**אוניברסיטת תל אביב**

**הפקולטה להנדסה ע"ש איבי ואלדר פליישמן**

**חקירה ומיטוב ביצועים של פרוטוקול 802.11 ברשתות ארוכות טווח**

מוגש ע"י

**לילך סבן**

העבודה נעשתה בביה"ס להנדסת חשמל

בהנחיית מר אדם משיח, מר איתן שילה ופרופ' אנטוני וייס

# רקע

בעשורים האחרונים עולם האינטרנט הולך ומתפתח. השימוש בתשתיות פיזיות של רשתות קוויות הולך ופוחת כשמנגד השימוש ברשתות קצה אלחוטיות הולך וגובר. רשתות אלו מבוססות על תקן 802.11 הידוע בכינויו Wi-Fi. רשתות Wi-Fi נמצאות בשימוש נרחב מאוד הן בבית והן במקומות ציבוריים, וכך מסייעות לנו להישאר מחוברים בכל מקום, בקלות וללא צורך בחיבור פיזי או בציוד מיוחד. בשונה מרשתות קוויות, המצריכות פריסת קווי תקשורת באזורים נרחבים, כל שנדרש על מנת להקים רשת Wi-Fi הוא התקן שיהווה נקודת גישה (access point) המשדרת בהתאם לתקן. בנוסף כיום מכשירים המממשים את התקן הופכים להיות זולים מאוד ופשוטים להפעלה. התקן ממשיך להתעדכן ולהשתפר במשך הזמן, ובמקביל התקנים פשוטים יותר עושים בו שימוש לצורך העברת מידע. יחד עם הגידול בנגישות האלחוטית, גדלים גם נפח התעבורה העובר ברשת העולמית ונפח התעבורה העובר בתווך האלחוטי.

תופעות אלו יוצרות אתגרים חדשים לספקיות רשתות ציבוריות. כך נוצר מצב בו תחום התדר, שהוא מוגבל, עמוס מאוד בשידורים, נוצרת בעיה משמעותית שקיימת ברשתות אלחוטיות - בעיית ההתנגשויות. כאשר מספר מכשירים משדרים באותו תדר או בתדרים חופפים, הם משתפים ביניהם את התווך האלחוטי ויש ביניהם תחרות על השימוש בו. ברשת אלחוטית ללא ניהול מרכזי, עלול להיווצר מצב בו מכשיר משדר מסגרת מבלי להתייחס לכך שהתווך תפוס בגלל שידורים של מכשירים אחרים, וכך המסגרת תתנגש עם מסגרות אחרות ולא תפוענח כראוי אצל מכשיר היעד. הדבר המגדיל את כמות השידורים החוזרים בלינק, וגורם לבזבוז של משאבים. בנוסף הגדלת כמות השידורים החוזרים מגדילה את העומס בתווך, מה שמגדיל את הסיכוי להתנגשות עוד יותר.

על מנת לנסות לפתור את הבעיה, קיים בתקן 802.11 מנגנון ה-CSMA/CA. המנגנון מנסה למנוע התנגשויות באמצעות sensing של התווך האלחוטי על ידי כל מכשיר שרוצה לצאת לשידור, וכן הוספת המתנות של זמנים רנדומליים (backoff) לפני שידור. מכשיר שרוצה לצאת לשידור לא יעשה זאת אם הוא חש שהתווך תפוס. המתנת הזמן הרנדומלי לפני השידור במקרה בו התווך תפוס מקטינה עוד יותר את ההסתברות להתנגשות, שכן רק מסגרות המשודרות על ידי מכשירים שהגרילו את אותו מספר רנדומלי והתחילו להמתין באותו הזמן עלולות להתנגש (בהנחה שזמן ההתפשטות קטן ביחס לפרמטרים האחרים בתקן). בנוסף, גם אם מכשיר חש שהתווך פנוי, הוא ממתין זמן נוסף שנקרא ( IFS (Inter Frame Spacing, אשר תלוי בסוג המסגרת אותה הוא רוצה לשדר וגם בערך פרמטר של הפרוטוקול הנקרא SlotTime. מטרת ההמתנה הזו היא לאפשר למסגרות לעבור את כל המרחק של הרשת (לפי זמן ההתפשטות) כדי שכל המכשירים יוכלו לקלוט אותן בזמן. מנגנון ה-CSMA/CA כפי שהוא מוגדר בתקן יוצא מתוך נקודת הנחה שזמן ההתפשטות באוויר קטן מאוד (מסדר גודל של 1usec).

אתגר מניעת ההתנגשויות גדול אף יותר ברשתות Wi-Fi ארוכות טווח (WiLD), בהן המרחק בין המכשירים המתקשרים גדול מאוד. למשל, כאשר רוצים ליצור חיבור אלחוטי בין רשתות Wi-Fi מרוחקות, כשבכל רשת הרכיבים קרובים זה לזה, משתמשים ברכיבים המקשרים בין הרשתות. במקרה כזה, זמן ההתפשטות באוויר גדול ואינו זניח, בשל המרחק הגדול בין המכשירים. כתוצאה מכך, קיים הפרש זמנים משמעותי בין זמן תחילת שידור של מסגרת על ידי מכשיר לזמן ההגעה של המסגרת למכשירים האחרים (כולל היעד). לכן, ניהול הגישה לתווך בעזרת מנגנון ה-CSMA/CA, כפי שהוא מוגדר בתקן, לא עובד מספיק טוב. מכשיר אשר חש את התווך לפני היציאה לשידור לא בהכרח ימתין מספיק זמן כדי להספיק לגלות שהתווך תפוס (כלומר לקלוט שידור של מכשיר אחר) לפני שיתחיל לשדר בעצמו. כך תקרה התנגשות בין המסגרות.

מטרת הפרויקט היא לחקור כיצד ניתן לשפר את יעילותן של רשתות Wi-Fi ארוכות טווח (מרחק של 5 ק"מ ומעלה בין התחנות המתקשרות) באמצעות שינוי בפרמטרים של מנגנון ה-CSMA/CA בפרוטוקול 802.11, והתאמתם למרחק בין המכשירים המתקשרים. הפרמטרים העיקריים בהם אנו מתמקדים הם ה-ACK timeout וה-SlotTime. יעילות הרשת נמדדת באחוז נפח התעבורה המבוזבז בשל התנגשויות, ושידורים חוזרים שנוצרים עקב כך.

נערכו מספר מחקרים קודמים הקשורים ברשתות Wi-Fi ארוכות טווח. כדי להתמודד עם בעיית ההתנגשויות ברשתות אלו, במאמר [1] הכותבים מציעים להגדיל משמעותית את ערך ה-SlotTime ל- של הרשת (ערך התלוי באורך הלינק המקסימלי ברשת). ערך זה מבטיח שכל מכשיר ימתין מספיק זמן כך שהוא יספיק "לשים לב" למסגרות שמגיעות ממכשירים אחרים לפני שיצא לשידור. אולם, החיסרון של ערך SlotTime כה גדול הוא שיש המתנות ארוכות יחסית לפני שליחת המידע, וזה עלול להיות בזבוז זמן שבו יכולנו להעביר עוד מידע ולהגדיל את ה-throughput של האפליקציה. מאמר [3], מראה שערך ה-ACK timeout המוגדר בתקן גורם לכך שעבור מרחק גדול מ-110 ק"מ בין המכשירים שמנסים להעביר מסגרת זה לזה, תמיד המסגרת תשלח שנית. זאת מפני שזמן ההתפשטות הגדול גורם לכך שה-ACK לעולם לא יגיע ליעד לפני פקיעת ה-timeout. זה גורם לירידה חדה ביעילות הערוץ (ירידה ב-BW האפקטיבי שנמצא באמת בשימוש להעברת המידע).

בפרויקט נבחן את הקשר בין גודל ה-SlotTime וה-timeout כפונקציה של ה-propagation delay להצלחת מנגנון הCSMA/CA. על מנת לעשות זאת, נכתוב סימולטור תוכנתי בMATLAB למנגנון ה-CSMA/CA. הסימולטור ייכתב בגישה מונחית אירועים, ויאפשר לדמות לינקים ומכשירים בעלי פרמטרים הניתנים לקונפיגורציה. אנו צפויים למצוא ערכי פרמטרים אופטימליים כתלות במרחק בין המכשירים המקשרים, שיפחיתו את כמות ההתנגשויות, יקטינו את כמות השידורים החוזרים ובמקביל יפגעו כמה שפחות בביצועים בשל ההמתנות הארוכות. אנו מקווים שכך יעילות הערוץ תגדל, כלומר רוחב הפס האפקטיבי הנמצא בשימוש להעברת המידע עצמו, יגדל.

# סקירת ספרות

## רקע כללי לגבי התקן

### תקן 802.11

802.11 הוא שם כולל למשפחת תקנים לתקשורת אלחוטית ברשתות מקומיות (LANs), כינוי מוכר שלו הוא Wi-Fi. הוא נוצר על ידי ה-IEEE (בוועדה מספר 802, קבוצה 11). Wi-Fi מכיל בתוכו עשרות תקנים שמסומנים באות אנגלית אחת, ועוד עשרות תקנים שמסומנים בשתי אותיות אנגליות, כולל תקנים עתידיים ותקנים שנמצאים כעת בפיתוח. התקן כולל 6 רמות פיזיות שמשתמשות באותה רמת MAC. תחום התדרים בו משתמשים תקני ה-Wi-Fi הוא 2.4GHz, 5GHz או גם וגם. הדבר יוצר בעיית חשיפה להפרעות מצד מכשירים אחרים שפועלים באותו תחום תדרים – למשל מיקרוגל שפועל גם הוא ב-2.4GHz. אזור ה-5GHz הינו אזור פחות עמוס בשידורים, כלומר תחום תדרים זה פנוי יותר. חלק מן התקנים משתמשים בשיטת האפנון OFDM. זוהי שיטת קידוד מידע ספרתי שמחולק בצפיפות להרבה תדרים שונים, כשהחלוקה מאפשרת התגברות על הפרעות ניחות. **תקני ה-Wi-Fi העיקריים הקיימים כיום**עם השנים, התפתחו תקני ה-Wi-Fi בשלושה מישורים: אבטחה (הלכה והשתפרה), קצב העברת נתונים (הלך וגבר), שימוש בתחומי תדר מרובים.

התקנים העיקריים הקיימים, החל מהתקן הראשון:

* 802.11 legacy – התקן המקורי, שוחרר ב-1997. ה-Bit rate בו הוא 1-2Mbps והתדר הוא 2.4GHz, יש בו CSMA/CA, דבר שמקריב חלק מקיבולת הערוץ. התקן מאפשר חופש רב למשתמש, אולם חופש זה עלול לגרום לחוסר תאימות בין מערכות. התקן ישן יחסית אך עדיין נמצא בשימוש במעט מערכות מסחריות.
* 802.11b – תיקון שיצא ב-1999. ה-Bit rate בו הוא עד 11Mbps והתדר הוא 2.4GHz. יש בו CSMA/CA, ולכן בפועל התקורה היא כמעט 40% (ב-UDP קצת יותר מב-TCP). לרוב משמש בתקשורת point-to-point. יקר פחות מהאחרים.
* 802.11a – גם הוא שוחרר ב-1999. ה-Bit rate בו הוא עד 54Mbps והתדר הוא 5GHz, מה שמפחית את ההפרעות החיצוניות והרעשים מהתדרים הסמוכים (אזור "פחות עמוס"). מאידך, זה מצריך יותר Aps כי צריך line of sight. הקצב הגבוה מתאפשר הודות לשימוש ב-OFDM של 52 ערוצים, וכן מתבצעת ירידה הדרגתית בקצב כשמתקבל אות חלש או כשיש הרבה הפרעות. אחד מחסרונות התקן הוא אי תאימות לתקן b, שיצא לפניו.
* 802.11g – יצא ב-2003. ה-Bit rate בו הוא עד 54Mbps והתדר הוא 2.4GHz. התקן תואם לתקן b, גם הוא משתמש ב-OFDM של שלושה ערוצים חופפים. השוני הוא שתקן g מהיר בהרבה ויקר רק במעט (בתיאוריה, לפחות. במציאות החשיפה הגבוהה להפרעות ולרעשים בשל קצבי ההעברה המהירים גרמה להורדת הקצב לקצב דומה לשל תקן b).
* 802.11n – נקרא גם Wi-Fi4. יצא בשנת 2009. ה-Bit rate בו הוא עד 600Mbps והוא עובד ב-dual baud – בשני תדרים: 5GHz ו-2.4GHz. זהו התקן הראשון שהשתמש ב-MIMO – האצה באמצעות שימוש במספר אנטנות במקביל. טווח הקליטה שלו גדול יותר משל הקודמים לו – כ-70 מטרים בתוך מבנה. זהו התקן הנפוץ ביותר כיום.
* 802.11ac – נקרא גם Wi-Fi5. אושר בשנת 2014. ה-Bit rate בו הוא עד 3.46Gbps והוא עובד ב- 5GHz אך יש גם מכשירים התומכים ב-2.4GHz לצורך תאימות לאחור. תקן זה משתמש גם הוא ב-MIMO – האצה באמצעות שימוש במספר אנטנות במקביל. הוא מסוגל לשדר בכמה ערוצים במקביל, וכן מתאים לשידורים לטווח רחוק יותר מהתקן הבא אחריו עם throughput גבוה, למשל בתוך WLAN ביתית.
* 802.11ad – יצא ב-2002. ה-Bit rate מאוד גבוה - עד 6.7Gbps והתדר הוא 60GHz, אך המחיר הוא המרחק – רדיוס תקשורת של עד 3.3 מטרים בלבד. כלומר, תקן זה מתאים יותר לשימוש במצבים של line of sight עם throughput גבוה – למשל לסנכרון קבצים או להעברת מולטימדיה לא דחוסה בתוך חדר. הסיבה לכך היא שהגלים בתדר 60GHz חשופים להפרעות קשות משמעותית מהתווך, לעומת הגלים בתדר 5GHz.

הערה: כל הקצבים המצוינים לעיל הם הקצבים שמאפשרת הרמה הפיזית, ולכן אינם הקצבים  
בפועל - ישנן יתירויות שמוסיפות השכבות שמעל השכבה הפיזית. בנוסף התווך משותף בין כלל המשתמשים, ויש גם הפרעות ורעשים הגורמים להורדת הקצב.

### סוגי התעבורה

ניתן להעביר ברשת Wi-Fi סוגים רבים של תעבורה: החל ממידע טקסטואלי, הודעות, קבצים וכלה בסרטי וידאו, קטעי אודיו וכדומה. רוחב הפס הוא מוגבל, ולכן קשה להעביר קבצים "כבדים" דרך Wi-Fi, כמו למשל קטעי וידאו לא דחוסים. התקנים החדשים, ac ו-ad (פירוט לגביהם מופיע בהמשך) מאפשרים "Gigabit Wi-Fi", רשת מהירה מאוד, יותר מה-Gigabit Ethernet. בעזרתם ניתן להעביר multimedia streaming בגודלם ובאיכותם המקוריים. ניתן להשתמש ב-Wi-Fi גם לביצוע Cellular data offloading – כלומר להעביר חלק מהתקשורת שנועדה במקור לרשת הסלולרית, לרשת Wi-Fi נגישה. הדבר מסייע בהפחתת העומס הנחווה ברשת הסלולרית, בייחוד במקומות הומי משתמשים כמו אוניברסיטה, קניון וכדומה.  
הטווח שנקודת גישה של Wi-Fi מכסה בתוך בית ממוצע עומד על כ-100 רגל שהם 30.5 מטרים. הטווח קטן בשל ניחות הנגרם מהקירות ומהרצפה. כשיש line of sight ניתן לעבוד בטווחים גדולים יותר. בנוסף ניתן להגדיל את הטווח גם בתוך שטח בנוי, באמצעות מכשירים לא יקרים הנקראים signal boosters.  
כשמתחילים תקשורת Wi-Fi ורוצים להקים נקודת גישה, חשוב לקנפג מספר דברים עבורה:

* SSID – שם הרשת, ברירת המחדל היא שם היצרן.
* Channel – הערוץ על גביו תתבצע התקשורת, מספר שלם בין 1 ל-11. ברירת המחדל היא 6, אך אם מכשירים קרובים משדרים גם הם ב-6, רצוי לבחור ערוץ אחר.
* WEP Key – קיימת אפשרות לבצע תקשורת מוצפנת, ושדה זה יכיל את מפתח ה-WEP בן 128 (למעשה 104) ביטים לצורך ההצפנה.

### אופני הפעולה

על מנת לייצר תקשורת Wi-Fi, הדבר היחיד שצריך הוא נקודת גישה – access point, או hotspot. ניתן לייצר תקשורת המונית ברשת אלחוטית כל עוד לכל המשתמשים יש גישה לנקודת הגישה והם נמצאים במרחק קטן מספיק מן המכשירים האחרים (בתקנים השונים הגלים מספיקים כדי לכסות טווחים שונים של מרחקים). במקרה בו יש אפשרות להתחבר לרשת, רוב המחשבים החדשים עושים זאת באופן אוטומטי.  
קיימים ב-Wi-Fi 3 אופני פעולה עיקריים:

* Infrastructure - תקשורת בין מכשירים רבים ברשת, עם תשתית מאורגנת וגוף שמנהל אותה, שנקרא access point.
* Ad-Hoc - ניתן לבצע תקשורת בין מכשירים מרובים, ללא תשתית מאורגנת. הצרכנים מתקשרים בינם לבין עצמם ללא גוף שמנהל את התקשורת.
* Direct Link - ניתן לייצר תקשורת point-to-point, בין שני משתמשים ספציפיים.

### אבטחה ב-Wi-Fi

ניתן ליצור נקודת גישה מאובטחת ב-Wi-Fi. זוהי נקודה שלא לכל המשתמשים יש גישה אליה, אלא רק למורשים – כלומר משתמשים שיודעים מידע מסוים. המידע הנחוץ הוא שם הרשת – SSID, וכן מפתח הצפנה. בעבר היה זה WEP Key – מפתח WEP (Wired Equivalent Privacy) שאמור להיות קשה לפריצה. קיימות שתי גרסאות של המפתח (כמות הביטים בפועל שמכיל המפתח קטנה ב-14 מהמספר המצוין) – 128-bit-encription וכן 64-bit-encription. במציאות, נמצא כי המפתח הקצר יותר קל יחסית לפריצה וגם המפתח הארוך אינו בטוח במיוחד, בין היתר מפני ששיטת WEP משתמשת בצופן הבלוקים RC4 שאינו בטוח (מבוסס על פעולת XOR בין הטקסט הגלוי למפתח הנגזר מסוד משותף ומה-IV, ולכן שימוש חוזר באותו המפתח הוא מסוכן. התוקף יכול להסיק מתוצאת ה-XOR של שתי הודעות המוצפנות באותו מפתח, מידע אודות הטקסט הגלוי. זאת מפני שתוצאת ה-XOR של ההודעות המוצפנות זהה לתוצאת ה-XOR של ההודעות הגלויות).  
לכן תקני Wi-Fi עברו להשתמש באימות באמצעות 802.11X המבוסס על EAP. בהמשך התקנים עברו להשתמש במפתח WPA ולאחר מכן במפתח WPA2 שבטוח אף יותר.   
בהמשך גם בהצפנת WPA2 נמצאה חולשה ופותחה מתקפה שפורצת את ההצפנה ומכונה KRACK. החולשה נובעת מכך שבעת יצירת מפתח ה-WPA2, מתבצעת 4-way-handshake בין שני הצדדים המתקשרים. הם מחליפים ביניהם מידע שחלקו מאומת ובאמצעותו נגזר מפתח ההצפנה שישמש אותם במהלך התקשורת. אחד מהערכים שמהם נגזר המפתח נקרא PTK (Pairwise Transmit Key). לשמירה על "טריות" מפתח ההצפנה, חשוב שערך ה-PTK יהיה חד פעמי, ושלא יתבצע שימוש באותו PTK יותר מפעם אחת. החולשה שהתגלתה ב-WPA2 מאפשרת לתוקף לגרום לשימוש חוזר באותם ערכי PTK ובפרט לגרום למכשיר להחזיר את ערך ה-PTK לערכו ההתחלתי (מעין ביצוע restart). התוקף משיג מטרה זו באמצעות ביצוע replay attack על ההודעה השלישית בלחיצת היד ה-4 שלבית. כך מסכן התוקף את בטיחות הערוץ עד כדי כך שהוא יכול לגלות את מפתח ההצפנה ולקרוא את כל התשדורות המוצפנות המועברות בערוץ התקשורת. זהו חור אבטחה חמור ב-WPA2.   
כיום כבר קיימת שיטת הצפנה מתקדמת יותר הנקראת WPA3.

### שיטת MIMO

חלק מהתקנים משתמשים בשיטה זו – Multiple Input Multiple Output. זוהי שיטה להגדלת הקיבולת של ערוץ אלחוטי, באמצעות שימוש במספר אנטנות הן בקליטה והן בשידור. ניתן ליישם שיטה זו ברשתות שאליהן מנסים לגשת משתמשים מרובים, וכן ברשתות שאליהן ניגש משתמש אחד. במקרה של משתמשים מרובים, פרוטוקולי 802.11 נותנים שירות לצרכנים בגישת FCFS (תור הוגן). במצב כזה יכול להיווצר עומס ברשת (Congestion) ולכן MIMO היא שיטה שעוזרת – ניתן לתת בעזרתה שירות למספר משתמשים במקביל. רוחב הפס מחולק באופן שוויוני לזרימות (streams). כך זמן ההמתנה של משתמש יורד. אחד החסרונות הוא שמכשירים שנמצאים בסמיכות זה לזה עלולים "להתחרות" על אותם זרימות.

### מבנה רשת Wi-Fi

הרשת בנויה מנקודת גישה – access point (AP) והרבה clients שמקיימים תקשורת ביניהם, בדרך כלל לא ישירות אלא באמצעות ה-APהמתווכת ביניהם. בתצורת Infrastructure הרשת מנוהלת על ידי ה-AP ואילו בתצורת Independent (Ad-Hoc) אין גורם שמנהל את הרשת.  
הרשת שמקצה AP (מבחינת טווח השידור והקליטה שלה) נקראת BSS – Basic Service Set. במקרים בהם יש מספר נקודות גישה שיוצרות מספר BSS's, האיחוד של כולן נקרא ESS – Extended Service Set. חשוב שתהיה חפיפה בין תתי הרשתות, על מנת למנוע מצבים בהם במקרה של נפילת אחת ה-AP's, לא תתחלק הרשת לתתי רשתות בלתי מקושרות.  
הגורם שמחבר (מבחינה לוגית) בין BSS או ESS לרשתות אחרות נקרא DS – Distribution System. לפעמים מכונה גם Wireless Bridge.  
קיימות שתי גישות לגבי אופן ההתחברות לרשת (בתצורת Infrastructure). AP יכול להיות פסיבי או אקטיבי.   
במקרה האקטיבי, אחת לפרק זמן קבוע (בדרך כלל כל 100 מילי-שניות, כלומר 10 פעמים בשנייה) משדרת ה-AP הודעה הנקראת Beacon ב-broadcast. ההודעה מגיעה לכל התחנות הנמצאות בטווח השידור של ה-AP, ומכילה מידע עליה ואת הפרטים הנחוצים לתחנה על מנת להתחבר אליה. זוהי למעשה הדרך של ה-AP "לפרסם את עצמה" ולהזמין תחנות להתחבר אליה. תחנה שמעוניינת להתחבר לרשת צריכה להמתין עד שתקלוט הודעת Beacon ואז להשתמש בפרטים הרשומים בה לצורך התחברות לרשת.   
במקרה הפסיבי, לעומת זאת, ה-AP כביכול מסתירה את עצמה, ולא מפרסמת מידע על עצמה כמו בגישה האקטיבית. עם זאת, תחנות שרוצות להתחבר יכולות להשיג את פרטי ה-AP משיחות שלה עם תחנות אחרות שנמצאות באותו טווח קליטה ושידור, וכך להתחבר בכל זאת (יש לדעת למשל את כתובת ה-MAC של ה-AP).

### סוגי מסגרות

המסגרות הנשלחות ברשת אלחוטית מתחלקות לשלושה סוגים:  
Data frames – מסגרות המכילות את התוכן, המידע העובר ברשת מתחנה לתחנה.  
Control frames – מסגרות שפועלות בשיתוף עם מסגרות המידע, על מנת להבטיח תעבורה אמינה וכן Carrier Sensing (גישה מסודרת לערוץ תוך מניעת התנגשויות). למשל מסגרות ACK ומסגרותCTS הן מסגרות מסוג control.  
Management frames – מסגרות המיועדות לניהול הרשת והחיבורים בה, ולבקרה על מי שמתחבר לנקודות הגישה. הן יכולות לחבר ולפצל רשתות, וכן להעביר את ה-association של משתמשים מ-AP ל-AP.

### תהליכים ברשת Wi-Fi

קיימים מספר תהליכים שמתבצעים ברשת:

Scanning – פעולה שמבצעים ה-clients, קיימים שני סוגי סריקות:   
סריקה פסיבית – במקרה בו ה-AP אקטיבי, כלומר מפרסם את עצמו, הלקוחות עוברים על התדרים הרלוונטיים בחיפוש אחר הודעות Beacon. הם מוצאים הודעה שהם עונים על דרישות ה-AP המפרסמת אותה ושולחים בקשה להתחבר – Probe Request. הם נענים ב-Probe Response ומתחברים לרשת דרך ה-AP.  
סריקה אקטיבית – במקרה בו ה-AP פסיבי, כלומר מסתיר את עצמו (silent), לקוח שמעוניין להתחבר אליו צריך לבקש מפורשות להתחבר, לפי ה-SSID של הרשת ומידע נוסף במידת הצורך. במידה והלקוח מעוניין לחפש רשתות זמינות ולא להתחבר לרשת מסוימת שהוא יודע את פרטיה, הוא יכול לשלוח Probe Requests ולהמתין לתגובה מ-AP ואז לנסות להתחבר אליה.  
Authentication – אימות של זהות הלקוח מול ה-AP ולהפך. יש גם גרסה ללא אימות אמתי, רק הכרה בשני הצדדים, הנקראת open system.  
Association – תהליך שמתבצע לאחר שלב האימות, לווידוי סופי שהכל מוכן לקראת התקשורת ולהסכמה על פרמטרי התקשורת.

## בעיית הגישה לערוץ

ברשת Wi-Fi, כמו בכל רשת תקשורת מרובת משתמשים, קיימת סכנת התנגשויות של מסגרות. כאשר ישנן מספר מכשירים המשדרים באותו ערוץ WiFi או בערוצים חופפים, הם משתמשים באותו תדר או בתדרים סמוכים. המכשירים משתפים ביניהם את התווך האלחוטי. כל מכשיר המעוניין לשדר מסגרת משתמש בתווך, וכך נוצרת תחרות בין המכשירים על השימוש בתווך. לכן, בהיעדר ניהול מרכזי ברשת (קיימות רשתות WiFi שיש בהן ניהול מרכזי, אך אנו מתמקדים בפרויקט ברשת ללא ניהול מרכזי), עלולות לקרות התנגשויות בין מסגרות שמשדרים המכשירים. אם שני מכשירים (או יותר) שולחים לאוויר מסגרות כך שנוצר מצב בו בצד קולט מסוים נקלטים שני שידורים (או יותר) בו זמנית, נוצרת התנגשות, מכיוון שהמכשיר הקולט לא יכול להפריד בין שתי המסגרות. מצב נוסף בו תיתכן התנגשות, הוא שבמהלך שידור מסגרת, מתקבלת מסגרת שמיועדת למכשיר המשדר.

המשמעות של התנגשות היא שמי שאמור לקבל את המסגרת לא יוכל לקבלה כראוי, מכיוון שהוא לא יוכל "להפריד" בין המידע ששייך למסגרת שמיועדת אליו למידע ששייך למסגרת שהתנגשה אתה.   
זה מצב בעייתי, כיוון שהוא פוגע בביצועי הלינק. מסגרות שהתנגשו לא תפוענחנה כראוי ויהיה צורך לשלוח אותן שוב – מה שמעלה את כמות ה-retransmissions. שידורים חוזרים מרובים הם דבר לא טוב משתי סיבות: ראשית, הם גורמים לבזבוז משאבים, מכיוון שהם מגדילים את ה-overhead התקשורתי של הלינק. שנית, בתקשורת אלחוטית זיהוי התנגשויות אינו אפקטיבי (אין collision detection, כי לא כל המכשירים מודעים לקיומם של כל שאר המכשירים ברשת, למשל במקרה של hidden node). לכן במקרה של התנגשות, לא ניתן לעצור את השידור וכך כל המידע במסגרות שהתנגשו הולך לאיבוד. שידורים חוזרים מרובים גורמים להגדלת נפח התעבורה, דבר המוביל להגדלת כמות ההתנגשויות, וכך נוצרת תגובת משוב חיובית.  
על מנת לנסות להפחית את כמות ההתנגשויות ברשתות WiFi, משתמשים במנגנון CSMA/CA. זהו מנגנון שמטרתו לנסות למנוע התנגשויות מראש, באמצעות sensing של התווך האלחוטי וכן הוספת המתנות של זמנים רנדומליים (backoff) לפני שידור, לפני שמתרחשת התנגשות (בשונה מ-CSMA/CD בו המתנת ה-backoff מתרחשת לאחר התנגשות). הנחות היסוד של המנגנון הן:

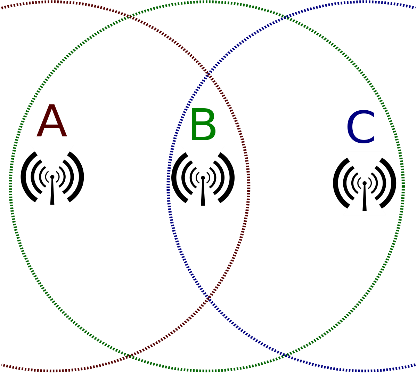
* כל מכשיר קולט את השידור של כל שאר המכשירים.
* זמן ההתפשטות באוויר (Air Propagation Delay) קטן ביחס לפרמטר השכבה הפיזית SlotTime

מנגנון ה-CSMA/CA משתמש בפרמטרי ה- IFS – Inter Frame Spacing. פרמטרים אלו קובעים את זמני ההמתנה של תחנה לפני שידור של מסגרת. הזמנים משתנים על פי סוג המסגרת, וניתנת עדיפות למסגרות שתפקידן בקרה, כמו למשל ACK . זמן ההמתנה הקטן ביותר נקרא SIFS – זהו זמן ההמתנה לפני שליחת ACK. זהו הזמן שלוקח למכשיר לחוש את סיומו של המסגרת הקודם ולהתחיל לשדר מסגרת משלו. זמן גדול במקצת נקרא PIFS – זהו זמן ההמתנה לפני שליחת מסגרות חשובות, למשל Beacon. הזמן הנורמטיבי נקרא DIFS – והוא מיועד לשליחת מסגרת המכילים data. זהו הזמן שיש להמתין לפני תחילת תהליך backoff. במקרה של התנגשות או כשהתווך נדגם כ"תפוס", יש לחכות בנוסף גם זמן backoff רנדומלי אשר נועד למנוע התנגשויות בעתיד. ה-backoff הוא כפולה שלמה של SlotTime, כלומר , כאשר num מוגרל מטווח הגדל אקספוננציאלית עם כמות ההתנגשויות (עד ערך מקסימלי מסוים).  
לפי מנגנון ה-CSMA/CA, מכשיר שרוצה לצאת לשידור עובר את השלבים הבאים:

* בודק האם התווך פנוי – Carrier Sensing.
* אם התווך פנוי – המכשיר ממתין זמן מסוים (IFS – Inter Frame Spacing) שתלוי בסוג המסגרת אותה רוצה המכשיר לשדר וגם בערך פרמטר ה-SlotTime. אם במהלך זמן ההמתנה התווך לא נתפס - המכשיר יוצא לשידור. מנגנון המתנה זה נועד לאפשר למסגרות ששודרו לעבור את המרחק של כל הרשת (Air propagation time) כדי שכלל המכשירים יוכלו לקלוט אותם מבלי שידרסו. בנוסף, המנגנון נותן עדיפות למסגרות "חשובות" יותר לרשת, למשל מסגרות ACK, ע"י שימוש ב-IFS קצר יותר למסגרות כאלו.
* אם התווך תפוס – המכשיר לא מתחיל לשדר, אלא ממתין עד שהתווך יהיה פנוי. כאשר התווך מתפנה, ממתין המכשיר זמן IFS בהתאם לסוג המסגרת.
* אם במהלך המתנת זמן IFS התווך נתפס (המכשיר זיהה תקשורת) – ההמתנה מופסקת, והמכשיר ממתין לתווך פנוי.
* לאחר שהתווך פנוי במשך זמן IFS, המכשיר מגריל זמן random backoff וממתין גם אותו בספירה לאחור. זמן זה הוא כפולה של פרמטר ה-SlotTime, והוא מוגרל מתוך טווח שגדל אקספוננציאלית עם כמות התנגשויות המסגרות (קיימים גם אלגוריתמי backoff נוספים). אם מונה ה-backoff מתאפס והתווך עדיין פנוי - המכשיר יוצא לשידור.
* אם במהלך ספירה לאחור של backoff התווך נתפס - מונה ה- backoff מוקפא. בהמשך, כשהתווך יתפנה, המכשיר לא יגריל backoff חדש אלא ימשיך לספור לאחור מאותה הנקודה, לאחר שימתין זמן IFS

התקן מניח כי זמן ההתפשטות (Air Propagation Delay) קטן משאר הזמנים במערכת (SlotTime, SIFS ועוד). אם כן, אם בתחילת התהליך התווך היה פנוי, אז עבור המתנה ארוכה מספיק מובטח שהתנגשות תקרה רק אם שני המכשירים התחילו בתהליך ה-Carrier Sensing בדיוק באותו ה-time slot. במקרה הזה הם יעברו את השלבים במקביל, יצאו לבסוף לשידור בדיוק באותו הזמן והמסגרות תתנגשנה. לעומת זאת, אם בתחילת התהליך התווך היה תפוס, או שהוא נתפס ברגע מסוים במהלך המתנת המכשירים, אז מובטח שתקרה התנגשות רק אם המכשירים הגרילו את אותו random backoff. בכל מקרה אחר לא אמורה לקרות התנגשות מכיוון שהמכשירים דוגמים את התווך ולא יוצאים לשידור אם הם מוצאים שהוא תפוס. לכן, מנגנון ה-CSMA/CA פותר את בעיית ההתנגשויות בהסתברות גבוהה כאשר כל המכשירים קולטים זה את זה וכאשר זמן ההתפשטות (Air Propagation Delay) קטן ביחס לערך ה-SlotTime.

אולם, לא תמיד זה המצב ברשתות WiFi גדולות. למשל, קיימת בעיית ה-Hidden Node. זהו מצב בו מכשיר מסוים A לא מודע לקיומו של מכשיר אחר C מכיוון שהוא לא נמצא בטווח ה-Carrier Sensing שלו, כלומר רדיוס הקליטה של A לא מכסה את כל ה- BSS ובפרט לא את C. איור להמחשה:



במקרה כזה, ייתכן מצב בו תחנה A שידרה הודעה לתחנה B, ובו זמנית תחנה C שידרה הודעה לתחנה A. B לא מודעת להודעה ששודרה על ידי C ולהפך, ולכן שתי התחנות יניחו שההודעות שודרו כשורה ויפספסו את ההתנגשות, למרות ש-B לא תקבל אף אחת מההודעות בצורה תקינה! לכן ברשת אלחוטית לא ניתן לזהות התנגשויות (ולכן לא ניתן להשתמש במנגנון ה-CSMA/CD). הדרך בה מתגברים על התנגשויות שכבר קרו היא באמצעות ACK timeout שיגרור retransmit.

מנגנון שנועד לסייע במניעת התנגשויות כאשר יש ה-hidden node הוא  
 NAV – Network Allocation Vector. מנגנון זה משתמש בשדה ה-duration הקיים במסגרת, שדה המציין את משך הזמן המשוער עד לסיום שידור המסגרת. באמצעות מידע זה, יכול ה-NAV להעריך את הזמן בו הערוץ יהיה תפוס מרגע תחילת שידור המסגרת (ה-duration שלו בתוספת זמני שידור ACK ובמידת הצורך גם RTS, CTS תוך התחשבות ב-propagation delay). כך ה-NAV יודע לכמה זמן הערוץ יהיה מושבת, ומתי ניתן לאפשר שליחת הודעות שוב.

מצב נוסף בו מנגנון ה-CSMA/CA המוגדר בתקן לא יעבוד כנדרש, הוא מצב בו המכשירים ברשת נמצאים במרחק גדול מאוד זה מזה, ולכן זמן ההתפשטות באוויר גדול. במקרה כזה קיים הפרש בין זמן שליחת מסגרת על ידי המקור לזמן קבלתה על ידי היעד. זהו המצב בו אנו מתמקדים.  
הזמנים (SIFS, SlotTime) בהם יש להשתמש במנגנון ה-CSMA/CA נגזרים מגודל הרשת. לזמנים אלו יש השפעה גדולה על הצלחת המנגנון במניעת התנגשויות. הזמנים שנקבעו בתקן [2] מבוססים על ההנחה שזמן ההתפשטות קטן מאוד (עד ). כך למשל ערך ה-SlotTime בתקן [2] מחושב על פי הנוסחה הבאה:

ניתן לראות שהערך מושפע מזמן העיבוד של שכבת ה-MAC את הפריים האחרון שהתקבל, זמן השינוי מסטטוס "receiver" לסטטוס "transmitter" וכן מה-Air Propagation Delay.  
כאשר מדובר בלינקים קצרים (מרחק שלא עולה על 1.35km בין המכשירים המתקשרים, כאשר משתמשים בתקן ac), זמן ההמתנה המחושב על פי ערך ה-SlotTime שנקבע בתקן מספיק, מכיוון שה-Air Propagation Delay למרחקים קצרים זניח. אולם עבור לינקים ארוכי טווח, על פי מאמר [3], מנגנון ה-CSMA נוטה להיכשל. בלינקים מעין אלו, ה-Air Propagation Delay כבר לא זניח, ניתן למצוא סף מרחק שהחל ממנו מתקיים Default SlotTime < Air Propagation Delay (עבור תקן ac הסף הוא 1.35km). במצב כזה, הסיכוי שיקרו התנגשויות בלינק הרבה יותר גדול – זמן ההמתנה לא מספיק כדי לאפשר למסגרת לעבור המרחק של כל הרשת. הדבר מקטין את ה-throughput ככל שהמרחק גדל.  
לכן, עבור לינקים ארוכי טווח, מנגנון ה-CSMA/CA המשתמש בפרמטרים שהוגדרו בתקן נוטה להיכשל.

מטרת הפרויקט היא לחקור כיצד ניתן לשפר את ביצועיו של לינק wifi ארוך טווח מסוג point to point (מרחק של 5km ומעלה בין התחנות המתקשרות), באמצעות שינוי בפרמטרים של מנגנון ה-CSMA/CA בפרוטוקולי 802.11, בדגש על תקנים n ו-ac (תקנים מתקדמים ושימושיים כיום). הפרמטר העיקרי בו אנו מתמקדים הוא ה-SlotTime.

# עבודות קודמות

בעבר בוצעו מגוון מחקרים בתחום הרשתות האלחוטיות של תקן 802.11. רשתות בתצורות שונות   
(Ad Hoc/Infrastructure), בתקנים שונים (b/n/ac) ובטווחי שידור שונים (Wifi/WiLD) נבחנו בניסויים אמיתיים וכן בעזרת סימולציות שונות.

המאמר [3] מציג מחקר שבו נחקרה תופעת אובדן מסגרות ברשתות WiLD, באזורים כפריים ועירוניים.   
רשתות WiLD הן פתרון זול ליצירת קישוריות בין תחנות במרחקים ארוכים, והשימוש בהן באזורים עירוניים וכפריים הולך וגדל. החיסרון העיקרי של רשתות מסוג זה הוא שהאמינות שלהן עלולה להיות נמוכה מאוד - הרבה מסגרות הולכות לאיבוד ברשת ולא מגיעות ליעדן כנדרש. על פי מדידות שהתבצעו, אחוז אובדן המסגרות הוא בעל variance גבוה ויכול להשתנות בין לינקים שונים.  
אובדן מסגרות יכול להיגרם הן בגלל הערוץ עצמו והן בגלל הפרוטוקול (802.11 MAC protocol). הערוץ גורם לאובדן מסגרות הנובע בעיקר מהימצאות ערוצי Wifi נוספים בטווח, וכן מהפרעות שלא קשורות ל-Wi-Fi ומהפרעות multipath. הפרוטוקול גורם לאובדן מסגרות הנובע מפקיעות timeout, מתקלות במנגנון ה-CSMA ומ-propagation delay.  
כותבי המאמר בחנו שישה לינקים פיזיים במרחקים משתנים, בין 2 ל-20 ק"מ, באזורים כפריים ועירוניים. בנוסף הם ערכו ניסויים באמולטור לרשתות אלחוטיות Spirent 5500 שבהם טווחי השידור היו ארוכים אף יותר. התעבורה עברה בתקשורת 802.11a/b/g.  
נמצא כי קיימות בפרוטוקול עצמו בעיות אשר מגדילות את אחוז איבוד המסגרות.  
ראשית, מנגנון ה-recovery בעייתי. המנגנון שואף לזהות בהקדם האפשרי שמסגרת לא הגיעה ליעדה ולשלוח אותה שוב כדי שהלינק יוכל להמשיך לעבוד כרגיל. המנגנון מבוסס על ACK/Timeout. אולם התגלה כי במרחקים ארוכים השימוש בו דווקא פוגע ביעילות הערוץ. בוצע ניסוי באמולטור (את תוצאותיו ניתן לראות ב-Figure 7). על פי פרמטר ה-timeout שנקבע ברשת, עבור מרחק שגדול מ-110 ק"מ בין המכשירים שמנסים להעביר מסגרת זה לזה, תמיד המסגרת תשלח שנית. זאת מפני שה-propagation delay הגדול גורם לכך שה-ACK לעולם לא יגיע ליעד לפני פקיעת ה-timeout. זה גורם לירידה חדה ביעילות הערוץ (ירידה ב-BW האפקטיבי שנמצא באמת בשימוש להעברת המידע). ללא שימוש ב-ACK's ניתן לראות שהיעילות היא הגבוהה ביותר האפשרית בערוץ בו השתמשו.   
כפועל יוצא לאבחנה זו, ניתן לראות כיצד אפשר לשפר את ניצולת הרשת באמצעות שינוי בפרמטר ה-timeout שלה בהתאם למרחק בין המכשירים המתקשרים.  
בנוסף, נמצא כי מנגנון ה-CSMA נוטה להיכשל במרחקים ארוכים. בשל ה-propagation delay הגדול במרחקים ארוכים, מנגנון ה-CSMA נכשל פעמים רבות מפאת הבנה מאוחרת מידי של התחנות שהערוץ עמוס. זה מוריד את ה-throughput ככל שהמרחק עולה.   
פתרון אחד לבעיה זו הוא שימוש בשיטת ריבוב הנקראת TDMA, מטרתה לשפר את CSMA במרחקים ארוכים. לפי שיטה זו, מחלקים את הזמן בין המשתמשים ברשת וכל משתמש משדר בזמן שהוקצה לו. ניתן לחלק את הזמן מראש, בצורה קבועה לכל משתמש, או לחלק באופן דינמי בהתאם לצורך של כל מכשיר (במקרה הזה יש צורך ברכיב שינהל את חלוקת הזמנים בין המכשירים). פתרון נוסף הוא שימוש ב-cumulative ACK's.   
כיוון אחר לפתרון הבעיה להשתמש במנגנון ה-CSMA/CA תוך שינוי פרמטר ה-slot time בהתאם ל-propagation delay ברשת. ה-SlotTime קובע את זמן ההמתנה של מכשיר לפני שידור, המתנה שנועדה לאפשר למסגרות לעבור את כל המרחק של הרשת. מאחר וערך ה-SlotTime המוגדר בתקן מבוסס על ההנחה שזמן ההתפשטות זניח, שינויו בהתאם לגודל זמן ההתפשטות, הנגזר מאורך הלינק, יכול למנוע התנגשויות. במאמר [1] הכותבים מציעים להגדיל את ערך ה-SlotTime ל- של הלינק (ערך המחושב באופן הבא, בהתאם לאורך הלינק: ). ערך זה מבטיח שכל מכשיר ימתין מספיק זמן כך שהוא יספיק "לשים לב" למסגרות שמגיעות מהמכשיר השני לפני שיצא לשידור. אולם החיסרון של ערך SlotTime כה גדול הוא שיש המתנות ארוכות יחסית לפני שליחת המידע, וזה עלול להיות בזבוז זמן שבו יכולנו להעביר עוד מידע ולהגדיל את ה-throughput של האפליקציה. לכן קיים Tradeoff : ככל שמגדילים את הSlotTime- פוגעים בביצועים כי מחכים יותר זמן, אבל מקטינים את הסיכוי להתנגשויות.  
בפרויקט נבחן את הקשר בין גודל ה-SlotTime כפונקציה של ה-propagation delay להצלחת מנגנון ה-CSMA.

# העבודה שלנו

## דוקומנטציה של הסימולטור הראשון שלנו - סימולטור Real Time

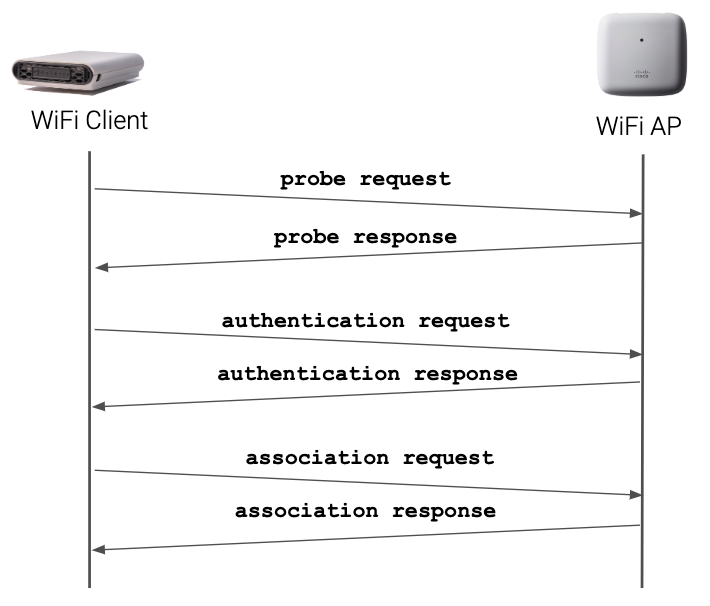
הסימולטור בנוי בגישה מונחית עצמים (OOP), הקוד כתוב בשפת Java, סביבת העבודה היא IntelliJ Idea 2019.2.   
מטרות כתיבת הסימולטור היו לימוד התקן והכרתו וכן בניית פלטפורמה שתאפשר הרצת ניסויים על קונפיגורציות שונות של פרמטרים של הפרוטוקול, כדי לחקור דרכים לשפר את הפרוטוקול.   
להלן קישור ל-repository ב-GitHub שמכיל את הקוד של הסימולטור:https://github.com/Lilach99/Simulator\_Wifi  
מאחר והסימולטור עובד ב-real time ומערב תכנות מקבילי, באופן טבעי הוא אינו פשוט למימוש ולהרחבה. יכולותיו מוגבלות בשל מורכבות המשימה.

**מפרט המחלקות מהן מורכב הסימולטור**

* Network - אובייקט שמייצג את הרשת, ה"עולם" בו התקשורת מתקיימת. מכיל נקודת גישה (AP), רשימת מכשירים שמחוברים לרשת (כלומר ל-AP), רשימת ערוצים שמקשרים בין המכשירים המחוברים לרשת (פירוט בהמשך), ותכונות נוספות (הניתנות לקונפיגורציה) של הרשת.
* Device - אובייקט המייצג מכשיר. בעל תכונות ניתנות לקונפיגורציה, כמו rate, וכן מכיל מידע נחוץ כמו הרשת אליה הוא מחובר, הערוץ דרכו הוא מחובר אליה, באפר של מסגרות אותן הוא רוצה לשלוח ועוד. אובייקט זה מכיל מופע של אובייקט בשם InputHandler - יחידה עצמאית ל"האזנה" למסגרות ולטיפול בקלט. ה-InputHandler מממש Interface שנקרא InputListener - כלומר הוא "מאזין" למסגרות שמגיעות מערוץ התקשורת המחבר אותו למכשיר אחר, ובהגעת מסגרת, דואג לשמור אותה וכן ליצור חבילת Ack מתאימה במידת הצורך.
* AP - אובייקט אשר יורש מ-Device. זהו סוג מיוחד של מכשיר אשר מהווה נקודת גישה ברשת. ההבדל בינו לבין מכשיר רגיל הוא שהוא מזוהה עם רשת מסוימת ומכשירים יכולים לתקשר רק איתו (בתצורת Infrastructure לפחות). AP מאזינה למסגרות מהמכשיר אליו היא מחוברת.
* Medium - אובייקט המייצג תווך, ערוץ תקשורת דו כיווני באוויר בין שני מכשירים. מכיל פרמטרים שונים הניתנים לקונפיגורציה: שני מכשירים ביניהם הוא מחבר, המרחק ביניהם (אורך הלינק), הסתברות להפלת מסגרת בדרך וכן שני באפרים המכילים את אינטרוולי הזמן בהם התווך תפוס, כל באפר מתאים לנקודת המבט של אחד המכשירים (לשימוש במנגנון ה-CSMA/CA). בנוסף התווך מממש Interface שנקרא TransmissionListener - כלומר הוא מאזין למסגרות הנשלחות על ידי כל אחד מקצותיו, ומשרת אותן לפי הפרוטוקול, כלומר מנסה להעביר כל מסגרת המגיעה אליו ליעדה (ההצלחה תלויה במספר פרמטרים, כמו הסיכוי להתנגשות והסיכוי לאיבוד מסגרת בדרך).
* Packet - אובייקט המייצג מסגרת, אנלוגי ל-frame בפרוטוקול 802.11. ישנם שלושה סוגי מסגרות היורשים מ-Packet: מסגרות Control (למשל ACK), מסגרות Management (למשל probe, auth, assoc), מסגרות Data (תקשורת רגילה שעוברת ברשת). מסגרת מכילה חותמות זמן לזמן יצירתה, זמן שליחתה וזמן הגעתה ליעד, וכן פרמטרים המאפיינים אותה: מקור, יעד, אורך, תוכן, האם השולח מצפה ל-ACK עליה.
* Standard - אובייקט המייצג את התקן לפיו פועלת הרשת - n או ac. מכיל את הפרמטרים של התקן, כגון תדר הפעולה, מרווחי הזמן במנגנון CSMA/CA (כגון slot time, congestion window minimum and maximum sizes, SIFS and so on).

### אופן עבודת הסימולטור

הסימולטור מאפשר ביצוע הדמיה של תקשורת אלחוטית peer to peer לפי פרוטוקול 802.11, בסטנדרטים n, ac לפי בחירת המשתמש. ניתן ליצור רשת ולחבר ל-AP של רשת זו מכשיר, ולקנפג את פרמטרי הלינק שנוצר וכן את פרמטרי המכשירים המחוברים.  
החיבור כולל "לחיצת יד" בת שלושה שלבים, בדומה למה שקורה בפרוטוקול 820.11:



בתום שלב זה, המכשירים יכולים להתחיל לתקשר, כלומר להעביר מסגרות זה לזה.

כאשר מכשיר רוצה לשלוח מסגרת, מתרחשים השלבים הבאים:

* המכשיר מכין את המסגרת - נקרא לה packet.
* המכשיר קורא לפונקציה sendPacket עם המסגרת packet.
* הפונקציה ממתינה לרגע בו התווך יהיה פנוי (מנגנון CSMA/CA). אם התווך עמוס, מוגרל random backoff שהוא זמן שהמכשיר מחכה לפני שהוא בודק שוב האם התווך פנוי.
* כאשר התווך פנוי, המכשיר ממתין IFS Time על פי סוג המסגרת, ותוך כדי ההמתנה מוודא שהתווך לא נתפס.
* אם לאורך כל זמן ההמתנה עד סופו התווך פנוי - המכשיר מעביר לאובייקט התווך המחבר בינו ובין יעד המסגרת את packet.
* אם התווך נתפס במהלך ההמתנה - המכשיר חוזר שלושה שלבים אחורה וממתין שוב לרגע בו התווך יהיה פנוי.
* התווך המקבל הודעה על הגעת המסגרת (כי הוא מממש TransmissionListener) בודק שנתוני המסגרת מתאימים לו, כלומר שהיעד והמקור של המסגרת הם שני קצותיו. אם לא - הוא מחזיר false - השליחה נכשלה.
* אם כן - הוא מוסיף לבאפר האינטרוולים מנקודת המבט של היעד אינטרוול חדש בו הוא תפוס. האינטרוול הוא   
  זאת מפני שזהו הזמן בו מבחינת היעד הערוץ יהיה תפוס - לפני שעובר היעד לא רואה שנשלחה מסגרת ולכן לא יכול לדעת שהערוץ תפוס (מה שעלול לגרום להתנגשויות), ולאורך כל זמן שליחתה שהוא הוא כן יודע שהערוץ תפוס.
* התווך מחשב את הזמן עד שהמסגרת תגיע במלואה ליעד, ומעביר אותו לשולח.
* השולח ממתין את זמן שליחת המסגרת ואז מתבצעת בדיקה שמטרתה לוודא שהמסגרת לא התנגשה עם מסגרות אחרות וכן שהערוץ לא זרק אותה (אחד הפרמטרים של הערוץ הוא ההסתברות שהוא יזרוק מסגרת). אם הכל תקין - המכשיר מודיע לערוץ שניתן לסיים את השליחה.
* התווך מודיע ליחידה שמטפלת בקלט (InputHandler) שהגיעה המסגרת packet (באמצעות פונקציה של InputListener). זמן הגעת המסגרת ליעד, הוא.
* ה-InputHandler של היעד מקבל את המסגרת, בונה חבילת ACK מתאימה אם יש צורך ומכניס אותה לתור המסגרות שעומדות להישלח.
* העדיפות של חבילת Control, למשל Ack, גבוהה יותר מהעדיפות של חבילת Data בתור. לכן חבילת ה-Ack אמורה להישלח ליעד יחסית מהר, בתקווה לפני פקיעת ה-timeout.
* השולח מממש מנגנון Fast Retransmit and Recovery. לכן עם רגע שליחת המסגרת הוא מפעיל Timer למשך זמן ה-Timeout שקונפג עבורו. אם עד תום הזמן לא הגיע ACK על המסגרת - המכשיר שולח אותה שוב, כי הוא מניח שמשהו השתמש בדרך. מספר השליחות החוזרות (Retries) הינו מוגבל וניתן לקינפוג.
* אופן חישוב ה-timeout - נוסחה שנמצאת בשימוש ברשתות היא . השתמשנו בנוסחה זו.
* זיהוי התנגשויות - פונקציה שכל מכשיר מריץ בכל פעם שהוא שולח מסגרת. בתום זמן המעבר של המסגרת כולה בתווך, הוא בודק האם קיים אינטרוול זמן בבאפר האינטרוולים העמוסים מנקודת המבט של יעד המסגרת, אשר נחתך עם האינטרוול של שליחת המסגרת הנוכחית. אם כן - הוא מזהה התנגשות בין המסגרות ולא מסיים את שליחת מסגרת זו.  
  זיהוי ההתנגשויות עובד נכון מפני שאם יש שתי מסגרות שנשלחות על הערוץ בכיוונים שונים ברגע מסוים בזמן - אז הן מתנגשות, וזה המקרה היחיד בו הן מתנגשות.
* אופן הסימולציה - יש להכניס כפרמטר את משך זמן הסימולציה (במילישניות) וכן את ה-rate של כל מכשיר. המכשירים מסכימים ביניהם על הקצב בו יישלחו המסגרות, ולפני תחילת פעולות השליחה, רצה פעולה שמכינה לכל מכשיר PriorityBlockingQueue עם המסגרות שאותן הוא עתיד לשלוח (זו וריאציה של PriorityQueue שעובדת טוב עם threads ו-concurrency). כמות המסגרות שנוצרות ומוכנסות לתור מחושבת על ידי . במשך זמן הסימולציה, כל מכשיר ישאף לרוקן את תור המסגרות לשליחה שלו, ולאחר שיסיים - ימתין שגם המכשיר השני יסיים ושזמן הסימולציה כולו יסתיים.

### תוצאות ניסויים בסימולטור

* השפעת כמות ה-Retries המאופשרת על אחוז איבוד המסגרות בערוצים עם דרגות רעש שונות:  
  נתוני הלינק הנבחן:   
  אורך - 50km  
  ערך timeout - (במילישניות) 210ms  
  קצב שליחה - 2pps  
  זמן סימולציה - 120sec  
  אחוז המסגרות שהצליחו להגיע בהצלחה ליעדן נמדד עבור ערוצים עם ערכים משתנים של NumRetries והסתברות לזריקת מסגרת על ידי הערוץ.   
  תוצאות - הערכים שבתאי הטבלה הם אחוזי ההצלחה של הרשת - כלומר אחוז המסגרות שהגיעו ליעדן בהצלחה והתקבל עליהן ack:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| הסתברות לזריקת מסגרת/NumRetries | 1 | 4 | 7 | 14 |
| 0 | 58.8 | 95 | 100 | 100 |
| 0.25 | 20.21 | 68.3 | 67.5 | 67.1 |
| 0.5 | 12.08 | 31.25 | 32.3 | 30.8 |
| 0.75 | 3.54 | 6.25 | 6.25 | 36.25 |
| 0.95 | 0 | 0.625 | 0.417 | 28.54 |

ניתן לראות שעבור ערוץ "נקי" מרעש, עבור ערך NumRetries של 4 - אחוזי ההצלחה גבוהים ועבור ערך 7 - כל התקשורת עוברת בהצלחה.  
לעומת זאת, בערוצים רועשים, גם עבור הסתברות נמוכה לזריקת מסגרת באופן אקראי, הגדלת NumRetries אינה משפרת את המצב באופן משמעותי עבור הסתברות זריקה של 0.25, מגיעים להצלחה של עד 70%, עבור הסתברות זריקה של 0.5 כבר מצליחים רק ב-30%. עבור הסתברויות זריקה גבוהות, כמו 0.75 ו-0.95, צריך NumRetries גבוה מאוד על מנת לקבל הסתברות הצלחה כלשהי, וגם אז ההסתברות אינה גבוהה מאוד - כ-30% הצלחה.  
המסקנה: השימוש ב-Retries תורם להגדלת הסתברות ההגעה של מסגרת ליעדה בהצלחה, אך גם לו יש מגבלות כשהערוץ רועש. בערוצים רועשים מאוד, יש צורך ב-NumRetries גדול יחסית כדי להגיע לרשת שמצליחה בהסתברות לא זניחה.

## שימוש בסימולטור חיצוני – NetSIm

Netsim - סימולטור מקצועי לרשתות חוטיות ואלחוטיות, תומך בפרוטוקולים רבים, ביניהם 802.11, תקנים a, b, g, n, ac. הוא מאפשר סימולציה של רשת Wi-Fi שמכילה מכשירים ולינקים לבחירת המשתמש, שליטה בפרמטרים רבים של הרשת וחקירתה. בנוסף הסימולטור כולל מספר ניסויים build in שניתן להריץ בו, כמו ניסוי לחקר תופעת ה-hidden node, חקירת QoS ברשת 802.11e ועוד.  
Netsim הוא כלי חזק מאוד לביצוע סימולציות.

בדיקת ההשפעה של תקשורת חיצונית בתדרים סמוכים על ה-throughput של חיבור point to point

נבנתה סימולציה של לינק wifi מסוג point to point - שתי תחנות שמחוברות ביניהן דרך AP, הנמצאות במרחק של 0.5 ק"מ זו מזו. בסמוך לתחנות אלו, פועלות מספר רשתות אלחוטיות בתדרים שונים.  
בחנו את ה-throughput של הלינק כאשר הרשתות הסמוכות משדרות באותו ערוץ wifi כמו הלינק (1), בערוצים סמוכים חופפים (1, 3 ו-4) ובערוצים סמוכים לא חופפים (1 ו-6). כמו כן בחנו את ביצועי הלינק כאשר לא קיימת תקשורת חיצונית מפריעה.   
הניסויים התבצעו במספר תקני wifi שונים: b, n, ac על מנת להשוות גם בין ביצועי התקנים.  
בתקן b התדר בו משתמשים הוא 2.4GHz, בתדר n בחנו את התדרים האפשריים 2.4GHz ו-5GHz. בתדר ac משתמשים בתדר 5Ghz.  
הממצאים: כאשר קיימת תקשורת חיצונית מפריעה, ביצועי הלינק פוחתים משמעותית מבחינת ה-throughput. כאשר התקשורת היא באותו ערוץ wifi בו משתמש הלינק, כלומר באותו תחום תדרים קטן יחסית, הביצועים הם הכי פחות טובים. גם כאשר הרשתות משתמשות בערוצים סמוכים החופפים בחלקם לערוץ בו משתמש הלינק הנבחן, ה-throughput נמוך יחסית ודומה מאוד ל-throughput כאשר הרשתות משתמשות באותו ערוץ בדיוק.  
כאשר הרשתות משתמשות בערוצי wifi שאינם חופפים לערוץ בו משתמש הלינק, למשל אם הלינק משתמש בערוץ 1 והרשתות כולן משתמשות בערוץ 6 - ה-throughput של הלינק הנבחן משתפר משמעותית וביצועיו קרובים מאוד לביצועים במקרה שבו אין תקשורת חיצונית מפריעה כלל.

בדיקת ההשפעה של פרמטרים שונים של הפרוטוקול על ה-throughput האפקטיבי

בחנו את ההשפעות של מגוון פרמטרים של פרוטוקול 802.11 בתקנים n, ac על ביצועי לינק 500m point to point, הן כאשר קיימת תקשורת חיצונית (באותו ערוץ wifi) המפריעה לתקשורת הנבחנת, והן כאשר לא קיימת תקשורת חיצונית מפריעה. כמו כן הסימולציות הורצו עם ספי RTS שונים. סף RTS הוא גודל המסגרת המינימלי בבתים שלפני שליחתו תישלח חבילת RTS.   
הספים שנבחנו: סף 0 - כלומר עבור כל מסגרת נשלח RTS; וסף 8000 - המקסימלי האפשרי (בבתים) כך שברוב המקרים לא תישלח חבילת RTS טרם שליחת המסגרת.

הפרמטרים שקונפגו (מרחק קבוע כרגע בין המכשירים):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| פרמטר | שכבה | ערכים שנבדקו בנוכחות תקשורת חיצונית מפריעה | ערכים שנבדקו ללא נוכחות תקשורת מפריעה | סף/ערך בו/ ממנו ה-throughput גבוה בנוכחות תקשורת מפריעה | סף/ערך בו/ממנו ה-throughput גבוה ללא נוכחות תקשורת מפריעה |
| Retry\_Limit | Data Link | 1, 4, 7 | 1, 2, 3, 4, 7 | 7 (יותר כנראה ישפר, אי אפשר לקנפג) | 4 |
| Packet\_Size | Application | 100, 500, 1460, 2000, 3000, 4000 (Bytes) | 2000, 3000, 40000 (Bytes) | מ- 2000, ה-throughput אינו משתפר כשמעלים את גודל המסגרת! | אין! ה-throughput עולה עם גודל המסגרת. |

מסקנה מניסויי גודל המסגרת - כאשר לא קיימת תקשורת חיצונית מפריעה, כלומר הלינק נמצא בסביבה "נקייה מרעשים" (לפחות מרעשים יזומים של הסימולציה), ככל שמעלים את גודל המסגרת, ה-throughput עולה. זה הגיוני, כי שליחת מסגרת גם כך "תופסת" את הערוץ ולכן עדיף, אם כבר שולחים מסגרת, לשלוח אותה בגודל כמה שיותר גדול, וכך יעבור יותר מידע ברשת ביחידת זמן וה-throughput יגדל.

לעומת זאת, בנוכחות תקשורת חיצונית מפריעה, קיים "סף" על גודל המסגרת שהחל ממנו, ה-throughput אינו משתפר גם אם מגדילים את גודל המסגרת מעל הסף. על פי ניסויים שערכנו, הסף הוא 2000 בתים. עבור גודל מסגרות כזה, מתקבל throughput של כ-4.5MBps, וגם כשמעלים את גודל המסגרת ל-3000 או 4000 בתים - ה-throughput לא משתפר.

הסבר אפשרי לתופעה זו הוא שכאשר יש תקשורת מפריעה אשר "מתחרה" עם הלינק הנבחן על תחום התדרים לשידור, ככל שאורך המסגרת הנשלחת גדול יותר, כך היא חשופה יותר להפרעות מצד רשתות אחרות בסביבה, וכפועל יוצא מכך סיכויי ההתנגשות שלה עם מסגרות מרשתות אחרות - עולים. התנגשויות כמובן פוגעות ב-throughput - פחות מסגרות מצליחות להגיע ליעדן בהצלחה בגלל התנגשויות.   
לכן, בנוכחות רשתות נוספות בסביבת הלינק הנבחן, הגדלה של גודל המסגרת ככל שנבחר לא תמשיך לשפר את ה-throughput לאחר שעוברים את הסף.

שינוי פרמטרי השכבה הפיזית SlotTime ו-SIFS

בוצעה עריכה של קובץ הקונפיגורציה InterfaceProperty.xml על מנת להפוך פרמטרים או לקונפיגורביליים (בין 1 ל-50 מיקרו שניות). הרציונל מאחורי קביעת הערכים לפרמטרים הללו **בתקן** - SIFS הוא קבוע (לסטנדרט) ואמור לגלם בתוכו את "זמן התגובה" של המכשיר, מרגע סיום שידור המסגרת האחרונה בתקשורת הקודמת (שהמכשיר מקבל/רואה), ועד תחילת שידור המסגרת הראשונה בתקשורת החדשה שרוצה המכשיר להתחיל לשדר. לכן נוסחת החישוב לערך ה-SIFS היא, על פי עמוד 843 בתקן:

כלומר ערך ה-SIFS מושפע מהעיכוב ה-RF-י, מזמן העיבוד של שכבת ה-MAC את הפריים האחרון שהתקבל, זמן השינוי מסטטוס "receiver" לסטטוס "transmitter" ועוד עיכוב PLCP. אלו פרמטרים שלא בדיוק תלויים בלינק עצמו, אלא יותר בסטנדרט בו משתמשים ובתווך.  
פרמטר ה-SlotTime, לעומת זאת, מורכב ממה שמורכב ה-SIFS למעט עיכוב ה-PCLP פלוס הזמן עד שתחילת הפריים הראשון שמשודר, אילו היה משודר בסוף זמן ה-SIFS, היה מתחיל להגיע למכונות אחרות ברשת. לכן הוא תלוי גם ב-AirPropagationDelay - פרמטר שמושפע, בין היתר, מאורך הלינק! הנוסחה:

השערה: שינוי בפרמטר ה-SlotTime כתלות באורך הלינק יכול לשפר את ביצועי הלינק, הן בסביבה נקייה והן בסביבה רועשת. לינק ארוך יותר יצטרך SlotTime ארוך יותר, כי ה-AirPropagationDelay בו גדול יותר.

בהשראת המאמר [1],אחד הפרמטרים שאנו סבורים שהתאמתו למרחק בין המכשירים תשפר את ביצועי הלינק, הוא ה-SlotTime - פרמטר של השכבה הפיזית, שיש לו חלק חשוב ביישום מנגנון ה-CSMA/CA. פרמטר זה קובע למשל את גדלי ה-IFS - Inter Frame Spacing. יש לכך חשיבות גדולה למניעת התנגשויות של מסגרות באמצעות carrier sensing נכון. התנגשות מסגרות היא תופעה שמגדילה את ה-overhead שעובר בתווך האלחוטי, דבר שפוגע בביצועי הלינק.  
הבנו שייתכן שהפתרון לשיפור ביצועי לינק ארוך טווח (50km, 100km) הוא הגדלה משמעותית יותר של פרמטר ה-SlotTime. במאמר [1], מתוארת בעיה ב-default value שניתן ל-SlotTime בתקן (עמוד 4 פסקה אחרונה ותחילת עמוד 5). הבעיה היא שנוסחת החישוב לפיה נקבע הערך היא:



הנוסחה כוללת סכימה של מספר פרמטרים של התקן עם ה-Air Propagation Delay.

למרחקים קטנים זה בסדר, כי ה-Air Propagation Delay באמת קטן מאוד.

אולם ככל שהמרחק גדל, ה-Air Propagation Delay גדל גם הוא. הנוסחה ל- Air Propagation Delay היא APD= (Link Lenght)/c כאשר c מהירות האור.

לפיכך, עבור לינקים ארוכים יש להגדיל את פרמטר זה באופן משמעותי מאוד וכתלות ב-Air Propagation Delay:

* ברוחב פס 2.4GHz (תקן n) - ערך ה-SlotTime הדיפולטי הוא , ועבור לינקים של מעל 3km, יש להגדיל אותו כך: . זה הגיוני מכיוון שתחת ההנחה ש- כאשר מהירות האור, מתקיים כי עבור לינק באורך 3km:

מעל מרחק של 3km, מתקיים ובמקרה כזה עלולות לקרות יותר התנגשויות - תחנות עלולות לא לראות בזמן מסגרות שמשודרות על ידי תחנות אחרות, ולצאת לשידור כשהתווך "תפוס", מה שיגרום להתנגשות!  
לכן, כדי לבדוק את ההצעה שבמאמר, בנינו קבוצת ניסויים בסימולטור שרצים על לינק point to point של 100km שפועל בסביבה רועשת, עם Antenna Gain = 50dbi.   
ראשית, ה-SlotTime המומלץ הוא:

הורצו ניסויים בסביבה רועשת, בה נבחנו הפרמטרים הבאים:

* Link Throughput - נפח התעבורה של כל מה שעובר בלינק, ביחידת זמן, כולל מידע שמוסיפות כל השכבות, מסגרות שהתנגשו, שהיו בהן שגיאות, retransmissions - הכל. ערך גדול של פרמטר זה אינו בהכרח סימן טוב - תלוי מה היה אחוז ההתנגשויות ומה ה-Throughput של שכבת האפליקציה. ייתכן Link Throughput גבוה שנובע מאחוז התנגשויות או retransmits גדול, שזה לא טוב ומבזבז משאבים.
* Applications Throughputs - נפח התעבורה שעובר **בהצלחה** רק בשכבת האפליקציה (שזה מה שבמקור רצו לשלוח), ביחידת זמן.
* Collision Percentage - אחוז המסגרות שהתנגשו, מחושב בשתי דרכים - אחוז מסגרות data שהתנגשו (מתוך כלל מסגרות ה-data ששודרו); אחוז מסגרות כללי שהתנגשו (מתוך כלל המסגרות, כולל control).

טבלת סיכום הניסויים:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 27.339 | 42.939 | 0.59 | 0.59 | 4.4 | 20 |
| 8.232 | 15.066 | 0.4 | 0.35 | 2 | 667 |
| 4.391 | 8.372 | 0.275 | 0.275 | 2 | 335 |
| 6.662 | 12.492 | 0.57 | 0.57 | 2.76 | 190 |
| 8.2999 | 15.328 | 0.42 | 0.42 | 2.5 | 230 |
| 4.5801 | 8.759 | 0.59 | 0.59 | 2.75 | 100 |

ניתן לראות שקיים tradeoff בין אחוז התנגשות המסגרות ל-Applications Throughputs. ניתן להשיג אחוזי התנגשות נמוכים מאוד, כלומר רוב המידע שעובר עובר בהצלחה, תחת ערכי SlotTime גבוהים מאוד, אבל זה במחיר של Application Throughput נמוך יחסית.   
ניתן אף להבחין שהערך המוצע בתקן, כלומר הערך הדיפולטי, אינו אופטימלי כלל במקרה של לינקים ארוכי טווח! אמנם ה-Application Throughput מגיע לשיאו עבור ערך זה, אבל מנגד גם ה-Link Throughput גבוה מאוד, כתוצאה מאחוז התנגשויות עצום של כמעט 50% מתוך מסגרות ה-data!  
עבור הערך המוצע במאמר, שהוא הגדול ביותר שנבחן, אחוז ההתנגשויות קטן משמעותית (ל-15% התנגשויות מתוך מסגרות ה-data), וגם ה-Link Throughput קטן - יש פחות overhead שנובע מהתנגשות מסגרות ושליחות חוזרות שלהן. אולם המחיר של שיפור זה הוא הקטנה של ה-Application Throughput, בכ-30%-40%.  
הערך שממזער Link Throughput תחת האילוץ שה-Application Throughput יגיע לשיאו (0.59) - - אחוז ההתנגשויות מתוך מסגרות ה-data בו הוא כ-9%, ובהתאם אין הרבה overhead - הלינק מעביר 2.75Mbps.

* ברוחב פס 5GHz (תקן ac) - ערך ה-SlotTime הדיפולטי הוא , ועבור לינקים של מעל 1.35km, יש להגדיל אותו כך: . זה הגיוני כי עבור לינק באורך 1.35km מתקיים:

מעל מרחק של 1.35km, מתקיים ובמקרה כזה עלולות לקרות יותר התנגשויות כפי שהסברתי.  
בנינו קבוצת ניסויים בסימולטור שרצים על לינק point to point של 50km שפועל בסביבה רועשת, ללא Path Loss (כי המטרה היא גם ככה לבדוק תזמונים).  
ראשית, ה-SlotTime המומלץ הוא:

הורצו ניסויים בסביבה רועשת, בה נבחנו אותם הפרמטרים:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 24.037 | 38.805 | 0.58 | 0.59 | 4.3 | 9 |
| 8.249 | 15.241 | 0.35 | 0.35 | 2.5 | 334 |
| 4.295 | 8.236 | 0.58 | 0.58 | 2.8 | 167 |
| 3.696 | 7.128 | 0.59 | 0.59 | 2.85 | 80 |
| 7.122 | 12.704 | 0.57 | 0.57 | 3 | 240 |

ניתן לראות שקיים tradeoff בין אחוז התנגשות המסגרות ל-Applications Throughputs. ניתן להשיג אחוזי התנגשות נמוכים מאוד, כלומר רוב המידע שעובר עובר בהצלחה, תחת ערכי SlotTime גבוהים מאוד, אבל זה במחיר של Application Throughput נמוך יחסית.   
ניתן אף להבחין שהערך המוצע בתקן, כלומר הערך הדיפולטי, אינו אופטימלי כלל במקרה של לינקים ארוכי טווח! אמנם ה-Application Throughput מגיע לשיאו עבור ערך זה, אבל מנגד גם ה-Link Throughput גבוה מאוד, כתוצאה מאחוז התנגשויות עצום של כמעט 40% מתוך מסגרות ה-data!  
עבור הערך המוצע במאמר, שהוא הגדול ביותר שנבחן, אחוז ההתנגשויות קטן משמעותית (ל-15% התנגשויות מתוך מסגרות ה-data), וגם ה-Link Throughput קטן - יש פחות overhead שנובע מהתנגשות מסגרות ושליחות חוזרות שלהן. אולם המחיר של שיפור זה הוא הקטנה של ה-Application Throughput, בכ-40%.  
הערך שממזער Link Throughput תחת האילוץ שה-Application Throughput יגיע לשיאו (0.59) - - אחוז ההתנגשויות מתוך מסגרות ה-data בו הוא כ-7%, ובהתאם אין הרבה overhead - הלינק מעביר 2.85Mbps. גם - חצי מהערך המומלץ על פי הנוסחה שבמאמר, מקבלים תוצאות טובות - Application Throughput של 0.58Mbps - כמעט השיא; Link Throughput קטן אף יותר של 2.8Mbps וכ-8% אחוז התנגשויות מתוך מסגרות ה-data.

**תוצאות סימולציות דומות על לינקים נוספים:**

לינק של 2km בתקן n, סביבה רועשת (אותו ערוץ, 1), ריצה של 15 שניות, עם PathLoss ובלי Antenna gain (אין צורך, לינק קצר דיו):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 18.096 | 23.968 | 0.89 | 0.59 | 4.1 | 20 |
| 4.548 | 8.696 | 0.59 | 0.59 | 2.75 | 13 |
| 3.346 | 6.351 | 0.59 | 0.59 | 2.74 | 14 |
| 3.827 | 7.339 | 0.59 | 0.59 | 2.74 | 7 |

* לינק של 25km בתקן ac, סביבה רועשת (אותו ערוץ, 36), ריצה של 15 שניות, עם PathLoss ו-Antenna gain של 50dbi:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 27.4009 | 43.015 | 0.59 | 0.59 | 4.6 | 9 |
| 7.825 | 14.514 | 0.46 | 0.46 | 2.755 | 167 |
| 3.877 | 7.465 | 0.59 | 0.59 | 2.87 | 83 |
| 3.692 | 7.121 | 0.59 | 0.59 | 2.87 | 41 |

* לינק של 75km בתקן ac, סביבה רועשת (אותו ערוץ, 36), ריצה של 15 שניות, עם PathLoss ו-Antenna gain של 50dbi:

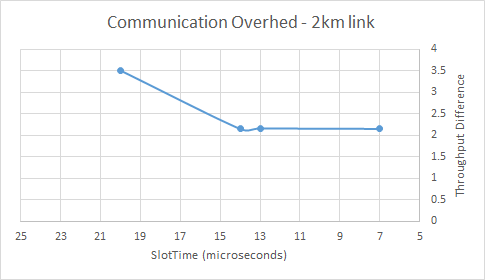
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 27.304 | 42.896 | 0.58 | 0.58 | 4.5 | 9 |
| 10.254 | 18.332 | 0.59 | 0.59 | 2.1 | 500 |
| 9.6303 | 17.569 | 0.25 | 0.25 | 2.25 | 250 |
| 5.884 | 11.114 | 0.58 | 0.58 | 3 | 125 |

נגדיר מדד נוסף לאיכות הלינק - Throughput Difference :

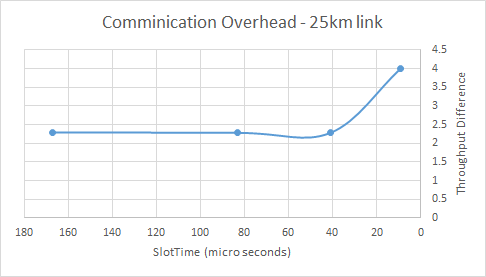
השאיפה היא שהמחסר יהיה קטן והמחוסר יהיה גדול - כך האפליקציה שולחת בהצלחה הרבה מידע ומצד שני ה-overhead של הלינק שנובע מהתנגשויות ומשגיאות, אינו גדול. כלומר השאיפה היא שההפרש יהיה מינימלי, זהו מדד ל-overhead התקשורתי ברשת. אנו מצפים שהוא תמיד יהיה חיובי, כיוון שה-Link Throughput כולל גם מידע ששכבות אחרות מוסיפות למסגרת של שכבת האפליקציה.

גרף שמראה את ה-Throughput Difference כפונקציה של ה-SlotTime עבור לינקים באורכים שונים:

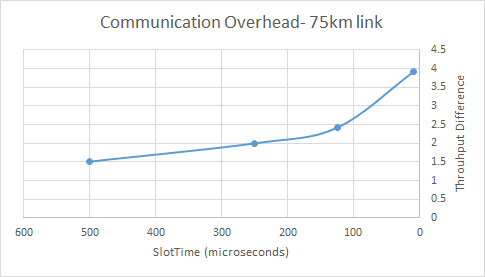
לינק של 2km:

****

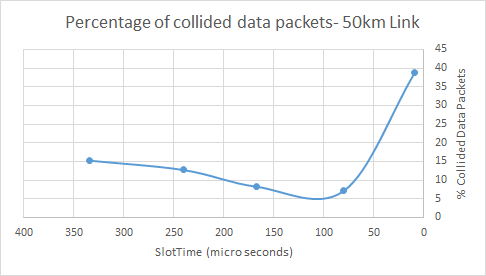
לינק של 25km:



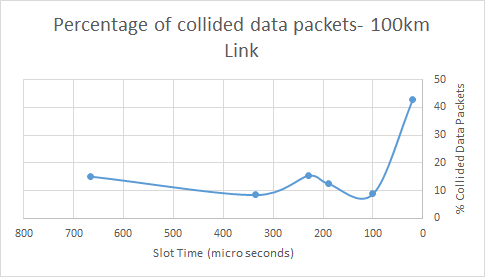
לינק של 75km:



אחוז התנגשות מסגרות (רק מתוך מסגרות ה-data) כפונקציה של Slot Time - ל-50km:



לינק של 100km:



ניתן לראות תופעה מעניינת: עבור לינקים ארוכי טווח, מעל 2km, ביצועי הלינק במובן של אחוז התנגשות מסגרות ו-overhead תקשורתי אינם אופטימליים עבור ערך ה-SlotTime הסטנדרטי שנקבע בתקן. לעומת זאת, עבור ערכי SlotTime גדולים יותר, הביצועים משתפרים (אם כי ברוב המקרים עבור ערך SlotTime גדול מדי, הביצועים שוב פוחתים).

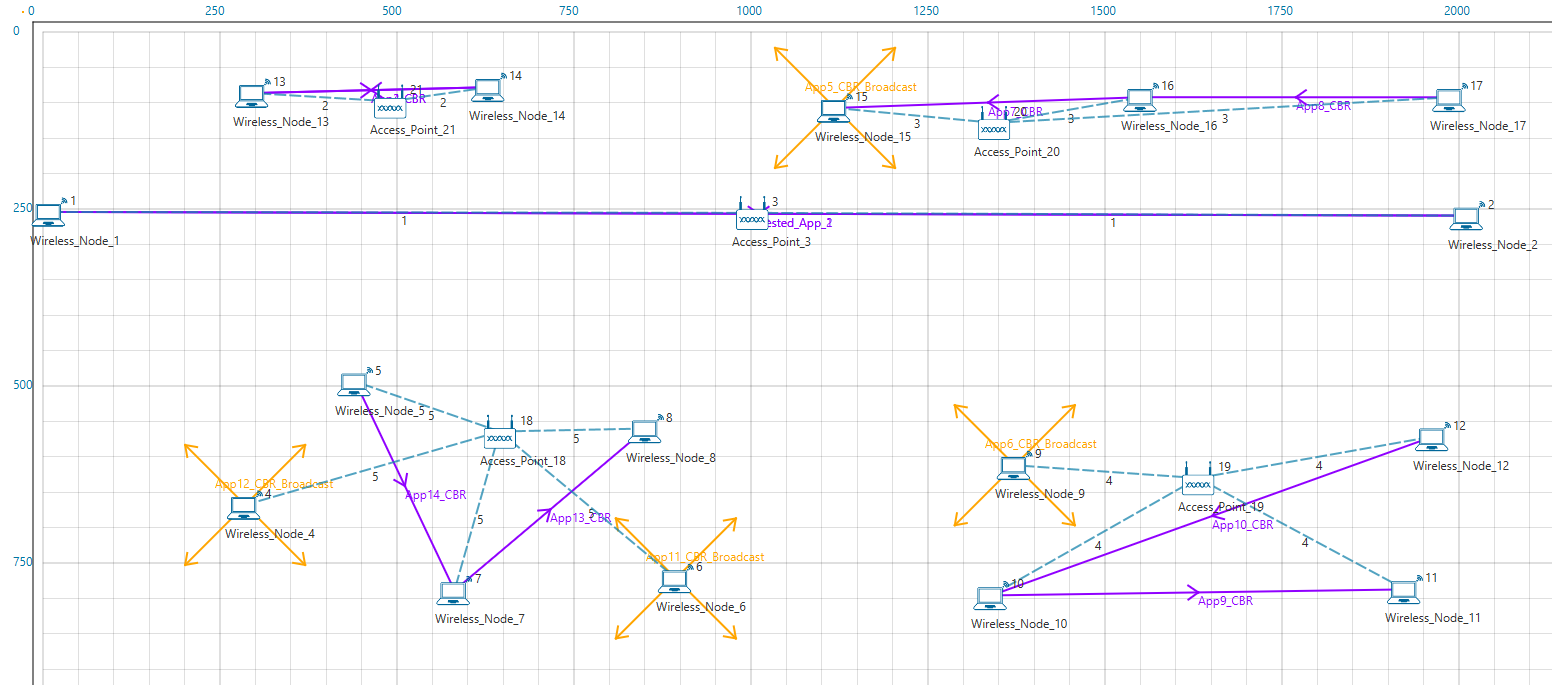
לעומת זאת עבור לינק של 2km, נראה כי ביצועי הלינק אופטימליים עבור ערך ה-SlotTime שבתקן, וכי הגדלת הערך פוגעת בביצועים (ה-communication overhead גדל).   
ניסויים אלה מאששים את השערתנו, שעבור לינקים ארוכי טווח, הגדלת ערך ה-SlotTime מעבר לזה שנקבע בתקן תשפר את ביצועי הלינק.

ניסויים מעמיקים יותר - בעזרת סקריפטים אוטומטיים, כמויות גדולות של ניסויים

ערכנו ניסויים רבים על לינקים ארוכי טווח באמצעות הסימולטור NetSim. זהו סימולטור מקצועי לרשתות חוטיות ואלחוטיות, תומך בפרוטוקולים רבים, ביניהם 802.11, תקנים a, b, g, n, ac. הוא מאפשר סימולציה של רשת Wi-Fi שמכילה מכשירים ולינקים לבחירת המשתמש, שליטה בפרמטרים רבים של הרשת וחקירתה.  
מדדי טיב הרשת שבחרנו על מנת למצוא את ערכי הפרמטרים הטובים ביותר הינם:   
1. Application goodput – נפח התעבורה שעוברת בהצלחה.  
2. Link Throughput – נפח כלל התעבורה העוברת בלינק, כולל retransmissions.  
 Percentage of collisions of data packets .3– אחוז ההתנגשויות של מסגרות data שעוברות בלינק, מתוך כלל המסגרות שעוברות בו, כולל retransmissions.  
בדקנו בסימולציות מספר ערכי SlotTime המביאים בחשבון את גודל ה-עבור לינקים באורכים שונים, הנעים בין 5km ל-100km, ומשדרים מסגרות בגדלים שונים (קצבי שידור שונים).

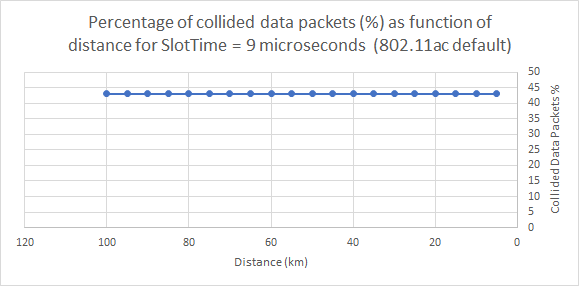
טופולוגיית הרשת בה הורצו הניסויים:

רשת המכילה את הלינק הנבחן, שאורכו משתנה, וכן 4 רשתות קטנות נוספות, שהמכשירים שהן מכילות משדרים באותו ערוץ WiFi כמו הלינק הנבחן, וכך יוצרות הפרעה. שינוי ה-SlotTime התבצע רק עבור מכשירי הלינק הנבחן.

תמונה להמחשה:

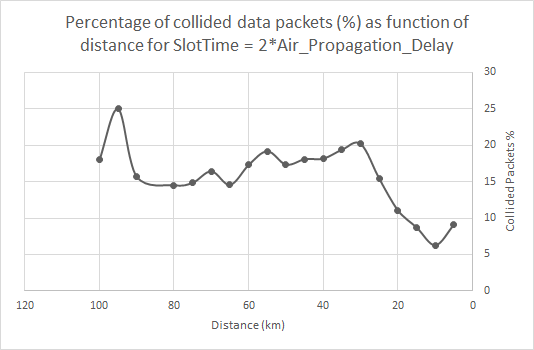
להלן גרפים שמראים את אחוז איבוד המסגרות כתוצאה מהתנגשות כפונקציה של המרחק עבור ערכי SlotTime שונים.

בכל הגרפים, APD הינו קיצור ל- , המחושב ביחס למרחק בין המכשירים המתקשרים, לפי הנוסחה הבאה: כאשר (מהירות האור).

****

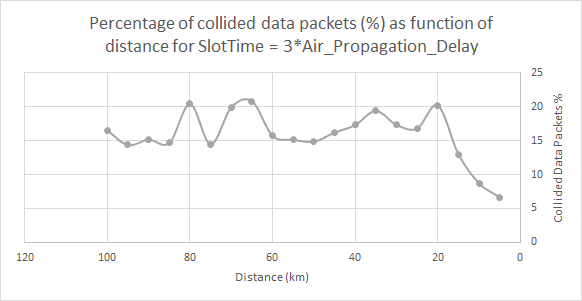
ניתן לראות שהביצועים עבור הערך הדיפולטי שנקבע בתקן לא ממש טובים - כמעט 50% מהמסגרות מתנגשות - overhead רציני ביותר!

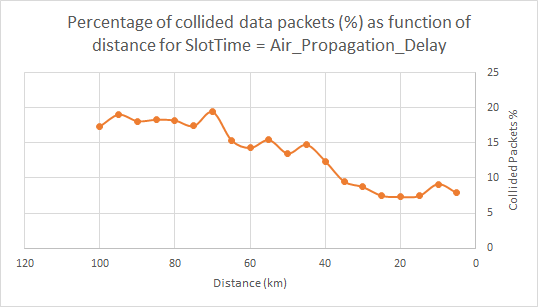
הביצועים עבור SlotTime שנקבע על פי הנוסחה שבמאמר:

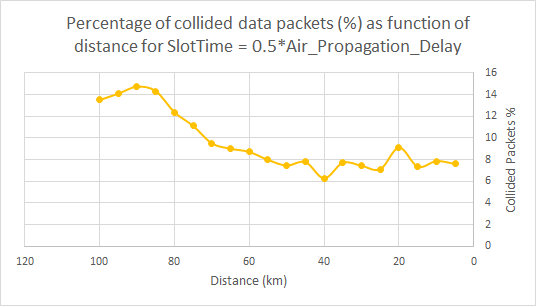


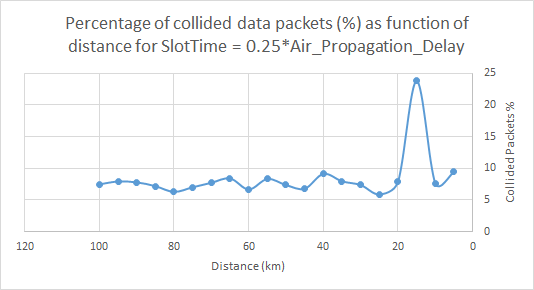
ניתן לראות שיפור משמעותי בביצועים, אחוז המסגרות יורד פי 2 לפחות עבור רוב המרחקים, ובחלק מהמקרים אף פי 3.

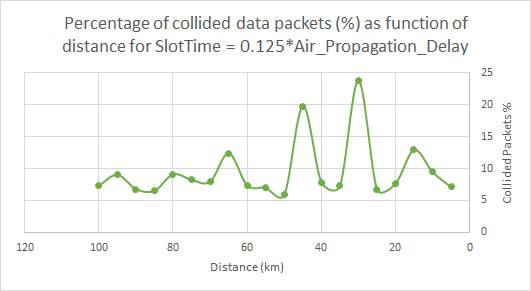
בדקנו מספר נוסחאות דומות ל-SlotTime כפונקציה של ה-Air Propagation Delay וקיבלנו את הגרפים הבאים:



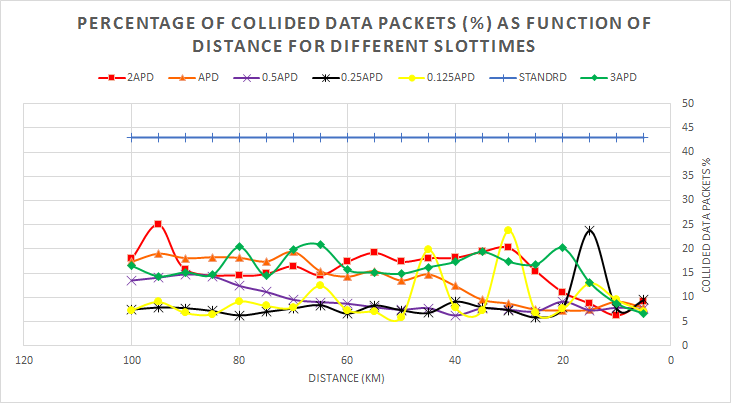




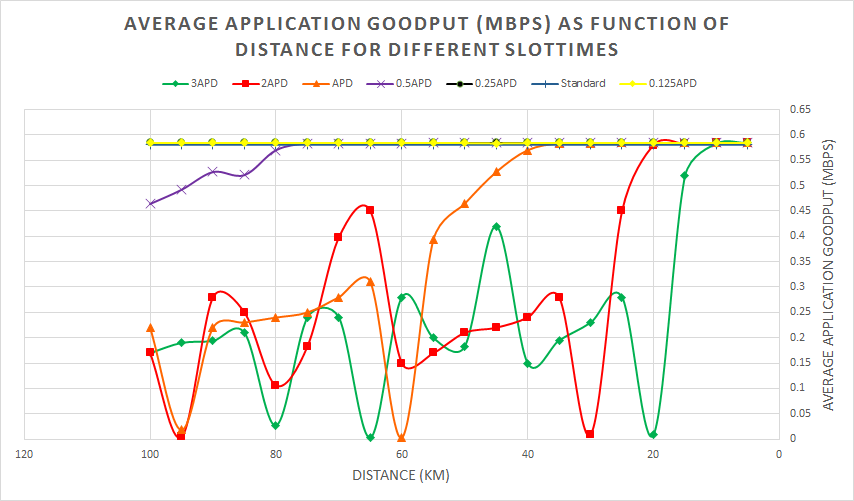




כל הגרפים באותה מערכת צירים:



גרפים גם ל-Goodput (Application Throughput):



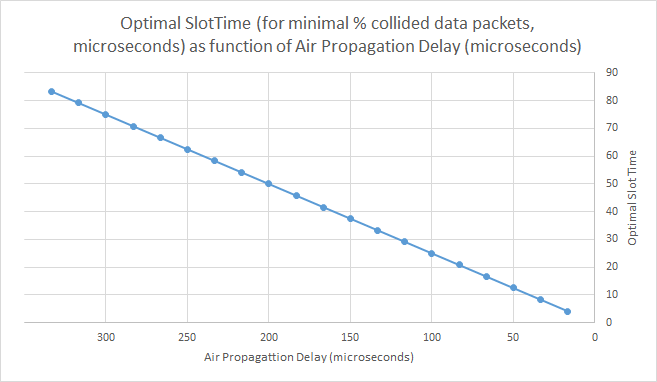
ניתן לראות שעבור ערך ה-SlotTime שבתקן - 9 מיקרו שניות (הקו הכחול בגרף העליון) - אמנם ה-goodput גבוה וקבוע, אבל כ-45% מהמסגרות מתנגשות. הערך הסטדנרטי קטן מה-Air Propagation Delay עבור כל המרחקים שנבחנו (כי כולם גדולים מ-1.35km). הדבר גורם ל-overhead תקשורתי חמור. לכן ערך זה אינו מתאים כלל ללינקים ארוכי טווח.

לעומת זאת, אם בוחרים את ערך הSlotTime- להיות או  
, משיגים את אותו goodput אופטימלי, כאשר יש הרבה פחות התנגשויות! לכן אלו ערכים טובים יותר. עבור הערך (העקומה השחורה בגרף העליון), אחוז ההתנגשויות הוא הקטן ביותר עבור רוב המרחקים, ולכן הוא הערך האופטימלי שהשגנו.

דבר שאינו הגיוני בגרפים אלו הוא שבחלק מהמקרים, ערכי SlotTime קטנים, יש פחות התנגשויות - דבר הנוגד את ההיגיון שבפרוטוקול.

בנוסף הגרפים אינם יציבים והמגמות בהן לא עקביות. לא ברור האם התופעות הללו אכן היו אמורות לקרות, או שהן נובעות מתקלות בסימולציה.

באופן כללי, הקשר בין ה-SlotTime האופטימלי (במובן של goodput - תעבורה שעוברת בלינק בהצלחה, ללא התנגשויות) לבין Air Propagation Delay מוצג בגרף הבא:



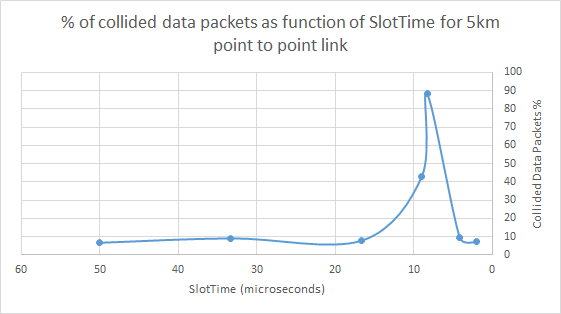
ניתן לראות שהנוסחה ל-SlotTime האופטימלי היא:

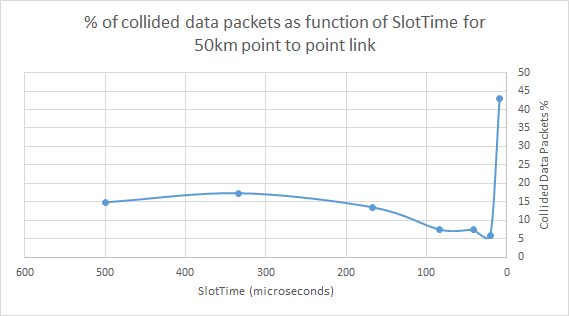
הערה: כמעט בכל המרחקים הביצועים של פקטור 0.25 ופקטור 0.125 היו דומים עד זהים.

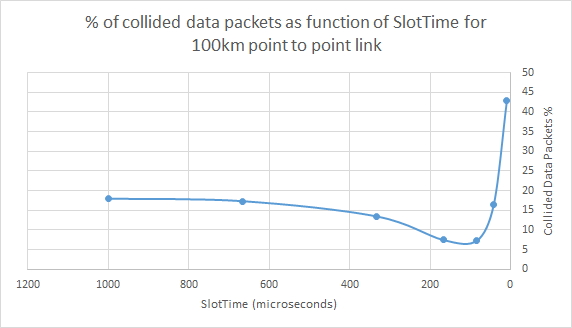
הנוסחה המוצעת במאמר, , הגיונית מאחר ובמרחקים ארוכים, בהם ה-APD גדול, לוקח למסגרות זמן להגיע ממכשיר קצה אחד למכשיר הקצה השני. כאשר משתמשים בערך ה-SlotTime הדיפולטי, סביר שתקרה הסיטואציה הבאה: מכשיר A דגם את התווך במסגרת מנגנון ה-CSMA, וגילה שהוא פנוי. המכשיר ממתין את זמן ה-IFS המתאים, שהוא קטן יחסית כי ה-SlotTime קטן, ואז יוצא לשידור. או אז מכשיר B דוגם את התווך, רואה שהוא פנוי כי בגלל ה-APD הגדול הנובע מהמרחק, המסגרת ששלח A עדיין לא הגיעה לטווח ה-carrier sensing של מכשיר B. גם לאחר ההמתנה של B את זמן ה-IFS, התווך יראה לו פנוי כי המסגרת עדיין לא תגיע לטווח ה-carrier sensing שלו. במקרה כזה, תתרחש התנגשות! תופעה זו מתרחשת באותה המידה (אותו אחוז התנגשויות) במרחקים שונים הגדולים מ-2km.

לכן, הנוסחה המוצעת במאמר הגיונית - היא תגרום לכך שהמכשירים ימתינו מספיק זמן על מנת שהמסגרת תעבור בין המכשירים ושניהם יוכלו לשים לב ולהיות מודעים גם למסגרות שנשלחו אחרי שדגמו את התווך וראו שהוא פנוי. במקרה כזה, תקרה התנגשות רק אם המכשירים ידגמו, ימתינו ואז יצאו לשידור בדיוק באותו הזמן.   
אולם, הערך המוצע בנוסחה גדול מאוד עבור מרחקים ארוכים - יכול להגיע לכ- במרחק של כ-90km. זה המון, גורם להמתנות ארוכות לפני היציאה לשידור ולירידה בניצולת הלינק וב-goodput עקב כך. בנוסף אחוז ההתנגשויות אמנם פוחת אך עדיין ערכו כ-15% - לא ערך זניח.   
לדעתנו הערך המוצע במאמר גדול מדי. ייתכן שעל אף חוסר ההיגיון שבתוצאות, הבחירה בנוסחה תהיה טובה - מחד גיסא, היא גורמת לכל מכשיר להמתין מספיק זמן כך שבהסתברות גבוהה הוא יקלטו מסגרת שמגיעה מהמכשיר השני (היא תגיע לטווח ה-carrier sensing שלו), ומאידך ההמתנה לא תהיה ארוכה מדי כך שלא תהיה פגיעה בניצולת הלינק וב-goodput.

עוד גרפים - אחוז התנגשות מסגרות כפונקציה של SlotTime למרחקים שונים:

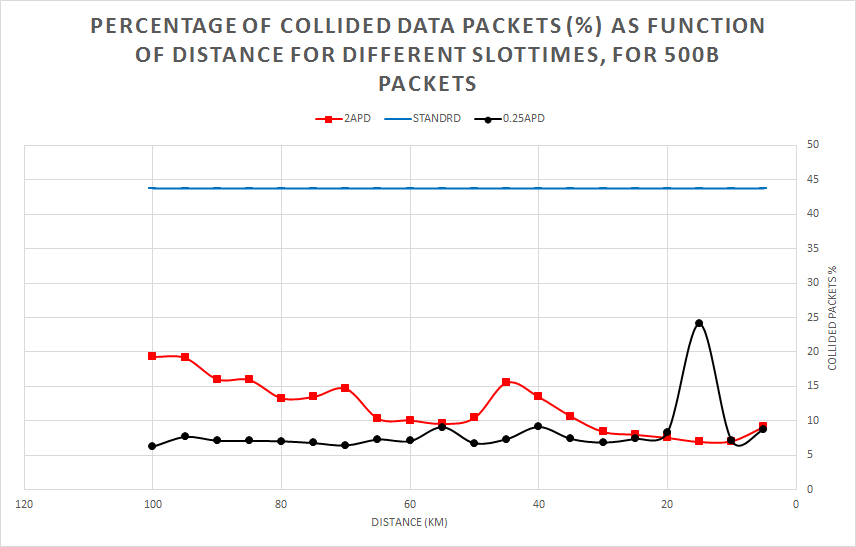


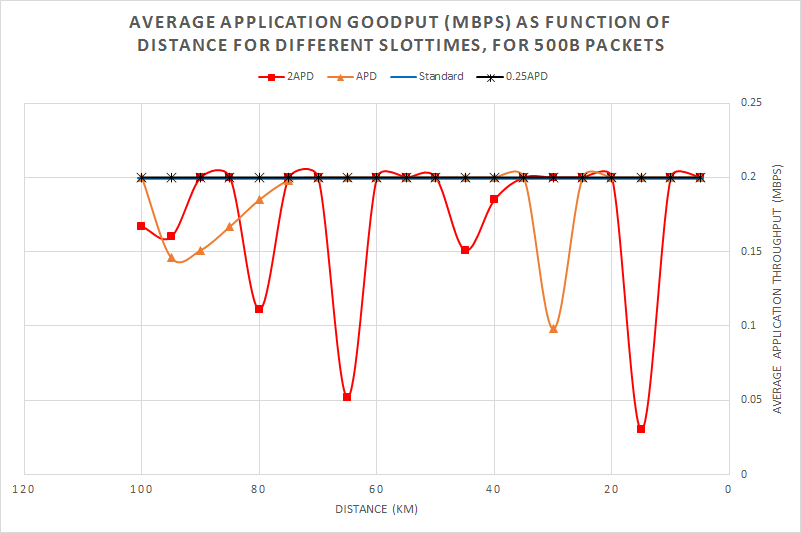


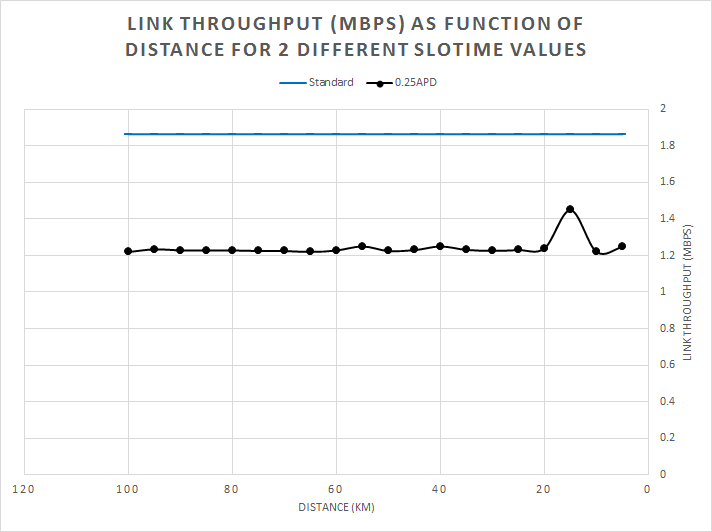


ניתן לראות שעבור שלושת המרחקים, ערך ה-SlotTime שבתקן אינו אופטימלי, והגדלתו (עד ערך מסוים) מקטינה את אחוז המסגרות המתנגשות, בהגדלה מוגזמת אחוז ההתנגשות עולה שוב

ערכנו ניסויים דומים על גדלי מסגרות שונים, ומצאנו שעבור גדלי מסגרות קטנים יותר, למשל 500Bytes (גודל המסגרת המקורי שבחנו היה 1460Bytes), מתקיימים קשרים דומים:

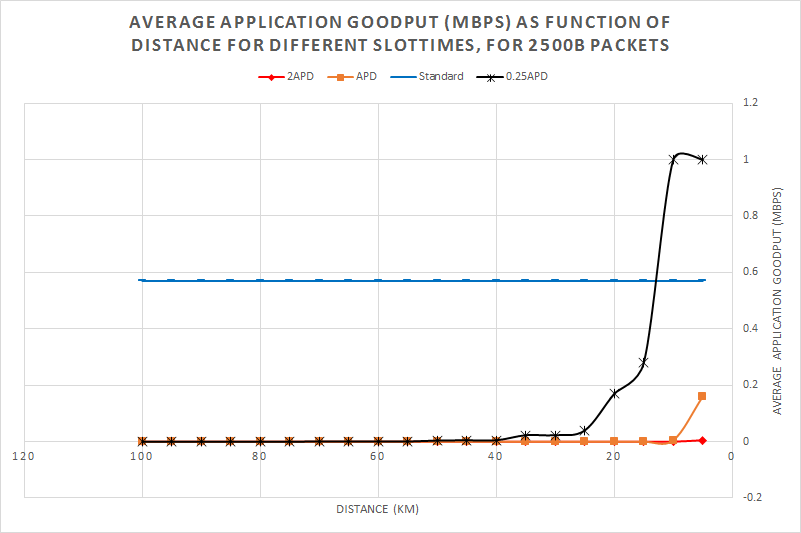






כלומר הנוסחה ל-SlotTime האופטימלי כפונקציה של ה- Air Propagation Delay היא עדיין . ניתן לראות שעבור ערך SlotTime שנקבע על פי הנוסחה, ה-Goodput של האפליקציה מגיע לשיאו (הוא נמוך יותר בהשוואה לניסויים הקודמים מאחר ובהם גודל המסגרת היה גדול יותר, לכן עובר יותר מידע ביחידת זמן - אלו אפליקציות CBR) במקביל ל-Link Throughput ואחוז התנגשויות מזעריים. עבור ערכי SlotTime גדולים יותר, וכן , הערך המוצע במאמר, ניתן לראות בגרף התחתון שיש צניחות רציוניות ב-Application Goodput, אלו ערכים לא יציבים ולא טובים מספיק, על אף שגם בהם אחוז ההתנגשויות נמוך בהשוואה לזה המושג עבור הערך הסטנדרטי.   
חשוב לציין שגם עבור ערך SlotTime סטנדרטי, , ה-Goodput של האפליקציה מגיע לשיאו. החיסרון במקרה הזה הוא שאחוז ההתנגשויות גבוה מאוד - כמעט 50%! זה גורם ל-overhead תקשורתי רציני מאוד. עבור ערך זה, ה-Link Throughput, שכולל את כל המידע שעובר בערוץ, כולל שליחות חוזרות ומסגרות מתנגשות, גבוה מאוד (כ-1.864Mbps לעומת כ-1.225Mbps בממוצע עבור שימוש בנוסחה שאנו מציעים, להשגת אותו Application Throughput - ניתן לראות בגרף השלישי מלמעלה). זה בזבוז משאבים שניתן למנוע על ידי שימוש בנוסחת התאמה של -SlotTime ל-APD.  
קיימות בגרפים נקודות סינגולריות, עליות וירידות חדות. כנראה שהדבר נובע מתקלות בסימולציה. למשל, עבור חלק מן המרחקים, בחלק מהניסויים נשלחו רק מסגרות בודדות בלינק, באופן מפתיע. ייתכן שהדבר נובע מהגדלה קיצונית של ה-backoff בתהליך ה-CSMA/CA, שגרמה להמתנות ארוכות של המכשיר לפני יציאתו לשידור ועקב כך לשליחת כמות מסגרות קטנה מאוד.ניתן לראות שהערך עבור ה-SlotTime משיג Application goodput מרבי במקביל ל-Link Throughput ואחוז התנגשויות מזעריים (פרט למרחק 15km, בו יש עלייה לא ברורה באחוז ההתנגשויות ובהתאם לכך עלייה ב-Link Throughput).

עבור גדלי מסגרות גדולים יותר, קיבלנו דווקא שהגדלת ה-SlotTIme לא משפרת את הביצועים עבור לינקים ארוכים מאוד (מעל 5km-10km). למשל עבור גודל מסגרת 2500Bytes:



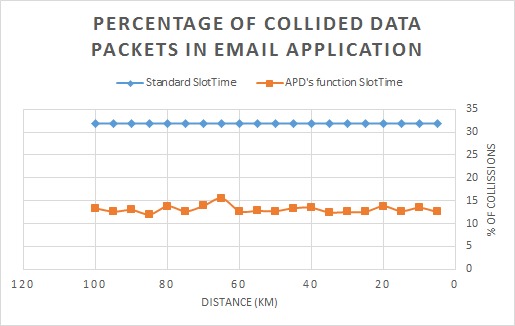
ניתן לראות שעבור מרחקים גדולים מ-10km, בשימוש בנוסחה , יש צניחה של ה-goodput לערך השואף ל-0, וזה מאוד לא טוב. עבור הערך שבתקן כן מקבלים את הערך המקסימלי האפשרי של goodput, אם כי המחיר הוא אחוזי התנגשות מסגרות גבוהים (כ-45% התנגשויות).

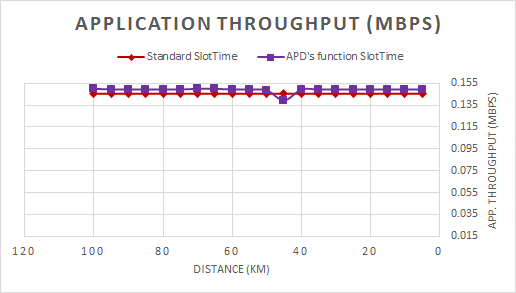
השערה מדוע זה קורה: בתקשורת של מסגרות גדולות בלינקים ארוכי טווח, המכשירים צריכים "לתפוס" את התווך לזמן רב. המתנה של SlotTime ארוך יותר מקשה עליהם להגיע למצב בו התווך נמצא לרשותם, וכשהמסגרות ארוכות, זה משפיע יותר - גם כשהם כבר מצליחים לתפוס את התווך, המסגרת מאוד חשופה להתנגשויות ולהפרעות של תקשורת חיצונית, היא ארוכה ונמצאת יותר זמן באוויר. ייתכן שבשל ההתנגשויות הרבות, ערך ה-random backoff הולך וגדל ב"לולאה אינסופית" ולכן כמות המסגרות הנשלחות קטנה ושואפת ל-0. לכן ה-Application Throughput שואף ל-0.

ערכנו סימולציות גם עבור תעבורה שמתאימה לאפליקציות מסוגים נוספים (מלבד CBR), גם בהן התוצאות לא היו ברורות לחלוטין. פירוט בדפים הבאים.

בדיקות על אפליקציות מסוגים נוספים (מלבד CBR):

**אפליקציית EMAIL** - הלינק ארוך הטווח הנבחן כולל שני Wireless Devices המחוברים ביניהם דרך Wireless AP, ומתקשרים באפליקציית EMAIL מעל WiFi בתקן ac.   
הלינק נמצא בסביבה רועשת, המכילה מספר רשתות ארוכות טווח המתקשרות באפליקציות CBR מעל WiFi בתקן ac באותו תדר (channel).   
נבחנו לינקים במרחקים שונים הנעים בין 5km ל-100km (בקפיצות של 5km), תוך שימוש בערכי SlotTimes שונים: הערך הסטנדרטי, שהוא וכן ערכים שהם פונקציה של ה-Air Propagation Delay של הלינק.  
הגרפים שלהלן מתייחסים ללינק הנבחן:

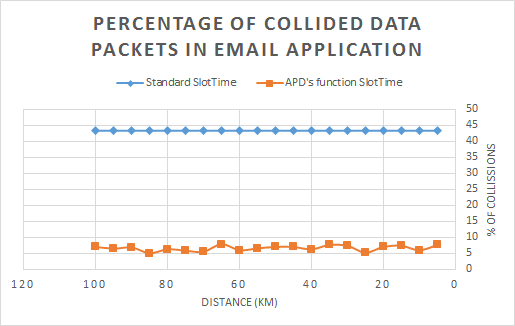


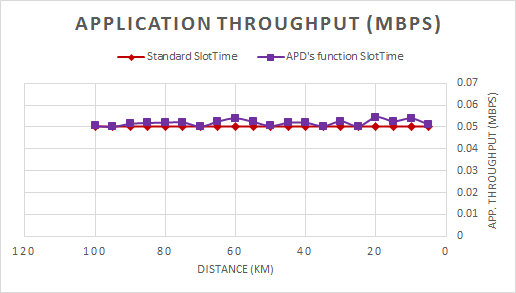


ניתן לראות כי גם באפליקציות EMAIL, שינוי ה-SlotTime לערך המביא בחשבון את ה-Air Propagation Delay בלינק ארוך טווח מקטין משמעותית את אחוז ההתנגשויות ביותר מפי 2!  
כמו כן, השינוי משפר במקצת את ה-Application Throughput ברוב המקרים (מביא אותו לערך אופטימלי - 0.149333, כלומר כל המסגרות נשלחות ומתקבלות בהצלחה).

**אפליקציית Video**: שוב, הלינק ארוך הטווח הנבחן כולל שני Wireless Devices המחוברים ביניהם דרך Wireless AP, ומתקשרים באפליקציית Video מעל WiFi בתקן ac.   
הלינק נמצא בסביבה רועשת, המכילה מספר רשתות ארוכות טווח המתקשרות באפליקציות CBR מעל WiFi בתקן ac באותו תדר (channel).   
נבחנו לינקים במרחקים שונים הנעים בין 5km ל-100km (בקפיצות של 5km), תוך שימוש בערכי SlotTimes שונים: הערך הסטנדרטי, שהוא וכן ערכים שהם פונקציה של ה-Air Propagation Delay של הלינק.

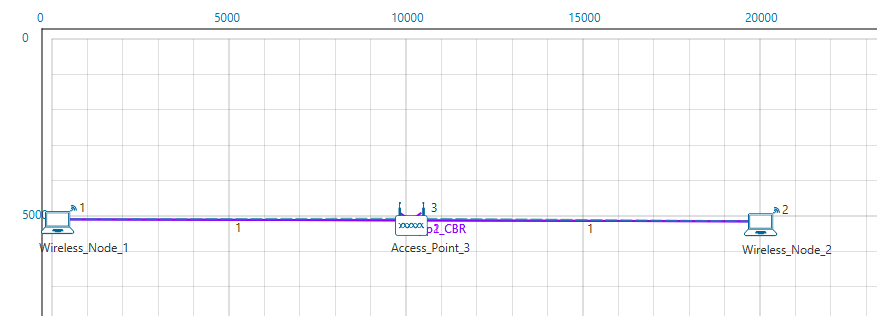
הגרפים שלהלן מתייחסים ללינק הנבחן:



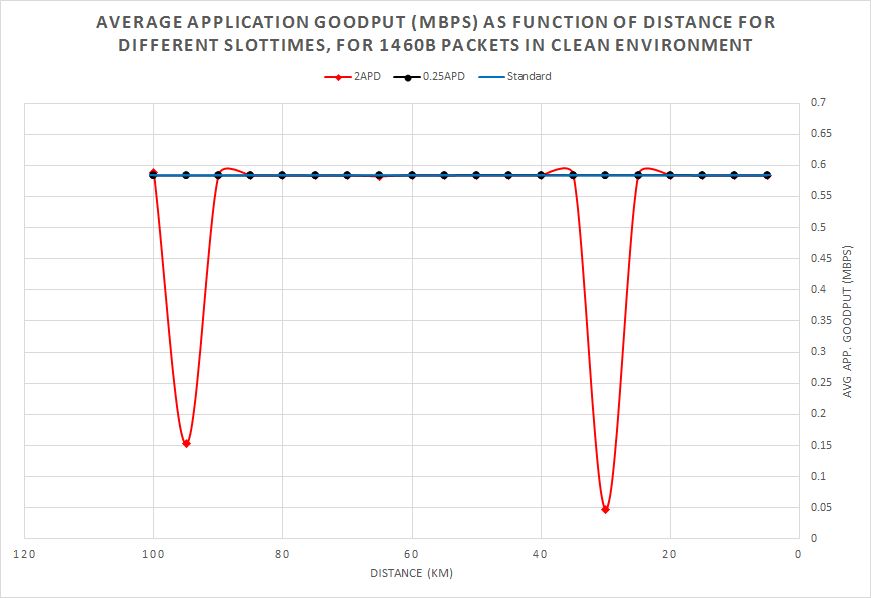


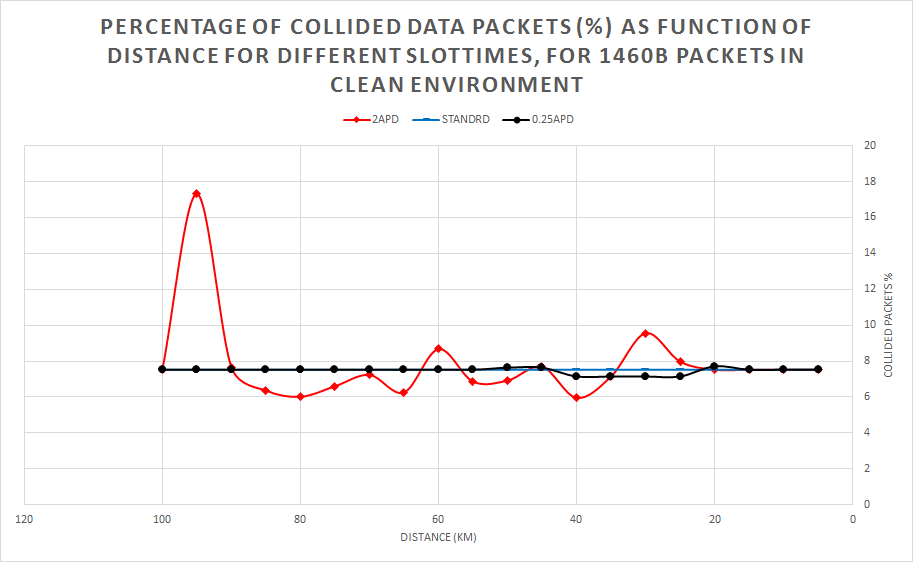
ניתן לראות כי התוצאות דומות גם באפליקציית Video: שינוי ה-SlotTime לכזה שלוקח בחשבון את ה-Air Propagation Delay מגדיל במקצת את ה-Application Throughput וכן מקטין משמעותית (פי 5 לערך) את אחוז ההתנגשויות.

התוצאות מאמתות את השערתנו ששינוי SlotTime במקרה של לינקים ארוכי טווח ישפר את ביצועי הלינק.

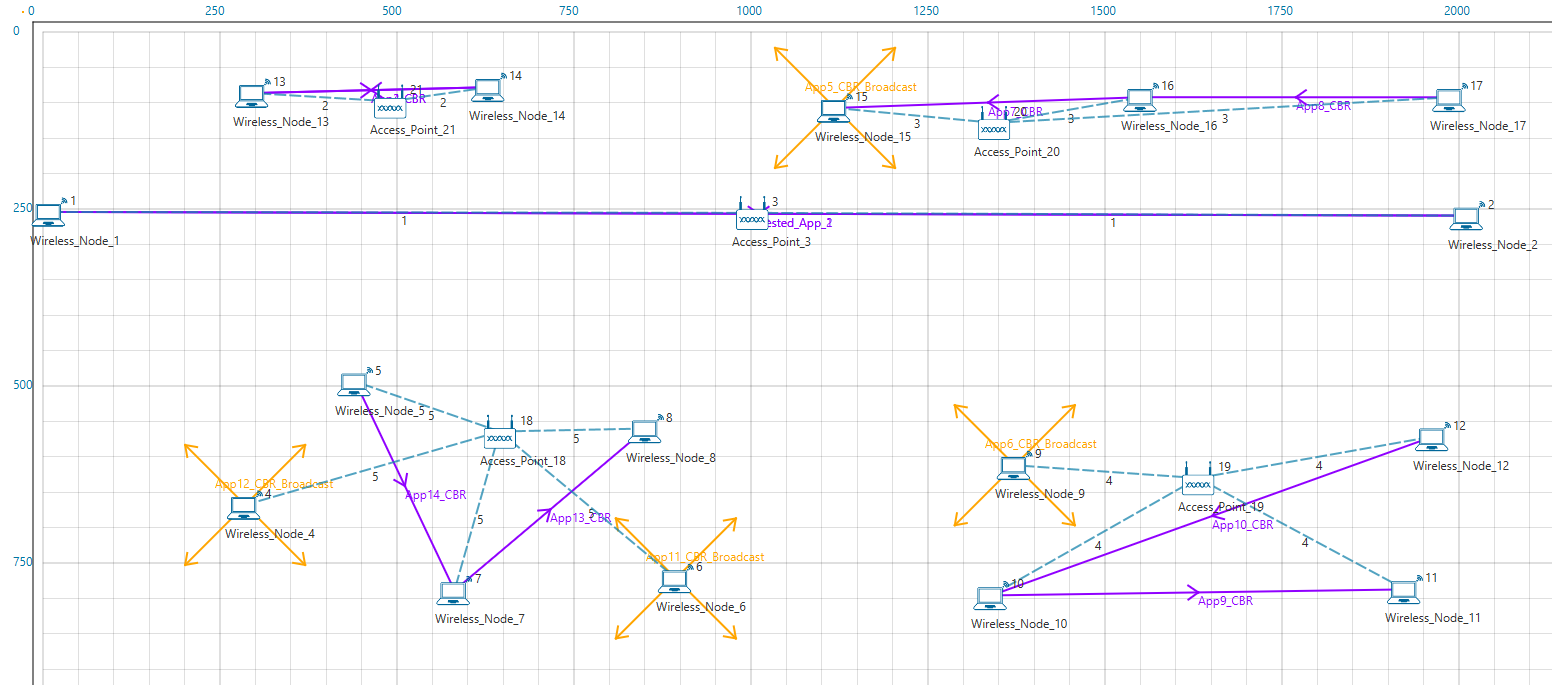
נערכו ניסויים זהים גם על לינק בודד שנמצא בסביבה נקייה מרעשים, כלומר ללא לינקים נוספים, כך:

גרפי התוצאות:



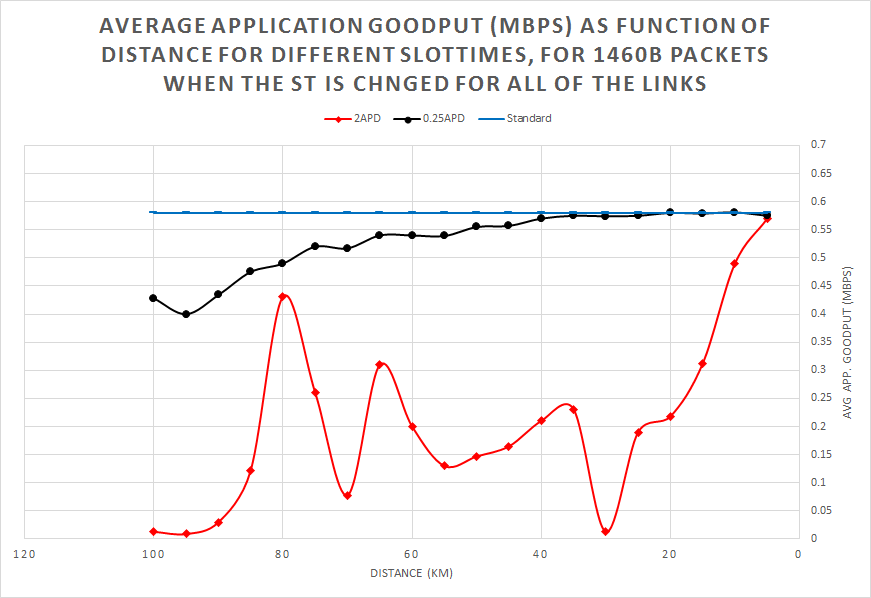


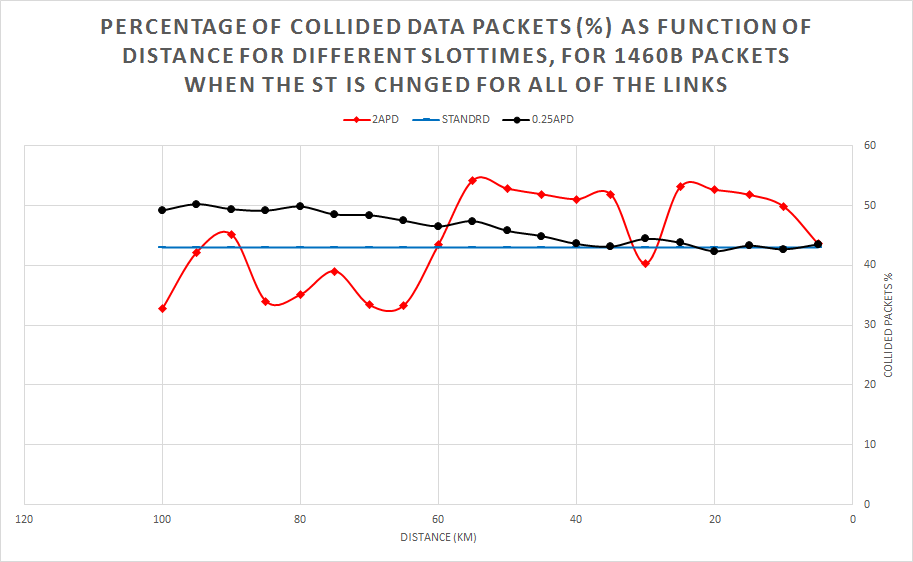
ערכנו גם ניסויים בהם שינינו את ה-SlotTime לערך הנבדק עבור כל הלינקים, כלומר גם עבור הלינק הנבחן וגם עבור הלינקים ברשתות השכנות, שיוצרים את הפרעות על ידי שידור באותו תדר.

תמונה להמחשת הטופולוגיה:

הלינק הנבחן הוא הלינק הארוך ביותר.

גרפי התוצאות:





לסיכום, התקבלו תוצאות לא לגמרי ברורות, ורובן לא תואמות את הציפייה שלנו ואת הידע שלנו לגבי הפרוטוקול. למשל, בחלק מן הגרפים קיימות נקודות סינגולריות לא ברורות וכן מגמות לא עקביות. ייתכן שהדבר נובע מכך שהסימולטור לא תוכנן לעבוד במקרי קצה ובמצבים לא רגילים כמו אלו שאנחנו בדקנו (לינקים ארוכים מאוד, שינוי פרמטרים של השכבה הפיזית).  
בכל זאת, אנו מעוניינים להמשיך לחקור ולערוך בדיקות כדי לקבל תוצאות מספקות. לכן החלטנו לכתוב סימולטור נוסף משלנו, שיהיה פשוט יותר מהראשון שכתבתי. הסימולטור מתמקד במימוש מנגנון ה-CSMA/CA בפרוטוקול 802.11 וכן הוא לא real time. הסימולטור נכתב ב-MATLAB ועל כן הוא single threaded. פירוט לגבי הסימולטור המתוכנן - בדפים הבאים.

## הסימולטור הנוכחי - סימולטור מונחה אירועים

הסימולטור בנוי בגישה מונחית אירועים (Events Based Programming), הקוד כתוב ב-MATLAB וסביבת העבודה היא MATLAB 2020.

**מטרה**: הסימולטור מדמה את תהליך ה-Carrier Sensing והשידור בלינקים point to point ברשת המכילה מספר מכשירים בעלי מרחקים שונים זה מזה (כמות המכשירים והמרחקים ניתנים לקינפוג), עבור ערכי SlotTime ופרמטרים נוספים ניתנים לקינפוג. הדגש הוא על הזמנים בהם מתרחשים אירועים "מעניינים", ועל זיהוי התנגשויות מסגרות המתרחשות בלינק.

### אפיון הסימולטור

**הקלט לסימולטור**:

הקלט מכיל מספר מבני נתונים מסוג struct, כשכל struct מכיל שדות הנוגעים לסוג מסוים של פרמטרים בעלי קשר לוגי ואיזור מסוים בו משתמשים בהם בקוד:

|  |  |  |  |
| --- | --- | --- | --- |
| איפה המבנה משמש בקוד | מה משותף לפרמטרים? | פרמטרים | שם הStruct |
| הפונקציה שמממשמת את התקן משתמשת בהם. יתר הקוד של הסימולציה לא משתמש בהם. ע"י זה שיש cell של מבנים כאלו, אפשר לתת פרמטרים שונים לכל device. המבנים האלו ישמשו לאתחל את מבני המצב של כל מכשיר. | פרמטרים לכל מכשיר. כולל הפרמטרים שמוגדרים בתקן ומגדירים את ההתנהגות של כל התקן. | * SIFS * SlotTime * Number of Retries * Ack Duration * Ack Timeout * Packet Length function | * Device Parameters Struct |
| החלק בסימולציה שממדל את ההתקדמות של השידורים בתווך (מתי התווך ניהיה תפוס בכל מכשיר וכו') | הפרמטרים שמתארים את המבנה הפיזי של הרשת (ללא קשר מי מתקשר עם מי) | * Number of Devices * Distances between Devices (matrix) | * Physical Network Information Struct |
| החלק בסימולציה שמטפל בתקשורת בין הרכיבים (למשל ניהול תורי הפקטות) | הפרמטרים המתארים את התקשורת בין הרכיבים השונים (מי מדבר עם מי ובאיזה קצב) | * Connections Params List (cell array of structs) | * Logical Network Information Struct |
| הקוד הכללי של הסימולציה (כל החלק שלא מטפל בדברים ספצים של הרשת או המכשירים) | הפרמטרים המתארים את ההתנהגות הכללית של הסימולציה ולא קשורים לרשת של המכשירים עצמה. | * finish time * Debug mode (bool) | * Simulation Parameters Struct |

**הפלט של הסימולטור:**

* מבנה נתונים המכיל עבור כל מסגרת מערך structs המכילים את הזמנים הרלוונטיים בכל שידור שלה (התחלה, סיום, זמן הגעה ליעד, זמן התנגשות אם קרתה).
* הזמן הכולל שעבר עד תום הסימולציה (עד שתורי הפקטות של שני המכשירים התרוקנו לחלוטין או זמן הסיום שהתקבל כקלט)
* מבנה נתונים המכיל מידע לגבי התוצאות בכל לינק (חד כיווני) - למשל כמות בתים שנשלחו/התקבלו/התנגשו, מופרדים לפי סוג - control או data.

מתוך נתונים אלו נוכל ליצור:

* גרף זמנים "בינארי" בשני צבעים: לכל מכשיר צבע משלו, בנקודה בזמן שבה מכשיר מסוים משדר מסגרת, מופיע סימון בצבע של מכשיר זה
* גרף שמראה באילו נקודות בזמן התרחשו התנגשויות
* אחוז ההתנגשויות שקרו במהלך הסימולציה
* ועוד

**אופן עבודת הסימולטור - גישה מונחית אירועים - Event based programming**

אנו מייצרים **הפרדה מלאה** בין מכונת המצבים של המכשיר, שבנויה על פי תקן 802.11, לבין הסימולציה עצמה.  
קיים מספר סופי של מצבים אפשריים עבור מכשיר, כאשר בכל רגע בזמן, מכשיר יכול להימצא במצב אחד בדיוק. המעברים בין המצבים נקבעים על פי התקן.

קיימים מספר **אירועי סימולציה** אפשריים, בהם הסימולציה מטפלת בפונקציית הסימולציה.   
אנו מתחזקים מערך דינמי ממוין (cell array) של אירועי סימולציה. כל אירוע הוא struct המכיל את השדות: סוג האירוע, התחנה אליה הוא משתייך, הזמן בו הוא קורה, ושדות אופציונליים (למשל מסגרת).

בהתאם, קיימים מספר **אירועי מכשיר** אפשריים, בהם המכשיר מטפל בפונקציית עדכון מצב מכשיר (פירוט בהמשך).

בנוסף מתחזקים מערך (cell array) שמכיל struct עבור כל **מכשיר**, המתאר את **מצבו הנוכחי** במערכת. בין השאר ה-struct מכיל משתנה המתאר את מצב התווך מנקודת המבט שלו (יש צורך בכך כיוון שאנו מניחים כי ה-APD לא זניח). המשתנה הוא למעשה מונה של כמות המכשירים "שמשתמשים בתווך" כעת, מנקודת מבטו של המכשיר. אם המונה הזה מכיל 0 - התווך פנוי מבחינת המכשיר, אחרת הוא תפוס.  
מבנה נתונים נוסף שנדרש הוא **תור פקטות** לשליחה לכל מכשיר (ייתכן שיהיה נוח להכניס אותו ל-struct של מצב המכשיר).

בסימולטור יש שתי פונקציות עיקריות: **פונקציית הסימולציה**, **פונקציית עדכון מצב מכשיר**.   
פונקציית הסימולציה מקבלת את פרמטרי הקלט לסימולציה. היא מכילה לולאת while שמסתיימת כשזמן סיום הסימולציה מגיע או לפני כן אם כל המכשירים סיימו לשלוח את מסגרותיהם. בכל איטרציה בלולאה, שולפים מהמערך את אירועי הסימולציה המתרחשים ביחידת הזמן הקרובה ביותר, ומטפלים בהם על פי switch על סוג האירוע. הטיפול יכול להכיל יצירת אירועי סימולציה עתידיים נוספים והכנסתם למערך, וכן טיפול במכשירים הרלוונטיים - בפונקציה נפרדת של עדכון מצב מכשיר.  
פונקציית עדכון מצב מכשיר מממשת את מכונת המצבים של המכשיר לפי התקן. היא מקבלת struct מצב מכשיר, struct משתני קלט מהסימולציה וכן אירוע מכשיר. היא מטפלת בעדכון המצב של המכשיר בהתאם למאורע הסימולציה שגרם לה להיקרא, שמיוצג על ידי אירוע המכשיר שהועבר לה כקלט, ומחזירה את מצב המכשיר המעודכן וכן אירועי סימולציה עתידיים להכנסה למערך (במידה וקיימים כאלה).

סוגי ה-structs הקיימים במערכת:

|  |  |  |
| --- | --- | --- |
| שם ה-struct | שדות | הסבר |
| devParams | * SIFS * SlotTime * Number of Retries * Ack Duration * Ack Timeout * Packet Length function | מכיל את כל פרמטרי הקלט הקשורים לתקן |
| phyNetParams | * Number of Devices * Distances between Devices (matrix) | הפרמטרים שמתארים את המבנה הפיזי של הרשת (ללא קשר מי מתקשר עם מי) |
| logNetParams | * Connections Params List (cell array of linkInfo structs) | הפרמטרים המתארים את התקשורת בין הרכיבים השונים (מי מדבר עם מי ובאיזה קצב) |
| linkInfo | * Source Device * Destination Device * Physical Layer Rate * Data rate * Min packet size * Max packet size * Packet dropoff rate | מתאר את התקשורת בין זוג מכשירים |
| simulationParams | * Finish Time * Packet Loss Percentage | מכיל את כל כל פרמטרי הקלט הקשורים לסימולציה |
| packet | * Source * Destination * Length * Type (control/data) | מתאר את השדות החשובים במסגרת |
| devState | * Station * Medium State (counter) * Device Current State (from the state machine) * Number of bytes it successfully sent * Number of bytes it received * Number of lost bytes (bytes which the device failed to send) * Current Packet (the packet that the station is trying to transmit using CSMA/CA or the packet that the station is transmitting now, can be an “empty packet”) * Queue of packets * Time of Backoff Start * Current Backoff * Current retry * Current CWND * CWmax * SIFS * ST (SlotTime) * DIFS(SIFS + 2\*ST) * Max. Number of Retries * ACK Duration * ACK Timeout * Packet Length function | מתאר את המצב הנוכחי של מכשיר ברשת |
| event | * Type * Time * Station * (optional) Packet * (optional, only for TIMER events) Timer type | מתאר מאורע |
| queue | * Tail * FIFO (array) | מתאר מבנה נתונים מסוג "תור" |

מפרט אירועי הסימולציה הקיימים במערכת:

|  |  |  |  |
| --- | --- | --- | --- |
| שם האירוע | הסבר | אירועי סימולציה עתידיים שהוא יוצר | אירוע המכשיר שהוא יוצר |
| START\_SIM | תחילת הסימולציה | GEN\_PACK  בזמן מוגרל (רנדומלי) לכל לינק (ייצור המסגרת הראשונה לשליחה לכל מכשיר ובחירת זמן שליחתה העתידי) | - |
| END\_SIM | סיום הסימולציה, יציאה מפונקציית הסימולציה והחזרת הפלטים הנדרשים | - | - |
| GEN\_PACK | יצירת מסגרת לשליחה בלינק מסוים | GENֹֹֹֹ\_PACK  נוסף לאותו לינק אחרי  inter arrival time | PACKET\_EXISTS למכשיר השולח |
| SET\_TIMER | חלף זמן מסוים שהמכשיר ביקש לספור (שימושי לספירת ACK TIMEOUT וכן DIFS ו-BACKOFF) | - | TIMER\_EXPIRED |
| CLEAR\_TIMER | יש לבטל Timer פעיל של מכשיר מסוים (אולי גם מסוג מסוים) | - | - |
| TRAN\_START | מתרחש שידור בתווך, של מסגרת שפרטיה ידועים | לכל המכשירים פרט למקור המסגרת נוצר אירוע סימולציה RECֹֹׁ\_START אחרי זמן APD (ה-APD מאפיין זוג מכשירים, נלקח ממטריצה מתאימה). למכשיר המקור נוצר אירוע סימולציה  TRAN\_END עתידי. | - |
| TRAN\_END | מסתיים שידור של מסגרת שפרטיה ידועים | לכל המכשירים פרט למקור המסגרת נוצר אירוע סימולציה RECֹֹׁ\_END אחרי זמן APD (ה-APD מאפיין זוג מכשירים, נלקח ממטריצה מתאימה). מכשיר המקור מעודכן שהשידור הסתיים בעזרת מאורע מכשיר מתאים. | TRAN\_END |
| REC\_START | מתחילה קבלת מסגרת, שפרטיה ידועים (בפרט סוגה - data או control) | - | REC\_START |
| REC\_END | מסתיימת קבלת מסגרת, שפרטיה ידועים | - | REC\_END |
| COLL\_INC | הגדלת מונה ההתנגשויות של הסימולציה | - | - |

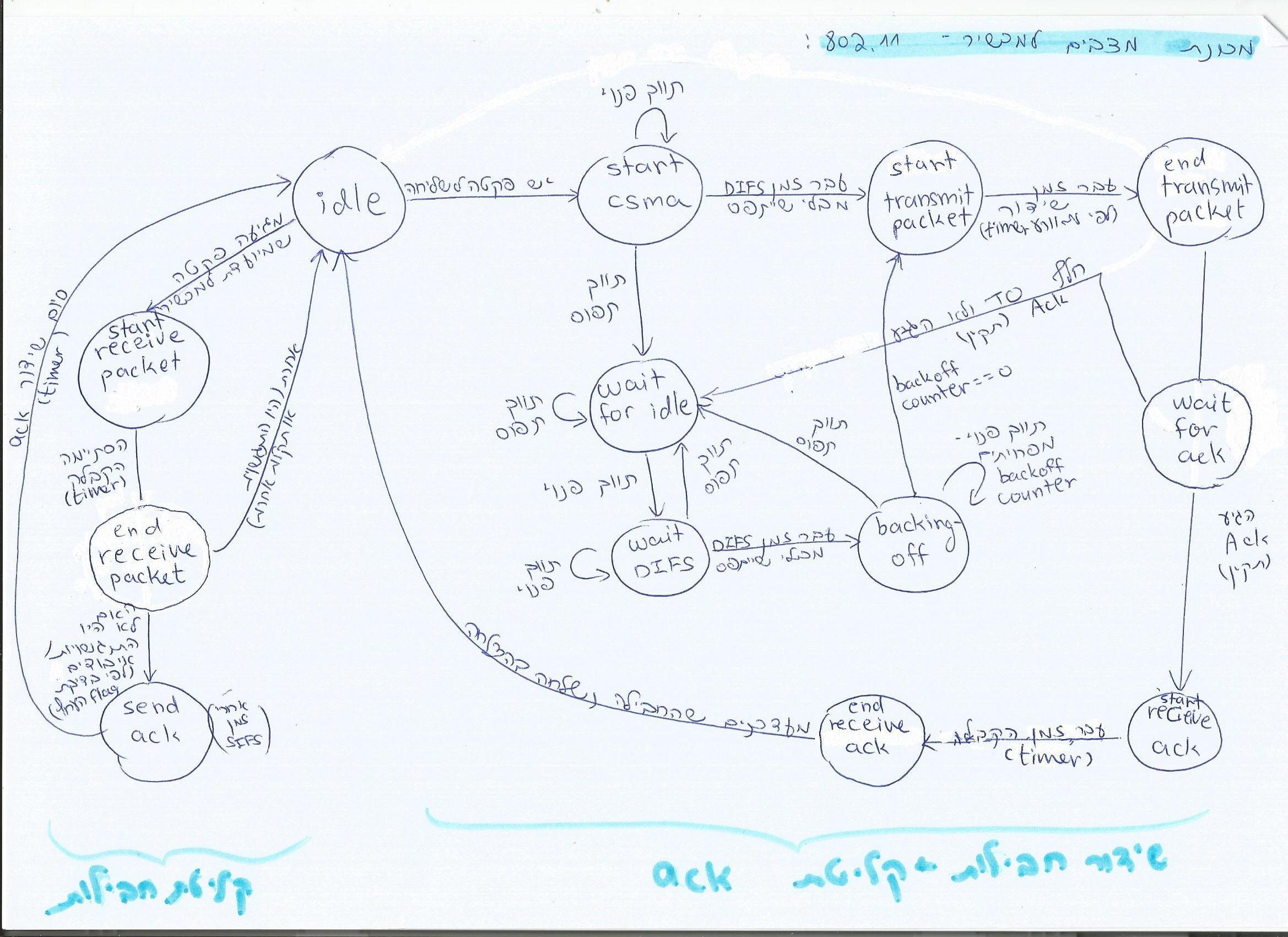
מפרט אירועי המכשיר האפשריים:

|  |  |  |
| --- | --- | --- |
| שם האירוע | הסבר | שינויי מצב מכשיר שהוא גורם ואירועי סימולציה עתידיים שנוצרים |
| PACKET\_EXISTS | הסימולציה מודיעה למכשיר שיש לו מסגרת לשליחה הממתינה בתור | אם המכשיר במצב idle, הוא עובר ל-start csma ומתחיל בניסיון שליחת המסגרת.  אם לא - בפעם הבאה שהמכשיר יחזור למצב idle, יווצר אירוע שיטפל בשליחת המסגרת הבאה בתור. |
| TRAN\_END | הסימולציה מודיעה למכשיר ששידור מסגרת מסוימת שלו הסתיים | תלוי במצב הנוכחי:  אם המכשיר משדר כעת מסגרת, הוא עובר למצב wait for ack.  אם המכשיר משדר כעת ack - הוא עובר למצב idle. |
| REC\_START | הסימולציה מודיעה למכשיר שמגיעה אליו מסגרת (שפרטיה ידועים) שעליו לקבל | אם המסגרת מיועדת למכשיר:  תלוי במצב הנוכחי: אם המכשיר ממתין כעת ל- ack וזו פקטת ack, הוא עובר למצב receive ack ומבטל את ה-timer על המסגרת - באמצעות CLEAR\_TIMER .  אם המכשיר במצב idle וזו פקטת data- הוא עובר למצב receive packet.  אחרת:  מתייחסים לכך כאל הודעת "תווך תפוס": אם למכשיר יש כעת timer פעיל - של backoff או DIFS - הוא מפסיק את הספירה, עובר למצב wait for idle ומייצר אירוע CLEAR\_TIMER לסימולציה, עבור ה-timer שיש לבטל בגלל תפיסת התווך. |
| REC\_END | הסימולציה מודיעה למכשיר שהוא סיים לקבל מסגרת שפרטיה ידועים | אם המסגרת מיועדת למכשיר:  תלוי במצב הנוכחי: אם המכשיר קיבל כעת ack, הוא עובר למצב idle ונוצר אירוע סימולציה CHECK\_QUEUE . אם המכשיר במצב receive packet, הוא מוודא שלא היו התנגשויות/איבוד מידע במסגרת שקלט, ובמידה והכל תקין, מייצר אירוע סימולציה TRANSMIT\_ACK\_START עתידי. אם קרתה התנגשות - עליו "להודיע" לסימולציה והוא מייצר מאורע סימולציה COLL\_INC.  אחרת:  מתייחסים לכך כאל הודעת "תווך פנוי":  אם המכשיר במצב wait for idle - הוא עובר ל-wait DIFS, ונוצר מאורע TIMER חדש מתאים אחרי זמן DIFS. |
| TIMER\_EXPIRED | הסימולציה מודיעה למכשיר שפקע timer כלשהו שהוא ביקש | תלוי במצב הנוכחי:  אם המכשיר ממתין ל-ACK - זהו timeout ולכן הוא עובר ל-wait for idle (מתחיל שוב את תהליך ה-backoff, כדי לעשות retransmit).  אם המכשיר בעל מונה backoff או DIFS פעיל, הוא מעדכן אותו בהתאם ומתרחש שינוי מצב למכונת המצבים . אם המונה שעודכן הוא מונה backoff, והוא התאפס והתווך פנוי (בדיקת שפיות) - נוצר מאורע סימולציה TRANSMITֹ\_PACKET\_START ועוברים למצב transmit packet.  אם המונה הוא מונה DIFS שהתאפס, וגם נמצאים במצב wait DIFS - עוברים ל-backing-off תוך יצירת אירוע TIMER מתאים ל-backoff. אם נמצאים במצב Start csma ומונה DIFS פקע - עוברים ל-start transmit packet ויוצרים מאורע סימולציה מתאים - TRANSMITֹ\_PACKET\_START. |

טבלת מצבי המכשיר האפשריים:

|  |  |  |
| --- | --- | --- |
| מצב | הסבר | מצבים אליהם ניתן לעבור ואירועים שנוצרים |
| IDLE | המכשיר פנוי כעת, לא שולח ולא מקבל. | * אם יש מסגרת לשליחה בתור - אם התווך פנוי,נעבור ל-csma start וניצור מאורע TIMER אחרי DIFS אם התווך תפוס - נעבור ל-wait for idle * אם מגיעה מסגרת המיועדת אלינו - נעבור ל-receive packet . |
| START\_CSMA | תחילת ניסיון שליחת מסגרת באמצעות מנגנון ה-Carrier Sensing. מתחילים לספור לאחור DIFS זמן בעזרת timer. | * אם התווך תפוס - נעבור ל-wait for idle * אם התווך פנוי במשך DIFS זמן (לפי מאורע timer מתאים) - נעבור ל-transmit packet ונייצר מאורע TRAN\_START * כמו כן, נוצר אירוע TRANSMITֹ\_END עם חלוף זמן השידור (המחושב ע"י המכשיר) |
| TRAN\_PACK | המכשיר כעת משדר מסגרת. | * כאשר זמן השידור הסתיים לפי המאורע המתאים- מייצרים מאורע TIMERעתידי בזמן המתאים, עבור ה-ACK timeout * אם מגיע REC\_START - מזוהה התנגשות! |
| WAIT\_FOR\_ACK | המכשיר ממתין ל-ACK על מסגרת ששלח. | * אם מתחיל להגיע ACK - עוברים ל- start receive ack \* * אם חלף Timeout ולא הגיע ACK - עוברים מיידית ל-wait for idle (אם כמות ה-retransmissions טרם חצתה את Num Retries. אחרת - מפסיקים לנסות) |
| REC\_ACK | המכשיר מקבל ACK | * אם מסתיימת קבלת ACK לפי מאורע מתאים - מבצעים בדיקת תקינות (יעד ומקור תואמים, מסגרת מתאימה וכו') , מעדכנים שהמסגרת שודרה בהצלחה במקום המתאים במצב המכשיר במידה וזה אכן כך. אם לא תקין - זורקים את המסגרת. * עוברים ל-idle * אם התווך הופך לעמוס תוך כדי הקליטה - מזוהה התנגשות! |
| WAIT\_FOR\_IDLE | המכשיר ממתין לרגע בו התווך יהיה פנוי (דבר שמאורע סימולציה אמור לאותת לו). | * אם התווך פנוי - עוברים ל-wait DIFS * פותחים טיימר אחרי DIFS זמן |
| WAIT\_DIFS | המכשיר ממתין שהתווך יהיה פנוי במשך DIFS זמן **ברצף**, עך ידי ספירה לאחור בעזרת timer (בדומה למצב start csma). | * אם התווך נתפס - עוברים ל-wait for idle * אם התווך פנוי במשך DIFS זמן (לפי מאורע timer מתאים) - נעבור ל-backing off ונייצר מאורע טיימר מתאים בתום זמן ה-backoff |
| BACKING\_OFF | המכשיר נמצא כעת בתהליך back off. ייתכן שיש לו מונה backoff שכבר אותחל ואז הוקפא, וייתכן שאין לו (ואז יהיה בשדה ה-backoff counter במצב המכשיר ערך מיוחד שמסמן זאת) ואז צריך להגריל random backoff | * אם התווך נתפס - עוברים ל-wait for idle ומעדכנים את מונה ה-backoff בהתאם * אם התווך פנוי ומונה ה-backoff במצב המכשיר מתאפס לפי מאורע timer מתאים - נעבור למצב transmit packet ונייצר מאורע TRANSMITֹ\_PACKET\_START |
| REC\_PACK | המכשיר מתחיל לקבל כעת פקטת מידע (לא ACK) | * אם הקבלה הסתיימה והמסגרת תקינה - עוברים ל-send ack , תוך יצירת מאורע TRAN\_START עבור ה-ACK בזמן המתאים . כמו כן, נוצר אירוע TRANSMITֹ\_END עם חלוף זמן השידור (המחושב ע"י המכשיר) * אחרת (היו תקלות בקליטת המסגרת) - עוברים למצב idle (המסגרת "נזרקת") * אם התווך הופך לעמוס תוך כדי הקליטה - מזוהה התנגשות! |
| SEND\_ACK | המכשיר משדר כעת פקטת ACK על פקטת מידע שקיבל. | * אם השידור הסתיים - עוברים ל-IDLE. * אם מגיע REC\_START - מזוהה התנגשות! |

מכונת מצבים המתארת את הקשרים בין מצבי מכשיר ואירועי מכשיר:



**הנחות והערות**:

* מתעלמים מעוצמות שידור, מניחים שכל התחנות "רואות" את כל השידורים, הם חזקים דיים, ולא קיימת בעיית ה-Hidden Node.
* הסימולציה מייצרת את הפקטות לכל לינק ומודיעה למכשיר השולח שנוצרה מסגרת שעליו לשלוח. המכשיר השולח מתחיל ב-sensing אם הוא פנוי, ואחרת - מכניס את המסגרת לתור.
* צריך לוודא מה קורה אם מכשיר מתחיל לקבל מסגרת תוך כדי שהוא ב-backoff.
* צריך לוודא מה קורה אם מכשיר מקבל ACK על מסגרת לאחר חלוף ה-TO, האם הוא מפסיק לנסות לעשות retransmit או ממשיך?
* אנו מניחים שלא ייתכן שמכשיר יתחיל לקבל מסגרת המיועדת אליו בזמן ההמתנה שלו ל-ACK על מסגרת ששלח, מאחר ואנו עוסקים בלינקים שהם point to point.

# ביבליוגרפיה

[1] S. Ntutumu, J. Reigadas, and R. Patra, “Comparison of MAC protocols for 802.11 based long distance networks,” ResearchGate, 2008.

[2] IEEE Computer Society, “IEEE Standard Association - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE 3 Park Avenue, New York, 2012.

[3] A. Sheth, S. Nedevschi, R. Patra, S. Surana, E. Brewer, and L. Subramanian, “Packet Loss Characterization in WiFi-based Long Distance Networks”, IEEE INFOCOM, 2007.

[4] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, E. M. Belding-Royer, “Understanding Link-Layer Behavior in Highly Congested IEEE 802.11b Wireless Networks”, E-WIND ‘05, 2005.