מסמך תיעוד קוד – אלגוריתם ניתוב מבוסס פרוטוקול RPL:

הערה: מסמך זה מתאר בנקודות את הלוגיקה שמאחורי המימוש ומהווה פירוט על ההערות שקיימות בקוד עצמו.

באלגוריתם ישנם שלושה מודולים מרכזיים:

* hyRPL - מודול שתפקידו זה לבצע המרה מפורמט של הודעות שהאלגוריתם יודע לשלוח ולקבל ל/מ- פורמט של רשימת מספרים לטובת שליחה/קבלה דרך המודם.
* root - מודול שמכיל את הפונקציונליות של שורש העץ המייצג את מבנה הרשת.
* station - מודול שמכיל את הפונקציונליות של תחנה בסיסית ברשת.

בנוסף קיים קובץ נוסף של פרמטרי macro בשם macros.hrl שמכיל את כל הפרמטרים שמוגדרים כ-define ומשותפים לכל המודולים הנ"ל.

המודולים root ו- station תוכננו כמכונות מצבים סופיות ומומשו באמצעות OTP gen-fsm.

ההודעות שמועברות בין תחנות באלגוריתם מועברות באחד או יותר מהתווכים: RF – 1, PLC – 2, שניהם – 3.

מבנה הודעות באלגוריתם הניתוב:

1. הודעת DIO – מבנה הודעה נכנסת:

{1, [SenderNumber, ReceiverNumber, Rank, VersionNumber ,NumberOfHops, SINR, Bitrate, Transmission\_medium]}

1 – סוג ההודעה של הודעת DIO הוא המספר 1.

SenderNumber – אטום – האינדקס של התחנה ממנה נשלחת ההודעה.

ReceiverNumber – אטום – תחנת היעד, בפועל שדה חסר משמעות כי יש לו ערך ברירת מחדל מכיוון שההודעה נשלחת כהודעת broadcast.

Rank – הדרגה של התחנה ממנה נשלחה ההודעה.

VersionNumber – מספר שתפקידו להודיע לתחנה כאשר עליה לאפס את המצב שלה.

NumberOfHops – מספר התחנות מהשורש לתחנה ממנה נשלחה ההודעה.

SINR – פרמטר רעש שמתקבל מהמודם (לא מופיע עבור הודעות יוצאות)

Bitrate – קצב העברת הנתונים ברשת (לא מופיע עבור הודעות יוצאות) – פרמטרים קבועים כתלות בתווך שבו מועברת ההודעה. מוגדר בקובץ macros.hrl.

Transmission\_medium – התווך דרכו התקבלה ההודעה (עבור הודעה נכנסת זה יהיה או 1 או 2 ועבור הודעה יוצאת תמיד יהיה 3).

הערות: 1. כאשר מתקבלת הודעת DIO בתחנה עם VersionNumber נמוך מזה של התחנה, היא מתעלמת מההודעה.

2. הודעה זו מופצת כהודעת broadcast.

1. הודעת DAO – מבנה הודעה:

{2, [SenderNumber,ReceiverNumber,Payload]}

2 – סוג ההודעה של הודעת DAO הוא המספר 2.

SenderNumber – אטום – האינדקס של התחנה ממנה נשלחת ההודעה.

ReceiverNumber – אטום – האינדקס של האב המועדף של התחנה השולחת אליו תועבר ההודעה עד שתגיע אל השורש.

Payload – התוכן של ההודעה – מאיזו תחנת מוצא יצאה ההודעה, מי האב המועדף של תחנת המוצא ומהו התווך האיכותי יותר בין שתי התחנות.

הערה: הודעה זו מופצת כהודעת unicast. בפועל, יתכן ותחנות שההודעה לא מיועדת אליהן יקבלו את ההודעה אבל יתעלמו ממנה.

1. הודעות Data:

הודעות Data הן אינן חלק מאלגוריתם הניתוב אלא תלויות באפליקציה שיכולה להשתמש באלגוריתם הניתוב כתשתית להעברת ה-Data בין התחנות ברשת. מכאן שהאפשרות להעביר הודעות Data בצורה גנרית מאפשרת להתאים את האלגוריתם למגוון אפליקציות שונות. נבחין בין סוגים שונים של הודעות Data שניתן להעביר בהסתמך על הידע באלגוריתם הניתוב:

1. הודעותData מוכוונות מסלול – הודעות שיוצאות מהשורש ויש מסלול מוגדר שנבנה ברמה של תחנת השורש בהסתמך על טבלת ה-source-route כל הדרך עד ליעד.
2. הודעות Data ללא מסלול – הודעות שיוצאות מתחנה כלשהי ברשת והיעד שלהן הוא השורש. במקרה זה כשתחנה תרצה ליזום הודעת Data היא תשלח את זה לאב המועדף שלה ללא מסלול ישיר עד לשורש.

הבסיס של הודעות ה-Data השונות הוא ב-header:

{3, [SenderNumber, ReceiverNumber, Path, Payload]}

3 – סוג ההודעה של הודעת Data הוא המספר 3.  
SenderNumber – אטום – האינדקס של התחנה ממנה נשלחת ההודעה.  
Path – במידה ויוזם ההודעה הוא השורש אז ה-Path הוא רשימת ה-hops ממנו עד ליעד. במידה ויוזם ההודעה הוא תחנת בסיס כלשהי אז ה-Path יוגדר להיות 1-.

ReceiverNumber – אטום – במידה ויוזם ההודעה הוא השורש אז זה האינדקס של התחנה הבאה במסלול אל היעד. במידה ויוזם ההודעה הוא תחנת בסיס כלשהי אז האינדקס של התחנה הוא האינדקס של אב המועדף של אותה תחנה.

Payload – התוכן של ההודעה עצמה – תלוי אפליקציה. חשוב לשים לב שהתוכן יכול להיות מחרוזת (רצף תווי ASCII) או מספרים בלבד כי התוכן חייב להיות מתורגם למספרים על מנת להישלח.

מודול root – האינדקס של התחנה הוא תמיד האטום '1':

המודול root מוגדר באמצעות מצב אחד: active. כלומר מרגע שתחנת השורש נוצרת היא תמיד פעילה. לשורש אין רשימת אבות פוטנציאליים והשורש לא מפיץ הודעות DAO. הדרגה של השורש היא 1 והדרגות של שאר התחנות הן גבוהות יותר יחסית לדרגה של השורש.

לשורש מס' פעולות מרכזיות:

1. הפצת הודעות DIO בהתאם למנגנון תזמון – Trickle (ראה פירוט בהמשך).
2. תחזוק טבלת source-route עדכנית, ETS(mode: set), באמצעות הודעות DAO שהתחנות שלחו לשורש וממנה ניתן לבנות מסלולים לכל אחת מהתחנות שנגישות ממנו.

מבנה כל רשומה ב-ETS: {Station\_Index,Station\_Parent\_Index,Transmission\_medium,TimeStamp} כאשר המפתח לחיפוש ב-ETS הוא ה-Station\_Index – האינדקס של התחנה, Station\_Parent\_Index – האינדקס של תחנת האב המועדף, Transmission\_medium – התווך בין התחנה לאב המועדף שלה, TimeStamp – חותמת זמן שדואגת שהרשומות בטבלה יהיו עדכניות.

1. ניתן להפיץ דרך השורש הודעה שתפקידה ל"זעזע" את הרשת, כלומר לגרום לכל התחנות לאפס את כל הפרמטרים שמגדירים אותן. הודעת "זעזוע" רשת מתבצעת ע"י שליחת event חיצוני – increment\_version\_number שמשמעותו הפצת DIO בהקדם שבה הפרמטר שמכונה Version Number עלה מה שגורם לאיפוס התחנות שמקבלות את ההודעה.
2. יכול להפיץ הודעות data כתלות באפליקציה מסוימת ובהסתמך על טבלת ה-source-route שנבנית כחלק מאלגוריתם הניתוב.

מודול station:

מודול station מוגדר באמצעות ארבעה מצבים: non\_active, active, dio\_block ו- disconnect.

תחנה מוגדרת ע"י מספר פרמטרים: דרגה, רשימת אבות פוטנציאליים (רשימה של אבות שהתחנה קיבלה מהם הודעת DIO והדרגה שלהם נמוכה מדרגת התחנה כאשר בראש הרשימה נמצא האב המועדף של התחנה), המרחק של התחנה מהשורש(מספר ה-hops), Version\_Number.

non\_active – מצב ראשוני של התחנה כשרק מקימים אותה. במצב זה התחנה לא שולחת שום הודעה ולמעשה מחכה להודעת DIO ראשונה כדי "להתעורר". בשלב זה כי מנגנוני התזמון של התחנה עוד לא הוקמו. בהינתן הודעת DIO התחנה עובר למצב dio\_block. אם התקבלו הודעות שהן לא הודעות DIO אז התחנה מתעלמת מהן. ברגע שתחנה עוברת ממצב זה, היא לא חוזרת אליו.

disconnect – מצב שבו תחנה מתנתקת מהרשת. מצב זה עלול להיווצר עקב אי קבלת הודעות DIO במשך זמן רב מהאבות הפוטנציאלים (לכל אב פוטנציאלי יש תוקף ברשימה), כלומר מצב שבו רשימת האבות הפוטנציאלים מתרוקנת או עקב קבלת הודעת DIO מאב פוטנציאלי עם דרגה infinity ובמקרה זה אותו אב פוטנציאלי מוסר מרשימת האבות ובמידה והרשימה החדשה ריקה אז התחנה מתנתקת מהרשת. כדי לצאת ממצב זה(ולעבור למצב dio\_block), התחנה צריכה לקבל הודעת DIO עם VersionNumber גדול או שווה לזה שלה מתחנה שיש לה דרגה סופית.

active – מצב שבו לתחנה יש לפחות אב פוטנציאלי אחד ברשימה והרשת יציבה – כלומר האב המועדף של התחנה נשאר אותו אחד והדרגה שלו יציבה. כאשר האב המועדף משתנה או הדרגה של האב המועדף משתנה אז התחנה עוברת למצב dio\_block. אם רשימת האבות של תחנה מתרוקנת אז התחנה עוברת למצב disconnect.

dio\_block – מצב שנוצר מתוך שאיפה להקטין את ההסתברות להיווצרות של מעגלים ברשת, כלומר מצב שבו תחנה א' היא האב המועדף של תחנה ב' וההיפך בו זמנית. התחנה תעבור למצב זה כאשר יש שינוי בדרגה של התחנה או עקב "זעזוע" של הרשת(ראה פירוט בהמשך). בדרך כלל תימצא במצב זה למשך זמן קצר – עד שיתקבל event מה-trickle שיאמר לתחנה להפיץ הודעת DIO עם הדרגה החדשה שלה. תחנה יכולה לעבור ממצב זה לשני מצבים: active או disconnect. התחנה תעבור למצב active במידה ויש לה לפחות אב פוטנציאלי אחד ברשימה, אחרת תעבור למצב disconnect. במידה והתקבלה הודעת DIO שה-VersionNumber בה גבוה יותר מזה של התחנה אז התחנה תאפס את הפרמטרים שלה ותישאר במצב dio\_block.

לתחנה יש דגל שנקרא DAO\_Flag שתפקידו הוא כאשר הרשת לא יציבה והתחנה מנסה לצאת לשידור ולהפיץ הודעת DIO למנוע הפצה של הודעת DAO עד אשר התחנה הצליחה להפיץ הודעה DIO. רוצים שהרשת קודם כל תתייצב ורק לאחר מכן להפיץ הודעת DAO.

מנגנוני תזמון – באלגוריתם הוחלט על מנגנוני התזמון הבאים (כל מנגנון תזמון בנוי מלפחות תהליך אחד):

1. טיימר שתפקידו להודיע לתחנה בסיסית שבה הוא נמצא מתי להפיץ הודעת DAO.

פונקציה: regular\_timer/3 במודול station כשהפרמטר האחרון הוא סוג ההודעה שמהווה את סוג ה-event שהתחנה הבסיסית תתמודד איתו. במקרה זה סוג ההודעה הוא {timeout, my\_dao}.

1. כאשר תחנה בסיסית מקבלת הודעת dao מתחנה בסיסית אחרת(הודעת unicast) התחנה המקבלת משהה את ההודעה פרק זמן קבוע מראש עד ההעברה שלו לתחנה הבאה בדרך לשורש של העץ. קבוע הזמן של ההשהייה מוגדר בקובץ macros.hrl בשם DAO\_DELAY. השהייה למשך אותו פרק זמן מתבצעת באמצעות טיימר שמודיע לתחנה שפרק זמן ההשהייה של הודעת dao מסוימת שהתקבלה הסתיים וניתן להעביר אותה הלאה.

הטיימר המתאים להודעת DAO כלשהי שולח event לתחנה מהצורה {forwardDAO\_timeout,Desc,TimerPid} כאשר Desc זה הרשומה של צאצא של התחנה ממנו יצאה ההודעה וה-TimerPid זה ה-Pid של הטיימר שנוצר עבור אותו צאצא.

פונקציה: timer\_forward\_dao/2 במודול station.

1. Trickle – אחד ממנגנוני התזמון העיקריים שמוגדרים באלגוריתם ותפקידו להתריע לתחנה מתי היא יכולה לצאת לשידור ולהפיץ הודעת DIO ברשת כתלות בעומס ברשת שנמדד לפי מספר ההודעות שמתקבלות כשהטיימר פועל. מנגנון זה מורכב משני תהליכים: האחד הוא תהליך שתפקידו לספור את כמות ההודעות מסוג DIO וDAO שהתקבלו בתחנה ולקבוע את פרק זמן הבא שהתהליך השני שמשמש כטיימר סטנדרטי יפעל בו. ה-Trickle יכול לקבל את ההודעות הבאות: increment – תעלה את מס' ההודעות שהתקבלו ב-1, timeout – התראה מהטיימר שהזמן עבר ו-inconsistency – שזו הודעה שמסמנת שהרשת לא יציבה ולכן התחנה תרצה לצאת לשידור בהקדם ועל כן פרק הזמן של הטיימר יהיה מינימלי.

ה-Trickle מודיע לתחנה שיש להפיץ הודעת DIO באמצעות שליחת ה-event: timeout\_trickle.  
פונקציות: trickle\_timer/2, trickle/2, trickle/5 במודולים root ו-station.

1. טיימר עדכון רשימת אבות פוטנציאליים – טיימר שתפקידו הוא לפעול לפרק זמן משתנה כתלות באב הפוטנציאלי הותיק ביותר של תחנה מסוימת על מנת להבטיח שכל אב פוטנציאלי לא ישאר ברשימה יותר מפרק זמן שמוגדר מראש בקובץ macros.hrl בשם ConstantDelayParent אלמלא התקבלה הודעת DIO מאותו אב פוטנציאלי.

הטיימר מודיע לתחנה שיש להפיץ הודעת DIO באמצעות שליחת ה-event: refresh\_parents\_list, TimeNow}} כאשר TimeNow זה חותמת זמן שליחת ההודעה מהטיימר.

פונקציה: parents\_update\_timer/2 במודול station.

1. תהליך שתפקידו כל פרק זמן קבוע מראש בקובץ macros.hrl – Refresh\_time\_Ets לעדכן את טבלת ה-source-route שמתוחזקת באמצעות ETS במודול של השורש כדי שהרשומות יהיו תקפות. פונקציה: update\_ets/1 במודול root.

פונקציות עזר:

חלק זה מסביר בקווים כללים את הפונקציונליות של פונקציות העזר המורכבות יותר שהתיעוד שלהן בקוד לוקה בחסר בלבד.

במודול station:

1. sort/2 – הפונקציה מכניסה לתוך רשימה ממוינת של אבות פוטנציאליים לפי הדרגה היחסית של תחנה דרך כל אחד מהם אב פוטנציאלי חדש במקום המתאים תוך שמירה על הסדר היחסי, כלומר אבות פוטנציאלים שהדרגה היחסית של התחנה דרכם זהה יופיעו בהתאם לסדר ההכנסה שלהן לרשימה (מי שנכנס מאוחר יותר לרשימה, יופיע באינדקסים גבוהים יותר בהשוואה לשאר האבות שהדרגה היחסית של התחנה דרכם זהה).
2. objective\_function/4 – פונקציה המחשבת את פונקציית המחיר – מדד לחישוב איכות התקשורת בין תחנות בשני תווכים: RF ו-PLC כפונקציה של שלושה פרמטרים: התווך, קצב העברת המידע ברשת(bitrate) ופרמטר רעש שמתקבל מהמודם(SINR).
3. headMatch/2 – שמקבלת שתי רשימות בודקת אם שני האבות המועדפים בשתי הרשימות זהים בשני מובנים – האינדקס של תחנת האב והתווך בין שתי התחנות(תחנת האב המועדף ותחנת הבן).
4. tailMatch/2 – בהינתן שתי רשימות של אבות פוטנציאליים בודקת האם האבות הותיקים ביותר בשתי הרשימות זהים מבחינת נקודת הזמן האחרונה שבה התקבלה מכל אחד מהם הודעת DIO.

במודול root:

1. calculatePath/3 – הפונקציה מחשבת באמצעות רקורסיית זנב את המסלול מהשורש לתחנת יעד כלשהי. המסלול שהפונקציה מחשבת מתוך טבלת ה-source-route הוא רשימה של רשומות מתחנה לתחנה כאשר התחנה הראשונה ממנה יוצאת ההודעה(השורש) מופיעה אחרונה במסלול ותחנת היעד מופיעה ראשונה – כלל מפתח לפיו כל הטיפול של העברת הודעות data לתחנה בהינתן מסלול עובד.

במודול hyRPL:

1. צמד הפונקציות convert\_path/1 ו- parsePath/1 ממומשות באמצעות רקורסיית זנב ואחראיות להפוך מסלול לרשימה של מספרים לטובת העברת המידע ברשת דרך המודם וההיפך בהתאמה תוך שמירה על האינווריאנטה שתמיד היעד הסופי שאליו ההודעה צריכה להגיע יופיע ראשון במסלול ומוצא יופיע אחרון ברשימה.
2. פונקצייה start/1 שמקבלת אטום מהצורה 'X' כאשר X הוא מספר טבעי כך שאם X הוא 1 אז התחנה המוקמת היא השורש ואם X הוא לא 1 אז התחנה היא תחנת בסיס.