# בעיית השידור ברשתות

## הגדרת הבעיה

בפרק זה נדון בבעיה הבאה: צומת v רוצה לשלוח הודעת שידור (Broadcast) M לכל שאר הצמתים בגרף לא מכוון G. הבעיה היא שהגרף הוא לא בהכרח קליקה וכל צומת יכול לשלוח הודעה רק לשכנים שלו בגרף, אליהם הוא מחובר דרך צלע. לכל צומת יש ID ייחודי וכל צומת מכיר את ה-ID של כל השכנים שלו, כך שכל צומת שמקבל הודעה יודע ממי הוא קיבל. המערכת היא א-סינכרונית, כלומר זוהי מערכת שבה כל ההודעות תמיד מגיעות ליעד (ללא תקלות), אך זמן העיכוב של כל הודעה אינו ידוע ונקבע על ידי יריב. כן ידוע שזמן העיכוב הוא סופי. בנוסף, זוהי מערכת מסוג event-based שבה כאשר צומת מקבל הודעה משכן הוא מבצע חישובים מקומיים ושולח הודעה לשכן אחד או יותר. נרצה למצוא אלגוריתם שידע לבצע שידור במערכת כזו עם סיבוכיות זמן וסיבוכיות שליחת הודעות מינימליים.

## אלגוריתם Flooding

זהו אחד האלגוריתמים המבוזרים הפשוטים ביותר. הרעיון הוא שכל צומת המקבל את M בפעם הראשונה ישלח אותו לכל השכנים שלו מלבד הצומת ממנה קיבל את ההודעה. שלבי האלגוריתם הם:

* צומת v שולח את ההודעה M לכל השכנים שלו.
* כל שאר הצמתים בקבלת M משכן כלשהו w: אם M לא התקבל בעבר שלח את M לכל השכנים חוץ מ-w. אחרת, אל תעשה כלום.

לפי שנחשב את סיבוכיות הזמן וסיבוכיות שליחת ההודעות נגדיר מספר הגדרות על מרחקים בגרף.

### הגדרת מרחקים בגרף

* - זהו המרחק הקצר ביותר בגרף G בין צומת v לצומת u.
* *- זהו המרחק המקסימלי מבין כל הצמתים אל* v*. .*
* *- זהו הרדיוס המינימלי מבין כל צמתי הגרף. .*
* *- זהו הרדיוס המקסימלי מבין כל צמתי הגרף או במילים אחרות המרחק המקסימלי בין כל שני צמתים. .*

*נשים לב שמתקיים: . האי-שוויון הראשון הוא בגלל שנוכל ליצור מסלול בין כל שני צמתים בגרף* v, u *באמצעות המרחק מ-*v *אל צומת הרדיוס ומצומת הרדיוס אל* u*, שהוא לכל היותר פעמיים הרדיוס של* G*. מכל אי-שוויוניים אלו נוכל להסיק שמבחינת סיבוכיות כל גדלים אלו שקולים.*

### סיבוכיות האלגוריתם

לצורך חישוב הסיבוכיות ננרמל את כל הזמנים שבהם היריב יכול לעכב את קבלת ההודעות לערכים בין 0 ל-1, נניח שכל החישובים הפנימיים לוקחים 0 שניות ושההודעה הראשונה נשלחה בזמן 0.

**סיבוכיות זמן** - של אלגוריתם Flooding מצומת v במערכת א-סינכרונית הוא (לא O של אלא בדיוק).

הוכחה: נוכיח בשני כיוונים שהסיבוכיות גדולה שווה ל-וגם קטנה שווה ל-, ולכן שווה לה.

כיוון 1 - עבור צומת כלשהי u עבורה בוודאי ש-u יקבל את ההודעה לפחות בזמן , שכן מניחים כל עיכוב הוא לכל היותר 1. טענה זו בעצם מלמדת אותנו שסיבוכיות זמן במערכת א-סינכרונית תמיד תהיה לפחות הסיבוכיות במערכת סינכרונית.

כיוון 2 - עבור צומת כלשהי u עבורה , בוודאי מתקיים . נוכיח באינדוקציה על t כי הזמן שלוקח ל-u לקבל את M הוא לכל היותר t.

בסיס: עבור נקבל כי ולכן הטענה מתקיימת.

צעד: נניח שהטענה נכונה עבור ונוכיח עבור t+1. יהי מסלול באורך t+1 מ-v אל u. לפי הנחת האינדוקציה, M יגיע אל בזמן לכל היותר t. לכן M יגיע אל u בזמן t+1 או לפני כן. מ.ש.ל.

**סיבוכיות הודעות** - יהי . לפי תיאור האלגוריתם u יכול לשלוח הודעה ל-w לכל היותר פעם אחת וכן בכיוון ההפוך. לכן מספר ההודעות הוא לכל היותר פעמיים מספר הצלעות - .

### עץ פורש

אלגוריתם Flooding יוצר לנו עץ פורש בגרף G. שורש העץ הוא v, ועבור כל צומת אחר - צומת האב שלו הוא הצומת הראשון ממנו קיבל את M. אמנם נשים לב שיכול להיות הבדל בצורת העץ בין מערכת סינכרונית למערכת א-סינכרונית. במערכת סינכרונית העץ הפורש שנוצר על ידי האלגוריתם תמיד ייראה אותו הדבר, ובו הדרגה של כל צומת u היא בדיוק , כלומר המרחק הקצר ביותר ממנו אל v. עץ כזה נקרא “עץ המסלולים הקצרים” או "עץ BFS". אולם במערכת א-סינכרונית כל עץ פורש יכול להיווצר על ידי אלגוריתם Flooding. מי שמחליט איזה עץ ייווצר הוא היריב. המקרה הגרוע ביותר הוא עץ פורש שמחבר את כל הצמתים בענף אחד ועומקו הוא n-1.

## Convergecast

הפעולה ההפוכה משידור (Broadcast) היא Convergecast שבה כל הצמתים שולחים הודעה לצומת v. סיבה אפשרית לרצות לעשות Convergecast היא כדי להודיע ל-v שכולם קיבלו את הודעת השידור שלו.

### אלגוריתם עבור Convergecast:

* לבנות עץ פורש שהשורש שלו הוא v.
* כל עלה u בעץ הפורש: שלח את ההודעה רק לצומת האב שלך.
* כל צומת פנימית w בקבלת הודעה מילד x: אם w קיבל הודעות **מכל** ילדיו שולח הודעה לצומת האב שלו, אחרת ממתין.
* צומת השורש v בקבלת הודעה מילד x: אם v קיבל הודעות **מכל** ילדיו אזי ה- Convergecastמסתיים, אחרת ממתין.

### סיבוכיות האלגוריתם:

**סיבוכיות הזמן** - בדיוק **עומק העץ** הפורש.

**סיבוכיות הודעות** - בדיוק שכן בעץ פורש כל צלע מוסיפה צומת.

## אלגוריתם Flooding/Echo

זהו אלגוריתם שקודם מבצע Broadcast משורש v ולאחר מכן Convergecast על העץ הפורש ש-Flooding יצר. תיאור מפורט של האלגוריתם:

1. הפעל אלגוריתם Flooding כדי לשלוח את M מ-v אל כל הצמתים.
2. Flooding במהלך ריצתו גם מייצר עץ פורש ששורשו v, כמו שתיארנו לעיל, על ידי כך שכל צומת מגדיר את הצומת הראשון ממנו קיבל את M להיות צומת האב בעץ הפורש. בנוסף, כדי שכל צומת ידע מי הבנים שלו, כל צומת המקבל את M במהלך Flooding, מודיע לצומת ממנה קיבל את ההודעה שהוא צומת האב שלו אם זוהי הפעם הראשונה, אחרת מודיע לו שהוא לא צומת האב שלו. צומת שכל הצמתים אליהם שלח את M מודיעים לו שהוא לא צומת האב שלהם, מגדיר את עצמו כעלה בעץ הפורש.
3. הפעל אלגוריתם של Convergecast כדי להחזיר חזרה הודעה מכל הצמתים אל v (זהו שלב ב-echo).

### סיבוכיות האלגוריתם:

**סיבוכיות הזמן** - חיבור הסיבוכיות של Flooding עם Convergecast. במערכת סינכרונית העץ הפורש שנוצר מאלגוריתם Flooding הוא עץ BFS ולכן הסיבוכיות היא . אולם במערכת א-סינכרונית היריב יכול לקבוע במקרה הגרוע שהעץ הפורש יהיה בעומק n-1. לכן הסיבוכיות היא . אלגוריתם זה זוהי דוגמה לכך שסיבוכיות במערכת א-סינכרונית יכול להיות גרוע הרבה יותר מסיבוכיות במערכת סינכרונית.

בסעיפים הבאים נחקור אלגוריתמים נוספים לבניית עץ פורש במערכת מבוזרת, שאמנם הסיבוכיות זמן שלהם יותר גדולה אך העץ הפורש שנבנה מהם יותר מאוזן, ואז אלגוריתם Convergecast ייקח פחות זמן.

**סיבוכיות הודעות** - בכל סוגי המערכות היא .

## אלגוריתם Dijkstra

זהו אלגוריתם שבו כל צומת מתחזק אומדן של המרחק שלו מ-v, בהתחלה כל המרחקים מאותחלים לאינסוף מלבד v המאותחל ל-0. בנוסף האלגוריתם מתחזק תור עדיפויות או ערימת מינימום שבתחילת האלגוריתם מכילה את כל הצמתים. בכל איטרציה מוציאים מהתור את הצומת u שהאומדן שלו הכי נמוך ומעדכנים את האומדן של כל השכנים שלו ממנו. כלומר לכל האומדן יהיה . נרצה להמיר אלגוריתם זה למערכת מבוזרת עם שמירה על המאפיינים הייחודים לו, שהם: בנייה של העץ הפורש רמה אחר רמה ושאין אפשרות לתקן אומדן של צומת לאחר שמוציאים אותו מהתור.

### אלגוריתם Dijkstra מבוזר

באלגוריתם זה בונים את "עץ המסלולים הקצרים" כך שבכל איטרציה r בונים את העץ עד רמה r.

* עבור בלולאה עד שהשורש v מסיים את האלגוריתם:
* צומת v שולח broadcast לכל הצמתים בעץ הפורש שנבנה עד שלב זה. בשידור זה מודיע "התחילו את שלב “r+1.
* השידור מגיע עד לעלים בעץ (הנמצאים ברמה r). עלים אלו שולחים הודעה לכל השכנים שלא קיבלו מהם את ההודעה "התחילו את שלב r+1", ובה הם מודיעים להם "הצטרפו לרמה r+1". הודעה אחרונה זו יכולה להגיע לצמתים ברמה r+1 או צמתים ברמה r.
* כל צומת u שמקבל את ההודעה "הצטרפו לרמה r+1" מצומת w, אם זו הפעם ראשונה (u ברמה r+1) מגדיר את w כצומת האב שלו ומחזיר לו ACK, אם זו לא הפעם הראשונה (u ברמה r) מחזיר לו NACK.
* כאשר עלה מקבל ACK/NACK מכל מי ששלח לו הודעה מתחיל Convergecast עם הודעה "התחילו שלב r+2". אמנם אם קיבל רק NACK מכל מי ששלח לו הודעה, הוא מבין שהוא עלה אמיתי בעץ הפורש הסופי, ומתחיל Convergecast עם ההודעה "סיימתי".
* כאשר v מקבל מכל הבנים את הודעת ה-Convergecast, אם בחלק מהן יש "התחילו שלב r+2" עובר לאיטרציה הבאה. אחרת, כל ההודעות הן "סיימתי", מסיים את האלגוריתם (אפשר להוסיף שמתחיל שידור לכל הצמתים עם הודעה "השידור הסתיים").

### סיבוכיות האלגוריתם

**סיבוכיות הזמן** - בכל איטרציה r מבצעים קודם Flooding בעץ הנוכחי T, שהסיבוכיות שלו היא . ולבסוף מבצעים Convergecast שהסיבוכיות שלו היא עומק העץ, שבמקרה הזה גם כן שווה ל-r. בנוסף, צריך להוסיף שתי יחידות זמן שבהם העלים שולחים לשכנים שלהם "הצטרפו לרמה r+1" ומקבלים חזרה ACK/NACK. סך הכל הסיבוכיות של כל איטרציה r היא . כדי לקבל את סיבוכיות הזמן של האלגוריתם נסכום את סיבוכיות כל האיטרציות, שמספרם לכל היותר , ונקבל **.**

**סיבוכיות הודעות** - נחלק לשני סוגי הודעות:

1. סיבוכיות ההודעות של Flooding ו-Convergecast בכל איטרציה הוא מספר הצלעות בעץ הפורש הנוכחי, שהוא n-1, כאשר n הוא מספר הצמתים בעץ. n הוא לכל היותר בגודל . אם נחשב זאת עבור כל איטרציה נקבל .
2. נספור את מספר ההודעות שהעלים בכל איטרציה r שולחים לשכנים שלהם ואת התגובה חזרה. נשים לב שניתן לחלק כל צלעות הגרף G בהתאם לרמות העץ לפי הודעות אלו, שכן כל ההודעות מרמה r לרמה r או לרמה r+1 ישלחו רק באיטרציה ה-r אך לא באיטרציות אחרות. לכן סכום ההודעות מסוג זה הוא , שכן לכל הודעה יש גם תגובה.

סך הכל, סיבוכיות ההודעות היא .

## אלגוריתם Bellman-Ford

גם באלגוריתם זה כל צומת מתחזק אומדן של המרחק שלו מ-v, בהתחלה כל המרחקים מאותחלים לאינסוף מלבד v המאותחל ל-0. עוברים בלולאה פעמים, כך שבכל איטרציה כל צומת מעדכן את המרחק שלו בהתאם לאומדן של כל השכנים שלו. כלומר האומדן של כל צומת u יהיה . נרצה להמיר אלגוריתם זה למערכת מבוזרת עם שמירה על המאפיינים הייחודים לו.

### אלגוריתם Bellman-Ford מבוזר

גם כאן כל צומת 𝑢 מאחסן מספר שלם המייצג את הניחוש הנוכחי שלו למרחק לשורש 𝑣. בכל פעם שצומת 𝑢 יכול לשפר את , הוא עושה זאת ואז מודיע לכל שכניו.

* אתחול: לכל צומת מאתחלים ולשורש v מאתחלים .
* השורש v שולח "1" לכל השכנים שלו.
* כל עוד האלגוריתם לא נעצר:
* כל צומת u בקבלת הודעה מצומת w עם ערך t המקיים : מעדכן , קובע w להיות צומת האב של u, ושולח הודעה “t+1” לכל השכנים מלבד w.

ידוע כי צריך לכל היותר סיבובים של השלב האחרון כדי שכל האומדנים יהיו מדויקים ונקבל את "עץ המסלולים הקצרים". אמנם כיצד נדע במקרים בהם מספיקים פחות סיבובים? יש לכך פתרון אך לא נלמד עליו בקורס זה.

### סיבוכיות האלגוריתם

**סיבוכיות זמן** - של אלגוריתם Bellman-Ford מבוזר הוא .

הוכחה: נוכיח בשני כיוונים שהסיבוכיות גדולה שווה ל-וגם קטנה שווה ל-, ולכן שווה לה.

כיוון 1 - עבור צומת כלשהי u עבורה בוודאי ש-u יקבל את ההודעה לפחות בזמן , שכן מניחים כל עיכוב הוא לכל היותר 1.

כיוון 2 - עבור צומת כלשהי u עבורה , בוודאי מתקיים . נוכיח באינדוקציה על t כי עד זמן t אכן יתקיים כנדרש.

בסיס: עבור נקבל כי ואכן בזמן 0 מתקיים .

צעד: נניח שהטענה נכונה עבור ונוכיח עבור t+1. יהי מסלול באורך t+1 מ-v אל u. לפי הנחת האינדוקציה, בזמן t יתקיים , ולכן לאחר לא יותר מסיבוב אחד u יקבל את ההודעה t מ- ויעדכן כנדרש.

**סיבוכיות הודעות**: יהי . לפי תיאור האלגוריתם u יכול לעדכן את האומדן שלו לכל היותר פעמים, שכן זהו המרחק הגדול ביותר האפשרי בעץ. לכן u יכול לשלוח הודעה ל-w לכל היותר פעמים. כן הוא גם בכיוון ההפוך מ-w ל-u. טענה זו נכונה לכל צלע בגרף, לכן מספר ההודעות המקסימלי האפשרי הוא **.**

## סיכום

**במערכת סינכרונית:**

* כדאי להשתמש באלגוריתם Flooding עם סיבוכיות זמן , וסיבוכיות הודעות .

**במערכת א-סינכרונית**:

* אלגוריתם Dijkstra מבוזר - סיבוכיות זמן , וסיבוכיות הודעות .
* אלגוריתם Bellman-Ford מבוזר - סיבוכיות זמן , וסיבוכיות הודעות .

נשים לב כי דיאקסטרה עדיף כאשר מתחשבים בעיקר בסיבוכיות הודעות ואילו בלמן פורד עדיף כאשר מתחשבים בעיקר בסיבוכיות זמן. האלגוריתם הטוב ביותר הקיים היום המציג פשרה בין סיבוכיות הזמן לשליחת הודעות הוא אלגוריתם המציג סיבוכיות זמן , וסיבוכיות הודעות . אלגוריתם זה מבוסס על “synchronizers” שהם כלי כללי לתרגום אלגוריתמים סינכרוניים לא-סינכרוניים.