρόγραμμα που είναι αφιερωμένο στις προσομοιώσεις, πρέπει να είναι κατασκευασμένο κατάλληλα ώστε να παρέχει άμεση ανίχνευση και ικανότητα επίλυσης λαθών. Κατά την διεκπεραίωση ενός πειράματος δεν είναι πρέπον οι ερευνητές να ξοδεύουν πολύ χρόνο στην καταπολέμηση λαθών τυπικού επιπέδου. Ένα πρόγραμμα προσομοίωσης πρέπει να επιτρέπει την προσαρμογή των επιλογών που παρέχει αλλά και την σύνδεση του με άλλα εργαλεία που θα αυξήσουν την ποιότητα και την ακρίβεια του πειράματος. Όπως προαναφέρθηκε, το OMNET++ το επιτυγχάνει με την συνεργασία του με προσομοιωτές όπως το SUMO. Τέλος, τα αρχεία που παράγονται από μια προσομοίωση και τα αρχεία μπορεί να ενταχθούν σε μία, θα πρέπει να είναι ελεύθερα προς επεξεργασία από άλλες εφαρμογές [22].
Το OMNET++ είναι μια σύνθεση από ενότητες οι οποίες επικοινωνούν με την ανταλλαγή μηνυμάτων. Οι βασικές ενότητες χαρακτηρίζονται ως απλές ενότητες, και ανήκουν στις βιβλιοθήκες κλάσεων τις γλώσσας προγραμματισμού C++. Κάθε απλή ενότητα μπορεί να αναμιχθεί με άλλες και να συνθέσουν μια σύνθετη κλάση. Οι κλάσεις κατατάσσονται σε μια ιεραρχία τις οποίας τα επίπεδα είναι άπειρα. Τα μηνύματα μεταξύ κλάσεων μεταφέρονται είτε απευθείας από μία κλάση σε άλλη είτε μιας προκαθορισμένης διαδρομής, ειδικά όταν ασχολούμαστε με ασύρματες επικοινωνίες. Το παρακάτω σχήμα αναπαριστά πως μια απλή ενότητα μπορεί να συνδεθεί με άλλες, επικοινωνώντας απευθείας μεταξύ τους.



Εικόνα 22 Συσχετισμός ενοτήτων στο OMNET++

Πηγή: https://doc.omnetpp.org/omnetpp/manual/ εν.2.1 στο manual.

Η ανώτερη ενότητα στην σκάλα της ιεραρχίας είναι η ενότητα συστήματος. Η ενότητα αυτή εμπεριέχει υποενότητες οι οποίες με την σειρά τους μπορεί να περιέχουν άλλες υποενότητες και ούτω καθεξής. Κάθε

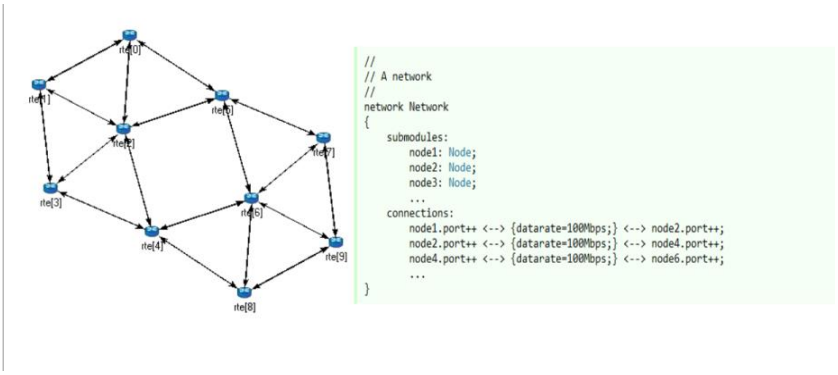
ενότητα που περιέχει άλλες ενότητες χαρακτηρίζεται ως σύνθετη ενότητα και λειτουργώντας με μια διάταξη από πάνω προς τα κάτω φτάνουμε στα κατώτερα επίπεδα της ιεραρχίας με τις απλές ενότητες που διατηρούν τους αλγορίθμους των μοντέλων.

Τα μηνύματα που ανταλλάζονται μεταξύ των απλών ενοτήτων είναι στην ουσία πλαίσια ή πακέτα αν πρόκειται για ένα δίκτυο επικοινωνίας, διεργασίες σε ένα δίκτυο ουράς ή κάποια άλλη φορητή οντότητα. Κάθε μήνυμα εμπεριέχει μια περίπλοκη σύνθεση από δεδομένα και στέλνεται είτε απευθείας στον παραλήπτη είτε μέσω προκαθορισμένης διαδρομής. Ο χρόνος της προσομοίωσης προχωρά όσο μια ενότητα παραλαμβάνει ένα μήνυμα. Το μήνυμα μπορεί να σταλεί από άλλη μονάδα ή και από την ίδια στον εαυτό της, σε μια προσπάθεια να υλοποιηθεί χρονοδιακόπτης στην προσομοίωση. Κάθε ενότητα έχει πύλες εισόδου και εξόδου, από τις οποίες κάθε μήνυμα εισέρχεται και εξέρχεται αντίστοιχα. Κάθε σύνδεση που προκύπτει μεταξύ απλών ενοτήτων, υφίσταται μόνο σε ένα μεμονωμένο στρώμα της ιεραρχίας τους. Μέσα στο δεδομένο μοντέλο της ιεραρχίας, τα μηνύματα κινούνται προς πάσα κατεύθυνση μέσω των απλών ενοτήτων, ενώ η σύνθετες ενότητες λειτουργούν σαν ένα δοχείο που παραδίδουν ένα μήνυμα στις εσωτερικές τους απλές ενότητες και έπειτα σε άλλες εξωτερικές.

Γλώσσα Προγραμματισμού NED

Η γλώσσα NED ή Network Description δίνει την δυνατότητα στον χρήστη να περιγράψει την δομή ενός μοντέλου προσομοίωσης δικτύου. Μέσω της γλώσσας NED γίνεται η κατασκευή ενοτήτων και η σύνδεση τους σε σύνθετες. Η NED υποστηρίζει την ύπαρξη της ιεραρχίας των ενοτήτων στο OMNET++ με σκοπό την αποφυγή επίλυσης πολύπλοκων προβλημάτων. Επίσης, η γλώσσα υποστηρίζει την εισαγωγή κι άλλων framework όπως το INET για συνεργασία. Ακόμη, η NED υποστηρίζει τεχνικές κληρονόμησης για ενότητες και κανάλια. Κάθε διαιρεμένο στοιχείο μπορεί να χρησιμοποιήσει καινούριες παραμέτρους και πύλες και στην περίπτωση των σύνθετων, μπορούν να αποκτήσουν καινούριες απλές μονάδες και κανάλια. Η NED υποστηρίζει δομές πακέτων τύπου Java, με σκοπό την ελάττωση των συγκρούσεων των μοντέλων για λόγους ονόματος. Κάθε απλό στοιχείο που χρησιμοποιείται μόνο τοπικά από ένα σύνθετο, μπορεί να ορίσει το scope του τοπικά, ώστε να μειωθεί η χαοτική εμφάνιση ονομάτων.

Παραδείγματα



Εικόνα 23 Δίκτυο κατασκευασμένο στην γλώσσα NED

Πηγή: https://doc.omnetpp.org/omnetpp/manual/ εν. 3.2.1 στο manual.

Το παραπάνω κολλάζ από στιγμιότυπα απεικονίζουν ένα εικονικό δίκτυο με δέκα κόμβους και δεξιά του τον κώδικα σε γλώσσα NED που το περιγράφει. Κάθε τμήμα κώδικα που περιβάλλεται από **//** υποδηλώνει πως πρόκειται για σχόλιο. Για να μπορεί ο κώδικας να εντοπίζεται από άλλα αρχεία διαμόρφωσης προσθέτουμε τον χαρακτηρισμό Network, οπού σε οποιαδήποτε άλλη περίπτωση θα μπορούσε να είναι κάποιο άλλο όνομα. Στα submodules ορίζεται η κατανομή των κόμβων από τον πρώτο προς τον τελευταίο και δίνεται ο τύπος που το χαρακτηρίζει ως κόμβο από την ενότητα **Node.** Τέλος, στα connections καθορίζεται πως θα συνδέονται οι κόμβοι μεταξύ τους. Κάθε βέλος με δύο άκρες ορίζει την αμφίδρομη επικοινωνία και κάθε **port++** αντιστοιχεί στην ενεργοποίηση μιας νέας πύλης στον κόμβο. Παράλληλα, ορίζεται ο ρυθμός αποστολής δεδομένων **datarate** με τιμή 100Mbps.

Ωστόσο, η μέθοδος της επαναλαμβανόμενης γραφής κώδικα που αφορά τον ρυθμό αποστολής δεδομένων μπορεί να φανεί αρκετά κουραστική και χρονοβόρα. Για αυτό, η NED μπορεί να ενθυλακώσει πληροφορίες που αφορούν ένα κανάλι και να τις αναπαράγει με την χρήση μιας μεταβλητής (C).

```
//
// A Network
//
network Network
{
    types:
        channel C extends ned.DatarateChannel {
            datarate = 100Mbps;
        }
    submodules:
        node1: Node;
        node2: Node;
        node3: Node;
        ...
    connections:
        node1.port++ <--> C <--> node2.port++;
        node2.port++ <--> C <--> node4.port++;
        node4.port++ <--> C <--> node6.port++;
        ...
}
```

Εικόνα 24 Κατασκευή καναλιού στην γλώσσα NED

Πηγή: https://doc.omnetpp.org/omnetpp/manual/ εν.3.2.2 στο manual.

Προγραμματιστικά, οι απλές ενότητες υλοποιούνται με την χρήση της λέξης **simple** πριν την δήλωση του ονόματος τους. Μέσα σε κάθε πλαίσιο δήλωσης μια απλής ενότητας μπορούν να δηλωθούν παράμετροι που αφορούν χαρακτηριστικά τους και πύλες εισόδου και εξόδου για την ανταλλαγή μηνυμάτων. Ωστόσο, και στις δύο περιπτώσεις δεν είναι υποχρεωτική η δήλωση τους.

```
simple Queue
{
    parameters:
        int capacity;
        @display("i=block/queue");
    gates:
        input in;
        output out;
}
```

Εικόνα 25 Κατασκευή απλής ενότητας

Πηγή: https://doc.omnetpp.org/omnetpp/manual/ ενότητα 3.3 στο manual.

Μια σύνθετη ενότητα μπορεί να περιέχει πάλι όσα περιέχει μια απλή αλλά πλέον εμπεριέχει και τις υποενότητες που μπορεί να αποτελείται. Κάθε υποενότητα δηλώνεται κάτω από την κατηγορία **submodules.** Αναλόγως, οι συνδέσεις δηλώνονται κάτω από την κατηγορία **connections.** Η συμπεριφορά της σύνδεσης μπορεί να καθοριστεί μέσω την ενοποίησης της με το κανάλι. Οι τύποι των ενοτήτων και των καναλιών που χρησιμοποιούνται δηλώνονται στην κατηγορία **types** και ισχύουν τοπικά. Κάθε σύνθετη ενότητα μπορεί να επεκταθεί μέσω κληρονόμησης ενοτήτων και συνδέσεων από άλλες ενότητες.

Είναι σημαντικό να αναφερθεί πως η γενική δήλωση μιας σύνθετης ενότητα περιλαμβάνει το statement **module** πριν το όνομα της.

```
module Host
{
    types:
        ...
    parameters:
        ...
    gates:
        ...
    submodules:
        ...
    connections:
        ...
}
```

Εικόνα 26 Κατασκευή σύνθετης ενότητας

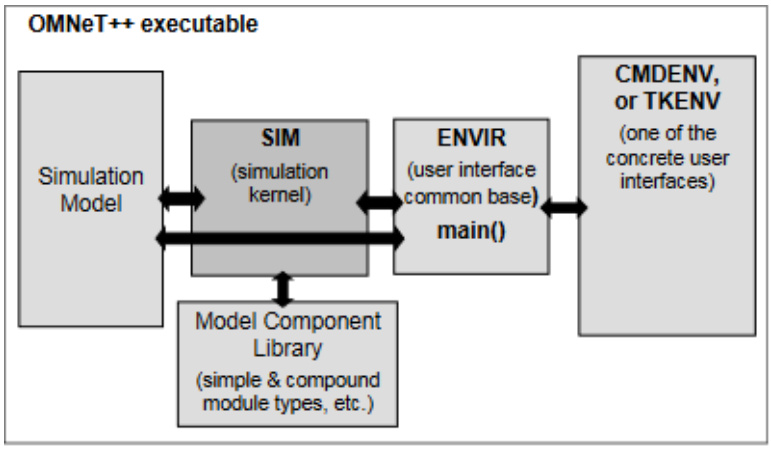
Πηγή: https://doc.omnetpp.org/omnetpp/manual/ εν. 3.4 στο manual.

Όλες οι παραπάνω πληροφορίες και οτιδήποτε άλλο μπορεί να αφορά την γλώσσα NED, έχει αναλυθεί πλήρως από τον σχετικό οδηγό του omnet++ που βρίσκεται δωρεάν στο διαδίκτυο. Αυτά τα παραδείγματα είναι μόνο κάποιες στοιχειώδεις αναφορές επάνω στην αρχιτεκτονική που θα έχει ένα πρόγραμμα της NED.

Περιβάλλον Προσομοίωσης

Για την εκκίνηση εκτέλεσης μιας προσομοίωσης, πρέπει πρώτα να εγκατασταθούν οι ενότητες, απλές και σύνθετες, των οποίων ο κώδικας έχει μεταγλωττιστεί σε κώδικα μηχανής. Έπειτα, το μοντέλο προσομοίωσης που έχει διαμορφωθεί χτίζεται από τον kernel της προσομοίωσης και από την βιβλιοθήκη της κλάσης **Sim**. Η εκτέλεση της προσομοίωσης γίνεται σε περιβάλλον που παρέχεται από κάποιες βιβλιοθήκες διεπαφής χρήστη όπως το **Envir**, το **Cmdenv** και το **Tkenv**. Τα τρία αυτά πλαίσια καθορίζουν την προέλευση των εισερχόμενων δεδομένων, τον προορισμό των αποτελεσμάτων της προσομοίωσης, τον τρόπο επεξεργασίας των αποτελεσμάτων του debugging του μοντέλου κ.α. [22].

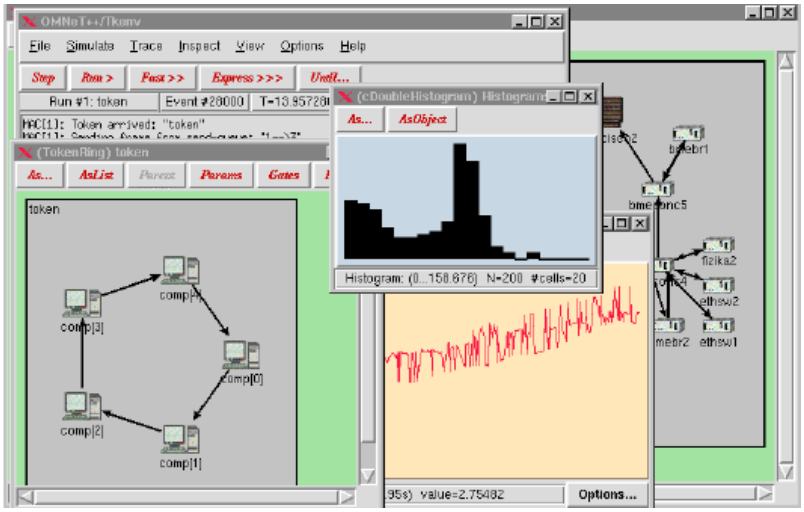
Με την εναλλαγή της βιβλιοθήκης διεπαφής χρήστη, ο χρήστης μπορεί να διαμορφώσει ένα πλήρες περιβάλλον στο οποίο θα τρέξει το πείραμα του. Αυτό επιτυγχάνεται μέσω της διεπαφής που βρίσκεται μεταξύ της Sim και των άλλων βιβλιοθηκών μιας και η κάθε μία λειτουργεί αυτόνομα από τις άλλες.



Εικόνα 27 Αρχιτεκτονική του OMNET++

Πηγή: https://doc.omnetpp.org/workshop2008/omnetpp40-paper.pdf σελ.4.

Για την δημιουργία του OMNET++ , υπήρξαν προαπαιτήσεις που αφορούσαν την λειτουργία του όπως η εύκολη ανίχνευση λαθών. Σχετικά στοιχεία έχουν υλοποιηθεί στο περιβάλλον Tkenv που αποτελεί το graphical user interface του OMNET++. Το Tkenv κάνει χρήση τριών βασικών τεχνικών: αυτόματα κινούμενα γραφικά, παράθυρα εμφάνισης αποτελεσμάτων και διερευνητές αντικειμένων. Με τα αυτόματα κινούμενα γραφικά μπορούν να απεικονιστούν οι ροές μεταφοράς μηνυμάτων και οι αλλαγές στην κατάσταση των κόμβων, χωρίς την σύνταξη κώδικα από τον χρήστη. Με τα παράθυρα εμφάνισης αποτελεσμάτων μπορούμε να εμφανίσουμε ξεχωριστά παράθυρα με τα αποτελέσματα για κάθε μία διαφορετική ενότητα ή ενός συνόλου ενοτήτων.

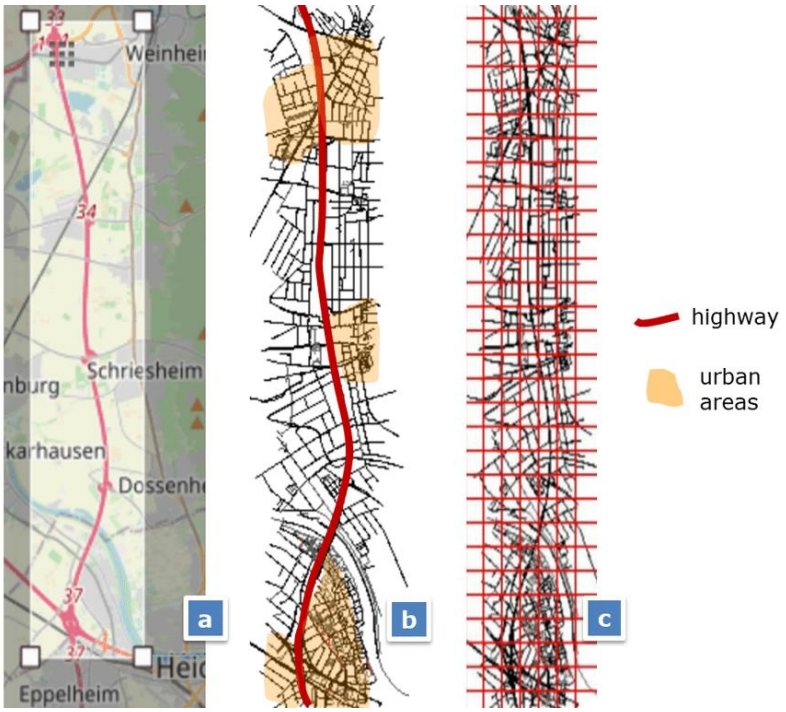


Εικόνα 28 Περιβάλλον χρήσης Tkenv

Πηγή: https://www.researchgate.net/figure/Screenshot-of-the-Tkenv-User-Interface-of-OMNeT_fig1_228460521

4.3 SUMO

Από τις αρχές του 2001, το γερμανικό διαστημικό κέντρο (DLR) είχε ξεκινήσει την δημιουργία του πακέτου προσομοίωσης SUMO. Το SUMO είναι ένα πακέτο ανοιχτού κώδικα, που υλοποιεί ολοκληρωμένα μοντέλα προσομοίωσης γεγονότων σε μεγάλα εικονικά οδικά δίκτυα. Η σουίτα του SUMO αποτελεί την πιο διαδεδομένη στον τομέα της προσομοίωσης οδικών δικτύων μιας και προηγούμενα προγράμματα βρισκόντουσαν σε πειραματικό στάδιο από ερευνητές και δεν λάμβαναν ουσιαστικές ενημερώσεις. Στην ουσία, επιτρέπει την δημιουργία εικονικών οχημάτων μικρής κλίμακας και την δρομολόγηση προκαθορισμένων διαδρομών πάνω σε ένα τυχαίο εικονικό οδικό δίκτυο. Τα μοντέλα που εμπεριέχονται στην σουίτα σχετίζονται με κάθε σημερινό παράγοντα που συναντά κανείς σε ένα πραγματικό δρόμο, όπως για παράδειγμα τα φανάρια. Τα εικονικά οδικά δίκτυα μπορούν να παραχθούν είτε μέσω μιας εφαρμογής ονόματι **netgen** είτε με την εισαγωγή ψηφιακών τμημάτων χαρτών πραγματικών δικτύων. Επίσης, με την εφαρμογή **netconvert** ο χρήστης μπορεί να εισάγει και χάρτες που έχουν κατασκευαστεί με άλλες εφαρμογές προσομοίωσης όπως και αρχεία με γεωχωρικά διανυσματικά δεδομένα ονόματι shapefiles [24].



Εικόνα 29 Αναπαράσταση εισαγωγής χάρτη στο περιβάλλον του SUMO

Πηγή:https://www.researchgate.net/figure/Real-map-used-a-street-map-imported-in-Sumo-b-and-grid-cell-division-of-the-map-c_fig2_345943060

Το παραπάνω παράδειγμα απεικονίζει πως ένα τμήμα πραγματικού ψηφιακού χάρτη μετατρέπεται σε χάρτη διαχειρίσιμο από το SUMO και πως ο εισακτέος χάρτης διαιρείται σε «κυψέλες» μέσω του πλέγματος.

Κάθε όχημα που θα χρησιμοποιηθεί στην προσομοίωση επιλέγεται μεμονωμένα και ξεχωρίζεται από τα υπόλοιπα μέσω ενός identifier (id). Ξεχωριστά για κάθε όχημα μπορεί να ρυθμιστεί η στιγμή αναχώρησης από το αρχικό σημείο, η ταχύτητα του οχήματος, ο δρόμος τον οποίο θα ακολουθήσει το όχημα κλπ. Ακόμη, κάθε όχημα μπορεί να ανήκει σε μια συγκεκριμένη κλάση που αφορά την ηχορύπανση που προκαλεί αλλά και τους ρύπους που απελευθερώνει.

Περισσότερες πληροφορίες που αφορούν το SUMO και την κατασκευή ενός ολοκληρωμένου οδικού δικτύου ακολουθούν στο κεφάλαιο 5, στην ενότητα 5.3.

vehicle:0 Parameter

Name	Value	Dynamic
lane [id]	gneE0_0	✓
position [m]	100.00	
lateral offset [m]	0.00	
speed [m/s]	12.63	
lateral speed [m/s]	0.00	
acceleration [m/s^2]	0.00	
angle [degree]	90.00	
slope [degree]	0.00	
speed factor	1.00	
time gap on lane [s]	-1.00	
waiting time [s]	0.00	
waiting time (accumulated, 100.00s) [s]	0.00	
time loss [s]	0.00	
impatience	0.00	
last lane change [s]	0.00	
desired depart [s]	0.00	✗
depart delay [s]	0.00	✗
odometer [m]	0.00	
remaining [#]	0	✗
stop info	next: lane:gneE0_0 pos:220.00	✓
line		✗
CO2 [mg/s]	2238.03	
CO [mg/s]	0.24	
HC [mg/s]	0.05	
NOx [mg/s]	0.69	
PMx [mg/s]	0.01	
fuel [ml/s]	0.96	
electricity [Wh/s]	0.00	
noise (Harmonoise) [dB]	63.68	
devices		✗
persons	0	
containers	0	
lcState right	unknown	✓
lcState left	unknown	✓

Εικόνα 30 Παράμετροι οχήματος

Πηγή: https://sumo.dlr.de/docs/sumo-gui.html

5. Αλγόριθμοι Ομαδοποίησης

Ομαδοποίηση ή clustering ονομάζεται μία διαδικασία διαμοιρασμού ενός συνόλου από πληροφορίες σε διαφορετικές ομάδες. Οι ομάδες που λαμβάνουν την εκάστοτε πληροφορία αποτελούνται από στοιχεία, τα οποία καθορίζουν τον χαρακτήρα ή τον σκοπό της κάθε ομάδας. Μία ομάδα από στοιχεία μοιράζεται ένα κοινό κομμάτι πληροφορίας που όμως δεν αφορά καμία άλλη ομάδα στοιχείων. Οι ομάδες κατηγοριοποιούνται ανάλογα με τις ανάγκες του προβλήματος που έχει παρουσιαστεί. Τα στοιχεία της ομάδας που συλλέγονται χρησιμοποιούν σαν γνώμονα την ολοκληρωτική ή σχετική ομοιότητα τους με τα υπόλοιπα στοιχεία. Αντίστοιχα, υπολογίζεται και κατά πόσο ένα στοιχείο διαφέρει από το άλλο ως προς την δομή του και τις ανάγκες. Έτσι, συντελούνται οι οριστικές ομάδες από έναν αλγόριθμο με τα στοιχεία να έχουν άμεση σχέση μεταξύ τους και μόνο [32].

5.1 Αναπαράσταση παραδειγμάτων αλγόριθμων ομαδοποίησης

Ο αλγόριθμος ομαδοποίησης είναι ουσιαστικά ο μηχανισμός στον οποίο βασίζεται όλη η τεχνική ομαδοποίησης των στοιχείων που συλλέγονται. Κατά πλειοψηφία, οι αλγόριθμοι ομαδοποίησης θα ανήκουν σε δύο μεγάλες οικογένειες αλγορίθμων: Είτε στις ιεραρχικές είτε στις διαμεριστικές.

Ξεκινώντας με τους ιεραρχικούς αλγόριθμους ομαδοποίησης, σκοπός μας είναι να φτιάξουμε ομάδες έχοντας για ξεκίνημα στοιχεία που αποτελούν μία ομάδα από μόνα τους. Ύστερα, θα μπορέσουμε να ενώσουμε τα στοιχεία σε ομάδες φτάνονται στο τελικό αποτέλεσμα.

Για αρχή, πρέπει να ορίσουμε έναν τρόπο ώστε να δημιουργούνται σημεία μέσα σε ένα πεδίο για το σύνολο των στοιχείων που έχουμε συλλέξει για ομαδοποίηση.

```
// Δομή που αντιπροσωπεύει ένα σημείο στον δισδιάστατο χώρο
struct Point {
   double x, y; // x,y αποτελούν συντεταγμένες για έναν δισδιάστατο χώρο.
};
```

Έπειτα, πρέπει να μπορούμε να υπολογίσουμε την απόσταση μεταξύ δύο σημείων.

```
// Συνάρτηση για τον υπολογισμό της απόστασης μεταξύ δύο σημείων
double Distance(const Point& p1, const Point& p2) {
   return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}
```

Κώδικας 1 Δομή για ιεραρχικό αλγόριθμο ομαδοποίησης

Αφότου έχει γνωστοποιηθεί πως εργαζόμαστε με έναν αλγόριθμο ιεραρχικής ομαδοποίησης, πρέπει πρώτα να χωρίσουμε το κάθε στοιχείο σε μεμονωμένη ομάδα και στη συνέχεια κατά την εκτέλεση του αλγορίθμου να ομαδοποιηθούν μέχρι τις τελικές ομάδες.

```
   // Κάθε σημείο αρχικά αποτελεί το δικό του cluster
   for (size_t i = 0; i < points.size(); ++i) {
      clusters[i].push_back(points[i]);}
```

Κώδικας 2 Διαχωρισμός στοιχείων

Στην συνέχεια, επιχειρούμε να εκτελέσουμε το πρώτο βήμα ομαδοποίησης με τα στοιχεία να φτιάχνουν τις πρώτες σύνθετες ομάδες. Η διαδικασία συνεχίζει μέχρι να δημιουργηθεί ο αριθμός ομάδων που έχουμε επιλέξει. Στην συγκεκριμένη περίπτωση είναι δύο.

```
// Ενώνοντας clusters μέχρι να μείνουν τόσα όσα ορίζει το numClusters
while (clusters.size() > numClusters) {
    double minDistance = numeric_limits<double>::max(); // Αρχικοποίηση με την μέγιστη τιμή double
    pair<size_t, size_t> minPair;  // Ζεύγος που αποθηκεύει τα clusters με την ελάχιστη απόσταση
```

Κώδικας 3 Ένωση clusters

Με την παρουσία νέων ομάδων κατά την εκτέλεση του πρώτου βήματος του αλγορίθμου, έχουν παρουσιαστεί νέες ομάδες που απαιτούν εκ νέου υπολογισμό απόστασης μεταξύ τους, για να φτιάξουν καινούριες ομάδες.

```
    // Υπολογισμός της απόστασης μεταξύ κάθε ζεύγους clusters
    for (size_t i = 0; i < clusters.size(); ++i) {
        for (size_t j = i + 1; j < clusters.size(); ++j) {
            double distance = Distance(clusters[i][0], clusters[j][0]);
            if (distance < minDistance) {
                minDistance = distance;
                minPair = {i, j};  // Αποθήκευση του ζεύγους με την μικρότερη απόσταση}}}
```

Κώδικας 4 Υπολογισμός απόστασης

Επομένως, το επόμενο βήμα είναι να συγχωνευτούν ξανά σε νέες ομάδες κρίνοντας από την απόσταση που έχουν μεταξύ τους.

```
// Συγχώνευση των δύο clusters που έχουν την μικρότερη απόσταση
clusters[minPair.first].insert(
   clusters[minPair.first].end(),
   clusters[minPair.second].begin(),
   clusters[minPair.second].end()  );
```

Κώδικας 5 Συγχώνευση clusters

Αυτή η διαδικασία, μπορεί να επαναληφθεί όσες φορές είναι αναγκαίο ώστε να συγχωνεύσουμε όλο το πλήθος κάποιων στοιχείων στον επιθυμητό αριθμό ομάδων που θέλουμε. Στην περίπτωση αυτή, το αποτέλεσμα είναι δύο ομάδες που δημιουργήθηκαν μεταξύ έξι διαφορετικών στοιχείων.

Πίνακας 1 Αποτελέσματα αλγόριθμου ιεραρχικής ομαδοποίησης

Cluster	Στοιχεία με συντεταγμένες(x,y)
Cluster 1	(1, 1) (2, 2) (3, 3)
Cluster 2	(8, 8) (9, 9) (10, 10)

Αντιθέτως, οι διαμεριστικοί αλγόριθμοι ομαδοποίησης απαιτούν μια διαφορετική προσέγγιση. Για τους διαμεριστικούς αλγόριθμους ομαδοποίησης, κάθε στοιχείο που συμπεριλαμβάνεται δεν μπορεί να ανήκει παράλληλα και σε κάποιο άλλο υποσύνολο ή ομάδα όπως γίνεται με τους ιεραρχικούς. Κάθε ομάδα είναι αυτόνομη και δεν σχετίζεται με τις υπόλοιπες αφού επεξεργαστούν από τον αλγόριθμο, όπως επίσης και κάθε ένα στοιχείο δεν σχετίζεται με άλλο από ξένη ομάδα.

Στην εργασία θα ασχοληθούμε με τον αλγόριθμο κ-μέσων που είναι άλλωστε και ο πιο διαδεδομένος διαμεριστικός αλγόριθμος. Ο αλγόριθμος χρησιμοποιεί κέντρα (k) για να συγχωνεύσει στοιχεία βάσει της κοντινότερης απόστασης.

Η διαδικασία αρχικά είναι πανομοιότυπη με του αλγόριθμου ιεραρχικής ομαδοποίησης, θέτοντας τα σημεία των στοιχείων σε δισδιάστατο χώρο και υπολογίζοντας την απόσταση μεταξύ τους.

Πρώτα πρέπει να ορίσουμε τα κέντρα των ομάδων. Στην περίπτωση αυτή, οι ομάδες θέλουμε πάλι να είναι δύο οπότε δύο θα είναι και τα κέντρα. Τα αρχικά κέντρα θα είναι τυχαία και θα επιχειρήσουμε να σχετίσουμε με αυτά τα υπόλοιπα στοιχεία βάση κοντινότερης απόστασης.

```
// Αρχικοποίηση των κέντρων τυχαία
srand(time(0));
for (int i = 0; i < k; ++i) {
   centroids[i] = points[rand() % points.size()];
   // Ανάθεση κάθε σημείου στο πλησιέστερο κέντρο
   for (const auto& point : points) {
      int nearestCentroid = 0;
      double minDistance = Distance(point, centroids[0]);
      for (int i = 1; i < k; ++i) {
         double distance = Distance(point, centroids[i]);
         if (distance < minDistance) {
            minDistance = distance;
            nearestCentroid = i;  // Βρίσκουμε το πλησιέστερο κέντρο}}
clusters[nearestCentroid].push_back(point);  // Προσθήκη του σημείου στο πλησιέστερο cluster
```

Κώδικας 6 Ορισμός κέντρων αλγορίθμου κ-μέσων

Όσο η διαδικασία συνεχίζει, θα υπολογίζονται νέα κέντρα, μέχρι να μην μπορούν να οριστούν καινούρια επειδή τα στοιχεία δεν βρίσκουν κοντινότερο κέντρο να συγχωνευθούν, μιας και η κίνηση όλων των στοιχείων στον χώρο είναι συνεχής παράλληλα με την διαδικασία της ομαδοποίησης.

```
// Επαναυπολογισμός των κέντρων
for (int i = 0; i < k; ++i) {
   Point newCentroid = calculateCentroid(clusters[i]);
   if (Distance(newCentroid, centroids[i]) > 0.0001) {
      centroids[i] = newCentroid;  // Ενημέρωση του κέντρου
```

Κώδικας 7 Υπολογισμός κέντρων εκ νέου

Το παραγόμενο αποτέλεσμα του προαναφερόμενου υπολογισμού δεν διαφέρει αισθητά από τον ιεραρχικό.

Πίνακας 2 Αποτελέσματα αλγόριθμου κ-μέσων

Clusters	Στοιχεία με συντεταγμένες(x,y)
Cluster 1	(10, 10) (9, 9) (8, 8)
Cluster 2	(1, 1) (2, 2) (3, 3)

Και τα δύο παραδείγματα αποτελούν μόνο επιφανειακές προσεγγίσεις για το θέμα των αλγόριθμων ομαδοποίησης και δεν αντιστοιχούν σε εφαρμογές για λύσεις πραγματικών προβλημάτων, ωστόσο παραδίδουν μία βάση για το πως μπορούμε να σκεφτόμαστε κατά την υλοποίηση πραγματικών αλγόριθμων ομαδοποίησης.

5.2 Εφαρμογή αλγορίθμων ομαδοποίησης για VANETs

Ένας αλγόριθμος ομαδοποίησης μπορεί να έχει εφαρμογή και σε ένα δίκτυο οχημάτων (VANET) με σκοπό την συλλογή και διαμοίραση δεδομένων σε μεμονωμένες ομάδες οχημάτων που αφορούν ένα συγκεκριμένο συμβάν. Δηλαδή, ένα σύνολο από οχήματα μπορούν να καταμεριστούν σε ομάδες που δεν

σχετίζονται απαραίτητα μεταξύ τους, διότι κάθε ομάδα διαχειρίζεται και επεξεργάζεται άλλο μήνυμα/προειδοποίηση που αφορά το οδικό δίκτυο. Άλλωστε σε ένα ρεαλιστικό παράδειγμα, δεν μπορούν όλα τα οχήματα σε ένα συγκεκριμένο δίκτυο οχημάτων να «στεγάζονται» υπό την ίδιο ομάδα αφού κάθε σημείο που βρίσκεται ένα όχημα μια δεδομένη στιγμή, δεν μπορεί άμεσα να αφορά και όλα τα υπόλοιπα οχήματα. Σαφώς, κάτι τέτοιο θα δημιουργούσε μεγάλη αναστάτωση στους οδηγούς που αναμένουν να λάβουν μόνο την ειδοποίηση που χρειάζονται, μιας και θα βομβαρδίζονταν από προειδοποιητικά μηνύματα που δεν τους αφορά. Για αυτό, ένας αλγόριθμος που μπορεί να καταμερίσει σωστά τα οχήματα αυτά σε ομάδες μεμονωμένες θα ορίσει την σωστή διανομή των δεδομένων που θα ανταλλάσσονται μεταξύ των άμεσα εμπλεκόμενων στοιχείων της ομάδας. Επομένως, κάθε ένα στοιχείο μιας ομάδας θα σχετίζεται μόνο με τα υπόλοιπα βάσει της υπηρεσίας που θα ζητήσει να έχει.

Με την χρήση του VEINS είναι εφικτή η αναπαράσταση ενός εικονικού πειράματος κατά το οποίο διαδραματίζεται σε πραγματικό χρόνο το αποτέλεσμα που μπορεί να επέλθει με την εφαρμογή αλγορίθμων ομαδοποίησης σε ένα δίκτυο οχημάτων όπου τα στοιχεία-οχήματα λαμβάνουν πληροφορίες, και ανάλογα το είδος της ανάγκης και της πληροφορίας που πρέπει να λάβει το κάθε ένα όχημα-στοιχείο, τα στοιχεία αυτά ομαδοποιούνται σε κατάλληλες ομάδες στις οποίες θα σχετίζονται όσο το δυνατό περισσότερο.

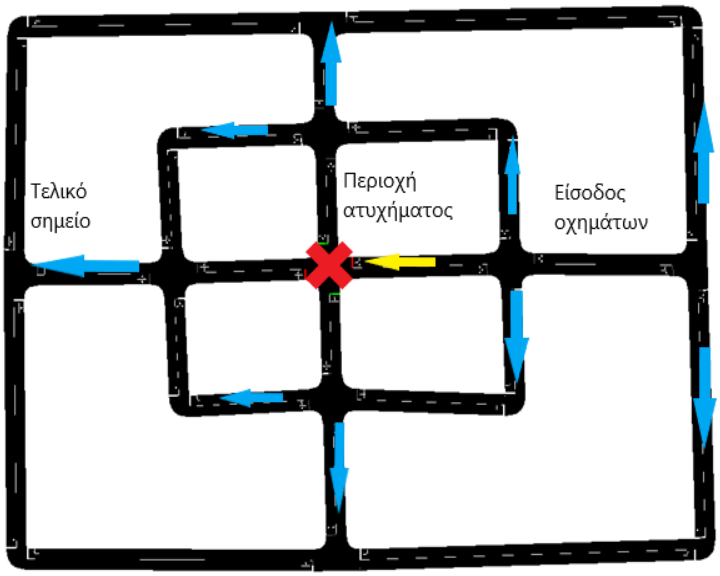
5.3 Υλοποίηση με VEINS framework

Η υλοποίηση που πραγματεύεται η εργασία θέτει σε εφαρμογή την συνεργασία των προαναφερόμενων εργαλείων του VEINS. Εφαρμόζεται η συνεργασία του SUMO και του OMNET++, τα οποία αντίστοιχα συμπεριλαμβάνουν επιμέρους κλάδους υλοποίησης δικτύων αλγορίθμων όπως η γλώσσα NED και η διεπαφή TraCI του SUMO, με σκοπό την προσπάθεια υλοποίησης ενός συνολικού εγχειρήματος που αφορά την εκ νέου δημιουργία δρομολόγησης ενός συνόλου από οχήματα μέσα σε ένα οδικό δίκτυο για την αποφυγή συμφόρησης στους δρόμους. Η υλοποίηση συνδυάζει αυτό το εγχείρημα με το να επιχειρήσει να ομαδοποιήσει τα οχήματα σε clusters μέσω αλγόριθμου ομαδοποίησης που θα βασιστεί στο μήνυμα που θα λάβει το κάθε όχημα για την περίπτωση ενός ατυχήματος σε κεντρικό σταυροδρόμι.

5.3.1 Ανάλυση υλοποίησης σε περιβάλλον προσομοίωσης του SUMO

Στο περιβάλλον προσομοίωσης του SUMO, ορίζεται ένα εικονικό οδικό δίκτυο στο οποίο κινούνται σε σειρά διάφορα οχήματα και εισέρχονται στο τετραγωνικό οδικό δίκτυο από την δεξιά πλευρά το

σχήματος. Ο σκοπός των οχημάτων είναι να διασχίσουν το κεντρικό σταυροδρόμι του τετραγώνου και να φτάσουν στην απέναντι πλευρά. Όμως, στο κεντρικό σταυροδρόμι έχει συμβεί ένα ατύχημα το οποίο εμποδίζει την διέλευση οχημάτων. Το πρώτο όχημα της σειράς (κίτρινο βελάκι) φτάνει νωρίτερα στο σταυροδρόμι από τα υπόλοιπα οχήματα αντικρίζοντας το εμπόδιο. Τα υπόλοιπα οχήματα, αφού δεν έχουν φτάσει ακόμα έχουν την ευκαιρία να επιλέξουν ένα διαφορετικό δρόμο προς την απέναντι πλευρά του τετραγώνου (μπλε βελάκια), ώστε να συνεχίσουν την πορεία τους ομαλά. Το εικονιζόμενο οδικό δίκτυο που σχεδιάστηκε στο περιβάλλον του SUMO (εικόνα 31), έγινε με την βοήθεια του προγράμματος netedit, που συνοδεύεται μαζί με το SUMO και αποτελεί ένα πρόγραμμα δημιουργίας και επεξεργασίας οδικών δικτύων.



Εικόνα 31 Παράδειγμα υλοποίησης στο περιβάλλον του SUMO

Στο περιβάλλον του netedit, επεξεργάζονται όλες οι παράμετροι που συνθέτουν ένα τελικό οδικό δίκτυο, όπως δρόμοι, σταυροδρόμια, διαβάσεις, φανάρια, ταμπέλες σήμανσης, διαβάσεις, οχήματα και πολλά άλλα. Κάθε ένα στοιχείο που προστίθεται από το netedit, αποθηκεύεται ως μεταφραζόμενο αρχείο σε γλώσσα XML ώστε να μπορούν να αναγνωριστούν οι παράμετροι του δικτύου και από άλλα προγράμματα. Στην συγκεκριμένη περίπτωση, το οδικό δίκτυο απαρτίζεται από οριζόντιους και κάθετους δρόμους διπλής κατεύθυνσης, και ορίζονται ως παράμετροι στο SUMO με το tag <lane> που αφορά κάθε λωρίδα κατεύθυνσης ξεχωριστά. Ωστόσο στην υλοποίηση, τα tags των δρόμων εισέρχονται ως

υποενότητες μέσα σε ένα διαφορετικό tag, το <edge>. Αυτό συμβαίνει διότι οι δρόμοι στο SUMO πρέπει να συνδέονται μεταξύ τους με κάθε τρόπο και το <edge> αποτελεί την παράμετρο της σύνδεσης των δρόμων. Κάθε <lane> περιέχει παραμέτρους που ορίζουν το αναγνωριστικό (id), την λωρίδα κατεύθυνσης (index) όπου ανάλογα με την τιμή ορίζεται και η λωρίδα και με το 0 να σημαίνει την πιο αριστερή λωρίδα, την επιτρεπόμενη ταχύτητα σε μέτρα/δευτερόλεπτο (speed), το μήκος της λωρίδας σε μέτρα (length) και την γεωμετρία της λωρίδας (shape). Αντίστοιχα, το tag <edge> απαιτεί επίσης ένα αναγνωριστικό αλλά και τον ορισμό της χρησιμότητας τους (function). Στην υλοποίηση, τα <edge> συνδέονται με τις άκρες από σταυροδρόμια, συνεπώς έχουν τιμή internal.

```
<edge id=":J12_0" function="internal">
    <lane id=":J12_0_0" index="0" speed="6.08" length="7.58" shape="-53.47,121.13 -55.52,120.81 -56.99,119.92 -57.89,118.45 -58.21,116.41"/> </edge>
```

Κώδικας 8 Ορισμός λωρίδας ενός δρόμου

Παράλληλα, υπάρχουν λωρίδες δρόμων σε αυτήν την προσομοίωση που συνδέουν αποκλειστικά δύο διαφορετικούς κόμβους ή σταυροδρόμια στο οδικό δίκτυο. Κάθε αναγνωριστικό για ένα <edge> ξεκινά με το γράμμα E και κάθε αναγνωριστικό για έναν κόμβο ή σταυροδρόμι ξεκινά με J. Επίσης, κάθε αναγνωριστικό για ένα <edge> μπορεί να έχει πρόθεμα (-) που εκφράζει την αντίθετη κατεύθυνση σε μια λωρίδα δρόμου. Π.χ. το <edge> E18 ενός δρόμου ξεκινά από τον κόμβο J16 και καταλήγει στον J17. Αντίθετα, το -E18 ξεκινά από τον J17 και καταλήγει στον J16. Ακόμη, το SUMO ορίζει την παράμετρο priority με σκοπό να διαφοροποιήσει τους δρόμος με βάση την προτεραιότητα. Ένα <edge> με χαμηλή τιμή (-1), δεν αντιστοιχεί σε ένα δρόμο με μεγάλη κίνηση και ροή άρα είναι χαμηλής προτεραιότητας. Αντιθέτως, οι τιμές 1 ή 2 για παράδειγμα εκφράζουν δρόμους με υψηλή προτεραιότητα.

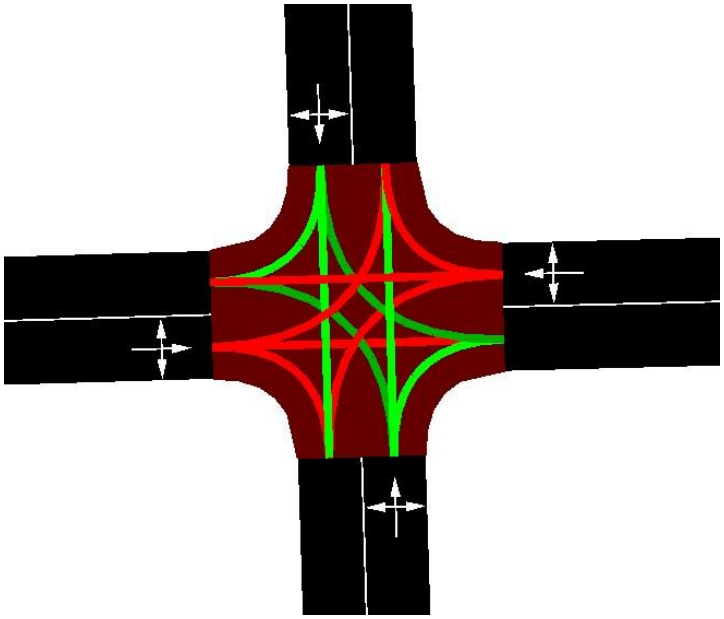
```
<edge id="-E18" from="J17" to="J16" priority="-1">
    <lane id="-E18_0" index="0" speed="13.89" length="62.78" shape="159.36,56.67 157.94,119.44"/>
  </edge>
```

Κώδικας 9 Ορισμός προτεραιότητας σε ένα δρόμο

Επιπλέον, στο κεντρικό σταυροδρόμι της υλοποίησης στο SUMO, έχουν τοποθετηθεί φανάρια για να ρυθμίζουν την κυκλοφορία των οχημάτων. Ένα φανάρι αποτελεί μία μεμονωμένη οντότητα στο netedit και στο SUMO καθώς ορίζεται με το tag <tlLogic>. Το <tlLogic> απαιτεί και αυτό με την σειρά παραμέτρους με τιμές που διαχειρίζονται την λειτουργία του. Όσο αφορά το αναγνωριστικό, ορίζεται το id του κόμβου στον οποίο τοποθετείται το φανάρι, όπως στην συγκεκριμένη περίπτωση που εισέρχεται το αναγνωριστικό του κεντρικού κόμβου. Ο τύπος του φαναριού ορίζει το αν η σηματοδότηση θα είναι στατική ή δυναμική. Η στατική σηματοδότηση είναι προκαθορισμένη και δεν αλλάζει σε σχέση με την δυναμική που μπορεί να αλλάξει ανάλογα με την κίνηση στον κόμβο. Το programID είναι το αναγνωριστικό για την συγκεκριμένη σηματοδότηση που δόθηκε στο φανάρι. Κάθε φανάρι μπορεί να έχει διαφορετική σηματοδότηση και πρέπει να ξεχωρίζεται από τις άλλες για το SUMO. Το offset αντιστοιχίζει την έναρξη του προγράμματος του φαναριού με την έναρξη της προσομοίωσης. Στην υλοποίηση ξεκινά παράλληλα με την προσομοίωση και για αυτό παίρνει την τιμή 0. Ως ένθετα tags υπάρχουν τα <phase> που ελέγχουν την σηματοδότηση. Η παράμετρος duration σαφώς ορίζει την διάρκεια που θα φωτίζει το εκάστοτε χρώμα σε δευτερόλεπτα και η state αντίστοιχα ορίζει το χρώμα της σηματοδότησης, με χαρακτήρες που αντιστοιχούν στο αρχικό γράμμα του χρώματος γραμμένους σε συμβολοσειρά. Κάθε ένας χαρακτήρας της συμβολοσειράς αφορά και μία λωρίδα πορείας (εικόνα 32).

```
<tlLogic id="J26" type="static" programID="0" offset="0">
      <phase duration="42" state="rrrGGgrrrGGg"/>
      <phase duration="3"  state="rrryyyrrryyy"/>
      <phase duration="42" state="GGgrrrGGgrrr"/>
      <phase duration="3"  state="yyyrrryyyrrr"/>
   </tlLogic>
```

Κώδικας 10 Σχεδιασμός φωτεινού σηματοδότη



Εικόνα 32 Διαδρομές σηματοδότησης στο σταυροδρόμι

Πίνακας 3 Περιπτώσεις σηματοδότησης

State	Σηματοδότηση
r	Το όχημα πρέπει να σταματήσει
y	Το όχημα πρέπει να επιβραδύνει
g	Το όχημα μπορεί να προχωρήσει αλλά δεν έχει προτεραιότητα
G	Το όχημα μπορεί να προχωρήσει

Προφανώς, μετέπειτα πρέπει να οριστούν και οι κόμβοι/σταυροδρόμια ως οντότητες ξεχωριστά. Για να επιτευχθεί αυτό, το SUMO χρησιμοποιεί το tag <junction> το οποίο με την σειρά του απαιτεί κάποιες παραμέτρους ώστε να συνεχιστεί ομαλά η δημιουργία της προσομοίωσης. Το αναγνωριστικό καθορίζει την ονομασία. Η παράμετρος type παίρνει την τιμή priority διότι, στο σταυροδρόμι θέλουμε να ισχύουν οι γνωστοί νόμοι της προτεραιότητας από τον πραγματικό ΚΟΚ. Οι τιμές χ και υ ορίζουν τις συντεταγμένες της τοποθεσίας του κόμβου στον χάρτη της προσομοίωσης. Έπειτα έχουμε τις εισερχόμενες λωρίδες στον κόμβο και τις εσωτερικές λωρίδες μέσα στον κόμβο αντίστοιχα. Η παράμετρος shape ορίζει και πάλι την γεωμετρία του κόμβου. Τα ένθετα tags <request> ελέγχουν την συμπεριφορά του σταυροδρομιού που αφορά την προτεραιότητα των οχημάτων και τις λωρίδες που συνδέονται. Όπως προαναφέρθηκε, η index ορίζει την λωρίδα που αναφερόμαστε. Η 0 είναι η πιο αριστερή και συνεχίζει με την 1 που είναι η επόμενη. Συνεχίζοντας, η response αφορά την προτεραιότητα στις λωρίδες, όπου 00 σημαίνει πως δεν υπάρχει προτεραιότητα, η foes αφορά τα ρεύματα από λωρίδες που μπορεί να συγκρούονται στον κόμβο, όπου και πάλι το 0 σημαίνει πως δεν υπάρχει, και τέλος, η cont αφορά τον κανόνα για το αν το ρεύμα της λωρίδας του κόμβου συνεχίζει χωρίς εμπόδιο ή θα χρειαστεί το όχημα να σταματήσει. Η τιμή 0 σημαίνει πως δεν υπάρχει συνέχεια στην λωρίδα και ίσως το όχημα να χρειαστεί να διακόψει την πορεία του.

Σε κάθε παράμετρο ένα ψηφίο αντιστοιχεί σε μία λωρίδα ενός δρόμου, όπου το 0 σημαίνει όχι ("δεν επιτρέπεται", "δεν υπάρχει σύγκρουση", "δεν συνεχίζεται") ενώ το 1 σημαίνει ναι ("επιτρέπεται", "υπάρχει σύγκρουση", "συνεχίζεται").

```
<junction id="J12" type="priority" x="-56.56" y="119.49" incLanes="-E45_0 E25_0" intLanes=":J12_0_0
:J12_1_0" shape="-53.49,122.73 -53.41,116.33 -59.81,116.43 -59.06,119.90 -58.18,121.12 -56.96,121.99
-55.39,122.53">
    <request index="0" response="00" foes="00" cont="0"/>
    <request index="1" response="00" foes="00" cont="0"/>
  </junction>
```

Κώδικας 11 Ορισμός σταυροδρομιού

Παράλληλα, οι συνδέσεις που γίνονται μεταξύ δύο δρόμων χωρίς να έχουμε δημιουργήσει ένα σταυροδρόμι οδηγούν στην παρουσία του tag <connection>. Στην υλοποίηση της εργασίας, δύο δρόμοι πρέπει να συνδεθούν μεταξύ τους για κάθε λόγο, αλλά κυρίως γιατί ο μηχανισμός δημιουργίας δρόμων του netedit δεν αφήνει άλλα περιθώρια από το να συνδέεις μεμονωμένα τμήματα δρόμου. Όταν για παράδειγμα συνδέσουμε τις δύο άκρες δύο δρόμων στο netedit, όπως η -E18 και η -E49, το SUMO το μεταφράζει σε σύνδεση με tag <connection>. Η παράμετροι fromLane και toLane καθορίζουν ποιες λωρίδες των δύο δρόμων θα χρησιμοποιηθούν για το όχημα. Συγκεκριμένα το 0 σημαίνει τις «πρώτες» λωρίδες του δρόμου. Η παράμετρος via θέτει ως τιμή το αναγνωριστικό του θεωρητικού κόμβου που σχηματίζεται κατά την ένωση των δύο δρόμων. Και τελικά, η dir με την τιμή l εκφράζει ότι η ροή στον δρόμο κάνει στροφή αριστερά σε λωρίδα και η state εκφράζει την σηματοδότηση για μία σύνδεση δρόμων, με το M να σημαίνει ότι επιτρέπεται η πορεία με βάση την σηματοδότηση που υπάρχει στον κόμβο/δρόμο. Βέβαια, αυτές οι δύο παράμετροι λαμβάνουν πολλές διαφορετικές τιμές (πίνακας 2).

```
<connection from="-E18" to="-E49" fromLane="0" toLane="0" via=":J16_0_0" dir="l" state="M"/>
```

Κώδικας 12 Σύνδεση δύο δρόμων

Πίνακας 4 Διαφορετικές τιμές για dir και state

dir	Κατεύθυνση	state	Σηματοδότηση
s	Η λωρίδα συνεχίζει ευθεία	M	Κίνηση με βάση την σηματοδότηση
l	Η λωρίδα κάνει στροφή αριστερά	m	Πορεία χωρίς προτεραιότητα
r	Η λωρίδα κάνει στροφή δεξιά	O	Απαγόρευση πορείας, επιτρέπεται δεξιά στροφή
L	Αριστερή αναστροφή	o	Απαγόρευση πορείας
R	Δεξιά αναστροφή	g	Πράσινο φανάρι
-		r	Κόκκινο φανάρι
-		y	Κίτρινο φανάρι

Επίσης, σε ορισμένες περιπτώσεις υπάρχει και η παράμετρος linkIndex που αφορά τις εσωτερικές συνδέσεις των λωρίδων σε έναν κόμβο. Όταν ένα όχημα εισέρχεται σε ένα κόμβο από μία λωρίδα, δεν

επιλέγει κάθε φορά την λωρίδα εξόδου από τον κόμβο, αλλά μέσω της linkIndex για εξομάλυνση της κίνησης στον κόμβο.

Αφού δημιουργήθηκαν όλα τα τμήματα δρόμων και κόμβων για τον χάρτη της προσομοίωσης, σειρά έχουν οι ορισμοί για τις διαδρομές που θα εκτελούν τα οχήματα επάνω στον χάρτη. Για να δημιουργηθεί το αρχείο διαδρομών, χρειάζεται πρώτα να χρησιμοποιήσουμε το tag <routes>. Εσωτερικά αυτής της ενότητας, τα tags <trip> θέτουν την κάθε διαδρομή στον χάρτη ξεχωριστά. Ως διαδρομή εννοούμε την κάθε σειρά από δρόμους που μπορεί να κινηθεί ένα όχημα. Ως παραμέτρους, θέτουμε πρώτο και κύριο πάντα το αναγνωριστικό, με την ονομασία vehicle, το χρονικό σημείο αναχώρησης του κάθε οχήματος μετρημένο σε δευτερόλεπτα και τις άκρες των δρόμων που πρώτον αναχωρεί το όχημα και δεύτερον που καταλήγει. Η συγκεκριμένη υλοποίηση πραγματεύεται ένα οδικό δίκτυο με 11 διαδρομές οχημάτων, ωστόσο οι εισαγωγές κι άλλων tags <trip> είναι πάντα εφικτές.

```
<trip id="vehicle_0" depart="0.00" from="-E32" to="-E31"/>
```

Κώδικας 13 Διαδρομή ενός οχήματος

Συνεχίζοντας με την υλοποίηση στο περιβάλλον του SUMO, κάνουμε χρήση της διεπαφής TraCI που συλλέγει δεδομένα και παραμετροποιεί στοιχεία της προσομοίωσης δυναμικά. Κάνοντας εισαγωγή της διεπαφής μέσω κώδικα σε γλώσσα Python και συσχετίζοντας τον κώδικα με την προσομοίωση, μπορούμε να διαχειριστούμε τα οχήματα που έχουν δημιουργηθεί από το SUMO.

```
import traci

import traci.constants as tc

# Σύνδεση με το αρχείο του SUMO

 traci.start(["sumo-gui", "-c", "ClusterRerouting.sumocfg"]) # Το sumocfg αρχείο συνθέτει όλα τα στοιχεία μαζί.

Ενα αρχείο sumocfg είναι της μορφής που δίνεται παρακάτω:

<configuration>

<input>

 <net-file value= "ClusterRerouting.net.xml"/> <!-- Το αρχείο με το σχήμα του οδικού δικτύου, των φαναριών και των συνδέσεων. -->

 <route-files value="ClusterRerouting.rou.xml"/><!-- Το αρχείο με τις διαδρομές <trip> των οχημάτων. -->

</input>

</configuration>
```

Κώδικας 14 Εισαγωγή TraCI σε python

Στην συγκεκριμένη περίπτωση, θέλουμε τα υπόλοιπα οχήματα εκτός του πρώτου, να αλλάζουν πορεία ώστε να αποφύγουν το εμπόδιο στο κεντρικό σταυροδρόμι. Όσο υπάρχουν οχήματα που εκτελούν πορεία στον χάρτη η προσομοίωση θα συνεχίσει να τρέχει.

```
while traci.simulation.getMinExpectedNumber() > 0:

  traci.simulationStep()
```

Κώδικας 15 Κανόνας συνθήκης επανάληψης

Στην πορεία πρέπει να ελέγξουμε πότε το πρώτο όχημα θα σταματήσει στο κεντρικό σταυροδρόμι. Οι παράμετρος για το όχημα παραμένει ίδια με του SUMO και ορίζουμε το σημείο ως J26 όπως έχει ονομαστεί το κεντρικό σταυροδρόμι στο SUMO. Αφότου φτάσει στον κεντρικό κόμβο, θέτουμε σε παύση την πορεία του

```
if traci.vehicle.getRoadID("vehicle_0") == "J26":
    traci.vehicle.setStop("vehicle_0", "J26", duration=300)
```

Κώδικας 16 Κανόνας συνθήκης αναδρομολόγησης

Ύστερα, δημιουργούμε μία άλλη επαναληπτική συνθήκη όπου διαχωρίζουμε τα υπόλοιπα 10 οχήματα που έχουμε στην προσομοίωση, σε δύο ομάδες. Σκοπός είναι οι δύο ομάδες να χωριστούν διαλέγοντας διαφορετικές πορείες δρόμων για να φτάσουν στην άλλη μεριά του τετραγώνου. Έτσι, τα πρώτα πέντε οχήματα θα διαλέξουν την δεξιά παράκαμψη βάσει του χάρτη, ενώ τα υπόλοιπα πέντε την αριστερή.
Ως δρόμοι, έχουν επιλεχθεί οι ακμές των δρόμων με την πρώτη να ορίζει το σημείο έναρξης αναδρομολόγησης και η δεύτερη τον προορισμό. Το εγχείρημα αυτό έχει αποτυπωθεί σε εικονικό παράδειγμα στην εικόνα 31 της εργασίας.

```
for vehicle_id in ["vehicle_1", "vehicle_2", "vehicle_3", "vehicle_4", "vehicle_5", "vehicle_6", "vehicle_7", "vehicle_8", "vehicle_9", "vehicle_10"]:
        # Επαναπροσδιορισμός διαδρομών ανά cluster
        if vehicle_id in ["vehicle_1", "vehicle_2", "vehicle_3", "vehicle_4", "vehicle_5"]:
            traci.vehicle.setRoute(vehicle_id, ["-E32", "-E25"])
        else:
            traci.vehicle.setRoute(vehicle_id, ["-E32", "E24"])
```

Κώδικας 17 Επαναληπτική διαδικασία αναδρομολόγησης

Για να τρέξει η εφαρμογή του TraCI, θα χρειαστεί να υπάρχει εγκατεστημένη η python στο σύστημα μας και να γράψουμε την εντολή python reroute.py σε μία γραμμή εντολών, όπου reroute είναι το όνομα του αρχείου TraCI. Η python με την εντολή **traci.start()** θα διαβάσει τα αρχεία του SUMO και θα εφαρμόσει επάνω στην προσομοίωση του οδικού δικτύου, τις εντολές του script.
Κατά την περαιτέρω εκτέλεση της προσομοίωσης θα επιχειρήσουμε να προσθέσουμε σε αυτήν την συνθήκη μία μέθοδο ομαδοποίησης με έναν σχετικό αλγόριθμο, όπου οι δύο ομάδες θα λειτουργούν ως clusters ανάλογα με την επιλογή διαδρομής του οχήματος. Με αυτόν τον τρόπο, θα αποφευχθεί η παραπάνω συμφόρηση που θα μπορούσε να δημιουργηθεί αν όλα τα οχήματα είχαν την ίδια πορεία.

Αναφορικά, η μέθοδος αναδρομολόγησης είναι σχετικά απλή. Το ίδιο απλή, απλώς περισσότερο χρονοβόρα, θα ήταν αν στις ίδιες συνθήκες προσθέταμε περισσότερα οχήματα. Δεν θα άλλαζε ιδιαίτερα η μορφή του κώδικα, απλώς θα υπήρχαν περισσότερα δεδομένα προς επεξεργασία για το SUMO και εμάς. Η υλοποίηση πραγματεύεται το ουσιαστικό κομμάτι της ιδέας και όχι την πρακτικότητα ή την απόδοση. Θεωρητικά σε ένα οδικό δίκτυο της κλίμακας του παρουσιαζόμενου στην εργασία, μερικά δεκάδες οχήματα είναι ικανοποιητικό πλήθος για την προσομοίωση με βάση των δρόμων που δίνονται για επιλογή επαναδρομολόγησης.

5.3.2 Ανάλυση υλοποίησης δικτύου σε γλώσσα NED στο περιβάλλον OMNET++

Παράλληλα με τον σχεδιασμό του οδικού δικτύου, πρέπει να δημιουργηθεί και η αντίστοιχη υποδομή για το δίκτυο που θα δεσπόζει στους δρόμους. Πλέον μεταφερόμαστε στο περιβάλλον ανάπτυξης λογισμικού του OMNET++ το οποίο δίνει την ευκαιρία να συνθέσουμε κώδικα χρησιμοποιώντας τα δικά του εργαλεία για την προσομοίωση. Τώρα, είναι εύκολο να συνταχθεί κώδικας γραμμένος στην γλώσσα NED, η οποία συνεργάζεται άμεσα με το περιβάλλον του OMNET++.

Ως παραμέτρους, λαμβάνουμε στον νου μας τα οχήματα, τις μονάδες αναμετάδοσης ασύρματων μηνυμάτων (radio medium) και τις σταθερές βάσεις επικοινωνίας (RSUs). Επίσης, απαιτείται σχεδιασμός αμφίδρομης επικοινωνίας μεταξύ όλων των μέσων στο δίκτυο, με τις απαραίτητες μεταβλητές που χρησιμεύουν για την επικοινωνία, όπως π.χ. πύλες αποστολής και λήψης μηνυμάτων.

Για αρχή, χρειάζεται η εισαγωγή κλάσεων από frameworks όπως το VEINS και το INET για να εισάγουμε μοντέλα στην προσομοίωση, έτοιμα χωρίς περαιτέρω διεργασίες. Διαφορετικά, θα έπρεπε να σχεδιαστούν εξ’ αρχής από εμάς πράγμα που φαντάζει ιδιαίτερα χρονοβόρο. Η μόνη αναγκαία εντολή της NED είναι η import για να συνδεθούν οι κλάσεις με τα αρχεία των frameworks.

```
import inet.node.inet.Router;

import inet.node.inet.StandardHost;

import inet.mobility.single.MassMobility;

import inet.physicallayer.ieee80211.packetlevel.Ieee80211ScalarRadioMedium;

import inet.physicallayer.ieee80211.packetlevel.Ieee80211ScalarRadio;

import inet.visualizer.intergrated.IntegratedVisualizer;

import org.car2x.veins.modules.mobility.traci.TraCIScenarioManager;

import org.car2x.veins.visualizer.roads.RoadsCanvasVisualizer;
```

Κώδικας 18 Εισαγωγή ενοτήτων σε NED αρχείο

Οι περισσότερες από τις κλάσεις του INET θα συσχετιστούν με τις ενότητες του δικτύου που θα δημιουργήσουμε προκειμένου να επιτευχθεί μια συνεχόμενη ασύρματη επικοινωνία από το πρώτο όχημα προς τα υπόλοιπα με την σειρά, μέσω των ενδιάμεσων δεκτών επικοινωνίας.

Πίνακας 5 Μοντέλα του INET για την υλοποίηση

Μοντέλο	Χρησιμότητα
Router	Δρομολογητής-Αντιστοιχίζεται με τα RSUs
StandardHost	Δρομολογητής-Αντιστοιχίζεται με τα οχήματα
MassMobility	Προσδίδει κίνηση στον χώρο για τα οχήματα
Ieee80211ScalarRadioMedium	Ταυτίζεται με ένα RSU για να στείλει δεδομένα
Ieee80211ScalarRadio	Ταυτίζεται με έναν κόμβο για να στείλει δεδομένα

Όμως, χρειάζονται και περαιτέρω κλάσεις που αφορούν πρακτικά την προσομοίωση, με την TraCiScenarioManager να είναι αρμόδια διαχείρισης της διεπαφής με το πρόγραμμα του SUMO και τον κώδικα αναδρομολόγησης που σχεδιάστηκε παραπάνω. Αντίστοιχα, οι κλάσεις IntegratedVisualizer και RoadsCanvasVisualizer θα βοηθήσουν ώστε να σχεδιαστεί ένα δισδιάστατο τοπίο προσομοίωσης στο

OMNET++ που θα απεικονίζει τον χάρτη του οδικού δικτύου του SUMO αλλά και τις μετακινήσεις των οχημάτων και λοιπών γραφικών και μοντέλων που θα προστεθούν.

Ξεκινώντας τώρα με την δόμηση του δικτύου σε γλώσσα NED, πρέπει να οριστεί ένα όνομα για το δίκτυο προς κατασκευή, το οποίο όνομα θα μπορεί να είναι αναγνωρίσιμο για τα υπόλοιπα προγράμματα που θα συνεργαστούν. Για την συγκεκριμένη περίπτωση της υλοποίησης, το όνομα θα είναι ClusterRerouting.

```
network ClusterRerouting
```

Κώδικας 19 Ονοματοδοσία δικτύου

Έπειτα, χρειαζόμαστε να δημιουργήσουμε την πρώτη και κύρια ενότητα, του αυτοκινήτου. Η ενότητα των οχημάτων είναι σύνθετη, και κληρονομεί στην ουσία την ιδιότητα του StandardHost με σκοπό να μπορεί να ανταλλάζει δεδομένα ως κόμβος. Τοποθετώντας την εντολή types ορίζουμε τους νέους τύπους ενότητας, όπως την CarType.

```
types:
    // Ενότητα της παραμέτρου του οχήματος
    module CarType extends StandardHost
```

Κώδικας 20 Δήλωση σύνθετης ενότητας

Η σύνθετη ενότητα CarType θα χρειαστεί ατομικά κάποιες παραμέτρους εντός της που θα μας βοηθήσουν να την εκμεταλλευτούμε κυρίως για το πρόγραμμα ομαδοποίησης και αποστολής μηνυμάτων που θα αναλυθεί παρακάτω. Η εισαγωγή των παραμέτρων εντός μίας ενότητας γίνεται με την εντολή **parameters**.

Πίνακας 6 Παράμετροι ενότητας CarType

Παράμετροι	Τύπος μεταβλητής	Λειτουργία
numCars	Ακέραιος	Πλήθος οχημάτων
k	Ακέραιος	Πλήθος cluster
ClusterIterationLimit	Ακέραιος	Όριο επαναλήψεων αλγόριθμου ομαδοποίησης
broadcastTime	Κινητής υποδιαστολή διπλής ακρίβειας	Χρονικό σημείο αποστολής μηνύματος από το πρώτο όχημα

Επειδή χρησιμοποιούμε πρόγραμμα με εικονική αναπαράσταση, μπορούμε μέσω της εντολής @display να δώσουμε μία γραφική απεικόνιση στα οχήματα όπως και σε κάθε άλλο στοιχείο του δικτύου.

```
@display("i=misc/car2");
```

Κώδικας 21 Προσθήκη εικόνας για μία ενότητα

Θέλοντας η ενότητα των οχημάτων να συσχετιστεί με τον αλγόριθμο της C++ που θα αναλυθεί παρακάτω, θέτουμε στην app.typename το όνομα της κλάσης που ευθύνεται για την νέα δρομολόγηση που υπολογίζεται στον κώδικα της C++. Η app είναι μια λογική μονάδα που εκτελεί εφαρμογές που σχετίζονται με την ενότητα στην οποία βρίσκεται εσωτερικά. Εδώ θέλουμε η ενότητα των οχημάτων να σχετίζεται με την εκτέλεση της εφαρμογής δρομολόγησης.

```
app.typename = ClusterRerouting;
```

Κώδικας 22 Εφαρμογή ενότητας CarType

Η κίνηση των οχημάτων μέσα στον χώρο της προσομοίωσης χαρακτηρίζεται από το Mobility.typename και την τιμή MassMobility.Σε σχέση με τα RSUs, τα οχήματα θέλουμε να εκτελούν ελεύθερη κίνηση μέσα στον χώρο και όχι να είναι στάσιμα.

```
Mobility.typename = MassMobility;
```

Κώδικας 23 Κινητικότητα της ενότητας

Για την επικοινωνία με τα υπόλοιπα μέσα, η wlan.typename δίνει στα οχήματα να χρησιμοποιούν μία διεπαφή ασύρματης επικοινωνίας που χρησιμοποιεί την τεχνολογία IEEE802.11.

```
wlan.typename = Ieee80211ScalarRadio;
```

Κώδικας 24 Μέσο επικοινωνίας ενότητας

Τέλος, οι θύρες επικοινωνίας των οχημάτων upperLayerIn και upperLayerOut χρησιμοποιούνται για επικοινωνία σε όλα τα στρώματα δικτύου με κύριο το Application Layer προκειμένου να υπάρχει υποστήριξη για την μεταφορά δεδομένων δρομολόγησης.

```
gates:
        Input upperLayerIn; // Θύρα Εισόδου
        output upperLayerOut; // Θύρα Εξόδου
```

Κώδικας 25 Θύρες ενότητας

Αντιστοίχως, πρέπει να σχεδιαστεί και η ενότητα των σταθερών βάσεων μετάδοσης. Η διαδικασία είναι αντίστοιχη, επιλέγοντας πως η ενότητα θα είναι σύνθετη και θα κληρονομεί τις ιδιότητες της κλάσης Router για να μπορεί να δρομολογεί αποστολές μηνυμάτων στους υπόλοιπους κόμβους.

```
module RSU extends Router
```

Κώδικας 26 Ενότητα βάσης επικοινωνίας

Η γραφική απεικόνιση του rsu θα είναι ως μία κεραία σταθερής βάσης.

```
@display("i=device/antennatower");
```

Κώδικας 27 Εικονίδιο για βάση επικοινωνίας

Η σταθερή βάση χρειάζεται να συμμετάσχει στην εφαρμογή της δρομολόγησης οπότε θα πρέπει να συσχετιστεί με αυτήν.

```
app.typename = "ClusterRerouting";
```

Κώδικας 28 Εφαρμογή ενότητας RSU

Οι βάσεις προφανώς και δεν απαιτούν καμία περαιτέρω επεξεργασία στην κίνηση τους μέσα στον χάρτη, οπότε η κινητικότητα τους μπορεί να δηλωθεί ως στάσιμη.

```
Mobility.typename = "StaticMobility";
```

Κώδικας 29 Κινητικότητα βάσης επικοινωνίας

Οι βάσεις επικοινωνίας χρειάζεται να επικοινωνούν σε φυσικό επίπεδο με ραδιοκύματα και με οχήματα αλλά και με άλλες βάσεις. Για αυτό εδώ χρησιμοποιούμε μια διαφορετική υλοποίηση του INET για την δικτύωση μέσω IEEE802.11. Το ScalarRadioMedium ρυθμίζει παράγοντες όπως την ισχύ του σήματος, τις απώλειες και τις μεταφορές. Συνεπώς, και οι θύρες εισόδου και εξόδου θα είναι διαφορετικές εφόσον αναφερόμαστε σε φυσικό επίπεδο μεταφοράς. Να σημειωθεί εδώ πως η θύρα εισόδου πρακτικά δεν χρειάζεται να δηλωθεί ξανά, μιας και η κλάση Router την περιέχει από μόνη της. Όμως η θύρα εξόδου πρέπει να δηλωθεί εκ νέου. Ακόμη, να σημειωθεί πως οι θύρες δηλώνονται ως διάνυσμα, επειδή στην ουσία ένας δρομολογητής μπορεί να έχει μια σειρά από θύρες και των δύο περιπτώσεων.

```
wlan.typename = "Ieee80211ScalarRadioMedium";
        gates:
          input radioIn[i];
          output radioOut[i];
```

Κώδικας 30 Μέσο επικοινωνίας και θύρες της βάσης

Εφόσον οι δύο κύριες ενότητες του δικτύου έχουν δημιουργηθεί, θα χρειαστούν επιπλέον κάποιες υποενότητες που παίζουν ρόλο στην σύνθεση. Ως υποενότητα εννοούμε μια οντότητα που κληρονομεί από μία άλλη ενότητα. Το πεδίο δήλωσης των υποενοτήτων βρίσκεται εκτός των πεδίων που δηλώσαμε ό,τι σχετίζεται με τις ενότητες και αρχικοποιείται με την εντολή **submodules**.

Οι πρώτες υποενότητες είναι οι radioMediumRSU και radioMediumCar και η χρήση τους από το περιβάλλον του OMNET++ είναι αναγκαία για την αναμετάδοση σημάτων και μηνυμάτων για την προσομοίωση. Λαμβάνουν τις ιδιότητες των κλάσεων IEEE80211ScalarRadioMedium και IEEE80211ScalarRadio και μέσω της @display τις τοποθετούμε σε συντεταγμένες στο χάρτη του δικτύου με σημεία x,y. Η @display σαφώς θα χρησιμοποιηθεί με τον ίδιο ακριβώς τρόπο και σε όλες τις άλλες υποενότητες που θα δηλωθούν μιας και είναι αναγκαίο από το πρόγραμμα του OMNET++ ακόμα και για υποενότητες που δεν έχουν κάποια πραγματική φυσική υπόσταση.

```
radioMediumRSU: IEEE80211ScalarRadioMedium {
        @display(“p=200,200”); }
radioMediumCar: IEEE80211ScalarRadio {
        @display(“p=200,91”);}
```

Κώδικας 31 Μέσο επικοινωνίας για τις ενότητες

Στην συνέχεια, θέλουμε να δημιουργήσουμε στην ουσία τους κόμβους του δικτύου, οι οποίοι θα κληρονομούν από την CarType μιας και μιλάμε για οχήματα, και θα ορίζονται ως ένας μονοδιάστατος πίνακας με αριθμό στοιχείων-κόμβων όσο επιλέξουμε εμείς. Η τιμή που θα πάρει κατά την δήλωση ο πίνακας για πεπερασμένο αριθμό κόμβων εσωτερικά, ισούται με την μεταβλητή numCars και η τιμή μπορεί να δηλωθεί στο αρχείο ρυθμίσεων παραμέτρων τύπου .ini που θα αναλυθεί αμέσως μετά.

```
car[numCars]: CarType {
        @display("p=50,50");}
```

Κώδικας 32 Υποενότητα της ενότητας CarType

Αμέσως μετά, πρέπει να δηλωθεί και η ύπαρξη μίας σταθερής βάσης δικτύου που θα κληρονομεί από την ενότητα της RSU. Για την αποφυγή λαθών κατά την εκτέλεση όλων των προγραμμάτων που θα συνεργάζονται, το όνομα της υποενότητας θα παραμείνει το ίδιο απλώς με μικρούς χαρακτήρες για να μην υπάρχει πρόβλημα από το OMNET++.

```
rsu: RSU{
        @display("p=100,100");}
```

Κώδικας 33 Υποενότητα της ενότητας RSU

Έπειτα, πρέπει να δηλώσουμε μια υποενότητα μια αόριστη φυσική υπόσταση για τις ανάγκες του OMNET++, η οποία λειτουργεί ως διαχειριστής ανταλλαγής δεδομένων μεταξύ του OMNET++ και του SUMO κάνοντας χρήση της διεπαφής TraCI. Η υποενότητα μπορεί να δηλωθεί ως manager και θα κληρονομεί από την κλάση TraCIScenarioManager που δηλώσαμε στην αρχή του προγράμματος.

```
manager: TraCIScenarioManager {
        @display("p=37,215");}
```

Κώδικας 34 Υποενότητα διαχειριστή δεδομένων

Τέλος πρέπει να δηλωθούν και δύο υποενότητες, που θα διαχειρίζονται την εικονοποίηση της προσομοίωσης στο περιβάλλον του OMNET++. Η πρώτη κληρονομεί από την κλάση IntegratedVisualizer και είναι υπεύθυνη για την απεικόνιση των βασικών στοιχείων του INET σε δισδιάστατο χάρτη, και η δεύτερη κληρονομεί από την κλάση RoadsCanvasVisualizer, η οποία είναι υπεύθυνη για την απεικόνιση του οδικού δικτύου του SUMO και των οχημάτων, διότι αυτό δεν είναι εφικτό από την πρώτη κλάση.

```
visualizer: IntegratedVisualizer {
        @display("p=102,230");}
roadsCanvasVisualizer: RoadsCanvasVisualizer{
        @display("p=115,153");}
```

Κώδικας 35 Υποενότητες εικονοποιητών

Στο τελευταίο στάδιο της δόμησης του δικτύου πρέπει να ορίσουμε τις συνδέσεις, δηλαδή να δείξουμε στην NED ποια στοιχεία συνδέονται μεταξύ τους και με ποιον τρόπο. Αρχικά πρέπει να ορίσουμε το είδος των συνδέσεων. Στην συγκεκριμένη περίπτωση, θέλουμε να υπάρχει ελαστικότητα με τις συνδέσεις και να είναι εφικτό κάποιες θύρες να μένουν χωρίς σύνδεση με κάποια άλλη.

```
connections allowunconnected:
```

Κώδικας 36 Συνθήκη συνδεσιμότητας

Αφού έχουμε γνωστοποιήσει από πριν στο πρόγραμμα ότι έχουμε φτιάξει έναν πίνακα από στοιχεία που απαιτούν σύνδεση, κάθε σύνδεση πρέπει να αρχικοποιείται με μια επαναληπτική διαδικασία. Τα οχήματα πρέπει να επικοινωνούν με τις σταθερές βάσεις, και η σύνδεση τους να αφορά τα δεδομένα της εφαρμογής δρομολόγησης. Κάθε κόμβος του μονοδιάστατου πίνακα που φτιάξαμε, συνδέεται με τις κατάλληλες θύρες στην βάση rsu.

```
for i=0..10 {
car[i].upperLayerOut --> rsu.radioIn[i];
car[i].upperLayerIn <-- rsu.radioOut[i]; }
```

Κώδικας 37 Βρόχος συνδέσεων των δύο ενοτήτων

Όμως, απαιτείται και αμοιβαία ανταλλαγή μηνυμάτων μεταξύ των οχημάτων. Η μόνη διαφορά είναι πως χρειάζεται ένα δεύτερο δείκτη στον πίνακα για να μετράει και να συγκρίνει παράλληλα, με γνώμονα όμως την αποφυγή σύνδεσης ενός κόμβου με τον εαυτό του. Για αυτό, ο δεύτερος δείκτης θέλουμε πάντα να δείχνει σε κελί με απόσταση ενός κελιού τουλάχιστον, από τον πρώτο δείκτη. Η γλώσσα NED δεν αφήνει πολλά περιθώρια σε συγγραφή μιας τέτοιας συνθήκης με διαφορετικό τρόπο, και για αυτό χρησιμοποιείται με τέτοιο τρόπο η συνθήκη for. Εναλλακτικά, θα μπορούσαμε εμβόλιμα να θέσουμε μια συνθήκη if για να αποφύγουμε το ίδιο πρόβλημα αλλά η NED δεν επιτρέπει κάτι τέτοιο να είναι εφικτό.

```
for i=0..10, for j=0..i-1{
car[i].upperLayerOut --> car[j].upperLayerIn;
car[j].upperLayerOut --> car[i].upperLayerIn;
 }
```

Κώδικας 38 Βρόχος συνδέσεων μεταξύ οχημάτων

5.3.3 Ανάλυση αρχείου παραμέτρων του δικτύου στο περιβάλλον OMNET++

Συμπληρωματικά, η προσομοίωση σαν σύνολο δεν είναι πραγματικά έτοιμη, μιας και χρειάζεται ένα ακόμα αρχείο ρυθμίσεων που θα καθοδηγεί μέσω των παραμέτρων που θέτει, τα μεμονωμένα στοιχεία του δικτύου προσομοίωσης. Το αρχείο είναι ένα κοινότυπο αρχείο ρυθμίσεων τύπου .ini και μέσα καθορίζει την τιμή κάθε μίας παραμέτρου που αφορά το δίκτυο. Η μορφή του είναι ιδιαιτέρως απλή και δεν απαιτεί κάποια επιπλέον γνώση σύνταξης κώδικα.

Για παράδειγμα, στην αρχή σύνταξης του αρχείου πρέπει να συνδέσουμε το αρχείο με τα υπόλοιπα, μέσω ονοματικής επιλογής, και να ορίσουμε την καθολική επίδραση των παραμέτρων στην υλοποίηση με την γραμμή [General] στην αρχή. Επιπλέον, πρέπει να ορίσουμε μερικές παραμέτρους που αφορούν μόνο την εκτέλεση της προσομοίωσης όπως η διάρκεια, ο δυναμικός έλεγχος για λάθη και ο τύπος εκτέλεσης της. Στην υλοποίηση επιλέγεται να εκτελείται ταχύρρυθμα χωρίς ενδιάμεσες εκτυπώσεις αποτελεσμάτων.

```
network = ClusterRerouting

sim-time-limit = 300s

debug-on-errors = true

cmdenv-express-mode = true
```

Κώδικας 39 Παράμετροι προσομοίωσης

Παρακάτω, ορίζονται οι ρυθμίσεις του αρχείου SUMO και συγκεκριμένα πρώτα θέτουμε τον διαχειριστή της επικοινωνίας μεταξύ των αρχείων του OMNET++ και του SUMO που θα είναι το TraCI διότι επιχειρούμε την δυναμική παραμετροποίηση της κυκλοφορίας μέσω νέας δρομολόγησης στην προσομοίωση. Επιπλέον θέλουμε και ο ρυθμός ανανέωσης του διαχειριστή TraCI να ισούται με 0,1 δευτερόλεπτα για ορθό συγχρονισμό.

```
*.manager.typename = "TraCIManager"

*.manager.updateInterval = 0.1s
```

Κώδικας 40 Παράμετροι διαχειριστή

Εφόσον γνωστοποιήσουμε στο αρχείο πως ο manager σχετίζεται με το διαχειριστή που ορίσαμε στο αρχείο της NED, το πρόγραμμα πρέπει να ξέρει για την συσκευή που τίθεται σε εφαρμογή η πλατφόρμα προσομοίωσης και για την θύρα επικοινωνίας που θα επικοινωνεί το OMNET++ με το SUMO.

```
*.manager.host = “localhost”

*.manager.port = “9999”
```

Κώδικας 41 Παράμετροι σύνδεσης εφαρμογών προσομοίωσης

Αναφορικά με την σύνταξη της κάθε παραμέτρου, ακολουθείται καθολικά από το αρχείο η ιεραρχική σύνταξη της κάθε παραμέτρου. Δηλαδή, αρχικά ο αστερίσκος υποδηλώνει την καθολική εφαρμογή της λειτουργίας μιας παραμέτρου σε όλα τα στοιχεία του δικτύου. Σε αντίθετη περίπτωση όπως θα φανεί και παρακάτω, θα γραφόταν ένα συγκεκριμένο στοιχείο αντί για αστερίσκο όπως το car. Μετά, γράφεται η κάθε ενότητα ή υποενότητα που σχετίζεται με το αρχικό στοιχείο και τελικά γράφεται η μεταβλητή που θα πάρει την τελική τιμή ρύθμισης που δημιουργήθηκε εντός της ενότητας. Ένα σχετικό παράδειγμα είναι και η εντολή .car[*].mobility.typename = "MassMobility" που δημιουργήσαμε στο αρχείο NED.

Επομένως, γνωρίζοντας πλέον την μορφή σύνταξης μια παραμέτρου στο αρχείο τύπου .ini ακολουθούν περιληπτικά αποσπάσματα από τις αναφορές για τις υπόλοιπες παραμέτρους του αρχείο της υλοποίησης.

Πίνακας 7 Κύριες παράμετροι οχημάτων

Παράμετρος	Τιμή	Λειτουργία
***.car[*].typename**	“CarType”	Συσχέτιση με την ενότητα CarType
***.car[*].mobility.typename**	“MassMobility”	Συσχέτιση της παραμέτρου κίνησης με την MassMobility.
***.car[*].app.typename**	“ClusterRerouting”	Συσχέτιση της παραμέτρου με την εφαρμογή δρομολόγησης

Κάθε όχημα λαμβάνει τα χαρακτηριστικά που θέλουμε για την υλοποίηση. Στην ουσία γνωστοποιούμε στο αρχείο ρυθμίσεων όσα έχουμε συντάξει στην NED μαζί με κάποιες επιπλέον διευκρινίσεις.

Αντίστοιχα για την σταθερή βάση του δικτύου, ακολουθούμε παρόμοια διαδικασία.

Πίνακας 8 Κύριες παράμετροι της σταθερής βάσης

Παράμετρος	Τιμή	Λειτουργία
***.rsu.typename**	rsu	Συσχέτιση με την ενότητα rsu
***.rsu.mobility.typename**	StaticMobility	Η βάση δεν κινείται μέσα στο χώρο
***.rsu.app.typename**	"ClusterRerouting"	Συσχέτιση της παραμέτρου με την εφαρμογή δρομολόγησης

Σειρά έχουν οι τεχνικές παράμετροι που αφορούν τα χαρακτηριστικά δικτύωσης που θα έχουν τα οχήματα και η βάση. Μεταξύ άλλων, είναι πολύ σημαντικό να καθοριστεί η σωστή τεχνολογία επικοινωνίας και για τις δύο ενότητες, όπως έγινε και στο αρχείο NED. Οι ομοιότητες στα χαρακτηριστικά έχουν σκοπό την αποφυγή ανωμαλιών αποστολής σήματος μεταξύ των στοιχείων του δικτύου.

Πίνακας 9 Παράμετροι επικοινωνίας

Παράμετροι	Τιμή	Λειτουργία
***.car[*].wlan[*].radio.typename**	"Ieee80211ScalarRadio"	Τύπος τεχνολογίας wlan
***.car[*].wlan[*].radio.transmitter.power**	30mW	Ισχύς εκπομπής σήματος
***.car[*].wlan[*].radio.displayCommunicationRange**	true	Ενεργοποίηση εικονικής ακτίνας σήματος
***.car[*].wlan[*].radio.bitRate**	6Mbps	Ταχύτητα μετάδοσης δεδομένων
***.car[*].wlan[*].radio.receiver.sensitivity**	-95dBm	Ευαισθησία λήψης
***.car[*].wlan[*].radio.radioMediumModule**	"Ieee80211ScalarRadio"	Παράμετρος επικοινωνίας με το radioMedium
***.rsu.wlan.typename**	"Ieee80211ScalarRadioMedium"	Τύπος τεχνολογίας wlan
***.rsu.wlan[*].radio.transmitter.power**	30mW	Ισχύς εκπομπής σήματος

***.rsu.wlan[*].radio.displayCommuniationRange**	true	Ενεργοποίηση εικονικής ακτίνας σήματος
***.rsu.wlan[*].radio.bitRate**	6Mbps	Ταχύτητα μετάδοσης δεδομένων
***.rsu[*].wlan[*].radio.radioMediumModule**	“Ieee80211ScalarRadioMedium”	Παράμετρος επικοινωνίας με το radioMedium
***.rsu.wlan[*].radio.receiver.sensitivity**	-95dBm	Ευαισθησία λήψης

Ακόμη, τα radioMedium μπορούν να λάβουν δικές τους τιμές στο αρχείο για τις παραμέτρους τους. Η ύπαρξη τους στην προσομοίωση είναι αμοιβαία. Το ένα δεν μπορεί να λειτουργεί χωρίς το άλλο. Οι περισσότερες παράμετροι έχουν μία προκαθορισμένη τιμή και δεν χρειάζεται να θέσουμε εμείς κάποια καινούρια αν δεν το θελήσουμε. Προφανώς και αυτό ισχύει για κάθε ενότητα και υποενότητα σε αυτήν την υλοποίηση.

Πίνακας 10 Παράμετροι radioMediumRSU και radioMediumCar

Παράμετροι	**Τιμή**	**Λειτουργία**
***.radioMediumRSU.typename**	"Ieee80211ScalarRadioMedium"	Τύπος τεχνολογίας wlan
***.radioMediumCar.typename**	"Ieee80211ScalarRadio"	Τύπος τεχνολογίας wlan

Αφότου ολοκληρωθεί η επεξεργασία των στοιχείων του δικτύου, παραμετροποιείται και η κλάση IntegratedVisualizer, που ευθύνεται για την εικονοποίηση του δικτύου στο OMNET++ σε δισδιάστατη μορφή. Παράλληλα, η RoadsCanvasVisualizer δεν δέχεται κάποια μετατροπή στις παραμέτρους της διότι χρησιμοποιείται μόνο για να απεικονίσει τα στοιχεία που συμπεριλαμβάνονται στο αρχείο του SUMO.

Αρχικά, θέλουμε να απεικονίζεται η κίνηση μετάδοσης και αποστολής ενός σήματος μεταξύ των κόμβων. Με τις displaySignals, displaySignalArrivals και displaySignalDepartures, ενεργοποιούμαι την απεικόνιση σημάτων που ξεκινάνε από έναν πομπό, ταξιδεύουν στον χώρο και καταλήγουν στον δέκτη.

```
*.visualizer.*.mediumVisualizer.displaySignals = true

*.visualizer.canvasVisualizer.mediumVisualizer.displaySignalArrivals = true

*.visualizer.canvasVisualizer.mediumVisualizer.displaySignalDepartures = true
```

Κώδικας 42 Παράμετροι απεικόνισης σημάτων

Η signalPropagationAnimationSpeed απεικονίζει την ταχύτητα μετάδοσης του σήματος στον χάρτη, ενώ η signalTransmissionAnimationSpeed απεικονίζει το πόσο γρήγορα μεταδίδεται ένα σήμα από τον πομπό στον δέκτη. Ενδεικτικά, οι τιμές που λαμβάνουν τέτοιου είδους παράμετροι ορίζονται ως τα μέτρα που διανύει ένα σήμα σε σχέση με την ταχύτητα μετάδοσης του φωτός που ισούται περίπου με 3e8=3 X 10^8. Άρα οι τιμές αντιστοιχούν σε μέτρα/3e8.

```
*.visualizer.*.mediumVisualizer.signalPropagationAnimationSpeed = 500/3e8

*.visualizer.*.mediumVisualizer.signalTransmissionAnimationSpeed = 50000/3e8
```

Κώδικας 43 Παράμετροι ταχύτητας αποστολής πακέτων

Παράλληλα, ολοκληρώνοντας την συγγραφή του αρχείου παραμέτρων, χρειάζονται να ρυθμιστούν μερικές παράμετροι που αφορούν τον αλγόριθμο ομαδοποίησης και αποστολής μηνυμάτων που θα αναλυθεί στην συνέχεια. Οι λειτουργίες τους έχουν εξηγηθεί κατά την δήλωση τους στο πρόγραμμα NED.

Πίνακας 11 Τιμές παραμέτρων αλγόριθμου ομαδοποίησης

Παράμετροι	Τιμή
numCars	11
k	2
clusterIterationLimit	10
broadcastTime	30

5.3.4 Ανάλυση υλοποίησης αλγόριθμου ομαδοποίησης και μετάδοσης μηνυμάτων

Η ενότητα αυτή αποτελεί και το τελευταίο τμήμα αυτής της υλοποίησης και αφορά τον αλγόριθμο που θα επιτρέψει στα αυτοκίνητα του δικτύου να λαμβάνουν μηνύματα και να ομαδοποιούνται σε ομάδες με γνώμονα τον δρόμο που θα επιλέξουν να πάρουν κατά την διαδικασία της νέας δρομολόγησης.

Σκοπός όλων των ενοτήτων που αφορούν την υλοποίηση, είναι η προσπάθεια παρουσίασης μίας πειραματικής προσέγγισης επάνω στο ζήτημα διατηρώντας την ουσία του εγχειρήματος. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η C++, κάνοντας την συνεργασία του αλγορίθμου με το πρόγραμμα του OMNET++ λίγο πιο απλή.

Ξεκινώντας με τον αλγόριθμο θα δημιουργήσουμε δύο διαφορετικά αρχεία κλάσεων που θα συνδεθούν ως επικεφαλίδες στο κύριο πρόγραμμα με τον αλγόριθμο. Ο λόγος διαχωρισμού των κλάσεων σε διαφορετικά αρχεία είναι απλός και αφορά την μορφή του κώδικα που θα παραμείνει πιο κατανοητή και διαχειρίσιμη.

Η πρώτη κλάση ουσιαστικά αποτελεί το μήνυμα που θα μεταδοθεί από το πρώτο όχημα σε όλα τα υπόλοιπα. Η μόνο μεταβλητή που περιέχει είναι οι συντεταγμένες του σημείου που έγινε το ατύχημα. Η κλάση δηλώνεται ως **message** διότι αποτελεί μία ειδική κλάση του OMNET++ που επιτρέπει την ανταλλαγή μηνυμάτων μεταξύ ενοτήτων στο πρόγραμμα. Αφού διαβάσει την **message**, το OMNET++ θα δημιουργήσει αυτόματα μία κλάση και θα μπορέσει να αναγνωρίσει τις μεταβλητές εντός της κλάσης ως μηνύματα και να τα θέτει προς αποστολή με την εντολή msg. Το αρχείο της κλάσης χρειάζεται να έχει την κατάληξη .msg και να περιέχει μέσα την βιβλιοθήκη CMessage του OMNET++.

```
#ifndef BROADCAST_MESSAGE_H

#define BROADCAST_MESSAGE_H

#include <iostream>

#include <cmessage.h>

message BroadcastMessage {

        string accidentLocation;}

#endif
```

Κώδικας 44 Αρχείο κλάσης μηνύματος

Το δεύτερο αρχείο κλάσης που θα δημιουργήσουμε αφορά την κλάση που χειρίζεται τον αλγόριθμο ομαδοποίησης. Το αρχείο μας είναι πολύ πρακτικό διότι η κλάση μας επιτρέπει να έχουμε σωστό έλεγχο όλων των μεταβλητών που θα χρησιμοποίησουμε για τον αλγόριθμο ομαδοποίησης.

Στο ξεκίνημα πρέπει να δηλώσουμε ορισμένες ρυθμίσεις για το αρχείο για την αποφυγή λαθών. Πρέπει να δηλωθούν οι βιβλιοθήκες **omnetpp.h**, **csimplemodule.h**, **cmessage.h**. Επίσης για να χρησιμοποιηθούν οι κλάσεις του OMNET++ πρέπει να υπάρχει και η εντολή **using namespace omnetpp**.

```
#ifndef CLUSTER_REROUTING_H

#define CLUSTER_REROUTING_H

#include <omnetpp.h>

#include <csimplemodule.h>

#include <cmessage.h>

using namespace omnetpp;
```

Κώδικας 45 Βιβλιοθήκες αρχείο κεφαλίδας εφαρμογής

Η κλάση θα έχει όνομα ClusterRerouting και θα κληρονομεί από την CSimpleModule διότι χρειαζόμαστε συναρτήσεις της όπως η initialize() και η handleMessage(). Οι μεταβλητές της κλάσης θα είναι ιδιωτικές εκτός από δύο συναρτήσεις της csimplemodule που θα είναι προστατευμένες. Συνεπώς σε αυτήν την κλάση φιλοξενούνται τα στοιχεία που θα καλούνται στο κύριο πρόγραμμα για να επεξεργαστούμε κάθε τμήμα της υλοποίησης.

```
Class ClusterRerouting : public CSimpleModule {
    private:
        Int numCars; //Αριθμός οχημάτων
        Int k; //Αριθμός θεμιτών ομάδων
        Int clusterIterationLimit; //Όριο επαναλήψεων αλγόριθμου ομαδοποίησης
        double broadcastTime; //Χρονικό σημείο μετάδοσης μηνύματος
        bool receiveMessage; // Σημαία που δηλώνει αν έχει ληφθεί το μήνυμα ή όχι
        void rerouteAfterAccident(); //Συνάρτηση έναρξης κώδικα ομαδοποίησης
        void rerouteCar(int carIndex, const char* route); // Νέα δρομολόγηση
    protected:
        virtual void initialize() override; // Αρχικοποίηση αλγόριθμου μετάδοσης μηνύματος
        virtual void handleMessage (cMessage *msg) override; }; // Διαχείριση μηνύματος
```

Κώδικας 46 Ορισμός κλάσης εφαρμογής

Πλέον με την ύπαρξη των δύο κλάσεων που είναι αναγκαίες για την υλοποίηση, μπορούμε να περάσουμε στο κύριο πρόγραμμα. Αναφορικά σκοπός είναι, με την αποστολή του μηνύματος για την ύπαρξη κλειστού δρόμου από το πρώτο αμάξι της προσομοίωσης, τα υπόλοιπα αμάξια πρέπει να επιλέξουν άλλο δρόμο διαφυγής για να συνεχίσουν την πορεία τους και να αποφευχθεί η συμφόρηση. Ανάλογα με ποιο μονοπάτι δρόμου διαλέξουν, που στην περίπτωση μας είναι δύο, θα χωριστούν και σε μία ομάδα μέσω

αλγόριθμου ομαδοποίησης κ-μέσων. Θεωρητικά το κεντροειδές στοιχείο του πλήθους είναι το όχημα που θα επιλέξει πρώτο την διαδρομή, με αποτέλεσμα να το ακολουθήσουν και άλλα στοιχεία. Αν ένα πεπερασμένο πλήθος οχημάτων χωριστεί σε δύο ίσα μέρη, εκτός του πρώτου, τότε επιτυγχάνεται διαμοιρασμός οχημάτων σε διαφορετικές πορείες δρόμων. Η υλοποίηση δεν θέτει πρακτικά κάποιο όριο στο πλήθος κόμβων-οχημάτων που μπορεί να επεξεργαστεί. Ωστόσο, επειδή αποτελεί μια πειραματική και αρχάρια προσέγγιση, τα δεδομένα που επεξεργάζονται διατηρούνται σε ένα σχετικά μικρό πλήθος.

Στην αρχή, πρέπει να συμπεριληφθούν όλες οι βιβλιοθήκες και κλάσεις που είναι αναγκαίες. Επίσης θέλουμε να θέσουμε πάλι το πεδίο namespace με τις ιδιότητες του OMNET++.

```
#include “ClusterRerouting.h” //Δήλωση κλάσης ClusterRerouting

#include “BroadcastMessage.h” //Δήλωση κλάσης BroadcastMessage

#include <omnetpp.h> //Εισαγωγή του framework omnet++

#include <vector> // Χρήση της vector για διαχείριση διανυσμάτων

#include <cmath> //Χρήση της cmath για διαχείριση μαθηματικών συναρτήσεων όπως abs() και pow()

#include <iostream> //Χρήση της iostream για εκτυπώσεις αποτελεσμάτων

#include <limits> //Χρήση της limits για συναρτήσεις όπως numeric_limits

#include <algorithm> //Χρήση της algorithm για συναρτήσεις όπως min() και max()

#include <random> //Χρήση της random για αλγόριθμους παραγωγής τυχαίων αριθμών

#include <csimplemodule.h> //Χρήση της csimplemodule για συναρτήσεις όπως initialize()

#include <cmessage.h> //Χρήση της cmessage για την διαχείριση μηνυμάτων μεταξύ ενοτήτων

using namespace omntepp;
```

Κώδικας 47 Βιβλιοθήκες προγράμματος εφαρμογής

Έπειτα, πρέπει να γνωστοποιήσουμε στο πρόγραμμα της C++ την ενότητα που θα χρησιμοποιηθεί ως κόμβος-στοιχείο στον αλγόριθμο. Η εντολή είναι η **Define_Module(CarType)**, όπου εσωτερικά βρίσκεται το όνομα της ενότητας που θέλουμε να χρησιμοποιήσουμε.

Στην αρχή του προγράμματος, θέλουμε να αρχικοποιήσουμε τα δεδομένα της κλάσης ClusterRerouting ώστε να μπορεί να τα αναγνωρίσει το σύστημα.

```
void ClusterRerouting::initialize() {
   // Ανάγνωση των παραμέτρων από το αρχείο .ini
   numCars = getParentModule()->par("numCars");          // Αριθμός οχημάτων
   k = par("k").intValue();                              // Αριθμός clusters για K-means
   clusterIterationLimit = par("clusterIterationLimit");  // Όριο επαναλήψεων του K-means
   // Χρονοπρογραμματισμός του πρώτου μηνύματος broadcast αν το όχημα είναι το πρώτο
   if (getIndex() == 0) {
      broadcastTime = par("broadcastTime").doubleValue(); // Χρόνος για μετάδοση από το πρώτο όχημα
      scheduleAt(simTime() + broadcastTime, new cMessage("broadcastMessage"));
   } else {
      receiveMessage = false;
   }}
```

Κώδικας 48 Αρχικοποίηση δεδομένων εφαρμογής

Αυτό που πραγματοποιείται στο παραπάνω τμήμα κώδικα είναι η δήλωση των μεταβλητών για το πρόγραμμα και η αντιστοίχιση των τιμών τους με αυτές που έχουμε θέσει στο αρχείο παραμέτρων τύπου .ini νωρίτερα. Για να λάβει την τιμή η μεταβλητή στο πρόγραμμα χρειάζεται η συνάρτηση **par()** η οποία

θα αντιστοιχίσει τις τιμές με το αρχείο παραμέτρων. Αρκεί τα δύο προγράμματα να βρίσκονται στο ίδιο φάκελο στο omnet++.

Μετά έχει σειρά η δόμηση της συνάρτησης handleMessage για την αποστολή και διαχείριση μηνυμάτων. Εδώ θέλουμε να ελέγξουμε αν το πρώτο όχημα είναι αυτό που θα στείλει το μήνυμα, αν το μήνυμα που θα στείλει είναι το σωστό και αν τα υπόλοιπα οχήματα θα το λάβουν, προκειμένου να ξεκινήσει ο αλγόριθμος νέας δρομολόγησης και ομαδοποίησης.

```
void ClusterRerouting::handleMessage(cMessage *msg) {
  //Έλεγχος αν το μήνυμα είναι το broadcastMessage και αν αυτό είναι το πρώτο όχημα
  if (strcmp(msg->getName(), "broadcastMessage") == 0 && getIndex() == 0) {
   // Δημιουργία και αποστολή μηνύματος με Broadcast
    BroadcastMessage *bcMessage = new BroadcastMessage("accidentWarning");
    bcMessage->setAccidentLocation("centralJunction"); // Περιοχή ατυχήματος
    send(bcMessage, "out"); // Στο τμήμα “out” θεωρείται η θύρα λήψης του CarType, upperLayerOut
    delete msg;  } // Μετά την αποστολή, το μήνυμα διαγράφεται για αποφυγή λαθών
```

Κώδικας 49 Διαχείριση δεδομένων

Αν το πρώτο όχημα έχει στείλει το σωστό μήνυμα με broadcast ξεκινά η συνάρτηση rerouteAfterAccident που αντιπροσωπεύει τον αλγόριθμο ομαδοποίησης κ-μέσων. Αν ένα όχημα λάβει δεύτερη φορά το ίδιο μήνυμα, θα επιστραφεί μήνυμα πίσω στο πρώτο όχημα για να το ενημερώσει.

```
else if (BroadcastMessage *bcMessage = dynamic_cast<BroadcastMessage *>(msg)) {
    EV << "Το όχημα " << getIndex() << " έλαβε προειδοποίηση ατυχήματος, ξεκινά αλλαγή πορείας\n";
    receiveMessage = true; // Σημαία επιβεβαίωσης λήψης μηνύματος από τα οχήματα
    rerouteAfterAccident(); // Συνάρτηση αλγόριθμου ομαδοποίησης κ-μέσων
    delete bcMessage; // Διαγραφή μηνύματος επιβεβαίωσης μετά την αποστολή
  } else {
    delete msg;
  }}
```

Κώδικας 50 Αποστολή μηνύματος επιβεβαίωσης

Εφόσον σταλούν τα μηνύματα και κληθεί η συνάρτηση για τον αλγόριθμο ομαδοποίησης από την handleMessage, σειρά έχει να συνταχθεί η rerouteAfterAccident.

```
void ClusterRerouting::rerouteAfterAccident() {
   int numCars = getParentModule()->par("numCars");  // Ανάκτηση αριθμού οχημάτων
   std::vector<int> carIndices(numCars - 1); //Λίστα δεικτών οχημάτων εκτός του πρώτου
   std::iota(carIndices.begin(), carIndices.end(), 1); // Αρχικοποίηση της λίστας με όριο το numCars-1
   std::vector<double> clusterCenters(k); // Διαφορετική λίστα για τα κέντρα των ομάδων
   std::mt19937 rng; // Γεννήτρια παραγωγής τυχαίων αριθμών Mersenne Twister
   rng.seed(std::random_device()()); // Αρχή παραγωγής τυχαίων αριθμών με την μέθοδο random_device
   std::uniform_real_distribution<double> dist(0, numCars - 1); // Διανομή τυχαίων αριθμών στην λίστα
   for (int i = 0; i < k; ++i) {
      clusterCenters[i] = dist(rng); // Ανάθεση τυχαίων θέσεων για τα κέντρα
   }
```

Κώδικας 51 Προετοιμασία αλγόριθμου ομαδοποίησης

Η παρακάτω μέθοδος αναπαριστά την προσπάθεια ανάθεσης των οχημάτων σε clusters με βάση την κοντινότερη απόσταση που έχει το κάθε σημείο από το κεντρικό ενός cluster.

```
std::vector<int> clusters(numCars - 1); // Λίστα που θα δείχνει το cluster που ανήκει το κάθε όχημα
for (int iter = 0; iter < clusterIterationLimit; ++iter) {
   for (int i = 1; i < numCars; ++i) {
      double minDist = std::numeric_limits<double>::max();
      int clusterIdx = 0;
      for (int j = 0; j < k; ++j) {
         double dist = std::abs(i - clusterCenters[j]);
         if (dist < minDist) {
            minDist = dist;
            clusterIdx = j;
         }
      }
      clusters[i - 1] = clusterIdx;
   }
```

Κώδικας 52 Πρώτο στάδιο ομαδοποίησης

Αφού ανατεθούν όλα τα οχήματα σε ένα cluster για πρώτη φορά, υπολογίζουμε ξανά τα κεντρικά σημεία των clusters.

```
    for (int j = 0; j < k; ++j) {
      double sum = 0;
      int count = 0;
      for (int i = 0; i < numCars - 1; ++i) {
        if (clusters[i] == j) {
          sum += i + 1;
          count++;
        }
      }
      if (count > 0) {
        clusterCenters[j] = sum / count;
      }
    }
  }
```

Κώδικας 53 Εκ νέου υπολογισμός clusters

Τέλος, σύμφωνα με τα clusters θα συμμετέχει και μία τελευταία συνθήκη που θα ορίζει την νέα δρομολόγηση των οχημάτων ανάλογα με την ομάδα που ανήκει κάθε όχημα. Οι δρόμοι που επιλέγονται μέσω της rerouteCar βασίζονται στο οδικό δίκτυο που χτίστηκε στην υλοποίηση του SUMO και ουσιαστικά αντιστοιχίζονται με την δεξιά και αριστερή παράκαμψη του σταυροδρομιού που φτάνουν στην απέναντι παράκαμψη του δρόμου.

```
    for (int i = 1; i < numCars; ++i) {
        if (clusters[i - 1] == 0) {
            cout << "Το όχημα " << i << " επιλέγει τη Διαδρομή A (Cluster 1)\n";
            rerouteCar(i, "RouteA"); // Κλήση συνάρτησης για δεξιά παράκαμψη δρόμου
        } else {
            cout << "Το όχημα " << i << " επιλέγει τη Διαδρομή B (Cluster 2)\n";
            rerouteCar(i, "RouteB"); // Κλήση συνάρτησης για αριστερή παράκαμψη δρόμου
        }
    }
}
```

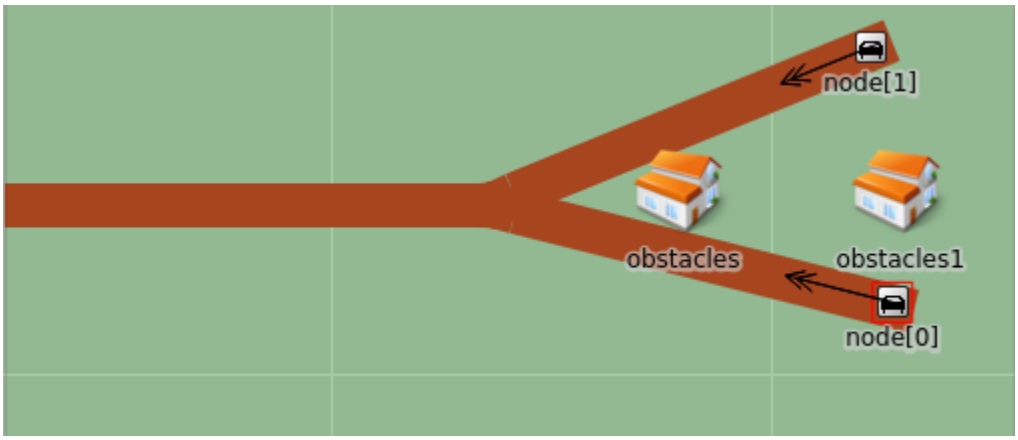
Κώδικας 54 Συνθήκη ορισμού νέου δρομολογίου οχημάτων

Με την σύνταξη και του προγράμματος μετάδοσης μηνυμάτων και ομαδοποίησης, τα βήματα της υλοποίησης ολοκληρώνονται. Τα αποτελέσματα ποικίλουν ανάλογα με τις παραμέτρους, τις επιλογές και τις απαιτήσεις που θα τεθούν ως δεδομένα κατά την υλοποίηση. Είναι πρέπον να αναφερθεί πως η υλοποίηση αυτή προσπαθεί να δώσει σημασία στα βήματα που πρέπει να ακολουθήσει κάποιος και δεν αποσκοπεί στο να παράγει αποτελέσματα. Σκοπός είναι η απόδειξη ότι το VEINS framework και τα λοιπά εργαλεία που χρησιμοποιήθηκαν, μπορούν να συνεργαστούν για την υλοποίηση μιας εργασίας με θεματολογία τα VANETs και όχι μόνο.

5.4 Υλοποίηση απλής εφαρμογής Vanet με Veins framework

Βέβαια, υλοποιώντας παράλληλα μία εφαρμογή με πολύ λιγότερες απαιτήσεις, μπορούμε να καταλήξουμε σε ένα σημείο όπου το αποτέλεσμα είναι υπαρκτό. Σκοπός αυτής της εφαρμογής είναι να απεικονίσει ένα πραγματικό αποτέλεσμα όπου, δύο οχήματα που λειτουργούν ως κόμβοι, μπορούν να έχουν επικοινωνία μεταξύ τους με ασύρματη σύνδεση. Σε αυτήν την περίπτωση, δεν εφαρμόζεται κάποιος αλγόριθμος ομαδοποίησης όπως αναφερθήκαμε προηγουμένως, αλλά συντίθεται ένα πρόγραμμα που αφορά το framework του Veins και το περιβάλλον του SUMO και του OMNET++.

Το σενάριο της προσομοίωσης αφορά δύο οχήματα που οδηγούν αντίστοιχα σε δύο διαφορετικούς δρόμους. Αυτοί οι δύο δρόμοι συνδέονται, δημιουργώντας ένα νοητό δέλτα, και η πορεία συνεχίζει σε ένα μοναδικό δρόμο. Οι δύο διαφορετικοί δρόμοι, χωρίζονται από πλήθος κτηρίων που αποτελούν εμπόδιο για την ανίχνευση κινούμενων οχημάτων από τον απέναντι δρόμο. Έτσι, οι δύο κόμβοι πρέπει να επικοινωνήσουν ασύρματα προκειμένου να μην συγκρουστούν όταν οι δρόμοι καταλήξουν να συνδέονται στον έναν ενιαίο.



Εικόνα 33 Δίκτυο Vanet

Το παραπάνω οδικό δίκτυο έχει σχεδιαστεί στο περιβάλλον του netedit, το εμβόλιμο πρόγραμμα του SUMO για σχεδιασμό οδικών δικτύων. Κάθε στοιχείο του οδικού δικτύου μεταφράζεται σε κώδικα XML προκειμένου να μπορέσουν να το διαβάσουν και άλλα προγράμματα όπως το OMNET++ και το framework του Veins.

Ο κώδικας που παράγεται από το τελικό αρχείο του SUMO δεν διαφέρει τεχνικά από το παράδειγμα της ενότητα 5.3.4, όπου μεταξύ άλλων υπάρχουν scripts που διαμορφώνουν τις ακμές των δρόμων πριν και αφότου ενωθούν:

```
<edge id="gneE2" from="gneJ2" to="gneJ3" priority="-1">
    <lane id="gneE2_0" index="0" speed="60.00" length="100.00" shape="144.17,95.84 58.23,60.87"/>
</edge>
```

Κώδικας 55 Λωρίδα δρόμου

Αλλά επίσης και scripts που δηλώνουν την ύπαρξη ενός σταυροδρομιού ή γενικότερης ένωσης πολλών ακμών δρόμων:

```
<junction id="gneJ3" type="priority" x="52.42" y="56.78" incLanes="gneE2_0 gneE3_0" intLanes=":gneJ3_0_0 :gneJ3_1_0" shape="57.63,62.35 58.83,59.39 58.22,59.04 58.16,58.88 58.25,58.73 58.49,58.59 58.90,58.46 58.13,55.36 55.26,56.15 54.25,56.43 53.24,56.63 51.99,56.74 50.29,56.78 50.29,59.98 53.12,60.27 54.07,60.62 55.00,61.09 56.11,61.67">
    <request index="0" response="00" foes="10" cont="0"/>
    <request index="1" response="01" foes="01" cont="0"/>
```

Κώδικας 56 Κώδικας για σταυροδρόμι

Ακόμη, συμπεριλαμβάνονται και scripts που ορίζουν τις διαδρομές των κόμβων όπως:

```
<trip id="vehicle_0" depart="0.00" departSpeed="25.00" color="yellow" from="gneE3" to="gneE4"/>
<trip id="vehicle_1" depart="0.00" departSpeed="22.00" color="red" from="gneE2" to="gneE4"/>
```

Κώδικας 57 Κώδικας διαδρομών οχημάτων

Τέλος, στα αρχεία του SUMO καλό είναι να υπάρχει και το σχετικό αρχείο διαμόρφωσης launchd που επιτρέπει την ομαλή επικοινωνία του SUMO με τα υπόλοιπα προγράμματα όπως το Veins και το OMNET++. Η μορφή του αρχείου εσωτερικά διαμορφώνεται κυρίως χειροκίνητα και είναι αρκετά χρήσιμη η παρουσία του.

```
<?xml version="1.0"?>
<launch>
<copy file="vanet.net.xml" />
<copy file="vanet.rou.xml" />
<copy file="vanet.sumocfg" type="config" /> <!—Sumocfg είναι το αρχείο διαμόρφωσης μόνο για το SUMO, προκειμένου να τρέξει η προσομοίωση ολοκληρωμένα -->
</launch>
```

Κώδικας 58 Κώδικας αρχείου launchd

Ξεκινώντας τώρα με το αρχείο NED, θέλουμε αρχικά να φτιάξουμε την σύνθετη ενότητα car.

```
module car extends VeinsInetCar{
}
```

Κώδικας 59 Όνομα σύνθετης ενότητας κόμβων

Δεν είναι ανάγκη να προστεθεί κάτι επιπλέον εντός της ενότητας μιας και παρατηρούμε πως κληρονομεί τα στοιχεία μίας έτοιμης οντότητας του Veins με όνομα VeinsInetCar η οποία χρησιμοποιείται από το framework για προσομοιώσεις με οχήματα.

Έπειτα, ορίζουμε το δίκτυο στο οποίο θα τρέξει η ενότητα **car** και παράλληλα θα δοθεί παράμετρος που θα σχηματοποιεί το εικονικό πλαίσιο που θα εκτυλίσσονται τα πάντα, ορίζοντας διαστάσεις και χρώματα.

```
network Vanet
{
   parameters:
      @display("bgb=650,500;bgg=100,1,grey95");
```

Κώδικας 60 Αρχικοποίηση δικτύου

Στην συνέχεια σειρά έχουν όλες οι υποενότητες που πρέπει να δηλωθούν για να είναι λειτουργικό το δίκτυο. Ανάμεσα τους θα βρεθούν υποενότητες που ορίζουν τον κόμβο σαν οντότητα, το μέσο επικοινωνίας για το δίκτυο, και τους εικονοποιητές της προσομοίωσης. Συνολικά παραθέτονται παρακάτω σε μορφή πίνακα. Κάθε νέα υποενότητα κληρονομεί από μία ήδη υπάρχουσα του inet ή του Veins framework τις οποίες όπως και στην προηγούμενη υλοποίηση τις προσθέτουμε με την **import**.

Σημαντικό είναι να αναφερθεί πως μεταξύ των υπόλοιπων υποενοτήτων που χρησιμοποιούνται, η υποενότητα node[0] είναι αυτή που ξεχωρίζει σε αυτήν την περίπτωση. Δηλώνεται όπως και όλες οι υπόλοιπες υποενότητες και αποτελεί την οντότητα του κόμβου, δηλαδή του οχήματος, που θα επικοινωνεί στο δίκτυο. Το Veins μέσω της ενότητα VeinsInetCar μετατρέπει εικονικά και πρακτικά έναν κοινό κόμβο δικτύου σε ένα όχημα που κινείται σε οδικό δίκτυο και έχει τις αντίστοιχες ιδιότητες διασύνδεσης. Η τιμή μηδέν μέσα σε αγκύλες βοηθά το Veins να καταλάβει πως κάθε μία ενέργεια του προγράμματος ξεκινά από τον πρώτο κόμβο πάντα, ο οποίος μπορεί να βρίσκεται μέσα σε ένα σύνολο από άλλους κόμβους. Δηλώνοντας την υποενότητα ως node[0] δεν παρατηρείται κάποια επιπλοκή στον κώδικα όταν θέλουμε να δώσουμε το δικαίωμα και σε άλλους κόμβους να κάνουν χρήση κάποιας εφαρμογής.

Πίνακας 12 Υποενότητες δικτύου Vanet

Υποενότητα	Κληρονομεί από:	Λειτουργία
visualizer	IIntegratedVisualizer	Απεικόνιση όσων αφορούν το δίκτυο.
configurator	Ipv4NetworkConfigurator	Ρύθμιση διευθύνσεων κόμβων για αποστολή μηνυμάτων.
radioMedium	Ieee80211ScalarRadiomedium	Μέσο επικοινωνίας για ασύρματη σύνδεση.
roadsCanvasVisualizer	RoadsCanvasVisualizer	Απεικόνιση του οδικού δικτύου SUMO.
manager	VeinsInetManager	Ρύθμιση διεπαφής μεταξύ Veins και SUMO.
physicalEnviroment	PhysicalEnviroment	Απεικόνιση φυσικών οντοτήτων σε ένα χάρτη.
obstacles	ObstacleControl	Απεικόνιση εμποδίων, όπως κτίρια.
Node[0]	car	Κόμβος που θα κληρονομεί από την ενότητα car.
Obstacles1	ObstacleControl	Απεικόνιση εμποδίων, όπως κτίρια.

Έπειτα και αφού ολοκληρώσουμε με το αρχείο NED πρέπει να συντάξουμε και το αντίστοιχο αρχείο τύπου .ini.

Οι ρυθμίσεις του αρχείου μπορούν να χωριστούν σε τμήματα για να είναι πιο κατανοητά.

Αρχικά πρέπει να ρυθμίσουμε τις παραμέτρους των κόμβων που σχετίζονται με την εφαρμογή που θα χρησιμοποιηθεί (βλ. παρακάτω για κώδικα).

```
*.node[*].numApps = 1 #Αριθμός εφαρμογών που θα χρησιμοποιήσουν οι κόμβοι. Μία ο καθένας

*.node[1].app[0].typename = "UdpBasicApp" #Εφαρμογή του node[1] που θα στείλει το μήνυμα

*.node[1].app[0].stopTime = 0.11s #Χρόνος προσομοίωσης που θα σταματήσει η αποστολή μηνυμάτων

*.node[1].app[0].destAddresses = "node[0]" # Τα μηνύματα θα σταλούν στον άλλο κόμβο, τον node[0]

*.node[1].app[0].destPort = 5000 #Θύρα επικοινωνίας των δύο κόμβων

*.node[1].app[0].messageLength = 1000B #Μέγεθος του κάθε μηνύματος σε χωρητικότητα

*.node[1].app[0].sendInterval = exponential(12ms) #Χρόνος παύσης πριν την αποστολή επόμενου μηνύματος

*.node[1].app[0].packetName = "Warning" #Περιεχόμενο του πακέτου μηνύματος του node[1]

*.node[1].wlan[0].opMode = "p" #Η επικοινωνία μέσω της διεπαφής wlan0 θα είναι peer-to-peer

*.node[1].wlan[0].radio.bandName = "5.9 GHz" #Μπάντα συχνότητας

*.node[1].wlan[0].radio.channelNumber = 3 #Κανάλι επικοινωνίας

*.node[1].wlan[0].radio.transmitter.power = 50mW #Ισχύς αποστολής μηνύματος

*.node[1].wlan[0].radio.bandwidth = 30 MHz #Εύρος ζώνης επικοινωνίας.
```

Κώδικας 61 Παράμετροι node[1]

Αντίστοιχα εργαζόμαστε και για τον node[0], οπότε πολλές παράμετροι δεν παρουσιάζουν καμία αλλαγή.

```
*.node[0].app[0].typename = "UdpSink" #Εφαρμογή του node[0] που θα λάβει το μήνυμα

*.node[0].app[0].destAddresses = "node[1]" # Επικοινωνεί μόνο με τον node[1]
```

Κώδικας 62 Παράμετροι node[0]

Στην συνέχεια πρέπει να καθορίσουμε παραμέτρους που αφορούν την κινητικότητα που θα έχουν οι κεραίες που σχετίζονται με κάθε κόμβο ξεχωριστά. Για το Veins, είναι σημαντικό να γνωρίζει πως θέλουμε να διαχειριστεί τις κεραίες όταν αυτές βρίσκονται θεωρητικά επάνω σε οχήματα τα οποία βρίσκονται σε κίνηση.

Η κινητικότητα της κεραίας συνάδει μόνιμα με την κίνηση που πραγματοποιεί ο κόμβος που την κατέχει. Έτσι μπορούμε να απεικονίσουμε την αποστολή σήματος από έναν κόμβο και όχι από κάποιο άλλο τυχαίο σημείο.

```
*.node[*].wlan[*].radio.antenna.mobility.typename = "AttachedMobility"
*.node[*].wlan[*].radio.antenna.mobility.mobilityModule = "^.^.^.^.mobility"
```

Κώδικας 63 Κινητικότητα κεραίας κόμβου

Επίσης, θέλουμε οι κεραίες να μην έχουν περιορισμό στην κίνηση που θα πραγματοποιούν στον χάρτη και να ακολουθούν μονίμως την κίνηση του κόμβου, για αυτό και θέτουμε την τιμή 0 παντού. Αυτές οι τιμές είναι αναγκαίες προκειμένου το Veins να ξέρει ότι η κινητικότητα της κεραίας είναι καθορισμένη από την αρχή, διότι δεν μπορεί το framework να πάρει κάποια πρωτοβουλία σχετικά με αυτό.

```
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMinX = 0m
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMaxX = 0m
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMinY = 0m
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMaxY = 0m
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMinZ = 0m
*.node[*].wlan[*].radio.antenna.mobility.constraintAreaMaxZ = 0m
```

Κώδικας 64 Περιθώρια κινητικότητας κεραίας κόμβου

Αμέσως μετά πρέπει να ρυθμίσουμε τις διευθύνσεις Ipv4 του δικτύου για να είναι επιτυχείς οι αποστολές μηνυμάτων. Με την HostAutoConfigurator η απόδοση διευθύνσεων όπως και άλλων στοιχείων για επικοινωνία δημιουργούνται αυτόματα και δεν χρειάζεται να κάνουμε κάτι άλλο εμείς. Επίσης, τα στοιχεία δικτύου του κόμβου θα αποδοθούν στην διεπαφή ασύρματης επικοινωνίας wlan0. Τέλος, οι διευθύνσεις των κόμβων που θα συμμετέχουν στην επικοινωνία θέλουμε να είναι στην ομάδα διευθύνσεων 224.0.0.1 για multicast.

```
*.node[*].ipv4.configurator.typename = "HostAutoConfigurator"

*.node[*].ipv4.configurator.interfaces = "wlan0"

*.node[*].ipv4.configurator.mcastGroups = "224.0.0.1"
```

Κώδικας 65 Παράμετροι επικοινωνίας κόμβων

Παράλληλα, χρειάζονται ρύθμιση και παράμετροι που αφορούν την προσομοίωση και την απεικόνιση της. Πρώτον επιλέγουμε η προσομοίωση να ανανεώνει το βηματισμό της κάθε 0.1 δευτερόλεπτα. Θέλουμε η προσομοίωση να τρέχει στο δικό μας σύστημα μόνο, να επικοινωνεί με τον manager μόνο στην θύρα 9999 και να κλείνει αυτόματα κατά την ολοκλήρωση.

```
*.manager.updateInterval = 0.1s

*.manager.host = "localhost"

*.manager.port = 9999

*.manager.autoShutdown = true
```

Κώδικας 66 Παράμετροι διαχειριστή προσομοίωσης και δεδομένων

Για να επισυνάψουμε τα αρχεία του SUMO με την προσομοίωση πρέπει να συμπεριλάβουμε το αρχείο vanet.launchd.xml. Ο manager αντιστοιχίζει την ενότητα των κόμβων με την ενότητα του Veins, VeinsInetCar, όπως επίσης και την κινητικότητα με την VeinsInetMobility. Με αυτό τον τρόπο είναι

σίγουρο ότι το Veins θα εφαρμόσει σωστά τις ιδιότητες του στην προσομοίωση και οι κόμβοι θα συμπεριφέρονται ως οχήματα που επικοινωνούν ασύρματα.

```
*.manager.launchConfig = xmldoc("vanet.launchd.xml")

*.manager.moduleType = "org.car2x.veins.subprojects.veins_inet.VeinsInetCar"

*.node[*].mobility.typename = "VeinsInetMobility"
```

Κώδικας 67 Σύνδεση Veins με SUMO

Μία ακόμη παράμετρος που είναι προαιρετική αλλά βοηθά στην δόμηση του χάρτη της προσομοίωσης, είναι η προσθήκη του αρχείου vanet.xml που ορίζει το πράσινο πλαίσιο που φαίνεται και στην εικόνα 33.

```
*.physicalEnvironment.config = xmldoc("vanet.xml")
```

Κώδικας 68 Προσθήκη προαιρετικού αρχείου προσομοίωσης

Τέλος, θέλουμε μερικές παραμέτρους που θα βοηθήσουν στην απεικόνιση τόσο του δικτύου σαν σύνολο, όσο και κάποιων επιπρόσθετων γραφικών που θα κάνουν την προσομοίωση πιο ευανάγνωστη. Για αυτό πρέπει να γνωστοποιήσουμε στο πρόγραμμα ότι θέλουμε στην ουσία απεικόνιση δυσδιάστατη.

```
[Config plain] #Απλή και γενική απεικόνιση χωρίς πρόσθετα δυσδιάστατα γραφικά

[Config canvas] #Εισαγωγή πρόσθετων

extends = plain #Τα δυσδιάστατα γραφικά χρειάζονται ως βάση την απλή απεικόνιση προσομοίωσης

description = "Enable enhanced 2D visualization" #Προαιρετική συμπερίληψη περιγραφής

**.radio.displayCommunicationRange = true #Απεικόνιση εύρους ακτίνας σήματος κόμβων

*.visualizer.mediumVisualizer.displaySignals = true #Απεικόνιση το σημάτων που ταξιδεύουν

*.visualizer.mobilityVisualizer.displayMovementTrails = true #Απεικόνιση από τα ίχνη πορείας κόμβων

*.visualizer.mobilityVisualizer.displayVelocities = true #Απεικόνιση βέλους πορείας κόμβων

*.roadsCanvasVisualizer.lineColor = "firebrick2" #Χρώμα του δρόμου της προσομοίωσης

*.roadsCanvasVisualizer.lineWidth = 10 #Πλάτος του δρόμου της προσομοίωσης
```

Κώδικας 69 Παράμετροι εικονοποίησης προσομοίωσης

Στο τελευταίο στάδιο της απλής αυτής υλοποίησης, πρέπει να συνδέσουμε το κενό, μέχρι τώρα, δίκτυο με μία ή περισσότερες εφαρμογές προκειμένου να αποδειχθεί πρακτικά πως είναι εφικτή η ασύρματη επικοινωνία δύο οχημάτων σε επίπεδο προσομοίωσης. Το omnet++, μας δίνει την δυνατότητα να δανειστούμε αρχεία, όπως εφαρμογές και βιβλιοθήκες, από έτοιμα αρχεία που παρέχουν τα frameworks inet και veins. Στην συγκεκριμένη περίπτωση θα δανειστούμε δύο βασικές εφαρμογές που παρέχει το framework του inet και θα τροποποιήσουμε κατάλληλα τις παραμέτρους του για να μπορέσουμε να έχουμε το καλύτερο εφικτό αποτέλεσμα. Η μόνη προεργασία που χρειάζεται ένας φάκελος με ένα δικό μας δίκτυο είναι να μοιράζεται τα αρχεία που έχουν οι φάκελοι των frameworks. Αυτό είναι μία τεχνική που αποδίδει διαρκώς σε κάθε τύπου εργασία που θέλει κάποιος να ασχοληθεί. Η σύνδεση των αρχείων γίνεται μέσω της σελίδας project references στην επιλογή ιδιοτήτων του φακέλου της εργασίας.

Θέλοντας να υλοποιήσουμε την ασύρματη επικοινωνία μεταξύ δύο κόμβων θα χρησιμοποιηθούν δύο βασικές εφαρμογές του inet framework, η UdpBasicApp και η UdpSink. Όπως εξηγήθηκε και στην ανάλυση του αρχείο τύπου .ini προηγουμένως, η πρώτη εφαρμογή επιτρέπει την αποστολή πακέτων δεδομένων

και η δεύτερη επιτρέπει την λήψη αυτών των πακέτων με βάση το πρωτόκολλο UDP που είναι ιδανικό για την περίπτωση της εργασίας.

Και οι δύο εφαρμογές αφότου αφομοιωθούν με τα αρχεία του δικτύου μπορούν να παράξουν κάποιο αποτέλεσμα μιας και στην φύση του αυτό το εγχείρημα δεν είναι πολύπλοκο. Το omnet++ φροντίζει να παρέχει αυτές τις έτοιμες λύσεις ώστε ο δημιουργός να ασχοληθεί μόνο με το πως να διευρύνει την προσομοίωση του, χωρίς να ξοδέψει περιττό χρόνο φτιάχνοντας από την αρχή μία εφαρμογή που θα μπορούσε απλώς να κάνει κάτι παρόμοιο. Παράλληλα, αυτό δεν αποκλείει το γεγονός ότι ο δημιουργός του δικτύου μπορεί να προσθέσει επιπλέον κώδικα σε ξεχωριστά αρχεία που θα βασίζονται απλώς στις συγκεκριμένες εφαρμογές. Κάτι ανάλογο έχει συμβεί και με την συγκεκριμένη εργασία, αλλάζοντας απλώς ορισμένες παραμέτρους που μας δίνονται για επεξεργασία. Όλες οι παράμετροι έχουν ήδη αναλυθεί παραπάνω στο αρχείο .ini.

```
localPort = par("localPort");
destPort = par("destPort");
startTime = par("startTime");
stopTime = par("stopTime");
packetName = par("packetName");
dontFragment = par("dontFragment");
if (stopTime >= SIMTIME_ZERO && stopTime < startTime)
    throw cRuntimeError("Invalid startTime/stopTime parameters");
selfMsg = new cMessage("sendTimer");
```

Εικόνα 34 Τμήμα κώδικα UDPBasicAPP με παραμέτρους

Αντίστοιχα για την εφαρμογή UdpSink ισχύει κάτι ανάλογο.

```
localPort = par("localPort");
startTime = par("startTime");
stopTime = par("stopTime");
if (stopTime >= SIMTIME_ZERO && stopTime < startTime)
    throw cRuntimeError("Invalid startTime/stopTime parameters");
selfMsg = new cMessage("UDPSinkTimer");
```

Εικόνα 35 Τμήμα κώδικα UDPSinkAPP με παραμέτρους

Παράλληλα, μπορούμε να αλλάξουμε τιμές και μεταβλητές που αφορούν τον χρόνο ή το περιεχόμενο των πακέτων μηνυμάτων που θα στείλει ο ένας κόμβος στον άλλο. Αυτές οι παραμετροποιήσεις αφορούν και πάλι το αρχείο .ini που αναφέρθηκε παραπάνω.

Οι δύο εφαρμογές στην ουσία δομούνται από μία μεγάλη σειρά εκτέλεσης διάφορων συναρτήσεων, όπου κάθε μία χειρίζεται κάτι μοναδικό που αφορά την ανταλλαγή μηνυμάτων. Πολλές από αυτές είναι κοινές και για τις δύο εφαρμογές. Για παράδειγμα, οι δύο εφαρμογές κάνουν χρήση της handleMessageWhenUp() που αφορά την λήψη ενός πακέτου από άλλο κόμβο.

Ωστόσο, επειδή η πρώτη εφαρμογή αφορά κυρίως την αποστολή δεδομένων και η δεύτερη την λήψη δεδομένων, υπάρχουν και διαφορετικές συναρτήσεις. Η UdpBasicApp χρησιμοποιεί την συνάρτηση sendPacket() που αφορά προφανώς την αποστολή πακέτων. Αντιθέτως η UdpSinkApp δεν κατέχει αυτήν την συνάρτηση αλλά κάνει χρήση της processPacket(), η οποία είναι αρμόδια για την λήψη πακέτων. Βέβαια, την συγκεκριμένη συνάρτηση την κατέχει και η πρώτη εφαρμογή διότι είναι ανάγκη η λήψη πακέτων ACK που επιβεβαιώνουν την ορθή αποστολή πακέτων από τον κόμβο 1 στον κόμβο 0. Επίσης, οι δύο εφαρμογές μπορεί να κάνουν χρήση μιας συνάρτησης ταυτόχρονα αλλά με διαφορετικό τρόπο, όπως η περίπτωση της setSocketOptions(). Και οι δύο κόμβοι χρησιμοποιούν την συνάρτηση για να αντιστοιχήσουν την θύρα επικοινωνίας τους με την ομάδα διευθύνσεων multicast. Όμως, ο κόμβος 1 που κάνει χρήση της UdpBasicApp, χρησιμοποιεί την setSocketOptions() για να καθορίσει τα headers των πακέτων που θα στείλει.

```
void UdpBasicApp::setSocketOptions()
{
    int timeToLive = par("timeToLive");
    if (timeToLive != -1)
        socket.setTimeToLive(timeToLive);

    int dscp = par("dscp");
    if (dscp != -1)
        socket.setDscp(dscp);

    int tos = par("tos");
    if (tos != -1)
        socket.setTos(tos);

    const char *multicastInterface = par("multicastInterface");
    if (multicastInterface[0]) {
        IInterfaceTable *ift = getModuleFromPar<IInterfaceTable>(par("interfaceTableModule"), this);
        InterfaceEntry *ie = ift->findInterfaceByName(multicastInterface);
        if (!ie)
            throw cRuntimeError("Wrong multicastInterface setting: no interface named \"%s\"", multicastInterface);
        socket.setMulticastOutputInterface(ie->getInterfaceId());
    }

    bool receiveBroadcast = par("receiveBroadcast");
    if (receiveBroadcast)
        socket.setBroadcast(true);

    bool joinLocalMulticastGroups = par("joinLocalMulticastGroups");
    if (joinLocalMulticastGroups) {
        MulticastGroupList mgl = getModuleFromPar<IInterfaceTable>(par("interfaceTableModule"), this)->collectMulticastGroups();
        socket.joinLocalMulticastGroups(mgl);
    }
    socket.setCallback(this);
}
```

Εικόνα 36 setSocketOptions-UdpBasicApp

```
void UdpSink::setSocketOptions()
{
    bool receiveBroadcast = par("receiveBroadcast");
    if (receiveBroadcast)
        socket.setBroadcast(true);

    MulticastGroupList mgl = getModuleFromPar<IInterfaceTable>(par("interfaceTableModule"), this)->collectMulticastGroups();
    socket.joinLocalMulticastGroups(mgl);

    // join multicastGroup
    const char *groupAddr = par("multicastGroup");
    multicastGroup = L3AddressResolver().resolve(groupAddr);
    if (!multicastGroup.isUnspecified()) {
        if (!multicastGroup.isMulticast())
            throw cRuntimeError("Wrong multicastGroup setting: not a multicast address: %s", groupAddr);
        socket.joinMulticastGroup(multicastGroup);
    }
    socket.setCallback(this);
}
```

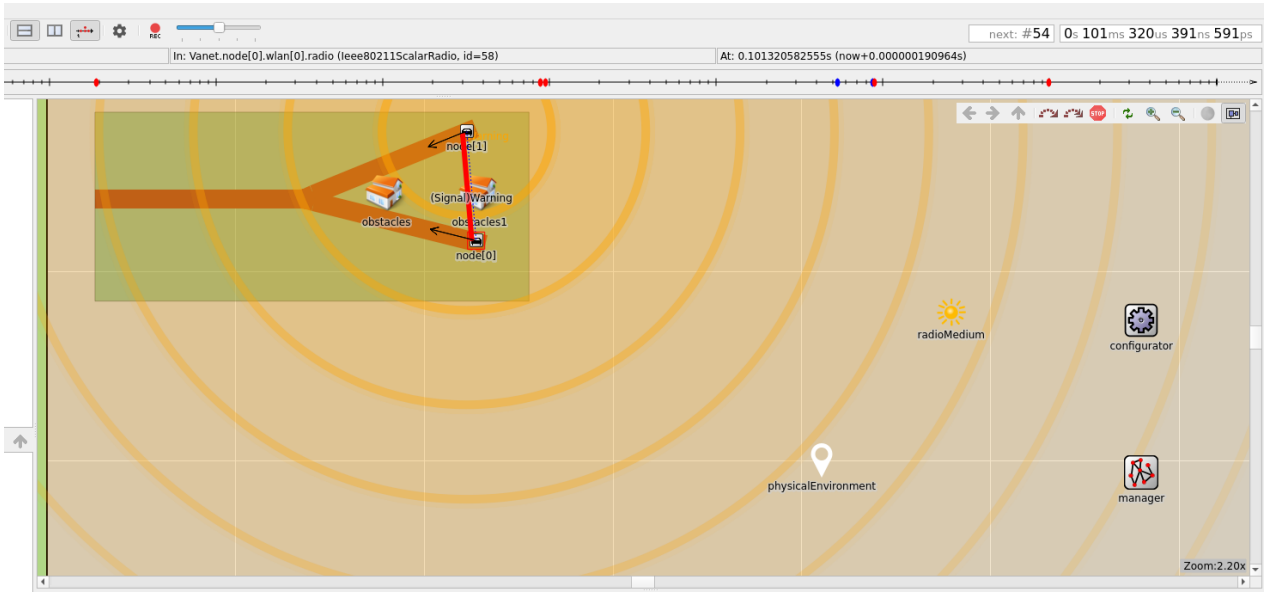
Εικόνα 37 setSocketOptions-UdpSinkApp

Όλο το περιεχόμενο των δύο προγραμμάτων βρίσκεται ελεύθερο για προβολή και επεξεργασία μέσα από το πρόγραμμα του omnet++ και το framework του inet που μπορεί να εγκατασταθεί παράλληλα. Στον φάκελο αρχείων του framework, αυτές οι εφαρμογές και πολλές άλλες βρίσκονται στο path: **inet -> src -> inet -> applications -> udpapp.**

Με την χρήση αυτών των εφαρμογών πρόκειται να υλοποιηθεί προσομοίωση που δύο οχήματα αποφεύγουν τυχόν σύγκρουση σε συμβολή δρόμων, στέλνοντας μηνύματα μεταξύ τους. Οι έτοιμες εφαρμογές του inet λειτουργούν ως templates που επάνω τους θα αφομοιωθούν τα αρχεία δικτύου. Αυτές οι εφαρμογές, όπως και πολλές άλλες, είναι ελεύθερες για επεξεργασία από το omnet++.

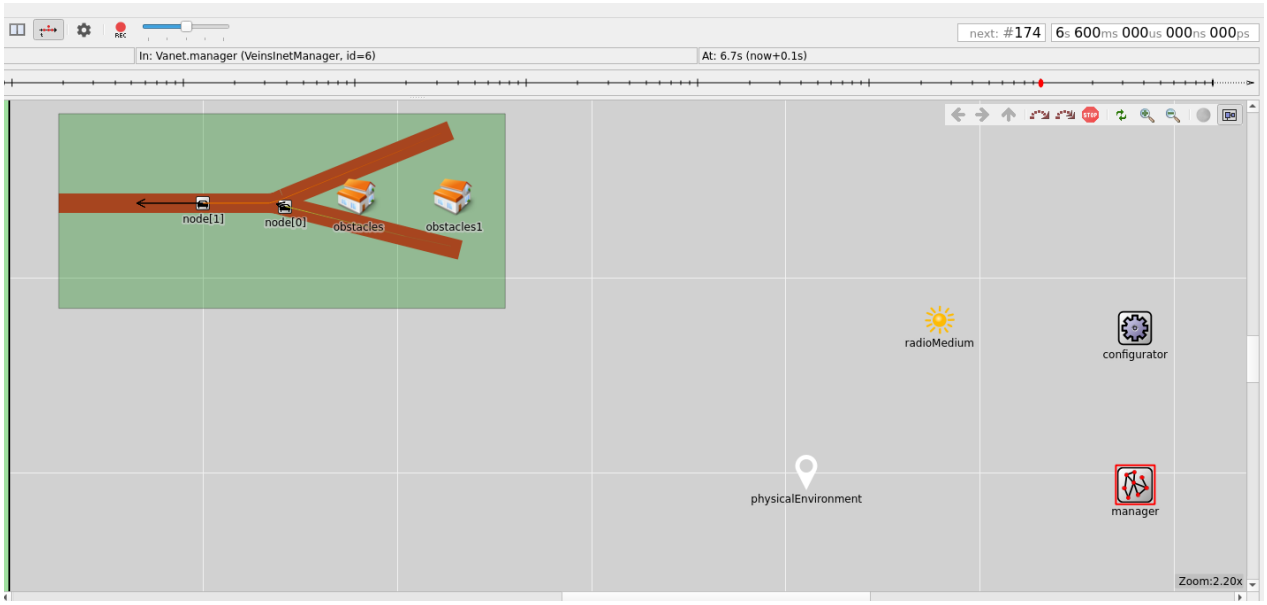
Κάνοντας build την πλέον έτοιμη εφαρμογή, και επιλύοντας τυχόν λάθη που μπορεί να εμφανιστούν το omnet++ μας εισάγει στο πρόγραμμα προσομοιώσεων του Qtenv, το οποίο είναι αντίστοιχο του Tkenv που αναφέρθηκε στην εργασία.

Όταν η προσομοίωση ξεκινά εμφανίζονται οι δύο κόμβοι που θεωρητικά κινούνται στους αντίστοιχους δρόμους. Στα επόμενα κλάσματα δευτερολέπτων που εκτυλίσσονται στην προσομοίωση, οι κόμβοι ανταλλάζουν τα απαραίτητα πακέτα μηνυμάτων προκειμένου οι οδηγοί να είναι ενημερωμένοι. Παρατηρείται ότι η ανταλλαγή μηνυμάτων συμβαίνει ενώ θεωρητικά τα οχήματα φαίνονται να είναι στάσιμα, αλλά επισημαίνεται πως υποθετικά η ανταλλαγή μηνυμάτων συμβαίνει μέσα σε κάποια κλάσματα δευτερολέπτων, το οποίο μπορεί να ισχύει και σε πραγματικές συνθήκες.



Εικόνα 38 Αποστολή μηνύματος

Εφόσον η ανταλλαγή μηνυμάτων ολοκληρωθεί, φαίνονται οι κόμβοι να θέτονται σε κίνηση και να παρατηρείται πως ο κόμβος 0 θα σταματήσει στην συμβολή των δρόμων για να περάσει ο κόμβος 1 που έχει προτεραιότητα, εφόσον πλέον οι οδηγοί είναι εξίσου ενημερωμένοι. Αυτό το σενάριο προσομοίωσης είναι ένα ακόμα σημαντικό και απλό παράδειγμα που αποδεικνύει πως μια τέτοια τεχνολογία θα μπορούσε να φανεί πολύ χρήσιμη σε πραγματικές συνθήκες.



Εικόνα 39 Αποτέλεσμα εγχειρήματος

Συγκριτικά με την πρώτη υλοποίηση, σε ένα σενάριο σαν και αυτό δεν θα μπορούσε να εφαρμοστεί κάποιος αλγόριθμος ομαδοποίησης. Αυτό το πρόγραμμα έχει σκοπό να δείξει πως μπορεί να λειτουργήσει η τεχνολογία των vanet σε μια προσομοίωση που παράγει πραγματικά αποτελέσματα και να αποδείξει πως τα αποτελέσματα αυτά, θα μπορούσαν να συμβάλουν στην ενίσχυση της ασφάλειας για την οδήγηση. Επίσης, είναι ανάγκη να υπάρχει υλικό που δεν πατάει μόνο σε θεωρητικό πλαίσιο που αφορά το αντικείμενο μελέτης, αλλά παραδείγματα που αποκλίνουν από την θεωρία και οδεύουν στα αποτελέσματα.

Συμπεράσματα

Η εργασία αυτή παράγει ως αποτέλεσμα μόνο ένα ελάχιστο τμήμα δεδομένων μπροστά στον πραγματικό όγκο πληροφορίας που αφορά το θέμα των ασύρματων επικοινωνιών μεταξύ οχημάτων σε οδικά δίκτυα. Σαφώς, ο κάθε ενδιαφερόμενος θα συναντήσει ένα αμέτρητο πλήθος πληροφορίας καθώς ερευνά όλο και βαθύτερα για το συγκεκριμένο θέμα, και κάθε φορά το αποτέλεσμα μπορεί να είναι πλήρως διαφορετικό. Θα ήθελα να σημειωθεί πως το παρών έγγραφο δεν αποτελεί αντιπροσωπευτική οδηγία για πραγματική εφαρμογή στο ζήτημα των επικοινωνιών οχημάτων, παρά μόνο μια αρχάρια ερευνητική προσέγγιση καθαρού ενδιαφέροντος.

Η εκπόνηση της εργασίας ήταν δύσκολη, έχοντας ως κύριο εμπόδιο την συλλογή δεδομένων και πρακτικών που μέχρι πριν δεν κατείχα σε ικανοποιητικό βαθμό. Η σύνταξη ενός ολοκληρωμένου κειμένου που θα περιέχει όλες τι βασικές, κατά την δικιά μου οπτική και προσέγγιση, πληροφορίες για τις επικοινωνίες οχημάτων και τα VANETS απαιτούσε προσοχή για τυχόν θεμελιώδη λάθη και αμέτρητο σεβασμό προς όλους τους ερευνητές που αναφέρονται ως πηγές στην εργασία, για την δουλειά και την προσβάσιμη πληροφορία που παρέδωσαν ελεύθερα στο διαδίκτυο. Η υλοποίηση του δικτύου σε προσομοίωση ήταν μια διαδικασία με πολλά εμπόδια και αντικείμενο πολύωρης έρευνας. Η τελική μορφή του προγράμματος που δόθηκε στο παρών έγγραφο είναι η καλύτερη δυνατή προσέγγιση.

Ως ενδεχόμενη προέκταση αλλά και διόρθωση ή/και συμπλήρωση της εργασίας, θα μπορούσαν να συμβούν πολλά, ξεκινώντας από την προσθήκη νέου και καλύτερου κώδικα για την επέκταση του προγράμματος που αναρτήθηκε ως παράδειγμα εφαρμογής. Τα σενάρια που μπορούν να υλοποιηθούν έχοντας κατά νου όσα ειπώθηκαν στην εργασία, είναι αμέτρητα και στηρίζονται καθαρά στην ευρηματικότητα, την δημιουργικότητα και προφανώς στις γνώσεις και πρακτικές που κατέχει ο εκάστοτε ερευνητής.

Βιβλιογραφία

[1] "Ad hoc δίκτυο," Wikipedia, 16 Δεκέμβριος 2022. [Online]. Available: https://el.wikipedia.org/wiki/Ad_hoc_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF.

[2] "Ad hoc," Wikipedia, 15 Οκτώβριος 2023. [Online]. Available: https://en.wikipedia.org/wiki/Ad_hoc.

[3] "Wireless ad hoc network," Wikipedia, 26 Αύγουστος 2023. [Online]. Available: https://en.wikipedia.org/wiki/Wireless_ad_hoc_network.

[4] M. Uy, "What is an Ad Hoc Wireless Network?," Liferwire tech for Humans, 22 Οκτώβριος 2021. [Online]. Available: https://www.lifewire.com/what-is-an-ad-hoc-wireless-network-2377409.

[5] A. R. Rajeswari, "A Mobile Ad Hoc Network Routing Protocols: A Comparative Study," IntechOpen, 02 Ιούλιος 2020. [Online]. Available: https://www.intechopen.com/chapters/72470.

[6] D. S. S. B. Mr L. Raja, "An Overview of MANET: Applications, Attacks and Challenges," *International Journalof Computer Science and Mobile Computing,* vol. 3, no. 1, pp. pg. 408-417, 2014.

[7] Z. J. H. Lidong Zhou, "Securing Ad Hoc Networks," IEEE Network, Ithaca, New york, 1999.

[8] B. S. M. C. Siva Ram Murthy, "Transport Layer and Security Protocols for Ad Hoc Wireless Networks," in *Ad Hoc Wireless Networks: Architectures and Protocols*, Pearson Education, Informit, 2005.

[9] C. Sommer, "Veins framework-Documentation," 18 Δεκέμβριος 2020. [Online]. Available: https://veins.car2x.org/documentation/.

[10] C. Sommer, "Veins: The Open Source Vehicular Network Simulation Framework," in *Recent Advances in Network Simulation*, Dresden, Springer, Cham, 2019, pp. 215-252.

[11] "IEEE 802.11e-2005," Wikipedia, 7 Αύγουστος 2022. [Online]. Available: https://en.wikipedia.org/wiki/IEEE_802.11e-2005.

[12] A. o. R. I. a. Businesses, "700 MHz BAND INTELLIGENT TRANSPORT SYSTEMS ARIB STANDARD," Association of Radio Industries and Businesses, Tokyo, Japan, 2012.

[13] F. D. Christoph Sommer, "Using The Right Two-Ray Model? A Measurement-based Evaluation of PHY Models in VANETs," Institute of Computer Science, University of Innsbruck, Innsbruck, 2011.

[14] S. M. F. S. Daniel Maier, "Deterministic Models of the Physical Layer through Signal Simulation," Institute of Embedded Systems/Real-Time Systems Faculty of Engineering and Computer Science and Psychology, Ulm University, Ulm, 2015.

[15] M. S. K. M. A. Anirudh Paranjothi, "Hybrid-Vehcloud: An Obstacle Shadowing Approach for VANETs in Urban Enviroment," School of Computer Science, University of Oklahoma και Department of Computing, East Tennessee State University, Norman, Oklahoma και Johnson City, Tennessee, USA.

[16] D. E. R. G. F. D. Christoph Sommer, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Enviroments," Dept. of Computer Science, University of Erlangen, Erlangen, 2011.

[17] A. B. C. S. David Eckhoff, "On the Impact of Antenna Patterns on VANET Simulation," Dept. of Computer Science, University of Erlangen και Dept. of Computer Science, Paderborn University, Paderbon και Erlangen, 2016.

[18] O. Team, "Introduction to INET Framework," OMNET++ Team, [Online]. Available: https://inet.omnetpp.org/docs/developers-guide/ch-introduction.html#what-is-inet-framework.

[19] O. Team, "Working with Packets," OMNET++ Team, [Online]. Available: https://inet.omnetpp.org/docs/developers-guide/ch-packets.html#overview.

[20] O. Team, "Using Sockets," OMNET++ Team, [Online]. Available: https://inet.omnetpp.org/docs/developers-guide/ch-sockets.html#l3-socket.

[21] O. Team, "OMNET++ Simulation Manual," OMNET++ Team.

[22] A. Varga, "The OMNET++ Discrete Event Simulation System," Dept. of Telecommunications, Budapest University of Technology and Economics, Budapest, 2001.

[23] R. H. András Varga, "An Overview of The OMNET++ Simulation Enviroment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools 2008*, Marseille, 2008.

[24] L. B.-W. J. E. D. K. Michael Behrisch, "SUMO-Simulation of Urban MObility: An Overview," in *SIMUL 2011: The Third International Conference on Advances in System Simulation*, Berlin, 2011.

[25] G. A. C. (DLR), "SUMO Documentation," German Aerospace Center (DLR), 08 Ιούνιος 2023. [Online]. Available: https://sumo.dlr.de/docs/index.html.

[26] G. A. C. (DLR), "SUMO-Installing," German Aerospace Center (DLR), 10 Οκτώβριος 2023. [Online]. Available: https://sumo.dlr.de/docs/Installing/index.html.

[27] S. M. M. Zaydoun Yahya Rawashdeh, "Communications in Vehicular Networks," in *Mobile Ad-Hoc Networks Applications*, Rijeka, IN TECH d.o.o, 2019, pp. 19-40.

[28] T. T. V. V. Mate Boban, "Modeling and Simulation of Vehicular Networks: Towards Realistic and Efficient Models," in *Mobile Ad-Hoc Networks Applications*, Rijeka, IN TECH d.o.o, 2019, pp. 41-66.

[29] M. B. a. R. C. Anna Maria Vegni, "Smart Vehicles, Technologies and Main Applications in Vehicular Ad hoc Networks," in *Vehicular Technologies - Deployment and Applications*, IntechOpen, 2013.

[30] C. Sommer, "Veins framework-Documentation/Modules," 13 Δεκέμβριος 2022. [Online]. Available: https://veins.car2x.org/documentation/modules/.

[31] R. A. S. L. A. V. G. V. R. L. A. E. B. Gabriel Alejandro Galaviz Mosqueda, "Survey on Multi-hop Vehicular Ad Hoc Networks under IEEE 802.16 Technology," in *Mobile Ad-Hoc Networks Applications*, Rijeka, IN TECH d.o.o., 2019, pp. 3-16.

[32] M. S. C. P. D. M. P. Ms. G. Nathiya, "An Analytical Study on Behavior of Clusters Using K Means, EM and K*Means Algorithm," *International Journal of Computer Science and Information Security,* vol. 7, no. 3, pp. 185-190, 2010.