אילן הינדי

[הקלד את תקציר המסמך כאן. התקציר הוא בדרך כלל סיכום קצר של תוכן המסמך. הקלד את תקציר המסמך כאן. התקציר הוא בדרך כלל סיכום קצר של תוכן המסמך.]

אלגוריתמים מבוזרים

פרק 3 : Snapshots

פרק 3 : Snapshots

תוכן עניינים

[1 הגדרות 4](#_Toc506881414)

[2 שימושים 4](#_Toc506881415)

[3 בעיות 4](#_Toc506881416)

[4 תנאים ל-consistent snapshot 5](#_Toc506881417)

[5 אלגוריתם Chandy-Lamport 5](#_Toc506881418)

[5.1 עקרונות 5](#_Toc506881419)

[5.2 Pseudo code 6](#_Toc506881420)

[5.3 דוגמא 7](#_Toc506881421)

[5.4 תוספת למימוש : דווח של ה – snapshot לתהליך היוזם 7](#_Toc506881422)

[5.4.1 כללי 7](#_Toc506881423)

[5.4.2 מבני נתונים והודעות 8](#_Toc506881424)

[5.4.3 מהלך האלגוריתם 8](#_Toc506881425)

[5.4.4 Pseudo code של כל האלגוריתם 10](#_Toc506881426)

[6 אלגוריתם Lai-Yang 13](#_Toc506881427)

[6.1 דרישות 13](#_Toc506881428)

[6.2 הרעיון המרכזי של האלגוריתם 13](#_Toc506881429)

[6.3 בעיות באלגוריתם 13](#_Toc506881430)

[6.4 פתרון לבעיות 13](#_Toc506881431)

[6.5 ניהול רשימת ההודעות 14](#_Toc506881432)

[6.6 Pseudo code 15](#_Toc506881433)

[6.7 דוגמא 17](#_Toc506881434)

[6.8 הוכחה שהאלגוריתם קונסיסטנטי 20](#_Toc506881435)

[6.8.1 תנאי 1 אם a>b ו-b presnapshot אז גם a presnapshot 20](#_Toc506881436)

[6.8.2 תנאי 2 אם הודעה m נמצאת בערוץ ו-a אירוע השליחה שלה ו-b אירוע הקבלה שלה אזי a presnapshot ו-b postsnapshot 20](#_Toc506881437)

[6.9 תוספת למימוש : דווח של ה – snapshot לתהליך היוזם ועבודה בסבבים 21](#_Toc506881438)

[6.9.1 שימוש במספר הסבב במקום הדגל 21](#_Toc506881439)

[6.9.2 דווח לתהליך היוזם 22](#_Toc506881440)

[6.9.3 תוספת נתונים 23](#_Toc506881441)

[6.9.4 שלב צבירת המשקל 23](#_Toc506881442)

[6.9.5 שלב שליחת ההודעות 23](#_Toc506881443)

[6.9.6 סינכרון בין האלגוריתם לאלגוריתם של הדווח 24](#_Toc506881444)

[6.9.7 ניתוח של מהלך האלגוריתם 25](#_Toc506881445)

[6.9.8 Pseudo code של כל האלגוריתם 28](#_Toc506881446)

[7 תשובות לשאלות 32](#_Toc506881447)

[7.1 שאלה 1 – האלגוריתם של Chandy-Lamport 32](#_Toc506881448)

[7.2 שאלה 2 - האלגוריתם של Chandy-Lamport 32](#_Toc506881449)

[7.2.1 עקרונות 32](#_Toc506881450)

[7.2.2 הוכחה שה-snapshot קונסיסטנטי 32](#_Toc506881451)

[7.3 שאלה 3 – האלגוריתם של Lai-Yang 32](#_Toc506881452)

[7.4 שאלה 4 – האלגוריתם של Lai-Yang 33](#_Toc506881453)

[7.4.1 עקרונות 33](#_Toc506881454)

[7.4.2 הוכח שהאלגוריתם קונסיסטנטי 33](#_Toc506881455)

[7.5 שאלה 5 - כללי 34](#_Toc506881456)

[7.5.1 עקרונות 34](#_Toc506881457)

[7.5.2 הוכח שהאלגוריתם קונסיסטנטי 34](#_Toc506881458)

פרק 3 : Snapshots

# הגדרות

Snapshot – מצב המשתנים הלוקליים בכל תהליך וההודעות בערוצים ב-configuration מסוים.

1. Basic message – הודעה של התהליך המרכזי
2. Control message- הודעה של תהליך ה-snapshot
3. Consistent snapshot- snapshot שמתאר קונפיגורציה של התוכנית (כלומר המצב שמתואר ב-snapshot מתואר גם בדיאגרמת המצבים של התוכנית)
4. Inconsistent snapshot – snapshot שמתאר קונפיגורציה שאינה קיימת של התוכנית (כלומר המצב שמתואר ב-snapshot לא מתואר בדיאגרמת המצבים של התוכנית)
5. Presnapshot- אירוע שקרה בתהליך לפני snapshot מקומי
6. Postsnapshot- אירוע שקרה בתהליך אחרי snapshot מקומי

# שימושים

לבחינת תכונות חיצוניות לתהליכים שנשארות נכונות מהרגע שהן נעשות נכונות לדוגמא:

* Deadlock
* Termination
* Garbage
* Restart after failure
* Debugging

# בעיות

נניח שתהליך מעונין לקבל את הסטטוס של הביצוע של אלגוריתם מבוזר הוא צריך לבקש מכל שאר התהליכים לקחת Snapshot מקומי. בנוסף תהליכים צריכים לחשב את המצב בערוצים של הודעות במעבר ברגע הלוגי של ה – snapshot.

הסיבוך הוא שתהליכים לוקחים snapshot מקומי בזמנים שונים לכן snapshot לא בהכרח מציין קונפיגורציה של הביצוע.

1. להבטיח שה-snapshot יהיה של configuration חוקי. כלומר שלא יוצר מצב שבגלל שהחישוב של ה-snapshot מופץ ברשת ולא מתרחש באותו זמן בכל התהליכים לא יקרה מצב שה-snapshots המקומיים לא מרכיבים קונפיגורציה לא קיימת
2. לפתח אלגוריתם שלא יפריע לאלגוריתם שעובד
3. לפתח אלגורית שלא דורש הקפאה של התהליך המרכזי
4. נסתכל על המצב הבא בערוץ PQ:
   1. בתהליך השולח אירוע השליחה postsnapshot כלומר הסדר היה לקיחת snapshot ואח"כ שליחת ההודעה
   2. בתהליך המקבל אירוע הקבלה הוא presnapshot כלומר הסדר היה קודם קבלת ההודעה וא"כ לקיחת snapshot
   3. זה מצב לא הגיוני מאחר ומקבלים סדר כזה : קבלת ההודעה – ביצוע snapshot – שליחת ההודעה שזה סדר לא הגיוני.
5. נסתכל על מצב בערוץ PQ
   1. בתהליך השולח אירוע השליחה הוא presnapshot כלומר הודעה נשלחת לפני לקיחת ה - snapshot
   2. בתהליך המקבל אירוע הקבלה הוא postsnapshot כלומר ההודעה התקבלה אחרי ה-snapshot
   3. זה מצב לא הגיוני מאחר ומקבלים סדר כזה שליחת ההודעה – ביצוע ה – snapshot – קבלת ההודעה (יש להם פה איזה אי הבנה בספר מצב כזה הוא מצב שיש הודעה בערוץ יכול להיות שהם מתכוונים לכך שזה לא חוקי שיש הודעה בערוץ ואז אין קונפיגורציה חוקית ????)

# תנאים ל-consistent snapshot

1. כל אירוע שהוא presnapshot a מקיים שכל אירוע b שמקיים ) b ≺ aכלומר b לפני a ב-casual order) הוא גם presnapshot. תנאי זה מבטיח שהפרש בזמני הדיווח של התהליכים לא יגרום לכך שתהליך אחד "יתקדם" מעבר לתהליך אחר.
2. הודעה נכללת בערוץ ב-snapshot אם ורק אם בתהליך ששולח היא presnapshot ובתהליך המקבל היא postsnapshot

# אלגוריתם Chandy-Lamport

## עקרונות

1. אחת התחנות יוזמת snapshot גלובלי
2. התחנה מבצעת snapshot מקומי ושולחת control message(marker) לכל הערוצים שיוצאים ממנה
3. כל תחנה מבצעת snapshot מקומי כאשר היא מקבלת את ה-marker הראשון. ושולחת הודעות marker לכל השכנים שלה.
4. המצב בערוץ מחושב ע"י התחנה ככל ההודעות שהגיעו לתחנה מהערוץ מאז ה –snapshot שהיא ביצעה כאשר התקבל ה – marker הראשון.
5. כאשר תחנה מקבלת marker היא שולחת לכל הערוצים שיוצאים ממנה את ה-snapshot המקומי בתוספת דווח על ההודעות שהתקבלו בערוץ שממנו הגיע ה-marker מאז ה-snapshot המקומי
6. הערוצים הם FIFO
7. כאשר התהליך מקבל marker מכל הערוצים הנכנסים התהליך מפסיק

## Pseudo code

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Messages | | | | | | | | | | | | | | | | | | | | | | |
|  | (m) | | | | - | | base algorithm message | | | | | | | | | | | | | | |  |
|  | (marker) | | | | - | | snapshot algorithm message | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Variables | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | - | | true if P takes local snapshot | | | | | | | | | | | | | | |  |
|  | makerp[c] | | | | - | | true if the marker message arrived to P through channel c | | | | | | | | | | | | | | |  |
|  | stateP[c] | | | | - | | a stack holding the messages to P arrived from channel c after the local snapshot in P | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Initialize | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | ← | false | | | | | | | | | | | | | | | |  |
|  | makerp[c] | | | | ← | false for all channel c | | | | | | | | | | | | | | | |  |
|  | stateP[c] | | | | ← | ∅ for all channel c | | | | | | | | | | | | | | | |  |
|  | if P wants to initiate snapshot | | | | | | | | | | | | | | | | | | | | |  |
|  |  | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | |  |
|  | end if | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Algorithm | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if P receives a basic message *(m)* through incoming channel *c0*** | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***recordedP*** = true and ***makerp[c]***= false | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***stateP[c]*** ← append(***stateP[c]***, ***(m)***) | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **if P receives *(marker)* through incoming channel *c0*** | | | | | | | | | | | | | | | | | | | | |  |
|  | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | | |  |
|  | ***makerp[c]***← true | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***makerp[c]*** = true for all incoming channels | | | | | | | | | | | | | | | | | | | | |  |
|  |  | terminate | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Procedure TakeSnapshot** | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***recordedP*** = false then | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***recordedP*** ← true | | | | | | | | | | | | | | | | | | | |  |
|  |  | send ***(marker)*** into each outgoing channel of ***P*** | | | | | | | | | | | | | | | | | | | |  |
|  |  | take local snapshot of the state of ***P*** | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## דוגמא



## תוספת למימוש : דווח של ה – snapshot לתהליך היוזם

### כללי

האלגוריתם הבא הוא גרסה של weight throwing algorithm לבדיקה האם כל הרשת סיימה.

1. בגרסה מצב סיום הוא מצב שבו תהליך קיבל (marker) מכל השכנים שלו ולפי האלגוריתם למציאת snapshot הוא עוצר
2. בדומה לתהליך למציאת termination לכל הודעת (marker) יתוסף משקל
3. כאשר התהליך היוזם מתחיל סבב הוא שם את המשקל שלו כ – 1
4. כאשר תהליך מקבל marker הוא מוסיף את המשקל של ההודעה למשקל שלו
5. כאשר תהליך רוצה לשלוח הודעת (marker) הוא מחלק את המשקל שלו לחצי ושולח עם כל הודעה את חצי המשקל שלו מחולק למספר היעדים של ההודעה
6. כאשר תהליך קיבל הודעות (marker) מכל השכנים שלו במקום לעצור הוא שולח הודעת תשובה שהיעד שלה הוא התהליך היוזם
7. כל תהליך שמקבל הודעת תשובה כזו מעביר אותה לכל השכנים שלו רק בתנאי שזו הפעם הראשונה שהוא קיבל הודעת תשובה מהתהליך הנ"ל
8. התהליך היוזם צובר את כל הסכומים שהתקבלו (פעם אחת מכל תהליך) וכאשר הסכום מגיע ל – 1 הוא יודע שהשלב הסתיים

### מבני נתונים והודעות

1. פרמטרים שמוחזקים בכל תהליך לטובת התהליך :
   1. waightP – משקל של התהליך שנצבר כאשר הוא מקבל הודעות (marker)
   2. ReceivedStapshotFrom – רשימה שמתאפסת כאשר מתחיל סבב וכוללת את כל התהליכים שמהם התקבלה הודעת SnapshotReport לסבב הנוכחי
2. פרמטרים שנוספים להודעת (marker) weight:
3. סוג הודעה נוסף (Report) – הודעה שבה מדווח ה – snapshot ההודעה כוללת את השדות הבאים:
   1. IdP : ה – id של התהליך המדווח
   2. snapshotP : ה – snapshot של התהליך המדווח
   3. WeightP : המשקל של התהליך המדווח כאשר הגיעו אליו הודעות (marker) מכל הערוצים

### מהלך האלגוריתם

1. כאשר התהליך היוזם מאתחל סבב בתהליך ה – snapshot הוא מאתחל weightinitiator ← 1
2. כאשר תהליך Pמקבל הודעת (marker, weightm):
   1. אם מדובר ב – (marker) הראשון בסבב (כלומר recordedP = false):
      1. weightP ← weightm
      2. ReceivedReportFrom← ∅
      3. לכל הודעה שנשלחת מוצמד משקל שהוא חצי מהמשקל שעל ההודעה מחולק במספר היעדים
   2. אחרת
      1. weightP ← weightP + weightm
   3. אם התקבלו הודעות מכל הערוצים הנכנסים
      1. הוסף את idP (ה – Id העצמי) לרשימה ReceivedReportFrom
      2. שלח הודעת (Report,IdP, snapshotP, weightP) לכל הערוצים היוצאים
3. כאשר תהליך P מקבל הודעה (Report, IdQ, snapshotQ, weightQ)
   1. אם IdQ נמצא ברשימה ReceivedReportFrom P זורק את ההודעה
   2. אחרת
      1. הוסף את IdQ לרשימה ReceivedReportFrom
      2. אם P אינו התהליך היוזם
         1. שלח את ההודעה בכל הערוצים היוצאים
      3. אחרת (אם P הוא התהליך היוזם)
         1. הצב : weightP ← weightP + weightQ
         2. שמור את snapshotQ
         3. אם weightp = 1 סיים

### Pseudo code של כל האלגוריתם

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Messages | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | (m) | | | | | | | | | | | | | - | | base algorithm message | | | | | | | | |  |
|  | (marker: weight) | | | | | | | | | | | | | - | | snapshot algorithm message | | | | | | | | |  |
|  | (report: id, snapshot, weight, round) | | | | | | | | | | | | | - | | report on snapshot | | | | | | | | |  |
|  |  | | | | | | | | | | | | |  | |  | | | | | | | | |  |
| Variables | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | | | | - | | true if P takes local snapshot | | | | | | | | | | | | | | |  |
|  | weightP | | | | | | | - | | sum of weight of messages | | | | | | | | | | | | | | |  |
|  | roundP | | | | | | | - | | The round which the process is in | | | | | | | | | | | | | | |  |
|  | receivedReportFromP | | | | | | |  | | A list of process ids from which report messages arrived in the current round | | | | | | | | | | | | | | |  |
|  | resultsP | | | | | | |  | | A list in the initiator to collect the results of the network | | | | | | | | | | | | | | |  |
|  | markerp[c] | | | | | | | - | | true if the marker message arrived to P through channel c | | | | | | | | | | | | | | |  |
|  | stateP[c] | | | | | | | - | | a stack holding the messages to P arrived from channel c after the local snapshot in P | | | | | | | | | | | | | | |  |
|  |  | | | | | | |  | |  | | | | | | | | | | | | | | |  |
| Initialize | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | recordedP | | | | | | | ← | | false | | | | | | | | | | | | | | |  |
|  | weightP | | | | | | | ← | | 0 | | | | | | | | | | | | | | |  |
|  | roundP | | | | | | | ← | | 0 | | | | | | | | | | | | | | |  |
|  | receivedReportFromP | | | | | | | ← | | ∅ | | | | | | | | | | | | | | |  |
|  | resultsP | | | | | | | ← | | ∅ | | | | | | | | | | | | | | |  |
|  | makerp[c] | | | | | | | ← | | false for all channel c | | | | | | | | | | | | | | |  |
|  | stateP[c] | | | | | | | ← | | ∅ for all channel c | | | | | | | | | | | | | | |  |
|  | if P wants to initiate snapshot | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | | | | |  |
|  | end if | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  | |  | |  |  |  | |  | |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if *P* receives a basic message *(m)* through incoming channel *c*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***recordedP*** = true and ***makerP[c]***= false | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | ***stateP[c]*** ← append(***stateP[c]***, ***(m)***) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | |  | |  | |  | | |  | | |  | | |  | | |  | | |  | | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |
|  | **if *P* receives *(marker)* through incoming channel *c*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | ***weightP***← ***weightP***+ ***(marker).weight*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | perform procedure TakeSnapshot | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | ***markerp[c]***← true | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***makerp[c]*** = true for all incoming channels then | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | | Perform procedure EndSnapshot | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | |  | |  | |  | | |  | | |  | | |  | | |  | | |  | | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  |  |
|  | **if *P* receives *(report)* through incoming channel *c*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***(report).id*** not in ***receivedReportFromP*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***receivedReportFromP*** ←append( | | | | | | | | | | | | | | | | | | | | | | | | | | ***receivedReportFromP*** | | | | | | | | | | | | | | | | | | |  |
|  |  |  | |  | |  | |  | |  | | |  | | |  | | |  | | |  | | |  | | | ***(report).id*)** | | | | | | | | | | | | | | | | | | |  |
|  |  | | **if** ***P*** is initiator | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | | ***resultsP*** ←append(***resultsP*, *(report).snapshot*)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | |  | | ***weightP***← ***weightP***+ ***(report).weight*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  | |  | |  |
|  |  | |  | | **if *weightP*** = 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | | **if *roundP* < *MaxRouns*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | |  | | perform procedureTakeSnapshot | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | | **else** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | |  | | terminate | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | | **else** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | | send the message to all the neigbors | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | |  |
|  |  | |  | |  | |  | |  | |  | | |  | | |  | | |  | | |  | | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Procedure TakeSnapshot** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **if** ***recordedP*** = false | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | **roundP** ← **roundP + 1** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | take local snapshot of the state of ***P*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***recordedP*** ← true | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***receivedReportFromP*** ← ∅ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | ***weightP***← ***weightP***/ 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | send ***(marker: weightP /*** num of neighbors***)*** | | | | | | | | | | | | | | | | | | | | | | | | | |  | | | | | | | | | | |  |
|  |  |  |  | | into each outgoing channel of ***P*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | **end if** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  | |  | |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  |  |
|  | **Procedure TakeSnapshot** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | if P is initiator | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | resultP ← append (resultP , snapshotP) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | else | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | send (report: idP, snapshotP, weightP, roundP) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  | |  | |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  | |  |
|  | receivedReportFromP ← append(receivedReportFromP , idP) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | makerP[c] ← false for all channel c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  | stateP[c] ← ∅ for all channel c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  |  | |  | |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  | |  |
|  |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |

# אלגוריתם Lai-Yang

## דרישות

1. אין דרישה שהערוצים יהיו fifo
2. כל תהליך יכול ליזום את התחלת התהליך

## הרעיון המרכזי של האלגוריתם

1. הנתונים שתהליך צריך ל – snapshot (בכל אלגוריתם):
   1. את המצב הפנימי שלו
   2. את ההודעות שהיו ממוענות אליו לאחר שהתהליך בצד השני של הערוץ ביצע snapshot ועדין לא הגיעו
2. ע"מ לממש את סעיף 1.2 מאחר ואין סדר בערוץ התהליך השולח צריך להפריד בין הודעות שנשלחו לפני שהוא ביצע snapshot (ושהתהליך המקבל צריך לכלול אותן ב – snapshot המקומי) לבין ההודעות שהתהליך השולח שלח לאחר שהוא ביצע snapshot ושאותן (ואת ההשפעה שלהן) צריך לכלול ב – snapshot הבא
3. ע"מ לבצע את ההפרדה הזאת התהליך השולח מוסיף false להודעות שהוא שלח לפני שהוא ביצע snapshot ו – true להודעות אחרי שהוא ביצע snapshot
4. כל עוד מתקבלות הודעות עם סיומת false התהליך המקבל מתנהג כאילו אין בקשה ל – snapshot במערכת
5. ברגע שהתהליך המקבל מקבל הודעה עם true הוא יודע שהתהליך השולח ביצע snapshot ולכן הוא מבצע snapshot. לאחר שהוא ביצע snapshot הוא מייצר רשימה של כל התהליכים עם סיומת false שהגיעו לאחר ביצוע ה – snapshot ע"מ להוסיף אותן ל – snapshot כהודעות בערוצים בזמן שהיה snapshot
6. כל עוד לא הסתיים השלב ההודעות עם סיומת true נצברות ולא מעובדות
7. כאשר כל ההודעות עם סיומתfalse הממוענות אליו הגיעו התהליך המקבל יודע שהסתיים שלב

## בעיות באלגוריתם

קיימות 2 בעיות באלגוריתם כפי שהוא פורט בסעיף הקודם

1. נניח שלאחר ביצוע ה – snapshot התהליך השולח לא שולח עוד הודעות. במקרה כזה התהליך המקבל לא יבצע snapshot
2. איך התהליך המקבל ידע שהוא סיים את השלב (כלומר שהגיעו אליו כל ההודעות עם סיומת false שנשלחו אליו

## פתרון לבעיות

1. כאשר תהליך מבצע snapshot הוא שולחcontrol message רשימה של כל ההודעות עם סיומת false שהוא שלח בערוץ.
2. כאשר התהליך המקבל מקבל control message הוא מבצע snapshot (אם הוא לא ביצע) ויודע כמה הודעות הוא אמור לקבל בערוץ וכך הוא יודע מתי לסיים (ע"מ לסיים צריכים להתקיים 2 תנאים : הגיעו הודעות control מכל השכנים, כל ההודעות עם סיומת false הגיעו)

## ניהול רשימת ההודעות

1. לפני שהתהליך מקבל snapshot התהליך שומר את כל ההודעות עם סיומת false שהוא קיבל
2. לאחר שהתהליך ביצע snapshot הוא מתחיל רשימה שניה של כל ההודעות עם סיומת false שהגיעו לאחר ביצוע ה – snapshot (הודעות בערוצים). כמו כן הוא מתחיל רשימה של ההודעות עם סיומת true שהוא שומר ע"מ לבצע כאשר מסתיים השלב
3. כאשר התהליך מקבל control message הוא יוצר רשימה שלישית של הודעות שהוא אמור לקבל ע"מ לסיים שלב. התהליך מוריד מרשימה זו את ההודעות שהוא כבר קיבל בסעיפים 1,2
4. כאשר מסתיים השלב התהליך מבצע את על ההודעות שהגיעו עם סיומת true שהוא שמר. את ההודעות שהן התוצר של העיבוד הוא שולח עם סיומת false

## Pseudo code

Messages

***(m)*** - base algorithm message

***(presnp: l)*** – snapshot algorithm control Message. The parameter is the number of messages (including the control message) that where sent from the source process with the tag false

Variables

*recordedP* – **true** if **P** took local snapshot

*sentP[PQ]* - The number of messages P sent to Q before

taking its local snapshot (variable of the

sending process)

*arrivedP[QP]* – The total number of messages arrived with false tag from Q to P

*expectedP[QP]* - The number of messages Q sent to P before

taking its local snapshot (variable of the

receiving process) (This parameter is sent by Q in it's control message)

*stateP[QP]*  – a list of false messages arrived to P from Q

after P took a locl snapshot

Initialize

*recordedP* ← **false**

*sentP[c]* ← **0** for all outgoing channel **c**

*arrivedP[c]* ← **0** for all incomming channel **c**

*expectedP[c]* ← **0** for all incomming channel **c**

*stateP[c]* ← **∅** for all incomming channel **c**

if **P** wants to initiate snapshot then

perform procedure TakeSnapshot

Algorithm

if P sends a basic message ***(m)*** into outgoing channel c0

**send {*m*,** *recordedP***}** into **c0**

**if** *recordedP =* **false**

*sentP[c0]* **←** *sentP[c0] + 1*

**end if**

if P receives ***(m, b)*** through an incomming channel c0

**if *b =* true**

perform procedure TakeSnapshot

**else**

*arrivedP[c0]* ← *arrivedP[c0]* + 1

**if** *recordedP =* **true**

*stateP[c0]* **←** *stateP[c0]* ∪ {***m***}

**if** *arrivedP[c]*+1 = expected*P[c]* for all incomming channels

**terminate**

**end if**

**end if**

if P receives ***(presnp, l)*** through an incomming channel c0

*expectedP[c0]* ← ***l***

perform procedure TakeSnapshot

**if** *arrivedP[c]*+1 = expected*P[c]* for all incomming channels

**terminate**

**end if**

Procedure TakeSnapshot

**if** *recordedP =* **false**

*recordedP ←* **true**

**send (presnp,** *sentP[c]+1)* into all ougoing channels c

take local snapshot of the state of P

**end if**

## דוגמא

נסתכל על רשת עם 2 תהליךים P ו-Q עם 2 ערוצים PQ ו-QP

1. P שולח 2 הודעות {m1, false} ו-{m2, false}



1. P מבצע snapshot מקומי ושולח control message עם הפירוט של ההודעות בסעיף 1



1. P שולח 2 הודעות {m3, true} ו-{m4, true}



1. נניח ש- {m3,true} מגיעה ל-Q ראשונה Q מבצע snapshot מקומי ושולח ל-P control message Q לא מבצע את העיבוד של m3עם פירוט ש 2 הערוצים ריקים



1. נניח שכעת מגיע ה-control message של P Q מסיק מההודעה שההודעות בערוץ הן {m1, false} ו-{m2, false} (כי הן מצוינות ב-control message ועדין לא הגיעו אליו). כאשר מגיע ה – control message ל – P הוא מסיק מזה שהערוץ QP ריק
2. P קיבל הודעות controlמכל השכנים שלו ואינו מצפה להודעות עם סיומת false לכן הוא מסיים את השלב. Q יסיים את השלב כאשר יגיעו ההודעות m1 , m2
3. האלגוריתם מסתיים עם הנתונים הבאים:
   1. Snapshot מקורי של P – (נשמר ב – P)
   2. Snapshot של Q – (נשמר ב – Q)
   3. PQ = {m1, false}, {m2, false} – (נשמר ב – Q)
   4. QP= ∅ - (נשמר ב – P)



1. בגלל ש-m3 ו-m4 הגיעו עם tag true Q מסיק שהן בוצעו אחרי ה-snapshot של P ולכן שייכות ל-שלב הבא

## הוכחה שהאלגוריתם קונסיסטנטי

### תנאי 1 אם a>b ו-b presnapshot אז גם a presnapshot

1. אם a ו-b שני אירועים באותו תהליך ברור שאם b presnapshot אז גם a presnapshot
2. נניח ש-a הוא אירוע שליחה ו-b הוא אירוע קבלה של אותה הודעה. מאחר ו-b presnapshot כלומר נכלל בדווח התוספת שלו צריכה להיות false כי אם התוספת שלו היתה true הוא לא היה נכלל בדווח. מאחר והתוספת של ההודעה היא false הרי שהיא נשלחה לפני ה-snapshot בתחנה השולחת

### תנאי 2 אם הודעה m נמצאת בערוץ ו-a אירוע השליחה שלה ו-b אירוע הקבלה שלה אזי a presnapshot ו-b postsnapshot

1. אם m מדווחת כהודעה בערוץ אזי ברור שאירוע השליחה שלה הוא presnapshot. התחנה המקבלת מדווחת על הודעה בערוץ אם היא עדין לא קיבלה את ההודעה והגיע control message שמציין שההודעה נשלחה. מאחר ומובטח ש-snapshot בתחנה המקבלת מתבצע עם קבלת ה-control message ברור ש-m עדין לא הגיעה לתחנה המקבלת
2. אם a הוא presnapshot זאת אומרת ש-m נשלחה עם סיומת false. כעת יכולות להיות 2 אפשרויות: m הגיעה לתחנת היעד לפני ה-snapshot במקרה כזה לא ידווח ש-m בערוץ. האפשרות השנייה היא ש-m הגיעה לתחנת היעד אחרי ה-snapshot כלומר b postsnapshot במקרה כזה מאחר ו-m עם תוספת false היא מדווחת כהודעה בערוץ.

## תוספת למימוש : דווח של ה – snapshot לתהליך היוזם ועבודה בסבבים

### שימוש במספר הסבב במקום הדגל

כאשר מדברים במימוש של סבבים במקום הפעלה אחת בעצם כל מה שמתואר באלגוריתם עם דגל false הן הודעות ששייכות לסבב הקודם וכל מה שמתואר באלגוריתם עם דגל true הן הודעות ששייכות לסבב הנוכחי.

שינוי נוסף הוא שהאלגוריתם סופר את הודעת ה – control גם כשהיא נשלחת וגם כשהיא מגיעה

כלומר מהלך האלגוריתם יהיה כדלקמן:

#### משתנים

round

count[c] לכל ערוץ נכנס מספר ההודעות בסבב הנוכחי שהגיעו בערוץ c

prevRoundCount[c] לערוץ הנכנס ממנו הגיעה ההודעה שגרמה ל – snapshot. משתנה זה הוא משתנה זמני שנועד לשמור את מספר ההודעות שהגיעו בסבב הקודם במקרה והגורם ל – snapshot הייתה הודעה של האלגוריתם הבסיסי

sent[c] לכל ערוץ יוצא מספר ההודעות שיצאו דרך הערוץ בשלב הנוכחי

expected[c] לכל ערוץ נכנס מספר ההודעות מהסבב הקודם שעדין מצפים להן בשלב זה (התנאי לסיום הסבב הוא ש expected[c] == |State[c]| מכל הערוצים (שונה מהאלגוריתם המקורי)

state[c] רשימת ההודעות מהסבב הקודם שהגיעו בסבב

#### כאשר מבצעים snapshot

1. מעלים את ה – round
2. שולחים control לכל ערוץ עם sent[c]
3. מאפסים את sent[c]
4. שומרים את count[c] של הערוץ ממנו הגיעה ההודעה שגרמה ל – snapshot במשתנה prevRoundCount[c]

#### כשאר מגיעה הודעת control

1. אם ההודעה הגיעה מהאב
   1. expected[c] = m.expected – prevRoundcount[c]
2. אחרת
   1. expected[c] = m.expected – count[c]
3. State[c] = ∅

#### כאשר מגיעה הודעה רגילה

אם מספר הסבב שעל ההודעה הוא כמספר הסבב של התהליך – מעלים את count[c]

אם מספר הסבב של ההודעה קטן ממספר התהליך מוסיפים את ההודעה ל – state[c]

### דווח לתהליך היוזם

1. הדיווח לתהליך היוזם יתבצע ע"י תהליך termination detection בדומה לתוספת לאלגוריתם Chandy – lamport.
2. בעקרון היינו צריכים להצמיד את ההודעה על המשקל להודעות control אבל קיימת בעיה שהתהליכים שולחים הודעות control בשני מקרים:
3. כאשר הם מקבלים הודעת control ראשונה
4. כאשר הם מקבלים הודעה של האלגוריתם הבסיסי הראשונה עם תוספת true

במקרה השני התהליך לא יכול לדעת בזמן שהוא מבצע snapshot ושולח control מה המשקל שיגיע אליו עם הודעת ה – control

הפתרון לבעיה הוא כזה:

אנחנו נעקוב אחר ההתפשטות של ה – snapshot ברשת. אם בהודעה שגרמה לביצוע ה – snapshot אין משקל ישמר האב ממנו הגיעה ההודעה . כאשר תגיע הודעת weight מהאב הנ"ל תשלחנה הודעות weight לכל השכנים.

צריך גם להתמודד עם מצב שבו ההודעה על המשקל עקפה את הודעת ה – control. לשם כך כאשר מתקבלת הודעה על משקל ומספר הסבב של ההודעה גדול ממספר הסבב של התהליך המשקל נשמר.

כאשר מתבצע snapshot לוקחים את המשקל מהערוץ שממנו הגיעה ההודעה מחלקים אותו ל – 2 ושולחים מחצית לשכנים

כאשר תגיע הודעת weight או control לא מהאב המשקל בהודעה ישמר בערוץ

### תוספת נתונים

1. לכל תהליך יהיו 3 ערכים נוספים:
   1. weight[c] לכל ערוץ המשקל שהגיע מהערוץ :
      1. אם הערוץ הוא הערוץ של האב – מחצית מהמשקל שהגיע מהאב
      2. אחרת כל המשקל שהגיע בערוץ
   2. parent האב ממנו הגיעה הודעת ה – control או ההודעה הראשונה מהסבב הנוכחי
   3. לתהליך היוזם : רשימת התהליכים מהם התקבל report בסבב הנוכחי
2. להודעות ה – control יתוסף משקל. למשקל הזה יהיו 2 ערכים אפשריים:
   1. 0
   2. משקל
3. בנוסף תהיה הודעת weight

### שלב צבירת המשקל

1. אם תהליך ביצע snapshot עקב קבלת הודעה של האלגוריתם הבסיסי עם מספר סבב שהוא הסבב הבא או הודעת control עם משקל 0:
   1. אם יש משקל בערוץ הנכנס weight[c] ←
   2. לכל הודעת control בערוץ יוצא מוצמד משקל
   3. נשמר הערוץ שממנו הגיע ההודעה כערוץ אב
2. אם תהליך ביצע snapshot עקב קבלת control עם משקל הוא יבצע :
   1. שמים בערוץ הנכנס weight[c] ←
   2. לכל הודעת control בערוץ יוצא מוצמד משקל
   3. נשמר הערוץ שממנו הגיע ההודעה כערוץ אב
3. כאשר תהליך קיבל הודעת weight :
   1. אם ההודעה היא מהאב :
      1. שמים בערוץ הנכנס weight[c] ←
      2. שליחת הודעות weight לכל הערוצים היוצאים עם
   2. אחרת weight[c] ← m.weight

### שלב שליחת ההודעות

1. כאשר תהליך מסיים (כלומר הוא קיבל הודעות control מכל השכנים ומשקל (באמצעות הודעת control או weight) מכל השכנים וכן הגיעו כל ההודעות מהסבב שלהם הוא מצפה הוא ישלח הודעת report לכל הערוצים היוצאים
2. כאשר תהליך שולח report הוא מאפס את רשימת המשקלים ואת הפרמטר של האב
3. כל תהליך שמקבל הודעת report מעביר אותה הלאה בכל הערוצים היוצאים
4. אם התהליך היוזם קיבל report :
   1. אם ה – report שייך לסבב הקודם הוא מוסיף את המשקל של ההודעה למשקל העצמי שלו ואם המשקל שווה ל – 1 הוא מוציא הודעה ומתחיל סבב חדש
   2. אחרת (המשקל אינו מהסבב הקודם – כלומר מסבבים מוקדמים יותר) הוא זורק את ההודעה

### סינכרון בין האלגוריתם לאלגוריתם של הדווח

#### הפרמטרים של המשקל ושל האב

תנאי הכרחי לשליחת דווח הוא שהתקבל משקל מכל הערוצים

תנאי הכרחי להתחלת snapshot ע"י התהליך היוזם הוא שהתקבל דווח מכל התהליכים

לכן תנאי הכרחי להתחלת snapshot ע"י תהליך שהתקבל משקל מכל הערוצים בסבב הקודם

לכן את המשתנים weight[c] ו – parent[c] ניתן לאפס כאשר שולחים report

#### הפרמטרים של רשימת המעבדים מהם התקבלה הודעת report

כאשר התהליך היוזם מוציא הודעה על ה – snapshot של סבב n כל התהליכים נמצאים בסבב n + 1. לאחר שהוא מוציא את ההודעה הוא עובר לסבב n + 2. כלומר כאשר תהליך נמצא בסבב n + 2 הוא יודע שהוא יכול לזרוק הודעות מסבב n (כלומר 2 סבבים קודם)

אם אנחנו רוצים מנגנון שימנע מהודעות report חוזרות להישלח שוב ע"י מעבד אנחנו צריכים לשמור את רשימת המעבדים שמהם הגיעו הודעות בסבב הנוכחי ובסבב הקודם.

לשם כך יהיו 2 רשימות של מעבדים :

* currentRoundReportSources
* prevRoundReportSources

בנוסף לכך לתהליך היוזם יהיו 2 משתני weight

currentRoundWeight

prevRoundWeight

כאשר מגיעה הודעת report:

1. אם ההודעה היא מסבב round – 2 זרוק את ההודעה
2. אחרת אם ההודעה היא מהסבב הנוכחי – בדוק האם המקור של ההודעה קיים ב – currentRoundReportSources
3. אם ההודעה היא מהסבב הקודם – בדוק האם המור של ההודעה קיים ב –
4. prevRoundReportSources
5. אם התשובה לשאלות היא true זרוק את ההודעה, אחרת :
   1. הוסף את המקור של ההודעה לרשימה המתאימה
   2. אם אתה התהליך היוזם :
      1. הוסף את המשקל לפרמטר המשקל המתאים
      2. אם prevRoundWeight = 1
      3. הוצא הודעה
      4. בצע snapshot

כאשר מבצעים snapshot

1. prevRoundReportSource ← currentRoundReportSources
2. currentRoundReportSources ← ∅
3. אם אתה התהליך היוזם
   1. prevRoundWeight ← currentRoundWeight

#### האלגוריתם של התהליך היוזם

כאשר התהליך היוזם מתחיל סבב הוא מתנהג כאילו הוא קיבל הודעת control לעצמו עם משקל 1.

לכן כל מה שהוא צריך להפעיל זה את הפרוצדורה TakeSnapshot ולשמור את המשקל במשתנה שלו. יש לו שני משתנים כאלה:

המשקל של הסבב הנוכחי

המשקל של הסבב הקודם

כאשר הוא מקבל הודעת control הוא מוסיף את המשקל למשתנה המתאים ובודק האם הוא הגיע ל – 1 ואם כן הוא מוציא הודעה

כאשר הוא מקבל הודעת report הוא מוסיף את הערך בתוצאה למשתנה המתאים ובודק אם הוא הגיע ל – 1 ואם כן הוא מוציא הודעה

כאשר מופעלת הפרוצדורה TakeSnapshot מוחלפים הפרמטרים

### ניתוח של מהלך האלגוריתם





### Pseudo code של כל האלגוריתם

Messages

***(m: round)*** - base algorithm message

***(presnp: expected, weight, round)*** – snapshot algorithm

Control Message

(the control message is

also counted)

***(wight: weight)*** – propagate weight from process that received

It's weight later

***(report: id, snapshot, weight, round)*** - report on snapshot

Variables

*roundP* – The current round

*sentP[PQ]* - The number of messages P sent to Q before

taking its local snapshot (variable of the

sending process)

*counterP[QP]* – The total number of messages arrived with the current round from Q to P

*prevRoundCountP[QP] –* The total number of messages arrived in the prev round from Q to P. This parameter is a temp parameter for the time between the snapshot and the arrive of the presnp.

*expectedP[QP]* – The number of messages expected from channel QP from the prev round (variable of the reciving process)

*stateP[QP]*– a list of messages from the prev round arrived to

*P from Q*

*parentChannelP –* The channel from which the message that

coused the snapshot came

*weightP[c]*- The weight arrived to P through c

*prevRoundReportSources* – A list of the processes that reports

from them sent in the prev round arrived

*currentRoundReportSources* – A list of the processes that

report from them sent in the current round arrived

*receivedReportFromP* - a list of processes which ***(report)*** from them passed throgh P

*prevRoundWeightP –* Only for initiator- The weight of the prev round

*currentRoundWeight*P **–** Only for initiator – The weight of the current round

Initialize

*RoundP* ← **0**

*sentP[c]* ← **0** for all outgoing channel **c**

*countP[c]*  ← **0** for all incomming channel **c**

*expectedP[c]* ← **0** for all incomming channel **c**

*stateP[c]* ← **∅** for all incomming channel **c**

*weightP[c]*← **0** for all incoming channel **c**

*parentChannelP ←* ⊥

*prevRoundReportSourcesP* ←**∅**

*currentRoundReportSourcesP* ←**∅**

*prevRoundWeight* ← **0**

*currentRoundWeight* ← **0**

if **P** wants to initiate snapshot then

perform procedure TakeSnapshot

Algorithm

if P sends a basic message ***(m)*** into outgoing channel ***c0***

**send {*m*,** *roundP***}** into ***c0***

*sentP[****c0****]* **←** *sentP[****c0****] + 1*

if P receives ***(m, round)*** through an incomming channel ***c0***

**if *m.round*** *>**roundP*

perform procedure TakeSnapshot

**if** ***m.round*** *=**roundP*

*countP* ← *countP* + 1

**else**

*stateP[c0]* **←** *stateP[c0]* ∪ {***m***}

**if**|*stateP[c]* + 1| = expected*P[c]* for all in channels **and**

*weightP[c]* > 0 for all in channel

**send** ***(report)*** to all neighbours

**end if**

**end if**

if P receives ***(presnp: expected, weight, round)*** through an incomming channel **c0**

**if *presnp.round*** *> roundP*

perform procedure TakeSnapshot

*expectedP[****c0****]* ← ***presnp.expected*** – *prevRoundCountP[c0]*

**else**

*expectedP[****c0****]* ← ***presnp.expected*** – *countP[c0]*

**if*****presnp.weight*** *> 0*

**if *c0*** = *parentChannelP*

*WeightP[****c0****] ←* ***presnp.weight****/2*

***messageWeight*** = ***message.weight***/(2 \* |out channels|

send weight with ***messageWeight*** to all out channels

**else**

*WeightP[****c0****] ←* ***presnp.weight***

**End if**

**Enf if**

**End if**

*countP[****c0****] ← 0*

**if**|stateP[c0]|+1 = expected*P[c]* for all incomming channels

**and** *weightP[c]* > 0 for all in channel

**send** ***(report)*** to all neighbours

**end if**

Procedure TakeSnapshot after receiving message ***(m)*** with weight ***weight*** through ***c0***

*roundP* **←** *roundP*+ 1

*prevRoundCountP[c0]* ←*countP[c0]*

take local snapshot of the state of P

*ParenrtP =* ***c0***

*messageWeight =* ***weight****/|outgoing channels|*

**for** each out channel c

**send (presnp,** *sentP[c]+1, messageWeight, round)*

*sentP[c] ← 0*

**end for**

*prevRoundReportSourcesP ← currentRoundReportSourcesP*

*currentRoundReportSourcesP ← ∅*

If P receives a ***(weight : weight)*** message through channel **c0**

*WeightP[****c0****] ←* ***weight.weight****/2*

***messageWeight*** = ***weight.weight***/(2 \* |out channels|

send weight with ***messageWeight*** to all out channels

If P recievs ***report(id, snapshot, weight, round)***

**if** *roundP* **> *report.round* + 1**

**exit**

**if** *roundP***= *report.round***

***listOfSources* =** *currentRoundReportSourcesP*

***weightSum* =** *currentRoundWeight***P**

**else**

***listOfSources* =** *prevRoundReportSourcesP*

***weightSum* =** *prevRoundWeight***P**

**if *report.id* in *listOfSources***

**exit**

**else**

***listOfSources* ← *listOfSources ∪ {report.id}***

**if** P **is** initiator

***weightSum ← weightSum + report.weight***

***weightSum ← weightSum + weightP[c]*** for all in channels

**if *weightSum = 1***

report results

*prevRoundWeightP = currentRoundWeightP*

*currentRoundWeightP* ***← 0***

perform **TakeSnapshot** like a ***control*** message with

weight 1 arrived from self

# תשובות לשאלות

## שאלה 1 – האלגוריתם של Chandy-Lamport

תן דוגמא להראות שאלגוריתם Chandy-Lamport נכשל אם הערוץ אינו fifo.

נסתכל בדוגמא המובאת. נניח ש-m1 שנשלחה בערוץ pq מגיעה ל-q לפני ה-marker. במקרה כזה q יבצע את ה-snapshot אחרי קבלת m1. קיבלנו אירוע קבלה שהוא presnapshot כאשר אירוע השליחה הוא postsnapshot וזה סותר את התנאי הראשון.

## שאלה 2 - האלגוריתם של Chandy-Lamport

הצע שינוי לאלגוריתם Chandy-Lamport שבו הודעות יכולות להישמר בתור בתחנה המקבלת ובתיאור מצב הרשת הערוצים כל הזמן ריקים.

### עקרונות

1. כאשר תחנה מקבלת marker ראשון היא מתחילה לאגור את ההודעות הנכנסות
2. כאשר תחנה קיבלה marker מכל הערוצים הנכנסים היא מממשת את כל ההודעות שבתור מבצעת snapshot מקומי ושולחת marker לכל הערוצים היוצאים .

### הוכחה שה-snapshot קונסיסטנטי

#### תנאי 1 אם a>b ו-b presnapshot אז גם a presnapshot

1. אם a ו-b שני אירועים באותו תהליך אזי ברור שהתנאי מתקיים
2. אם a אירוע הקבלה ו-b אירוע השליחה של אותה הודעה ו-b presnapshot אזי ההודעה הגיעה לפני ה-marker הראשון שהגיע לתחנה המקבלת כי אחרת היא הייתה נכללת ב-snapshot . מכאן שההודעה הגיעה גם לפני ה-marker שהתחנה השולחת שלחה ולכן גם השליחה היא presnapshot.

## שאלה 3 – האלגוריתם של Lai-Yang

תן דוגמא שבה אלגוריתם Lai-Yang מחשב snapshot שהוא אינו configuration של ה-execution

ע"מ ש-snapshot יהיה configuration צריך שלא יהיו הודעות בערוצים. ע"מ להשיג מצב כזה צריך שלפני ההודעה הראשונה עם סימול true או ה-control-message יגיעו כל ההודעות עם סימול false.

לכן הדוגמא שניתנה אינה מחשבת קונפיגורציה של המערכת.

## שאלה 4 – האלגוריתם של Lai-Yang

עדכן את אלגוריתם Lai-Yang כך שיתמוך בסדרה של snapshots

### עקרונות

1. במקום לסמל את ההודעות ב-true או false נסמנן במספר ה-snapshot האחרון לפני השליחה. לדוגמא הודעות שנשלחו לפני ה-snapshot הראשון יסומנו כ-0 הודעות שנשלחו לפני ה-snapshot השני יסומנו – 1.
2. כל תחנה שומרת את השלב שבו היא נמצאת.
3. כל תחנה שומרת את ההודעות שהתקבלו מאז ה-snapshot. את ההודעות שנשלחו בשלב אחד שלפני זה שבו היא נמצא מציינת כהודעות בערוץ ואת ההודעות משלבים מאוחרים יותר היא שומרת . את ההודעות עם סימול השלב שלפני הנוכחי היא מבצעת.
4. כאשר תחנה מקבלת control-message היא שולחת בכל הערוצים שיוצאים הודעת control-message עם ה-snapshot שהיא בצעה ועם מצב הערוצים העדכני.
5. תחנה יודעת ששלב הסתיים כאשר היא לא אמורה לקבל הודעות עם השלב שבו היא נמצאת. תחנה יודעת שהיא לא אמורה לקבל הודעות עם השלב שבו היא נמצאת אם ב-control-message שהגיע מהערוץ יש את כל ההודעות שהגיעו בערוץ בשלב אחד לפני זה שבו נמצאת התחנה.
6. כאשר תחנה מגיעה למסקנה שיש להעלות שלב היא מבצעת את כל ההודעות בשלב הנוכחי מבצעת snapshot מקומי, מעלה שלב ושולחת control-message לכל ערוץ יוצא עם פירוט ה-snapshot וההודעות שנשלח בערוץ בזמן ביצוע ההודעות של השלב הקודם.

### הוכח שהאלגוריתם קונסיסטנטי

#### תנאי 1 אם a>b ו-b presnapshot אז גם a presnapshot

1. אם a ו-b אירועים על אותה תחנה ברור שאם b presnapshot אז גם a presnapshot
2. אם b אירוע השליחה ו-a אירוע הקבלה של אותה הודעה ו-b presnapshot משלב מסוים אזי ההודעה מסומנת כשלב שלפני השלב שבו נמצאת התחנה . מאחר ותחנת היעד לא עוברת שלב לפני שהתקבלו כל ההודעות עם אותו שלב מהתחנה השולחת אזי גם a presnapshot.

#### תנאי 2 אם הודעה m נמצאת בערוץ ו-a אירוע השליחה שלה ו-b אירוע הקבלה שלה אזי a presnapshot ו-b postsnapshot

1. אם m מדווחת כהודעה בערוץ ב-snapshot מסוים אזי היא עם בשלב אחד לפני התחנה והגיעה אחרי ה-snapshot לכן אירוע הקבלה הוא postsnapshot. אם היא שלב אחד לפני ה-snapshot זאת אומרת שהיא נשלחה ב-snapshot בתהליך השולח עם שלב קודם. על מנת שהתהליך המקבל יעלה שלב הוא צריך לקבל מהתהליך השולח גם את ההודעות של השלב שלפני שנשלחו בערוץ וגם את ה-control-message המציין שההודעות נשלחו לפני ה-control-message ההודעה נשלחה על ידי התהליך השולח לפני ה-control-message ולכן אירוע השליחה הוא presnapshot.
2. אם קבלת ההודעה היא postsnapshot יש 2 אפשרויות:
   1. ההודעה משלב שלפני השל שבו נמצאת התחנה והגיעה אחרי ה-snapshot ולכן היא מדווחת בערוץ.
   2. אם הודעה מדווחת כהודעה שהתקבלה בערוץ אזי היא משלב לפני השלב שבו נמצאת התחנה ולכן היא נשלחה לפני ה-snapshot של השלב שבו נמצאת התחנה השולחת.

## שאלה 5 - כללי

תכנן אלגוריתם ל-snapshot עבור רשת לא מכוונת (רשת שבה התעבורה היא דו כוונית) והערוצים לא FIFO כאשר: 1. Marker-messages נשלחות עם מספר ההודעות (של התחנה ללא control) עם מספר ההודעות שנשלחו בערוץ לפני ה-marker-message. 2.acknoledgement. 3.אפשרות הקפאה של התהליך שרץ על התחנה.

### עקרונות

1. תחנה יוזמת snapshot ושולחת marker-message עם מספר ההודעות שנשלחו לפני ה-snapshot.
2. התהליך מתחיל כאשר התחנה היוזמת מבצעת snapshot מקומי ושולחת marker-message בכל הערוצים היוצאים עם מספר ההודעות שנשלחו לפני ה-snapshot בערוץ.
3. לאחר ביצוע ה-snapshot התהליך של התחנה מוקפא.
4. כל הודעה שמגיעה מערוץ מאושרת ע"י acknowledgement.
5. לאחר שכל ההודעות שתחנה שולחת אושרו ומספר ההודעות שהגיעו מכל התחנות השכנות שווה לזה שצוין ב-marker-message שהגיע מהתחנה השכנה , התהליך של התחנה מבצע את ההודעות שנשלחו מבצע snapshot ונשלחת marker-message לכל הערוצים עם מספר ההודעות שנשלחו בזמן ביצוע ההודעות של השלב הקודם.

### הוכח שהאלגוריתם קונסיסטנטי

#### תנאי 1 אם a>b ו-b presnapshot אז גם a presnapshot

1. אם a ו-b הם אירועים עוקבים על אותה תחנה אזי התנאי ברור
2. נניח ש-a הוא אירוע שליחה של הודעה ו-b אירוע קבלה של ההודעה. אם b presnapshot אזי ההודעה הגיעה לפני ה-snapshot. מאחר והתחנה השולחת מוקפאת לאחר ה-snapshot לא יכול להיות שההודעה יצאה לאחר ה-snapshot.

#### תנאי 2 אם הודעה m נמצאת בערוץ ו-a אירוע השליחה שלה ו-b אירוע הקבלה שלה אזי a presnapshot ו-b postsnapshot

להלן תיאור סדרת האירועים בתחנות:

1. התחנה השולחת שולחת הודעות לפני ה-snapshot
2. התחנה השולחת שולחת marker-message
3. התחנה המקבלת נמצאת בהקפאה
4. התחנה המקבלת מקבלת חלק (או את כל) ההודעות שלפני ה-snapshot
5. התחנה המקבלת מאשרת את כל ההודעות שהגיעו
6. התחנה המקבלת מפסיקה את ההקפאה כאשר לכל ההודעות שהיא שלחה בערוצים היוצאים הגיע אישור ומספר ההודעות בערוצים הנכנסים שווה למספר שצוין ב-marker-message
7. התחנה המקבלת מבצעת את ההודעות שהגיעו (כלומר את ההודעות שהתחנה השולחת שלחה לפני ה-snapshot).
8. התחנה המקבלת שולחת control-message לכל ערוץ יוצא עם מספר ההודעות שהיא שלחה בערוץ

נניח מדווח שבערוץ יש הודעה מכאן שההודעה הגיעה לתחנה המקבלת אחרי ה-snapshot ולכן b הוא postsnapshot. מאחר והתחנה השולחת מוקפאת לאחר ה-snapshot ההודעה שמדווחת בערוץ נשלחה לפני ה-snapshot ומכאן ש-a הוא presnapshot/

נניח ש-b הוא postsnapshot אזי ההודעה מדווחת בערוץ. ומאחר ו-a הוא presnapshot אזי ההודעה נשלחה לפני ה-snapshot הספציפי ואינה שייכת ל-snapshot עתידי.