

NTP - איך לא לאחר באינטרנט

מאת אלעד קמינסקי ויואב בם

הקדמה

NTP או Network Time Protocol, הינו פרוטוקול אשר עוזר לסנכרן את כל שעוני המחשבים ברשת לשעונים של שרתים בעלי זמן עדכני ומדויק יותר. מדויק לפי מה? קיימים שעונים כגון שעונים אטומיים או שעוני GPS בעלי רמת דיוק גבוהה במדידת זמן ולכן הזמן במחשבים אחרים בנוי עליהם, הפרוטוקול הוא מעין דרך תקשורת ביניהם כדי שכל שעוני המחשב בעולם יהיו מסונכרנים לפיהם.

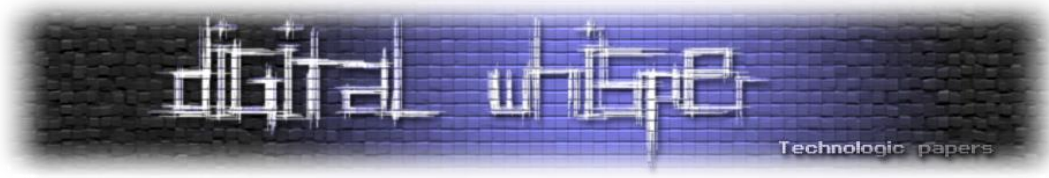
הפרוטוקול NTP פועל בשכבת האפליקציה מעל פרוטוקול UDP תחת פורט 123. הוא נכנס לשימוש עוד לפני שנת 1985 ועדיין בשימוש כיום. דבר זה הופך אותו לאחד הפרוטוקולים הישנים ביותר שעדיין בשימוש. במהלך המאמר נבחן את היסטוריית הפרוטוקול, חשיבותו בעולם מדעי המחשב והשימושים הנפוצים ביותר שלו. בנוסף לכך, נחקור כיצד הוא עובד מאחורי הקלעים, כמו כן נסתכל על בעיות אפשריות של הפרוטוקול וכיצד הוא מתמודד איתן.

זהו לא המאמר היחיד ב-Digital Whisper שמציין את הפרוטוקול הזה. ב-2014 יורי סלובודיאניוק כתב מאמר על התקפות סייבר בישראל בשם "[פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי](#)" שם הוא ציין את הפוטנציאל של התקפת DDoS על שרתי NTP.

שנתיים לאחר מכן, יורי פרסם מאמר על ניהול חומת האש בשם "[איך לא מומלץ לנהל את ה-Firewall שלך](#)". במאמר זה הוא ציין אודות הבעיות של סטיית זמנים בלוגים עקב איש שימוש בשרת NTP.

בשנת 2021 במאמר של ר. בקר וא. חורי, "[מתקפות מניעת שירות מוגברות](#)", הם הציגו שימוש ב-NTP לביצוע Amplification Attacks.

כמו כן מאמר נוסף משנת 2021 שנכתב על-ידי נ. כהן וע. מליאנקר, "[First Step to Tame a Kerberos](#)", הציג את ההסתמכות של פרוטוקול Kerberos על סנכרון זמנים על-ידי NTP.



היסטוריית הפרוטוקול

פרופ' דיוויד ל. מילס הוא דר' למחשבים ומדעי התקשורת (נכון ל-2023, הוא עדיין חי). ב-1977, הוא התחיל לעבוד בחברה COSMAT, החברה אשר המציאה את מה שהיום נחשב האב הקדמון של האינטרנט, ARPANET. התפקיד שלו היה לוודא שכל שעוני המחשבים שברשת יהיו מתואמים. ב-1980 הוא עשה זאת ע"י המצאת אחד מפרוטוקולי התקשורת הישנים ביותר, NTP. לאחר מכן האלגוריתם עוד התפתח למה שהוא היום.

הגרסה הראשונה ביותר של NTP יצא בסביבות 1980 ותועדה ב- IEN-173 וקצת לאחר מכן ב-RFC-778. באותה תקופה, הפרוטוקול נקרא Internet Clock Service (שירות שעון אינטרנט) ושומש לפרוטוקול HELLO, אחד מבין פרוטוקולי הניתוב הישנים ביותר. ב-RFC-958 ה-NTP הוצג לראשונה. המסמך זה הוסבר פורמט ה-packet ואת החישובים הבסיסיים שהפרוטוקול מבצע. באותו זמן, הפרוטוקול לא התמודד עם שגיאות קלות בתדירות העברת המידע. לבסוף, ב-1988, פורסם תיעוד מלא של הפרוטוקול ב-RFC-1059 שבה צורף אפשרות למצב סימטרי וגם למצב של שרת-לקוח.

הגרסה השנייה של NTP הוסיפה אימות מפתח עם סימטרי לפרוטוקול (נסביר בפירוט בפרק "זמן ואבטחה") ותוארה ב-RFC-1119 בערך שנה לאחר הוצאת הגרסה הראשונה. בנוסף לכך, באותו הזמן פותחה תוכנה בשם "XNTP" ע"י דניס פרגסון באוניברסיטת טורונטו. התוכנה השתמשה ב-NTP ואף שפרה באותו בכך שהוסיפה לו דיוק ויציבות משמעותי וטיפול במקרים של זינוק שניה (נפרט יותר בפרק 3). בנוסף לכך, ה"חברה לציוד אלקטרוני" הוציאה פרוטוקול בשם "שירות לתיאום זמן דיגיטלי" (DTSS) שעשה את אותו הדבר כמו NTP.

הגרסה השלישית צירפה את הרעיונות הטובים של NTP ו-DTSS לפרוטוקול יחיד שתואר ב-RFC-1305, ב-1992. גרסה זו הוסיפה תיקון סטטיסטי לשגיאות ויוצאי דופן במדידת קצב התעבורה של הפקטות ברשת. כמו כן, נוסף לפרוטוקול מצב השידור.

מאז הגרסה השלישית, הפרוטוקול שופר באופן רציף. באמצע שנות ה-2000, ניהול הפרוטוקול עבר מידי של מילס לאדם בשם הרלן סטן. ב-2010, פורסם RFC-5905 ובו תוארה הגרסה הרביעית והאחרונה בעת כתיבת המאמר. גרסה זו שיפרה משמעותית את דיוק הסנכרון לרמה של מילי-שניה יחידה של שגיאה בין שעונים המחברים לו. בנוסף לכך, נוסף מצב multicast לפרוטוקול, אופשר אימות עם מפתח ציבורי, אמינות הפרוטוקול שופרה והומעטה תעוברת האינטרנט שהפרוטוקול יצר. מאז ועד היום הפרוטוקול עוד משתפר ומתפתח.

חשיבות הפרוטוקול בעולם מדעי המחשב והשימושים הנפוצים

כפי שצוין בהקדמה, NTP הוא אחד מהפרוטוקולים הישנים ביותר שעדיין בשימוש. זה מכיוון שהוא ממלא צורך מאוד בסיסי ברשת של מחשבים: סנכרון זמן. למה סנכרון בין שעוני המחשבים ברשת הוא חשוב? הסיבות מוצגות להלן:

אימות מצריך תיאום בזמנים

אחת מההתקפות סייבר הנפוצות ביותר באימות משתמשים היא התקפת "שליחה מחדש" (replay attack). בהתקפה זו, התוקף מצליח "להתיישב" על קו התקשורת ולפענח את ההודעות בין הלקוח לשרת האימות. כאשר לא מופעלים מנגנונים נגד התקפה זו, התוקף מסוגל לעכב, למחוק ולשלוח מחדש הודעות ובכך לפרוץ את פרוטוקול האימות.

אחד מהמנגנונים היעילים ביותר בלוחמה במתקפות מסוג זה הוא טביעות זמן (timestamps). הרעיון הוא שאם עבר יותר מדי זמן בין השליחה לקבלה של הודעות, אז ככל הנראה מישהו מפריע להודעה לעבור כמו בהתקפת "שליחה מחדש". בשביל לדעת כמה זמן עבר בין השליחה לקבלה, פשוט השולח מוסיף להודעה את הזמן הנוכחי והמקבל מוודא שהזמן הוא בתווך הגיוני (לרוב כמה מילי-שניות בודדות) מהזמן הנוכחי.

דוגמה טובה לכך היא הפרוטוקול Kerberos. בפרוטוקול זה, שרת צד שלישי מאמת את זהותו של הלקוח ושולח לו כרטיס אשר בעזרתו השרת יכול לדעת שהלקוח מאומת. לכל כרטיס כזה יש זמן תפוגה שאחריו הוא לא מתקבל ע"י השרת. למידע נוסף על הפרוטוקול, אפשר לקרוא את המאמר "1-st Step to Tame a Kerberos: Know Your Enemy" שצוין בהקדמה.

בעיה אחת שיכולה לצוץ משיטה זו היא שכאשר שעוני המחשבים לא מתואמים. במצב כזה, השיטה כולה נופלת וכמעט אף הודעה לא תתקבל. לכן, זה קריטי שהשעונים בין המחשבים היא מכוונים בדיוק רב.

רישום בלי זמנים הוא לא רישום

אחד מהאספקטים הכי חשובים בתכנות הוא רישום (logging). באופן כללי, כאשר תכנית רצה היא לרוב עושה רישום של איזה תהליכים היא סיימה, איזה ערכים היא חישבה וכדומה. רישום הוא חשוב מכיוון שהוא משפר ברמה משמעותית את היכולת לתקן בעיות בתוכנה, לשפר את ביצועיה ולוודא התמדה בנכונות התוכנה. בנוסף לכך, רישום יכול לעזור לזהות אירועי אבטחה במערכת ההפעלה או אפליקציה כלשהי אחרת. הודעות ברישום תמיד חייבות להגיע עם טביעת זמן ויש לכך כמה סיבות. הסיבה הראשונה אופטימיזציה.

כאשר בין תהליכים בתוכנה יש טביעת זמן ממנה ניתן להבין יותר טוב מה לוקח יותר זמן ומה אפשר להשאיר כמו שהוא בתוכנה.



סיבה נוספת היא ניתוח ביחס לעולם החיצוני. כאשר תוכנה כמו אתר אינטרנט, משחק ווידאו או פשוט שרת ענן קורסת, חשוב להבין את הגורם לכך והוא לרוב חיצוני. הבנה של הזמן המדויק של הקריסה מסוגלת להביא בדיוק את המידע הזה.

כמובן שכדי שהזמן ברישום יהיה מדויק, בייחוד בסביבה רשתית כאשר היא רצה על שרת נפרד, שעוני כל המחשבים אשר לוקחים חלק בהרצת או פיתוח התוכנה צריכים להיות מתואמים.

בשביל לעבוד יחד, צריך לדעת את השעה

כפי שצוין בפרק על היסטורית הפרוטוקול, NTP פותח ב-1980 מה שעושה אותו לאחד מהפרוטוקולים הישנים ביותר באינטרנט. למה היה צריך כל-כך מוקדם פרוטוקול כזה? מכיוון שסנכרון זמן עזר לתהליך הראשון שבשבילו נוצרו רשתות מחשבים, מחשוב משותף. למחשבים באותה תקופה לא היה מספיק כוח מחשבים כדי לחשב דברים ממש מסובכים ולכן נוצרו מערכות המקשרות מחשבים. הבעיה ש-NTP בא לפתור הייתה שרשת המחשבים הסתמכה על זה שכל שעוני המחשבים צריכים להיות מכוונים.

בנוסף לעבודה בין מחשבים, תיאום זמן הוא גם חשוב בעבודה בין אנשים. אנו לוקחים כמובן מעליו הרבה מהדברים שכתוב עליהם שעה מסוימת; אך חשוב להבין שבעבר, רישום השעה באופן מהימן התאפשר אך ורק בזכות הפרוטוקול NTP. על קבצים, מיילים, פוסטים ברשתות החברתיות, הודעות מידיעות, commit-ים בשירותי קוד פתוח ועוד הרבה. רישומים אלה הם הכרחיים לעבודה יעילה ומסודרת.

כמובן, רשתות מחשבים לא רק מתקשרות בתוך עצמם, אלה גם מציעות שירות ללקוחות. דבר זה גרם לתקנים לחול על שירותים אלו שגרמו להם, בין השאר, להשאיר טביעת זמן מדויקת על מה שהם מספקים. דוגמה טובה לכך היא שעוד בשנות ה-60, העברות כסף גדולות צריכות היו להתלוות ברישום מדויק של הזמן שבו הם עברו מסיבות חשבוניות ומשפטיות.

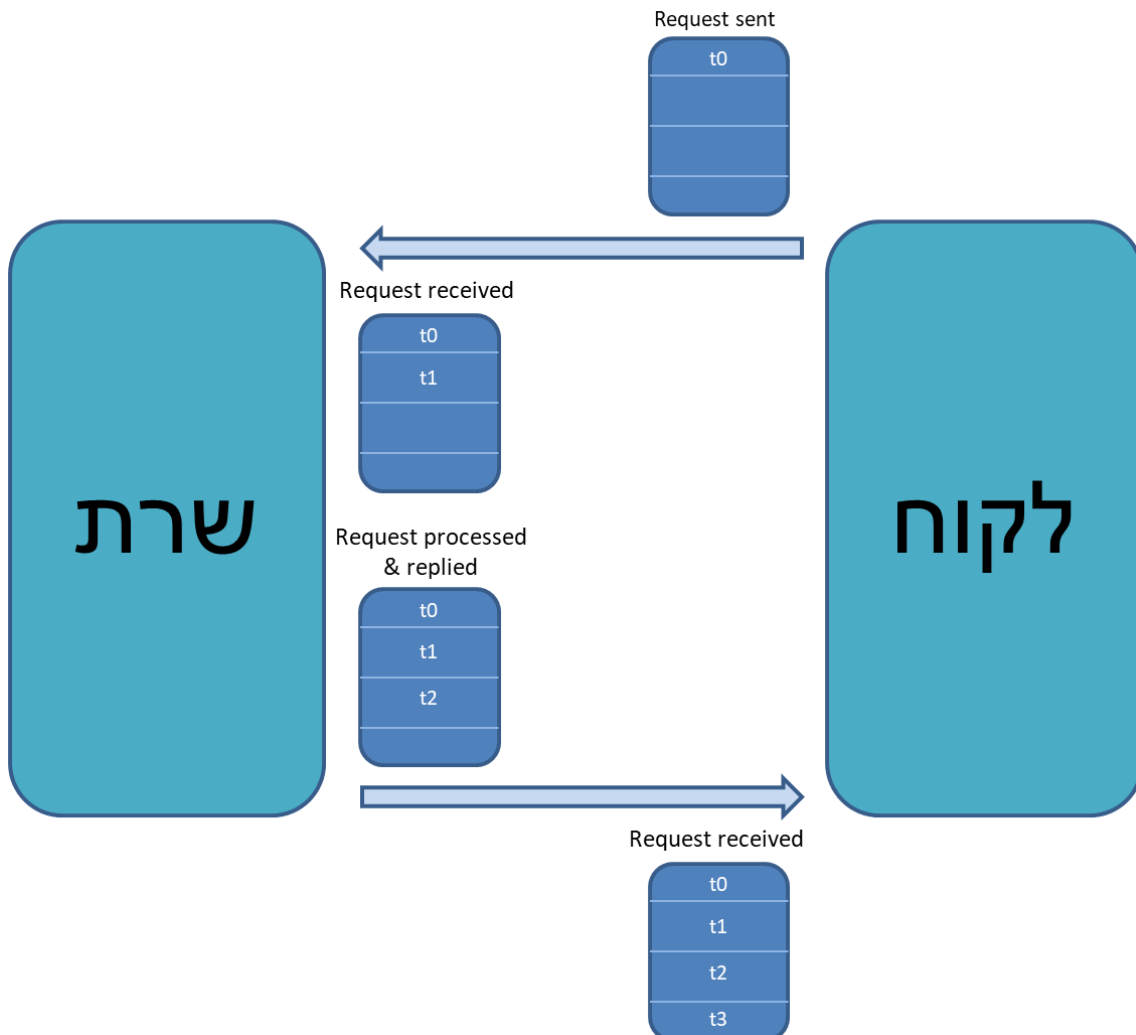
כיצד הפרוטוקול עובד מאחורי הקלעים

אלגוריתם הסנכרון

בפשטות, הפרוטוקול בנוי מ-4 שלבים:

1. הלקוח שולח לשרת חבילת NTP לבקשת תיאום זמן. בחבילה נכלל זמן הבקשה לפי השעון של הלקוח המקומי.
2. השרת מקבל את החבילה ומקליט את זמן הקבלה לפי השעון המקומי שלו.
3. השרת מעבד את החבילה ושולח תגובה ומכליל את זמן הקבלה שהוקלט וזמן השליחה המקומי.
4. הלקוח מקבל את החבילה ומקליט את זמן הקבלה לפי הזמן המקומי שלו.

כך זה נראה:

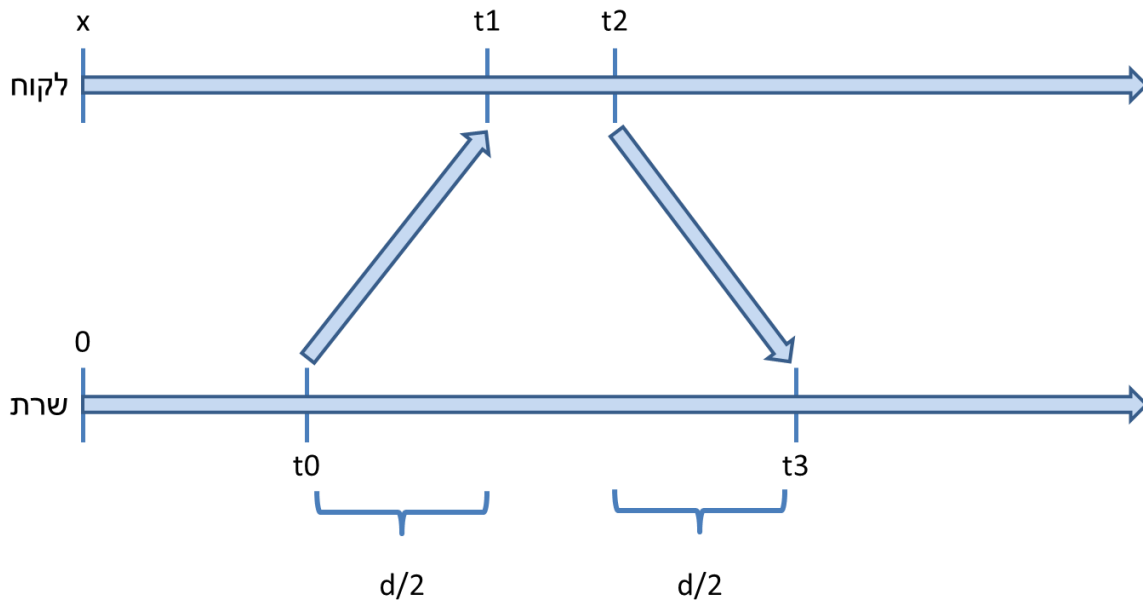


לאחר שהלקוח מקבל את כל המידע הזה הוא מריץ חישובים על מנת לחשב מספר דברים. הלקוח מחשב את הפרשי הזמנים בין השעונים ואת הזמן שלקח לחבילה לעשות את כל המעברים מינוס זמן העיבוד שלה בצד השרת.

כיצד מתבצע החישוב? ראשית, כמה סימונים:

- t_0 - זמן שליחת החבילה הראשונה לפי הלקוח
- t_1 - זמן קבלת החבילה לפי השרת
- t_2 - זמן שליחת חבילת התגובה לפי השרת
- t_3 - זמן קבלת חבילת התגובה לפי הלקוח
- d - הזמן שלוקח לחבילה מסע הלך-חזור בין השרת ללקוח ללא זמן העיבוד ע"י השרת
- x - ההפרש בין שעות הלקוח לשעות השרת

(כל המשתנים הם במילי-שניות)



(שרטוט של צירי הזמן עם המשתנים המוגדרים)

נשים לב כי בין הזמן שהחבילה יוצאת מהלקוח ועד שהיא חוזרת עליו עוברים $t_3 - t_0$ מילי-שניות, ולשרת לוקח $t_2 - t_1$ מילי-שניות לעבד את החבילה. לכן, מתקיים: $d = (t_3 - t_0) - (t_2 - t_1)$. אם נעשה את ההנחה שתקשורת לוקחת אותו זמן בכל כיוון, הזמן שלוקח לחבילה להגיע מהלקוח לשרת ומהשרת ללקוח הוא $2/d$ מילי-שניות ולכן מתקיימים:

$$t_0 + \frac{d}{2} + x = t_1$$

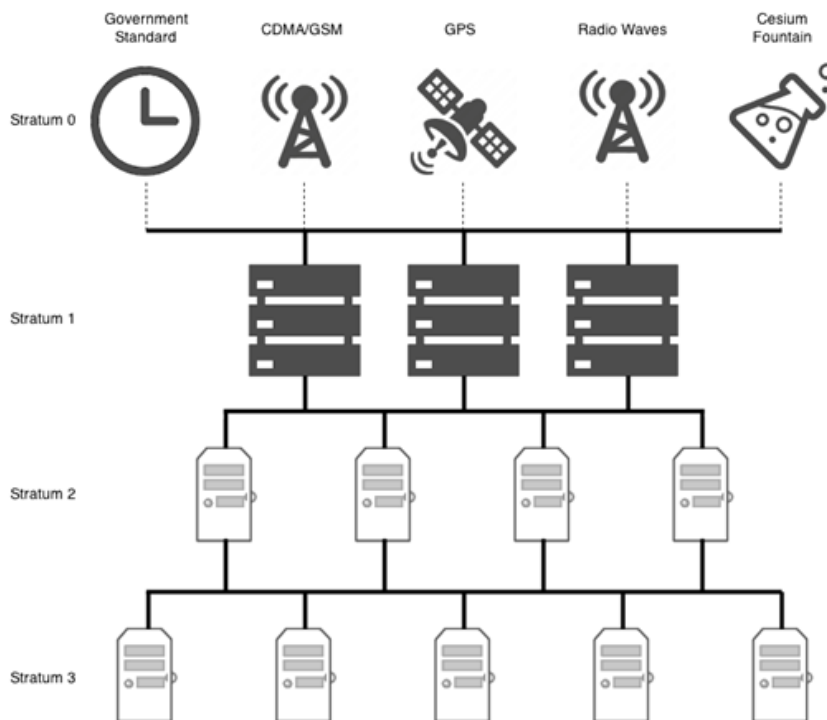
$$t_2 + \frac{d}{2} - x = t_3$$

התהליך הנ"ל חוזר מספר פעמים כדי למנוע חישובים הנובעים מנתונים יוצאים מן הכלל (הקונבנציה היא לפחות 6 פעמים ב-10 דקות הראשונות של החיבור לרשת). ובכל פעם מוצאים את d, x , עושים להם ניתוח סטטיסטי בו מוצאים את ההודעות בהם התקשורת לקחה כמות זמן יוצאת דופן ומוסיפים לשעון את ה- x הממוצע. חשוב לציין, אם ערכי החזרה מפוזרים ואינם עקביים אפילו לאחר הניתוח הסטטיסטי, הסנכרון ייעשה שוב פעם כנגד שרת אחר כדי לוודא את תוצאות הסנכרון המקורי.

תעודף שרתים והיררכיית סטרטום

פרוטוקול NTP משתמש בדירוג היררכי של מקורות הזמן השונים. כל שכבה בהיררכיה מכונה סטרטום (stratum) ומצורף לה מספר אשר מייצג את הדרגה שלה, הדרגה הראשונה הינה 0 ומשומשת לשעונים המדויקים ביותר שזמן נמדד ביחס אליהם ולכן נקראים שעוני התייחסות (reference clocks).

שרת אשר מסונכרן כנגד שרת אחר בעל סטרטום n יהיה מדרגה סטרטום n+1. כלומר המספר המצורף לשרת מייצג את מרחקו משעוני ההתייחסות ונועד למנוע התייחסות מעגלית, כלומר כאשר שעון B מסתנכרן בעזרת שעון A שמסתנכרן בעזרת שעון C שמסתנכרן בעזרת שעון B ואז נוצר מעגל ביניהם כלומר שעונים אלה יהיו מנותקים מהרשת העולמית ולא יהיו מסונכרנים לפי הזמן המקובל בעולם. כדי למנוע מקרה כזה, שרת מסתנכרן רק עם סטרטום נמוך ממנו ומעדכן את הסטרטום שלו להיות גבוהה ב-1 מזה אליו הוא מסונכרן. מחשבים בעלי סטרטום 0 (נקראים גם "שעוני ייחוס") הם החשובים ביותר מכיוון שהזמן ברשת נתמך על גבם, לכן מחשבים כאלה הם השרתים עם השעונים המדויקים ביותר כגון: שעונים אטומיים, שעוני לוויין או שעוני רדיו אחרים, או שעונים מסונכרנים בעזרת פרוטוקול PTP (פרוטוקול סנכרון אחר מדויק יותר).



[שרטוט של מבנה הסטרטום מתוך Baptiste Dauphin's doc]

בנוסף לכך ששיטה זו מונעת מעגלים, היא גם מאפשרת שרשת כיוון ליניארית אלה עץ. דבר זה משפר מאוד את היעילות והעמידות של הרשת שכן אם מחשב אחד נופל זה לא מפיל את כל רשת ה-NTP וגם הפצה של הזמן היא מאוד מהירה.

זינוק שניה

כשלומדים על פרוטוקול הסנכרון זמן, צריך לשאול את עצמנו: איך מודדים זמן?

שעה מוגדרת כחלקי 24 של יממה, דקה מוגדרת כחלקי 60 של שעה ושניה מוגדרת כחלקי 60 של דקה. אלו ההגדרות התיאוריות של יחידות הזמן. בחיים האמתיים, בזמן שאנחנו מודדים שניה כפרק זמן קבוע, אורך יממה הוא דבר שמשתנה בעקבות תופעות אסטרונומיות חיצוניות או געשיות פנימיות. דבר זה גרם להמצאה של זינוק שניה. שעונים תמיד סופרים את הזמן כמו בהגדרות התאורטית עד שאסטרונומים מבינים שכל השעונים מקדימים בשניה ואז צריך שכל השעונים יוסיפו שנייה ליממה. דבר זה נעשה בכך שהשעון מציג:

23:59:58

23:58:59

23:59:60

00:00:00

אחד מהתפקידים של NTP הוא להפיץ את הידע על זינוק שניה.

לכל שעון ייחוס יש קובץ המציין באיזה ימים יש זינוק שנייה או שהשעון עליו הוא מחובר מודיע שיש זינוק שנייה באותו היום. מחשב המקבל הודעה של זינוק שנייה מעדכן את מערכת ההפעלה שלו ומעביר אותה לכל המחשבים שהוא "מכיר" בסטרטום שמתחתיו. כך, באמצעות NTP, רשתות ענקיות של מחשבים יכולות להתעדכן על זינוק שנייה במהירות לוגריתמית.

מצבי הפרוטוקול

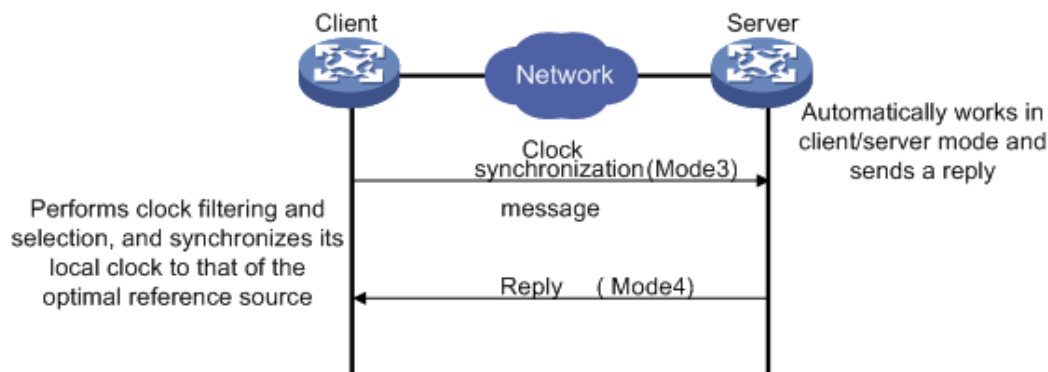
מכשירים שמשתמשים ב-NTP יכולים ליישם סנכרון שעונים ב-4 מצבים שונים:

1. מצב לקוח/שרת
2. מצב עמיתים (peers)
3. מצב תשדיר (broadcast)
4. מצב רב-נתיב (multicast)

מצב לקוח/שרת: מצב זה הוא המצב המרכזי שדרכו לקוחות מסנכרנים את הזמן עם שרתים.

לקוח שולח הודעת סנכרון שעונים לשרתים כאשר שדה המצב מאותחל ל-3 (מצב לקוח), השרת שמקבל את ההודעה אוטומטית עובד במצב שרת ושולח תגובה עם שדה מצב עם הערך 4 (מצב שרת). כאשר הלקוח מקבל את התגובה מהשרתים הוא מפעיל את תהליך הסינון ובחירה שראינו ומסונכרן את שעונו עם השרת האופטימלי.

במצב זה לקוח יכול להסתנכרן עם שרת אבל לא להיפך. כפי שניתן לראות בתרשים הבא:

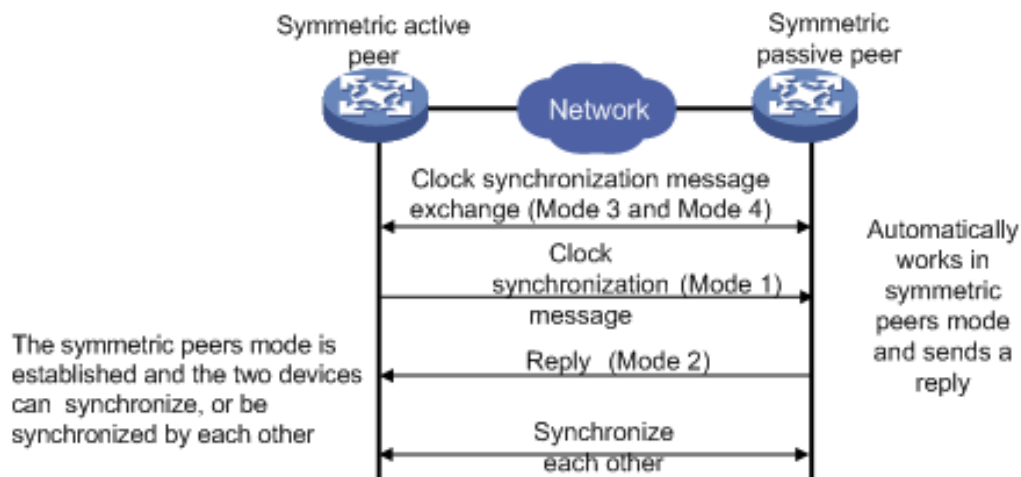


[מקור: https://techhub.hpe.com/eginfolib/networking/docs/switches/5120si/cg/5998-8503_nmm_cg/content/436051971.htm]

מצב עמיתים: הרעיון מאחורי מצב עמיתים הוא שעמית פעיל ועמית סביל יכולים להסתנכרן אחד עם השני, אם יש הבדל ב-stratums העמית עם ה-stratum הגבוה יותר יסתנכרן עם זה עם ה-stratum הנמוך יותר.

השימוש העיקרי של מצב עמיתים הוא לסנכרן בין מכשירים בעלי אותו stratum כגיבוי אחד לשני כאשר השרתים היותר מדויקים לא מתפקדים. אם מכשיר אחד נכלא למצב בו הוא לא יכול לתקשר עם מכשירים בעלי stratum נמוך יותר, הוא עדיין יכול להסתנכרן עם שרת מסונכרן בעל stratum שווה.

כיצד מצב זה עובד? המכשיר שפועל במצב עמיתים פעיל שולח הודעת סנכרון שעונים באינטרוולים עם שדה מצב 1 (עמית פעיל); המכשיר שמקבל את ההודעות אוטומטית מתחיל לעבוד במצב עמיתים סביל ושולח תגובה עם שדה מצב 2 (עמית סביל). בכך, הצד הפעיל מסונכרן מהסביל אלא אם כן הפעיל הוא עם סטרטום נמוך יותר ואז הסביל מסתנכרן לפיו. כפי שניתן לראות בתרשים:

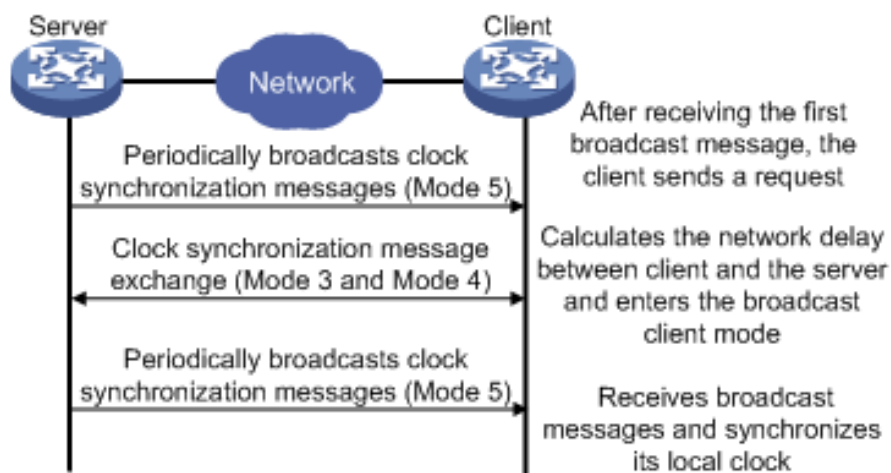


[מקור: https://techhub.hpe.com/eginfolib/networking/docs/switches/5120si/cg/5998-8503_nmm_cg/content/436051971.htm]

מצב תשדיר: לקוח תשדיר יכול להסתגל לשרת תשדיר אבל לא להיפך.

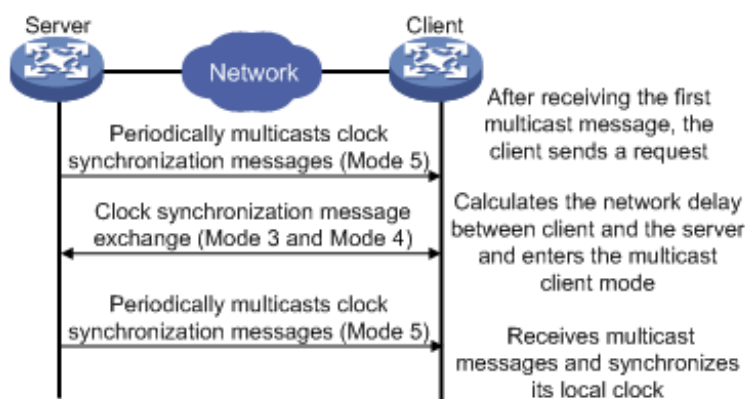
המטרה של מצב תשדיר היא עבור קונפיגורציות בהן יש מעט שרתים והרבה לקוחות ואז כל הלקוחות צריכים להסתגל למעט השרתים. סנכרון לפי מצב תשדיר הוא פחות מדויק בגלל שלקוח מסתגל למעט שרתים ולכן תהליך הסינכרון פחות טוב.

כיצד מצב זה עובד? שרת שולח הודעות סנכרון שעונים לכתובת 255.255.255.255 באינטרוולים עם שדה מצב 5, לקוחות מקשיבים להודעות תשדיר שמגיעות וברגע שהם מקבלים הודעה הם מסתגרים לפי מצב לקוח/שרת לשרת ששלח את ההודעה וחוזרים להקשיב להודעות תשדיר והשרת חוזר למצב תשדיר. כפי שניתן לראות בתרשים:



[מקור: https://techhub.hp.com/eginfolib/networking/docs/switches/5120si/cg/5998-8503_nmm_cg/content/436051971.htm]

מצב רב-נתיב: דומה מאוד למצב תשדיר, ההבדל היחיד הוא שמצב רב-נתיב יכול לסנכרן עבור subnets שונים משלו לעומת תשדיר שיכול לסנכרן רק מכשירים ב-subnet שלו. השרת שולח הודעות לכתובת הרב-נתיב שלו או לכתובת ברירת המחדל שהיא 224.0.1.1 עם שדה מצב 5 ומשם התהליך זהה לתהליך במצב התשדיר. כפי שניתן לראות בתרשים:



[מקור: https://techhub.hp.com/eginfolib/networking/docs/switches/5120si/cg/5998-8503_nmm_cg/content/436051971.htm]

מבנה החבילה

אחרי ההבנה של כל חלקי הפרוטוקול, אפשר לקרוא את מבנה ה-packet של הפרוטוקול. חבילת NTP בנויה ממספר שורות של 32-ביט, מבנה החבילה הוא:

LI	VN	Mode	Strat	Poll	Prec
Root delay					
Root dispersion					
Reference ID					
Reference timestamp (64)					
Origin timestamp (64)					
Receive timestamp (64)					
Transmit timestamp (64)					
Extension field (optional)					
Extension field (optional)					
Key identifier					
Message digest (128)					

[לקוח מתוך what-when-how.com]

חלק גדול מהשדות בחבילה אינן רלוונטיות להבנת הפרוטוקול ולכן לא יהי הסבר מעמיק עליהם. להלן הסבר מעט שטחי יותר על כל שדה בחבילה:

1. LI - דגל שמציין האם יש זינוק שנייה.
2. VN - גרסת הפרוטוקול.
3. Mode - מצב הפרוטוקול.
4. Strat - סטרטום (דרגת) השולח.
5. Poll - החזקה של 2 הקרובה ביותר למרחק הזמן המקסימלי בין חבילות רצופות. כלומר כל כמה זמן הלקוח רוצה לקבל פקטות, ככל שהשעון של הלקוח מדויק יותר כך יצטרך פחות בדיקות והערך יהיה גדול יותר.
6. Prec - החזקה של 2 הקרובה ביותר לדיוק השעון של השולח. ניתן להשיג את דיוק השעון באמצעות system call לקרנל (clock_gettime בלינוקס). (יכול להיות שלילי)
7. Root Delay - הזמן שלוקח לחבילה מסע הלך-חזור בין השרת ללקוח ללא זמן העיבוד ע"י השרת בשניות. ככל שהמספר גדול יותר החיבור בין המחשבים איטי יותר. (מספר עשרוני)
8. Root dispersion - הטעות המקסימלית שיכולה להיות בשעון המקור לעומת שעוני ההתייחסות.



9. Reference ID - קוד ASCII של ארבע תווים המייצגים את השעון שעליו שעון ההתייחסות מכוון. לדוגמא, אם השעון התייחסות מכוון לפי שעון ה-GPS אז הקוד יהיה GPS ואם הוא מכוון לפי שעון הטלפון של חייל הים האמריקאי, הקוד יהיה USNO. הקוד הזה נשלח אך ורק אם החבילה נשלחת משעון ההתייחסות. אם לא אז הקוד יהיה כתובת ה-IP בחילוק ל-4 האוקטטות.
10. כל ה-Timestamp-ים הם השדות אשר ממלאים כאשר רץ האלגוריתם המוסבר בחלק 1.
11. Key identifier - מפתח המשמש לאבטחת הפרוטוקול. יוסבר בפרק זמן ואבטחה.

צילום ה-packet של NTP ב-Wireshark:

```
90 bytes captured (720 bits) on interface \Device\NPF_{3308248F-9B90-4C73-A849-CD27932AE803}, id 0
Ethernet II, Src: Sagemcom_6d:6e:b7 (b0:bb:e5:6d:6e:b7), Dst: IntelCor_7d:4d:21 (04:d3:b0:7d:4d:21)
Internet Protocol Version 4, Src: 20.101.57.9, Dst: 10.0.0.19
User Datagram Protocol, Src Port: 123, Dst Port: 123
Network Time Protocol (NTP Version 3, server)
  Flags: 0x1c, Leap Indicator: no warning, Version number: NTP Version 3, Mode: server <
    [Request In: 485]
    [Delta Time: 0.084072000 seconds]
    Peer Clock Stratum: secondary reference (3)
    Peer Polling Interval: 17 (131072 seconds)
    Peer Clock Precision: 0.000000 seconds
    Root Delay: 0.001831 seconds
    Root Dispersion: 0.031387 seconds
    Reference ID: 25.66.230.0
    Reference Timestamp: Jun 17, 2023 13:26:21.774614599 UTC
    Origin Timestamp: Jun 17, 2023 13:32:53.057831499 UTC
    Receive Timestamp: Jun 17, 2023 13:32:52.821612699 UTC
    Transmit Timestamp: Jun 17, 2023 13:32:52.821615399 UTC
```

זמן ואבטחה

כפי שהוסבר בהיסטוריית הפרוטוקול, כבר בגרסה השנייה של הפרוטוקול, נוסף - מנגנון אבטחה. למה? איך המנגנון הזה עובד? בפרק הזה נענה על כל השאלות האלו.

כדי להבין להבין למה צריך לאבטח את הפרוטוקול, נדמיין עולם בו הפרוטוקול הוא בלי אבטחה, כמו בגרסה הראשונה של NTP. נניח ויש לנו בעולם הזה דלת בניין נעולה עם קוד ששומרת את הפרחחים מחוץ ללובי. כמו רוב הדלתות בניין עם קוד, הדלת הזו תמיד נעולה עד שמקלידים בה את הקוד ואז, למשך 3 שניות, הדלת פתוחה ואז היא ננעלת שוב. הדלת שומרת על זמן מדויק באמצעות תחזוק שעון שהיא מכוונת באמצעות NTP. נניח שאחד הפרחחים למד מחשבים בפקולטה ורוצה לפרוץ את הדלת.

איך יעשה זאת? מכיוון שה-NTP לא מאובטח, הוא יכול להריץ שרת שיתחזה לשרת NTP עם סטרטום גבוהה ב-1 משל הדלת ולאחר שאדם פותח אותה, הוא שומר על השעון באותו שנייה ועכשיו הוא יכול לעשות בלאגן...

דוגמה זו אומנם נראית סתמית ולא ממש רלוונטית לעולם הסייבר אך המון מערכות אבטחה עובדות כמו דלתות בניין עם קוד. לדוגמה, מספר פקודות בלינקס צריכות הרשאות של משתמשים מיוחדים כמו משתמש-על או משתמש שורש. כדי להריץ פקודות מסוג זה מוסיפים אותה כארגומנט לפקודה "sudo" ואז, לפני שהפקודה רצה, הטרמינל מבקש להכניס את סיסמת המשתמש שבעל הרשאה לביצוע הפקודה.

פעמים רבות נרצה להריץ כמה פקודות מסוג זה אחד אחרי השניה אך זה מאוד לא נעים להכניס כל פעם מחדש את הסיסמה. לכן, פעם אחת של הכנסת סיסמה היא כמו פתיחה של דלת שדרכה יכולות לעבור כל פקודה למשך 5-15 דקות תלויי הפצה (וניתן לשנות את הזמן הזה). כמו בדוגמה הקודמת, אם הפרוטוקול לא היה מאובטח, היה אפשרי להכניס דרך ה"דלת" הזו כל פקודה זדונית שרוצים ללא סיסמה. בנוסף לכך, כפי שצוין בפרק על השימושים הנפוצים, גם פרוטוקולי אימות כמו Kerberos.

צריכים זמן מדויק כדי למנוע התקפת "שליחה מחדש" ואם האקר מסוגל להתעסק עם השעונים של המחשבים, הוא יוכל פשוט לעבור בדילוג את המחסום הזה ולבצע התקפה מסוג זה כאוות נפשו.

אחרי שהבנו למה כל כך חשוב שהפרוטוקול הזה יהיה מאובטח, נשאל: איך הוא שומר את עצמו בטוח? כדי להבין את זה, צריך קודם כל להבין איך מאבטחים דברים ברשת באופן כללי.

בפשטות, יש שתי שיטות עיקריות לאבטחה, מפתח סימטרי ואסימטרי. בשתי השיטות האלו האבטחה עובדת בכך שלמחשב השולח יש מספר מאוד גדול וסודי k_l שנקרה ה"מפתח הנועל" ולמקבל יש גם מספר מאוד גדול (אך לא בהכרח סודי) k_o שנקרה ה"מפתח הפותח". המפתחות קשורים אחד לשני בכך שקיימת פונקציות $P_{lock}(m, k)$, $P_{check}(m, c, k)$ כך שלכל 2 הודעות m_1, m_2 , מתקיים:

$$P_{check}(m_2, P_{lock}(m_1, k_l), k_o) \Leftrightarrow m_1 = m_2$$

אך כאשר תוצאות הפונקציות וההודעות ידועות, אי-אפשר בזמן סביר למצוא את המפתח.

במילים אחרות, המפתח הנועל הופך הודעה למספר רנדומלי שבעזרת המפתח הפותח ניתן לוודא שהוא באמת נוצר מההודעה וע"י המפתח הנועל. כאשר המחשב רוצה לשלוח הודעה m שהיא בוודאות שלו, הוא מוסיף לה את $c = P_{lock}(m, k_l)$ ואז המחשב המקבל יוכל לבדוק את אמינות ההודעה ע"י $P_{check}(m, c, k_o)$. אם המפתח הנועל מתפרסם, כל אחד יוכל לנעול כל הודעה ואי-אפשר יותר לשלוח הודעות אמינות. בפרט, ב-NTP ה-c שצוינה תצורף בשדה ב-packet שנקרה key digest.

בעבר, היו רק פונקציות שעובדות רק אם תמיד המפתח הנועל והפותח הם אותו הדבר (כלומר המפתחות סימטריים) ואז המחשבים היו צריכים שיהיה להם מספר משותף רנדומלי שלא ידוע לאף מחשב אחר. יש מספר דרכים ליצור מצב כזה אך לא נפרט עליהם. לכן, מכיוון ש-NTP הוא פרוטוקול מאוד ישן, אחד מהדרכים לאבטח אותו הוא באמצעות מפתח סימטרי שנקבע מראש עם שרת ה-NTP באמצעות פרוטוקול שנקרה NTS.

כעבור מספר שנים, מתמטיקאים הצליחו למצוא פונקציות שבהן ניתן שהמפתחות יהיו שונים ושלא יהיה אפשר בזמן סביר למצוא את המפתח הנועל מתוך הפותח ומכך נוצרה שיטת האבטחה שהיום שולטת באינטרנט: מפתח ציבורי - מפתח פרטי. בשיטה זו, המפותח הפותח הוא ציבורי וידוע לכולם בזמן שהפרטי ידוע רק למחשב השולח. שיטה זו טובה יותר מכיוון שאין צורך בהחלפת מפתחות התחלתית מסובכת ומסוכנת, אפשר פשוט לשלוח הודעות באופן חופשי לכל מחשב והוא יידע בוודאות מי שלח את ההודעה. לכן, כפי שצוין בפרק ההיסטוריה, שיטה זו גם נוספה לאבטחת הפרוטוקול.

סיכום

במאמר זה ראינו את ההיסטוריה של אחד מפרוטוקולי האינטרנט המבוגרים ביותר, מחיתוליו בשנים המוקדמות של האינטרנט על למצבו הנוכחי. כמו כן, ראינו בפירוט כיצד ניתן באופן מפוזר וללא אף נקודת כשל יחידה לעשות את ה(לכאורה)בלתי אפשרי: לשמור על כל השעונים באינטרנט מכוונים. ראינו כי NTP הינו פרוטוקול חיוני המאפשר לסנכרן את שעוני המחשבים ברשת לשעונים של שרתים בעלי זמן מדויק יותר. לבסוף ראינו כיצד ניתן להשתמש במנגנון הישן אך חשוב הזה בשביל מעשי זדון.

כמו כן, אופן פעולת פרוטוקול הינו על-ידי שליחות הודעות בין שרת ולקוח אשר מכילים מידע אודות השעות. על ידי כך ניתן לסנכרן שעונים תוך התייחסות להפרשים בניהם. אם היה דבר אחד שהיינו רוצים שכל קורא ייקח מהמאמר, הדבר הזה הוא: שכל דבר, לא משנה כמה נאיבי משעמם ו/או טריוויאלי הוא נראה, ניתן לניצול זדוני.

על המחברים

אלעד קמינסקי, בן 16 מזיכרון יעקוב ומשתתף בתכנית **אודיסאה**, מרכז מדעני העתיד, באוניברסיטת תל אביב לנוער. במסגרת תכנית אודיסאה, שבה אנחנו לומדים במסלול סייבר בשנה ג'.

יואב בם, בן 17 מראשון לציון, תלמיד בתכנית אודיסאה במרכז מדעני העתיד, באוניברסיטת תל אביב לנוער - שנה ג'.

באחד מהשיעורים שלנו באודיסאה איחרנו בעת החזרה מההפסקה. על כן, קיבלנו מטלה לכתיבת המאמר על NTP. אנו רוצים להודות לד"ר **שלומי בוטנרו** על הנחייתו לכתיבת המאמר ולאחיו של אלעד על העזרה בבחירת הנושא.



ביבליוגרפיה

- <https://www.engineersgarage.com/how-clock-in-computer-works/>
- https://en.wikipedia.org/wiki/Network_Time_Protocol
- <https://www.cisco.com/c/en/us/support/docs/availability/high-availability/19643-ntp.html#anc13>
- <https://www.eecis.udel.edu/~mills/exec.html>
- <https://www.globalknowledge.com/ca-en/resources/resource-library/articles/a-guide-to-network-time-protocol-ntp/>
- <https://www.techtarget.com/searchnetworking/definition/Network-Time-Protocol>
- <http://what-when-how.com/computer-network-time-synchronization/ntp-packet-header-ntp-reference-implementation-computer-network-time-synchronization/>
- <https://ieeexplore.ieee.org/document/9043039>
- <http://www.scientific-devices.com.au/pdfs/WeTransfer-NZvJB6Cw/Time%20&%20Frequency/Importance%20of%20Network%20Time%20Synchronization.pdf>
- <https://www.ntp.org/ntpfaq/NTP-s-def-hist/>
- <https://www.ntp.org/reflib/memos/hist.txt>
- <https://www.rfc-editor.org/rfc/rfc5905.txt>
- <https://www.baeldung.com/linux/sudo-extend-session-time>