ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΠΡΟΗΓΜΕΝΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΔΙΚΤΥΑ»

ΕΠΕΚΤΑΣΗ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΠΡΟΣΟΜΟΙΩΣΗΣ NS-3 ΓΙΑ ΤΗΝ ΥΠΟΣΤΗΡΙΞΗ ΜΕΤΑΠΟΜΠΩΝ ΣΕ ΔΙΚΤΥΑ LTE

Κωνσταντίνος Ν. Κουτράκης

Επιβλέποντες Καθηγητές: Καλόξυλος Αλέξανδρος, Τσούλος Γεώργιος

Οκτώβριος 2014

Στη σύζυγό μου Μαρία και στο γιο μου Νίκο
για την υποστήριξη και την κατανόηση που έδειξαν
κατά τη διάρκεια εκπόνησης της παρούσας εργασίας

Στους γονείς μου Νίκο και Φωτεινή
για την αμέριστη συμπαράστασή τους
σε κάθε βήμα της ζωής μου

Ευχαριστίες

Αρχικά θέλω να ευχαριστήσω όλους τους καθηγητές μου για τις γνώσεις που μου μετέφεραν κατά τη διάρκεια της φοίτησής μου στο Μεταπτυχιακό Πρόγραμμα Σπουδών του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Πελοποννήσου με τίτλο «Προηγμένα Τηλεπικοινωνιακά Συστήματα & Δίκτυα».

Ιδιαίτερα ευχαριστώ τον Επίκουρο Καθηγητή κύριο Καλόξυλο Αλέξανδρο για την πολύτιμη βοήθεια και την επιστημονική καθοδήγηση που μου προσέφερε. Επίσης η συνεισφορά του Αναπληρωτή Καθηγητή κυρίου Τσούλου Γεωργίου ήταν κρίσιμη ώστε να ολοκληρωθεί η διαδικασία εκπόνησης της παρούσας εργασίας και τον ευχαριστώ θερμά για αυτό.

Ακόμα ευχαριστώ τον υποψήφιο Διδάκτορα του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών κύριο Μπαρμπουνάκη Σωκράτη για την εξαιρετική συνεργασία και την καθοριστική βοήθειά του στην υλοποίηση του κώδικα προσομοίωσης που περιέχεται στην εργασία μου.

Επέκταση του περιβάλλοντος προσομοίωσης ns-3 για την υποστήριξη μεταπομπών σε δίκτυα LTE

Σημαντικοί όροι: LTE, SAE, 4G, διεπαφές, σταθμός βάσης, μεταπομπή, ρυθμαπόδοση, καθυστέρηση, λογισμικό προσομοίωσης κυψελωτών δικτύων διακριτών γεγονότων, ns-3, επέκταση.

Extension of ns-3 simulation environment in order to support handovers in LTE networks

Keywords: LTE, SAE, 4G, interfaces, base station, handover, throughput, delay, discrete event simulation software of cellular networks, ns-3, extension.

Περιεχόμενα

Περίληψη

Τα συστήματα κινητών επικοινωνιών είναι μια επιστημονική περιοχή με έντονο ερευνητικό ενδιαφέρον. Υπάρχει μια πληθώρα προσομοιωτών που χρησιμοποιούνται για την πειραματική επαλήθευση προτεινόμενων εργασιών. Ο πιο διαδεδομένος είναι ο προσομοιωτής διακριτών γεγονότων ns-3. Πρόκειται για λογισμικό ανοιχτού κώδικα που υποστηρίζεται από ένα μεγάλο αριθμό προγραμματιστών παγκοσμίως. Μια από τις χρήσεις του είναι και η προσομοίωση δικτύων LTE. Αν και περιέχει ένα πολύ μεγάλο σύνολο λειτουργιών, μια βασική του έλλειψη είναι η απουσία υποστήριξης μεταπομπών ανάμεσα σε σταθμούς βάσης διαφορετικών συχνοτήτων. Οι αλγόριθμοι μεταπομπών που υποστηρίζει λειτουργούν μόνο όταν οι σταθμοί βάσης λειτουργούν στην ίδια συχνότητα.

Η συνεισφορά της παρούσας εργασίας έγκειται στην επέκταση του προσομοιωτή ns-3 ώστε να υποστηρίζει μεταπομπές κινητών τερματικών ανάμεσα σε σταθμούς βάσης που λειτουργούν σε διαφορετικές συχνότητες. Η λύση που προτείνουμε βασίζεται στη χρήση ενός δεύτερου κινητού τερματικού που λειτουργεί ως η σκιά του πρώτου (shadow terminal), λαμβάνοντας μετρήσεις από τους σταθμούς βάσης που δεν μπορεί να “ακούσει” το πρώτο κινητό τερματικό.

Abstract

Mobile communication systems is an area of intense scientific research interest. There is a plethora of simulators used for the experimental verification of proposed papers. The most widely used is the discrete event simulator ns-3. It is an open source software that is supported by a large number of developers worldwide. One of its uses is the simulation of LTE networks. Although it contains a large set of functions, a basic shortcoming is the lack of support of handovers between base stations which are operating in different frequencies. Handover algorithms are supported only when the base stations operate in the same frequency.

The contribution of this work is to expand the simulator ns-3 in order to support handovers of mobile terminals between base stations which are operating in different frequencies. The solution that we propose is based on the use of a second mobile terminal (shadow terminal) that operates as the shadow of the first one, which is taking measurements from the base stations that the first terminal cannot “listen”.

Εισαγωγή

Η σημερινή εποχή χαρακτηρίζεται, όχι υπερβολικά, ως «εποχή της πληροφορίας». Ο χρήστης έχει στην κατοχή του ένα μεγάλο πλήθος ετερογενών τεχνολογιών που του προσφέρουν την πολυπόθητη διασύνδεση με άλλους χρήστες. Στις ασύρματες επικοινωνίες το βασικό εργαλείο που προσφέρεται είναι το κινητό τηλέφωνο το οποίο έχει εξελιχθεί σε ένα σημείο αναφοράς που εξυπηρετεί ένα μεγάλο σύνολο υπηρεσιών. Τα σημερινά έξυπνα τηλέφωνα μπορούν να χρησιμοποιηθούν για την πραγματοποίηση κλήσεων (φωνητικών και βιντεοκλήσεων), για την πρόσβαση στο διαδίκτυο, για ψυχαγωγία (μουσική, βίντεο, παιχνίδια), για ανταλλαγή δεδομένων με κοντινούς χρήστες (Bluetooth, WiFi-Direct) κοκ.

Ο σημαντικότερος στόχος για τις κινητές επικοινωνίες είναι η βελτίωση της διαθέσιμης χωρητικότητας των δικτύων ώστε να εξυπηρετούνται οι διαρκώς αυξανόμενες απαιτήσεις των χρηστών για μεταφορά δεδομένων. Σήμερα γίνεται λόγος για τις κινητές επικοινωνίες $5^{ης}$ γενιάς (5G). Στην ουσία πρόκειται για την προσπάθεια συνδυασμού ετερογενών τεχνολογιών (GPRS, EDGE, HSPA, LTE, WiFi κλπ) ώστε τα κινητά να χρησιμοποιούν την αποδοτικότερη λύση για την επίτευξη του στόχου που προαναφέρθηκε.

Προκειμένου να χρησιμοποιηθεί μια νέα τεχνολογία σε εμπορικά συστήματα θα πρέπει να δοκιμαστεί πρώτα σε περιβάλλοντα προσομοίωσης ώστε να διαπιστωθούν και να διορθωθούν τα όποια προβλήματα ανακύψουν. Το πιο διαδεδομένο περιβάλλον προσομοίωσης είναι ο ns-3. Πρόκειται για έναν προσομοιωτή διακριτών γεγονότων που έχει γίνει αποδεκτός από την ακαδημαϊκή και ερευνητική κοινότητα. Η τελευταία σταθερή έκδοση, κατά τη διάρκεια εκπόνησης της παρούσας εργασίας, είναι η 3.19. Για τα δίκτυα $4^{ης}$ γενιάς χρησιμοποιείται η μονάδα προσομοίωσης (simulation module) LTE, η οποία είναι το αποτέλεσμα της δουλειάς του LENA project.

Το LTE module (LENA) είναι βασισμένο στην $8^{η}$ έκδοση της 3GPP (Realease 8). Αυτό έχει ως συνέπεια να μην υποστηρίζεται από τις τερματικές συσκευές (κινητά) η λήψη μετρήσεων από σταθμούς βάσης διαφορετικών συχνοτήτων. Το αποτέλεσμα είναι η αδυναμία των κινητών τερματικών να εκτελέσουν μεταπομπές (handover) ανάμεσα σε σταθμούς βάσης που εκπέμπουν σε διαφορετικές συχνότητες.

Η συνεισφορά αυτής της εργασίας είναι η επέκταση του προσομοιωτή LENA ώστε να υποστηρίζονται οι μεταπομπές των κινητών ανάμεσα σε σταθμούς βάσης διαφορετικών συχνοτήτων (π.χ. macrocell και picocell/femtocell). Η λύση προκύπτει από τη χρήση ενός δεύτερου κινητού τερματικού (κινητό σκιά, shadow terminal) και τη δημιουργία ενός μηχανισμού σύγκρισης των τιμών RSRQ σε πραγματικό χρόνο. Όταν η τιμή RSRQ του πρώτου κινητού γίνει μικρότερη από την αντίστοιχη του κινητού σκιά (το οποίο εξυπηρετείται από σταθμό βάσης διαφορετικής συχνότητας από το πρώτο) δίνεται η εντολή μεταπομπών ώστε να ανταλλάξουν σταθμούς βάσης μεταξύ τους.

Η παρούσα εργασία αποτελείται από έξι κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζεται το σύστημα LTE/SAE. Στο δεύτερο γίνεται αναφορά στο περιβάλλον προσομοίωσης ns-3 και στο τρίτο αναλύεται η λειτουργία του συστήματος LENA και η υποστήριξη μεταπομπών. Ακολουθεί το τέταρτο κεφάλαιο που περιέχει τις αλλαγές που πραγματοποιήθηκαν ώστε το LENA να υποστηρίξει τις μεταπομπές ανάμεσα σε σταθμούς βάσης διαφορετικών συχνοτήτων και το πέμπτο με τα αποτελέσματα των σεναρίων προσομοίωσης που χρησιμοποιήθηκαν. Τέλος στο έκτο κεφάλαιο βρίσκονται τα συμπεράσματα που εξήχθησαν. Η εργασία ολοκληρώνεται με την παρουσίαση του κώδικα που αναπτύχθηκε ώστε να γίνει η πειραματική επαλήθευση της προτεινόμενης λύσης για την επέκταση του LENA.

Κεφάλαιο 1^{ο}: Το σύστημα LTE/SAE

1.1. Ιστορική αναδρομή των κινητών επικοινωνιών

1.1.1. Η πρώτη γενιά (1G)

Η πρώτη γενιά των κινητών επικοινωνιών παρουσιάστηκε στις αρχές της δεκαετίας του 1980. Αυτά τα συστήματα χρησιμοποιούσαν αναλογικές τεχνικές επικοινωνιών, παρόμοιες με αυτές του αναλογικού ραδιοφώνου. Οι κυψέλες ήταν αρκετά μεγάλες και τα συστήματα δε χρησιμοποιούσαν αποδοτικά το διαθέσιμο φάσμα. Αυτό είχε ως συνέπεια τη μικρή χωρητικότητα σε σχέση με τα σημερινά δεδομένα. Οι κινητές συσκευές των χρηστών ήταν αρκετά μεγάλες και ακριβές. Αυτός ήταν και ο βασικός λόγος που αυτή η τεχνολογία δεν μπορούσε να προωθηθεί για ευρεία εμπορική χρήση.

Πριν από τη λεγόμενη πρώτη γενιά (1G) κινητών επικοινωνιών υπήρχαν και άλλες τεχνολογίες. Η βασικής τους διαφορά βρισκόταν στη λειτουργία των κυψελών. Οι κινητές επικοινωνίες που προηγήθηκαν από το 1G επικεντρώνονταν σε πολύ ισχυρούς σταθμούς βάσης που είχαν μια περιοχή κάλυψης περίπου 80 χιλιομέτρων. Τα διαθέσιμα συχνοτικά κανάλια ήταν περιορισμένα, οπότε σε αρκετές περιπτώσεις οι χρήστες ήταν υποχρεωμένοι να περιμένουν αρκετά μέχρις ότου αποκτήσουν ένα. Για παράδειγμα στην πόλη της Νέας Υόρκης του 1976 υπήρχαν 543 χρήστες οι οποίοι εξυπηρετούνταν μόνο από 12 συχνοτικά κανάλια.

Μέχρι τις αρχές της δεκαετίας του 1980 δεν υπήρχαν πρότυπα για τις κινητές επικοινωνίες. Το τοπίο άλλαξε όταν αρχικά οι σκανδιναβικές χώρες προχώρησαν στη δημιουργία του προτύπου Nordic Mobile Telephone (NMT). Οι πρώτες αγορές που υποδέχθηκαν τις κινητές επικοινωνίες ήταν της Σουηδίας και της Ολλανδίας το 1981. Το 1982 προστέθηκε η Νορβηγία και η Φινλανδία. Οι ΗΠΑ ακολούθησαν το 1983.

1.1.2. Η δεύτερη γενιά (2G)

Η δεύτερη γενιά των κινητών επικοινωνιών ξεκίνησε στις αρχές της δεκαετίας του 1990. Η βασική διαφορά από την πρώτη γενιά ήταν η χρήση ψηφιακής τεχνολογίας, η οποία οδήγησε τόσο στη βελτίωση της χρήσης του διαθέσιμου φάσματος όσο και στη δημιουργία μικρότερων και οικονομικότερων κινητών συσκευών. Εκτός από τη φωνητική κλήση το 2G υποστηρίζει και την αποστολή και λήψη μηνυμάτων κειμένου (Short Message Service, SMS).

Το πιο γνωστό σύστημα δεύτερης γενιάς είναι το Global System for Mobile Communications (GSM) το οποίο αρχικά σχεδιάστηκε για την Ευρώπη. Αργότερα χρησιμοποιήθηκε σε όλο τον κόσμο. Ένα άλλο σύστημα που σχεδιάστηκε παράλληλα από την Qualcomm για την αγορά των ΗΠΑ είναι το cdmaOne.

Η περίοδος χρήσης της δεύτερης γενιάς κινητών επικοινωνιών συνέπεσε με την εξάπλωση της χρήσης του Internet. Αυτό είχε ως συνέπεια να προστεθεί στις υπάρχουσες υπηρεσίες του 2G (φωνή και γραπτά μηνύματα) και η δυνατότητα επεξεργασίας δεδομένων. Αυτό επετεύχθη με την προσθήκη του packet-switched domain στο ήδη υπάρχον circuit-switched domain. Το πρώτο βήμα έγινε με το General Packet Radio Service (GPRS) και χαρακτηρίστηκε ως γενιά 2.5 (2.5G). Ακολούθησε η γενιά 2.75 (2.75G), με το όνομα Enhanced Data rates for GSM Evolution (EDGE) [1]. Το πρότυπο αυτό δημιουργήθηκε από τη 3GPP ως μέλος της οικογένειας του GSM.

1.1.3. Η τρίτη γενιά (3G)

Η τρίτη γενιά κινητών επικοινωνιών παρέχει υπηρεσίες φωνητικών κλήσεων, γραπτών μηνυμάτων, μεταφοράς δεδομένων, βιντεοκλήσεις και τηλεόρασης (mobile TV). Υπάρχουν τρεις διαφορετικές τεχνολογίες τρίτης γενιάς.

Η πρώτη και κυρίαρχη τεχνολογία ονομάζεται Universal Mobile Telecommunications System (UMTS). Σχεδιάστηκε με βάση το GSM. Το core network διατηρήθηκε σχεδόν αυτούσιο. Το air interface όμως επανασχεδιάστηκε από την αρχή και διαθέτει δύο υλοποιήσεις. Η πρώτη ονομάζεται Wideband Code Division Multiple Access (WCDMA) και χρησιμοποιείται σχεδόν σε ολόκληρο τον κόσμο. Η δεύτερη ονομάζεται Time Division Synchronous Code Division Multiple Access (TD-

SCDMA). Το TD-SCDMA σχεδιάστηκε στην Κίνα ώστε να ελαττωθεί η εξάρτηση της χώρας από δυτικές τεχνολογίες και να μειωθεί το κόστος από τα δικαιώματα αυτών των τεχνολογιών [2].

Η δεύτερη τεχνολογία ονομάζεται cdma2000 και χρησιμοποιήθηκε στη βόρεια Αμερική.

Η τρίτη είναι η Worldwide Interoperability for Microwave Access (WiMAX) η οποία αναπτύχθηκε από την Institute of Electrical and Electronics Engineers (IEEE) σύμφωνα με το πρότυπο 802.16.

1.2. Η μετάβαση από την τρίτη στην τέταρτη γενιά επικοινωνιών (4G)

Οι κινητές επικοινωνίες αρχικά σχεδιάστηκαν για να καλύψουν τις ανάγκες των χρηστών όσον αφορά σε τηλεφωνικές συνδέσεις. Για ένα μεγάλο χρονικό διάστημα αυτή ήταν η κυρίαρχη μορφή χρήσης αυτών των συστημάτων. Από το τέλος της δεκαετίας του 1990, όπου η χρήση του internet μπήκε για τα καλά στην καθημερινή ζωή των ανθρώπων, η ανάγκη για κινητά δεδομένα (mobile data) αυξανόταν διαρκώς. Στην αρχή ο ρυθμός δεν ήταν τόσο γρήγορος. Από το τέλος της δεκαετίας του 2000 ξεκινά μια ραγδαία αύξηση της ζήτησης για κινητά δεδομένα.

Αυτή η ραγδαία αύξηση της κίνησης στα δίκτυα δεύτερης και τρίτης γενιάς οδήγησε, στο τέλος της δεκαετίας του 2000, στη συμφόρησή τους. Αυτό είχε ως συνέπεια να αναζητηθούν τρόποι αύξησης της χωρητικότητας των δικτύων. Όσον αφορά στα δίκτυα τρίτης γενιάς χρησιμοποιήθηκαν τα HSPA και HSPA+. Παρά τη βελτίωση που επέφεραν στα συστήματα 3G οι προβλέψεις για το μέλλον έκαναν επιτακτική την ανάγκη για αλλαγές εκ βάθρων.

Το 2004 η 3GPP (3rd Generation Partnership Project) ξεκίνησε τη μελέτη πάνω σε μια εξέλιξη του UMTS σε βάθος χρόνου (Long Term Evolution, LTE). Ο στόχος ήταν τα συστήματα που βασίζονται σε πρότυπα της 3GPP να παρέχουν υψηλές ταχύτητες δεδομένων και χαμηλές καθυστερήσεις σύμφωνα με τις απαιτήσεις των χρηστών.

Αν και το LTE θεωρείται η μετεξέλιξη του UMTS στην πραγματικότητα δεν έχουν και πολλά κοινά μεταξύ τους [3]. Το UMTS RAN (Radio Access Network) έχει δύο βασικά συστατικά: το Universal Terrestrial Radio Access (UTRA) το οποίο είναι το air interface που περιέχει το κινητό τερματικό (UE) και το Universal Terrestrial Radio

Access Network (UTRAN) που περιέχει το Radio Network Controller (RNC) και το σταθμό βάσης ή node B (NB).

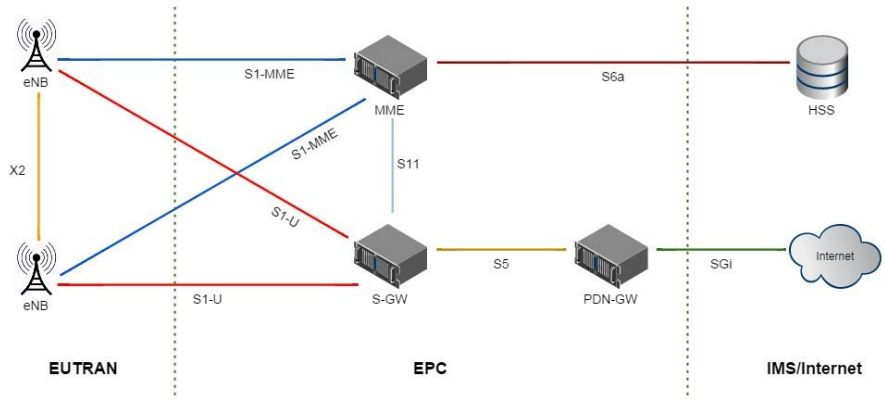
Ο όρος LTE, αν και αναφέρεται στο air interface, χρησιμοποιείται για να περιγράψει όλο το νέο σύστημα. Αν και δεν είναι απόλυτα ορθό έχει καθιερωθεί. Το νέο σύστημα ονομάζεται Evolved Packet System (EPS) και αποτελείται από το Evolved Universal Terrestrial Radio Access Network (EUTRAN) και το Evolved Packet Core (EPC).

1.3. Η αρχιτεκτονική LTE/SAE

Συνολικά για τα δίκτυα τέταρτης γενιάς έχει επικρατήσει ο όρος Evolved Packet System (EPS). Αποτελούνται από δύο συστήματα. Το core network και το radio access network. Το core network, στην αρχή της δημιουργίας του, ονομαζόταν System Architecture Evolution (SAE) αλλά στη συνέχεια επικράτησε ο όρος Evolved Packet Core (EPC). Αυτό σημαίνει πως αν και ο όρος SAE δεν είναι αυτός που επικράτησε μπορεί να χρησιμοποιείται ως ταυτόσημος με τον όρο EPC. Παρομοίως, αναφορικά με το air interface, οι όροι LTE και EUTRAN (Evolved Universal Terrestrial Radio Access Network) μπορούν να θεωρούνται ταυτόσημοι. Συνολικά μπορούμε να θεωρήσουμε πως οι παρακάτω όροι αντιπροσωπεύουν το νέο σύστημα τέταρτης γενιάς (4G):

- EPS (συνολικά το σύστημα)
- LTE / SAE (air interface / core network)
- EUTRAN / EPC (air interface / core network)

Στο επόμενο σχήμα παρουσιάζεται μια απεικόνιση υψηλού επιπέδου για το σύστημα LTE/SAE.



Σχήμα 1.1: Παρουσίαση του LTE/SAE

1.4. Τα μέρη του συστήματος LTE/SAE

1.4.1. Evolved Universal Terrestrial Radio Access Network (EUTRAN)

1.4.1.1. Σταθμός Βάσης, Evolved Node B (eNB)

Είναι το μοναδικό στοιχείο του συστήματος στο air interface. Είναι υπεύθυνο για τη σύνδεση της κινητής συσκευής κάθε χρήστη με το σύστημα [7]. Ελέγχει τις ραδιοεκπομπές από και προς τα UE. Εξυπηρετεί πολλούς ρόλους όπως έλεγχο διαχείρισης και συνδέσεων, προγραμματισμό δεδομένων χρηστών, έλεγχο σηματοδοσίας κλπ. Η βασική διαφορά του EUTRAN από το UTRAN (3G) είναι πως στο νέο σύστημα το eNB παίζει το ρόλο του σταθμού βάσης και του Radio Network Controller (RNC). Αυτή η απλούστευση του σχεδιασμού του air interface μεταφέροντας το έλεγχο πολλών λειτουργιών στο eNB έχει ως αποτέλεσμα τη μείωση των καθυστερήσεων λόγω μικρότερης ανάγκης για σηματοδοσία.

1.4.1.2. Home eNB (HeNB) και Home eNB Gateway (HeNB-GW)

Εκτός του eNB μπορούν να χρησιμοποιηθούν δύο ακόμα οντότητες με τα ονόματα home eNB (HeNB) και home eNB Gateway (HeNB GW) [7].

Τα HeNB έχουν τα ακόλουθα χαρακτηριστικά:

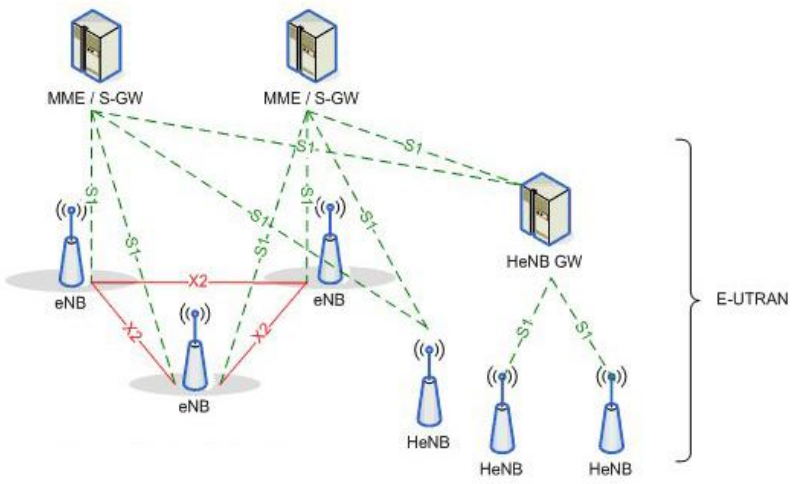
- Τοποθετούνται στο χώρο των χρηστών και χρησιμοποιούν το διαθέσιμο φάσμα του παρόχου
- Ενισχύουν την κάλυψη και τη χωρητικότητα του δικτύου
- Εμπεριέχουν όλες τις λειτουργίες του eNB με τις αντίστοιχες παραμετροποιήσεις

Τα HeNB-GW που σχετίζονται με τα HeNB λύνουν το πρόβλημα των πολλαπλών διεπαφών S1 που θα χρειάζονταν. Οι τρόποι χρήσης των HeNB GW χωρίζονται σε τρεις κατηγορίες:

- Κλειστή πρόσβαση. Σε αυτή την κατηγορία μόνο μια περιορισμένη ομάδα χρηστών μπορεί να χρησιμοποιήσει στο HeNB
- Ανοιχτή πρόσβαση. Το HeNB είναι προσβάσιμο από όλες τις κινητές συσκευές σα να ήταν ένας σταθμός βάσης eNB

- Υβριδική πρόσβαση. Και πάλι όλες οι κινητές συσκευές μπορούν να συνδεθούν με το HeNB, αλλά σε περίπτωση συμφόρησης τα μέλη της ομάδας χρηστών έχουν προτεραιότητα

Στην επόμενη εικόνα παρουσιάζονται τα στοιχεία που αποτελούν το EUTRAN:



Σχήμα 1.2: Παρουσίαση του EUTRAN [5]

1.4.2. Evolved Packet Core (EPC)

1.4.2.1. Serving Gateway (S-GW)

Είναι το σημείο σύνδεσης του air interface με το core network [7]. Ανήκει στο επίπεδο του χρήστη (user plane) και παρέχει τις ακόλουθες λειτουργίες:

- Είναι το σημείο αναφοράς κατά την εκτέλεση μεταπομπών των UE ανάμεσα σε eNB καθώς επίσης για την κινητικότητα μεταξύ δικτύων 3GPP
- Δρομολόγηση και επαναπροώθηση πακέτων
- Προσωρινή αποθήκευση πακέτων (buffering)
- Επισήμανση πακέτων στο επίπεδο μεταφοράς για το uplink και το downlink.
- Συλλογή δεδομένων χρεώσεων (Charging Data Record, CDR)
- Διαχείριση εμπειρίας υπηρεσίας για σκοπούς χρεώσεων (Quality of service Class Identifier, QCI)

1.4.2.2. Packet Data Network Gateway (PDN-GW)

Είναι ανάμεσα στο S-GW και τα εξωτερικά δίκτυα δεδομένων (PDNs) [7]. Στην ουσία είναι η πύλη του EPC προς τον έξω κόσμο. Περιλαμβάνει τις ακόλουθες υπηρεσίες:

- Καθορισμό IP διευθύνσεων στα UE
- Φιλτράρισμα δεδομένων που μπορεί να γίνει σε επίπεδο κάθε χρήστη
- Χρέωση υπηρεσιών τόσο στο uplink όσο και στο downlink

1.4.2.3. Mobility Management Entity (MME)

Το MME είναι υπεύθυνο για τη σηματοδοσία των UE προς άλλες οντότητες του δικτύου [7]. Αυτή η σηματοδοσία αφορά στο επίπεδο του ελέγχου (control plane) και όχι στο επίπεδο του χρήστη (user plane). Η σηματοδοσία ελέγχου περνάει μόνο από το MME ενώ η σηματοδοσία χρήστη περνάει από τα S-GW και P-GW. Τα MME παρέχουν τις ακόλουθες λειτουργίες:

- Σηματοδοσία στο Non Access Stratum (NAS)
- Ασφάλεια στη σηματοδοσία NAS
- Έλεγχο ασφάλειας στο Access Stratum
- Επιλογή των S-GW και P-GW
- Επιλογή άλλου MME κατά τη διαδικασία μεταπομπής
- Επιλογή του Serving GPRS Support Node (SGSN) κατά τη μεταπομπή προς συστήματα 2G ή 3G
- Επιλογή αντίστοιχων ελεγκτών μη 3GPP δικτύων
- Διαχείριση της λίστας αναζήτησης περιοχής (Tracking Area, TA list)
- Διαχείριση διεθνούς ή εθνικής περιαγωγής
- Αυθεντικοποίηση χρηστών
- Δημιουργία των φορέων συνδέσεων με το δίκτυο (bearer)
- Αναζήτηση κινητού όταν βρίσκεται σε κατάσταση αναμονής (idle state)

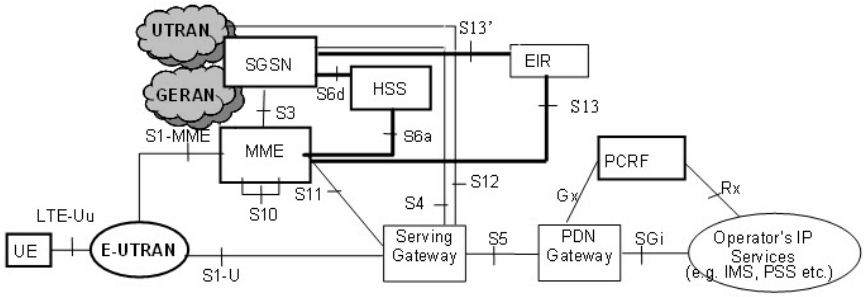
1.4.2.4. Home Subscriber Server (HSS)

Είναι στην ουσία μια βάση δεδομένων που περιέχει πληροφορίες σχετικές με το προφίλ του χρήστη και τις υπηρεσίες του [7]. Αλληλεπιδρά με όλα τα MME του

δικτύου. Οι κυριότερες λειτουργίες είναι η αυθεντικοποίηση, η κρυπτογράφηση και η προστασία κάθε χρήστη.

1.4.3. Διεπαφές (Interfaces)

Στο σχήμα που ακολουθεί εμφανίζονται οι διεπαφές μεταξύ των οντοτήτων που βρίσκονται εντός του EPS καθώς και αυτές μεταξύ του EPS και άλλων δικτύων 3GPP όπως 2G και 3G.



Σχήμα 1.3: Οι διεπαφές του LTE/SAE [8]

1.4.3.1. Διεπαφή Uu

Είναι μεταξύ του UE και του eNB. Αντίστοιχη διεπαφή στα συστήματα 3G είχε το RNC και τώρα πια έχει ενσωματωθεί στο σταθμό βάσης. Για το λόγο αυτό στο air interface η μόνη οντότητα που υπάρχει και καλύπτει όλους τους ρόλους είναι το eNB.

1.4.3.2. Διεπαφή X2

Πραγματοποιεί τη σύνδεση των eNB μεταξύ τους [6]. Κυρίως αναφέρεται στις πιθανές μεταπομπές των UE ανάμεσα στα eNB. Επίσης μεταφέρει την απαραίτητη σηματοδοσία για τη διαχείριση των ραδιοπόρων. Σε πρακτικό επίπεδο χρησιμοποιούνται συνήθως οπτικές ίνες ώστε να επιτυγχάνονται οι επιθυμητές τιμές καθυστερήσεων.

1.4.3.3. Διεπαφή S1

Υπάρχουν δύο ειδών διεπαφών S1. Η S1-MME και η S1-U. Η S1-MME αναφέρεται στη σηματοδοσία ελέγχου ανάμεσα στο eNB και το MME, ενώ η S1-U στη μεταφορά των δεδομένων του χρήστη μεταξύ του eNB και του S-GW. Τα δεδομένα αυτά χρησιμοποιούν το πρωτόκολλο GTP (GPRS Tunneling Protocol) το οποίο κάνει ενθυλάκωση στα δεδομένα καθώς κινούνται εντός του EPC.

1.4.3.4. Διεπαφή S3

Η διεπαφή αυτή βρίσκεται ανάμεσα στο MME και το SGSN των άλλων 3GPP δικτύων (2G και 3G). Εξυπηρετεί στον έλεγχο της κινητικότητας ανάμεσα σε αυτά τα δίκτυα.

1.4.3.5. Διεπαφή S4

Ενώνει το S-GW με το SGSN. Στην ουσία εξυπηρετεί την κινητικότητα των χρηστών μεταξύ των δικτύων 3GPP προσφέροντας ένα GTP-τούνελ για τη μεταφορά των δεδομένων κατά τη διάρκεια των μεταπομπών.

1.4.3.6. Διεπαφή S5

Βρίσκεται ανάμεσα στο S-GW και το P-GW. Διαχειρίζεται το τούνελ καθώς και τη μεταφορά των δεδομένων ανάμεσα στις δύο αυτές οντότητες και διευκολύνει την αναζήτηση νέου S-GW στην περίπτωση μεταπομπών των UE.

1.4.3.7. Διεπαφή S6a

Μεταφέρει τις πληροφορίες σχετικά με την αυθεντικοποίηση και τη συνδρομή των χρηστών μεταξύ του MME και του HSS.

1.4.3.8. Διεπαφή S11

Μεταφέρει τη σηματοδοσία μεταξύ του S-GW και του MME

1.4.3.9. Διεπαφή SGi

Ορίζεται μεταξύ του P-GW και των υπολοίπων IP δικτύων εκτός του EPS.

1.4.4. Στοίβες πρωτοκόλλων

Η αρχιτεκτονική των πρωτοκόλλων του LTE χωρίζονται σε δύο βασικές κατηγορίες. Σε αυτά που αναφέρονται στον έλεγχο (control plane) και σε αυτά που αναφέρονται στο χρήστη (user plane) [9].

Στην πλευρά του ελέγχου το πρωτόκολλο Radio Resource Control (RRC) δημιουργεί τη σηματοδοσία που χρειάζεται να ανταλλάξουν το UE με το σταθμό βάσης. Στην πλευρά του χρήστη η κάθε εφαρμογή δημιουργεί πακέτα δεδομένων τα οποία τα διαχειρίζονται πρωτόκολλα όπως IP, TCP, UDP. Και στις δύο πλευρές όλες οι πληροφορίες, πριν δοθούν στο φυσικό επίπεδο για μετάδοση, θα επεξεργαστούν από τα πρωτόκολλα Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC) και Medium Access Control (MAC).

1.4.4.1. Στοίβα πρωτοκόλλων του χρήστη (user plane)

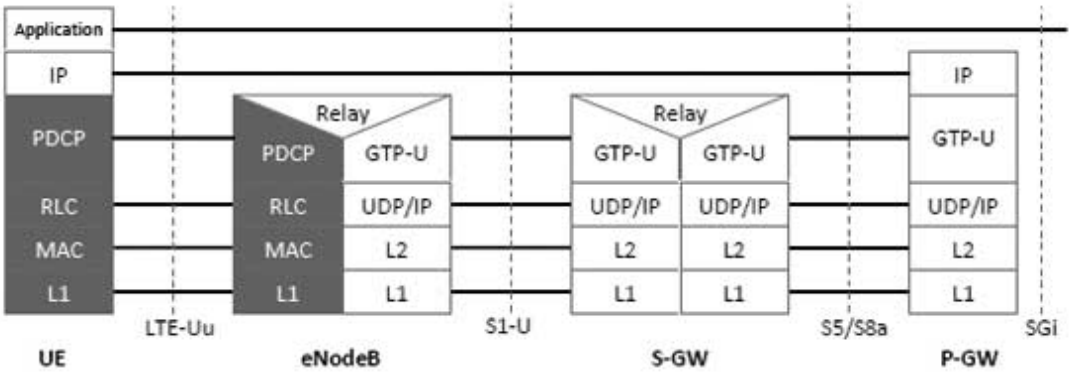
Η στοίβα πρωτοκόλλων του χρήστη περιέχει τα ακόλουθα επίπεδα:

- Packet Data Convergence Protocol (PDCP)
- Radio Link Control (RLC)
- Medium Access Control (MAC)
- Φυσικό επίπεδο

Τα πακέτα που παραλαμβάνει ένα επίπεδο ονομάζονται Service Data Unit (SDU) ενώ αυτά που εξάγει ονομάζονται Packet Data Unit (PDU).

Τα πακέτα που κινούνται εντός του EPC, από το P-GW έως το eNB, ενθυλακώνονται μα χρήση κάποιου πρωτοκόλλου κάθε φορά ανάλογα με τη διεπαφή που χρησιμοποιείται. Για παράδειγμα στις διεπαφές S1-U (ανάμεσα στο eNB και το S-GW) και S5 (ανάμεσα στο S-GW και το P-GW) χρησιμοποιείται το πρωτόκολλο GPRS Tunneling Protocol (GTP).

Στην εικόνα που ακολουθεί παρουσιάζονται οι στοίβες πρωτοκόλλων για το user plane.



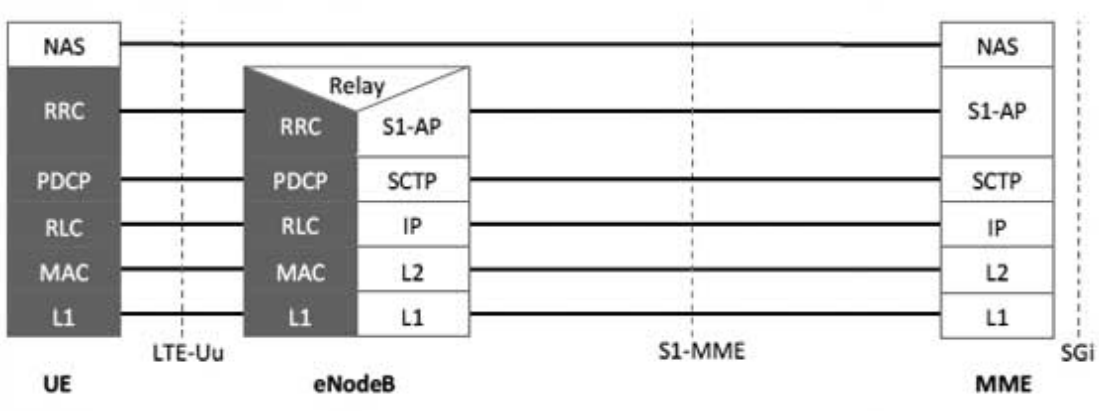
Σχήμα 1.4: Στοίβα πρωτοκόλλων του χρήστη (user plane) [10]

1.4.4.2. Στοίβα πρωτοκόλλων ελέγχου (control plane)

Χρησιμοποιείται η ίδια στοίβα πρωτοκόλλων με το user plane με την προσθήκη του Radio Resource Control (RRC) το οποίο καθορίζει τη λειτουργικότητα ανάλογα με την κατάσταση του UE. Δύο είναι οι πιθανές καταστάσεις του UE:

- Σε αναμονή (idle). Το UE βρίσκεται σε μια κυψέλη και παρακολουθεί ένα paging κανάλι ούτως ώστε να αντιληφθεί εισερχόμενες κλήσεις
- Συνδεδεμένο (connected). Το UE δίνει πληροφορίες στο eNB σχετικά με την ποιότητα του downlink καναλιού καθώς και την κατάσταση των γειτονικών κυψελών ώστε να βοηθήσει το EUTRAN να εντοπίσει την καλύτερη κυψέλη για αυτό.

Στην παρακάτω εικόνα παρουσιάζονται οι στοίβες πρωτοκόλλων για το control plane.



Σχήμα 1.5: Στοίβα πρωτοκόλλων ελέγχου (control plane) [10]

1.4.4.3. Ο ρόλος των πρωτοκόλλων NAS, RRC, PDCP, RLC, MAC

Το **Non Access Stratum (NAS)** είναι ένα σύνολο πρωτοκόλλων μεταξύ του UE και του MME που σκοπό έχουν μεταξύ άλλων να υποστηρίζουν την κινητικότητα των UE και τη διατήρηση της IP σύνδεσης μεταξύ των UE και του P-GW [11].

Το **Radio Resource Control (RRC)** υποστηρίζει όλη την απαραίτητη σηματοδοσία ανάμεσα στα UE και το eNB. Πιο συγκεκριμένα περιλαμβάνονται οι διαδικασίες που σχετίζονται με την κινητικότητα των τερματικών συσκευών όπως επίσης και η διαχείριση της σύνδεσής τους με το δίκτυο [11]. Ως παράδειγμα μπορεί να αναφερθεί η σηματοδοσία που αφορά στην αυθεντικοποίηση του κινητού από το δίκτυο.

Το Packet **Data Convergence Protocol (PDCP)** είναι υπεύθυνο για την υλοποίηση της ασφάλειας, δηλαδή της κρυπτογράφησης και της ακεραιότητας των ραδιοφορέων (radio bearers). Αυτοί οι ραδιοφορείς χρησιμοποιούνται τόσο στο user plane όσο και στο control plane [11]. Στο user plane αφορούν σε ροές πληροφορίας που αντιστοιχούν σε συγκεκριμένα δεδομένα (πλαίσια φωνής, steaming video κλπ). Στο control plane αναφέρονται στη σηματοδοσία των επιπέδων NAS και RRC που δημιουργούνται από το EPC.

Το **Radio Link Control (RLC)** προσφέρει στο PDCP υπηρεσίες που αντιστοιχούν στο επίπεδο 2 του OSI, όπως για παράδειγμα κατάτμηση πακέτων και ένα μηχανισμό για διόρθωση λαθών το Automatic Repeat Request (ARQ) [11].

Το **Medium Access Control (MAC)** έχει ως κύριο ρόλο την πολυπλεξία των λογικών καναλιών στα κανάλια μεταφοράς αφού πρώτα εκτελέσει μια διαχείριση προτεραιοτήτων [11]. Οι ροές που πολυπλέκονται σε ένα κανάλι μεταφοράς μπορούν να αφορούν έναν ή περισσότερους χρήστες. Το MAC επίσης υποστηρίζει μια διαδικασία επαναποστολής που ονομάζεται Hybrid Automatic Repeat Request (HARQ). Τέλος μεταφέρει τις ροές στο φυσικό επίπεδο το οποίο με τη σειρά του εκτελεί κωδικοποίηση καναλιού πριν τη μετάδοση στο air interface.

1.4.5. Κανάλια επικοινωνίας

Οι πληροφορίες που ανταλλάσσονται μεταξύ των διαφορετικών πρωτοκόλλων χρησιμοποιούν διάφορα κανάλια. Το LTE χρησιμοποιεί τρεις διαφορετικούς τύπους καναλιών που περιέχουν έναν αριθμό καναλιών έκαστος [12].

1.4.5.1. Λογικά κανάλια (logical channels)

Τα λογικά κανάλια καθορίζουν τον τύπο των δεδομένων που μεταφέρονται. Μεταφέρουν δεδομένα και μηνύματα σηματοδοσίας μεταξύ των πρωτοκόλλων RLC και MAC. Υπάρχουν δύο κατηγορίες λογικών καναλιών:

- Λογικά κανάλια ελέγχου που μεταφέρουν μηνύματα σηματοδοσίας στο control plane. Αυτά με τη σειρά τους χωρίζονται σε κοινά κανάλια (point to multipoint) σε όλους τους χρήστες και σε αφοσιωμένα κανάλια (point to point) που μπορούν να χρησιμοποιηθούν από ένα μόνο χρήστη
- Λογικά κανάλια κίνησης που μεταφέρουν δεδομένα στο user plane

Στον παρακάτω πίνακα παρουσιάζονται τα λογικά κανάλια του LTE:

Όνομα καναλιού	**Ακρωνύμιο**	**Κανάλι ελέγχου**	**Κανάλι κίνησης**
Broadcast Control Channel	BCCH	√	
Paging Control Channel	PCCH	√	
Common Control Channel	CCCH	√	
Dedicated Control Channel	DCCH	√	
Multicast Control Channel	MCCH	√	
Dedicated Traffic Channel	DTCH		√
Multicast Traffic Channel	MTCH		√

Πίνακας 1.1: Τα λογικά κανάλια του LTE [13]

1.4.5.2. Κανάλια μεταφοράς (transport channels)

Τα κανάλια μεταφοράς καθορίζουν το με ποιο τρόπο θα μεταφερθούν τα δεδομένα προς το φυσικό επίπεδο. Μεταφέρουν δεδομένα και μηνύματα σηματοδοσίας μεταξύ του MAC και του φυσικού επιπέδου.

Στον παρακάτω πίνακα παρουσιάζονται τα κανάλια μεταφοράς του LTE:

Όνομα καναλιού	**Ακρωνύμιο**	**Downlink**	**Uplink**
Broadcast Channel	BCH	√	
Downlink Shared Channel	DL-SCH	√	
Paging Channel	PCH	√	
Multicast Channel	MCH	√	
Uplink Shared Channel	UL-SCH		√
Random Access Channel	RACH		√

Πίνακας 1.2: Τα κανάλια μεταφοράς του LTE [13]

1.4.5.3. Φυσικά κανάλια (physical channels)

Στα φυσικά κανάλια μεταφέρονται τα δεδομένα και η σηματοδοσία ελέγχου στο φυσικό επίπεδο. Η μεταφορά τους γίνεται σε διαφορετικά επίπεδα των φυσικών καναλιών. Για το λόγο αυτό χωρίζονται σε δύο κατηγορίες:

- Φυσικά κανάλια δεδομένων, τα οποία παρουσιάζονται στον πίνακα που ακολουθεί:

Όνομα καναλιού	**Ακρωνύμιο**	**Downlink**	**Uplink**
Physical Downlink Shared Channel	PDSCH	√	
Physical Broadcast Channel	PBCH	√	
Physical Multicast Channel	PMCH	√	
Physical Uplink Shared Channel	PUSCH		√
Physical Random Access Channel	PRACH		√

Πίνακας 1.3: Φυσικά κανάλια δεδομένων του LTE [13]

- Φυσικά κανάλια ελέγχου, τα οποία παρουσιάζονται στον παρακάτω πίνακα:

Όνομα καναλιού	**Ακρωνύμιο**	**Downlink**	**Uplink**
Physical Control Format Indicator Channel	PCFICH	√	
Physical Hybrid ARQ Indicator Channel	PHICH	√	
Physical Downlink Control Channel	PDCCH	√	
Relay Physical Downlink Control Channel	R-PDCCH	√	
Physical Uplink Control Channel	PUCCH		√

Πίνακας 1.4: Φυσικά κανάλια ελέγχου του LTE [13]

1.5. Βασικές παράμετροι του LTE

1.5.1. Χρήση συχνοτικών ζωνών

Υπάρχει ένας αυξανόμενος αριθμός συχνοτικών ζωνών που καθορίζονται για χρήση από το LTE. Δύο είναι οι τρόποι χρήσης του διαθέσιμου φάσματος προκειμένου να επιτευχθεί αμφίδρομη επικοινωνία στα συστήματα τέταρτης γενιάς.

- Frequency Division Duplexing (FDD): χρειάζεται ένα ζευγάρι ζωνών, με ένα συχνοτικό κενό ανάμεσά τους. Το κενό αυτό είναι απαραίτητο για την αποφυγή παρεμβολών.
- Time Division Duplexing (TDD): Χρησιμοποιείται η ίδια συχνοτική ζώνη στην άνω και στην κάτω ζεύξη αλλά με χρονικό διαχωρισμό μεταξύ τους.

Τα εύρη ζωνών που χρησιμοποιούνται είναι 1.4MHz, 3MHz, 5MHz, 10MHz, 15MHz και 20MHz.

1.5.2. Διαμορφώσεις άνω και κάτω ζεύξης

Οι διαμορφώσεις που χρησιμοποιούνται στην άνω και στην κάτω ζεύξη είναι οι ακόλουθες:

- Άνω ζεύξη: QPSK και 16QAM
- Κάτω ζεύξη: QPSK, 16QAM και 64QAM

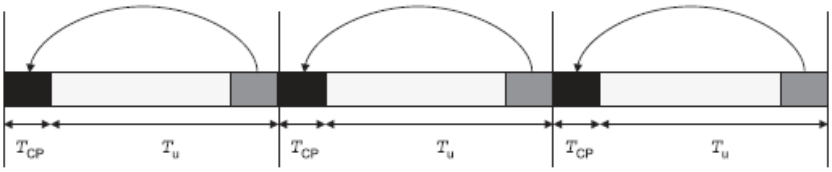
1.5.3. Μέθοδοι πολλαπλής πρόσβασης στο κανάλι μετάδοσης

Ένα από τα πιο σημαντικά θέματα που έπρεπε να λυθούν για την αποδοτικότερη χρήση του κοινού καναλιού ήταν η μέθοδος πολλαπλής πρόσβασης σε αυτό από διαφορετικούς χρήστες, τόσο στην άνω όσο και στην κάτω ζεύξη [14]. Υπάρχουν διάφορες τεχνολογίες. Αυτές που χρησιμοποιήθηκαν ήταν για την κάτω ζεύξη το Orthogonal Frequency Division Multiple Access (OFDMA) και για την άνω ζεύξη το Single Carrier Frequency Division Multiple Access (SC-FDMA).

1.5.3.1. Orthogonal Frequency Division Multiplex (OFDM)

Η τεχνολογία OFDM βασίζεται στην πολυπλεξία συχνοτικών φερόντων (subcarriers) που μεταφέρουν πληροφορίες σε ένα κανάλι επικοινωνίας. Το κάθε OFDM σύμβολο κωδικοποιείται σε διαφορετική συχνότητα και περιέχει δεδομένα. Λόγω των διαφορετικών μονοπατιών (multipath) που μπορεί να ακολουθήσει κάθε σήμα, δημιουργείται το φαινόμενο της διασυμβολικής παρεμβολής καθώς πολλαπλά αντίγραφα του αρχικού σήματος ανιχνεύονται από την κεραία. Αυτό μπορεί να αντιμετωπιστεί αν παρεμβληθεί ένας κενός χρόνος ανάμεσα στην εκπομπή κάθε

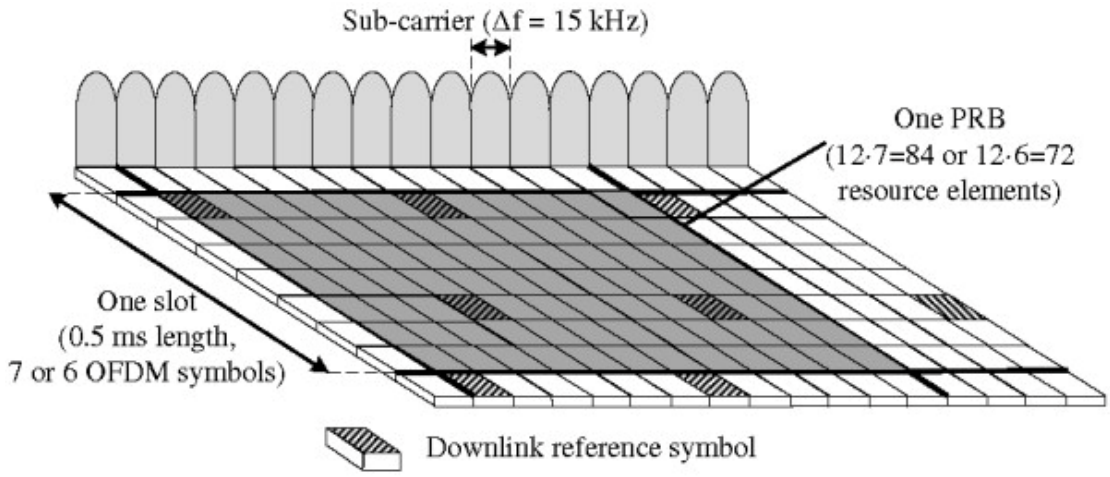
συμβόλου. Το σχήμα αυτό γίνεται αποδοτικότερο αν στον κενό αυτό χρόνο γίνει εκπομπή ενός τμήματος του τέλους του αρχικού σήματος (κυκλικό πρόθεμα, cyclic prefix).



Σχήμα 1.6: Το κυκλικό πρόθεμα (cyclic prefix) [9]

1.5.3.2. Orthogonal Frequency Division Multiple Access (OFDMA)

Το OFDMA έχει επιλεγεί για την κάτω ζεύξη. Τα κυριότερα πλεονεκτήματα είναι η αντοχή στο φαινόμενο του multipath και η αποδοτικότερη χρήση του διαθέσιμου φάσματος.



Σχήμα 1.7: Παρουσίαση του OFDMA [15]

Τα επιμέρους χαρακτηριστικά του είναι τα ακόλουθα:

- Το διαθέσιμο εύρος ζώνης (1.4, 3, 5,10,15 και 20MHz) χωρίζεται σε Resource Block (RB ή Physical Resource Block-PRB) των 180kHz έκαστο. Για παράδειγμα όταν το διαθέσιμο εύρος ζώνης είναι 20MHz το πλήθος των RB είναι 100.
- Το κάθε RB περιέχει 12 φέροντα (subcarriers) των 15KHz έκαστο.

- Σε κάθε time slot των 0.5 msec εκπέμπονται 7 σύμβολα OFDM ανά subcarrier, δηλαδή συνολικά 84 σύμβολα (7x12). Σε κάθε τερματική συσκευή εκχωρείται είτε ένα είτε περισσότερα resource blocks. Κάθε 1 msec μπορεί να αλλάζει αυτή η εκχώρηση (Transmission Time Interval, TTI).
- Κάθε κινητό μπορεί να εκπέμπει με διαφορετική διαμόρφωση ανάλογα με την ποιότητα του καναλιού μετάδοσης.

Το βασικό πλεονέκτημα του OFDMA είναι πως επειδή τα σύμβολα μεταδίδονται ταυτόχρονα μπορεί να χρησιμοποιηθεί χαμηλότερος ρυθμός μετάδοσης που έχει ως αποτέλεσμα την εξάλειψη της διασυμβολικής παρεμβολής (ISI).

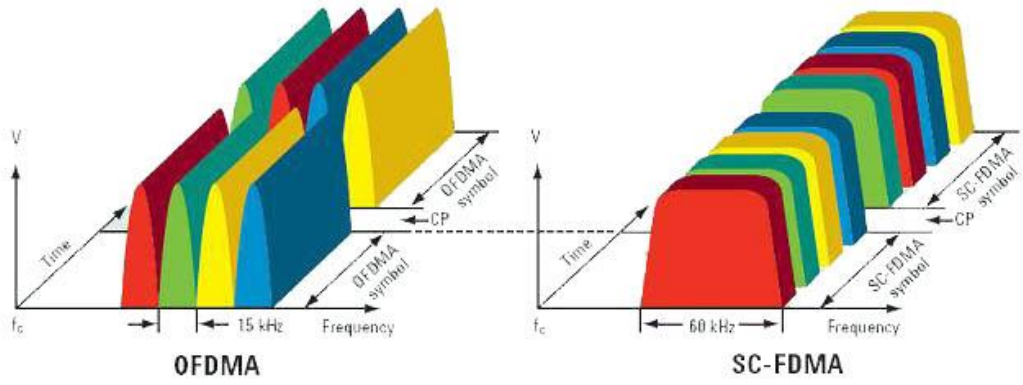
Το μειονέκτημα αυτής της μεθόδου είναι πως χρειάζεται μεγάλη ισχύ εκπομπής για να αποφευχθούν προβλήματα κατά την αποκωδικοποίηση στη λήψη.

1.5.3.3. Single Carrier FDMA (SC-FDMA)

Όπως έχει ήδη αναφερθεί το βασικό πρόβλημα του OFDMA είναι η υψηλή μέση τιμή της ισχύος (peak-average-power-ratio, PAPR). Αυτό έχει ως συνέπεια τη φασματική εξάπλωση των συμβόλων που οδηγεί σε διασυμβολική παρεμβολή και υψηλότερο ρυθμό σφαλμάτων (bit-error-rate, BER).

Στην κάτω ζεύξη το φαινόμενο αυτό μπορεί να αντιμετωπιστεί με τη χρήση ενισχυτών ισχύος καθώς και ειδικών μηχανισμών που δεν μπορούν να χρησιμοποιηθούν στις κινητές συσκευές των χρηστών.

Στην άνω ζεύξη χρησιμοποιείται η τεχνική του SC-FDMA η οποία κατά βάση εξαπλώνει συχνοτικά τα σύμβολα ενός χρήστη σε μια ομάδα φερόντων (subcarriers) συνεχόμενων ή μη. Αυτό έχει ως συνέπεια τη μείωση του PAPR.



Σχήμα 1.8: Σύγκριση OFDMA με το SC-FDMA [16]

1.5.4. Τεχνικές πολλαπλών κεραιών

Στις ασύρματες επικοινωνίες υπάρχουν τέσσερις βασικοί τρόποι πρόσβασης του καναλιού [17].

- Single Input Single Output (SISO)
- Single Input Multiple Output (SIMO)
- Multiple Input Single Output (MISO)
- Multiple Input Multiple Output (MIMO)

Η σημασία της χρήσης πολλαπλών κεραιών στη βελτίωση της απόδοσης των ασύρματων συστημάτων ήταν ήδη γνωστή για πολλά χρόνια. Η εμπορική όμως χρήση αυτών των τεχνικών έγινε στις αρχές του 2000 και πιο συγκεκριμένα στη Release 7 του HSDPA (High Speed Downlink Packet Access). Σύντομα όμως, με την εμφάνιση του LTE, δόθηκε η ευκαιρία να σχεδιαστεί ένα νέο σύστημα ασύρματων επικοινωνιών με τεχνικές MIMO ως ένα από τα βασικά συστατικά του.

Η βασική ιδέα είναι να χρησιμοποιηθεί η μετάδοση μέσω πολλαπλών μονοπατιών (multipath) όχι ως πηγή παρεμβολών αλλά ως ενίσχυση της ρυθμαπόδοσης (throughput) του συστήματος. Επίσης με τη χρήση πολλαπλών κεραιών στον πομπό και το δέκτη, μαζί με περίπλοκη επεξεργασία σήματος, επιτυγχάνεται αύξηση της χωρητικότητας του καναλιού.

1.6. Από το LTE στο LTE-Advanced

1.6.1. Η 8^{η} έκδοση του LTE (3GPP Release 8)

Στην 8^{η} έκδοση της 3GPP, όπου ολοκληρώθηκε το Δεκέμβριο του 2008, ορίστηκαν τα πρότυπα που αφορούσαν στο LTE RAN (Radio Access Network) και στο EPC (Evolved Packet Core) [18]. Τα βασικά χαρακτηριστικά που έφερε η Release 8 είναι τα ακόλουθα:

- Ταχύτητες για την κάτω ζεύξη στα 300Mbps και για την άνω ζεύξη στα 75Mbps.

- Πλήρη υποστήριξη για χρήση του φάσματος μέσω FDD (Frequency Division Duplex) και TDD (Time Division Duplex).
- Το EPC είναι πλέον ένα full IP δίκτυο. Αφαιρέθηκαν τα στοιχεία κυκλωμάτων μεταγωγής κυκλωμάτων που προϋπήρχαν.
- Υιοθετήθηκαν οι τεχνικές OFDMA και SC-FDMA για την κάτω και άνω ζεύξη αντίστοιχα ώστε να βελτιωθεί η χρησιμοποίηση του διαθέσιμου φάσματος.
- Υποστήριξη έξι καναλιών εύρους ζώνης από 1.4 έως 20MHz.
- Υποστήριξη για χωρική πολυπλεξία (MIMO) για μέχρι και τέσσερα επίπεδα στην κάτω ζεύξη.

1.6.2. Η 9η έκδοση του LTE (3GPP Release 9)

Η 9η έκδοση θεωρείται ως η ενδιάμεση μεταξύ της 8ης και της 10ης η οποία και αναφέρεται στο LTE-Advanced. Στη Release 9 υπάρχουν περίπου 80 διακριτές λειτουργίες από τις οποίες άλλες είναι νέες και άλλες δεν πρόλαβαν να ολοκληρωθούν στην 8η έκδοση [3]. Μερικές από αυτές παρουσιάζονται παρακάτω:

- Προσθήκη τεσσάρων νέων συχνοτικών ζωνών (18, 19,20 και 21 για FDD).
- Χρήση των Home Base Station (HeNB), ή αλλιώς femtocell. Πρόκειται για κεραίες μικρής κάλυψης που λειτουργούν εντός μιας μακροκυψέλης που εξυπηρετείται από ένα eNB και καλύπτουν χρήστες με χρήση υπάρχουσας σύνδεσης DSL. Μπορούν να χρησιμοποιούν είτε το ίδιο κανάλι με το eNB είτε δικό τους.
- Multimedia Broadcast Multicast Service (MBMS). Η υπηρεσία τηλεόρασης MBMS υπήρχε υλοποιημένη στο φυσικό επίπεδο από την έκδοση 8. Στην πράξη ξεκίνησε να λειτουργεί από την 9η έκδοση αλλά με πολλούς περιορισμούς.
- Self Organizing Networks (SON). Με τη χρήση τέτοιων δικτύων επιτυγχάνεται η αυτόματη παραμετροποίηση ενός eNB στο δίκτυο, αυτόματη βελτιστοποίηση της χωρητικότητας και της κάλυψης, αυτοματοποίηση των μεταπομπών, διαχείριση φόρτου του δικτύου κλπ.

1.6.3. Η 10η έκδοση του LTE (3GPP Release 10, LTE-Advanced)

Η 10η έκδοση της 3GPP αναφέρεται ως LTE-Advanced [19]. Οι σημαντικότερες προσθήκες που έγιναν είναι οι ακόλουθες:

- Υποστήριξη μεγαλύτερων ευρών ζωνών (wider bandwidths) και συνάθροιση φερόντων (carrier aggregation). Προκειμένου να επιτευχθούν οι επιθυμητές ταχύτητες στην κάτω ζεύξη (1Gbps) έπρεπε να χρησιμοποιηθούν εύρη ζώνης των 100MHz. Επειδή όμως η εύρεση διαθέσιμων συνεχόμενων 100MHz ήταν πολύ δύσκολη υιοθετήθηκε η συνάθροιση μη συνεχόμενων φερόντων. Αυτό έχει ως συνέπεια τις αλλαγές σε όλα τα επίπεδα, από το φυσικό μέχρι το RRC.
- Αλλαγή στο σχήμα εκπομπής στην άνω ζεύξη με τη χρήση χωρικής πολυπλεξίας μέχρι και τεσσάρων επιπέδων.
- Τομεοποιημένο SC-FDMA στην άνω ζεύξη. Δίνει τη δυνατότητα σε ένα UE να χρησιμοποιεί μη συνεχόμενες ομάδες φερόντων (subcarriers).
- Αλλαγή στο σχήμα εκπομπής στην κάτω ζεύξη με χρήση χωρικής πολυπλεξίας από τέσσερα έως οκτώ επίπεδα.
- Επαναμετάδοση (relaying). Αν και η ιδέα της αναμετάδοσης του σήματος ενός eNB από έναν αναμεταδότη δεν είναι καινούρια, έχει δεχθεί πολλές βελτιώσεις. Για παράδειγμα ο αναμεταδότης μπορεί επιλεκτικά να αναμεταδίδει την κίνηση προς ένα UE μειώνοντας ταυτόχρονα και τις παρεμβολές. Αυτό μπορεί να το επιτύχει τερματίζοντας τη στοίβα πρωτοκόλλων έως το επίπεδο 3 ώστε να λειτουργεί ως ένας ασύρματος δρομολογητής.
- Προσθήκη νέων συχνοτικών ζωνών (22, 23, 24, 25 για FDD και 41, 42, 43 για TDD).

1.6.4. Η 11η έκδοση του LTE (3GPP Release 11)

Μερικές από τις προσθήκες που συνεισέφερε η 11η έκδοση [20] είναι:

- Συντονισμένη εκπομπή από πολλαπλά σημεία (Coordinated Multipoint Transmission, CoMP). Η βασική διαφορά του MIMO από το CoMP είναι πως στη δεύτερη περίπτωση οι πομποί δε βρίσκονται στο ίδιο φυσικό σημείο.

- Εμπλουτισμός του Physical Downlink Control Channel (PDCCH). Η προσθήκη της συνάθροισης φερόντων (carrier aggregation) καθώς και της συντονισμένης εκπομπής από πολλαπλά σημεία (CoMP) απαιτούν περισσότερη σηματοδοσία.
- Βελτιστοποίηση του μηχανισμού αποφυγής των παρεμβολών με τη χρήση σημάτων αναφοράς (Reference Signals, RS).
- Προσθήκη νέων συχνοτικών ζωνών (26, 27,28 για FDD και 44 για TDD).

1.6.5. Η 12η έκδοση του LTE (3GPP Release 12)

Η δωδέκατη έκδοση βρίσκεται σε εξέλιξη κατά το διάστημα εκπόνησης της παρούσας εργασίας. Ο εκτιμώμενος χρόνος ολοκλήρωσης της έκδοσης είναι ο Δεκέμβριος του 2014. Μερικά από τα πεδία έρευνας της έκδοσης αυτής είναι:

- Περισσότερη ενεργειακή οικονομία.
- Υποστήριξη διαφορετικών εφαρμογών και μείωση της απαραίτητης σηματοδοσίας.
- Διαχωρισμό συχνοτήτων μεταξύ μεγάλων (eNB) και μικρών κυψελών (HeNB), με τις μικρές να λειτουργούν σε μεγαλύτερες συχνότητες (πχ 3.5GHz).
- Απευθείας επικοινωνία συσκευής με συσκευή (Device to Device, D2D).
- Υπηρεσίες εγγύτητας (proximity services).

Κεφάλαιο 2ο: Το περιβάλλον προσομοίωσης ns-3

2.1. Παρουσίαση του προσομοιωτή

Το ns-3 (Network Simulator 3) είναι ένας προσομοιωτής δικτύων διακριτών γεγονότων. Πρόκειται για ελεύθερο λογισμικό που χρησιμοποιεί την άδεια GNU GPLv2 και προορίζεται κυρίως για ερευνητικούς και εκπαιδευτικούς σκοπούς.

Ο όρος «διακριτά γεγονότα» στο πεδίο των προσομοιώσεων αναφέρεται στη μοντελοποίηση της λειτουργίας ενός συστήματος ως μια διακριτή ακολουθία γεγονότων στο χρόνο. Η κατάσταση του συστήματος αλλάζει όταν συμβεί κάποιο γεγονός κάποια συγκεκριμένη στιγμή. Στο διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών γεγονότων θεωρείται πως η κατάσταση του συστήματος παραμένει σταθερή.

Το ns-3, όπως υποδηλώνει το όνομά του, είναι το τρίτο μέλος μιας οικογένειας προσομοιωτών δικτύων διακριτών γεγονότων:

- Network Simulator 1 (ns-1). Αναπτύχθηκε στο Lawrence Berkeley National Laboratory το 1995. Προήλθε από έναν παλαιότερο προσομοιωτή με το όνομα REAL. Ο πυρήνας του ns-1 ήταν σε C++ και τα σενάρια ήταν Tcl script. Δε αναπτύσσεται ούτε χρησιμοποιείται πλέον.
- Network Simulator 2 (ns-2). Πρωτοπαρουσιάστηκε το 1996 και ο πυρήνας του είναι επίσης σε C++. Η χρήση της Tcl αντικαταστάθηκε από την Object Tcl του MIT η οποία είναι μια αντικειμενοστραφής διάλεκτος της Tcl. Η τελευταία έκδοση του NS-2, το 2009, δεν συντηρείται και ο προσομοιωτής δε γίνεται αποδεκτός για επιστημονικές δημοσιεύσεις.
- Network Simulator 3 (ns-3). Αποφασίστηκε να δημιουργηθεί από την αρχή και να μην έχει συμβατότητα με το NS-2. Η ανάπτυξή του άρχισε το 2006 αποκλειστικά σε C++. Στη συνέχεια προστέθηκε ένα framework για τη δημιουργία Python bindings (pybindgen) και η χρήση του Waf build system. Η έκδοση του προσομοιωτή η οποία χρησιμοποιήθηκε στην παρούσα εργασία είναι η 3.19.

Ο πυρήνας του προσομοιωτή ns-3 μπορεί να υποστηρίξει έρευνα σε IP και μη-IP δίκτυα. Η μεγάλη πλειοψηφία των χρηστών επικεντρώνεται σε ασύρματα IP δίκτυα

όπως LTE, Wi-Fi, WiMAX καθώς και σε στατικά και δυναμικά πρωτόκολλα δρομολόγησης όπως OLSR και AODV για IP εφαρμογές.

Ο ns-3 μπορεί να αλληλεπιδράσει με πραγματικά συστήματα [21]. Παραδείγματος χάριν οι χρήστες μπορούν να εκπέμψουν και να παραλάβουν πακέτα που δημιουργήθηκαν από τον προσομοιωτή σε πραγματικές δικτυακές συσκευές.

Υπάρχει μια μεγάλη κοινότητα χρηστών και προγραμματιστών που αναλαμβάνει το φορτίο της συντήρησης, ανάπτυξης, και τεκμηρίωσης ενός τόσο πολυδιάστατου συστήματος προσομοιώσεων.

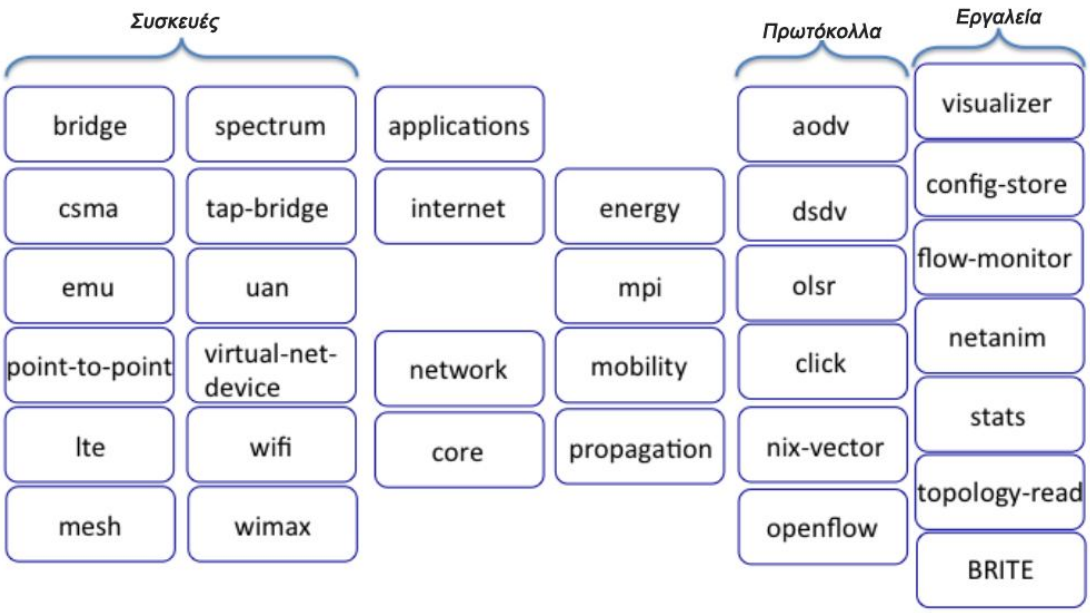
2.2. Διάρθρωση του προσομοιωτή

Αρχικά είναι απαραίτητο να γίνει ένας διαχωρισμός ανάμεσα στα μοντέλα προσομοίωσης (models) και στις μονάδες προσομοίωσης (modules) [22].

- Τα μοντέλα προσομοίωσης (models) είναι απλοποιημένες αναπαραστάσεις πραγματικών αντικειμένων, πρωτοκόλλων και συσκευών.
- Οι μονάδες προσομοίωσης (modules) είναι βιβλιοθήκες του ns-3 που αντιστοιχούν σε πραγματικά συστήματα (LTE, WiFi κλπ). Τα προγράμματα που δημιουργούνται στο ns-3 συνδέονται με τις αντίστοιχες βιβλιοθήκες ώστε να εκτελέσουν συγκεκριμένα σενάρια.

Το ns-3 προσφέρει μοντέλα προσομοίωσης σχετικά με το πώς λειτουργούν και συμπεριφέρονται δίκτυα μεταγωγής πακέτων. Παρόλο που η πλειοψηφία των χρηστών χρησιμοποιεί τον προσομοιωτή για σενάρια σχετικά με δίκτυα και Internet πρωτόκολλα, η χρήση του μπορεί να επεκταθεί και σε non-Internet-based συστήματα.

Οι μονάδες προσομοίωσης (modules) που παρέχει το λογισμικό παρουσιάζονται στο παρακάτω σχήμα:



Σχήμα 2.1: Μονάδες προσομοίωσης ns-3 [23]

Στο module network εμπεριέχονται οι κόμβοι, τα socket, οι ουρές, τα πακέτα κλπ. Αντιστοίχως στο module core περιέχονται στοιχεία όπως smart pointers, callbacks, event, scheduler, logging, tracing κλπ.

Το κάθε module έχει την ακόλουθη διάρθρωση φακέλων, χωρίς αυτό να σημαίνει πως χρησιμοποιούνται όλοι οι φάκελοι σε όλα τα module.

model/ : περιέχει τον πηγαίο κώδικα για το κεντρικό κομμάτι του module

helper/ : περιέχει κώδικα για τις helper classes

examples/ : περιέχει παραδείγματα

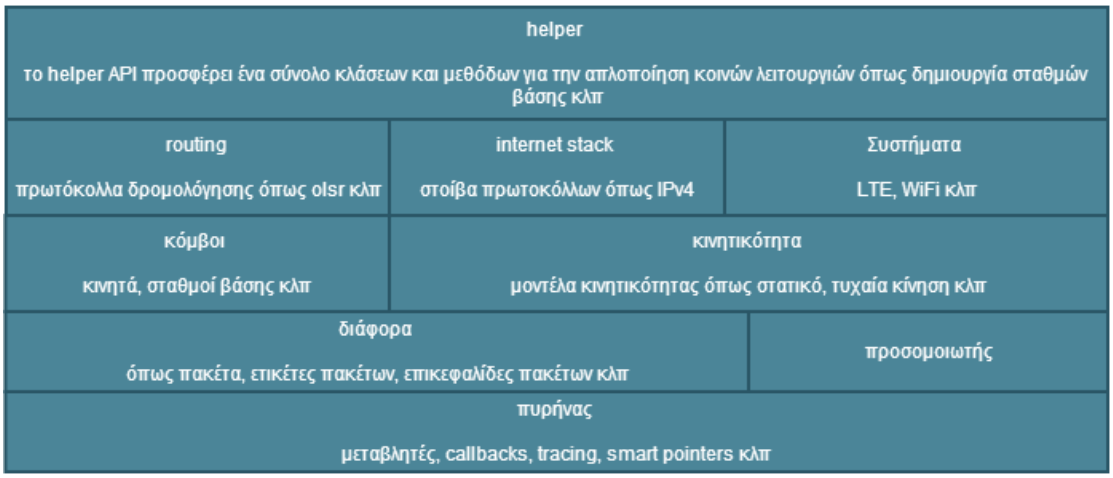
tests/ : παραδείγματα για την επικύρωση στοιχείων του προσομοιωτή

bindings/ : αρχεία σχετικά με την python

doc/ : έγγραφα τεκμηρίωσης

wscript/ : το ισοδύναμο του “Makefile”. Δίνει τη δυνατότητα να γίνεται compilation σε προγράμματα που αποτελούνται από διαφορετικά αρχεία (*.cc και *.h).

Στο παρακάτω σχήμα παρουσιάζεται μια αναπαράσταση σε μορφή στοίβας όλης της δομής του προσομοιωτή.



Σχήμα 2.2: Η διάρθρωση του ns-3

Ο προσομοιωτής σχεδιάστηκε ως ένα σύνολο βιβλιοθηκών που μπορούν να συνδυάζονται είτε μεταξύ τους είτε με εξωτερικές βιβλιοθήκες [24]. Σε αυτό οφείλεται και η απουσία γραφικού περιβάλλοντος, σε σχέση με άλλους προσομοιωτές δικτύων, καθώς διαθέτει σπονδυλωτή διάρθρωση. Αν και οι χρήστες είναι υποχρεωμένοι να εργάζονται σε γραμμή εντολών, μπορούν να χρησιμοποιήσουν άλλα εργαλεία για ανάλυση και οπτικοποίηση των δεδομένων τους. Χαρακτηριστικά παραδείγματα είναι το Gnuplot για γραφικές αναπαραστάσεις δεδομένων και το Wireshark για την ανάλυση δικτυακών πακέτων.
Χρησιμοποιείται κυρίως σε συστήματα Linux αλλά υποστηρίζεται από το FreeBSD και το Cygwin (για Windows). Επίσης είναι στη φάση της ανάπτυξης και η εγγενής υποστήριξη στο Visual Studio της Microsoft.

2.3. Δομή ενός προγράμματος προσομοίωσης

Η γενική μορφή ενός προγράμματος προσομοίωσης είναι η ακόλουθη [22]:

```
// Βοηθητικες συναρτησεις

int main (int argc, char *argv[]) {
…
}
```

Πιο συγκεκριμένα τα τμήματα από τα οποία αποτελείται ένα σενάριο στο ns-3 είναι τα ακόλουθα:

- Προσθήκη των κατάλληλων header files

```
• #include "ns3/core-module.h"
• #include "ns3/network-module.h"
• #include "ns3/internet-module.h"
• #include "ns3/point-to-point-module.h"
• #include "ns3/applications-module.h"
...
```

- Χρήση του ns-3 project namespace

```
using namespace ns3;
```

- Ενεργοποίηση ή απενεργοποίηση των μηνυμάτων στο τερματικό με αναφορά στο όνομα κάθε στοιχείου

```
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int main (int argc, char *argv[])
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
...
}
```

- Δημιουργία της τοπολογίας του σεναρίου

```
NodeContainer nodes;
nodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

```
...
```

- Δημιουργία της στοίβας Internet

```
InternetStackHelper stack;
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

- Δημιουργία των εφαρμογών

```
•   UdpEchoServerHelper echoServer (9);
•
•   ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
...
```

- Εκτέλεση του σεναρίου

```
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

2.4. Πηγές πληροφόρησης σχετικές με το ns-3

2.4.1. Παγκόσμιος ιστός

Υπάρχουν διάφορες τοποθεσίες που πρέπει να γνωρίζει ένας χρήστης του ns-3:

- Η βασική ιστοσελίδα του συστήματος είναι η http://www.nsnam.org
- Το Wiki που συμπληρώνει τη βασική ιστοσελίδα http://www.nsnam.org/wiki
- Ο πηγαίος κώδικας βρίσκεται στη σελίδα http://code.nsnam.org. Εκεί βρίσκεται το τρέχον development tree στο αποθετήριο ns-3-dev.

2.4.2. Πηγαίος κώδικας και τεκμηρίωση

Όπως κάθε άλλο πολύπλοκο σύστημα, έτσι και το ns-3, χρειάζεται έναν τρόπο ώστε να οργανώνει τις αλλαγές στον κώδικα και στην τεκμηρίωσή του. Όσον αφορά στον κώδικα έχει χρησιμοποιηθεί το Mercurial. Η τεκμηρίωσή του έχει βασιστεί στο Doxygen.

2.4.3. Build system

Μετά την αποθήκευση τοπικά του πηγαίου κώδικα θα πρέπει να γίνει compiling ώστε να δημιουργηθούν χρήσιμα προγράμματα. Για αυτό το σκοπό το ns-3 χρησιμοποιεί το Waf, το οποίο είναι ένα Python-based build system.

2.4.4. Περιβάλλον ανάπτυξης

Όπως έχει ήδη αναφερθεί τα μοντέλα του προσομοιωτή είναι γραμμένα σε C++, οπότε θεωρείται απαραίτητη η γνώση προγραμματισμού από το χρήστη. Το περιβάλλον ανάπτυξης σεναρίων είναι η γραμμή εντολών. Επικουρικά μπορούν να χρησιμοποιηθούν και άλλα εργαλεία επεξεργασίας των δεδομένων που παράγονται από το ns-3.

2.5. Προετοιμασία και εγκατάσταση

Το ns-3 είναι ένα σύνολο βιβλιοθηκών που λειτουργούν συνδυαστικά. Τα προγράμματα των χρηστών μπορούν να είναι είτε σε C++ είτε σε Python και να συνδέονται σε βιβλιοθήκες ή/και να δέχονται εισόδους από αυτές. Ο προσομοιωτής προσφέρεται ως πηγαίος κώδικας, οπότε είναι απαραίτητο πρώτα να γίνει build στις βιβλιοθήκες και κατόπιν στα προγράμματα.
Αν και θα μπορούσε να προσφέρεται ως pre-built βιβλιοθήκες για συγκεκριμένα συστήματα αυτό δε συμβαίνει καθώς έτσι δίνεται η δυνατότητα στους χρήστες που το επιθυμούν να τροποποιούν τον πηγαίο κώδικα.

2.5.1. Προαπαιτούμενα (για συστήματα Ubuntu/Debian)

Ο πυρήνας του ns-3 χρειάζεται την εγκατάσταση gcc/g++ από την έκδοση 3.4 και άνω, καθώς και Python από την έκδοση 2.4 και άνω. Επίσης χρειάζονται και μερικά πακέτα ώστε να υποστηρίζονται διαφορετικές επιλογές του προσομοιωτή. Παρακάτω παρουσιάζονται τα αντίστοιχα για την έκδοση 12.04 του Ubuntu:

```
sudo apt-get install gcc g++ python
sudo apt-get install gcc g++ python python-dev
sudo apt-get install mercurial
sudo apt-get install bzr
sudo apt-get install gdb valgrind
sudo apt-get install gsl-bin libgsl0-dev libgsl0ldbl
sudo apt-get install flex bison libfl-dev
sudo apt-get install g++-3.4 gcc-3.4
sudo apt-get install tcpdump
sudo apt-get install sqlite sqlite3 libsqlite3-dev
sudo apt-get install libxml2 libxml2-dev
sudo apt-get install libgtk2.0-0 libgtk2.0-dev
sudo apt-get install vtun lxc
sudo apt-get install uncrustify
sudo apt-get install doxygen graphviz imagemagick
sudo apt-get install texlive texlive-extra-utils texlive-latex-extra
sudo apt-get install python-sphinx dia
sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev
sudo apt-get install libboost-signals-dev libboost-filesystem-dev
```

```
sudo apt-get install openmpi-bin openmpi-common openmpi-doc
libopenmpi-dev
sudo apt-get install gcc-multilib
```

2.5.2. Μεταφόρτωση με τη χρήση Tarball, build και εκτέλεση των test

Το Tarball είναι μια συγκεκριμένη μορφή φακέλου λογισμικού στην οποία όλα τα απαραίτητα αρχεία είναι συγκεντρωμένα σε ένα φάκελο που συνήθως είναι συμπιεσμένος [24]. Όλες οι εκδόσεις του προσομοιωτή είναι διαθέσιμες στη συγκεκριμένη μορφή έτοιμες για μεταφόρτωση στο τοπικό σύστημα κάθε χρήστη. Η διαδικασία είναι σχετικά απλή. Αρκεί να επιλεγεί η έκδοση του προσομοιωτή, να μεταφορτωθεί τοπικά και να αποσυμπιεστεί.
Αν υποτεθεί πως ο τοπικός φάκελος αποθήκευσης λέγεται workspace και βρίσκεται στο path /workspace, πληκτρολογούμε στο τερματικό:

```
$ cd
$ mkdir workspace
$ cd workspace
$ wget http://www.nsnam.org/releases/ns-allinone-3.20.tar.bz2
$ tar xjf ns-allinone-3.20.tar.bz2
$ cd ns-allinone-3.20
$ ls
bake      constants.py   ns-3.20               README
build.py  netanim-3.103  pybindgen-0.16.0.825  util.py
```

Στο σημείο αυτό μπορούμε να κάνουμε build την έκδοση του ns-3:

```
$ ./build.py --enable-examples --enable-tests
…
…
Waf: Leaving directory `/path/to/workspace/ns-allinone-3.20/ns-3.20/build'
'build' finished successfully (6m25.032s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      emu
energy                   fd-net-device            flow-monitor
internet                 lte                      mesh
mobility                 mpi                      netanim (no Python)
network                  nix-vector-routing       olsr
point-to-point           point-to-point-layout    propagation
spectrum                 stats                    tap-bridge
test (no Python)         tools                    topology-read
```

```
uan                       virtual-net-device        wifi
wimax

Modules not built (see ns-3 tutorial for explanation):
brite                     click                     openflow
visualizer

Leaving directory `./ns-3.20'
```

Ακολουθεί η εκτέλεση των test που περιέχονται στο σύστημα:

```
$ ./test.py -c core
Waf: Entering directory `/path/to/workspace/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/path/to/workspace/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (1.799s)

Modules built:
aodv                      applications              bridge
click                     config-store              core
csma                      csma-layout               dsdv
emu                       energy                    flow-monitor
internet                  lte                       mesh
mobility                  mpi                       netanim
network                   nix-vector-routing        ns3tcp
ns3wifi                   olsr                      openflow
point-to-point            point-to-point-layout     propagation
spectrum                  stats                     tap-bridge
template                  test                      tools
topology-read             uan                       virtual-net-device
visualizer                wifi                      wimax

PASS: TestSuite ns3-wifi-interference
PASS: TestSuite histogram

...

PASS: TestSuite object
PASS: TestSuite random-number-generators
92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)
```

Στην περίπτωση που η τελευταία γραμμή δεν εμφανίζει μηνύματα σφαλμάτων το σύστημα έχει γίνει built σωστά και μπορεί να χρησιμοποιηθεί από το χρήστη.

2.5.3. Εκτέλεση σεναρίων προσομοίωσης

Τα σενάρια προσομοίωσης εκτελούνται με χρήση του Waf [24]. Τα αρχεία με κατάληξη cc τα οποία περιέχουν τον κώδικα του σεναρίου πρέπει να βρίσκονται στο φάκελο /scratch. Χρησιμοποιείται η επιλογή --run του Waf.

Αν για παράδειγμα το πρόγραμμα που περιέχει ένα σενάριο προσομοίωσης και βρίσκεται στο φάκελο /scratch είναι το progname.cc, η εκτέλεσή του γίνεται με χρήση της εντολής

```
$ ./waf --run progname
```

Άρα προκειμένου να εκτελεστεί το βασικό πρόγραμμα σε όλες τις γλώσσες προγραμματισμού «Hello World» εκτελούμε την εντολή

```
$ ./waf --run hello-simulator
```

και λαμβάνουμε την έξοδο

```
Hello Simulator
```

Αυτό σημαίνει πως ο προσομοιωτής είναι έτοιμος για χρήση.

Κεφάλαιο 3ο: Το σύστημα LENA και η υποστήριξη μεταπομπών

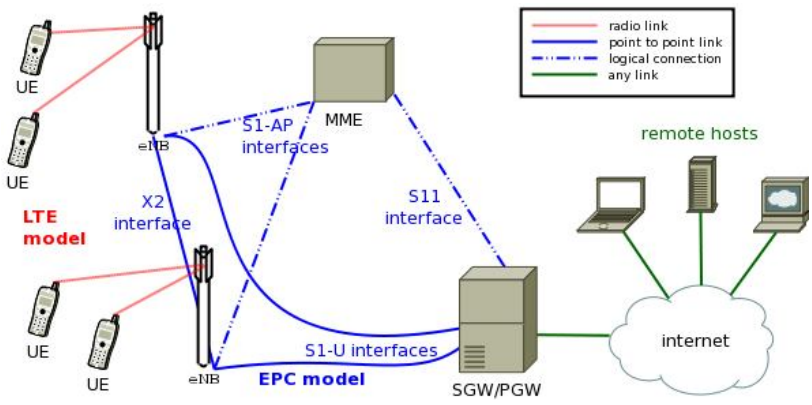
Το σύστημα LENA είναι ένα LTE module του προσομοιωτή δικτύων ns-3 [25]. Δημιουργήθηκε με τη συνεργασία της εταιρείας Ubiquisys (πλέον τμήμα της Cisco) και του Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). Η Ubuquisys είναι μια εταιρεία που ειδικεύεται στην ανάπτυξη έξυπνων κεραιών ενώ το CTTC είναι ένα κέντρο ερευνών σχετικά με τις τεχνολογίες επικοινωνιών.

Το συγκεκριμένο σύστημα προσομοίωσης δίνει τη δυνατότητα του ελέγχου της λειτουργίας τόσο μεγάλων (macro) όσο και μικρών (femto) κυψελών ούτως ώστε να επιτευχθεί η καλύτερη διαλειτουργικότητά τους. Στα δίκτυα WCDMA οι macrocells και femtocells λειτουργούν ανεξάρτητα. Αντίθετα στο LTE λειτουργούν σαν ένα Self Organized Network (SON).

Το σύστημα LENA μπορεί να χρησιμοποιηθεί για το σχεδιασμό και την αποτίμηση της απόδοσης uplink και downlink schedulers, αλγορίθμων διαχείρισης ραδιοπόρων, εξισορρόπηση φόρτου, διαχείριση κινητικότητας κλπ.

Υπάρχουν δύο βασικά συστατικά του συστήματος LENA:

- Το μοντέλο LTE που περιλαμβάνει τη στοίβα πρωτοκόλλων (RRC, PDCP, RLC, MAC, PHY). Αυτές οι οντότητες υπάρχουν τόσο σε κάθε UE όσο και σε κάθε eNB.
- Το μοντέλο EPC που περιλαμβάνει τις διεπαφές του δικτύου (X2, S1, κλπ), τα πρωτόκολλα και τις οντότητες. Οι οντότητες και τα πρωτόκολλα υπάρχουν εντός των MME, S-GW, P-GW και μερικώς στα eNB.



Σχήμα 3.1: Το μοντέλο LTE/EPC του συστήματος LENA [29]

3.1. Κριτήρια σχεδίασης μοντέλων

3.1.1. Μοντέλο LTE

Το μοντέλο LTE σχεδιάστηκε ώστε να υποστηρίζει τα ακόλουθα:

- Διαχείριση ραδιοπόρων (Radio Resource Management).
- Προγραμματισμό πακέτων προσανατολισμένο στην ποιότητα υπηρεσίας (QoS-aware Packet Scheduler).
- Συντονισμό παρεμβολών μεταξύ κυψελών (Inter-cell Interference Coordination).
- Δυναμική πρόσβαση φάσματος (Dynamic Spectrum Access).

Το μοντέλο αναλύει μέχρι το επίπεδο του Resource Block (RB), το οποίο είναι το βασικό συστατικό όσον αφορά στο διαμοιρασμό των πόρων. Επίσης μπορεί να υποστηρίζει μερικές δεκάδες eNB και μερικές εκατοντάδες UE.

3.1.2. Μοντέλο EPC

Ο βασικός σκοπός του μοντέλου EPC είναι να προσφέρει τη δυνατότητα στα UE να αποκαθιστούν σύνδεση IP μέσω του μοντέλου LTE. Προκειμένου να το επιτύχει υποστηρίζει τη διασύνδεση πολλών UE στο Internet μέσω ενός δικτύου eNB που συνδέονται σε μια οντότητα S-GW/P-GW. Οι παρακάτω παραδοχές υπάρχουν στο μοντέλο EPC:

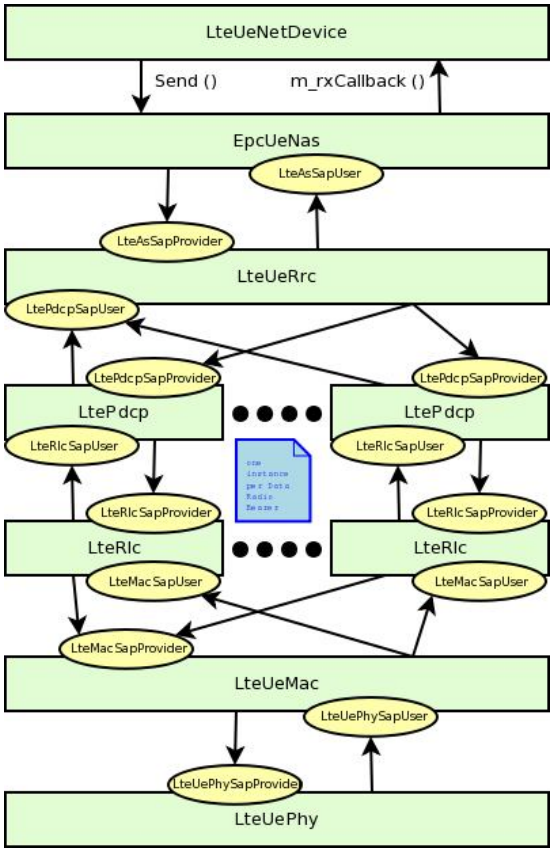
- Τα S-GW και P-GW είναι υλοποιημένα σε μία οντότητα που ονομάζεται SGW/PGW. Υποστηρίζεται μόνο το IPv4.
- Ο βασικός σκοπός του EPC είναι να προσομοιώνει την απόδοση εφαρμογών, που είναι βασισμένες σε TCP ή UDP, από άκρο σε άκρο.
- Παρέχεται η δυνατότητα στα UE να χρησιμοποιούν εφαρμογές που έχουν διαφορετικά προφίλ ποιότητας υπηρεσίας (QoS). Αυτό σημαίνει πως πρέπει να υποστηρίζονται διαφορετικοί EPS φορείς (bearers) για κάθε UE. Αυτό σημαίνει πως τόσο στο UE, στην άνω ζεύξη, όσο και στο P-GW, στην κάτω ζεύξη, θα πρέπει να γίνεται ταξινόμηση της κίνησης TCP/UDP.

- Το μοντέλο EPC επικεντρώνεται στην προσομοίωση των ενεργών χρηστών. Έτσι η λειτουργικότητα που αφορά στην κατάσταση αναμονής (idle state) δε μοντελοποιείται.
- Το μοντέλο επιτρέπει τη μεταπομπή μεταξύ δύο eNB που βασίζεται στη διεπαφή X2.

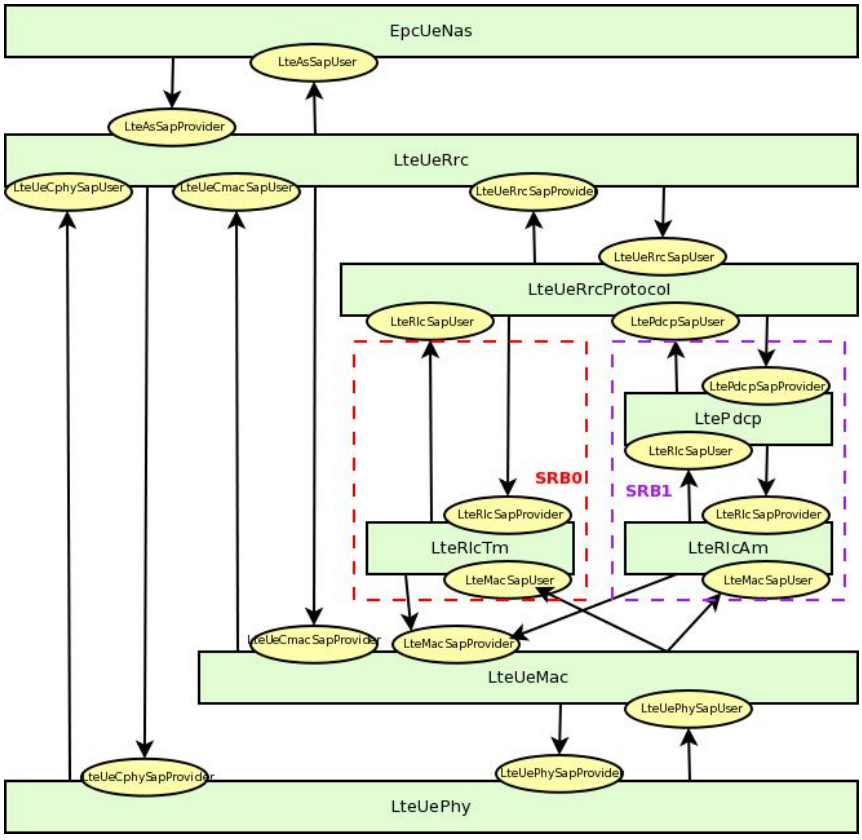
3.2. Αρχιτεκτονική μοντέλου LTE

3.2.1. Αρχιτεκτονική UE

Η αρχιτεκτονική της στοίβας πρωτοκόλλων του UE σχετικά με τα δεδομένα και τον έλεγχο (data και control plane) παρουσιάζονται στα παρακάτω σχήματα:



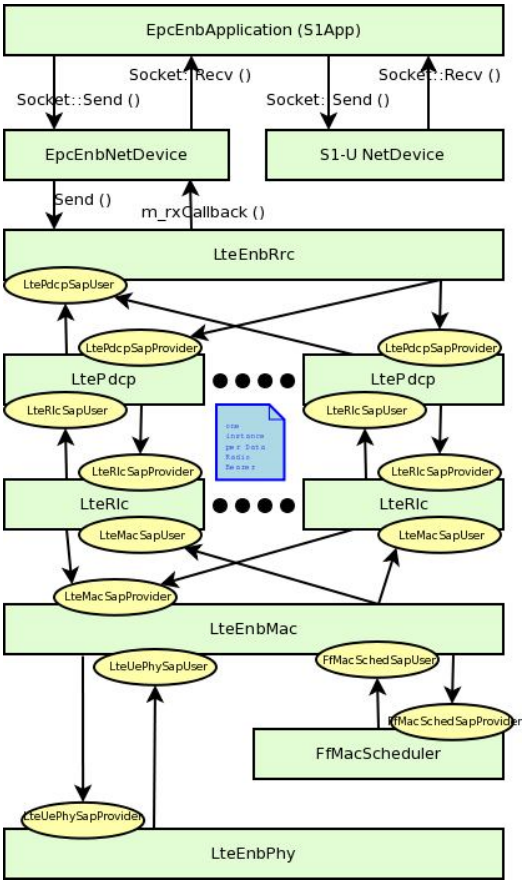
Σχήμα 3.2: Στοίβα πρωτοκόλλων δεδομένων UE [25]



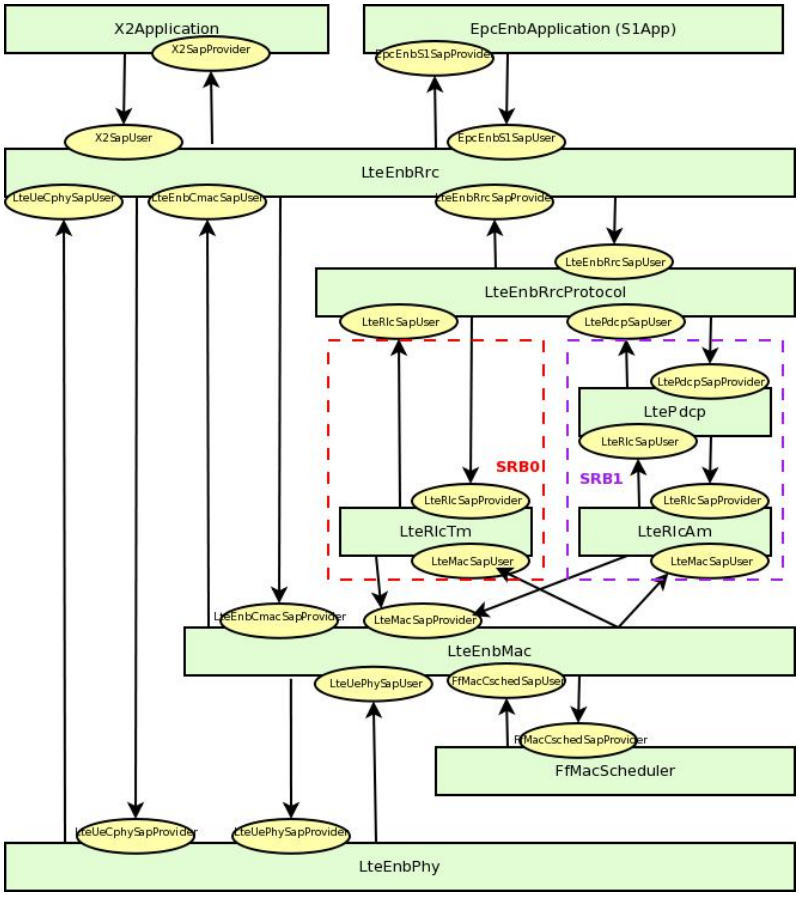
Σχήμα 3.3: Στοίβα πρωτοκόλλων ελέγχου UE [29]

3.2.2. Αρχιτεκτονική eNB

Η αρχιτεκτονική της στοίβας πρωτοκόλλων του eNB σχετικά με τα δεδομένα και τον έλεγχο (data και control plane) παρουσιάζονται στα παρακάτω σχήματα:



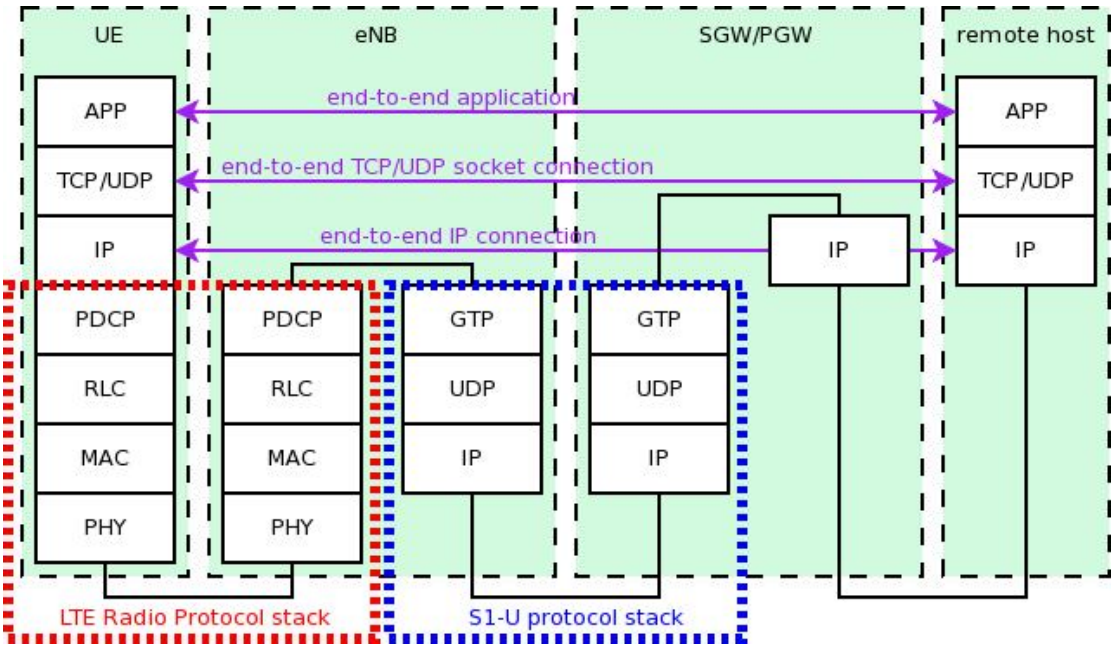
Σχήμα 3.4:Στοίβα πρωτοκόλλων δεδομένων eNB [25]



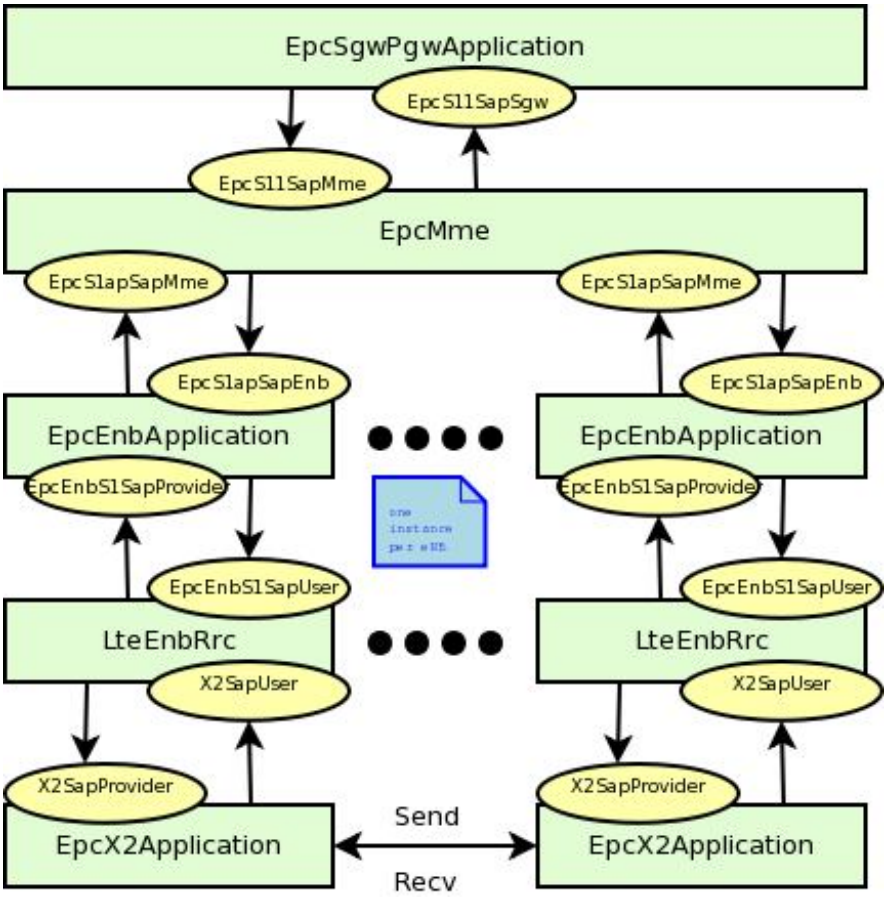
Σχήμα 3.5: Στοίβα πρωτοκόλλων ελέγχου eNB [29]

3.2.3. Αρχιτεκτονική EPC

Όσον αφορά στη στοίβα πρωτοκόλλων δεδομένων (data plane) και επειδή οι οντότητες S-GW και P-GW είναι μοντελοποιημένες ως μία δεν υπάρχει η ανάγκη για την ύπαρξη των διεπαφών S5 και S8 όπως καθορίζονται από τη 3GPP.
Σχετικά με την αρχιτεκτονική ελέγχου οι διεπαφές που έχουν μοντελοποιηθεί είναι οι S1-AP, X2-AP και S11.



Σχήμα 3.6: Στοίβα πρωτοκόλλων δεδομένων LTE/EPC [25]



Σχήμα 3.7: Μοντέλο ελέγχου EPC [25]

3.3. Κανάλι και διάδοση

Για τις ανάγκες μοντελοποίησης καναλιού, το LTE module χρησιμοποιεί τη διεπαφή SpectrumChannel από το spectrum module. Υπάρχουν δύο υλοποιήσεις της συγκεκριμένης διεπαφής, το SingleModelSpectrumChannel και το MultiModel SpectrumChannel. Το LTE module για να λειτουργήσει χρειάζεται τη χρήση του MultiModelSpectrumChannel διότι είναι αναγκαία η υποστήριξη σε διαφορετικές συχνότητες και εύρη ζώνης.

3.3.1. Χρήση των κτηρίων στο LTE

Το προτεινόμενο μοντέλο διάδοσης που χρησιμοποιείται στο LTE είναι αυτό που παρέχεται από το Buildings module. Το μοντέλο LTE χρησιμοποιεί μόνο FDD και υλοποιεί την άνω και την κάτω ζεύξη χωριστά. Συνεπώς διενεργούνται οι ακόλουθοι υπολογισμοί:

- eNB με UE (εσωτερικού και εξωτερικού χώρου)
- femtocell (εσωτερικού και εξωτερικού χώρου) με UE (εσωτερικού και εξωτερικού χώρου)

Το Buildings model δεν ξεχωρίζει το τύπο του κόμβου που εκπέμπει (UE, eNB, femtocell), παρά μόνο τη θέση του (εσωτερικού ή εξωτερικού χώρου) και το ύψος του (άξονας z). Έτσι στην περίπτωση ενός eNB που βρίσκεται σε εξωτερικό χώρο και πάνω από το ύψος της οροφής του κτηρίου θα χρησιμοποιηθεί μοντέλο διάδοσης σταθμού βάσης μακροκυψέλης. Αντιθέτως στην περίπτωση εσωτερικού χώρου ή σε ύψος χαμηλότερο της οροφής κτηρίου θα χρησιμοποιηθεί μοντέλο διάδοσης femtocell.

3.3.2. Μοντέλο εξασθένησης

Ως μαθηματικό μοντέλο διάδοσης στο κανάλι χρησιμοποιείται η συνάρτηση rayleighchan του Matlab. Επιλέχθηκε γιατί παρέχει μια αποδεκτή μοντελοποίηση και στο πεδίο του χρόνου και στο πεδίο της συχνότητας.

3.3.3. Κεραίες

Το μοντέλο LTE PHY, που βασίζεται στο SpectrumPhy, υποστηρίζει τη μοντελοποίηση μέσω της κλάσης AntennaModel του ns-3. Κάθε μοντέλο που βασίζεται σε αυτή την κλάση μπορεί να συσχετιστεί τόσο με eNB όσο και με UE. Το προκαθορισμένο μοντέλο που χρησιμοποιούν τα eNB και UE είναι το IsotropicAntennaModel.

3.4. Διεπαφή X2 και υποστήριξη μεταπομπών

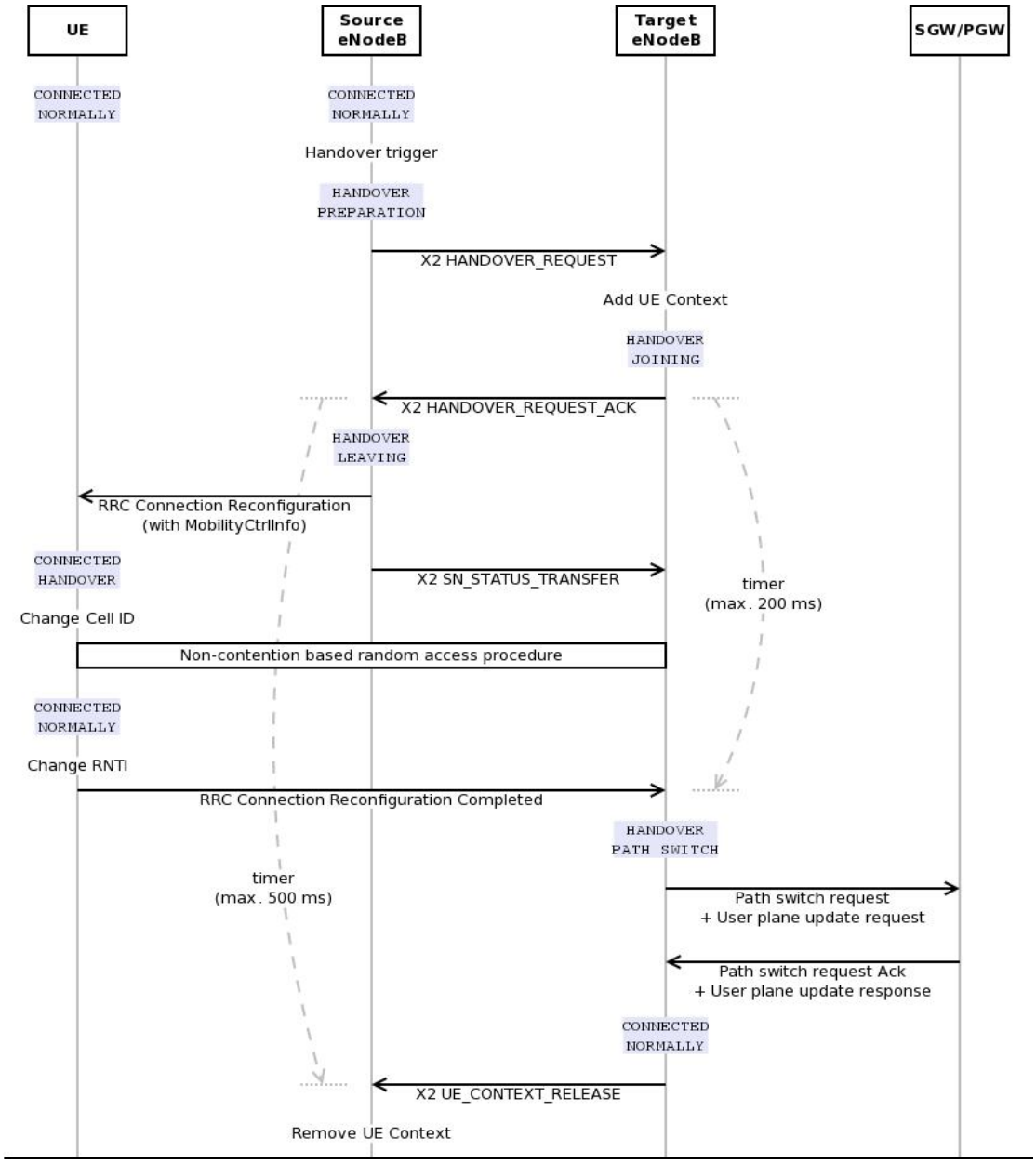
Η διεπαφή X2 συνδέει τα eNB μεταξύ τους με συνδέσεις point-to-point. Το X2 παρέχει τις ακόλουθες διεργασίες όσον αφορά στη διαχείριση κινητικότητας που αφορά στη μεταπομπή (handover) X2.

- Handover Request
- Handover Request Acknowledgement
- Sequence Number (SN) Status Transfer
- UE Context Release

Προς το παρόν ο προσομοιωτής υποστηρίζει το seamless handover και όχι το lossless handover. Η βασική διαφορά ανάμεσα στους δύο αυτούς τύπους μεταπομπών είναι η ανοχή στις καθυστερήσεις (delay) και στις αναμεταδόσεις (retransmission) πακέτων. Το seamless handover είναι ευαίσθητο στις καθυστερήσεις. Αυτό σημαίνει πως στην περίπτωση μεγάλων καθυστερήσεων προτιμάται η αναμετάδοση ενός πακέτου. Από την άλλη πλευρά το lossless handover είναι ευαίσθητο στις αναμεταδόσεις, που σημαίνει πως έχει μεγαλύτερη ανοχή στις καθυστερήσεις. Ο τρόπος μεταπομπής που επιλέγεται λοιπόν από το eNB εξαρτάται από τον τύπο της κίνησης που εξυπηρετείται. Αν ο τύπος της κίνησης είναι VoIP (μικρά σε μέγεθος πακέτα) προτιμάται το seamless handover, μια και η αναμετάδοση των πακέτων δε θα δημιουργήσει μεγάλη αύξηση στην κίνηση. Στην περίπτωση κίνησης τύπου HTTP για παράδειγμα (μεγάλα σε μέγεθος πακέτα) προτιμάται το lossless handover.

Στην παρακάτω εικόνα παρουσιάζεται ο τρόπος εκτέλεσης ενός handover στο μοντέλο. Οι ετικέτες με επισήμανση αναφέρονται σε αλλαγές κατάστασης RRC των UE και eNB. Επίσης υπάρχουν δύο μετρητές (timers). Ο *handover leaving timer* ελέγχεται από το αρχικό eNB. Ο *handover joining timer* ελέγχεται από το τελικό

eNB. Ο πρώτος μπορεί να καθοριστεί από το HandoverLeavingTimeoutDuration και ο δεύτερος από το HandoverJoiningTimeoutDuration όπου είναι χαρακτηριστικά του LteEnbRrc. Όταν ένας από τους δύο μετρητές μηδενιστεί το handover αποτυγχάνει.



Σχήμα 3.8: Διάγραμμα ακολουθίας μεταπομπής X2 (X2-based handover) [26]

3.5. Παράδειγμα μεταπομπής βασισμένης στο X2 (X2-based handover)

Το παράδειγμα που ακολουθεί βρίσκεται υλοποιημένο στον προσομοιωτή ns-3 και εκτελεί μια μεταπομπή ενός UE από το eNB που είναι συνδεδεμένο σε ένα άλλο eNB [27]. Οι βασικές παράμετροι της προσομοίωσης είναι οι ακόλουθες:

- Η διάρκεια της προσομοίωσης είναι 300 msec (σειρά 141).
- Η απόσταση μεταξύ των eNB είναι 100 μέτρα (σειρά 142).
- Έχουν απενεργοποιηθεί οι αυτόματες μεταπομπές (σειρά 163).
- Έχει οριστεί να γίνει μεταπομπή στα 100 msec (σειρά 304)

Ακολουθεί ο κώδικας του σεναρίου:

```
1  /* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
2  /*
3   * Copyright (c) 2012 Centre Tecnologic de Telecomunicacions de Catalunya (CTTC)
4   *
5   * This program is free software; you can redistribute it and/or modify
6   * it under the terms of the GNU General Public License version 2 as
7   * published by the Free Software Foundation;
8   *
9   * This program is distributed in the hope that it will be useful,
10  * but WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  * GNU General Public License for more details.
13  *
14  * You should have received a copy of the GNU General Public License
15  * along with this program; if not, write to the Free Software
16  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
17  *
18  * Author: Manuel Requena <manuel.requena@cttc.es>
19  */
20
21 #include "ns3/core-module.h"
22 #include "ns3/network-module.h"
23 #include "ns3/internet-module.h"
24 #include "ns3/mobility-module.h"
25 #include "ns3/lte-module.h"
26 #include "ns3/applications-module.h"
27 #include "ns3/point-to-point-module.h"
28 #include "ns3/config-store-module.h"
29
30 using namespace ns3;
31
32 NS_LOG_COMPONENT_DEFINE ("LenaX2HandoverExample");
33
34 void
35 NotifyConnectionEstablishedUe (std::string context,
36                                uint64_t imsi,
37                                uint16_t cellid,
38                                uint16_t rnti)
39 {
40   std::cout << Simulator::Now ().GetSeconds () << " " << context
41             << " UE IMSI " << imsi
42             << ": connected to CellId " << cellid
43             << " with RNTI " << rnti
44             << std::endl;
45 }
46
47 void
48 NotifyHandoverStartUe (std::string context,
49                        uint64_t imsi,
50                        uint16_t cellid,
51                        uint16_t rnti,
52                        uint16_t targetCellId)
53 {
54   std::cout << Simulator::Now ().GetSeconds () << " " << context
55             << " UE IMSI " << imsi
56             << ": previously connected to CellId " << cellid
57             << " with RNTI " << rnti
58             << ", doing handover to CellId " << targetCellId
```

```
59            << std::endl;
60 }
61
62 void
63 NotifyHandoverEndOkUe (std::string context,
64                        uint64_t imsi,
65                        uint16_t cellid,
66                        uint16_t rnti)
67 {
68   std::cout << Simulator::Now ().GetSeconds () << " " << context
69             << " UE IMSI " << imsi
70             << ": successful handover to CellId " << cellid
71             << " with RNTI " << rnti
72             << std::endl;
73 }
74
75 void
76 NotifyConnectionEstablishedEnb (std::string context,
77                                 uint64_t imsi,
78                                 uint16_t cellid,
79                                 uint16_t rnti)
80 {
81   std::cout << Simulator::Now ().GetSeconds () << " " << context
82             << " eNB CellId " << cellid
83             << ": successful connection of UE with IMSI " << imsi
84             << " RNTI " << rnti
85             << std::endl;
86 }
87
88 void
89 NotifyHandoverStartEnb (std::string context,
90                         uint64_t imsi,
91                         uint16_t cellid,
92                         uint16_t rnti,
93                         uint16_t targetCellId)
94 {
95   std::cout << Simulator::Now ().GetSeconds () << " " << context
96             << " eNB CellId " << cellid
97             << ": start handover of UE with IMSI " << imsi
98             << " RNTI " << rnti
99             << " to CellId " << targetCellId
100            << std::endl;
101 }
102
103 void
104 NotifyHandoverEndOkEnb (std::string context,
105                         uint64_t imsi,
106                         uint16_t cellid,
107                         uint16_t rnti)
108 {
109   std::cout << Simulator::Now ().GetSeconds () << " " << context
110             << " eNB CellId " << cellid
111             << ": completed handover of UE with IMSI " << imsi
112             << " RNTI " << rnti
113             << std::endl;
114 }
115
116
122 int
123 main (int argc, char *argv[])
124 {
125   // LogLevel logLevel = (LogLevel)(LOG_PREFIX_FUNC | LOG_PREFIX_TIME | LOG_LEVEL_ALL);
126
127   // LogComponentEnable ("LteHelper", logLevel);
128   // LogComponentEnable ("EpcHelper", logLevel);
129   // LogComponentEnable ("EpcEnbApplication", logLevel);
130   // LogComponentEnable ("EpcX2", logLevel);
131   // LogComponentEnable ("EpcSgwPgwApplication", logLevel);
132
133   // LogComponentEnable ("LteEnbRrc", logLevel);
134   // LogComponentEnable ("LteEnbNetDevice", logLevel);
135   // LogComponentEnable ("LteUeRrc", logLevel);
136   // LogComponentEnable ("LteUeNetDevice", logLevel);
137
138   uint16_t numberOfUes = 1;
139   uint16_t numberOfEnbs = 2;
140   uint16_t numBearersPerUe = 2;
141   double simTime = 0.300;
142   double distance = 100.0;
143
144   // change some default attributes so that they are reasonable for
145   // this scenario, but do this before processing command line
146   // arguments, so that the user is allowed to override these settings
147   Config::SetDefault ("ns3::UdpClient::Interval", TimeValue (MilliSeconds (10)));
148   Config::SetDefault ("ns3::UdpClient::MaxPackets", UintegerValue (1000000));
149   Config::SetDefault ("ns3::LteHelper::UseIdealRrc", BooleanValue (false));
150
151   // Command line arguments
```

```
152   CommandLine cmd;
153   cmd.AddValue ("numberOfUes", "Number of UEs", numberOfUes);
154   cmd.AddValue ("numberOfEnbs", "Number of eNodeBs", numberOfEnbs);
155   cmd.AddValue ("simTime", "Total duration of the simulation (in seconds)", simTime);
156   cmd.Parse (argc, argv);
157
158
159   Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
160   Ptr<PointToPointEpcHelper> epcHelper = CreateObject<PointToPointEpcHelper> ();
161   lteHelper->SetEpcHelper (epcHelper);
162   lteHelper->SetSchedulerType ("ns3::RrFfMacScheduler");
163   lteHelper->SetHandoverAlgorithmType ("ns3::NoOpHandoverAlgorithm");  // no handover algorithm
164
165   Ptr<Node> pgw = epcHelper->GetPgwNode ();
166
167   // Create a single RemoteHost
168   NodeContainer remoteHostContainer;
169   remoteHostContainer.Create (1);
170   Ptr<Node> remoteHost = remoteHostContainer.Get (0);
171   InternetStackHelper internet;
172   internet.Install (remoteHostContainer);
173
174   // Create the Internet
175   PointToPointHelper p2ph;
176   p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
177   p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
178   p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
179   NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
180   Ipv4AddressHelper ipv4h;
181   ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
182   Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
183   Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);
184
185
186   // Routing of the Internet Host (towards the LTE network)
187   Ipv4StaticRoutingHelper ipv4RoutingHelper;
188   Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
189   // interface 0 is localhost, 1 is the p2p device
190   remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);
191
192   NodeContainer ueNodes;
193   NodeContainer enbNodes;
194   enbNodes.Create (numberOfEnbs);
195   ueNodes.Create (numberOfUes);
196
197   // Install Mobility Model
198   Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
199   for (uint16_t i = 0; i < numberOfEnbs; i++)
200     {
201       positionAlloc->Add (Vector (distance * 2 * i - distance, 0, 0));
202     }
203   for (uint16_t i = 0; i < numberOfUes; i++)
204     {
205       positionAlloc->Add (Vector (0, 0, 0));
206     }
207   MobilityHelper mobility;
208   mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
209   mobility.SetPositionAllocator (positionAlloc);
210   mobility.Install (enbNodes);
211   mobility.Install (ueNodes);
212
213   // Install LTE Devices in eNB and UEs
214   NetDeviceContainer enbLteDevs = lteHelper->InstallEnbDevice (enbNodes);
215   NetDeviceContainer ueLteDevs = lteHelper->InstallUeDevice (ueNodes);
216
217   // Install the IP stack on the UEs
218   internet.Install (ueNodes);
219   Ipv4InterfaceContainer ueIpIfaces;
220   ueIpIfaces = epcHelper->AssignUeIpv4Address (NetDeviceContainer (ueLteDevs));
221   // Assign IP address to UEs, and install applications
222   for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
223     {
224       Ptr<Node> ueNode = ueNodes.Get (u);
225       // Set the default gateway for the UE
226       Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ueNode->GetObject<Ipv4> ());
227       ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
228     }
229
230
231   // Attach all UEs to the first eNodeB
232   for (uint16_t i = 0; i < numberOfUes; i++)
233     {
234       lteHelper->Attach (ueLteDevs.Get (i), enbLteDevs.Get (0));
235     }
236
237
238   NS_LOG_LOGIC ("setting up applications");
239
```

```
240   // Install and start applications on UEs and remote host
241   uint16_t dlPort = 10000;
242   uint16_t ulPort = 20000;
243
244   // randomize a bit start times to avoid simulation artifacts
245   // (e.g., buffer overflows due to packet transmissions happening
246   // exactly at the same time)
247   Ptr<UniformRandomVariable> startTimeSeconds = CreateObject<UniformRandomVariable> ();
248   startTimeSeconds->SetAttribute ("Min", DoubleValue (0));
249   startTimeSeconds->SetAttribute ("Max", DoubleValue (0.010));
250
251   for (uint32_t u = 0; u < numberOfUes; ++u)
252     {
253       Ptr<Node> ue = ueNodes.Get (u);
254       // Set the default gateway for the UE
255       Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ue->GetObject<Ipv4> ());
256       ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
257
258       for (uint32_t b = 0; b < numBearersPerUe; ++b)
259         {
260           ++dlPort;
261           ++ulPort;
262
263           ApplicationContainer clientApps;
264           ApplicationContainer serverApps;
265
266           NS_LOG_LOGIC ("installing UDP DL app for UE " << u);
267           UdpClientHelper dlClientHelper (ueIpIfaces.GetAddress (u), dlPort);
268           clientApps.Add (dlClientHelper.Install (remoteHost));
269           PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory",
270                                                InetSocketAddress (Ipv4Address::GetAny (), dlPort));
271           serverApps.Add (dlPacketSinkHelper.Install (ue));
272
273           NS_LOG_LOGIC ("installing UDP UL app for UE " << u);
274           UdpClientHelper ulClientHelper (remoteHostAddr, ulPort);
275           clientApps.Add (ulClientHelper.Install (ue));
276           PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory",
277                                                InetSocketAddress (Ipv4Address::GetAny (), ulPort));
278           serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
279
280           Ptr<EpcTft> tft = Create<EpcTft> ();
281           EpcTft::PacketFilter dlpf;
282           dlpf.localPortStart = dlPort;
283           dlpf.localPortEnd = dlPort;
284           tft->Add (dlpf);
285           EpcTft::PacketFilter ulpf;
286           ulpf.remotePortStart = ulPort;
287           ulpf.remotePortEnd = ulPort;
288           tft->Add (ulpf);
289           EpsBearer bearer (EpsBearer::NGBR_VIDEO_TCP_DEFAULT);
290           lteHelper->ActivateDedicatedEpsBearer (ueLteDevs.Get (u), bearer, tft);
291
292           Time startTime = Seconds (startTimeSeconds->GetValue ());
293           serverApps.Start (startTime);
294           clientApps.Start (startTime);
295
296         } // end for b
297     }
298
299
300   // Add X2 inteface
301   lteHelper->AddX2Interface (enbNodes);
302
303   // X2-based Handover
304   lteHelper->HandoverRequest (Seconds (0.100), ueLteDevs.Get (0), enbLteDevs.Get (0), enbLteDevs.Get (1));
305
306   // Uncomment to enable PCAP tracing
307   //p2ph.EnablePcapAll("lena-x2-handover");
308
309   lteHelper->EnablePhyTraces ();
310   lteHelper->EnableMacTraces ();
311   lteHelper->EnableRlcTraces ();
312   lteHelper->EnablePdcpTraces ();
313   Ptr<RadioBearerStatsCalculator> rlcStats = lteHelper->GetRlcStats ();
314   rlcStats->SetAttribute ("EpochDuration", TimeValue (Seconds (0.05)));
315   Ptr<RadioBearerStatsCalculator> pdcpStats = lteHelper->GetPdcpStats ();
316   pdcpStats->SetAttribute ("EpochDuration", TimeValue (Seconds (0.05)));
317
318
319   // connect custom trace sinks for RRC connection establishment and handover notification
320   Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/ConnectionEstablished",
321                    MakeCallback (&NotifyConnectionEstablishedEnb));
322   Config::Connect ("/NodeList/*/DeviceList/*/LteUeRrc/ConnectionEstablished",
323                    MakeCallback (&NotifyConnectionEstablishedUe));
324   Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/HandoverStart",
325                    MakeCallback (&NotifyHandoverStartEnb));
326   Config::Connect ("/NodeList/*/DeviceList/*/LteUeRrc/HandoverStart",
327                    MakeCallback (&NotifyHandoverStartUe));
```

```
328   Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/HandoverEndOk",
329                    MakeCallback (&NotifyHandoverEndOkEnb));
330   Config::Connect ("/NodeList/*/DeviceList/*/LteUeRrc/HandoverEndOk",
331                    MakeCallback (&NotifyHandoverEndOkUe));
332
333
334   Simulator::Stop (Seconds (simTime));
335   Simulator::Run ();
336
337   // GtkConfigStore config;
338   // config.ConfigureAttributes ();
339
340   Simulator::Destroy ();
341   return 0;
342
343 }
```

Κεφάλαιο 4ο: Επέκταση του συστήματος LENA για την υποστήριξη μεταπομπών σε κεραίες διαφορετικών συχνοτήτων

4.1. Περιγραφή και λύση του προβλήματος

Η έκδοση του module LTE (LENA project) που είναι ενσωματωμένο στον προσομοιωτή ns-3 είναι βασισμένη στην έκδοση 8 της 3GPP. Αυτό έχει ως συνέπεια τα κινητά (UE) να μην μπορούν να λάβουν μετρήσεις από σταθμούς βάσης (eNB ή HeNB) που λειτουργούν σε διαφορετικές συχνότητες, άρα να μην είναι εφικτές οι μεταπομπές των κινητών τερματικών ανάμεσά τους.

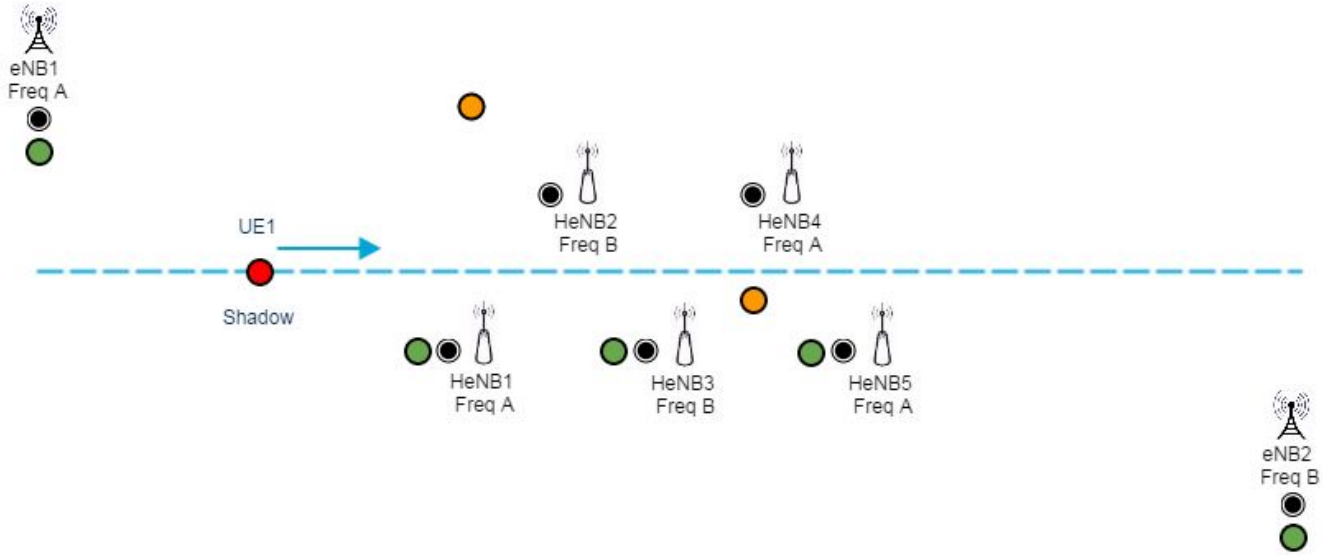
Ο βασικός λόγος που επιθυμούμε τέτοιες μεταπομπές είναι η χρήση, εκ μέρους του κινητού, του σταθμού βάσης που θα του παράσχει την καλύτερη ποιότητα σήματος. Ένας ακόμα λόγος είναι και η αποφόρτιση του δικτύου [1]. Μια τέτοια περίπτωση είναι ένας σταθμός βάσης που εξυπηρετεί μεγάλο αριθμό κινητών. Κάποια από αυτά θα μπορούσαν να εξυπηρετηθούν από ένα κοντινό femtocell με αποτέλεσμα την καλύτερη κατανομή των πόρων του συστήματος. Χαρακτηριστικό παράδειγμα είναι ένα εμπορικό κέντρο όπου υπάρχει μεγάλη συγκέντρωση χρηστών. Ένα μέρος αυτών θα μπορούσε να εξυπηρετηθεί από μερικές μικρές κυψέλες (femtocell) που θα ήταν τοποθετημένες κατάλληλα σε διάφορα σημεία του εμπορικού κέντρου.

Προκειμένου να ξεπεραστεί το εμπόδιο που προαναφέρθηκε προτείνουμε τη χρήση ενός δεύτερου κινητού τερματικού που θα «κλειδώνει» σε μια συχνότητα διαφορετική από το πρώτο, που είναι και το βασικό αντικείμενο της προσομοίωσης. Το κινητό αυτό θα βρίσκεται πάντα στο ίδιο σημείο με το κινητό που μελετάται ώστε να χρησιμεύει ως γέφυρα ανάμεσα στις κυψέλες που λειτουργούν σε διαφορετικές συχνότητες. Λόγο της φύσης του δεύτερου κινητού ο νέος μηχανισμός μεταπομπών που προτείνουμε ονομάζεται «τερματικό σκιά» (shadow terminal).

4.2. Συνοπτική παρουσίαση της τοπολογίας της προσομοίωσης

Η τοπολογία του σεναρίου στο οποίο βασίζεται ο κώδικας του προσομοιωτή ns-3 αποτελείται από επτά σταθμούς βάσης, εκ των οποίων οι δύο αντιστοιχούν σε eNB

και οι υπόλοιποι πέντε σε HeNB. Επίσης χρησιμοποιούνται δέκα κινητά τερματικά. Μια αναπαράσταση της τοπολογίας παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 4.1: Η τοπολογία του σεναρίου προσομοίωσης στο ns-3

Ο πλήρης κώδικας που αναπαριστά το σενάριο και χρησιμοποιήθηκε στα πειράματα που εκτελέστηκαν βρίσκεται στο παράρτημα της παρούσας εργασίας.

4.3. Αναλυτική παρουσίαση της τοπολογίας της προσομοίωσης

4.3.1. Γενική εικόνα του συστήματος

Το σύστημα χρησιμοποιεί Frequency Division Duplex (FDD). Αυτό σημαίνει πως ο πομπός και ο δέκτης λειτουργούν σε διαφορετικές συχνότητες. Άλλη για την άνω ζεύξη (uplink) και άλλη για την κάτω ζεύξη (downlink). Ανάμεσά τους υπάρχει ένα guard band για την αποφυγή παρεμβολών.

Το σενάριο περιλαμβάνει δέκα κινητά τερματικά (UE). Δύο από αυτά είναι τα βασικά αντικείμενα της παρούσας εργασίας. Το πρώτο (UE1) είναι αυτό που μελετούμε τη συμπεριφορά του στο σύστημα (ρυθμαπόδοση και καθυστέρηση στην άνω και στην κάτω ζεύξη, μεταπομπές μεταξύ κεραιών ίδιων αλλά και διαφορετικών συχνοτήτων). Το δεύτερο (shadow terminal) είναι αυτό που παρέχει τις κατάλληλες μετρήσεις ώστε

να λειτουργήσει ο μηχανισμός μεταπομπών, για το UE1, μεταξύ διαφορετικών συχνοτήτων. Μαζί με τα UE1 και shadow terminal υπάρχει και ένα ακόμα UE που ο ρόλος του είναι κινείται μαζί τους και να προσφέρει κίνηση στο σύστημα. Και τα τρία αυτά κινητά βρίσκονται στο κόκκινο σημείο του σχήματος 5.1. Τα υπόλοιπα κινητά τερματικά προσφέρουν κίνηση στο σύστημα και είναι τοποθετημένα είτε σε σταθερά σημεία (πέντε πράσινα σημεία στο σχήμα 5.1) είτε σε τυχαία σημεία κινούμενα επίσης με τυχαίο τρόπο όσον αφορά στην κατεύθυνση και στην ταχύτητα (δύο πορτοκαλί σημεία στο σχήμα 5.1).

4.3.2. Το EUTRAN

Όπως έχει προαναφερθεί η μοναδική οντότητα στο air interface του συστήματος είναι ο σταθμός βάσης. Στην τοπολογία της προσομοίωσης που περιέχεται στην εργασία υπάρχουν δύο ειδών σταθμοί βάσης, eNB και HeNB, οι οποίοι λειτουργούν σε δύο διαφορετικές συχνοτικές περιοχές. Οι περιοχές αυτές είναι η A (Freq A) και η B (Freq B). Η συχνότητα A αντιστοιχεί στα 1930MHz για την άνω ζεύξη και στα 2120MHz για την κάτω ζεύξη (Band 1) [4]. Η συχνότητα B αντιστοιχεί στα 1890MHz για την άνω ζεύξη και στα 1970MHz για την κάτω ζεύξη (Band 2) [4].

4.3.2.1. Τα eNB

Στο σύστημα υπάρχουν δύο eNB. Πρόκειται για δύο ισοτροπικές κεραίες με ισχύ εκπομπής 40dBm. Το eNB1 λειτουργεί στη συχνότητα A ενώ το eNB2 στη συχνότητα B. Το εύρος ζώνης του καναλιού (channel bandwidth) είναι 10MHz. Στον κώδικα ορίζουμε το πλήθος των resource block για τα eNB σε 50 πράγμα που σημαίνει (παράγραφος 2.5.3.2) πως το υπολογιζόμενο εύρος ζώνης καναλιού είναι 50x180kHz=9MHz. Σε αυτό τον αριθμό αν προσθέσουμε και το guard band (διαχωρίζει το uplink από το downlink) που είναι περίπου ίσο με 10% έχουμε συνολικά τα 10MHz του εύρους ζώνης του καναλιού.

4.3.2.2. Τα HeNB

Είναι πέντε ισοτροπικές κεραίες που δημιουργούν ένα πλέγμα ανάμεσα στα δύο eNB, όπως φαίνεται στο σχήμα 5.1. Η ισχύς εκπομπής τους είναι 23dBm. Το εύρος ζώνης

του καναλιού είναι 3MHz τα οποία προκύπτουν από τα διαθέσιμα 15 resource block που ορίζουμε στον κώδικα σύμφωνα με όσα προαναφέρθηκαν για τα eNB.

4.3.3. Το EPC

Ο προσομοιωτής ns-3 παρέχει μια απλουστευμένη μορφή αναπαράστασης για το EPC. Η μόνη οντότητα που αντιστοιχεί στο Evolved Packet Core είναι ο συνδυασμός ενός Serving Gateway (S-GW) και ενός Packet Data Network Gateway (PDN-GW). Αυτή η οντότητα επικοινωνεί με τους σταθμούς βάσης μέσω S1-U διεπαφών. Η σύνδεσή της με εξωτερικά συστήματα (internet) επιτυγχάνεται μέσω μιας διεπαφής SGi.

4.4. Περιγραφή της λειτουργίας του συστήματος προσομοίωσης

Με την έναρξη του προγράμματος δημιουργούνται όλα τα αντικείμενα που αποτελούν το σύστημα. Τα δύο βασικά κινητά (UE και shadow terminal) ξεκινούν την κίνησή τους προς το πλέγμα των HeNB. Αρχικά το UE είναι συνδεδεμένο με το eNB1 (Freq A) και το shadow terminal με το HeNB2 (Freq B). Αυτό σημαίνει πως το UE λαμβάνει μετρήσεις σχετικές με την ποιότητα του καναλιού που αφορούν στους σταθμούς βάσης που λειτουργούν στη συχνότητα A (eNB1, HeNB1, HeNB4, HeNB5) ενώ το shadow terminal σχετικές με τους σταθμούς βάσης που λειτουργούν στη συχνότητα B (eNB2, HeNB2, HeNB3).

Ο βασικός αλγόριθμος μεταπομπών που χρησιμοποιείται είναι ο A2A4. Αυτός συγκρίνει τις τιμές RSRQ (Reference Signal Received Quality) που λαμβάνει κάθε κινητό από όλους τους σταθμούς βάσης της ίδιας με αυτό συχνότητας. Ο A2A4 δίνει την εντολή για εκτέλεση μεταπομπής του κινητού όταν η τιμή RSRQ, που του παρέχει ο σταθμός βάσης που το εξυπηρετεί, γίνει μικρότερη από μία τιμή και ταυτόχρονα η αντίστοιχη τιμή RSRQ που αντιστοιχεί σε ένα διπλανό σταθμό βάσης είναι μεγαλύτερη από μια δεύτερη τιμή.

Κατά την εκκίνηση της προσομοίωσης κάθε κινητό συνδέεται με ένα σταθμό βάσης. Το ns-3 δεν παρέχει εγγενώς τη δυνατότητα στα κινητά τερματικά να λαμβάνουν μετρήσεις από σταθμούς βάσης που λειτουργούν σε διαφορετικές συχνότητες. Το UE1 συνδέεται με το eNB1 που λειτουργεί στη συχνότητα A. Αυτό έχει ως συνέπεια

να μη συμβούν μεταπομπές κατά την κίνησή του προς σταθμούς βάσης που λειτουργούν στη συχνότητα B, ακόμα και αν η ποιότητα του καναλιού προς αυτούς είναι καλύτερη. Στο σημείο αυτό γίνεται απαραίτητη η ύπαρξη του shadow terminal, το οποίο όντας συνδεδεμένο σε ένα HeNB διαφορετικής συχνότητας (HeNB2), λαμβάνει μετρήσεις στη συχνότητα B. Προκειμένου όμως οι μετρήσεις αυτές να είναι αξιοποιήσιμες από το UE1 υλοποιήθηκε ένας μηχανισμός σύγκρισης των τιμών RSRQ κάθε χρονική στιγμή που λαμβάνονται από τα UE1 και shadow terminal. Στην περίπτωση που η ποιότητα του καναλιού γίνει καλύτερη προς ένα σταθμό βάσης που λειτουργεί σε μια διαφορετική συχνότητα, δίνεται εντολή μεταπομπής στο UE1. Αυτός ο μηχανισμός αποτελείται από δύο μέρη. Το πρώτο συγκρίνει τις τιμές RSRQ των UE και shadow terminal κάθε χρονική στιγμή που αυτές λαμβάνονται. Μόλις το κριτήριο που έχει δοθεί γίνει αληθές (ποιότητα καναλιού της συχνότητας που λειτουργεί το shadow terminal είναι καλύτερη από την αντίστοιχη ποιότητα της συχνότητας που λειτουργεί το UE1) καλείται το δεύτερο μέρος που υλοποιεί τις μεταπομπές. Αναφερόμαστε σε μεταπομπές και όχι σε μεταπομπή. Από τη μια πλευρά το UE1 μεταφέρεται από το σταθμό βάσης με συχνότητα A στο σταθμό βάσης που βρίσκεται τη συγκεκριμένη στιγμή το shadow terminal (συχνότητα B). Ταυτόχρονα το «κινητό σκιά» μεταφέρεται στο σταθμό βάσης που ήταν συνδεδεμένο το UE1. Αυτή η αντιστροφή συνδέσεων που γίνεται είναι απαραίτητη διότι μπορεί και στο μέλλον να βρεθεί eNB ή HeNB που αν και λειτουργεί σε διαφορετική συχνότητα από το UE1 να έχει καλύτερη ποιότητα καναλιού και να χρειάζεται πάλι μεταπομπή μεταξύ κεραιών διαφορετικών συχνοτήτων.

Στο σενάριο της προσομοίωσης λειτουργούν ταυτόχρονα δύο διαφορετικοί μηχανισμοί μεταπομπών. Ο πρώτος είναι ο αλγόριθμος A2A4. Ο δεύτερος είναι το αντικείμενο της παρούσας εργασίας, δηλαδή η επέκταση με τη χρήση ενός κινητού τερματικού σκιά του ns-3 ώστε να υλοποιεί μεταπομπές μεταξύ κεραιών που λειτουργούν σε διαφορετικές συχνότητες.

Κεφάλαιο 5^{ο}: Αποτελέσματα της προσομοίωσης

Το κεφάλαιο αυτό αφορά στα αποτελέσματα των προσομοιώσεων που προέκυψαν από την εκτέλεση διαφορετικών σεναρίων με χρήση του προγράμματος ns-3. Το επιδιωκόμενο αποτέλεσμα είναι η αποτύπωση της λειτουργίας του νέου μηχανισμού διαχείρισης μεταπομπών που σχεδιάστηκε αλλά και η χρησιμότητά του όσον αφορά στη βελτιστοποίηση του παρεχόμενου σήματος προς το κινητό τερματικό με συνέπεια την καλυτέρευση της ποιότητας υπηρεσίας που του προσφέρεται από το δίκτυο του παρόχου.

Δύο είναι τα τμήματα που δομούν το συγκεκριμένο κεφάλαιο:

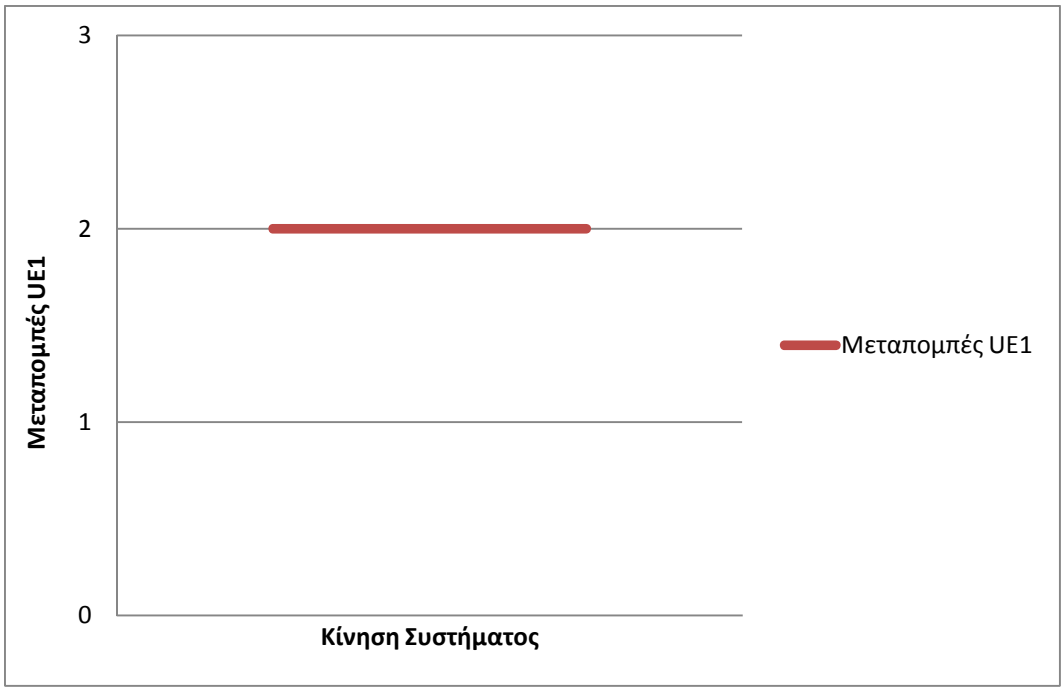
1. Στο πρώτο σενάριο δε χρησιμοποιείται ο προτεινόμενος μηχανισμός διαχείρισης μεταπομπών. Αυτό θα έχει ως αποτέλεσμα τη μη διεξαγωγή μεταπομπών του κινητού UE1 προς σταθμούς βάσης διαφορετικών συχνοτήτων.
2. Στο δεύτερο σενάριο χρησιμοποιείται ο νέος μηχανισμός μεταπομπών. Κατά συνέπεια αναμένονται περισσότερες μεταπομπές του κινητού UE1, διότι θα περιλαμβάνονται και αυτές που θα γίνονται ανάμεσα σε σταθμούς βάσης διαφορετικών συχνοτήτων.

Και στις δύο περιπτώσεις που προαναφέρονται χρησιμοποιείται η ίδια τοπολογία στον προσομοιωτή ns-3, όπως αυτή έχει αποτυπωθεί στην παράγραφο 4.3. Το μόνο που αλλάζει είναι πως στην πρώτη περίπτωση ο νέος μηχανισμός διαχείρισης μεταπομπών είναι απενεργοποιημένος.

Τα πειράματα, και στις δύο περιπτώσεις είναι δομημένα με τον ίδιο τρόπο. Εκτελούνται μια σειρά τριών πειραμάτων, πρώτα χωρίς το μηχανισμό και μετά με το μηχανισμό. Το κάθε ένα από αυτά τα τρία πειράματα διαφοροποιείται από το άλλο όσον αφορά στην κίνηση (traffic) που δίνεται στο σύστημα. Οι σταθμοί βάσης στους οποίους τροποποιείται η κίνηση είναι τα HeNB1, HeNB3 και HeNB5 (σχήμα 4.1). Η κίνηση αυτή προσφέρεται από τα σταθερά κινητά τερματικά που είναι τοποθετημένα σε κάθε έναν από αυτούς (πράσινα σημεία στο σχήμα 4.1). Η κίνηση έχει τρεις διαβαθμίσεις: μικρή, μεσαία και μεγάλη.

5.1. Προσομοιώσεις χωρίς τη χρήση του νέου μηχανισμού μεταπομπών

Στο παρακάτω γράφημα παρουσιάζεται το αποτέλεσμα που αφορά στον αριθμό των μεταπομπών του UE1 (handovers) στην περίπτωση που δε χρησιμοποιείται ο προτεινόμενος μηχανισμός διαχείρισης μεταπομπών στον προσομοιωτή ns-3. Ο αλγόριθμος που χρησιμοποιείται A2A4, ο οποίος δίνει την εντολή για την εκτέλεση μιας μεταπομπής όταν η στάθμη RSRQ του κινητού τερματικού UE1 στην κυψέλη που εξυπηρετείται γίνει μικρότερη από ένα όριο, καθώς επίσης και όταν η στάθμη RSRQ από ένα γειτονικό σταθμό βάσης ξεπεράσει ένα κατώφλι.



Σχήμα 5.1: Συνολικός αριθμός μεταπομπών UE1

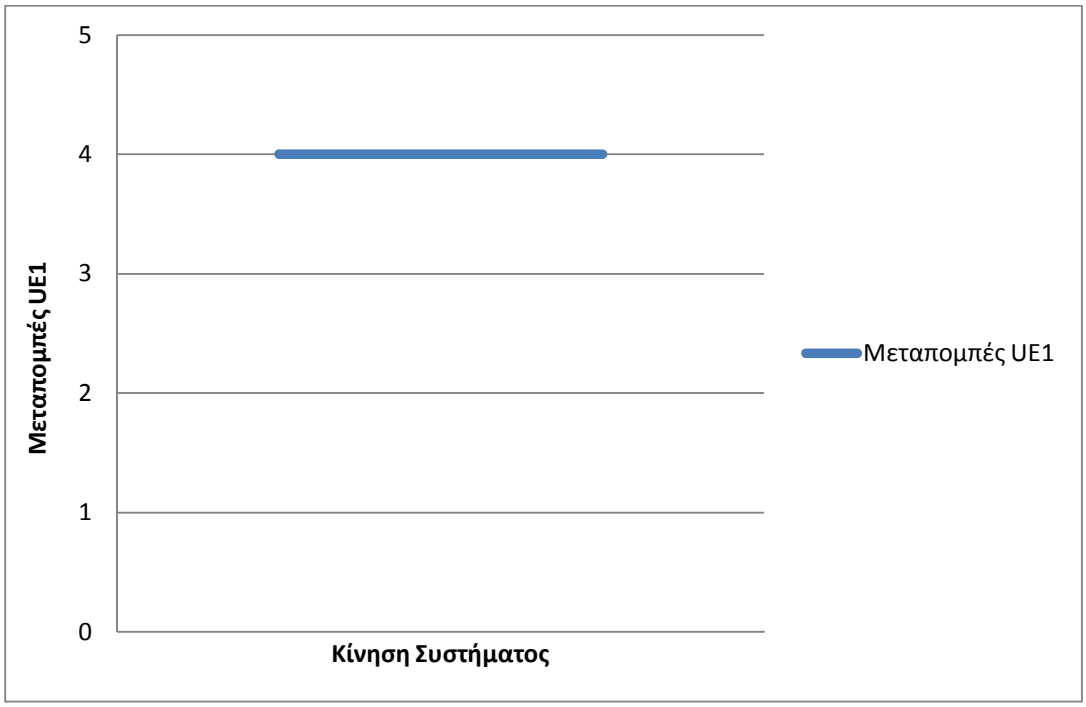
Ο αριθμός των μεταπομπών του UE1 με τη χρήση του αλγορίθμου A2A4 είναι ανεξάρτητος από την αύξηση του φορτίου στα HeNB1, HeNB3 και HeNB5 όπως αναμενόταν άλλωστε [28]. Ο πίνακας που ακολουθεί παρουσιάζει τις μεταπομπές που πραγματοποίησε το UE1:

Χρονική στιγμή (sec)	Αρχικός σταθμός βάσης	Τελικός σταθμός βάσης	Μηχανισμός μεταπομπής
52,03	eNB1	HeNB1	Αλγόριθμος A2A4
152,07	HeNB1	eNB1	Αλγόριθμος A2A4

Πίνακας 5.1: Μεταπομπές UE1 χωρίς το νέο μηχανισμό

5.2. Προσομοιώσεις με τη χρήση του νέου μηχανισμού μεταπομπών

Στην περίπτωση που χρησιμοποιηθεί ο προτεινόμενος μηχανισμός (shadow terminal) μαζί με τον αλγόριθμο A2A4, ο συνολικός αριθμός των μεταπομπών που πραγματοποιούνται αυξάνεται και ισούται πλέον με τέσσερις.



Σχήμα 5.2: Συνολικός αριθμός μεταπομπών UE1

Στον επόμενο πίνακα παρουσιάζονται αναλυτικότερα οι μεταπομπές του UE1:

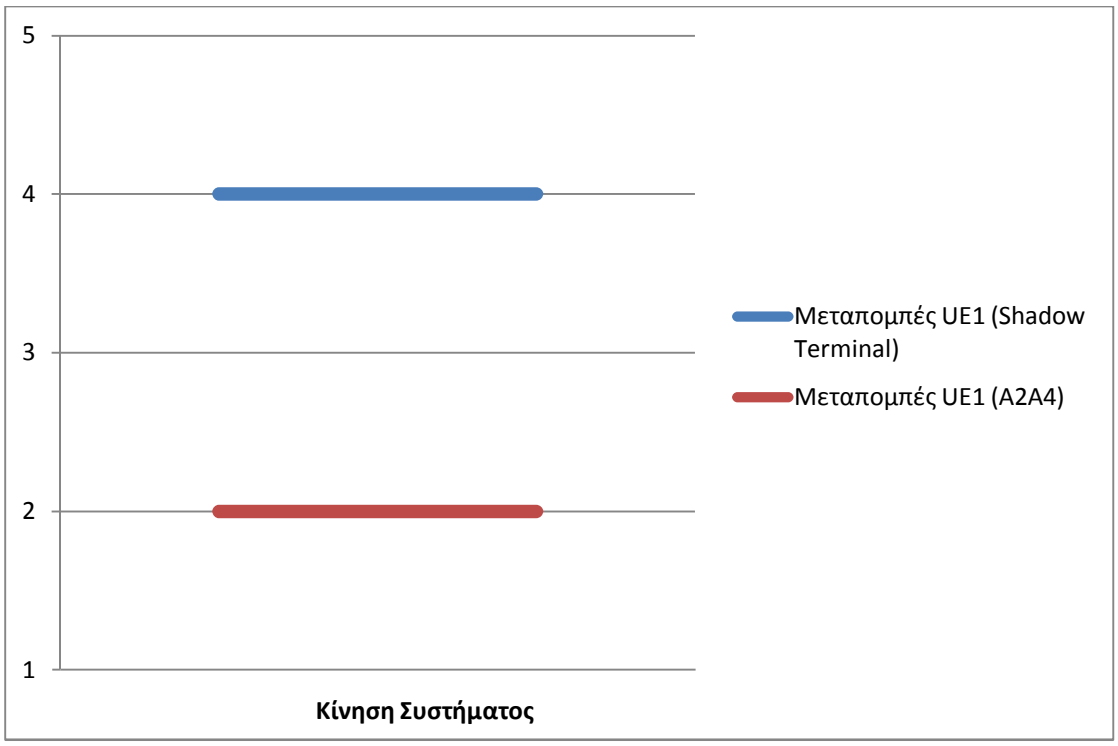
Χρονική στιγμή (sec)	Αρχικός σταθμός βάσης	Τελικός σταθμός βάσης	Μηχανισμός μεταπομπής
17,07	eNB1	HeNB2	Shadow Terminal
135,11	HeNB2	HeNB3	Αλγόριθμος A2A4
152,07	HeNB3	eNB1	Shadow Terminal
247,11	eNB1	eNB2	Shadow Terminal

Πίνακας 5.2: Μεταπομπές UE1 με τον προτεινόμενο μηχανισμό

5.3. Σύγκριση των αποτελεσμάτων

Στο σχήμα που ακολουθεί παρουσιάζεται η αλλαγή που επέρχεται στο συνολικό αριθμό μεταπομπών που πραγματοποιεί το UE1 καθ' όλη τη διάρκεια της προσομοίωσης και στις δύο περιπτώσεις, δηλαδή αρχικά με τη χρήση μόνο του

αλγορίθμου A2A4 και στη συνέχεια με την προσθήκη του νέου μηχανισμού διαχείρισης μεταπομπών (shadow terminal).



Σχήμα 5.11: Σύγκριση συνολικού αριθμού μεταπομπών του UE1

Τα αποτελέσματα εμφανίζουν μια αισθητή αύξηση του συνολικού αριθμού των μεταπομπών, για την ακρίβεια το διπλασιασμό τους. Αυτό φανερώνει πως με τη χρήση του νέου μηχανισμού διαχείρισης μεταπομπών το κινητό τερματικό UE1 είναι σε θέση να εκμεταλλευτεί περισσότερους σταθμούς βάσης, ώστε να επιλέξει κάθε φορά αυτόν που θα του προσφέρει καλύτερη ποιότητα σήματος (RSRQ) η οποία θα οδηγήσει στην καλύτερη εξυπηρέτησή του από το σύστημα.

Κεφάλαιο 6^{ο}: Συμπεράσματα

Ο προσομοιωτής δικτύων ns-3 έχει καθιερωθεί από τη διεθνή ακαδημαϊκή κοινότητα τόσο για εκπαιδευτικούς όσο και για ερευνητικούς σκοπούς. Λόγω του ότι ακολουθεί την πολιτική του ανοιχτού λογισμικού, έχει το πλεονέκτημα να επεκτείνεται με όλο και περισσότερες δυνατότητες από μια μεγάλη μερίδα τω χρηστών του.

Στον τομέα του LTE, μέσω του LENA project, έχει τοποθετηθεί στην κορυφή των προτιμήσεων των ερευνητών παγκοσμίως. Ένα πεδίο έρευνας με αλματώδη ανάπτυξη στις μέρες μας είναι η χρησιμοποίηση ετερογενών κυψελωτών δικτύων για την εξυπηρέτηση κινούμενων χρηστών.

Μια δυνατότητα του προσομοιωτή που έλειπε θεωρούμε πως ήταν η υποστήριξη μεταπομπών ανάμεσα σε σταθμούς βάσης διαφορετικών συχνοτήτων. Αυτές οι μεταπομπές είναι πολύ χρήσιμες κυρίως σε αστικά περιβάλλοντα όπου συνυπάρχουν κεραίες που λειτουργούν σε διαφορετικές συχνότητες (macrocells και picocells/femtocells). Αυτή η συνύπαρξη επιβάλλεται για την καλύτερη εξυπηρέτηση των χρηστών (πχ την αποφόρτιση του δικτύου με χρήση των femtocell). Η αδυναμία όμως του ns-3 να υποστηρίξει τέτοιες μεταπομπές καθιστούσε ανέφικτη την προσομοίωση αντίστοιχων λύσεων. Με την προσθήκη του μηχανισμού διαχείρισης μεταπομπών που προτείνουμε (χρήση τερματικού σκιά) γίνεται πλέον εφικτή η αξιοποίηση του ns-3 για τη μελέτη και αξιολόγηση αντίστοιχων λύσεων.

Παράρτημα: Ο κώδικας του σεναρίου προσομοίωσης

Ο πηγαίος κώδικας του σεναρίου που χρησιμοποιήθηκε για τα πειράματα των προσομοιώσεων είναι ο ακόλουθος:

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Model Author: Konstantinos N. Koutrakis, koutrakis@gmail.com
 *
 *
 */


#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/config-store-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/epc-x2-sap.h"
#include "ns3/lte-ue-net-device.h"
#include <ns3/Headers.h>
#include <sstream>
#include <fstream>
#include <iomanip>
#include <algorithm>
#include <list>
#include "ns3/random-variable-stream.h"
#include "ns3/random-walk-2d-mobility-model.h"


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SHADOW TERMINAL");

int current_IMSI;                   // Storing current UE RSRQ measurement
double rsrq1 = -100;                // Initial value of RSRQ for UE1
double rsrq2 = -100;                // Initial value of RSRQ for UE2
int HOstop = 1;                     // Allow only one HO for UE1 from eNB to HeNB
int handoverCalls = 0;              // Times myHandover was called
int handoversDone1 = 0;             // Times actual handover took place
uint16_t servingCellUe1 = 0;        // Serving cell of UE 1
uint16_t servingCellShadow = 0;     // Serving cell of shadow terminal
int handoverFailures = 0;
uint16_t global_numBearersPerUe;
uint16_t global_eNB_bandwidth;
uint16_t global_HeNB_Bandwidth;
static double UE_speed = 0.8;


/************************************************** Callback Funtions **********************************************/


void NotifyConnectionEstablishedUe (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti)
{
  std::cout << context
            << " UE IMSI " << imsi
            << ": connected to CellId " << cellid
            << " with RNTI " << rnti
            << std::endl;
}

void NotifyHandoverStartUe (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti, uint16_t targetCellId)
{
  std::cout << context
            << " UE IMSI " << imsi
            << ": previously connected to CellId " << cellid
            << " with RNTI " << rnti
```

```
            << ", doing handover to CellId " << targetCellId
            << std::endl;

////////////////////////////////// Increase overall handovers of UE1
  if (imsi == 1) {
       handoversDone1++;
  }
}

void NotifyHandoverEndOkUe (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti)
{
  std::cout << context
            << " UE IMSI " << imsi
            << ": successful handover to CellId " << cellid
            << " with RNTI " << rnti
            << std::endl;
}

void NotifyConnectionEstablishedEnb (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti)
{
  std::cout << context
            << " eNB CellId " << cellid
            << ": successful connection of UE with IMSI " << imsi
            << " RNTI " << rnti
            << std::endl;
}

void NotifyHandoverStartEnb (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti, uint16_t targetCellId)
{
  std::cout << context
            << " eNB CellId " << cellid
            << ": start handover of UE with IMSI " << imsi
            << " RNTI " << rnti
            << " to CellId " << targetCellId
            << std::endl;
}

void NotifyHandoverEndOkEnb (std::string context, uint64_t imsi, uint16_t cellid, uint16_t rnti)
{
  std::cout << context
            << " eNB CellId " << cellid
            << ": completed handover of UE with IMSI " << imsi
            << " RNTI " << rnti
            << std::endl;
}

////////////////////////////////// Transmission power of eNBs and HeNBs
static ns3::GlobalValue g_macroEnbTxPowerDbm ("macroEnbTxPowerDbm",
                                              "TX power [dBm] used by macro eNBs",
                                              ns3::DoubleValue (40.0),
                                              ns3::MakeDoubleChecker<double> ());
static ns3::GlobalValue g_homeEnbTxPowerDbm ("homeEnbTxPowerDbm",
                                             "TX power [dBm] used by HeNBs",
                                             ns3::DoubleValue (23.0),
                                             ns3::MakeDoubleChecker<double> ());

////////////////////////////////// Frequency bands and bandwidth of eNBs and HeNBs
static ns3::GlobalValue g_macroEnbDlEarfcn ("macroEnbDlEarfcn",
                                            "DL EARFCN used by macro eNBs",
                                            ns3::UintegerValue (100),
                                            ns3::MakeUintegerChecker<uint16_t> ());
static ns3::GlobalValue g_homeEnbDlEarfcn ("homeEnbDlEarfcn",
                                           "DL EARFCN used by HeNBs",
                                           ns3::UintegerValue (1000),
                                           ns3::MakeUintegerChecker<uint16_t> ());
static ns3::GlobalValue g_macroEnbBandwidth ("macroEnbBandwidth",
                                             "bandwidth [num RBs] used by macro eNBs",
                                             ns3::UintegerValue (50),
                                             ns3::MakeUintegerChecker<uint16_t> ());
static ns3::GlobalValue g_homeEnbBandwidth ("homeEnbBandwidth",
                                            "bandwidth [num RBs] used by HeNBs",
                                            ns3::UintegerValue (15),
                                            ns3::MakeUintegerChecker<uint16_t> ());

////////////////////////////////// Duration of simulation
static ns3::GlobalValue g_simTime ("simTime",
                                   "Total duration of the simulation [s]",
                                   ns3::DoubleValue (250.0),
                                   ns3::MakeDoubleChecker<double> ());

////////////////////////////////// Creation of Radio Environment Map (REM)
static ns3::GlobalValue g_generateRem ("generateRem",
                                       "if true, will generate a REM and then abort the simulation;"
                                       "if false, will run the simulation normally (without generating any REM)",
                                       ns3::BooleanValue (false),
                                       ns3::MakeBooleanChecker ());

////////////////////////////////// Creation of Evolved Packet Core (EPC) and parameters
```

```
static ns3::GlobalValue g_epc ("epc",
                               "If true, will setup the EPC to simulate an end-to-end topology, "
                               "with real IP applications over PDCP and RLC UM (or RLC AM by changing "
                               "the default value of EpsBearerToRlcMapping e.g. to RLC AM ALWAYS). "
                               "If false, only the LTE radio access will be simulated with RLC SM. ",
                               ns3::BooleanValue (true),
                               ns3::MakeBooleanChecker ());
static ns3::GlobalValue g_epcDl ("epcDl",
                                 "if true, will activate data flows in the downlink when EPC is being used. "
                                 "If false, downlink flows won't be activated. "
                                 "If EPC is not used, this parameter will be ignored.",
                                 ns3::BooleanValue (true),
                                 ns3::MakeBooleanChecker ());
static ns3::GlobalValue g_epcUl ("epcUl",
                                 "if true, will activate data flows in the uplink when EPC is being used. "
                                 "If false, uplink flows won't be activated. "
                                 "If EPC is not used, this parameter will be ignored.",
                                 ns3::BooleanValue (true),
                                 ns3::MakeBooleanChecker ());
static ns3::GlobalValue g_useUdp ("useUdp",
                                  "if true, the UdpClient application will be used. "
                                  "Otherwise, the BulkSend application will be used over a TCP connection. "
                                  "If EPC is not used, this parameter will be ignored.",
                                  ns3::BooleanValue (true),
                                  ns3::MakeBooleanChecker ());
static ns3::GlobalValue g_fadingTrace ("fadingTrace",
                                       "The path of the fading trace (by default no fading trace "
                                       "is loaded, i.e., fading is not considered)",
                                       ns3::StringValue (""),
                                       ns3::MakeStringChecker ());
static ns3::GlobalValue g_numBearersPerUe ("numBearersPerUe",
                                           "How many bearers per UE there are in the simulation",
                                           ns3::UintegerValue (1),
                                           ns3::MakeUintegerChecker<uint16_t> ());
static ns3::GlobalValue g_srsPeriodicity ("srsPeriodicity",
                                          "SRS Periodicity (has to be at least greater than the number of UEs per eNB)",
                                          ns3::UintegerValue (160),
                                          ns3::MakeUintegerChecker<uint16_t> ());


/**************************************************************************************************************************************************/
/************************************** MODEL DESCRIPTION ******************************************************************************************/
/**************************************************************************************************************************************************
 * 2 eNBs
 * 5 HeNBs
 * 10 UEs: UE1 = main terminal of the scenario
 *         UE2 = shadow terminal
 *         8 UEs which are generating traffic to the system
 *
 *
 * eNB1
 * freqA
 * (800,100)                                                <---- 20m ---->
 *                                                 HeNB2                  HeNB4
 *                                                 freqB                  freqA
 *
 *                                 UE1 ---->
 *                     + Shadow Terminal
 *                                 (960,0)
 *       <========== 200 m ===========>                                                <======= 400 m ======>
 *
 *                                       HeNB1                HeNB3                HeNB5
 *                                       freqA                freqB                freqA
 *                                             <---- 20m ---->       <---- 20m ---->                                  eNB2
 *                                                                                                                    freqB
 *                                                                                                                    (1500,-300)
 *
 *

**************************************************************************************************************************************************/


////////////////////////////////////// Creation of containers for eNBs, HeNBs and UEs
Ptr <LteHelper> lteHelper = CreateObject<LteHelper> ();
NodeContainer homeEnbs,macroEnbs,ues;
NetDeviceContainer macroEnbDevs,homeEnbDevs,ueDevs;

Ptr<LteEnbNetDevice> currentNetwork;
Ptr<LteEnbNetDevice> checkingNetwork;
Ptr<LteEnbNetDevice> theNetwork;
ns3::UeManager::State  thestate;

////////////////////////////////////// These variables are used for creating the callback paths
uint32_t n_testUEs;
uint32_t nBS;
```

```
////////////////////////////////////// The type of macrocell is 0 and femto is 1
int typeOfCell;

////////////////////////////////////// Output file for RSRQ logging
////////////////////////////////////// Output files for Throughput and Delay (UL & DL) logging
std::ofstream rsrqFile;
std::ofstream throughputFileUl;
std::ofstream throughputFileDl;

////////////////////////////////////// typeOfCell for eNBs is 0
Ptr<LteEnbNetDevice>  GetEnbDeviceFromCellId (uint16_t cellId) {
  for (uint16_t i=0;i<macroEnbDevs.GetN();i++) {
    Ptr<LteEnbNetDevice> tempDeviceObj = macroEnbDevs.Get (i)->GetObject<LteEnbNetDevice>();
    uint16_t i_cell_id  = tempDeviceObj-> GetCellId();
    if (i_cell_id == cellId) {
      theNetwork = tempDeviceObj;
      typeOfCell = 0;
    }
  }

////////////////////////////////////// typeOfCell for HeNBs is 1
  for (uint16_t i=0;i<homeEnbDevs.GetN();i++) {
    Ptr<LteEnbNetDevice> tempDeviceObj = homeEnbDevs.Get (i)->GetObject<LteEnbNetDevice>();
    uint16_t i_cell_id  = tempDeviceObj-> GetCellId();
    if (i_cell_id == cellId) {
      theNetwork = tempDeviceObj;
      typeOfCell = 1;
    }
  }
  return theNetwork;
}

void HandoverFailure (uint16_t cellId) {
  theNetwork = GetEnbDeviceFromCellId(cellId);
  Ptr<LteEnbRrc> enbRrcForLoad = theNetwork->GetObject<LteEnbNetDevice>()->GetRrc();
}

////////////////////////////////////// Variables for report and recieve measurements of UEs
int num_ReportUeMeasurementsCallback = 0;
int num_ReceiveMeasurementsCallback = 0;

////////////////////////////////////// Handover mechanism for manual HO triggering
void myHandover1 () {
       if (((servingCellUe1 == 1) || (servingCellUe1 == 2)) && (HOstop == 1)) {
                  lteHelper->DoHandoverRequest (ueDevs.Get (0), macroEnbDevs.Get (servingCellUe1-1), homeEnbDevs.Get
(servingCellShadow-3));
                  lteHelper->DoHandoverRequest (ueDevs.Get (1), homeEnbDevs.Get (servingCellShadow-3), macroEnbDevs.Get
(servingCellUe1-1));
                  HOstop++;
         }
   }

void myHandover2 () {
       if (((servingCellUe1 == 1) || (servingCellUe1 == 2)) && (HOstop == 1)) {
                  lteHelper->DoHandoverRequest (ueDevs.Get (0), macroEnbDevs.Get (servingCellUe1-1), macroEnbDevs.Get
(servingCellShadow-1));
                  lteHelper->DoHandoverRequest (ueDevs.Get (1), macroEnbDevs.Get (servingCellShadow-1), macroEnbDevs.Get
(servingCellUe1-1));
                  HOstop++;
         }
   }

void myHandover3 () {
              if (((servingCellUe1 == 3) || (servingCellUe1 == 4) || (servingCellUe1 == 5) ||
                            (servingCellUe1 == 6) || (servingCellUe1 == 7)) && (HOstop == 2)) {
                        lteHelper->DoHandoverRequest (ueDevs.Get (0), homeEnbDevs.Get (servingCellUe1-3), macroEnbDevs.Get
(servingCellShadow-1));
                        lteHelper->DoHandoverRequest (ueDevs.Get (1), macroEnbDevs.Get (servingCellShadow-1), homeEnbDevs.Get
(servingCellUe1-3));
                        HOstop = 1;
               }
   }

////////////////////////////////////// Compare RSRQ values of UE1 and Shadow
void rsrqComparison (double rsrq1, double rsrq2) {
     if (rsrq1 < rsrq2) {
              if ((servingCellUe1 == 1) || (servingCellUe1 == 2)) {
                       if ((servingCellShadow == 3) || (servingCellShadow == 4) || (servingCellShadow == 5) ||
                            (servingCellShadow == 6) || (servingCellShadow == 7)) Simulator::Schedule(Seconds(0),
&myHandover1);
                  else Simulator::Schedule(Seconds(0), &myHandover2);
              }
              if ((servingCellUe1 == 3) || (servingCellUe1 == 4) || (servingCellUe1 == 5) ||
                            (servingCellUe1 == 6) || (servingCellUe1 == 7)) Simulator::Schedule(Seconds(0), &myHandover3);
     }
}

void GetInfoForHOFailure (uint16_t rnti, uint16_t cellId, double rsrp, double rsrq, bool servingCell) {
```

```
  handoverCalls++;
  Ptr<LteUeNetDevice> ueLteDevice = ueDevs.Get (0)->GetObject<LteUeNetDevice>();
  Ptr<LteUeRrc> ueRrc = ueLteDevice->GetRrc();

////////////////////////////////////// Simulation logging on terminal
  std::cout << "\nSimulator::Now: " << Simulator::Now()/1000000000;
  std::cout << "\n************************************* ";
  std::cout << "\nRNTI of UE: " << rnti;
  std::cout << "\nExamined Cell ID: " << cellId;
  std::cout << "\nRSRQ: " << rsrq;
  std::cout << "\nHandovers Done Overall UE1: " << handoversDone1;
  std::cout << "\nHandover Failures: " << handoverFailures << "\n";
  Ptr<MobilityModel> mob1 = ues.Get(0)->GetObject<MobilityModel>();
  Vector pos1 = mob1->GetPosition ();
  std::cout << "\nUE1 POS: x=" << pos1.x << ", y=" << pos1.y << std::endl;
}

////////////////////////////////////// UE measurements
void ReportUeMeasurementsCallback (std::string path,
                                   uint16_t rnti, uint16_t cellId, double rsrp, double rsrq, bool servingCell)
  {
    num_ReportUeMeasurementsCallback++;
    GetInfoForHOFailure(rnti, cellId, rsrp, rsrq, servingCell);
    rsrqFile << Simulator::Now().GetSeconds() << "\t" << rnti << "\t" << cellId << "\t"<< rsrq << std::endl;

////////////////////////////////////// Simulation is stable
    if (Simulator::Now().GetSeconds() > 0.5) {

////////////////////////////////////// Set current_IMSI from the path
     if (nBS < 9) current_IMSI  = uint32_t(path[10]) - 47 - nBS;
         else {
                current_IMSI = ((uint32_t(path[10]) - 47) * 10) + (uint32_t(path[11]) -47);
                current_IMSI  = current_IMSI - nBS;
                std::cout << "current IMSI for 2 digit path of UEs: " << current_IMSI <<std::endl;
              }
     std::cout << "Current IMSI: " << current_IMSI << std::endl;
     }

////////////////////////////////////// Store RSRQ for UE1 (rsrq1) and Shadow (rsrq2)
    if ((current_IMSI == 1) && (servingCell == 1)) {
              rsrq1 = rsrq;
              servingCellUe1 = cellId;
              // std::cout << "TYPE OF CELL IS: " << typeOfCell << std::endl;
     }
    if ((current_IMSI == 2) && (servingCell == 1)) {
              rsrq2 = rsrq;
              servingCellShadow = cellId;
              // std::cout << "TYPE OF CELL IS: " << typeOfCell << std::endl;
     }

////////////////////////////////////// RSRQ comparison in order to trigger manual HO
    rsrqComparison (rsrq1, rsrq2);   // Enable or disable the new HO mechanism
  }

////////////////////////////////////// Callback for recieved measurements
void RecvMeasurementReportCallback (std::string path,
                                    uint64_t imsi, uint16_t cellId, uint16_t rnti, LteRrcSap::MeasurementReport meas)
{
  num_ReceiveMeasurementsCallback++;
}



/********************************************** MAIN ***************************************************************************/



int
main (int argc, char *argv[])
{
  Config::SetDefault ("ns3::UdpClient::Interval", TimeValue (MilliSeconds (1)));
  Config::SetDefault ("ns3::UdpClient::MaxPackets", UintegerValue (1000000));
  Config::SetDefault ("ns3::LteRlcUm::MaxTxBufferSize", UintegerValue (10 * 1024));
  Config::SetDefault ("ns3::LteHelper::UseIdealRrc", BooleanValue(false));

////////////////////////////////////// A way to avoid deterministic nature of simulator
////////////////////////////////////// NS_GLOBAL_VALUE="RngRun=2" ./waf --run shadow (RngRun=3, 4, 5...)
  RngSeedManager::SetSeed (3);

////////////////////////////////////// A2A4Rsrq Handover algorithm
  lteHelper->SetHandoverAlgorithmType ("ns3::A2A4RsrqHandoverAlgorithm");
  lteHelper->SetHandoverAlgorithmAttribute ("ServingCellThreshold",
                                            UintegerValue (30));
  lteHelper->SetHandoverAlgorithmAttribute ("NeighbourCellOffset",
                                            UintegerValue (1));
  CommandLine cmd;
  cmd.Parse (argc, argv);
  ConfigStore inputConfig;
```

```
  inputConfig.ConfigureDefaults ();

////////////////////////////////////// Parse again so you can override input file default values via command line
  cmd.Parse (argc, argv);

////////////////////////////////////// The scenario parameters get their values from the global attributes defined above
  UintegerValue uintegerValue;
  DoubleValue doubleValue;
  BooleanValue booleanValue;
  StringValue stringValue;
  GlobalValue::GetValueByName ("macroEnbTxPowerDbm", doubleValue);
  double macroEnbTxPowerDbm = doubleValue.Get ();
  GlobalValue::GetValueByName ("homeEnbTxPowerDbm", doubleValue);
  double homeEnbTxPowerDbm = doubleValue.Get ();
  GlobalValue::GetValueByName ("macroEnbDlEarfcn", uintegerValue);
  uint16_t macroEnbDlEarfcn = uintegerValue.Get ();
  GlobalValue::GetValueByName ("homeEnbDlEarfcn", uintegerValue);
  uint16_t homeEnbDlEarfcn = uintegerValue.Get ();
  GlobalValue::GetValueByName ("macroEnbBandwidth", uintegerValue);
  uint16_t macroEnbBandwidth = uintegerValue.Get ();
  global_eNB_bandwidth = macroEnbBandwidth;
  GlobalValue::GetValueByName ("homeEnbBandwidth", uintegerValue);
  uint16_t homeEnbBandwidth = uintegerValue.Get ();
  global_HeNB_Bandwidth = homeEnbBandwidth;
  GlobalValue::GetValueByName ("simTime", doubleValue);
  double simTime = doubleValue.Get ();
  GlobalValue::GetValueByName ("epc", booleanValue);
  bool epc = booleanValue.Get ();
  GlobalValue::GetValueByName ("useUdp", booleanValue);
  bool useUdp = booleanValue.Get ();
  GlobalValue::GetValueByName ("fadingTrace", stringValue);
  std::string fadingTrace = stringValue.Get ();
  GlobalValue::GetValueByName ("numBearersPerUe", uintegerValue);
  uint16_t numBearersPerUe = uintegerValue.Get ();
  global_numBearersPerUe = numBearersPerUe;
  GlobalValue::GetValueByName ("srsPeriodicity", uintegerValue);
  uint16_t srsPeriodicity = uintegerValue.Get ();
  Config::SetDefault ("ns3::LteEnbRrc::SrsPeriodicity", UintegerValue(srsPeriodicity));

////////////////////////////////////// Number of eNBs, HeNBs, UEs
  uint32_t nMacroEnbs = 2;
  NS_LOG_UNCOND ("nMacroEnbs = " << nMacroEnbs);
  uint32_t nHomeEnbs = 5;
  NS_LOG_UNCOND ("nHomeEnbs = " << nHomeEnbs);
  uint32_t nUEs = 10;
  NS_LOG_UNCOND ("Overall nUEs = " << nUEs);

////////////////////////////////////// Creation of A2A4_RSRQoutput.txt file that logs UEs measurements
  rsrqFile.open("A2A4_RSRQoutput.txt");
  rsrqFile << "time\tRNTI\tCellID\tRSRQ " << std::endl;
  throughputFileUl.open("ThroughputUplink.txt");
  throughputFileUl << "FlowID\tSourceAddress\tDestinationAddress\tTxPackets\tThroughputUL(Kbps)\tDelayUL " << std::endl;
  throughputFileDl.open("ThroughputDownlink.txt");
  throughputFileDl << "FlowID\tSourceAddress\tDestinationAddress\tTxPackets\tThroughputDL(Kbps)\tDelayDL " << std::endl;

////////////////////////////////////// This number is used for creating the callback paths
  n_testUEs = 3;
  NS_LOG_UNCOND ("n TestUEs = " << n_testUEs);

////////////////////////////////////// This number is used to parse the callback path correctly for the UEs
  nBS = nMacroEnbs + nHomeEnbs;
  NS_LOG_UNCOND ("Overall nBS = " << nBS);

////////////////////////////////////// Creation of HeNBs
  homeEnbs.Create (nHomeEnbs);

////////////////////////////////////// Creation of eNBs
  macroEnbs.Create(nMacroEnbs);

////////////////////////////////////// Creation of UEs
  ues.Create (nUEs);

////////////////////////////////////// Install mobility model to eNB
  Ptr<ListPositionAllocator> MacroEnbPositionAlloc = CreateObject<ListPositionAllocator> ();
  MobilityHelper MacroEnbMobility;

////////////////////////////////////// Place eNBs and install mobility model
  MacroEnbPositionAlloc->Add(Vector (800, 100, 0.0));
  MacroEnbPositionAlloc->Add(Vector (1500, -300, 0.0));
  MacroEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  MacroEnbMobility.SetPositionAllocator(MacroEnbPositionAlloc);
  MacroEnbMobility.Install(macroEnbs);

////////////////////////////////////// Place HeNBs and install mobility model
  Ptr<ListPositionAllocator> HomeEnbPositionAlloc = CreateObject<ListPositionAllocator> ();
  MobilityHelper HomeEnbMobility;
  HomeEnbPositionAlloc->Add(Vector (1010, -30, 0.0));
  HomeEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
```

```
  HomeEnbMobility.SetPositionAllocator(HomeEnbPositionAlloc);
  HomeEnbMobility.Install(homeEnbs.Get(0));
  HomeEnbPositionAlloc->Add(Vector (1030, 30, 0.0));
  HomeEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  HomeEnbMobility.SetPositionAllocator(HomeEnbPositionAlloc);
  HomeEnbMobility.Install(homeEnbs.Get(1));
  HomeEnbPositionAlloc->Add(Vector (1050, -30, 0.0));
  HomeEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  HomeEnbMobility.SetPositionAllocator(HomeEnbPositionAlloc);
  HomeEnbMobility.Install(homeEnbs.Get(2));
  HomeEnbPositionAlloc->Add(Vector (1070, 10, 0.0));
  HomeEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  HomeEnbMobility.SetPositionAllocator(HomeEnbPositionAlloc);
  HomeEnbMobility.Install(homeEnbs.Get(3));
  HomeEnbPositionAlloc->Add(Vector (1090, -10, 0.0));
  HomeEnbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  HomeEnbMobility.SetPositionAllocator(HomeEnbPositionAlloc);
  HomeEnbMobility.Install(homeEnbs.Get(4));

////////////////////////////////////// Install mobility model of UE4, UE5, UE6, UE7 and UE8
  MobilityHelper ueMobility;
  ueMobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
  for (uint16_t m = 3; m < nUEs-2; m++) {
    ueMobility.Install(ues.Get(m));
  }

////////////////////////////////////// The rest UEs that create traffic
  if (nUEs > 3) {

////////////////////////////////////// Static UE4 at eNB1
    for (uint16_t i = 3; i < 4; i++) {
      ues.Get (i)->GetObject<MobilityModel> ()->SetPosition (Vector (800.0, 110.00 , 0));
      ues.Get (i)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (0, 0, 0));
    }

////////////////////////////////////// Static UE5 at eNB2
    for (uint16_t i = 4; i < 5; i++) {
      ues.Get (i)->GetObject<MobilityModel> ()->SetPosition (Vector (1500, -310.00 , 0));
      ues.Get (i)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (0, 0, 0));
    }

////////////////////////////////////// Static UE6 at HeNB1
    for (uint16_t i = 5; i < 6; i++) {
      ues.Get (i)->GetObject<MobilityModel> ()->SetPosition (Vector (1011, -30.00 , 0));
      ues.Get (i)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (0, 0, 0));
    }

////////////////////////////////////// Static UE7 at HeNB3
    for (uint16_t i = 6; i < 7; i++) {
      ues.Get (i)->GetObject<MobilityModel> ()->SetPosition (Vector (1051, -30.00 , 0));
      ues.Get (i)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (0, 0, 0));
    }

////////////////////////////////////// Static UE8 at HeNB5
    for (uint16_t i = 7; i < 8; i++) {
      ues.Get (i)->GetObject<MobilityModel> ()->SetPosition (Vector (1091, -30.00 , 0));
      ues.Get (i)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (0, 0, 0));
    }

////////////////////////////////////// Set position of UE9
////////////////////////////////////// The initial position is random inside a rectangle of (1000,-60) to (1220,60)
////////////////////////////////////// UE9 is moving randomly inside this rectangle with a random speed
   MobilityHelper ue9Mobility;
   ue9Mobility.SetPositionAllocator ("ns3::RandomRectanglePositionAllocator",
                                     "X", StringValue ("ns3::UniformRandomVariable[Min=1000|Max=1220]"),
                                     "Y", StringValue ("ns3::UniformRandomVariable[Min=-60|Max=60]"));
   ue9Mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                                 "Bounds", RectangleValue (Rectangle (1000, 1220, -60, 60)));
   ue9Mobility.Install (ues.Get(8));

////////////////////////////////////// Set position of UE10
////////////////////////////////////// The initial position is random inside a rectangle of (1000,-60) to (1220,60)
////////////////////////////////////// UE10 is moving randomly inside this rectangle with a random speed
   MobilityHelper ue10Mobility;
   ue10Mobility.SetPositionAllocator ("ns3::RandomRectanglePositionAllocator",
                                      "X", StringValue ("ns3::UniformRandomVariable[Min=1000|Max=1220]"),
                                      "Y", StringValue ("ns3::UniformRandomVariable[Min=-60|Max=60]"));
   ue10Mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                                  "Bounds", RectangleValue (Rectangle (1000, 1220, -60, 60)));
   ue10Mobility.Install (ues.Get(9));

}

////////////////////////////////////// UE1 mobility
  MobilityHelper testuelMobility;
  testuelMobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
  testuelMobility.Install(ues.Get(0));
  ues.Get (0)->GetObject<MobilityModel> ()->SetPosition (Vector (960.00, 00.00 , 0));
```

```
  ues.Get (0)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (UE_speed, 0, 0));

////////////////////////////////////// UE2 (shadow terminal) mobility
     MobilityHelper testue2Mobility;
     testue2Mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
     testue2Mobility.Install(ues.Get(1));
     ues.Get (1)->GetObject<MobilityModel> ()->SetPosition (Vector (960.00, 00.00 , 0));
     ues.Get (1)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (UE_speed, 0, 0));

////////////////////////////////////// UE3 mobility
     MobilityHelper testue3Mobility;
     testue3Mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
     testue3Mobility.Install(ues.Get(2));
     ues.Get (2)->GetObject<MobilityModel> ()->SetPosition (Vector (960.00, 00.00 , 0));
     ues.Get (2)->GetObject<ConstantVelocityMobilityModel> ()->SetVelocity (Vector (UE_speed, 0, 0));

////////////////////////////////////// Spectrum Channel implementation
  lteHelper->SetSpectrumChannelType ("ns3::MultiModelSpectrumChannel");

////////////////////////////////////// Set scheduler
  lteHelper->SetSchedulerType("ns3::RrFfMacScheduler");

////////////////////////////////////// Fading model
  if (!fadingTrace.empty ())
    {
      lteHelper->SetAttribute ("FadingModel", StringValue ("ns3::TraceFadingLossModel"));
      lteHelper->SetFadingModelAttribute("TraceFilename", StringValue (fadingTrace));
    }
  Ptr<PointToPointEpcHelper> epcHelper;
  if (epc)
    {
      epcHelper = CreateObject<PointToPointEpcHelper> ();
      lteHelper->SetEpcHelper (epcHelper);
    }

////////////////////////////////////// Propagation Loss model
  lteHelper->SetAttribute ("PathlossModel", StringValue ("ns3::FriisPropagationLossModel"));

////////////////////////////////////// Macro eNBs configuration
   Config::SetDefault ("ns3::LteEnbPhy::TxPower", DoubleValue (macroEnbTxPowerDbm));

////////////////////////////////////// eNB1
   lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
   lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (macroEnbDlEarfcn));         // Frequency A
   lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (macroEnbDlEarfcn + 18000)); // Frequency A
   lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (macroEnbBandwidth));
   lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (macroEnbBandwidth));
   macroEnbDevs.Add ( lteHelper->InstallEnbDevice (macroEnbs.Get(0)));

////////////////////////////////////// eNB2
   lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
   lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (homeEnbDlEarfcn));         // Frequency B
   lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (homeEnbDlEarfcn + 18000)); // Frequency B
   lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (macroEnbBandwidth));
   lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (macroEnbBandwidth));
   macroEnbDevs.Add ( lteHelper->InstallEnbDevice (macroEnbs.Get(1)));

////////////////////////////////////// Home eNBs configuration
  Config::SetDefault ("ns3::LteEnbPhy::TxPower", DoubleValue (homeEnbTxPowerDbm));

////////////////////////////////////// HeNB1
    lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
     lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (macroEnbDlEarfcn));         // Frequency A
    lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (macroEnbDlEarfcn + 18000)); // Frequency A
    lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (homeEnbBandwidth));
    lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (homeEnbBandwidth));
    homeEnbDevs.Add ( lteHelper->InstallEnbDevice (homeEnbs.Get(0)));

////////////////////////////////////// HeNB2
    lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
     lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (homeEnbDlEarfcn));         // Frequency B
    lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (homeEnbDlEarfcn + 18000)); // Frequency B
    lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (homeEnbBandwidth));
    lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (homeEnbBandwidth));
    homeEnbDevs.Add ( lteHelper->InstallEnbDevice (homeEnbs.Get(1)));

////////////////////////////////////// HeNB3
    lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
     lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (homeEnbDlEarfcn));         // Frequency B
    lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (homeEnbDlEarfcn + 18000)); // Frequency B
    lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (homeEnbBandwidth));
    lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (homeEnbBandwidth));
    homeEnbDevs.Add ( lteHelper->InstallEnbDevice (homeEnbs.Get(2)));

////////////////////////////////////// HeNB4
    lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
     lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (macroEnbDlEarfcn));         // Frequency A
    lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (macroEnbDlEarfcn + 18000)); // Frequency A
```

```
    lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (homeEnbBandwidth));
    lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (homeEnbBandwidth));
    homeEnbDevs.Add ( lteHelper->InstallEnbDevice (homeEnbs.Get(3)));

////////////////////////////////////// HeNB5
    lteHelper->SetEnbAntennaModelType ("ns3::IsotropicAntennaModel");
     lteHelper->SetEnbDeviceAttribute ("DlEarfcn", UintegerValue (macroEnbDlEarfcn));         // Frequency A
    lteHelper->SetEnbDeviceAttribute ("UlEarfcn", UintegerValue (macroEnbDlEarfcn + 18000)); // Frequency A
    lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (homeEnbBandwidth));
    lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (homeEnbBandwidth));
    homeEnbDevs.Add ( lteHelper->InstallEnbDevice (homeEnbs.Get(4)));

////////////////////////////////////// Add X2 interface to connect eNBs and HeNBs
  if (epc) {
      lteHelper->AddX2Interface (macroEnbs);
      lteHelper->AddX2Interface (homeEnbs);
      for (uint8_t i=0;i<macroEnbs.GetN();i++) {
        for (uint8_t j=0;j<homeEnbs.GetN();j++) {
            lteHelper->AddX2Interface (macroEnbs.Get(i), homeEnbs.Get(j));
        }
      }
    }

  ueDevs = lteHelper->InstallUeDevice (ues);

  Ipv4Address remoteHostAddr;
  Ipv4StaticRoutingHelper ipv4RoutingHelper;
  Ipv4InterfaceContainer ueIpIfaces;
  Ptr<Node> remoteHost;

////////////////////////////////////// Setting up internet and remote host
  if (epc)
    {

////////////////////////////////////// Create a single remote host
      NodeContainer remoteHostContainer;
      remoteHostContainer.Create (1);
      remoteHost = remoteHostContainer.Get (0);
      InternetStackHelper internet;
      internet.Install (remoteHostContainer);

////////////////////////////////////// Install mobility model for remote host
      Ptr<ListPositionAllocator> RemoteHostPositionAlloc = CreateObject<ListPositionAllocator> ();
      MobilityHelper RemoteHostMobility;
      RemoteHostPositionAlloc->Add(Vector (0, 0, 0.0));
      RemoteHostMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
      RemoteHostMobility.SetPositionAllocator(RemoteHostPositionAlloc);
      RemoteHostMobility.Install(remoteHost);

////////////////////////////////////// Create the Internet & the P-GW
      PointToPointHelper p2ph;
      p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
      p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
      p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
      Ptr<Node> pgw = epcHelper->GetPgwNode ();
      NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
      Ipv4AddressHelper ipv4h;
      ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
      Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
      remoteHostAddr = internetIpIfaces.GetAddress (1);
      Ipv4StaticRoutingHelper ipv4RoutingHelper;
      Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
      remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);

////////////////////////////////////// Install mobility model for P-GW
      Ptr<ListPositionAllocator> PGWPositionAlloc = CreateObject<ListPositionAllocator> ();
      MobilityHelper PGWMobility;
      PGWPositionAlloc->Add(Vector (50, 0, 0.0));
      PGWMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
      PGWMobility.SetPositionAllocator(PGWPositionAlloc);
      PGWMobility.Install(pgw);

////////////////////////////////////// Install the IP stack on the UEs
      internet.Install (ues);
      ueIpIfaces = epcHelper->AssignUeIpv4Address (NetDeviceContainer (ueDevs));
    }

////////////////////////////////////// Attach UEs 1,2(shadow),and 3 to eNBs and HeNB
  lteHelper->Attach (ueDevs.Get(0), macroEnbDevs.Get(0));                // UE1 => eNB1
  lteHelper->Attach (ueDevs.Get(1), homeEnbDevs.Get(1));         // UE2 or Shadow Terminal => HeNB2
  lteHelper->Attach (ueDevs.Get(2), macroEnbDevs.Get(0));                // UE3 => eNB1

////////////////////////////////////// Rest of the UEs to create traffic
  if (nUEs > 3) {

////////////////////////////////////// Attach rest of the UEs to eNBs and HeNBs
    for (uint16_t i = 3; i < 4; i++) {
      lteHelper->Attach (ueDevs.Get(i), macroEnbDevs.Get(0));   // UE4 attached to eNB1
```

```
    }

    for (uint16_t i = 4; i < 5; i++) {
      lteHelper->Attach (ueDevs.Get(i), macroEnbDevs.Get(1));   // UE5 attached to eNB2
    }

    for (uint16_t i = 5; i < 6; i++) {
      lteHelper->Attach (ueDevs.Get(i), homeEnbDevs.Get(0));   // UE6 attached to HeNB1
    }

    for (uint16_t i = 6; i < 7; i++) {
      lteHelper->Attach (ueDevs.Get(i), homeEnbDevs.Get(2));   // UE7 attached to HeNB3
    }

    for (uint16_t i = 7; i < 8; i++) {
      lteHelper->Attach (ueDevs.Get(i), homeEnbDevs.Get(4));   // UE8 attached to HeNB5
    }

    for (uint16_t i = 8; i < 9; i++) {
      lteHelper->Attach (ueDevs.Get(i), homeEnbDevs.Get(3));   // UE9 attached to HeNB4
    }

    for (uint16_t i = 9; i < 10; i++) {
      lteHelper->Attach (ueDevs.Get(i), homeEnbDevs.Get(4));   // UE10 attached to HeNB5
    }
}

  NS_LOG_UNCOND ("\n...setting up applications...\n");

////////////////////////////////// Install and start applications on UEs and remote host
  uint16_t dlPort = 10000;
  uint16_t ulPort = 20000;
  uint16_t otherPort = 30000;

  ApplicationContainer clientApps;
  ApplicationContainer serverApps;

  Ptr<UniformRandomVariable> startTimeSeconds = CreateObject<UniformRandomVariable> ();
  if (useUdp)
    {
      startTimeSeconds->SetAttribute ("Min", DoubleValue (0));
      startTimeSeconds->SetAttribute ("Max", DoubleValue (0.010));
    }
  else
    {
      startTimeSeconds->SetAttribute ("Min", DoubleValue (0.100));
      startTimeSeconds->SetAttribute ("Max", DoubleValue (0.110));
    }

////////////////////////////////// Test UEs (1,2,3) traffic
  for (uint32_t u = 0; u < 3; ++u) {
    Ptr<Node> ue = ues.Get (u);
    Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ue->GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
    for (int k = 0; k<1; k++) {
      ++ulPort;
      ++dlPort;
      ++otherPort;
      PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), dlPort));
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), ulPort));
      PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), otherPort));
      serverApps.Add (dlPacketSinkHelper.Install (ues.Get(u)));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      serverApps.Add (packetSinkHelper.Install (ues.Get(u)));

      UdpClientHelper dlClient (ueIpIfaces.GetAddress (u), dlPort);
      dlClient.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      dlClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper client (ueIpIfaces.GetAddress (u), otherPort);
      client.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      client.SetAttribute ("MaxPackets", UintegerValue(1000000));

      clientApps.Add (dlClient.Install (remoteHost));
      clientApps.Add (ulClient.Install (ues.Get(u)));

      clientApps.Add (client.Install (ues.Get(u)));
    }
    serverApps.Start (Seconds (0.1));
    clientApps.Start (Seconds (0.1));
  }


////////////////////////////////// Rest UEs (4,5) traffic
```

```
  for (uint32_t u = 3; u < 5; ++u) {
    Ptr<Node> ue = ues.Get (u);
    Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ue->GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
    for (int k = 0; k<numBearersPerUe; k++) {
      ++ulPort;
      ++dlPort;
      ++otherPort;
      PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), dlPort));
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), ulPort));
      PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), otherPort));
      serverApps.Add (dlPacketSinkHelper.Install (ues.Get(u)));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      serverApps.Add (packetSinkHelper.Install (ues.Get(u)));

      UdpClientHelper dlClient (ueIpIfaces.GetAddress (u), dlPort);
      dlClient.SetAttribute ("Interval", TimeValue (MilliSeconds(100)));
      dlClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds(100)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper client (ueIpIfaces.GetAddress (u), otherPort);
      client.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      client.SetAttribute ("MaxPackets", UintegerValue(1000000));

      clientApps.Add (dlClient.Install (remoteHost));
      clientApps.Add (ulClient.Install (ues.Get(u)));

      clientApps.Add (client.Install (ues.Get(u)));
    }
    serverApps.Start (Seconds (0.1));
    clientApps.Start (Seconds (0.1));
  }

////////////////////////////////// Rest UEs (6,7,8) traffic
  for (uint32_t u = 5; u < 8; ++u) {
    Ptr<Node> ue = ues.Get (u);
    Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ue->GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
    for (int k = 0; k<numBearersPerUe; k++) {
      ++ulPort;
      ++dlPort;
      ++otherPort;
      PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), dlPort));
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), ulPort));
      PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), otherPort));
      serverApps.Add (dlPacketSinkHelper.Install (ues.Get(u)));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      serverApps.Add (packetSinkHelper.Install (ues.Get(u)));

      UdpClientHelper dlClient (ueIpIfaces.GetAddress (u), dlPort);
      dlClient.SetAttribute ("Interval", TimeValue (MilliSeconds(1)));
      dlClient.SetAttribute ("MaxPackets", UintegerValue(10000000));

      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds(1)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(10000000));

      UdpClientHelper client (ueIpIfaces.GetAddress (u), otherPort);
      client.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      client.SetAttribute ("MaxPackets", UintegerValue(1000000));

      clientApps.Add (dlClient.Install (remoteHost));
      clientApps.Add (ulClient.Install (ues.Get(u)));

      clientApps.Add (client.Install (ues.Get(u)));
    }
    serverApps.Start (Seconds (0.1));
    clientApps.Start (Seconds (0.1));
  }

////////////////////////////////// Rest UEs (9,10) traffic
  for (uint32_t u = 8; u < nUEs; ++u) {
    Ptr<Node> ue = ues.Get (u);
    Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ue->GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
    for (int k = 0; k<numBearersPerUe; k++) {
      ++ulPort;
      ++dlPort;
      ++otherPort;
      PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), dlPort));
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), ulPort));
      PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), otherPort));
      serverApps.Add (dlPacketSinkHelper.Install (ues.Get(u)));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      serverApps.Add (packetSinkHelper.Install (ues.Get(u)));
```

```
      UdpClientHelper dlClient (ueIpIfaces.GetAddress (u), dlPort);
      dlClient.SetAttribute ("Interval", TimeValue (MilliSeconds(100)));
      dlClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds(100)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper client (ueIpIfaces.GetAddress (u), otherPort);
      client.SetAttribute ("Interval", TimeValue (MilliSeconds(10)));
      client.SetAttribute ("MaxPackets", UintegerValue(1000000));

      clientApps.Add (dlClient.Install (remoteHost));
      clientApps.Add (ulClient.Install (ues.Get(u)));

      clientApps.Add (client.Install (ues.Get(u)));
    }
    serverApps.Start (Seconds (0.1));
    clientApps.Start (Seconds (0.1));
  }

////////////////////////////////////// Flow monitor tracing
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor;

////////////////////////////////////// Install Flow Monitor to devices
  monitor = flowmon.Install(ues.Get(0));
  monitor = flowmon.Install(macroEnbs);
  monitor = flowmon.Install(homeEnbs);
  monitor = flowmon.Install(remoteHost);

  Simulator::Stop(Seconds(simTime));

////////////////////////////////////// Enable traces
// lteHelper->EnablePhyTraces ();
// lteHelper->EnableMacTraces ();
// lteHelper->EnableRlcTraces ();
  lteHelper->EnablePdcpTraces ();

  Ptr<RadioBearerStatsCalculator> pdcpStats = lteHelper->GetPdcpStats ();
  pdcpStats->SetAttribute ("EpochDuration", TimeValue (Seconds (0.1)));

////////////////////////////////////// PDCP trace output files
  pdcpStats->SetAttribute ("DlPdcpOutputFilename", StringValue ("shadow_Pdcp_downlink.txt"));
  pdcpStats->SetAttribute ("UlPdcpOutputFilename", StringValue ("shadow_Pdcp_uplink.txt"));

  std::ofstream my_stats;

////////////////////////////////////// Connect custom trace sinks for RRC connection establishment
////////////////////////////////////// and handover notification
  Config::Connect ("/NodeList/*/DeviceList/*/ ",
                   MakeCallback (&NotifyConnectionEstablishedEnb));
  Config::Connect ("/NodeList/*/DeviceList/*/LteUeRrc/ConnectionEstablished",
                   MakeCallback (&NotifyConnectionEstablishedUe));
  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/HandoverStart",
                   MakeCallback (&NotifyHandoverStartEnb));
  Config::Connect ("/NodeList/7/DeviceList/*/LteUeRrc/HandoverStart",
                   MakeCallback (&NotifyHandoverStartUe));
  Config::Connect ("/NodeList/8/DeviceList/*/LteUeRrc/HandoverStart",
                   MakeCallback (&NotifyHandoverStartUe));
  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/HandoverEndOk",
                   MakeCallback (&NotifyHandoverEndOkEnb));
  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/RecvMeasurementReport",
                   MakeCallback (&RecvMeasurementReportCallback));

  std::stringstream sss;
  std::string nBS_str;
  std::string path_str;

  for (uint32_t i = 0; i < n_testUEs; i++ ) {
    sss.clear();
    sss << nBS + i;
    sss >> nBS_str;
    path_str = "/NodeList/";
    path_str = path_str.append (nBS_str);
    path_str = path_str.append("/DeviceList/*/LteUePhy/ReportUeMeasurements");
    Config::Connect (path_str, MakeCallback (&ReportUeMeasurementsCallback));
    }

  Simulator::Run();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

////////////////////////////////////// Calculation of throughput and delay (uplink and downlink) of UE1
  double throughputUl = 0.0;
```

```
  double throughputDl = 0.0;
  double delayUl = 0;
  double delayDl = 0;
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != stats.end (); ++i)
    {
      Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);

      if (t.sourceAddress=="7.0.0.2" && t.destinationAddress=="1.0.0.2" ) {
                throughputUl=i->second.txBytes * 8.0 /(i->second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds())/ 1024;
          delayUl=i->second.delaySum.GetSeconds()/i->second.rxPackets;
          throughputFileUl << i->first << "\t" << t.sourceAddress << "\t" << t.destinationAddress << "\t" <<
                                                  i->second.txPackets << "\t" << throughputUl << "\t" <<
                                                  delayUl << "\t" << std::endl;
      }
      if (t.sourceAddress=="1.0.0.2" && t.destinationAddress=="7.0.0.2" ) {
                throughputDl=i->second.rxBytes * 8.0 /(i->second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds())/ 1024;
          delayDl=i->second.delaySum.GetSeconds()/i->second.rxPackets;
          throughputFileDl << i->first << "\t" << t.sourceAddress << "\t" << t.destinationAddress << "\t" <<
                                                  i->second.txPackets << "\t" << throughputDl << "\t" <<
                                                  delayDl << "\t" << std::endl;
      }
    }

////////////////////////////////// Creation of XML file
  monitor->SerializeToXmlFile("shadow.xml", true, true);

  rsrqFile.close();
  throughputFileUl.close();
  throughputFileDl.close();

  Simulator::Destroy();

  return 0;

}
```

Βιβλιογραφία

[1] *Radio Protocols for LTE and LTE-Advanced* – SeungJune Yi, SungDuck Chun, YoungDae Lee, SungJun Park, SungHoon Jung

[2] *UMTS* – Javier Sanchez, Mamadou Thioune

[3] *The LTE/SAE deployment handobook* – Jyrki T. J. Penttinen

[4] 3GPP Specifications TS 36.101, "Evolved Univesral Terrestrial Radio Access (E-UTRA); User Equipment (UE); radio transmission and reception"

[5] http://blog.3g4g.co.uk/2009/03/home-e-nodeb-architecture-in-release-8.html

[6] *Mobile and wireless communications for IMT-Advanced and beyond* – Afif Osseiran, Jose F. Monserrat, Werner Mohr

[7] *From GSM to LTE, An introduction to mobile networks and mobile broadband* – Martin Sauter

[8] http://diameter-protocol.blogspot.gr/2012/07/diameter-interfaces-in-lte-epc.html

[9] *LTE the UMTS Long Term Evolution, from theory to practice, second edition* – Stefania Sesia, Issam Toufic, Matthew Baker

[10] http://www.tutorialspoint.com/lte/lte_radio_protocol_architecture.htm

[11] *Evolved Packet System, The LTE and SAE evolution of 3G UMTS* – Pierre Lescuyer, Thierry Lucidarme

[12] UMTS *Long Term Evolution, Technology introduction* – C. Gessner, A. Roessler, M. Kottkamp

[13] http://www.tutorialspoint.com/lte/lte_communication_channels.htm

[14] *LTE and the evolution to 4G wireless* – Moray Rumney

[15] http://jwcn.eurasipjournals.com/content/2012/1/287

[16] http://ecee.colorado.edu/~ecen4242/LTE/radio.htm

[17] *An introduction to LTE, LTE, LTE-Advanced, SAE and 4G mobile communications* – Christopher Cox

[18] 3GPP TR 21.101, "Technical Specifications and Technical Reports for a UTRAN-based 3GPP system"

[19] http://www.3gpp.org/IMG/pdf/2009_10_3gpp_IMT.pdf

[20] 3GPP Specification TS 33.401, "System Architecture Evolution; Security Architecture", 2012

[21] http://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world

[22] http://www.nsnam.org/docs/manual/ns-3-manual.pdf

[23] http://personal.ee.surrey.ac.uk/Personal/K.Katsaros/media/NS-3-Presentation-2013.pdf

[24] http://www.nsnam.org/docs/release/3.19/tutorial/ns-3-tutorial.pdf

[25] http://www.nsnam.org/tutorials/consortium13/lte-tutorial.pdf

[26] http://www.nsnam.org/docs/models/html/lte-design.html

[27] http://www.nsnam.org/doxygen/lena-x2-handover_8cc_source.html

[28] *A. Kaloxylos, S. Barmpounakis, P. Spapis, N. Alonistioti*, "An efficient RAT selection mechanism for 5G cellular networks", IWCMC 2014, August 4 - 8, Nicosia, Cyprus

[29] *Nicola Baldo, Manuel Requena-Esteso, Marco Miozzo,* "An open source model for the simulation of LTE handover scenarios and algorithms in ns-3", MSWiM'13, November 3-8, 2013, Barcelona, Spain