בס"ד

המרכז האקדמי לב

החוג להנדסת תכנה

פרויקט גמר

Realistic Network Simulator (RNS)

מגישים:

נדב וויס ונפתלי מונטג

מנחים:

ד"ר נדב שוייצר וד"ר אריאל שטולמן

אלול תשע"ט

בס"ד

המרכז האקדמי לב

החוג להנדסת תכנה

פרויקט גמר

Realistic Network Simulator (RNS)

מגישים:

נדב וויס ונפתלי מונטג

מנחים:

ד"ר נדב שוייצר וד"ר אריאל שטולמן

אלול תשע"ט

תודות

1. תודה לקב"ה שהביאנו עד הלום
2. תודה מיוחדת לדוקטור נדב שוייצר על העזרה לאורך כל הפרויקט. התמיכה, הן מבחינת הזמן שהוא השקיע והן מבחינת המאמץ וההדרכה הרבה.
3. תודה למחלקת מחשבים, בפרט לדוקטור מוטי רייף, מר אבי טרייסטמן, פרופסור דן בוכניק, דוקטור אריה טייטלבאום, וגברת חנה קליין פרופסורים ואנשי מקצוע שעזרו לנו לעתים קרובות ככל שביקשנו.

תודה למרכז האקדמי לב, למרצים, למנהלים ולכל הצוות שסיפקו לנו את כל העזרה הדרושה לאורך הדרך.

תודה נוספת, למשפחות שלנו שעזרו לנו, תמכו בנו ועזרו לנו במהלך הפרויקט.

תודה אחרונה, לכל מי שעזר לנו במהלך הלימודים והפרויקט ושמו אינו מופיע כאן.

תקציר

פרויקט זה נועד כדי לספק סימולטור רשת מציאותי ומדויק המאפשר לחוקרים לבחון את האפקטיביות של מתקפות ו/או הגנות על רשתות MANET, המדמות רשתות אמיתיות הנמצאות בשימוש במגוון רחב של יישומים.

האתגר היה ליצור כלי סימולציות ידידותי למשתמש, כזה שיעניק לחוקר את הגמישות להפעיל סימולציה תוך שימוש בנתונים רלוונטיים שנבחרו על ידו, ממגוון מקורות שאינם בהכרח מתוכננים לעבוד יחד. בנוסף, נדרש שהכלי יוכל להריץ במהירות וביעילות מספר רב של סימולציות המייצגות תסריטים שונים.

הכלי שמוצג בפרויקט זה [Realistic Network Simulator] מחבר ארבעה כלים שונים: SUMO, BonnMotion, OSRM ו-ns-3. לכל אחד מהכלים הללו ישנם יתרונות ומגבלות, וכל אחד מהם כולל פרמטרים שונים. בדו"ח זה מתוארים כלים אלה בפירוט, ומוסבר מדוע הם נבחרו לשימוש בפרויקט זה. כמו כן, ישנו הסבר כיצד להתקין ולהגדיר כלים אלו כדי שישתלבו בכלי הסופי, על מנת לאפשר הפעלה אוטומטית של כל אחד מהם כדי ליצור את הסימולציה הרצויה.

בדו"ח זה מתואר תהליך קביעת הצירוף האופטימלי של כלים ליצירת סימולציה, תוך התחשבות בכך שלכל אחד מן הכלים יש קלטים ופלטים שונים. ומובא הסבר כיצד בוצעה אופטימיזציה לקוד, וכיצד בוצעו סימולציות קטנות לפני הפעלת סימולציות מרובות וגדולות כדי למנוע הרצות מיותרות. כל תוצאות הסימולציה נשמרות בתבנית שניתן לנתח כדי להציג את המידע הרלוונטי על יעילות הגנות והתקפות בהתאם לתרחיש שהוגדר על ידי המשתמש.

בפרויקט זה נדגים באמצעות הסימולטור המוצע הרצת סימולציה של רשת MANET שהוקמה בין מכוניות הנעות באופן אקראי. כדי שהסימולציה תהיה קרובה למציאות היא משתמשת במפות של מקומות אמיתיים תוך התחשבות בהשפעה של מבנים מקומיים על עוצמת האות שנשלח על ידי כלי רכב. בעזרת הכלי ניתן לשנות פרמטרים בודדים כדי להפעיל סימולציות עם סוגים שונים של התקפות והגנות על אותה הרשת ובכך לראות את ההשפעה של אותם הפרמטרים על הסימולציה. בדוח זה מובאות דוגמאות של סימולציות מרובות שהורצו בכדי לאמת את נכונות הקוד ולהמחיש את אפקטיביות הכלי.

תוכן עניינים

תודות 3
תקציר 4
תוכן עניינים 5
רשימת טבלאות 7
רשימת איורים 8
רשימת נספחים 9
1. מבוא 10
2. רקע היסטורי 11
3. רקע תיאורטי 12
3.1 רשתות MANET 12
3.2 פרוטוקול OLSR 12
3.3 התקפות והגנות ברשתות MANET 14
3.3.1 התקפת Node Isolation 14
3.3.2 הגנת DCFM 14
4. מבנה הפרויקט 15
4.1 מבוא 15
4.2 OSRM 17
4.3 BonnMotion 17
4.4 SUMO 17
4.5 Projection Correction 18
4.6 ns-3 18
4.7 המנתח (Parser) 18
4.8 הכלי הסופי - Realistic Network Simulator 18
5. הקוד 20
5.1 המפות והבניינים 20
5.2 הסימולציה עצמה 22
5.3 המנתח (Parser) 39
5.4 הכלי השלם 40
6. הסיבות לצורת המימוש הנוכחית 41
7. קשיים 42
8. היתרונות במימוש הנוכחי 42
9. הדגמת יעילות הכלי באמצעות סימולציות 43
9.1 מבוא 43
9.2 הרצות עם צמתים במיקום קבוע 44

9.3 הרצות עם צמתים שנעים בקווים ישרים 45
9.4 הרצות עם צמתים שנעים על גבי מפה 46
10. ביבליוגרפיה 47
11. נספח א' - הוראות התקנה 48
1.11 דרישות קדם 48
2.11 התקנת חלקי הפרויקט 48
11.2.1 BonnMotion 48
11.2.2 OSRM 49
11.2.3 SUMO 50
11.2.4 ns-3 50
11.2.4.1 NetAnim 51
3.11 דוגמת שימוש בכלים 51
11.4 דרישות לצורך הרצת הכלי 52
11.4.1 פירוט הפרמטרים 52
11.4.1.1 פרמטרים להגדרה ראשונית: 52
11.4.1.2 פרמטרים להרצה ספציפית: 53
12. נספח ב' - דוגמת הרצה 54
Abstract 55

רשימת טבלאות

טבלה 1: השוואה בין SUMO ל-BonnMotion 41

רשימת איורים

איור 1: דוגמה לשרת MANET 13
איור 2: מבנה הפרויקט 15
איור 3: רשימת הפרמטרים המוגדרים בקוד, והסבר שלהם 38
איור 4: דוגמה לקובץ מסוג ns_params 38
איור 5: מיקום קבוע ללא בניינים 44
איור 6: מיקום קבוע ללא בניינים במפות 44
איור 7: מיקום קבוע עם בניינים במפות 44
איור 8: כמות ההודעות שהגיעו אל הקורבן - מיקום קבוע 44
איור 9: קווים ישרים bonn 45
איור 10: קווים ישירים ns-3 45
איור 11: כמות ההודעות שהגיעו אל הקורבן - תנועה בקווים ישרים 45
איור 12: מפות עם בניינים 46
איור 13: מפות בלי בניינים 46
איור 14: כמות ההודעות שהגיעו אל הקורבן - תנועה בתוך מפות 46
איור 15: דוגמה לתוכן של config.csv 54
איור 16: דוגמה לתוכן של Isolation.csv 54

רשימת נספחים

11. נספח א' - הוראות התקנה 48
1.11 דרישות קדם 48
2.11 התקנת חלקי הפרויקט 48
11.2.1 BonnMotion 48
11.2.2 OSRM 49
11.2.3 SUMO 50
11.2.4 ns-3 50
11.2.4.1 NetAnim 51
3.11 דוגמת שימוש בכלים 51
11.4 דרישות לצורך הרצת הכלי 52
11.4.1 פירוט הפרמטרים 52
12. נספח ב' - דוגמת הרצה 54

1. מבוא

Mobile ad hoc networks [1] המוכרים כ-MANETs או wireless ad hoc networks הן רשתות הבונות את עצמן באופן מתמיד מאוסף מכשירים שמתקשרים ביניהם. כל אחד מהמכשירים מתפקד כראוטר בכך שהוא מעביר מידע בין מכשירים אחרים ברשת. כל אחד מהמכשירים ברשת כזו יכול לנוע בחופשיות ללא תלות באחרים, ובכך לשנות תדיר את טופולוגיית הרשת, לכן הקשרים בין הצמתים צריכים להתעדכן בתדירות גבוהה. רשתות אלה יכולות לעמוד בפני עצמן או להתחבר לרשתות אינטרנט אחרות.

הטופולוגיה המבוזרת, הדינאמית והאוטונומית של MANET מהווה יתרון. עם זאת, היא פגיעה לכל מיני מתקפות ולכן מאתגר להגן עליה .רשתותMANET בעלות ערך רב במיוחד לאפליקציות בתחומי הרפואה, הגנת הסביבה, תקשורת לחילוץ באסונות - והרבה מיישומים אלו עוסקים בעניינים של חיים ומוות.

הכלי המתואר בדו"ח (RNS) מאפשר לחוקרים להריץ סימולציות של מגוון של התקפות אפשריות על רשתות MANET כדי לזהות חולשות ולבדוק יעילות של הגנות כנגדן.

2. רקע היסטורי

ישנם מספר פתרונות להדמיית רשתות, ביניהם QualNet [2], GridRoofnet [3] ו-OPNET [4].

GridRoofnet תוכנן כדי לעצב רשתות Ad Hoc עוד כאשר הם היו רעיון חדש. GridRoofnet התבססה על רכיבי מדף והשתמשה במערכת הפעלה מבוססת קוד פתוח דמוית יוניקס. GridRoofnet השתמשה בגרסה של פרוטוקול ניתוב Ad Hoc הנקרא DSDV [5]. בפרויקט זה המיקום של הצמתים היה קבוע, ומטרתו הייתה לפתור את בעיית השינוי הדינמי בניתוב בין הצמתים (למשל, כאשר חלק מהצמתים כבו).

QualNet מבוסס על GloMoSim (Global Mobile Information System Simulator) [6] אשר מנצל ביצוע מקבילי כדי להקטין את זמן הסימולציה של מודלים עם רמת דיוק גבוהה על רשתות תקשורת גדולות. סימולטור זה תומך כיום בפרוטוקולים לרשת אלחוטית בלבד והוא מבוסס על מודלים של סימולציה לאירועים ייחודיים. השימוש המיועד שלה הוא צבאי ותעשייתי.

OPNET נוצר כדי להפחית את זמן הריצה של סימולציות על ידי ניצול יכולות מקבילות ומבוזרות. פרוטוקולי הרשת עוצבו באמצעות שיטה מונחית עצמים. למוצר יש ממשקי משתמש נרחבים עם תרשימים, טבלאות וגרפים אינטואיטיביים. השימוש המיועד שלה הוא תקשורת עסקית ותקשורת אישית.

כל הכלים הללו שימשו לעיצוב רשתות. מטרת כלי הסימולציה המוצג להלן היא לסייע לחוקרי אבטחת סייבר ללמוד חולשות הן מנקודת המבט של התקיפה והן מנקודת המבט של ההגנה, אשר יתוארו בפירוט בפרק 2.3.

3. רקע תיאורטי

3.1 רשתות MANET

רשתות MANET או Mobile Ad Hoc NETworks [1] הן רשתות הבנויות מאוסף צמתים שמתקשרים ביניהם ללא תשתית מוקדמת וללא צד שלישי שמנהל את הרשת. הצמתים ברשת יכולים להעביר מידע בין אחד לשני ללא צורך בתשתית קיימת של שרתים ולוויינים. מכיוון שהצמתים ברשתות אלו רשאים לנוע בחופשיות לכל הכיוונים, הקישוריות שלהם עם צמתים אחרים יכולה להשתנות כל הזמן. עם זאת פרוטוקול הרשת אמור להבטיח שהקישוריות בין כל צמתי הרשת תישמר. הצמתים ברשת בדרך כלל מוגבלים מבחינת רוחב פס.

ישנם מספר סוגים של פרוטוקולים לניתוב ברשתות אלו, ניתן לסווג אותם לשניים עיקריים: פרואקטיבי (Proactive) וריאקטיבי (Reactive).

<u>פרוטוקול פרואקטיבי (Proactive):</u>

סוג פרוטוקול זה מייצר טבלאות ניתוב על ידי הפצה מחזורית של טבלאות הניתוב של הצמתים ברשת. היתרון העיקרי בכך הוא שכל צומת יכול למצוא נתיב לצומת יעד כלשהיא בצורה מהירה בעקבות העדכונים החוזרים ונשנים. החסרונות העיקריים בכך הם הכמות הגדולה של המידע שעובר לצורך תחזוקת הרשת ותגובה איטית יחסית לשינויים.

<u>פרוטוקול ריאקטיבי (Reactive):</u>

סוג פרוטוקול זה מייצר טבלאות ניתוב על פי דרישה - רק כאשר נוצר צורך בניתוב ועל ידי הפצת המידע ברשת. היתרון העיקרי בכך הוא בחיסכון של עומס על הרשת. לעומת זאת, החסרונות העיקריים בכך הם משך הזמן שלוקח לצומת למצוא נתיב לצומת יעד כלשהיא, ואפשרות להצפה של הרשת בעקבות הפצה של הודעות דרישה מרובות.

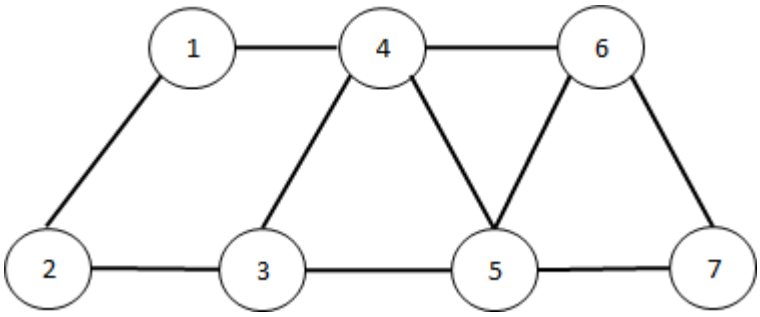
מכיוון שבפרויקט זה מתעסקים בפרוטוקול OLSR נרחיב עליו:

3.2 פרוטוקול OLSR

פרוטוקול OLSR (Optimized Link State Routing) [7] הוא פרוטוקול ניתוב פרואקטיבי מבוסס IP שניתן להשתמש בו ברשתות MANET. פרוטוקול זה משתמש בשתי הודעות ניהול, הודעת HELLO והודעת TC (Topology Control) כדי לגלות מידע על צמתים ולהפיץ אותו ברשת. צמתים ברשת משתמשים במידע הזה על מנת למצוא את הצומת הבא בשרשרת הניתוב לכל צומת ברשת.

הודעת ה-HELLO נשלחת באופן מחזורי על ידי כל צומת לכל הצמתים שממוקמים במרחק קפיצה אחת ממנו, ומכילה בתוכה את רשימת כל השכנים של הצומת הנוכחי. לכן, הודעות אלו משמשות לצורך גילוי כל הצמתים במרחק של עד שתי קפיצות מהצומת המפרסם (מציאת כל השכנים של שכניך). הקבוצה הקטנה ביותר של צמתים שמאפשרת לשלוח הודעה לכל הצמתים במרחק שתי קפיצות מהצומת הנוכחי באמצעותם בלבד נקראת צמתי MPR. ע"י צמתי ה-MPR בלבד נקבל תת רשת שבאמצעותה ניתן לשלוח הודעות ניתוב לשאר הצמתים ברשת, על מנת ליידע אותם על הטופולוגיה של הרשת. לכן, רק צמתי MPR מעבירים הודעות המכילות את רשימת צמתי ה-MPR של השולח - הודעות אלו נקראות הודעות TC. מכיוון שהודעות TC מכילות רק את צמתי ה-MPR של הצומת השולח ולא את כל הצמתים במרחק קפיצה אחת ממנו, הודעות אלו מעבירות מידע חלקי על הטופולוגיה של הרשת. האופטימיזציה (כפי שמופיע בשם OLSR) היא בזה שהודעות תקורה יועברו לכל שכני השולח, אבל רק מי שמונה על ידי השולח כ-MPR ימשיך להעביר אותן הלאה - קרי: חיסכון בתקורת הרשת.

לדוגמה ברשת הבאה:



איור 1: דוגמה לשרת MANET

השכנים במרחק קפיצה אחת של צומת 2 הם צמתים 1 ו-3. השכנים במרחק שתי קפיצות הם 4 ו-5. לכן, MPR אפשרי של צומת 2, הוא הצומת 3 (במקרה זה הוא גם היחידי).

3.3 התקפות והגנות ברשתות MANET

פרק זה סוקר בקצרה את התקפות והגנות המופיעות בהרצות הסופיות כדי לבדוק את יעילות הכלי שפיתחנו.

3.3.1 התקפת Node Isolation

מטרתה של התקפת node isolation היא לבודד צומת ספציפי מתוך הרשת כך שהוא לא יקבל הודעות משאר הצמתים שברשת. ההתקפה מונעת מצומת מסוים לקבל הודעות משאר הצמתים, ע"י כך שהיא גורמת לשאר הצמתים לחשוב שהצומת לא קיים. המימוש המעשי של התקפה זו תלוי בפרוטוקול הניתוב בו משתמשים. כאמור, במסגרת מחקר זה מדובר בפרוטוקול OLSR.

ההתקפה מסתמכת על אלגוריתם בחירת ה-MPR-ים. הצומת התוקף מוסיף להודעת ה-HELLO שהוא שולח לצומת הקורבן הודעה שקרית. בהודעה זו הוא מדווח שכל השכנים במרחק 2 קפיצות מהקורבן הם שכנים במרחק קפיצה אחת ממנו (מהתוקף). בנוסף, הצומת התוקף טוען שיש לו עוד צומת שכן שלא קיים. בכך התוקף מבטיח את בחירתו כ-MPR היחיד של T (מכיוון שהוא הצומת היחיד המכסה את כל הצמתים הממוקמים במרחק 2 קפיצות מהקורבן).

כעת, כל הודעת TC שתישלח ע"י הקורבן תגיע לכל השכנים במרחק של עד שתי קפיצות ממנו, וכדי להגיע לשאר האנשים שברשת ההודעה תהיה חייבת לעבור דרך הצומת התוקף (מכיוון שהוא MPR). אם הצומת התוקף יימנע מעבר של הודעות אלו הוא יגרום לשאר הרשת (דרגה שלישית ומעלה של הנתקף) לחשוב שהקורבן עזב את הרשת, וממילא שאר הצמתים יימנעו מלשלוח הודעות לקורבן. צריך לציין, שלמרות ההתקפה, השכנים הישירים והשכנים במרחק שתי קפיצות מהנתקף עדיין ידעו על קיומו באמצעות הודעות ה-HELLO שהקורבן ממשיך לשלוח. התוקף יודע מי הם השכנים במרחק שתי קפיצות מהמותקף בעזרת הודעות ה-TC.

3.3.2 הגנת DCFM

פתרון זה מובא ב-[8]. הפתרון עוסק במניעה של התקפת Node Isolation על ידי זיהוי של צמתים ששולחים הודעות חשודות, ומציאת סתירות בין מה שצמתים אלו שולחים למה שנשלח ע"י שאר משתמשי הרשת. בפתרון זה מאמתים את הודעת ה-HELLO שמקבלים מכל שכן.

ישנם שלושה שלבים לטיפול בהודעת ה-HELLO:

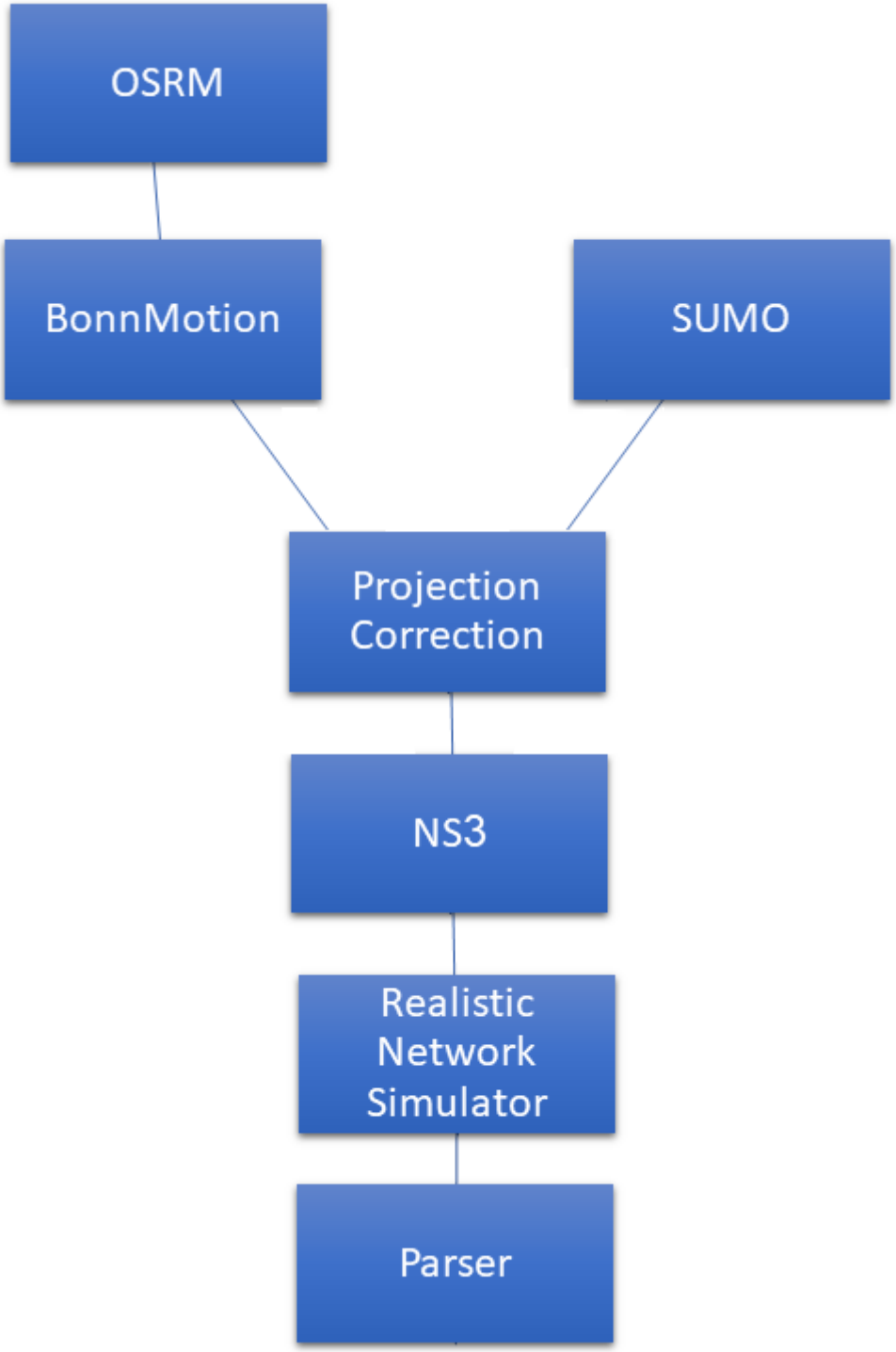
1. מוודאים שהודעת ה-HELLO של הצומת השולח הגיונית: כלומר אם צומת A טוענת בהודעת HELLO שצמתים B ו-C הם שכנים ישירים שלו, אז צריך שגם צומת B וגם צומת C יפרסמו בהודעות ה-HELLO שלהם את צומת A, ואם הם לא יעשו כך ייתכן ש-A משקר בהודעת ה HELLO שלו (הצומת A מסומן כחשוד).
2. בודקים לכל צומת שמופיע בהודעת HELLO של A האם קיים שכן לצומת הזה (נקרא לשכן Z) שהוא:
   2.1. לא מופיע בהודעת HELLO של הצומת A.
   2.2. נמצא במרחק של לפחות 3 צמתים מהצומת הקורבן.
   2.3. האם A מינה את אחד משכניו (ממי שמופיע בהודעת ה HELLO) בתור MPR, או הפוך, שמכסה את הצומת Z.

   אם לא, הצומת A מסומן כחשוד.
3. אם A טוען שכל הצמתים ברשת, חוץ מהשכנים של הצומת המותקף עצמו שכנים שלו, ובכך מתגבר על כלל מספר 2, מחשיבים את A בתור צומת חשוד.

עדיין, הצומת התוקף יכול לרמות אלגוריתם זה במקרים מסוימים ולכן כל צומת ברשת צריך לבצע עוד בדיקות כדי לדעת האם התקפה כזו יכולה להתרחש בעזרתו וללא ידיעתו. לפרוט מלא כיצד זה מתבצע ניתן לעיין במאמר המקורי [8].

4. מבנה הפרויקט

4.1 מבוא



איור 2: מבנה הפרויקט

מטרת הפרויקט היא לספק לחוקרים את האפשרות לבצע הרצות ולבדוק אותן בצורה פשוטה ונוחה. RNS מחבר ביחד כמה כלים על מנת לאפשר ביצוע הרצות מדויקות יותר ועל מנת לספק כמה שיותר כלים שונים לרשות החוקר שמשתמש בו. הכלי אחראי על ריכוז הפעולות לביצוע, על מנת להקל על השימוש בכל הכלים יחד. מאפיין חשוב נוסף של RNS הוא שניתן להחליף חלקים בפרויקט בצורה פשוטה יחסית. למשל, המבנה הנוכחי של הכלים מאפשר לבדוק יעילות של מתקפות והגנות על גבי רשתות MANET בצורה הדומה למציאות במידה מרבית. RNS מאפשר לבצע סימולציות של תקשורת ברשתות MANET כאשר הצמתים מדמים מכוניות נוסעות בכביש. על מנת ליצור סימולציה שתואמת כמה שיותר את המציאות, כלי הרכב בסימולציה נעים בתוך כבישים שהוצאו מתוך מפות של מקומות אמיתיים. זה מבוצע תוך התחשבות בהשפעת הבניינים על עוצמת האות שנשלח על ידי הרכבים (אות WiFi נחלש במעבר דרך קירות) יובא להלן הסבר על כל אחד מחלקי הפרויקט כאשר הסבר על היתרונות והסיבות לשימוש בכלים אלו יובאו בפרקים 6 ו-8.

OSRM 4.2

OSRM [9] הוא מנוע ניתוב מבוסס c++, שפותח לצורך מציאת הדרך הקצרה ביותר בין שתי נקודות במפת דרכים. OSRM הוא כלי קוד מקור פתוח עם רישיון BSD 2-Clause[1], והוא עובד על פורמט מפות חינמי בשם OpenStreetMap [10]. ניתן להשתמש ב-OSRM באמצעות API מבוסס בקשות HTTP, או בתור ספרייה של c++ בצורה המאפשרת כיוונון מדויק יותר ותקורה פחותה.

BonnMotion 4.3

BonnMotion [11] הוא כלי מבוסס Java שיוצר ומנתח תרחישי תנועת כלי רכב. BonnMotion הוא כלי קוד מקור פתוח עם רישיון GNU General Public License[2]. הכלי מכיל בתוכו אפשרות לייצוא של תרחישי התנועה למגוון סימולטורים של תקשורת (בין היתר ns-3) הכלי מפותח בשיתוף עם קבוצה מאוניברסיטת Bonn בגרמניה. כלי זה תומך במגוון רחב של מודלי תנועה, למשל (מהמודלים בהם נגענו):

- Random Waypoint model - עצירה למספר שניות ותנועה במהירות קבועה ליעד אקראי באופן מחזורי.
- Random Walk model - תנועה אקראית במרחב נתון.
- Random Street model - תנועה אקראיות בתוך רחובות הנתונים על ידי מפת דרכים.

בפרויקט זה נעשה שימוש ב-BonnMotion לצורך יצירה של קבצי תנועה של רכבים לקובץ, שבהמשך נקרא על ידי הכלי ns-3. כאשר חישוב המסלולים בתוך מפת הדרכים הנתונה מתבצע על ידי BonnMotion תוך שימוש ב-OSRM.

SUMO 4.4

SUMO [12] הוא כלי קוד מקור פתוח עם רישיון GNU General Public License[3] שפותח על ידי המרכז הגרמני לחקר החלל (DLR) במטרה ליצור כלי המכיל מגוון רחב של כלי עזר למידול של תחבורת רכבים והולכי רגל, הכולל תמיכה במגוון רחב של פורמטים לייצוג תווי הדרך. הכלי מכיל בתוכו כל מיני כלי עזר, כגון: שליטה במהירות כלי התחבורה, שליטה בנתיב הנסיעה (ימין או שמאל), שליטה ברמזורים, כלי להצגת תוצאות הסימולציה, ייצוא קובץ פלט לשימוש ב-ns-3 ועוד.

הכלי תוכנן עם שתי מטרות עיקריות: תוכנה מהירה, ותוכנה פורטבילית (ניתנת להעברה). לכן, הגרסה הראשונה פותחה לשימוש על ידי שורת הפקודה בלבד - ללא ממשק גרפי, והיה צריך להכניס כל פרמטר באופן ידני. זה מגדיל את מהירות הביצוע כי חוסכים את ההדמיה הוויזואלית האיטית. כמו כן, כדי לעמוד במטרות אלו התוכנה חולקה למספר חלקים. לכל חלק מטרה מסוימת ויש להפעיל אותו בנפרד. זה גורם ל- SUMO להיות שונה מחבילות סימולציה אחרות שבהן הקצאות דינאמיות מהמשתמש מתבצעת בתוך הסימולציה עצמה, ולא באמצעות יישום חיצוני כמו ב-SUMO. הפיצול הזה מאפשר הרחבה של כל יישום בתוך החבילה בצורה קלה, כי כל יישום קטן יותר מאשר כלי כבד שמשרת כמה מטרות שונות ביחד. כמו כן, SUMO מאפשר שימוש במבני נתונים מהירים יותר המותאמים למטרה ספציפית לפי כל הרצה, במקום להשתמש במבני נתונים מסובכים ואטומים. למרות כל היתרונות הללו אפילו המפתחים של SUMO מודעים לכך שהשימוש בכלי בצורה בה הוא כתוב לא נוח בהשוואה לחבילות סימולציה אחרות.

כלי זה שימושי בפני עצמו, אך בדרך כלל יש צורך בשימוש בכלי נוסף (למשל, ns-3) על מנת להוסיף לסימולציה אפשרויות של תקשורת בין כלי הרכב.

חלק מן האפשרויות הקיימות ב-SUMO:

- הרצה עם סוגים שונים של כלי רכב
- רחובות בעלי מספר נתיבים ומעבר בין נתיבים
- רמזורים
- צד נסיעה בכביש
- ממשק גרפי מהיר מבוסס OpenGL
- תגובתיות גבוהה (עד 10,0000 כלי רכב במכונה של 1GHz)

[1] https://github.com/Project-OSRM/osrm-backend/blob/master/LICENSE.TXT
[2] https://github.com/fabricesb/DCEP-Sim/blob/master/bonn-motion/bonnmotion-3.0.1/GPL
[3] https://github.com/ec-europa/sumo/blob/master/LICENSE.md

- תומך במודלים המבוססים על בני אדם
- תמיכה גבוהה במערכות הפעלה שונות

בסופו של דבר, השימוש היחיד בכלי הזה בתוך הפרויקט היה לצורך ייצוא של קובץ המכיל את טופולוגיית הבניינים מתוך מפת דרכים נתונה, הסבר לכך יובא בפרק 6. קובץ זה נצרך לצורך חישוב היחלשות אות ה-WiFi במעבר דרך קירות. חישוב זה נעשה על ידי ספרייה חיצונית [13] כאשר יש שימוש בנוסחה המכפילה קבוע מסוים במספר הקירות בהם עובד האות. יש לציין שיש התחשבות גם בהיחלשות הטבעית של האות במעבר באוויר.

4.5 Projection Correction

מכיוון שמדובר במפות של מקומות אמיתיים, SUMO ו-BonnMotion עובדים עם קואורדינטות גיאוגרפית (Latitude ו-Longitude). לעומת זאת ההרצות בפועל מתקיימות בשטח קטן מתוך המפה, ולכן הכלים הללו ממירים את הקואורדינטות הגיאוגרפיות לגודל במטרים. אבל, יש כמה שיטות לבצע המרה שכזו. לכן, מכיוון ש-BonnMotion ו-SUMO משתמשים במערכות קואורדינטות שונות לצורך ייצוא הפלט שלהם, נדרש לבצע צעד נוסף כדי לתאם בין שתי מערכות הקואורדינטות (Projection Correction).

4.6 ns-3

ns-3 [14] הינו סימולטור רשת המיועד בעיקר למטרות מחקר ומטרות לימודיות. מדובר בכלי חינמי המופץ תחת רישיון GPLv2[4]. ns-3 מכיל בתוכו מגוון רחב של ספריות לצרכים שונים ומגוונים. למשל, ספריות הממשות פרוטוקולי תקשורת רבים (גם ברשתות רגילות וגם ברשתות Ad-hoc - ביניהם מימוש של פרוטוקול OLSR), ספריות למימוש סוגים שונים של טכנולוגיות (Ethernet, WiFi, וכדו'), ספריות עזר לצורך יצירה או ייבוא של תנועות במרחב ועוד דברים נוספים רבים.

ns-3 מכיל מגוון רחב של פרוטוקולים שנמצאים בשימוש בעולם האמיתי ומאפשר הטמעה שלהם בסימולציות. בנוסף, תשתית התוכנה של ns-3 מעודדת פיתוח של מודלים לסימולציות מציאותיות מספיק כך שבשילוב עם הפרוטוקולים הקיימים יהיה אפשר להשתמש ב-ns-3 כאמולטור רשת בזמן אמת.

4.7 המנתח (Parser)

כאשר מריצים סימולציות צריך בסופן גם לנתח את המידע שמקבלים מהן. המידע הזה מאפשר לחשב את אחוזי הצלחה של מתקפות או הגנות שאותם רוצים לבדוק. בפרויקט זה קיים מנתח ייעודי כדי לראות בכל סימולציה כמה הודעות מתוך סך ההודעות שנשלחו על ידי הצומת השולח הגיעו ליעדן, וכמה אבדו בדרך. המנתח עובר על כל קבצי הפלט של הסימולציות, ולפי הודעות ייעודיות שמודפסות בכל סימולציה מסדר את הנתונים בצורה שבה אפשר להשוות בין הרצה להרצה. בין אם מדובר על הרצה ללא התקפה או בהרצה עם התקפה ובלי הגנה או בהרצה עם התקפה והגנה וכו'.

4.8 הכלי הסופי - Realistic Network Simulator

הכלי הסופי מחבר מספר תתי-כלים שונים לכלי אחד, כאשר המטרה הסופית היא ליצור סימולציה של כלי רכב ברשת Ad-Hoc שמתקשרים ביניהם באמצעות WiFi (בפרוטוקול OLSR). הצמתים ברשת הם רכבים הנוסעים בצורה אקראית בתוך מפת דרכים נתונה ומתחזקים את הרשת באמצעות שליחת הודעות ביניהם. בעזרת הכלי הסופי אפשר למשל להוסיף צומת זדוני ברשת שמבצע התקפה על זוג רכבים המתקשרים ביניהם, כאשר כל המידע נשמר לצורך חישוב סטטיסטיקות עתידיות.
אפשרות נוספת היא, לשלוט בסוג התנועה של התוקף. למשל, תוקף שעוקב אחרי השולח, תוקף שנע באקראיות ועוד. ישנה אפשרות נוספת להוסיף גם צומת שמנסה לזהות התקפות ברשת של הצומת הזדוני ובכך לבדוק יעילות של הגנות כלשהן כנגד התוקף המדובר. וכל זה בעזרת שינוי בקובץ אחד, דבר המקל על חוקר שזקוק לכלל הכלים הנ"ל.

בשלב ראשון, הכלי מוציא ממפת דרכים נתונה את תוואי הבניינים על מנת ליצור סימולציה המדמה את המציאות. כלומר, היחלשות של האות במעבר בתוך קירות של בניינים. זה נעשה על ידי הכלי SUMO שמסוגל להוציא מפה של בניינים מתוך מפת דרכים נתונה. לאחר שלב זה נוצר קובץ המכיל את טופולוגית הבניינים.

[4] https://github.com/nsnam/ns-3.16-git/blob/master/LICENSE

בשלב שני, מייצרים תנועות אקראיות של רכבים בתוך מפת הדרכים הנתונה למספר הרכבים המבוקש ושומרים אותם לשימוש עתידי - שלב זה נעשה על ידי הכלי BonnMotion.

לאחר מכן, שני הפלטים הנ"ל מיוצאים לפורמט שניתן לקריאה באמצעות ns-3 שבעזרתו מריצים בפועל את הסימולציה כולל התנועות, התקשורת (תוך שימוש בטופולוגית הבניינים לצורך חישוב החלשות האות בפועל) וההתקפות וההגנות במידה וישנן. קוד זה שומר את כל תוצאות הסימולציות בקבצים מיוחדים לצורך ניתוח הנתונים שמתבצע באמצעות סקריפט (Parser) הכתוב בשפת פייתון שיודע להציג את המידע הרלוונטי בקשר ליעילות ההגנות והמתקפות ברשת הנתונה (לפי הודעות ייעודיות שמודפסות בכל סימולציה).

5. הקוד

5.1 המפות והבניינים

אמנם, המפות מיוצרות על ידי כלי אחד (BonnMotion) והבניינים על ידי כלי אחר (SUMO) אבל יש קוד אחד שאחראי על הייצור של שניהם מכיוון ששניהם מחולצים מתוך אותו קובץ מפות בפורמט OSM (Open Street Map). ניתן למצוא קבצים כאלו באינטרנט (לדוגמא, קובץ בשם berlin-latest.osm.pbf) ולהמיר אותם לקובץ osm על ידי הפקודה הבאה:

```
osmconvert ../maps/berlin-latest.osm.pbf -o=berlin-latest.osm
```

נביא כאן את הקוד שאחראי על יצירת קבצי התנועות על פי המפה הנתונה ועל יצירת קובץ הבניינים.

הקוד המובא לעיל הינו גרסה ספציפית של הכלי, ניתן למצוא את כל השינויים בגרסאות הבאות ב-GitHub [15]:

```
1. #!/bin/bash
2. # Vars
3. B1="13.3863801,52.510189,13.3928171,52.5158761"
4. B2=`echo $B1 | tr ',' ' '`
5. OFFSET=10
6.
7. Function createBuildingsFile() {
8.       if [ ! -f ../maps/berlin-latest-${B1}.buildings.xml ]; then
9.             netconvert --remove-edges.isolated --keep-edges.in-geo-
   boundary ${B1} --osm-files ../maps/berlin-latest.osm -o ../maps/berlin-
   latest-${B1}.net.xml
10.                  polyconvert --prune.in-net --osm.keep-full-type --
   net-file ../maps/berlin-latest-${B1}.net.xml --osm-files
   ../maps/berlin-latest.osm --type-file ../maps/buildings.typ.xml -o
   ../maps/berlin-latest-${B1}.buildings.xml
11.            fi
12.      }
13.
14.      function oneRound() {
15.            i=$1
16.            if [ ! -f berlin-latest-${B1}-${N}-${i}.movements.gz ];
   then
17.                  # Run script
18.                  ../bin/bm -f berlin-latest-${B1}-${N}-${i}
   RandomStreet -n ${N} -B ${B2} -u http://localhost:5000 -s 1.5 2 -C 1 -p
   30 -d 300 -o ../maps/berlin-latest.osm.pbf
19.                  retVal=$?
20.                  if [ $retVal -ne 0 ]; then
21.                        echo "ERROR"
22.                        exit 1
23.                  fi
24.            fi
25.            python3 convert.py berlin-latest-${B1}-${N}-
   ${i}.movements.geo.gz ../maps/berlin-latest-${B1}.buildings.xml OFFSET
   > /dev/null
26.            retVal=$?
27.            if [ $retVal -ne 0 ]; then
28.                  echo "ERROR"
29.                  exit 1
30.            fi
31.            ../bin/bm NSFile -f berlin-latest-${B1}-${N}-${i} -b >
   /dev/null
32.            retVal=$?
33.            if [ $retVal -ne 0 ]; then
34.                  echo "ERROR"
35.                  exit 1
36.            fi
```

```
37.             mv berlin-latest-${B1}-${N}-${i}.ns_movements berlin-
   latest-${B1}-${N}-${i}-1.ns_movements
38.             retVal=$?
39.             if [ $retVal -ne 0 ]; then
40.                   echo "ERROR"
41.                   exit 1
42.             fi
43.       }
44.
45.       function runScript () {
46.             R=$1
47.
48.             for ((i=$2;i<$3;i+=1)); do
49.                   #echo "This is run $i" >> $LOGFILE
50.                   # if [ ! -f berlin-latest-${B1}-${N}-${i}.ns_params
   ]; then
51.                   oneRound $i > /dev/null
52.                   local d=`date`
53.                   echo Finished run $i on process $R at $d
54.                   # fi
55.             done
56.             echo Finished all runs at $d
57.       }
58.
59.       # Call function to run on cores
60.       function runOnCores(){
61.             for ((i=0;i<$CORES;i+=1)); do
62.                   FROM=$(( $FIRSTI + $PERCORE * $i ))
63.                   TO=$(( $FROM + $PERCORE ))
64.                   runScript $i $FROM $TO &
65.                   #echo $i $FROM $TO
66.             done
67.             # Leftovers
68.             FROM=$(($FIRSTI + $PERCORE*$CORES))
69.             if [ $TOTALRUNS -gt $FROM ]; then
70.                   #echo $CORES $FROM $TOTALRUNS
71.                   runScript $CORES $FROM $TOTALRUNS
72.             fi
73.             wait
74.       }
75.
76.       set -m # Don\'t lose job control!
77.       # Make sure TOTALRUNS % CORES = 0. Otherwise there would be
   leftover process.
78.       TOTALRUNS=500
79.       CORES=10
80.       FIRSTI=0
81.       PERCORE=$(($TOTALRUNS/$CORES))
82.
83.       createBuildingsFile
84.       for ((N=30; N<= 100; N+=100)); do
85.             runOnCores&
86.             wait
87.       Done
```

המשתנה ששמו B1 בשורה 3 מגדיר את השטח שבו ינועו הרכבים בתוך המפה כאשר הפורמט הוא:

Minimum Longitude, Minimum Latitude, Maximum Longitude, Maximum Latitude

הפונקציה createBuildingsFile אחראית ליצור את קובץ בניינים מתוך המפה הנתונה (הפלט נשמר בקובץ מסוג buildings.xml). הפונקציה oneRound אחראית ליצור קובץ תנועות יחיד שמספרו i מתוך הכמות הכללית של ההרצות.

שורה 19 אחראית לייצר את התנועות על ידי BonnMotion, שורה 27 אחראית לתאם בין הקואורדינטות של התנועות לבין הבניינים (מכיוון ש SUMO ו-BonnMotion עובדים בקואורדינטות שונות). שורה 33 ממירה את קובץ התנועות לפורמט שונה, כדי לאפשר את יבוא התנועות לתוך ns-3.

הקוד בשורות 61 עד 91 אחראי לייצור כמות של קבצי תנועה שונים, השווה לערך של TOTALRUNS בצורה מקבילית בעזרת הליבות של המעבד.

5.2 הסימולציה עצמה

נתחיל בהסבר של מבנה התיקיות של ns-3 ונסביר היכן בדיוק צריך כל קטע קוד להופיע. כאשר מתקינים את הכלי ns-3 כל הקבצים מופיעים בתיקייה בשם ns-allinone-3.24.1 כאשר 3.24.1 מייצג את מספר הגרסה הנוכחית. בתוך תיקייה זו מופיעה תיקייה בשם ns -3.24.1 בהתאם, נכנה תיקייה זו בשם "התיקייה הראשית". בתוך התיקייה הראשית ישנם כמה דברים חשובים:

1. תיקייה בשם src המכילה מגוון ספריות שניתן להשתמש בהן לצורך הרצה של קבצי קוד שנכתב על ידי המשתמש.
2. תיקייה בשם scratch המכילה קבצי קוד שנכתבו על ידי המשתמש לצורך הרצה על ידי ns-3.
3. קובץ בשם waf שתפקידו לבצע הרצות של דברים בפועל על ידי ns-3. למשל, קימפול מחדש של כל הספריות והקודים של ns-3, שינוי פרמטרים פנימיים של ns-3 והרצה של קבצי קוד בפועל.

הפרויקט מורכב מהחלקים הבאים:

חלק אחד הוא העתק של הספרייה olsr שתפקידה לאפשר למשתמש להריץ סימולציות עם הפרוטוקול OLSR - כאשר שינינו את שמה ל-iolsr. הקודים העיקריים למימוש ההתקפה מופיעים בקבצים src/iolsr/model/iolsr-routing-protocol.h ו-src/iolsr/model/iolsr-routing-protocol.cc.

חלק שני הוא קובץ הרצה (main) שמשתמש בספרייה זו לצורך ביצוע של סימולציות בפועל ומופיע בקובץ scratch/stable_network_mod_2.cc.

התוכן של קובץ זה מובא להלן:

```
1. #include "ns3/core-module.h"
2. #include "ns3/network-module.h"
3. #include "ns3/internet-module.h"
4. #include "ns3/iolsr-module.h"
5. #include "ns3/mobility-module.h"
6. #include "ns3/wifi-module.h"
7. #include "ns3/dsdv-module.h"
8. #include "ns3/output-stream-wrapper.h"
9. #include "ns3/netanim-module.h"
10. #include "ns3/udp-client-server-helper.h"
11. #include<string>
12. using namespace ns3;
13.
14. #include "ns3/topology.h"
15. #include <fstream>
16. #include <iterator>
17.
18. NS_LOG_COMPONENT_DEFINE("StableNetworkRouteMod");
19.
20. Bool      isPartOf6Cycle(Ipv4Address      m_mainAddress,      std::map<Ipv4Address,
    std::set<Ipv4Address> > graph) {
21.    if (graph[m_mainAddress].size() != 2) {
22.          return false; // Must have exectly 2 neighbors for cycle
23.    }
```

```
24.    {
25.            // This code doesn't account for fictives...
26.            // Both cases can be united but for sake of simplicity they are seperate.
27.            // Case middle:
28.            bool ok = true;
29.            std::set<Ipv4Address> parts;
30.            std::set<Ipv4Address> level2;
31.            parts.insert(m_mainAddress);
32.            for          (std::set<Ipv4Address>::const_iterator          it          =
    graph[m_mainAddress].begin(); it != graph[m_mainAddress].end(); ++it) {
33.                    const Ipv4Address& u = *it;
34.                    if (graph[u].size() != 2) {
35.                          ok = false;
36.                          break;
37.                    }
38.                    parts.insert(u);
39.                    for (std::set<Ipv4Address>::const_iterator it = graph[u].begin();
    it != graph[u].end(); ++it) {
40.                          if (*it != m_mainAddress) level2.insert(*it);
41.                    }
42.            }
43.            parts.insert(level2.begin(), level2.end());
44.            if (ok && (level2.size() == 2) && (parts.size() == 5)) {
45.                    // Possible middle case
46.                    std::set<Ipv4Address>::const_iterator it = level2.begin();
47.                    std::set<Ipv4Address> &a = graph[*it];
48.                    ++it;
49.                    std::set<Ipv4Address> &b = graph[*it];
50.                    std::set<Ipv4Address> intersect;
51.                    std::set_intersection(a.begin(),   a.end(),   b.begin(),   b.end(),
    std::inserter(intersect, intersect.begin()));
52.
53.                    std::vector<Ipv4Address> intersectionWithoutPrev;
54.                    std::set_difference(intersect.begin(),            intersect.end(),
    parts.begin(),          parts.end(),          std::inserter(intersectionWithoutPrev,
    intersectionWithoutPrev.begin()));
55.                    if (intersectionWithoutPrev.size() > 0) return true;
56.            }
57.
58.            parts.clear();
59.            level2.clear();
60.
61.            // Case side:
62.            ok = true;
63.            Ipv4Address neiWithMany = "0.0.0.0", neiWithTwo = "0.0.0.0", neiOfNei =
    "0.0.0.0";
64.            //Ipv4Address  neiWithMany  =  uint32_t(0),  neiWithTwo  =  uint32_t(0),
    neiOfNei = uint32_t(0);
65.            for          (std::set<Ipv4Address>::const_iterator          it          =
    graph[m_mainAddress].begin(); it != graph[m_mainAddress].end(); ++it) {
66.                    const Ipv4Address& u = *it;
67.                    if (graph[u].size() != 2) {
68.                          if (!ok) break; // If the second one doesn't have rank 2
69.                          ok = false;
```

```
70.                         neiWithMany = u;
71.                         continue;
72.                 }
73.                 neiWithTwo = u;
74.                 std::set<Ipv4Address>::const_iterator itTmp = it;
75.                 if (neiWithMany == "0.0.0.0") neiWithMany = *(++itTmp);
76.                 parts.insert(m_mainAddress);
77.                 parts.insert(neiWithMany);
78.                 parts.insert(neiWithTwo);
79.                 for (std::set<Ipv4Address>::const_iterator it2 = graph[u].begin();
    it2 != graph[u].end(); ++it2) {
80.                         const Ipv4Address& v = *it2;
81.                         if (v == m_mainAddress) continue;
82.                         if (graph[v].size() != 2) return false;
83.                         neiOfNei = v;
84.                         parts.insert(v);
85.                 }
86.                 if (graph[neiWithTwo].count(neiWithMany) > 0) return false;
87.                 if (graph[neiWithMany].count(neiOfNei) > 0) return false;
88.
89.                 std::set<Ipv4Address> ringCandidates;
90.                 // Gather neighbors of neiWithMany
91.                 for       (std::set<Ipv4Address>::const_iterator       it       =
    graph[neiWithMany].begin(); it != graph[neiWithMany].end(); ++it) {
92.                     std::set<Ipv4Address> tmp;
93.                         std::set_union(ringCandidates.begin(),
    ringCandidates.end(),   graph[*it].begin(),   graph[*it].end(),   std::inserter(tmp,
    tmp.begin()));
94.                         ringCandidates = tmp;
95.                 }
96.
97.                 std::set<Ipv4Address> intersect;
98.                 //std::set_intersection(graph[neiWithMany].begin(),
    graph[neiWithMany].end(),      graph[neiOfNei].begin(),      graph[neiOfNei].end(),
    std::inserter(intersect, intersect.begin()));
99.                 std::set_intersection(ringCandidates.begin(),
    ringCandidates.end(),        graph[neiOfNei].begin(),        graph[neiOfNei].end(),
    std::inserter(intersect, intersect.begin()));
100.                        if (intersect.size() > 0) return true;
101.                        return false;
102.                }
103.        }
104.        return false;
105.    }
106.
107.    size_t nodesIn6Ring(NodeContainer* cont) {
108.        // Build directed graph from Hello
109.        std::map<Ipv4Address, std::set<Ipv4Address> > graph;
110.        std::set<Ipv4Address> n;
111.        for (size_t i = 0; i < cont->GetN(); ++i) {
112.                const std::pair<Ipv4Address, std::vector<Ipv4Address> > r = cont-
    >Get(i)->GetObject<RoutingProtocol>()->GetSymNeighbors();
113.                for      (std::vector<Ipv4Address>::const_iterator      it      =
    r.second.begin(); it != r.second.end(); ++it) {
```

```
                        graph[r.first].insert(*it);
                }
                n.insert(r.first);
        }

        size_t count = 0;
        for (std::set<Ipv4Address>::const_iterator it = n.begin(); it != n.end(); ++it) {
                if (isPartOf6Cycle(*it, graph)) ++count;
        }
        return count;
    }
    // Count and print the amount of nodes that have detected themselves to be in a 6-ring
    static void PrintCount6Ring(NodeContainer* cont) {
        size_t count = 0;
        for (size_t i = 0; i < cont->GetN(); ++i) {
                Ptr<RoutingProtocol> pt = cont->Get(i)->GetObject<RoutingProtocol>();
                //NS_LOG_INFO("Node id: " << i << "\tTime: " << Simulator::Now().GetSeconds() << "\tRequireFake? " << pt->RequireFake());
                if (pt->isPartOf6Cycle()) ++count;
        }
        //NS_LOG_INFO("Total fakes required: " << count << " out of " << cont->GetN());
        std::cout << "In 6 ring: " << count << "/" << cont->GetN() << " Inside" << std::endl;
        std::cout << "In 6 ring: " << nodesIn6Ring(cont) << "/" << cont->GetN() << " Outside" << std::endl;
    }

    // Count the nodes that will require fake nodes
    static void PrintCountFakeNodes(NodeContainer* cont) {
        //ns3::iolsr::RoutingProtocol rp;
        unsigned int count = 0;
        for (unsigned int i = 0; i < cont->GetN(); ++i) {
                Ptr<RoutingProtocol> pt = cont->Get(i)->GetObject<RoutingProtocol>();
                //NS_LOG_INFO("Node id: " << i << "\tTime: " << Simulator::Now().GetSeconds() << "\tRequireFake? " << pt->RequireFake());
                if (pt->RequireFake()) ++count;
        }
        NS_LOG_INFO("Total fakes required: " << count << " out of " << cont->GetN());
        std::cout << "Fictive: " << count << "/" << cont->GetN() << std::endl;
    }

    static void PrintMprFraction(NodeContainer* cont) {
        double sum = 0;
        for (unsigned int i = 0; i < cont->GetN(); ++i) {
                Ptr<RoutingProtocol> pt = cont->Get(i)->GetObject<RoutingProtocol>();
                //NS_LOG_INFO("Node id: " << i << "\tTime: " << Simulator::Now().GetSeconds() << "\tFraction: " << pt->FractionOfMpr());
```

```
156.                sum += pt->FractionOfMpr();
157.         }
158.         NS_LOG_INFO("Total fraction of MPR: " << sum / (double)cont->GetN());
159.         NS_LOG_INFO(sum / (double)cont->GetN());
160.         //... NS_LOG_INFO broke. Using cout...
161.         //std::cout << "Total fraction of MPR: " << sum / (double) cont->GetN()
   << std::endl;
162.         std::cout << "MPR: " << (sum / (double)cont->GetN()) << std::endl;
163.      }
164.
165.      static void PrintRiskyFraction(NodeContainer* cont, Ipv4Address ignore =
   Ipv4Address("0.0.0.0")) {
166.         double sum = 0;
167.         for (unsigned int i = 0; i < cont->GetN(); ++i) {
168.                Ptr<RoutingProtocol>          pt          =          cont->Get(i)-
   >GetObject<RoutingProtocol>();
169.                sum += pt->FractionOfNodesMarkedAsRisky(ignore);
170.         }
171.         std::cout << "Risky%: " << (sum / double(cont->GetN())) << std::endl;
172.      }
173.
174.
175.      static void PrintTcPowerLevel(NodeContainer* cont) {
176.         double sumLsr = 0;
177.         double sumOlsr = 0;
178.         for (unsigned int i = 0; i < cont->GetN(); ++i) {
179.                Ptr<RoutingProtocol>          pt          =          cont->Get(i)-
   >GetObject<RoutingProtocol>();
180.                sumLsr += pt->tcPowerLevel(true);
181.                sumOlsr += pt->tcPowerLevel(false);
182.         }
183.         //NS_LOG_INFO("TC Level: " << sumOlsr / (double) cont->GetN() << " (lsr:
   " << sumLsr / (double) cont->GetN() << ")");
184.         std::cout << "TC Level: " << sumOlsr / (double)cont->GetN() << " \nlsr: "
   << sumLsr / (double)cont->GetN() << "\n";
185.      }
186.
187.      static void ExecuteIsolationAttack(NodeContainer* cont) {
188.         // Make it pick a node at random later...
189.         Ipv4Address    target    =    cont->Get(2)->GetObject<RoutingProtocol>()-
   >ExecuteIsolationAttack();
190.         target.IsBroadcast(); // Kill the unused warning
191.         //NS_LOG_INFO("Executing node isolation attack on: " << target);
192.         //NS_LOG_INFO("Attacker address: " << cont->Get(2)->GetObject<Ipv4>()-
   >GetAddress(1,0));
193.         //NS_LOG_INFO("Victim    test:  "  <<  cont->Get(25)->GetObject<Ipv4>()-
   >GetAddress(1,0));
194.         //std::cout << "Executing node isolation attack on: " << target <<
   std::endl;
195.         //std::cout << "Attacker address: " << cont->Get(2)->GetObject<Ipv4>()-
   >GetAddress(1,0) << std::endl;
196.      }
197.
198.      static void ExecuteIsolationAttackMassive(NodeContainer* cont) {
```

```
199.         // Let the first 30% nodes attack and see what happens
200.         for (uint32_t i = 5; i < cont->GetN() * 0.3 + 5; ++i) {
201.                //                     cont->Get(i)->GetObject<RoutingProtocol>()-
   >ExecuteIsolationAttack();
202.                cont->Get(i)->GetObject<RoutingProtocol>()-
   >ExecuteIsolationAttack(Ipv4Address("10.0.0.1"));
203.         }
204.      }
205.
206.      static  void  ExecuteIsolationAttackBug(NodeContainer*  nodes,  Ipv4Address
   target) {
207.         for (uint32_t i = 2; i < nodes->GetN(); ++i) {
208.                if                   (nodes->Get(i)->GetObject<RoutingProtocol>()-
   >isItNeighbor(target)) {
209.                       nodes->Get(i)->GetObject<RoutingProtocol>()-
   >ExecuteIsolationAttack(target);
210.                       break;
211.                }
212.         }
213.      }
214.
215.      static void PercentageWithFullConnectivity(NodeContainer* cont) {
216.         uint32_t count = 0;
217.         for (uint32_t i = 0; i < cont->GetN(); ++i) {
218.                uint32_t         routingTableSize         =         cont->Get(i)-
   >GetObject<RoutingProtocol>()->getRoutingTableSize();
219.                if (routingTableSize == cont->GetN() - 1) {
220.                       ++count;
221.                }
222.         }
223.         double result = count / (double)cont->GetN();
224.         std::cout << "Routing Precentage: " << result << "\n";
225.         Simulator::Schedule(Seconds(10), &PercentageWithFullConnectivity, cont);
226.      }
227.
228.      static void ActivateFictiveDefence(NodeContainer* cont) {
229.         for (unsigned int i = 0; i < cont->GetN(); ++i) {
230.                cont->Get(i)->GetObject<RoutingProtocol>()-
   >activateFictiveDefence();
231.         }
232.      }
233.
234.      static void PrintReceivedPackets(Ptr<UdpServer> udpServer) {
235.         uint32_t count = udpServer->GetReceived();
236.         std::cout << "Received Udp Packets: " << count << std::endl;
237.      }
238.
239.      static void AssertConnectivity(NodeContainer* cont) {
240.         //for (unsigned int i=0; i < cont->GetN(); ++i){
241.         //  if  (cont->Get(i)->GetObject<RoutingProtocol>()->getRoutingTableSize()
   != cont->GetN() - 1) {
242.         //        Simulator::Stop();
243.         // }
244.         //}
```

```
245.         if (cont->Get(0)->GetObject<RoutingProtocol>()->getRoutingTableSize() !=
   cont->GetN() - 1) {
246.                 Simulator::Stop();
247.         }
248.      }
249.      static void AbortOnNeighbor(Ptr<Node> node, Ipv4Address address) {
250.         if (node->GetObject<RoutingProtocol>()->isItUpToTwoHop(address)) {
251.                 // New
252.                 std::cout << "AbortOnNeighbor\n";
253.                 Simulator::Stop();
254.         }
255.      }
256.
257.      static void TrackTarget(Ptr<Node> target, Ptr<Node> tracker) {
258.         Vector vec = target->GetObject<MobilityModel>()->GetPosition();
259.         vec.x += 8;
260.         vec.y += 0;
261.         tracker->GetObject<MobilityModel>()->SetPosition(vec);
262.      }
263.
264.      static void PrintTables(Ptr<Node> n, std::string fname) {
265.         std::ofstream o;
266.         o.open((std::string("_TwoHop") + fname + std::string(".txt")).c_str());
267.         const    TwoHopNeighborSet    &two    =    n->GetObject<RoutingProtocol>()-
   >getTwoHopNeighborSet();
268.         for (TwoHopNeighborSet::const_iterator it = two.begin(); it != two.end();
   ++it) {
269.                 o << *it << "\n";
270.         }
271.         o.close();
272.         o.open((std::string("_Topology") + fname + std::string(".txt")).c_str());
273.         const     TopologySet     &tp     =     n->GetObject<RoutingProtocol>()-
   >getTopologySet();
274.         for (TopologySet::const_iterator it = tp.begin(); it != tp.end(); ++it) {
275.                 o << *it << "\n";
276.         }
277.         o.close();
278.         o.open((std::string("_Neighbor") + fname + std::string(".txt")).c_str());
279.         const     NeighborSet     &nei     =     n->GetObject<RoutingProtocol>()-
   >getNeighborSet();
280.         for (NeighborSet::const_iterator it = nei.begin(); it != nei.end(); ++it)
   {
281.                 o << *it << "\n";
282.         }
283.         o.close();
284.      }
285.
286.      static void IneffectiveNeighorWrite(NodeContainer *cont) {
287.         std::ofstream o;
288.         o.open("_mpr.txt");
289.         for (unsigned int i = 0; i < cont->GetN(); ++i) {
290.                 const  MprSet  mpr  =  cont->Get(i)->GetObject<RoutingProtocol>()-
   >getMprSet();
```

```
291.                o << cont->Get(i)->GetObject<Ipv4>()->GetAddress(1, 0).GetLocal()
   << ":";
292.                for (MprSet::const_iterator it = mpr.begin(); it != mpr.end();
   ++it) {
293.                       o << *it << ",";
294.                }
295.                o << "\n";
296.         }
297.         o.close();
298.         o.open("_neighbors.txt");
299.         for (unsigned int i = 0; i < cont->GetN(); ++i) {
300.                const       NeighborSet       neighbor       =       cont->Get(i)-
   >GetObject<RoutingProtocol>()->getNeighborSet();
301.                o << cont->Get(i)->GetObject<Ipv4>()->GetAddress(1, 0).GetLocal()
   << ":";
302.                for (NeighborSet::const_iterator it = neighbor.begin(); it !=
   neighbor.end(); ++it) {
303.                       o << it->neighborMainAddr << ",";
304.                }
305.                o << "\n";
306.         }
307.         o.close();
308.      }
309.
310.
311.      /*
312.      static void PrintTopologySet(NodeContainer* cont)
313.      {
314.         ns3::iolsr::RoutingProtocol rp;
315.         for (unsigned int i=0;i<cont->GetN();i++)
316.         {
317.                Ptr<RoutingProtocol>          pt          =          cont->Get(i)-
   >GetObject<RoutingProtocol>();
318.                NS_LOG_INFO("Node      id:      "<<i<<"                  "<<"Time:
   "<<Simulator::Now().GetSeconds());
319.                pt->PrintTopologySet();
320.         }
321.      }
322.      */
323.
324.
325.
326.
327.      int main(int argc, char *argv[]) {
328.         // Time
329.         Time::SetResolution(Time::NS);
330.
331.         // Variables
332.         uint32_t nNodes = 100;
333.         double dMaxGridX = 500.0;
334.         uint32_t nMaxGridX = 500;
335.         double dMaxGridY = 500.0;
336.         uint32_t nMaxGridY = 500;
337.         bool bMobility = false;
```

```
338.         uint32_t nProtocol = 0;
339.         double dSimulationSeconds = 301.0;
340.         uint32_t nSimulationSeconds = 301;
341.         std::string mProtocolName = "Invalid";
342.         bool bSuperTransmission = false;
343.         bool bPrintAll = false;
344.         bool bPrintFakeCount = false;
345.         bool bPrint6RingCount = false;
346.         bool bPrintMprFraction = false;
347.         bool bPrintRiskyFraction = false;
348.         bool bIsolationAttack = false;
349.         bool bIsolationAttackBug = false;
350.         bool bIsolationAttackNeighbor = false;
351.         bool bEnableFictive = false;
352.         bool bHighRange = false;
353.         bool bPrintTcPowerLevel = false;
354.         bool bNeighborDump = false;
355.         bool bIsolationAttackMassive = false;
356.         bool bConnectivityPrecentage = false;
357.         bool bUdpServer = false;
358.
359.         // Added variables
360.         std::string paramsFile = "PATH/???.ns_params";
361.         //std::string bldgFile = "PATH/???.buildings.xml";
362.         std::string traceFile = "PATH/???.ns_movements";
363.
364.         // Parameters from command line
365.         CommandLine cmd;
366.         //cmd.AddValue("nNodes", "Number of nodes in the simulation", nNodes);
367.         //cmd.AddValue("nMaxGridX", "X of the simulation rectangle", nMaxGridX);
368.         //cmd.AddValue("nMaxGridY", "Y of the simulation rectangle", nMaxGridY);
369.         //cmd.AddValue("nSimulationSeconds",  "Amount  of  seconds  to  run  the
   simulation", nSimulationSeconds);
370.         // New parameter
371.         cmd.AddValue("paramsFile", "Ns2 movement params file", paramsFile);
372.         //cmd.AddValue("bldgFile", "SUMO buildings file", bldgFile);
373.         cmd.AddValue("traceFile", "Ns2 movement trace file", traceFile);
374.
375.         cmd.AddValue("bMobility",  "Delcares  whenever  there  is  movement  in  the
   network", bMobility);
376.         cmd.AddValue("nProtocol", "IOLSR=0, DSDV=1", nProtocol);
377.         cmd.AddValue("bSuperTransmission",  "Transmission  boost  to  node  X?",
   bSuperTransmission);
378.         cmd.AddValue("bPrintAll", "Print routing table for all hops", bPrintAll);
379.         cmd.AddValue("bPrintFakeCount",  "Print  amount  of  fake  nodes  required",
   bPrintFakeCount);
380.         cmd.AddValue("bPrint6RingCount",  "Print  amount  of  nodes  in  6  Ring",
   bPrint6RingCount);
381.         cmd.AddValue("bPrintMprFraction",     "Print     fraction     of     MPR",
   bPrintMprFraction);
382.         cmd.AddValue("bPrintRiskyFraction",    "Print    fraction    of    risky",
   bPrintRiskyFraction);
383.         cmd.AddValue("bPrintTcPowerLevel",     "Print     average     TC     size",
   bPrintTcPowerLevel);
```

```
384.         cmd.AddValue("bIsolationAttack", "Execute isolation attack by a node",
   bIsolationAttack);
385.         cmd.AddValue("bIsolationAttackNeighbor", "Execute isolation attack by a
   random neighbor", bIsolationAttackNeighbor);
386.         cmd.AddValue("bIsolationAttackMassive", "Execute isolation attack by many
   nodes", bIsolationAttackMassive);
387.         cmd.AddValue("bEnableFictive",   "Activate   fictive   defence   mode",
   bEnableFictive);
388.         cmd.AddValue("bHighRange", "Higher wifi range", bHighRange);
389.         cmd.AddValue("bNeighborDump", "Neighbor dump", bNeighborDump);
390.         cmd.AddValue("bConnectivityPrecentage",  "Print  connectivity  precetage
   every X seconds", bConnectivityPrecentage);
391.         cmd.AddValue("bIsolationAttackBug",  "Have  an  attacker  stick  to  it's
   target", bIsolationAttackBug);
392.         cmd.AddValue("bUdpServer", "Try to send Udp packets from random nodes to
   the attacked node", bUdpServer);
393.         cmd.Parse(argc, argv);
394.         // New way of reading parameters
395.         std::ifstream params;
396.         std::string dummy;
397.         params.open(paramsFile.c_str());
398.         if (params.is_open()) {
399.                 params >> dummy >> dummy >> nMaxGridX >> dummy;
400.                 params >> dummy >> dummy >> nMaxGridY >> dummy;
401.                 params >> dummy >> dummy >> nNodes;
402.                 params >> dummy >> dummy >> nSimulationSeconds;
403.                 params.close();
404.
405.                 // Round up the floating point
406.                 nMaxGridX++;
407.                 nMaxGridY++;
408.
409.                 nMaxGridY += 8;
410.         }
411.         else {
412.                 return 1;
413.         }
414.
415.         if (nSimulationSeconds > 10.0) dSimulationSeconds = nSimulationSeconds;
   // Force minimum time
416.         if (nMaxGridX > 10.0) dMaxGridX = nMaxGridX; // Force minimum size. Revert
   to default.
417.         if (nMaxGridY > 10.0) dMaxGridY = nMaxGridY; // Force minimum size. Revert
   to default.
418.
419.         // Build network
420.         NodeContainer nodes;
421.         nodes.Create(nNodes);
422.
423.         // Add wifi
424.         WifiHelper wifi;
425.         //wifi.SetStandard(WIFI_PHY_STANDARD_80211g);
426.         YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();
427.         YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();
```

```
428.         NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default();
429.         if (bHighRange) {
430.                 wifiPhy.Set("TxGain", DoubleValue(20));
431.                 //wifiPhy.Set("TxGain", DoubleValue(12.4));
432.
    //wifiChannel.AddPropagationLoss("ns3::RangePropagationLossModel",      "MaxRange",
    DoubleValue(250));
433.         }
434.         else {
435.                 wifiChannel.AddPropagationLoss("ns3::RangePropagationLossModel",
    "MaxRange", DoubleValue(100));
436.         }
437.
438.         // Load buildings topology
439.         Topology::LoadBuildings(bldgFile);
440.         wifiChannel.AddPropagationLoss
    ("ns3::ObstacleShadowingPropagationLossModel");
441.
442.         wifiPhy.SetChannel(wifiChannel.Create());
443.         wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager");
444.         wifiMac.SetType("ns3::AdhocWifiMac");
445.         //wifiChannel.AddPropagationLoss       ("ns3::RangePropagationLossModel",
    "MaxRange", DoubleValue (105));
446.         NetDeviceContainer adhocDevices = wifi.Install(wifiPhy, wifiMac, nodes);
447.
448.         // Rig Node for huge wifi boost
449.         if (bSuperTransmission) {
450.                 NodeContainer superNodes;
451.                 superNodes.Create(1);
452.                 wifiPhy.Set("RxGain", DoubleValue(500.0));
453.                 wifiPhy.Set("TxGain", DoubleValue(500.0));
454.                 NetDeviceContainer superDevices = wifi.Install(wifiPhy, wifiMac,
    superNodes);
455.                 adhocDevices.Add(superDevices);
456.                 nodes.Add(superNodes);
457.         }
458.         // Install IOLSR / DSDV
459.         IOlsrHelper iolsr;
460.         DsdvHelper dsdv;
461.         Ipv4ListRoutingHelper routeList;
462.         InternetStackHelper internet;
463.         std::stringstream tmpStringStream;
464.         std::string fName = "Stable_Network_Stream";
465.         fName += "_n";
466.         tmpStringStream << nNodes;
467.         fName += tmpStringStream.str();
468.         tmpStringStream.str("");
469.         fName += "_x";
470.         tmpStringStream << nMaxGridX;
471.         fName += tmpStringStream.str();
472.         tmpStringStream.str("");
473.         fName += "_y";
474.         tmpStringStream << nMaxGridY;
475.         fName += tmpStringStream.str();
```

```
476.         tmpStringStream.str("");
477.         fName += "_r";
478.         tmpStringStream << RngSeedManager::GetRun();
479.         fName += tmpStringStream.str();
480.         tmpStringStream.str("");
481.         fName += ".txt";
482.         //Ptr<OutputStreamWrapper>                    stream                    =
   Create<OutputStreamWrapper>("Stable_Network_Stream_Run",std::ios::out);
483.         //Ptr<OutputStreamWrapper>                    stream                    =
   Create<OutputStreamWrapper>(fName,std::ios::out);
484.
485.         switch (nProtocol) {
486.         case 0:
487.                 routeList.Add(iolsr, 100);
488.                 if (!bPrintAll) {
489.                         //iolsr.PrintRoutingTableEvery(Seconds(10.0), nodes.Get(1),
   stream);
490.                 }
491.                 else {
492.                         Ptr<OutputStreamWrapper>              stream              =
   Create<OutputStreamWrapper>(fName, std::ios::out);
493.                         iolsr.PrintRoutingTableAllEvery(Seconds(10.0), stream);
494.                 }
495.                 break;
496.         case 1:
497.                 routeList.Add(dsdv, 100);
498.                 if (!bPrintAll) {
499.                         //dsdv.PrintRoutingTableEvery(Seconds(10.0),  nodes.Get(1),
   stream);
500.                 }
501.                 else {
502.                         Ptr<OutputStreamWrapper>              stream              =
   Create<OutputStreamWrapper>(fName, std::ios::out);
503.                         dsdv.PrintRoutingTableAllEvery(Seconds(10.0), stream);
504.                 }
505.                 break;
506.         default:
507.                 NS_FATAL_ERROR("Invalid routing protocol chosen " << nProtocol);
508.                 break;
509.         }
510.         internet.SetRoutingHelper(routeList);
511.         internet.Install(nodes);
512.
513.         // Install IP
514.         Ipv4AddressHelper addresses;
515.         addresses.SetBase("10.0.0.0", "255.0.0.0");
516.         Ipv4InterfaceContainer interfaces;
517.         interfaces = addresses.Assign(adhocDevices);
518.
519.         // Install mobility
520.         MobilityHelper mobility;
521.         Ptr<UniformRandomVariable>                  randomGridX                  =
   CreateObject<UniformRandomVariable>();
```

```
522.         Ptr<UniformRandomVariable>                 randomGridY                 =
   CreateObject<UniformRandomVariable>();
523.         randomGridX->SetAttribute("Min", DoubleValue(0));
524.         randomGridX->SetAttribute("Max", DoubleValue(dMaxGridX));
525.         randomGridY->SetAttribute("Min", DoubleValue(0));
526.         randomGridY->SetAttribute("Max", DoubleValue(dMaxGridY));
527.
528.         Ptr<RandomRectanglePositionAllocator>          taPositionAlloc          =
   CreateObject<RandomRectanglePositionAllocator>();
529.         taPositionAlloc->SetX(randomGridX);
530.         taPositionAlloc->SetY(randomGridY);
531.         mobility.SetPositionAllocator(taPositionAlloc);
532.
533.         if (bMobility) {
534.                /*mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
535.                             "Bounds", RectangleValue (Rectangle (0, dMaxGridX, 0,
   dMaxGridY)),
536.                             //"Speed",
   StringValue("ns3::UniformRandomVariable[Min=0|Max=2.0]"),
537.                             "Speed",
   StringValue("ns3::UniformRandomVariable[Min=1.5|Max=2.0]"),
538.                             //"Speed",
   StringValue("ns3::UniformRandomVariable[Min=9.5|Max=10.0]"),
539.                             "Time", TimeValue(Seconds(3.0)),
540.                             "Mode",
   EnumValue(RandomWalk2dMobilityModel::MODE_TIME));*/
541.
542.                             // Install Mobility Model from bonnmotion tool
543.                Ns2MobilityHelper ns2 = Ns2MobilityHelper(traceFile);
544.                ns2.Install(nodes.Begin(), nodes.End());
545.         }
546.         else {
547.                mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
548.                mobility.Install(nodes);
549.         }
550.
551.         if (bUdpServer) {
552.                UdpServerHelper udpServerHelper(80);
553.                ApplicationContainer apps = udpServerHelper.Install(nodes.Get(0));
554.                Ptr<UdpServer> udpServer = udpServerHelper.GetServer();
555.                UdpClientHelper udpClientHelper(Ipv4Address("10.0.0.1"), 80);
556.                udpClientHelper.SetAttribute("Interval", TimeValue(Seconds(7)));
557.                udpClientHelper.SetAttribute("MaxPackets", UintegerValue(18));
558.
559.
560.                // Choose random node to become sender
561.                //Ptr<UniformRandomVariable>                 rnd                 =
   CreateObject<UniformRandomVariable> ();
562.                //rnd->SetAttribute ("Min", DoubleValue (3));
563.                //rnd->SetAttribute ("Max", DoubleValue (nodes.GetN()-1));
564.
565.
566.                // Install UdpClient on said node
567.                //uint32_t sendingNode = rnd->GetInteger();
```

```
568.                uint32_t sendingNode = 1;
569.                apps.Add(udpClientHelper.Install(nodes.Get(sendingNode)));
570.                //apps.Add(udpClientHelper.Install(nodes.Get(1)));
571.                apps.Start(Seconds(60));
572.                apps.Stop(Seconds(1024));
573.                if (true) {
574.                       for (size_t i = 30; i < nSimulationSeconds - 5; i = i + 1)
    {
575.                              Simulator::Schedule(Seconds(i),     &AbortOnNeighbor,
    nodes.Get(sendingNode), Ipv4Address("10.0.0.1"));
576.                       }
577.                }
578.                Simulator::Schedule(Seconds(250),           &PrintReceivedPackets,
    udpServer);
579.         }
580.
581.         if (bPrintFakeCount) {
582.                //Ptr<OutputStreamWrapper>                  wrap                  =
    Create<OutputStreamWrapper>("StableNetwork-Mod-FakeNodes.txt", ios::out);
583.                Simulator::Schedule(Seconds(120), &PrintCountFakeNodes, &nodes);
584.                Simulator::Schedule(Seconds(240), &PrintCountFakeNodes, &nodes);
585.         }
586.         if (bPrintMprFraction) {
587.                Simulator::Schedule(Seconds(120), &PrintMprFraction, &nodes);
588.                Simulator::Schedule(Seconds(240), &PrintMprFraction, &nodes);
589.         }
590.         if (bPrintRiskyFraction) {
591.                Ipv4Address ignore = Ipv4Address("0.0.0.0");
592.                if (bIsolationAttackBug) ignore = Ipv4Address("10.0.0.3");
593.                Simulator::Schedule(Seconds(60),    &PrintRiskyFraction,    &nodes,
    ignore);
594.                Simulator::Schedule(Seconds(240),   &PrintRiskyFraction,   &nodes,
    ignore);
595.         }
596.         if (bPrintTcPowerLevel) {
597.                Simulator::Schedule(Seconds(120), &PrintTcPowerLevel, &nodes);
598.                Simulator::Schedule(Seconds(240), &PrintTcPowerLevel, &nodes);
599.         }
600.         if (bIsolationAttack) {
601.                Simulator::Schedule(Seconds(8), &ExecuteIsolationAttack, &nodes);
602.         }
603.         if (bIsolationAttackMassive) {
604.                Simulator::Schedule(Seconds(8),    &ExecuteIsolationAttackMassive,
    &nodes);
605.         }
606.         if (bIsolationAttackBug) {
607.
608.                nodes.Get(2)->GetObject<RoutingProtocol>()-
    >ExecuteIsolationAttack(Ipv4Address("10.0.0.1"));
609.                Vector     vec     =     nodes.Get(0)->GetObject<MobilityModel>()-
    >GetPosition();
610.                vec.x += 8;
611.                vec.y += 0;
612.                nodes.Get(2)->GetObject<MobilityModel>()->SetPosition(vec);
```

```
613. //DynamicCast<YansWifiPhy>(DynamicCast<WifiNetDevice>(nodes.Get(2)->GetDevice(0))->GetPhy())->SetTxGain(1.2);
614.
615.                 for (size_t i = 1; i < nSimulationSeconds; ++i) {
616.                        Simulator::Schedule(Seconds(i), &TrackTarget, nodes.Get(0), nodes.Get(2));
617.                 }
618.          }
619.          if (bIsolationAttackNeighbor) {
620.                 Simulator::Schedule(Seconds(8),        &ExecuteIsolationAttackBug, &nodes, Ipv4Address("10.0.0.1"));
621.          }
622.          if (bEnableFictive) {
623.                 Simulator::Schedule(Seconds(0), &ActivateFictiveDefence, &nodes);
624.                 //Simulator::Schedule(Seconds    (121),    &ActivateFictiveDefence, &nodes);
625.          }
626.          if (bNeighborDump) {
627.                 Simulator::Schedule(Seconds(120),        &IneffectiveNeighorWrite, &nodes);
628.          }
629.          bool bAssertConnectivity = false;
630.          if (bAssertConnectivity) {
631.                 // Assert connectivity
632.                 Simulator::Schedule(Seconds(119), &AssertConnectivity, &nodes);
633.          }
634.          if (bConnectivityPrecentage) {
635.                 Simulator::Schedule(Seconds(10),  &PercentageWithFullConnectivity, &nodes);
636.          }
637.
638.          if (bPrint6RingCount) {
639.                 //Ptr<OutputStreamWrapper>                 wrap                 = Create<OutputStreamWrapper>("StableNetwork-Mod-FakeNodes.txt", ios::out);
640.                 Simulator::Schedule(Seconds(120), &PrintCount6Ring, &nodes);
641.                 Simulator::Schedule(Seconds(240), &PrintCount6Ring, &nodes);
642.          }
643.
644.          if (false) {
645.                 Simulator::Schedule(Seconds(251),    &PrintTables,    nodes.Get(0), std::string("V"));
646.                 Simulator::Schedule(Seconds(251),    &PrintTables,    nodes.Get(2), std::string("A"));
647.                 Simulator::Schedule(Seconds(251),        &IneffectiveNeighorWrite, &nodes);
648.          }
649.          // Print Topology Set
650.          //Simulator::Schedule(Seconds (35) , &PrintTopologySet, &nodes);
651.
652.          // Animation
653.          /*AnimationInterface anim ("Stable_Network_animation.xml");
654.          anim.SetMobilityPollInterval (Seconds (1));
655.          for(uint32_t i=0; i<nNodes;i++)
```

```
656.                anim.UpdateNodeSize (i, 10, 10);*/
657.
658.                //AnimationInterface anim("Stable_Network_animation.xml");
659.                //anim.UpdateNodeColor(nodes.Get(0), 0, 255, 255);
660.                //anim.UpdateNodeColor(nodes.Get(1), 0, 255, 0);
661.                //anim.UpdateNodeColor(nodes.Get(2), 0, 0, 255);
662.                //anim.SetMobilityPollInterval(Seconds(1));
663.
664.                // Pcap
665.                //wifiPhy.EnablePcapAll ("Stable_Network_");
666.
667.                // Run simulation
668.         NS_LOG_INFO("Run simulation.");
669.         Simulator::Stop(Seconds(dSimulationSeconds));
670.
671.         Simulator::Run();
672.         Simulator::Destroy();
673.
674.         return 0;
675.      }
```

בקובץ לעיל מופיעות מספר פונקציות, הפונקציות בהן השתמשנו לצורך ההרצות שיובאו בהמשך (פרק 9) הן:

```
static void ExecuteIsolationAttackBug(NodeContainer* nodes, Ipv4Address target)
static void ActivateFictiveDefence(NodeContainer* cont)
static void AssertConnectivity(NodeContainer* cont)
static void AbortOnNeighbor(Ptr<Node> node, Ipv4Address address)
int main(int argc, char *argv[])
```

נביא כאן הסבר של הפונקציה הראשית ששמה main ושל הפונקציות הנוספות בהן השתמשנו בפועל. שורות 365 עד 393 אחראיות על הגדרת הפרמטרים שהמשתמש יכול לשנות בקוד.

הרצה של הפקודה:

```
waf --run 'stable_network_mod_2 --PrintHelp'
```

תאפשר לראות רשימה של כל הפרמטרים המוגדרים בקוד והסבר שלהם כמו בתמונה הבאה:

```
nmontag@netlab-307:~/ns-allinone-3.24.1/ns-3.24.1$ ./waf --run 'stable_network_mod_2 --PrintHelp'
Waf: Entering directory `/data/home/nmontag/ns-allinone-3.24.1/ns-3.24.1/build'
Waf: Leaving directory `/data/home/nmontag/ns-allinone-3.24.1/ns-3.24.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.051s)
stable_network_mod_2 [Program Arguments] [General Arguments]

Program Arguments:
    --paramsFile:                Ns2 movement params file [PATH/???.ns_params]
    --bldgFile:                  SUMO buildings file [PATH/???.buildings.xml]
    --traceFile:                 Ns2 movement trace file [PATH/???.ns_movements]
    --bMobility:                 Delcares whenever there is movement in the network [false]
    --nProtocol:                 IOLSR=0, DSDV=1 [0]
    --bSuperTransmission:        Transmission boost to node X? [false]
    --bPrintAll:                 Print routing table for all hops [false]
    --bPrintFakeCount:           Print amount of fake nodes required [false]
    --bPrint6RingCount:          Print amount of nodes in 6 Ring [false]
    --bPrintMprFraction:         Print fraction of MPR [false]
    --bPrintRiskyFraction:       Print fraction of risky [false]
    --bPrintTcPowerLevel:        Print average TC size [false]
    --bIsolationAttack:          Execute isolation attack by a node [false]
    --bIsolationAttackNeighbor:  Execute isolation attack by a random neighbor [false]
    --bIsolationAttackMassive:   Execute isolation attack by many nodes [false]
    --bEnableFictive:            Activate fictive defence mode [false]
    --bHighRange:                Higher wifi range [false]
    --bNeighborDump:             Neighbor dump [false]
    --bConnectivityPrecentage:   Print connectivity precetage every X seconds [false]
    --bIsolationAttackBug:       Have an attacker stick to it's target [false]
    --bUdpServer:                Try to send Udp packets from random nodes to the attacked node [false]

General Arguments:
    --PrintGlobals:              Print the list of globals.
    --PrintGroups:               Print the list of groups.
    --PrintGroup=[group]:        Print all TypeIds of group.
    --PrintTypeIds:              Print all TypeIds.
    --PrintAttributes=[typeid]:  Print all attributes of typeid.
    --PrintHelp:                 Print this help message.
```

איור 3: רשימת הפרמטרים המוגדרים בקוד, והסבר שלהם

שורות 395 עד 413 אחראיות על קריאה של פרמטרים נוספים מתוך קובץ מסוג ns_params שנוצר על ידי BonnMotion.

```
set val(x) 2542.292266838718
set val(y) 2455.1326024094597
set val(nn) 30
set val(duration) 300.0
```

איור 4: דוגמה לקובץ מסוג ns_params

שורות 420-421 מגדירות אובייקט המכיל את כל הצמתים שבסימולציה הנוכחית כאשר מספר הצמתים הוא nNodes. שורות 424 עד 433 מגדירות את הפרמטרים הרלוונטיים לתקשורת WiFi בין הצמתים כאשר בפועל הדגל bHighRange היה מופעל בהרצות שביצענו על מנת לחזק את עוצמת האות ומרחק הקליטה המקסימאלי. שורות 458 עד 511 מגדירות את פרוטוקול התקשורת והאינטרנט כאשר בפועל השתמשנו בפרוטוקול OLSR. שורות 513 עד 517 אחראיות להגדרת כתובת IP לכל צומת ברשת מהתווך 10.0.0.0/8. שורות 528 עד 549 מגדירות את תנועת הצמתים כאשר אם הדגל bMobility דלוק הצמתים נעים במפה הנתונה על ידי traceFile (הפלט מן הפרק הקודם) ואם הוא כבוי הצמתים עומדים במקום. שורות 551 עד 579 אחראיות להגדיר תקשורת שרת לקוח בין צמתים 1 ו-0 בהתאמה כאשר השרת שולח מקסימום של 18 הודעות. דגל נוסף שנעשה בו שימוש הוא bIsolationAttackBug בשורות 606 עד 618 שאחראי להגדיר שצומת שמספרה 2 תתקוף את צומת מספר 0 באופן כזה שהיא תמיד תמצא 8 מטרים מימינה, זה נעשה על ידי הפונקציה ExecuteIsolationAttackBug שדואגת לבצע זאת. והדגל bEnableFictive אחראי להפעיל את ההרצה עם הגנה כנגד המתקפה הזו זה נעשה על ידי הפונקציה ActivateFictiveDefence שדואגת לבצע זאת.

הפלט הסופי של הקוד הנתון הוא אחד משתי האפשרויות הבאות:

- Received Udp Packets: X

כאשר X הוא מספר החבילות מתוך 18 שהתקבלו אצל השרת

- AbortOnNeighbor

במידה והתוקף הוא שכן ישיר (OneHop) של הצומת השולח ולכן לא ההגנה לא תוכל לעזור במקרה זה

5.3 המנתח (Parser)

הקוד של המנתח הוא:

```
1. #!/usr/bin/env python3
2. import sys, re
3. import numpy
4. if (len(sys.argv) == 1):
5.       print("Usage is " + sys.argv[0] + " file.txt")
6.       exit()
7.
8. with open(sys.argv[1], 'r') as f:
9.       contents = f.read()
10.      it = re.finditer('This is[ \w]*$', contents, re.MULTILINE)
11.      regex = re.compile('^This is[ \w]*$', re.MULTILINE)
12.      tmp = regex.sub('', contents)
13.      tmp = re.sub('[a-zA-Z:\n]', '', tmp)
14.      tmp = re.sub('^\s+', '', tmp)
15.      tmp = re.split('[\s]+', tmp)
16.      lst = list()
17.      for x in tmp:
18.            lst.append(int(x))
19.      #print(sum(lst)/len(lst))
20.      print("File: " + sys.argv[1])
21.      print("Average: {}".format(numpy.mean(lst)))
22.      print("Std: {}".format(numpy.std(lst)))
23.      print("Median: {}".format(numpy.median(lst)))
24.      print("Source: {}".format(len(lst)))
```

המנתח מקבל קבצים המכילים את הפלט של כל ההרצות ומחשב את המספר הממוצע של חבילות שהגיעו, השונות והחציון. בעזרת מידע זה אפשר להשוות בין מקרים בהם מתבצעת התקפה למקרים בהם לא, ובין מקרים בהם מתבצעת הגנה למקרים בהם לא.

5.4 הכלי השלם

כפי שכבר הוזכר לעיל הכלי RNS מורכב ממספר כלים שצריך להתקין: SUMO, BonnMotion, OSRM, ns-3.

עד עתה כדי להריץ סימולציה היה צריך להריץ כל כלי בנפרד. המשתמש נאלץ לכתוב סקריפט נפרד לכל אחד מהכלים, ולעקוב באופן שוטף על מצב ההרצה כדי לדעת מתי הוא יכול להמשיך ולהתקדם לחלק הבא בסימולציה. הכלי הסופי, בעצם חוסך עבודה וזמן על ידי כך שהוא מריץ באופן אוטומטי את החלקים הרלוונטיים מהכלים האלו.

לצורך כך ישנו קובץ קונפיגורציה (config.cfg) שבו בלבד יש צורך לשנות פרמטרים. להוראות ההתקנה יוצמד הסבר מלא על כל אחד מהפרמטרים.

לדוגמה:

- פרמטר B מקבל (קובע) את הגבולות של המפה שבה נרצה להשתמש בסימולציה.
- פרמטר NS_ATTACKTYPE מקבל (קובע) אילו מתקפות נרצה להריץ בסימולציה.
- פרמטר NS_DEFENCE מקבל (קובע) אילו הגנות נרצה להריץ בסימולציה.

אחרי שממלאים את קובץ הקונפיגורציה עם הנתונים הרצויים כל מה שנשאר לעשות זה להריץ את הקובץ magic.sh.

6. הסיבות לצורת המימוש הנוכחית

טבלה 1: השוואה בין SUMO ל-BonnMotion

	BonnMotion	SUMO
יתרונות	הרשת נשארת יציבה הודות לתנועה קבועה ואחידה	כלי המאפשר לבצע מגוון רחב של שינויים בפרמטרים של הסימולציה (לדוגמה הצד בו נוסעים בכביש) הכלי מאפשר גם להוציא טופולוגיית בניינים מתוך מפה נתונה.
חסרונות	הכלי מאפשר רק לייצר תנועות ולא מאפשר להוציא את טופולוגית הבניינים	כלי רכב מסוימים נעלמים אל מחוץ למפה במהלך הסימולציה בעקבות יציאה מגבולות המפה

אחת האפשרויות למימוש שונה היא למשל לקחת את תנועות הרכבים מ-SUMO במקום מ-BonnMotion. אולם הבעיה בתנועות שמיוצרות על ידי SUMO היא שרכבים יכולים לצאת מגבולות המפה ורכבים חדשים אינם נכנסים לסימולציה. לכן, עלול להיווצר מצב בו רשת התקשורת תתנתק מכיוון שמספר הרכבים נמוך מידי. לכן, העדפנו להשתמש בתנועות המיוצרות על ידי BonnMotion.

השתמשנו דווקא ב-ns-3 בפרויקט מכיוון שהוא מאפשר גמישות גדולה, כי כך אפשר לשלוט בקוד וכך אפשר לשנות בקלות פרמטרים וצורת מימוש שונות. למשל, ns-3 מאפשר כתיבת קוד אחד אחיד לסימולציה עם התקפה, ללא התקפה, ועם הגנה וללא הגנה.

7. קשיים

אחד הקשיים בפרויקט הזה היה להצליח למצוא את השילוב האופטימלי של כלים כך שתיווצר סימולציה המתארת באופן כמה שיותר מדויק את המציאות כפי שהיא. למשל, מציאת הכלי הנכון לייצור תנועות כלי הרכב במפת הדרכים ותיאום בין כלים שונים כך שהם יוכלו לעבוד אחד עם השני למרות שהם לא נועדו לכך.

קושי נוסף היה לבצע הרצה של כמות גדולה מאוד של סימולציות בזמן סביר. הצלחנו להקל על קושי זה על ידי אופטימיזציה של הקוד ועל ידי הרצת סימולציה קטנה יותר בכל פעם, לפני סימולציה גדולה על מנת לוודא שהסימולציה לא נכשלת ובכך לחסוך זמן מיותר של הרצה.

עוד לפני תחילת העיסוק עם חיבור הכלים, היה קושי ללמוד כיצד להתקין ולהשתמש במגוון הכלים השונים. מכיוון שכל כלי עובד בצורה שונה, מצפה לקלט שונה ומייצר פלט שונה.

8. היתרונות במימוש הנוכחי

- הכלי מדמה את המציאות בצורה יותר מדויקת מכלים קיימים. בגלל שימוש במפות של מקומות אמיתיים והתחשבות בהשפעה של הבניינים על התקשורת בין הצמתים.
- הכלי מאפשר למשתמש להריץ סימולציות עם פרמטרים שונים בצורה מרוכזת (כל הפרמטרים של כל חלקי הפרויקט מאוחסנים בקובץ אחד) ובכך לראות את ההשפעה של אותם הפרמטרים על הסימולציה
- הכלי יכול לשמש למשך זמן רב לצורך בדיקות ומאמרים עתידים בגלל הפשטות של השימוש בכלי ובגלל האפשרות לשנות את הכלי בהתאם לצורך האישי של החוקר ולמחקר שהוא מבצע.

9. הדגמת יעילות הכלי באמצעות סימולציות

9.1 מבוא

ההרצות שביצענו בעזרת כלי זה בוצעו בשלוש קטגוריות שונות לצורך השוואה ובדיקת תקינות של הקוד. כאשר ההרצות בוצעו במשך 300 שניות; באזור בגודל 750 מטר על 1000 מטר, מלבד ההרצות שבמפות בהן השטח הוא 600 מטר על 1500 מטר.

בכל הקטגוריות הנ"ל הרצנו את הסימולציות עם פרוטוקול OLSR כאשר בכולן נשלחו על ידי צומת מסוים לצומת הנתקף הודעות UDP בכל 7 שניות עד למקסימום של 18 הודעות. כל ההרצות בוצעו תוך שימוש בהתקפת Isolation והגנת DCFM כנגדה. התוקף בהתקפת ה-Isolation נצמד לצומת המקבל במרחק של 8 מטרים מימינו בכל הסימולציות. בנוסף, על מנת לוודא שהתוצאות עקביות לאורך זמן הרצנו כל סימולציה עם 500 תנועות ומיקומים התחלתיים שונים.

בחלק מן הקטגוריות, בוצעה הרצה נוספת עם עוצמת אות חלשה של אנטנות הקלט על מנת להדגים את העובדה שהאות נחלש על ידי הבניינים ובהן יהיה קשה יותר לתוקף לתקשר עם צמתים אחרים.

התבצע וידוא שהקישוריות ברשת מתקיימת על ידי זריקת סימולציות בהן המקבל לא מכיר את השולח. בנוסף, במקרים בהם הצומת השולח נצא במרק של עד שני קישורים מהשולח ההתקפה לא תוכל להצליח ולכן זרקנו גם הרצות אלו.

הקטגוריות של ההרצות הן:

1. הרצות שבוצעו עם צמתים שעומדים במקום (מיקום קבוע) לכל אורך הסימולציה. פעם אחת בעזרת מודל בשם Static של BonnMotion המציב את הצמתים בנקודות אקראיות ופעם נוספת בעזרת סקריפט מיוחד שהשתמש במודל RandomStreet של BonnMotion אך קיבע את כל הצמתים למקומם ההתחלתי לכל אורך הסימולציה (מיקומים קבועים בתוך כבישים של מפת דרכים נתונה).
2. הרצות שבוצעו עם צמתים שזזים בקווים ישרים. פעם אחת בעזרת במודל RandomDirection של BonnMotion ופעם נוספת בעזרת מודל RandomDirection2D של ns-3.
3. הרצות שבוצעו בתוך מפת דרכים על גבי כבישים של העיר ברלין. כאשר, כלי הרכב נסעו במהירות הנעה בין 5.4 קמ"ש לבין 7.2 קמ"ש, וזמן העצירה המקסימלי של כל מכונית מוגבל ל-30 שניות. זה בוצע בעזרת מודל RandomStreet של BonnMotion.

   הערה: כדאי לשים לב שכאשר בוחרים את הגודל של הרצה מבוססת מפות הגודל המעשי של נקבע על פי הכבישים המוכלים באותו אזור במפה.

תוך כדי ביצוע הרצות אלו נתגלתה בעיה במנגנון שליחת ההודעות, הבעיה הייתה שכאשר אין קישוריות בין השולח למקבל (מה שיקרה בעקבות ההתקפה) הודעה שאמורה להישלח כעבור 7 שניות לא נשלחת בכלל (מכיוון שאין קישוריות) ולכן היא לא נספרת בתור הודעה מתוך הכמות המקסימלית של 18 ולכן בפועל נוצר מצב שבו לוגית נשלחו יותר מ-18 הודעות (18 הודעות כל ההודעה שנכשלה ועוד אחת נוספת).

בפועל מכיוון שלאחר שכל ההודעות נשלחו אין צורך להמשיך את הסימולציה, ומכיוון שהייתה בעיה במנגנון שליחת ההודעות במהלך הרצת סימולציות אלו עצרנו את הריצה זמן קצר לאחר זמן סיום שליחת ההודעות שאמור היה להיות, 188 שניות (ההודעות נשלחות לאחר 60 שניות וישנן 18 בהפרשים של 7 שניות, סך הכל 186 שניות).

דבר נוסף שבדקנו הוא האם החיבור בין BonnMotion ל-ns-3 עובד כמו שצריך. כלומר, האם ns-3 קורא ומבצע באופן נכון את התנועות שמגיעות מ-BonnMotion. לשם כך יצרנו קובץ אנימציה של התנועות בסימולציה של ns-3 והשוונו בין האנימציה הזו לבין אנימציה נוספת שנעשתה באמצעות כלי מיוחד שכתבנו. המסקנה אליה הגענו היא ש ns-3 מפרש ומבצע נכון את התנועות שנשלחות אליו באמצעות BonnMotion.

על מנת להבין את הטבלאות שמוצגות להלן, יש לקרוא את ההסברים המובאים בנספח ב'.

9.2 הרצות עם צמתים במיקום קבוע

	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	18	0	18	211
Blackhole_n30_AttackType-0_Defence-1_log.txt	18	0	18	211
Blackhole_n30_AttackType-1_Defence-0_log.txt	0	0	0	201
Blackhole_n30_AttackType-1_Defence-1_log.txt	17.1990049751	3.6117826809	18	201

איור 5: מיקום קבוע ללא בניינים

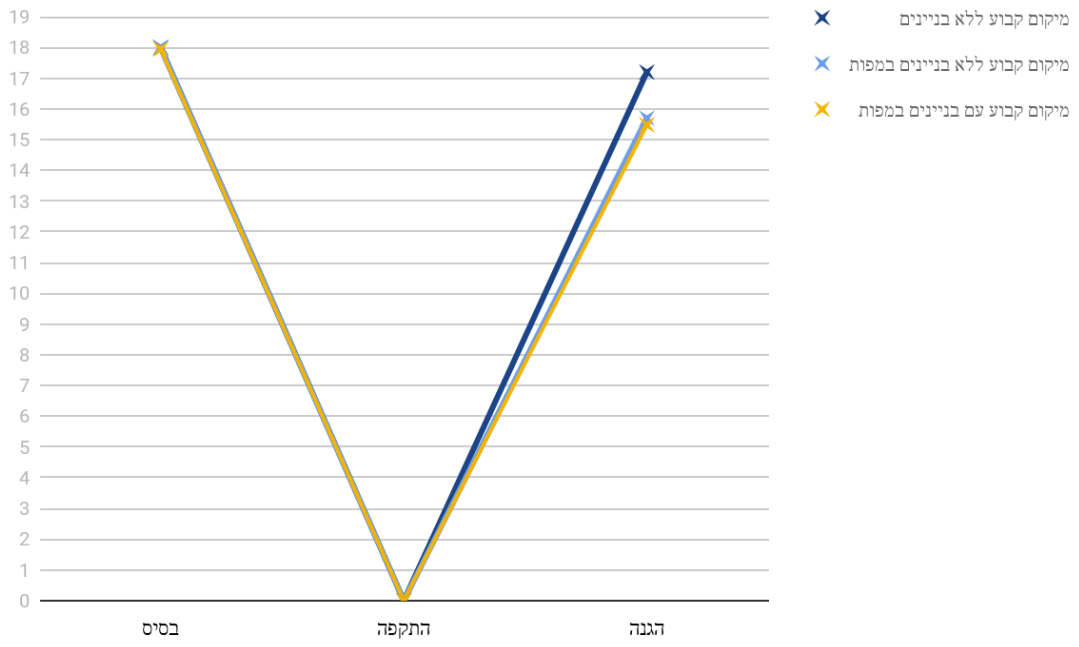
	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	18	0	18	122
Blackhole_n30_AttackType-0_Defence-1_log.txt	18	0	18	122
Blackhole_n30_AttackType-1_Defence-0_log.txt	0	0	0	123
Blackhole_n30_AttackType-1_Defence-1_log.txt	15.6991869919	5.9304025273	18	123

איור 6: מיקום קבוע ללא בניינים במפות

	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	17.9491525424	0.5499993471	18	118
Blackhole_n30_AttackType-0_Defence-1_log.txt	17.7966101695	1.2924770728	18	118
Blackhole_n30_AttackType-1_Defence-0_log.txt	0	0	0	120
Blackhole_n30_AttackType-1_Defence-1_log.txt	15.4833333333	5.9735295727	18	120

איור 7: מיקום קבוע עם בניינים במפות

הגרף הבא מראה את כמות ההודעות שהגיעו אל הצומת הקורבן (מתוך מקסימום של 18 הודעות)



איור 8: כמות ההודעות שהגיעו אל הקורבן - מיקום קבוע

הרצות אלו נועדו להראות שהכלי עובד כמו שצריך וכדי להוות מקרה בסיס לשאר ההרצות שביצענו באמצעות הכלי. דבר נוסף שניתן לראות מגרף זה הוא שהבניינים אכן משפיעים על הסימולציה בכך שהם מחלישים את התקשורת וגורמים לפחות הודעות שנשלחו להגיע ליעדן גם במקרה הבסיס.

9.3 הרצות עם צמתים שנעים בקווים ישרים

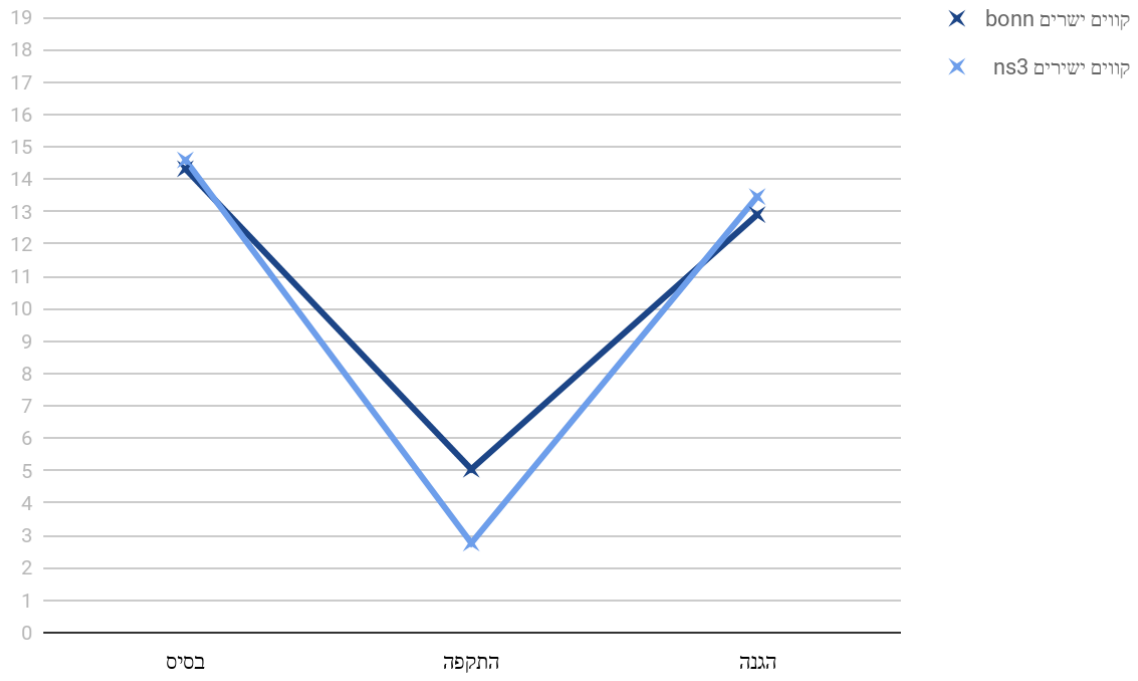
	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	14.3240740741	3.9424769045	16	216
Blackhole_n30_AttackType-0_Defence-1_log.txt	14.4697674417	3.7784908322	16	215
Blackhole_n30_AttackType-1_Defence-0_log.txt	5.0462962963	3.1472629474	5	216
Blackhole_n30_AttackType-1_Defence-1_log.txt	12.9082568807	4.8102361887	14.5	218

איור 9: קווים ישרים bonn

	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	14.59541985	5.276013163	17	262
Blackhole_n30_AttackType-0_Defence-1_log.txt	14.72900763	5.226094079	17	262
Blackhole_n30_AttackType-1_Defence-0_log.txt	2.767790262	3.133486527	2	267
Blackhole_n30_AttackType-1_Defence-1_log.txt	13.46441948	5.795003655	16	267

איור 10: קווים ישירים ns-3

הגרף הבא מראה את כמות ההודעות שהגיעו אל הצומת הקורבן (מתוך מקסימום של 18 הודעות):



איור 11: כמות ההודעות שהגיעו אל הקורבן - תנועה בקווים ישרים

הרצות אלו נועדו להראות שהכלי מסוגל לבצע סימולציות עם תנועה של צמתים ברשת ולהדגים את ההשפעה של התנועה על ההודעות שנשלחות ברשת.

9.4 הרצות עם צמתים שנעים על גבי מפה

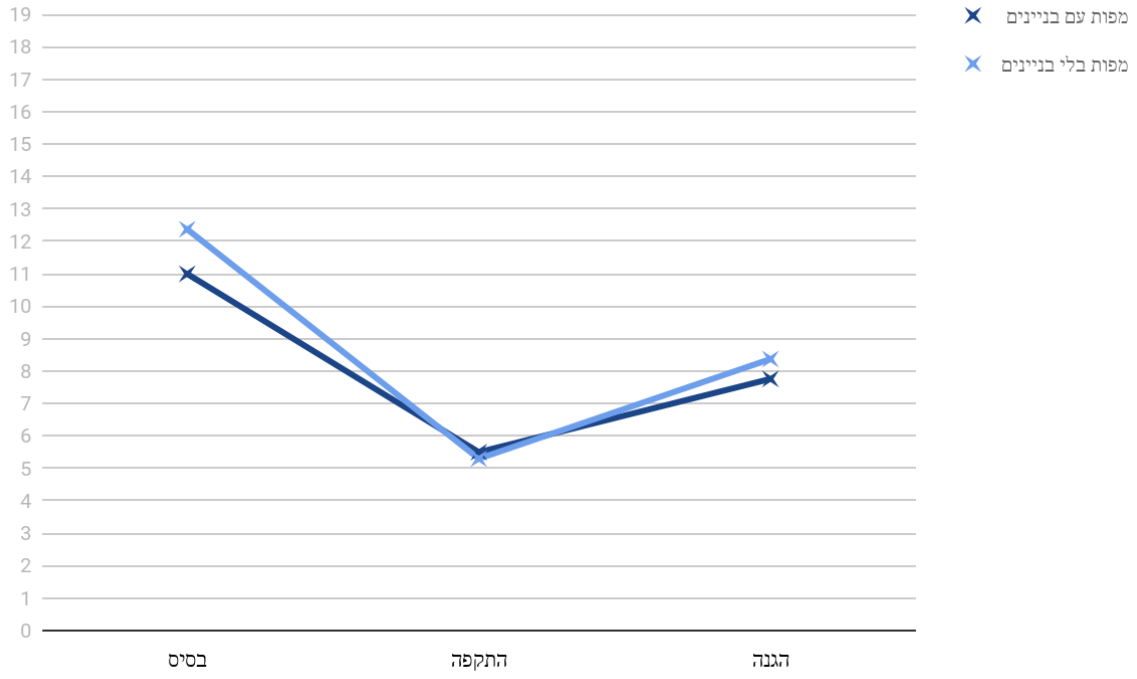
	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	11	3.3466401061	13	5
Blackhole_n30_AttackType-0_Defence-1_log.txt	11.8095238095	2.1847751899	12	42
Blackhole_n30_AttackType-1_Defence-0_log.txt	5.5	1.7078251277	6	6
Blackhole_n30_AttackType-1_Defence-1_log.txt	7.76	2.5578115646	8	50

איור 12: מפות עם בניינים

	mean	std	median	length
Blackhole_n30_AttackType-0_Defence-0_log.txt	12.375	2.7810744327	13	8
Blackhole_n30_AttackType-0_Defence-1_log.txt	13.3448275862	1.6524672644	14	87
Blackhole_n30_AttackType-1_Defence-0_log.txt	5.3	2.1	4.5	10
Blackhole_n30_AttackType-1_Defence-1_log.txt	8.3723404255	2.6133212037	9	94

איור 13: מפות בלי בניינים

הגרף הבא מראה את כמות ההודעות שהגיעו אל הצומת הקורבן (מתוך מקסימום של 18 הודעות):



איור 14: כמות ההודעות שהגיעו אל הקורבן - תנועה בתוך מפות

הרצות אלו נועדו להדגים כיצד הכלי עובד בפועל. כלומר, ביצוע של סימולציות המבוססות על תנועת רכבים במפת דרכים תוך התחשבות בטופולוגית הבניינים. כאשר ההרצה עם הבניינים נועדה להראות שהבניינים משפיעים על תוצאות הסימולציה.

10. ביבליוגרפיה

[1] S. Corson and J. Macker, “RFC 2501 Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations”, 1999.

[2] Qualnet Simulator, “Scalable Network Technologies,” https://www.scalable-networks.com/qualnet-network-simulation

[3] Benjamin A. Chambers, “The Grid roofnet: a rooftop ad hoc wireless network”, *Master’s thesis*, Massachusetts Institute of Technology, May 2002.

[4] Network Simulator, “OPNET Technologies,” https://www.riverbed.com/gb/products/steelcentral/opnet.html

[5] Perkins, Charles E.; Bhagwat, Pravin (1994). "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers"

[6] Xiang Zeng, Rajive Bagrodia, Mario Gerla, GloMoSim: a library for parallel simulation of large-scale wireless networks, Proceedings of the twelfth workshop on Parallel and distributed simulation, p.154-161, May 26-29, 1998, Banff, Alberta, Canada

[7] T. Clausen and P. Jacquet, “RFC 3626 Optimized Link State Routing Protocol (OLSR)”, 2003.

[8] Schweitzer, Nadav & Stulman, Ariel & Shabtai, Asaf & David Margalit, Roy. (2015). Mitigating Denial of Service Attacks in OLSR Protocol Using Fictitious Nodes. IEEE Transactions on Mobile Computing. 15. 1-1. 10.1109/TMC.2015.2409877.

[9] Luxen, Dennis & Vetter, Christian. (2011). Real-time routing with OpenStreetMap data. GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems. 513-516. 10.1145/2093973.2094062.

[10] https://wiki.osmfoundation.org/wiki/Main_Page

[11] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn: "BonnMotion - a Mobility Scenario Generation and Analysis Tool," in Proc. of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools '10), Torremolinos, Malaga, Spain, 2010.

[12] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 5 (3&4):128-138, December 2012.

[13] Scott E. Carpenter, Mihail L. Sichitiu, An obstacle model implementation for evaluating radio shadowing with ns-3, Proceedings of the 2015 Workshop on ns-3, p.17-24, May 13-14, 2015, Barcelona, Spain

[14] Riley G.F., Henderson T.R. (2010) The ns-3 Network Simulator. In: Wehrle K., Güneş M., Gross J. (eds) Modeling and Tools for Network Simulation. Springer, Berlin, Heidelberg

[15] https://github.com/nmontag/Realistic-Network-Simulator

[16] https://github.com/Project-OSRM/osrm-backend/tree/a62c10321c0a269e218ab4164c4ccd132048f27148f271/third_party/variant

[17] https://github.com/nmontag/Realistic-Network-Simulator/blob/master/files/magic.sh

[18] https://github.com/nmontag/Realistic-Network-Simulator/blob/master/files/config.cfg

[19] https://github.com/nmontag/Realistic-Network-Simulator/blob/master/files/config.shlib

11. נספח א' - הוראות התקנה

1.11 דרישות קדם

- Ubuntu 16.04.4 LTS (Xenial Xerus) 64-bit
- JDK
- Non-root user with sudo privileges

1. Install the software-properties-common package:
   `sudo apt install software-properties-common`
2. Add missing repositories:
   ```
   sudo add-apt-repository universe
   sudo add-apt-repository ppa:sumo/stable
   ```
3. Update apt:
   `sudo apt update`
4. Install requirements:
   ```
   sudo apt install default-jdk build-essential git cmake pkg-config
   libbz2-dev libstxxl-dev libstxxl1v5 libxml2-dev libzip-dev libboost-
   all-dev lua5.2 liblua5.2-dev libtbb-dev libluabind-dev
   libluabind0.9.1v5 gcc-4.7 g++-4.7 g++-4.7-multilib python python-dev
   python-setuptools mercurial bzr libc6-dev libc6-dev-i386 libxml2 sqlite
   sqlite3 libsqlite3-dev tcpdump flex bison libfl-dev uncrustify vtun lxc
   libboost-signals-dev libboost-filesystem-dev openmpi-bin openmpi-common
   openmpi-doc libopenmpi-dev gsl-bin libgsl-dev libgsl2 libcgal-dev qt5-
   default sumo sumo-tools sumo-doc osmctools python3 python3-pip
   ```
5. Install python packages:
   `pip3 install pyproj numpy`

2.11 התקנת חלקי הפרויקט

BonnMotion 11.2.1

1. Go to home directory:
   `cd ~`
2. Download BonnMotion:
   `wget http://sys.cs.uos.de/bonnmotion/src/bonnmotion-3.0.0.zip`
3. Extract:
   `unzip bonnmotion-3.0.0.zip`
4. [Optional] Remove the zip:
   `rm bonnmotion-3.0.0.zip`
5. Change directory to extracted folder:
   `cd bonnmotion-3.0.0/`
6. Run the installation script:
   `./install` (it finds your Java installation path automatically).

7. After successfully installing it should print something like this:
```
BonnMotion 3.0.0

OS: YOUR_JAVA_VERSION
Java: YOUR_LINUX_DISTRIBUTION


Help:
  -h                            Print this help

Scenario generation:
  -f <scenario name> [-I <parameter file>] <model name> [model options]
  -hm                           Print available models
  -hm <module name>             Print help to specific model

Application:
  <application name> [Application-Options]
  -ha                           Print available applications
  -ha <application name>        Print help to specific application
```

8. Change GC options:
```
sed -i -e 's/-Xmx512m -Xss10m/-XX:-UseGCOverheadLimit -
XX:+UseConcMarkSweepGC -Xmx10G -Xss1G/g' ./bin/bm
```

OSRM 11.2.2

1. Go to home directory using:
```
cd ~
```
2. Download OSRM v4.8.1:
```
wget https://github.com/Project-OSRM/osrm-backend/archive/v4.8.1.tar.gz
```
3. Extract:
```
tar -xzf v4.8.1.tar.gz
```
4. [Optional] Remove the tar:
```
rm v4.8.1.tar.gz
```
5. Change directory to the following folder:
```
cd osrm-backend-4.8.1/third_party
```

Now we need to replace some files from [16].

1. Download the tar from GitHub using:
```
wget https://github.com/nmontag/Realistic-Network-
Simulator/raw/master/files/variant.zip
```
2. Extract:
```
unzip -o variant.zip
```
3. [Optional] Remove the zip:
```
rm variant.zip
```
4. Copy source code changes from BonnMotion:
```
cp -a ~/bonnmotion-3.0.0/doc/OSRM/osrm-backend-4.8.1/descriptors/*
../descriptors/
```

5. Make a build folder and build OSRM: [Note: This needs to be run with gcc 5.4.0]
   ```
   mkdir ../build; cd ../build; cmake ..; make
   ```
6. Copy files from BonnMotion:
   ```
   cp -a ~/bonnmotion-3.0.0/doc/OSRM/osrm-backend-4.8.1/build/* .
   ```
7. Edit start_routed.sh:
   ```
   sed -i '$ d' start_routed.sh; printf "trap 'kill -TERM \$PID'
   TERM\n./osrm-routed \$osrmfile -i 0.0.0.0 -p \$port -t
   \$threads\nPID=\$!\nwait \$PID" >> start_routed.sh
   ```

SUMO 11.2.3

1. Set SUMO_HOME variable:
   ```
   echo 'export SUMO_HOME="/usr/share/sumo"' >> ~/.bashrc
   source ~/.bashrc
   ```

ns-3 11.2.4

1. Return to home directory using:
   ```
   cd ~
   ```
2. Download ns-3 tarball:
   ```
   wget http://www.nsnam.org/release/ns-allinone-3.24.1.tar.bz2
   ```
3. Extract:
   ```
   tar xjf ns-allinone-3.24.1.tar.bz2
   ```
4. [Optional] Delete the tar:
   ```
   rm ns-allinone-3.24.1.tar.bz2
   ```
5. Move to ns-3 modules directory:
   ```
   cd ns-allinone-3.24.1/ns-3.24.1/src
   ```
6. Download iolsr module:
   ```
   wget https://github.com/nmontag/Realistic-Network-
   Simulator/raw/master/files/iolsr.tar.gz
   ```
7. Extract:
   ```
   tar xzf iolsr.tar.gz
   ```
8. [Optional] Delete the tar:
   ```
   rm iolsr.tar.gz
   ```
9. Download obstacle module:
   ```
   wget https://github.com/nmontag/Realistic-Network-
   Simulator/raw/master/files/obstacle.tar.gz
   ```
10. Extract:
    ```
    tar zxf obstacle.tar.gz
    ```
11. [Optional] Delete the tar:
    ```
    rm obstacle.tar.gz
    ```
12. Move to ns-3 home:
    ```
    cd ../
    ```
13. Set gcc version:
    ```
    sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.7 40
    --slave /usr/bin/g++ g++ /usr/bin/g++-4.7
    ```
14. Build:
    ```
    ./waf configure --build-profile=optimized --enable-examples --enable-
    tests
    ```

NetAnim 11.2.4.1

כלי זה משמש לצורך הצגת אנימציה של הרצה שבוצעה על ידי ns-3. כלי זה אינו חלק מהפרויקט אבל הוא יכול להיות לעזור למשתמש להבין מה קורה בכל הרצה. לכן, הסבר על התקנת כלי זה מובן להלן:

1. Enter NetAnim directory:
   cd ~/ns-allinone-3.24.1/netanim-3.106
2. Build NetAnim:
   `qmake NetAnim.pro; make`

3.11 דוגמת שימוש בכלים

מובאת כאן דוגמה להרצה של כל חלקי הפרויקט באופן עצמאי, למרות ש-RNS כבר מבצע את כל ההרצות האלו באופן אוטומטי. זה מובא כאן לצורך הבנה של הכלים עצמם.

1. Go back to BonnMotion directory:
   `cd ~/bonnmotion-3.0.0`
2. Create a new folder for map files:
   `mkdir maps; cd maps`
3. Download a pbf file:
   `wget http://download.geofabrik.de/europe/germany/berlin-latest.osm.pbf`
4. Download types file:
   `wget https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/buildings.typ.xml`
5. Convert pbf to osm:
   `osmconvert ../maps/berlin-latest.osm.pbf -o=berlin-latest.osm`
6. Go to OSRM build directory:
   `cd ~/osrm-backend-4.8.1/build/`
7. Prepare the map:
   `./prepare_pbf.sh ../../bonnmotion-3.0.0/maps/berlin-latest`
8. Run an OSRM server instance (in the background):
   `sudo -b ./start_routed.sh ~/bonnmotion-3.0.0/maps/berlin-latest.osrm 5000 8`
9. Create a new folder to run out of:
   `mkdir ~/bonnmotion-3.0.0/out; cd ~/bonnmotion-3.0.0/out`
10. Run BonnMotion:
    `../bin/bm -f s1 RandomStreet -n 50 -B 13.387442 52.510168 13.393 52.515104 -u http://127.0.0.1:5000 -s 5 14 -C 1 -o ../maps/berlin-latest.osm.pbf`
11. Generate network:
    `netconvert --remove-edges.isolated --keep-edges.in-geo-boundary 13.387442,52.510168,13.393000,52.515104 --osm-files ../maps/berlin-latest.osm -o ../maps/berlin-latest.net.xml`
12. Generate buildings:
    `polyconvert --prune.in-net --osm.keep-full-type --net-file ../maps/berlin-latest.net.xml --osm-files ../maps/berlin-latest.osm --type-file ../maps/buildings.typ.xml -o ../maps/berlin-latest.buildings.xml`
13. Download the projection connection script:
    `wget https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/convert.py`

14. Run the script:
```
python3 convert.py s1.movements.geo.gz ../maps/berlin-latest.buildings.xml
```
15. Export to ns file:
```
~/bonnmotion-3.0.0/bin/bm NSFile -f s1 -b
```
16. Download ns script:
```
wget https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/stable_network_mod_2.cc -P ~/ns-allinone-3.24.1/ns-3.24.1/scratch/
```
17. Move to build directory:
```
cd ~/ns-allinone-3.24.1/ns-3.24.1
```
18. Run:
```
./waf build
```

11.4 דרישות לצורך הרצת הכלי

על מנת להריץ את RNS יש צורך להוריד קובץ בשם magic.sh [17] ולקנפג באמצעות קובץ בשם config.cfg [18] את כל הפרמטרים הרלוונטיים להרצת הפרויקט. בתוכם, כתובת אינטרנט לסקריפטים נוספים, פרמטרים של סימולציית התקשורת וכו'.

יש צורך להוריד קובץ נוסף בשם config.shlib [19] שמטרתו לאפשר לקובץ magic.sh לקרוא את הפרמטרים הכתובים בקובץ config.cfg.

11.4.1 פירוט הפרמטרים

11.4.1.1 פרמטרים להגדרה ראשונית

פרמטרים אלו נועדו להגדרה ראשונית של הכלי ולא אמורים לשנות אותם באופן תדיר.

map_files_location - מקבל את הנתיב בו רוצים לשמור את קבצי הבסיס של המפה (pbf ותנועות וכדו')

bm_run_location - מקבל את הנתיב בו יתבצעו הרצות של bonnmotion ובו נקבל את כל קבצי התנועה לצורך הרצות של ns

pbf_continent – מקבל את היבשת של המפה שבה רוצים להשתמש

pbf_country – מקבל את המדינה של המפה שבה רוצים להשתמש

pbf_file_name – מקבל את השם של הקובץ ספציפית של המפה שבה רוצים להשתמש

osrm_build_folder - מקבל את הנתיב של תיקיית ה-build של osrm

buildings_typ_link – קישור להורדת קובץ תיאור מבנים עבור bonnmotion

convert_file_link – קישור להורדת סקריפט שממיר את התנועות, כך שיתאימו לכלים המוכלים בפרויקט

convert_file_name – השם של הסקריפט מהפרמטר convert_file_link

bm_script_link – קישור לסקריפט שמריץ בעצם את ה-bonnmotion עם האופציות הרצויות

bm_script_name – השם של הסקריפט מהפרמטר bm_script_link

ns_code_link – קישור לקוד שמריץ את הפונקציונליות של ns

ns_code_name – השם של קובץ הקוד מהפרמטר ns_code_link

ns_script_link – קישור לסקריפט שמריץ את הקוד ns לפי האופציות הרצויות

ns_script_name – השם של הקובץ קוד מהפרמטר ns_script_link

ns_location מקבל את הנתיב של התיקייה בה מריצים את הסימולציה עצמה ובה מקבלים את התוצאות הסופיות

port – הפורט שעליו שרת ה-osrm מאזין

num_cores – מספר הליבות שמוקצות עבור הרצת שרת ה-osrm

11.4.1.2 פרמטרים להרצה ספציפית

הפרמטרים האלו מגדירים דברים שקשורים להרצות עצמן ואותם נשנה באופן תדיר בהתאם להרצה הרצויה.

B – מקבל את הגבולות של המפה שבה נרצה להשתמש. הגבולות יהיו בקואורדינטות בצורה עשרונית. הסדר שלהן יהיה: נקודה שמאלית תחתונה ואז נקודה ימנית עליונה, צריך שהסדר יהיה longitude ואז latitude

BM_TOTALRUNS – מספר ההרצות הכולל שנרצה לבצע ב-bonnmotion

BM_CORES – מספר הליבות המוקצות עבור ההרצות של bonnmotion

BM_FIRSTI – המספר הסידורי שממנו מתחילים את ההרצות bonnmotion

BM_NODES – מערך שמכיל רשימה של איברים, כאשר כל אחד מהם מגדיר כמות צמתים עליהם בעבור הרצה מסוימת [לדוגמה אם נרצה לבצע הרצה עם 7 צמתים ואחת נוספת עם 42 צמתים נכתוב (7 42)]

ns_run_location - שם שנועד כדי להבדיל בין תיקיות ההרצה שונות

NS_TOTALRUNS – מספר ההרצות הכולל שנרצה לבצע ב-ns (בד"כ יהיה זהה לפרמטר BM_TOTALRUNS)

NS_CORES – מספר הליבות שמקצות עבור ההרצות של ns

NS_FIRSTI – מספר הסידורי שממנו מתחילים את ההרצות ns

NS_NODES – מערך שמכיל רשימה של איברים, כאשר כל אחד מהם מגדיר כמות צמתים בעבור הרצה מסוימת [לדוגמה אם נרצה לבצע הרצה עם 7 צמתים ואחת נוספת עם 42 צמתים נכתוב (7 42)] (בד"כ יהיה זהה לפרמטר BM_NODES)

NS_ATTACKTYPE – יקבל מערך עם הערכים 0-3 (כולל) בלבד, אך לא חייב את כולם. נבחר ערכים לפי האופציות שנרצה להריץ. 0 יגדיר הרצות ללא התקפה, 1 attack isolation 3 silent blackhole 2 blackhole

NS_ATTACKLOC - יקבל מערך עם הערכים 1-5 (כולל) בלבד, אך לא חייב את כולם. נבחר ערכים לפי האופציות שנרצה להריץ. 1 תוקף רנדומלי על המפה, 2 תוקף ליד הנתקף, 3 תוקף בזווית, 4 תוקף בין שניים שמנסים לתקשר, 5 תוקף שעוקב אחרי נתקף

NS_DEFENCE – יקבל מערך עם הערכים 0-2 (כולל) בלבד, אך לא חייב את כולם. נבחר ערכים לפי האופציות שנרצה להריץ. 0 יגדיר הרצה ללא התקפה, 1 יגדיר הרצה עם DCFM ו-2 הרצה עם DCFM משופר

במקרה ורוצים להוסיף עוד סוגי התקפה או הגנה צריך לממש אותם בקובץ ns-allinone-3.24.1/ns-3.24.1/src/iolsr/model/iolsr-routing-protocol.cc אחרי שמוסיפים את הפונקציונליות הרצויה צריך לדאוג להוסיף לקובץ הרצה קריאות אליו ולערוך את סקריפט ההרצה של ה-ns כדי שיוכל להפעיל את הקוד החדש בצורה מתאימה. אחרי השינויים האלה יהיה אפשר להרחיב את טווח המספרים ב-NS_ATTACKTYPE/NS_ATTACKLOC/NS_DEFENCE בקונפיגורציה בהתאם וככה להריץ בפשטות התקפות/הגנות נוספים.

12. נספח ב' - דוגמת הרצה

מובאת כאן דוגמת הרצה של הקובץ magic.sh עם הפרמטרים הבאים בקובץ config.cfg:

```
#file locations
map_files_location=/data/home/naweiss/bonnmotion-3.0.0/maps
bm_run_location=/data/home/naweiss/bonnmotion-3.0.0/out_new_berlin_no_buildings_abort_norouting_gain_50
pbf_continent=europe
pbf_country=germany
pbf_file_name=berlin-latest
osrm_build_folder=/data/home/naweiss/osrm-backend-4.8.1/build
buildings_typ_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/buildings.typ.xml
buildings_typ_name=buildings.typ.xml
convert_file_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/convert.py
convert_file_name=convert.py
bm_script_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/bonn.sh
bm_script_name=bonn.sh
ns_location=/data/home/naweiss/ns-allinone-3.24.1/ns-3.24.1
ns_code_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/stable_network_mod_2.cc
ns_code_name=stable_network_mod_2
ns_script_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/runs_maps.sh
ns_script_name=runs_maps.sh
parser_script_link=https://github.com/nmontag/Realistic-Network-Simulator/raw/master/files/getAvg.py
parser_script_name=getAvg.py

#For osrm "Server"
port=5000
num_cores=6

B=13.417795,52.514784,13.432824,52.517876

#For bonnmotion
MIN_SPEED=1.5
MAX_SPEED=2
DURATION=300
CLIPPING=1
PAUSE=30
OFFSET=10
BM_TOTALRUNS=500
BM_CORES=10
BM_FIRSTI=0
BM_NODES=30

#for ns
ns_run_location=new_berlin_no_buildings_abort_norouting_gain_50
NS_TOTALRUNS=500
NS_CORES=10
NS_FIRSTI=0
NS_NODES=30
NS_ATTACKTYPE=0 1
NS_DEFENCE=0 1
buildings=false
```

איור 15: דוגמה לתוכן של config.csv

לאחר הרצה של הקוד הנ"ל מתקבל קובץ בשם Isolation.csv בתיקייה run שבתוך ns-3.

	mean	std	median	length
Blackhole_n30_AttackType-0_AttackLoc-6_Defence-0_log.txt	15.008	1.77987	15	500
Blackhole_n30_AttackType-0_AttackLoc-6_Defence-1_log.txt	15.194	1.722894	15	500
Blackhole_n30_AttackType-1_AttackLoc-6_Defence-0_log.txt	13	2.901724	13	500
Blackhole_n30_AttackType-1_AttackLoc-6_Defence-1_log.txt	13.89	2.635508	14	500

איור 16: דוגמה לתוכן של Isolation.csv

ניתן לראות בכל שורה בטבלה סוג הרצה אחר. כאשר AttackType שערכו 1 משמעות שהתקפה פועלת, אחרת ערכו 0, ו-Defence שערכו 1 אומר שההגנה פועלת, אחרת ערכו 0.

העמודות מייצגות את הערכים הבאים לפי הסדר משמאל לימין:

- כמות ההודעות הממוצעת שהגיעו ליעדן
- סטיית התקן של כמות ההודעות
- החציון של כמות ההודעות
- כמות ההרצות שהצליחו מתוך כמות ההרצות הכוללת

Abstract

This project addresses the need for a highly realistic and accurate network simulator to enable researchers to test the effectiveness of attacks on and protection of simulated Mobile Ad Hoc Networks (MANET) that closely resemble real networks used in a wide variety of applications.

The challenge was to create a user-friendly simulation tool that gives the researcher the flexibility to run a simulation using specific relevant data selected from a variety of sources that are not all designed to work together. The desired tool must quickly and efficiently run a large number of simulations representing different circumstances.

The tool we created [Realistic Network Simulator] connects four different tools: SUMO, BonnMotion, OSRM, ns-3. Each of these tools has advantages and limitations, and each includes different parameters. We describe these tools in detail in the body of this document and explain why we selected them. We also document how to install and configure them for optimal operation when using our tool, which enables the user to automatically run the parts of each separate tool which are relevant for the desired simulation.

We describe the process of determining the optimal combination of tools to create a simulation, taking into account that they each have different inputs and outputs. We explain how we optimized our code, testing small simulations before running multiple, large simulations to demonstrate the advantages of our tool. All simulation results are saved in a format that can be parsed to display the relevant information about the effectiveness of the defenses and attacks in the given network.

In this project our simulator uses the example of a MANET network established between randomly moving automobiles. To create the most realistic simulation possible we used maps of real places where our simulator could include the effect of local buildings on the strength of the signal sent by the vehicles. With our tool it is possible to easily change individual parameters to run simulations of various types of attacks on and defenses of the network and thus see the effect of those parameters on the simulation. We describe multiple simulations that we ran to compare and validate the code and demonstrate the effectiveness of our tool.