

BGP - הפרוטוקול של האינטרנט

גרסא	נערך על ידי	הופץ בתאריך	כותב המסמך
1.04	-	03.04.12	גיא צברדלינג

הקדמה

את המסמך הזה כתבתי כדי להבין היטב איך עובד BGP ואיך ליישם אותו נכון, מטרה ראשית להגיע מוכן לראיונות עבודה הדורשים יידע בBGP כגון ISP למיניהם וזאת גם הסיבא שבחרתי ללמוד דווקא CCIP, המטרה השניה היא להכין את עצמי למבחן של סיסקו בנושאי BGP, כתבתי את המסמך הזה בצורה מופשטת שכל מי שרוצה יוכל להשתמש בו וללמוד את הפרוטוקול, הכל כתוב בשפה פשוטה, ובמידה ויש לכם הערות, אתם מוזמנים לשלוח לי דוא"ל בנושא: Guy.Zwerdling@gmail.com

בהצלחה.

הסטוריה ועובדות

פרוטוקול ה-BGP הינו הפרוטוקול הכי גדול ומורכב להעברת חבילות IP ולנתב אותם ברשת האינטרנט לפחות בימים אלו, סוגת פרוטוקול זה הינו distance vector שעובד עם path שלא כמו link-state שמחשב את העלות ליעד מבחינת רוחב פס או השהייה, ה-BGP הינו בגרסא 4 היום והגרסאות הקודמות 3 ו-2 כבר לא בשימוש היום וחשוב לזכור שזה פרוטוקול BGP ולא EGP (Exterior Gateway Protocol) למרות שהוא נחשב כEGP, למעשה EGP כבר לא נמצא בשימוש בימים אלו.

הפרוטוקול עוצב בצורה כזאת שהוא יוכל לנתב לחבילות IP דרך מערכות אוטונומיות גדולות (AS) ובהמשך למאמר הזה אנחנו נראה איך זה מגיע לידי ביטוי.

ישנם שני סוגים של BGP

- iBGP - נתבים המוגדרים על סוגה זאת עובדים זה עם זה בתוך אותה אוטונומיה

- eBGP - נתבים המוגדרים על סוגה זו עובדים בין שני אוטונומיות שונות כלומר העברת נתונים מאוטונומיה שונה לאוטונומיה שלנו

אנו משתמשים ב-eBGP כדי להעביר ניתובים מאוטונומיה אחת לאוטונומיה אחרת שהיא כבר לא באחריותנו או להפך, ב-iBGP אנו למעשה מעבירים את הניתובים ב-BGP בתוך הרשת שלנו, נשאלת השאלה למה שנשתמש ב-BGP לניתובים פנימיים ברשת שלנו (ובהמשך נחדד למה לא כדאי לעביר ניתובים ב-BGP ברשת שלנו)? התשובה היא למקרה מסויים בו אנו ISP ואנו רוצים להעביר את הניתובים דרך הרשת שלנו ל-AS שונות ולכן נשתמש ב-iBGP.

פרוטוקול BGP עובד על גבי TCP, אין לו keepalive ייחודי, הוא למעשה משתמש במנגנון של אמינות של TCP כדי לעשות את, מה שאומר שהוא סוג של פרוטוקול ברמת האפליקציה וה-TCP הוא למעשה הkeepalive של פרוטוקול זה

ברירת מחדל בנוגע לניתובים BGP מוצא את הדרך הטובה ביותר לרשתות על ידי הנתבי עם מספר ה-AS הקטנה ביותר, בדומה ל-RIP שעובד על פי קפיצות של ציודים שיש ברך ליעד, BGP עושה את אותה פעולה רק שהפעם מדובר על AS שיש בדרך. חשוב לזכור שבגלל זה אולי נרצה לשנות את הניתוב כי אנו יודעים שלמרות שהקפיצות גדולות כדי

להגיע לאותה רשת אך רוחב הפס גבוהה יותר ולכן ננסה לעשות מניפולציות כדי לגרום לBGP לעשות זאת.

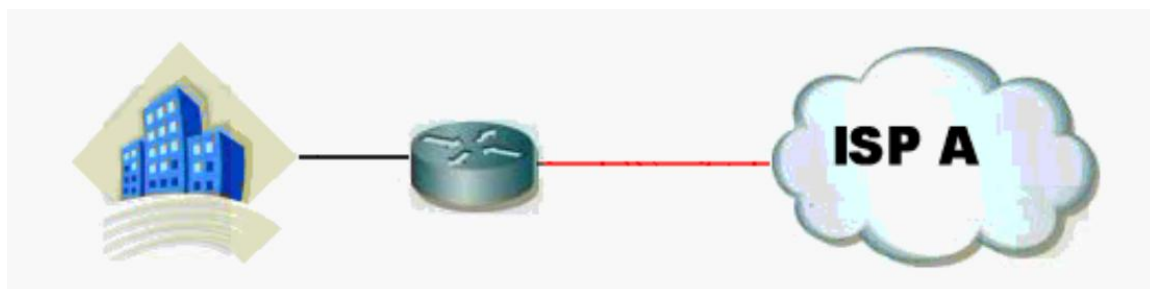
כדי לעשות מניפולציות על BGP לבחור נתיב מסויים במקום הנתיב הרגיל נשתמש בPolicy ונכליל בתוכו יכולות מסוימות של BGP כדי לבצע את הבחירה הזאת שאנו רוצים כהדרך הטובה ביותר ליעד.

הBGP תוכנן בצורה שהוא יעבוד לאט, על עדכון batch נשלח בין נתבי iBGP כל 5 שניות, ובין נתבי eBGP על 30 שניות, מה שאומר שאם נפלה רשת מסויימת שמחוברת לנתב שמדבר BGP אזי הוא יעדכן על זה רק כל 30 שניות לשכן eBGP שמקושר אליו, ואותו שכן יפיץ את העדכון הזה רק לאחר 30 שניות לשכן ebgp אחר, וכאשר מדברים על רשת האינטרנט זה מספרים גבוהים והעדכונים יהיו איטיים מאוד, נשאלת השאלה למה אם כן בחרו דווקא בפרוטוקול מסוג זה כפרוטוקול של אינטרנט, התשובה היא בגלל רשת האינטרנט שהיא כל כך גדולה. הרי ישנם רשתות שנופלות ועולות וכל נפילה כזאת מצריכה עדכון, אז נניח רשתות גדולות וקטנות נפלו ועלו או שרק מקצתן או רובן נפלו ועלו, כל נפילה כזאת מצריכה עדכון ואם הBGP היה מספק עדכונים בצורה מהירה כזאת סביר להניח שהיה כאוס ברשת האינטרנט שהרי כמו שאמרנו כל שניה יש איי שם רשת שנופלת ועולה וזה דבר שיכול לצרוך המון רוחב פס, לכן יצרו את הBGP באופן כזה שייקח זמן לעדכונים הללו לעבור הלאה.

סיבות להרצת BGP

אם אנחנו מחזיקים ברשת גדולה המשמשת כ-ISP זאת סיבה מספיק טובה ליישם BGP כי למעשה ISP רוצה לעביר ניתובים ותעבורה דרך ה-AS שלו ולכן להגדיר BGP במצב זה יכול להיות סיבה טובה להגדרת BGP.

אם אנו ארגון המכיל קישור יחיד ובודד ל-ISP זאת לא סיבה להגדיר BGP כיוון שיש לנו קישור יחיד ונוכל להגדיר ניתוב פשוט כדי להגיע לייעדים שלנו דרך אותו ISP

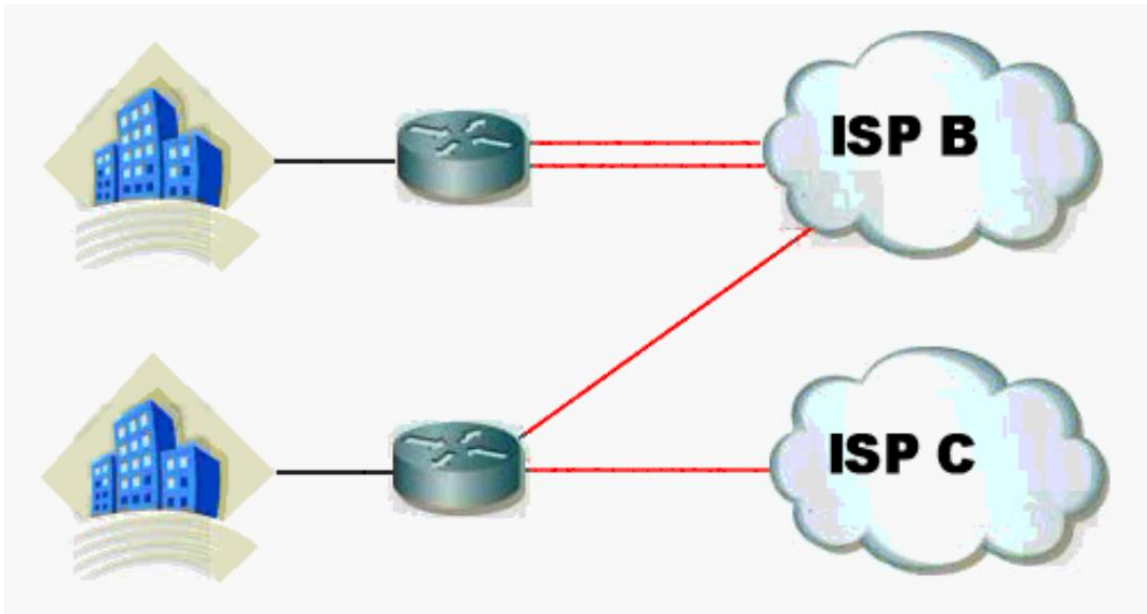


אם אנו ארגון שיש לו שני קישורים לאותו ISP גם זה לא סיבה להגדיר BGP מכיוון שנוכל לעשות חלוקת עומס או לשמור ממשק אחד האקטיב ואחד כגיבוי לראשון יוצא שכל פעם יש לנו רק ממש אחד מול ה-ISP שלנו



אם יש לנו שני ממשקים לשני ISP שונים זאת כן סיבה להגדיר BGP כי אולי לקבל שירות מסויים מה-ISP הראשון יחשב לעשות זאת כהדרך היעילה והמהירה להגיע לשם

שלא כמו ה-ISP השני שדרכו הדרך ליד יותר ארוכה ולכן נרצה להגדיר BGP כדי לדעת את הדרך הטובה ביותר ליעד דרך ה-ISP-ים הללו.



החוק של BGP

יש חוק לפי RFC 1771 שחייבים לזכור:

ה-BGP לא יכול לגרום לאוטונומיה אחת להשפיע על הניתוב של האוטונומיה של השכן ולגרום לו לבחור בניתוב כפי שאתה רוצה שהוא יהיה.

למעשה מה שזה אומר שלא ניתן לשלוט בניתובים של ה-AS השכנה, אנו למעשה יכולים לשלוט רק בניתובים שיש לנו בתוך האוטונומיה שלנו ולבחור בניתובים הטובים ביותר באוטונומיה שלנו בלי לעשות מניפולציה על הניתובים של ה-AS השכן.

זה כפי שאני לא אתן לאף אחד להגיד לי איך להכין את השוק שלי ככה אני לא יכול להגיד לאחרים איך לכין את השוק שלהם.

שכנות בBGP

חשוב לדעת שכל עוד שמדובר על שכנות אנו חייבים להגדיר את השכנות ידני שלא כמו פרוטוקולי ניתוב אחרים, כאשר הרעיון נובע מזה שאנו לא רוצים שיווצר אוטומטית שכנות בין הנתב שלנו לAS אחרת שאנו לא מכירים כי זה יכול להביא לידי יכולות תקיפה על הרשת שלנו או לגרום לבעיות ניתוב שיפצו מאותה נקודה שלא מורשת להתחבר לרשת שלנו.

כמו כן ישנם מספר סוגי מצבים בין שני שכנים

- idle - מצב שנראה ישירות לאחר שהגדרנו שכן חדש

- active - מצב שאם נשאר תקוע זה אומר שיש בעיה להקים שכנות מול השכן ושהנתב באופן אקטיבי מנסה ליצור שכנות מול השכן

- open sent - מצב שאומר שהנתב שלנו שלח הודעה לשכן להקמת שכנות

- open confirmed - אומר שהנתב שלנו קיבל את ההודעה מהשכן להקמת שכנות

- established - במצב זה נראה שזורם מידע בינינו לבין השכן וכך נדע שהשכנות הוקמה בהצלחה בין השניים

לאחר שהשכנות הוקמה בהצלחה הודעות Hello או יותר נכון הודעות של TCP ישלחו כל 60 שניות בין שני נתבים, והתמודדות עם נפילה תהיה כל 180 שניות, מה שאומר שאכן בכל נפילה של רשת שתהיה אנו נמתין 180 שניות לפני שנשלח עדכון על נפילה (חשוב לזכור שעדכון על נפילה ישלח רק כל 30 שניות בין eBGP כמו שציינתי קודם)

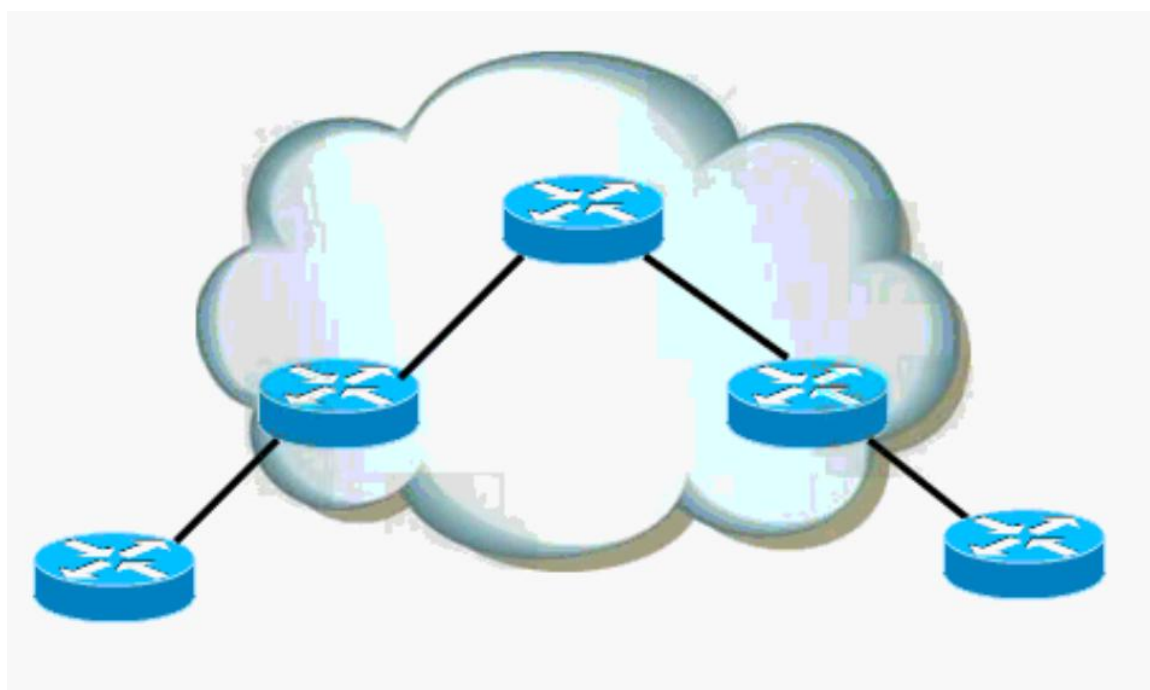
יש לנו יכולת להשתמש באוטנטיקציה של MD5 להגדרת סיסמא בבין שני נתבים האמורים להשתמש בשכנות של BGP.

חוק הסנקרון בBGP

החוק הזה קצת מבלבל לדעתי, והוא עובד באופן הבא:

ניתובים המגיעים על ידי BGP לנתב iBGP חייבים להיות מאומתים על ידי פרוטוקול ניתוב פנימי לרשת, אחרת אין לעביר את הניתוב הלאה.

זה אומר שאם נתב מחובר לנתב אחר בiBGP כלומר הם באותה אוטונומיה והנתב הראשון שלח לשני כתובות ניתובים מסויימים, הנתב השני יודא שהוא יכול להגיע לניתובים אלו אחרת לא יפיץ את הניתובים הללו הלאה. ניתן דוגמא למה דבר זה הינו שימושי במקרים מסויימים.



נניח שיש לנו נתב R1 שהוא מחובר בקישור יחיד לR2, ונתב R2 מחובר בחיבור יחיד לנתב R3, נתב R1 אינו מחובר ישירות לנתב R3 אך הם מוגדרים כשכנים, במקרה זה אם נתב R1 יקבל ניתוב של משהו שהוא הגיע מeBGP ויעביר אותו הלאה לR3 ונתב R3 יפיץ את הניתוב הלאה, במקרה זה אם יגיעו חבילות לR3 המיועדות לכוון הרשתות שמאחורי R1 הוא ידאג לעביר את הניתובים לכוון R1 דרך R2 וכשR2 יקבל את החבילות הוא לא יידע מה לעשות אותם כי הוא לא דובר BGP והוא לא יודע איך להגיע ליעד הזה ולכן יש לנו את החוק של סנקרון בBGP, אבל במקרה ואנו יודעים שהרשת מחוברת full-mash וכלל

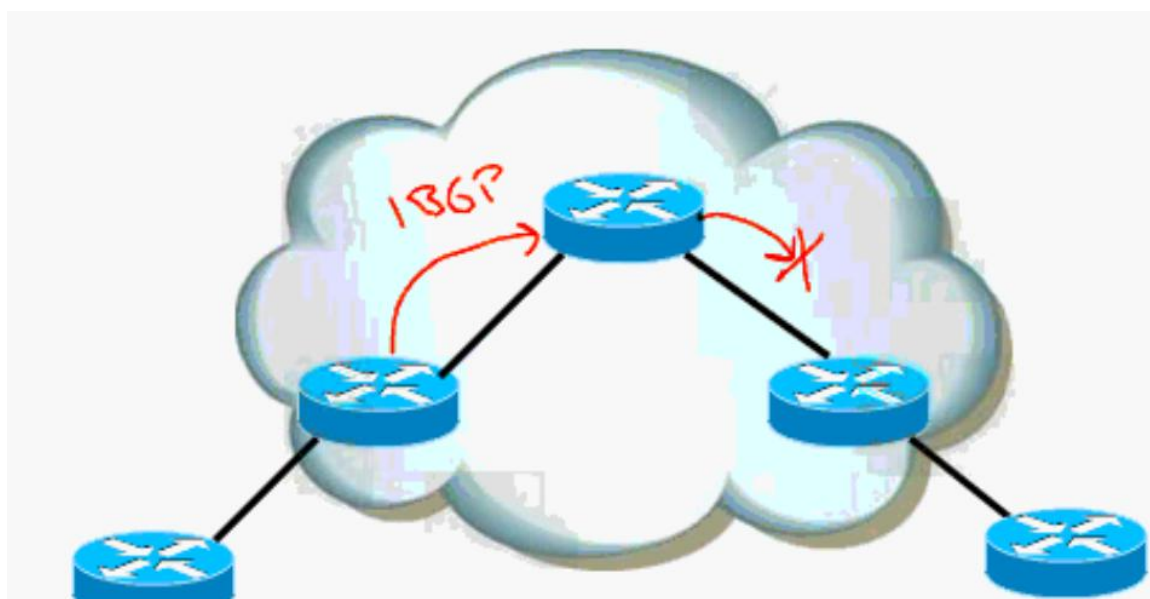
הנתבים מדברים BGP ואנו רוצים לוודא שהניתובים מגיעים לכל קצוות הרשת נוכל להסיר את הsynchronization מהנתבים בטופולוגיה של BGP.

החוק של split horizon

בפשטות החוק הזה אומר לעולם אל תעביר נתונים על ניתובים למקור ממנו קיבלת את הניתובים הללו.

בBGP זה יותר מורכב מזה, בBGP החוק אומר את האופן הבא:

אם קיבלת את הניתוב מiBGP לעולם אל תעביר את הניתוב לנתב שכן שהוא iBGP.



מה שאומר שאם נתב מסויים קיבל ניתוב מנתב שכן שהוא עובד iBGP אזי הוא לא יעביר את הניתוב לכל נתב שכן אחר שגם הוא עובד iBGP, המטרה כאן היא למנוע לופים במקרה של full-mesh, אבל במקרה ואין לנו full-mesh אנו ניהיה בבעיה כיוון שהניתוב לא יוכל להגיע הלאה לכלל קצוות הרשת עם ההגדרה הזאת ולכן נוכל להשתמש ב route reflector כדי שהניתוב הזה יגיע לכלל קצוות הרשת.

תכונות של BGP

יש לנו ב-BGP המון יכולות להגדיר מה הנתיב המועדף לכוון היעד, ה-BGP מחשב את העלות ה-metric הטוב ביותר ליעד על ידי סדרה של תכונות שיש לו, יש לנו מספר קטגוריות של תכונות:

- well known - תכונות שחייבות להיות מוגדרות על מוצריו של היצרן והוא מכיל עוד שני תתי קטגוריות שונות

- mandatory - אלו התכונות המחייבות להיות בכל יצרן של BGP

- discroshionary - אלו התכונות שאופצינאליות למי שמגדיר את הרשת, הוא יכול להשתמש בהן או לא לפי מה שהוא צריך

- optional - לא חייב להיות עם תמיכה לכל היצרנים של BGP כיוון שהם לא רוצים מסיבה כזאת או אחרת לייצר את זה במוצר הם יכולים לעשות זאת

- transative - אומר שאותה יכולת שיש ליצרן ימשיך להיות מועבר ברשת בין אם היצרן האחר תומך בזה או לא

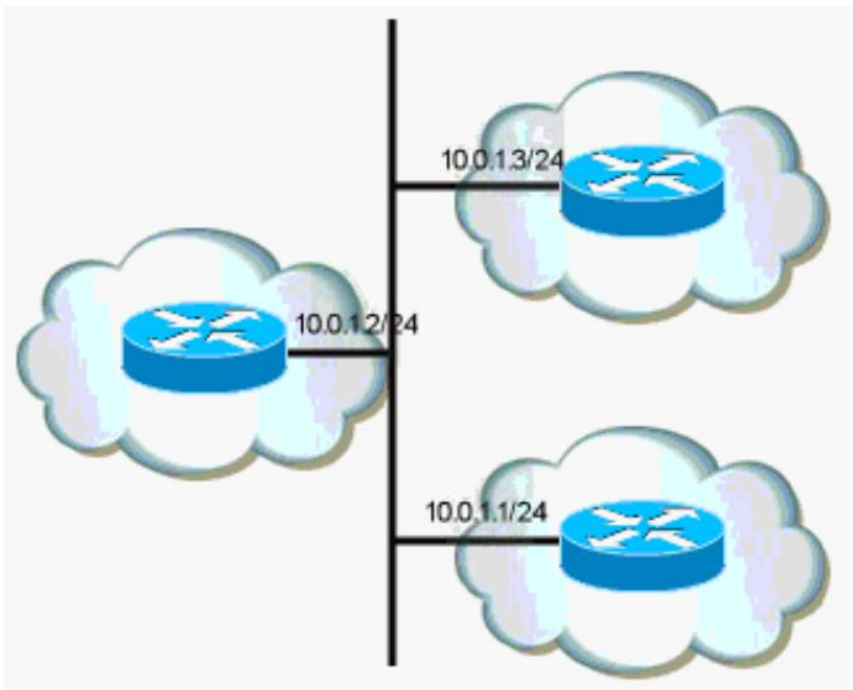
- non-transative - אומר שאותו תכונה תוסר מחבילת ה-BGP על ידי נתב שלא מבין אותה או שהוא בוחר לא להמשיך עם תכונה זו

סוגי תכונות:

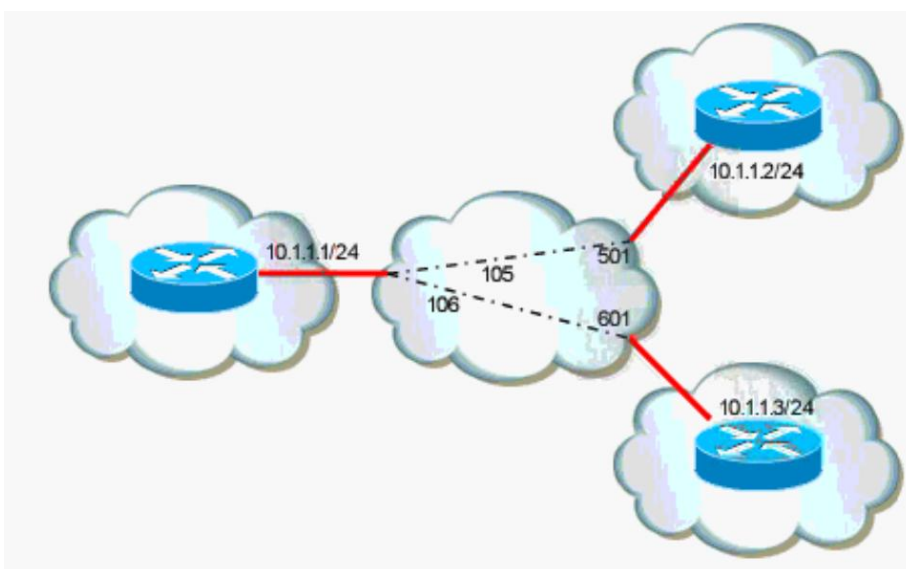
- AS-Path - סוג תכונה לבחירת הדרך המועדפת על פי מספר ה-AS שהחבילה עוברת בדרך, למעשה לכל יעד ברשת יהיה רשום לו כמה AS צריך לעבור כדי להגיע אליו, אופן עבודה זה מונע גם לופים כיוון שאם נתב יקבל ניתוב על רשת שכדי להגיע אליה האחת האוטונומיות הרשומות הינה של אותו נתב אזי הוא יודע שיש לופ והוא לא יקבל את העדכון הזה

- next-hop - כל נתב שמפיץ ניתובים יכול להפיץ אותם לשכנו עם שינוי קל בחבילה של הניתוב שהוא עצמו הקפיצה הבאה ליעד, ככה שכל נתב בטופולוגיה יכול לשנות את הקפיצה הבאה כדי להגיד שזה הוא וככה החבילה תעבור את כל הנתבים עד אשר תגיע ליעד שלה, במקרה שכל הנתבים נמצאים באותו סגמנט הנתב המקבל את כאשר יפיץ

לשכנו אשר נמצא באותו סגמנט הוא לא ישנה את הקפיצה הבאה כיוון שמדובר על סגמנט משותף,



במקרה של FR אם עשו יישום של NBMA MULTIPOINT NETWORK אז למרות שהם באותו סגמנט עדיין לא כולם מחוברים ישירות ולכן צריך למצא לזה פתרון כי כן צריך שהנתב המעביר כן יגדיר את עצמו כהקפיצה הבאה, במקרה זה יש שני אופציות, או להגדיר את ה FR כ-point to point בין כולם, אם לא אז צריך להגדיר דברים מיוחדים ב BGP כדי שבתצורה זאת זה עדיין יעבוד



origin - תכונה זאת אומרת אומרת לנתב איפה הניתוב שהגיע עכשיו נוצר ויכולים להיות שלושה אופציות

1. I - (IGP) - אומר שהניתוב נוצר על ידי נתב שהוא פנימי לאוטונומיה ונוצר על ידי פקודת network בנתבים באוטונומיה

2. E - (EGP) - אומר לנתב שהניתוב נוצר על ידי נתב חיצוני לאוטונומיה

3. Unknown (?) - ברגע שעשו הפצה מחוץ הBGP לתוך הBGP מה שנקרא redistribute במקרה זה יהיה סימן שאלה שמחווה על לא ידוע

local preference - נותן לנו שליטה להעדיף ניתוב מסויים על פני ניתוב אחר, ככל שהערך שלו גבוהה יותר ככה זה מועדף יותר

weight - תקני לסיסקו בלבד, נותן שליטה לניתוב רק בנתב שעליו הגדרנו את זה, זה לא משפיע על כלל הנתבים.

automatic aggregate - מייצע את הנתב שהניתוב שהוא מקבל הינו מסוכם ויתכן שיש עוד תת סגמנטים

MED - multi-exit discriminator - זה סוג של מטריק שאנו מציעים לאוטונומיה השכנה באיזה נתיב להשתמש, במצב ברירת מחדל אין שימוש בזה בנוסף חשוב לזכור שהערך הנמוך ביותר הוא הטוב ביותר

aggregaator - אומר לנתב מי עשה את הסיכום של הניתוב, כלומר מי IP ADDRESS שעשה את הסיכום

community - זה למעשה מטייג את הניתוב מכיוון מסויים

הקריטריון לבחינת הניתוב המועדף

0. לא יכלול ניתוב בטבלת הניתוב אם אין לנו דרך להגיע לקפיצה הבאה של אותו ניתוב.

1. יעדיף את המשקל weight הגבוהה ביותר

2. יעדיף את ה local pref הגבוהה ביותר
3. יעדיף ניתובים שהנתב בעצמו יצר אותם
4. יעדיף את הדרך הקצרה ביותר ל AS של היעד
5. יעדיף את הערך origin הנמוך יותר EGP < IGP >?
6. יעדיף את ערך ה MED שהוצע על ידי אוטונומיה אחרת
7. יעדיף נתיב חיצוני EBGP מאשר נתיב פנימי IBGP לכוון היעד
8. נתיב שהוא IBGP נעדיף את הנתיב שהכי קרוב אלינו - כלומר מהשכן IGP שהכי קרוב אלינו
9. נתיב שהוא EBGP נעדיף את הנתיב שהוא הכי ישן מבחינת עדכון (כי הכי ישן הוא הכי יציב)
10. העדפה של הנתיב דרך הנתב עם ה BGP router ID הנמוך ביותר

חשוב לדעת לפני שעובדים עם BGP

ישנם מספר דברים שחשוב לזכור או לדעת לפני שאנחנו מחליטים להגדיר BGP כי זה יכול לגרום לכל מיני דברים עתידיים שלא רצויים:

- מומלץ שיהיה לנו RAM128 בנתב עליו עתיד ה BGP לרוץ עם כל טבלאות הניתוב שלו בגלל הגדלים שלהם

- חשוב לדעת שיש ב IOS פיצרים שצורכים CPU, חשוב לזכור שגם אם לא משתמשים בפיצרים הללו זה עדיין צורך CPU או משאבים מהמערכת, לכן חשוב לדעת שאם יש פיצר שאנחנו לא משתמשים בו מומלץ לכבות את אותו פיצר ולנסות להגדיר את הנתב בצורה כזאת שיתמוך רק ב BGP

- מומלץ למנוע console login, שזה גם לוקח הרבה משאבים במיוחד שאנחנו נשתמש בדיבאג על הציוד, לכן מומלץ לעבוד עם מערכת syslog שתאגור לנו את הלוגים ולא לאגור אותם בציוד עצמו

- כדי לדאוג שהנתב לא ינצל את כל המשאבים רק על ניתובים וידאג למקורות בשביל המערכת של הנתב עצמו מומלץ להשתמש ב Scheduler Allocated במצב גלובאל

אני רוצה לעצור פה ולהתעקב על דוגמאות שניתן לעשות כדי לנצל את המשאבים של המערכת בצורה טובה

אנחנו נרצה לבדוק שני דברים עיקריים, כמה ה CPU נצרך מהמערכת על BGP וכמה זיכרון נצרך

נרצה לחתוך את הנתונים הללו על ידי מספר פקודות, קודם כל בואו נכין פקודת אליאס שתוכל לעזור לנו בהמשך

```
alias exec proc show proc cpu | excl 0.00%__0.00%__0.00%
```

הפקודה הזאת מבטיחה שכל פעם שנבצע את הפקודה proc אנחנו נראה את התהליכים של ה CPU שרצים מבלי לראות את התהליכים שלא צורכים מהנתב כלום בחמש שניות האחרונות או דקה אחרונה או חמש דקות אחרונות (0.00%__0.00%__0.00%), זה יעזור לנו לעשות חתך ולראות בדיוק מה נצרך בנתב

```
show proc cpu | inc BGP
```

יציג לנו את התהליכים הספציפיים של BGP, ויש ארבעה מהם שנראה אותם כאשר נגדיר את BGP בשלמות

- BGP router - יציג לנו את הנתונים מבחינת צריכה על שכנות בין הנתבים לנתב שלנו

- BGP I/O - לוקח צריכה בשביל חבילות מידע של הפרוקטוקול, זה כולל עדכונים והודעות של keepalive על גבי TCP

- BGP Scanner - צורך על העץ של הפרוטוקול, כלומר בודק שאכן הכל מנגן כשורה
שהניתובים אכן תקפים וזמינים ולוודא שהקפיצה הבאה לא השתנתה ומוצא את הניתוב
הטוב ביותר ליעד

- BGP open - הצריכה של שיחת TCP על קשרי שכנות חדשה מול שכן חדש

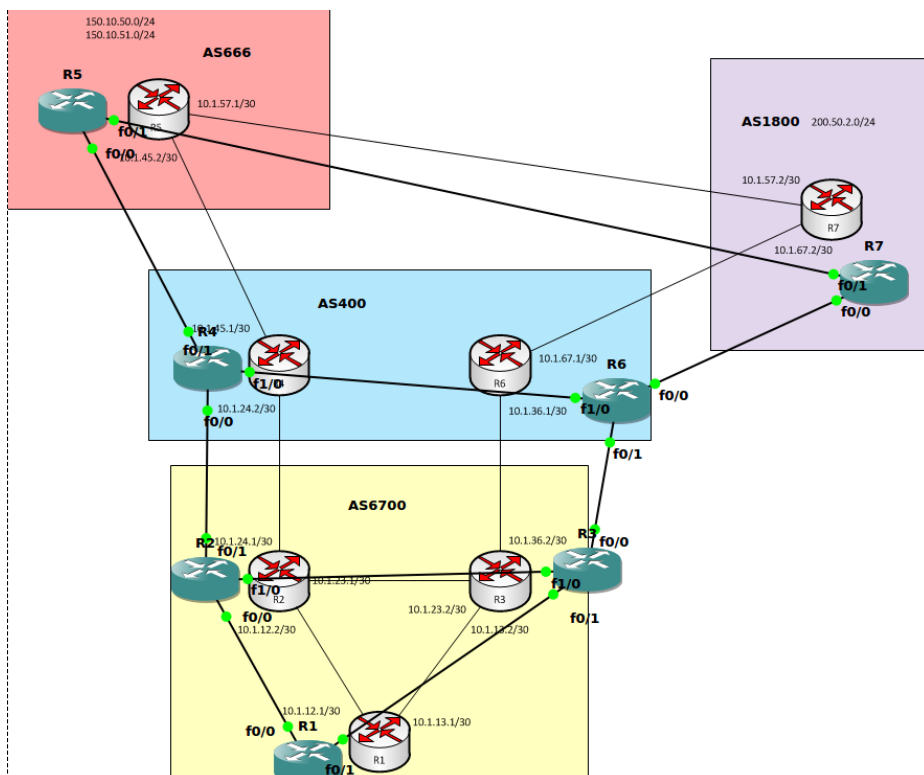
נוכל להשתמש באותה פקודה כדי לראות כמה זיכרון תכלס נלקח על כל תהליך כזה

show proc mem | inc BGP

חשוב לדעת את כל הנתונים הללו לפני שאנחנו מחליטים להריץ BGP על מנת שהוא ירוץ
בנתב בצורה יעילה יותר וניצול טוב של כל המשאבים שיש לנו ברשת

הגדרות של BGP

יצרתי תצורת רשת יחסית פשוטה שעליה נבדוק מספר דברים, את הרשת הזאת יצרתי
בסימולטור בשם GNS3 ובכל נתב יש לי IOS של 3750:



את קבצי ההגדרות לתצורה זאת ניתן להוריד מהאתר בו ראיתם את מסמך זה.

ראשית נצא מתוך נקודת הנחה שהרשת ב BGP מוגדרת כפי הנדרש, לכל נתב יש BGP שרץ בתוך האוטונומיה שלו וכדי לעשות זאת אנו נדרשים להקליד את הפקודה הבאה:

```
Router bgp <AS number>
```

בביצוע הפקודה הזאת אנו למעשה אומרים לנתב לעבוד BGP, עכשיו נדרש להגדיר שכנות

```
neighbor <ip address> remote-as <as number>
```

```
neighbor <ip address> next-hop-self
```

```
neighbor <ip address> update source loopback 1
```

למעשה הגדרנו את ה-AS של השכן, כמו כן אמרנו לנתב שכל ניתוב שאתה מפיץ הלאה הקפיצה הבאה תיהיה אתה לפני שאתה מפיץ הלאה, בנוסף הגדרנו את הפרוטוקול לעבור על כתובת מקור שתציג אותו בשכנות והיא loopback 1, חשוב לזכור שעל הנתבים האחרים חייב להיות מופיע בטבלאות הניתוב המקבילות ניתוב המציין איך להגיע לכתובת loopback1 בנתב שהפיץ את הניתוב

את כלל הנתבים בתצורה אני מגדיר עם loopback מקומי שאיתו אני אעבוד על שכנות כדי שיהיה לנו יותר קל לנתר ולתפעל תקלות בתצורה, שימו לב התצורה אכן לא מוגדרת כמו שצריך על מנת שנוכל לעשות בדיקות ולראות מה חסר

הכתובות מתחלקות באופן הבא:

```
network number> . <vlan number> . <local router number/remote router >  
<number> . <random
```

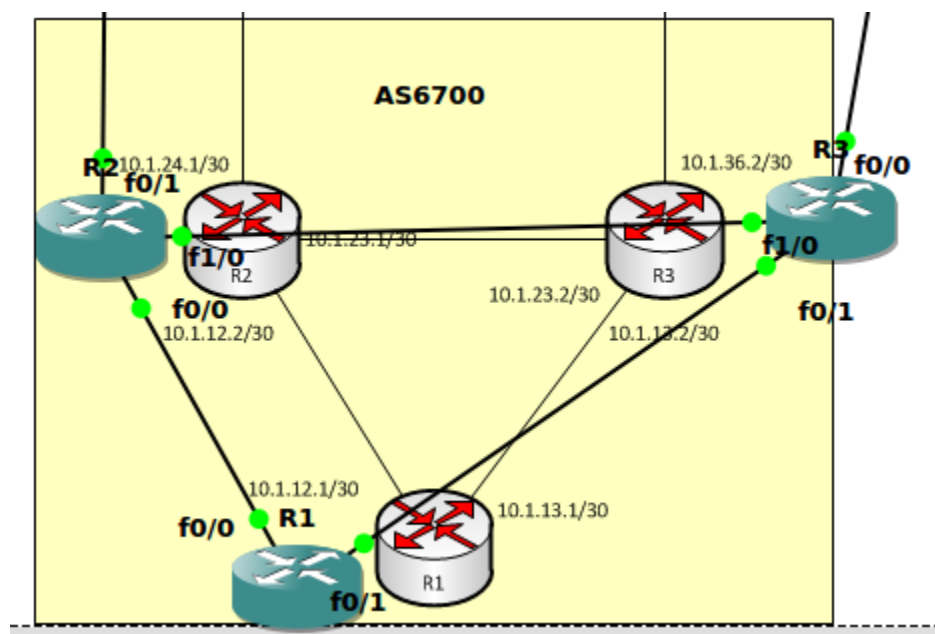

דוגמא לכתובות כדי שיהיה מובן היטב מה עשיתי ולמה התכוון המשורר:

10.1.12.1 - את הכתובת הזאת הגדרתי בנתב R1, כתובת הרשת (כלל הרשת) הינה 10, כתובת ה-VALN הינו 1, האוקטטה השלישית עומדת לקישור שבין R1 לבין R2 ולכן הוא 12, האוקטטה האחרונה היא רנדומאלית ובמקרה שלי ציינתי 1, לא לשכוח שמדובר על קלאס C ולא קלאס A כפי שנדמה מבחינת הסבנטציה.

אתם יכולים לעקוב אחרי מה שביצעתי או לעשות הגדרות בעצמם.

בדיקות תקינות - troubleshooting

את הבדיקה הראשונה נתחיל באוטונומיה 6700



אז דבר ראשון נבדוק את BGP שלנו

show ip bgp sum

```

R1#show ip bgp sum
BGP router identifier 1.1.1.1, local AS number 6700
BGP table version is 1, main routing table version 1

Neighbor      V    AS MsgRcvd MsgSent  TblVer  InQ  OutQ Up/Down  State/PfxRcd
2.2.2.2        4   6700      0       0        0    0    0  never   Active
3.3.3.3        4   6700      0       0        0    0    0  never   Active
R1#

```

נוכל לראות שרשומים שני נתבים 2.2.2.2, 3.3.3.3 אך המצב שלהם לעולם לא היה פעיל מול נתב R1

כדי להתמודד עם בעיה זאת נצטרך לבדוק שהקישור בין הנתבים תקין ולכן אני אבצע את הפקודות הבאות:

Ping 10.1.12.2

Ping 2.2.2.2

```

R1#ping 10.1.12.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.12.2, timeout is 2 seconds:
!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 16/25/44 ms
R1#ping 2.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R1#

```

בפקודה הראשונה נוכל לראות שהקישור תקין כי הפינג עבר וחזר בהצלחה, אבל בפקודה השניה אין לנו קישור תקין, וחשוב לזכור שמדובר על פינג לכוון ה loopback2 בנתב R2 ולכן נבצע את הפקודה הבאה:

```

R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        I - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        fo/o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback1
10.0.0.0/30 is subnetted, 2 subnets
C       10.1.13.0 is directly connected, FastEthernet0/1
C       10.1.12.0 is directly connected, FastEthernet0/0
R1#

```

בביצוע פקודה זאת אנו בודקים למעשה שהנתב יודע להגיע לLOOPBACK2 בנתב R2 ומכיוון שאין לנו ניתוב כזה אנו צריכים להוסיף אותו ידנית.

```
R1(config)#ip route 2.2.2.2 255.255.255.255 10.1.12.2
```

לאחר מכן נבדוק שוב קישוריות לR2

```
R1#ping 2.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/13/28 ms
R1#
```

אחרי שראינו שהקישוריות תקינה בין שני הנתבים נוודא שאכן ה BGP בין שניהם עובד תקין

```
R1#sh ip bgp sum
BGP router identifier 1.1.1.1, local AS number 6700
BGP table version is 1, main routing table version 1

Neighbor      V     AS MsgRcvd MsgSent  TblVer  InQ  OutQ Up/Down State/PfxRcd
2.2.2.2        4    6700      0       0        0    0    0 never Active
3.3.3.3        4    6700      0       0        0    0    0 never Active
R1#
```

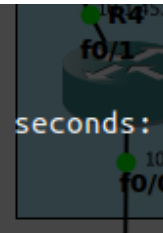
ניתן לראות כי עדיין לא קמה השכנות בין שני הנתבים, חשוב לזכור שב BGP לוקח זמן לכל עדכון ולכן אולי כדאי להמתין לפחות חצי דקה ולבדוק שוב, במקרה שלנו גם לאחר ההמתנה עדיין אין שכנות תקינה ולכן נבצע את הבדיקה הבאה:

```
R1#ping 2.2.2.2 source 1.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
.....
Success rate is 0 percent (0/5)
R1#
```

כפי שניתן לראות אנו מבצעים פינג עם כתובת מקור שהיא ה LOOPBACK של הנתב שלנו אזי אין קישוריות תקינה בין שני הנתבים, חשוב לזכור שאת בדיקה זאת אנו מבצעים מכיוון שהגדרנו את הפקודה UPDATE SOURCE בנתב תחת ה BGP ולכן כל חבילת מידע שתעבור לכוון השכן על הקמת שכנות תכיל את כתובת המקור 1.1.1.1, מה שנדרש

במצב זה לעשות הוא לבדוק ולוודא שמהנתב השני יש תקינות תקשורת לכוון הכתובת
1.1.1.1

```
R2#ping 1.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R2#
```

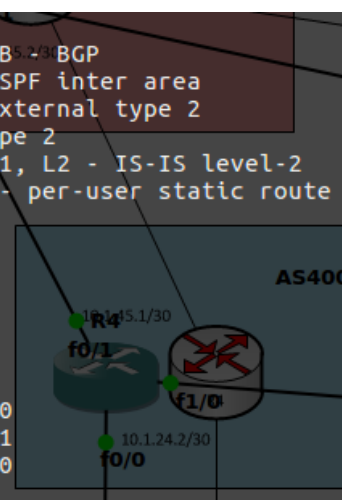


לאחר שראינו שאין תקינות תקשורת לכוון הכתובת הנדרשת אנו צריכים לבדוק בטבלת
הניתוב שלנו בנתב R2

```
R2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

2.0.0.0/32 is subnetted, 1 subnets
C       2.2.2.2 is directly connected, Loopback2
10.0.0.0/30 is subnetted, 3 subnets
C       10.1.12.0 is directly connected, FastEthernet0/0
C       10.1.24.0 is directly connected, FastEthernet0/1
C       10.1.23.0 is directly connected, FastEthernet1/0
R2#
```



למעשה לנתב R2 אין מושג איך להגיע לכתובת יעד 1.1.1.1 ולכן נגדיר ניתוב סטאטי
ונחזור לנתב R1 כדי לראות אם יש שכנות

```
R2(config)#ip route 1.1.1.1 255.255.255.255 10.1.12.1
```

לאחר שהגדרנו את הניתוב הזה אם חזרנו לR1 לאחר זמן מה כבר בCONSOL נ וכל
לראות לוג המתייחס לשכנות בין שני הנתבים הללו

```
*Mar  1 00:53:43.059: %BGP-5-ADJCHANGE: neighbor 2.2.2.2 Up
```

נוכל להקליד שוב את הפקודה שהקלדנו ממקודם כדי לראות את השכנות בין הנתבים.

```
R1#sh ip bgp sum
BGP router identifier 1.1.1.1, local AS number 6700
BGP table version is 1, main routing table version 1

Neighbor      V    AS MsgRcvd MsgSent  TblVer  InQ  OutQ Up/Down  State/PfxRcd
2.2.2.2       4   6700     10      10       1    0    0 00:06:39      0
3.3.3.3       4   6700      0       0       0    0    0 never      Active
R1#
```

לאחר שראינו שהשכנות תקינה נעבור להגדיר כן בכל שאר הנתבים בטופולוגיה כדי לראות אם יש שכנות תקינה

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	4	6700	19	19	1	0	0	00:15:28	0
3.3.3.3	4	6700	0	0	0	0	0	never	Active

R2(config)#
 *Mar 1 01:09:19.299: %BGP-5-ADJCHANGE: neighbor 3.3.3.3 Up
 R2(config)#do sh ip bgp sum
 BGP router identifier 2.2.2.2, local AS number 6700
 BGP table version is 1, main routing table version 1

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	4	6700	19	19	1	0	0	00:15:44	0
3.3.3.3	4	6700	4	4	1	0	0	00:00:08	0

R2(config)#

במקרה של נתב R3 מול R6 גיליתי שלא היה מוגדר בכלל שכנות, כדי לטפל בזה הוצרתי להגדיר שכנות מחדר שמו שציינתי למעלה ואז בטבלת ה BGP ראיתי את הנתון הבא:

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	4	6700	20	20	1	0	0	00:16:28	0
2.2.2.2	4	6700	16	16	1	0	0	00:12:34	0
6.6.6.6	4	400	0	0	0	0	0	never	Idle

R3#

חשוב לזכור שמצב IDLE על שכן אומר שמעולם לא נשלח חבילת מידע לכיוון השכן, וזה מכיוון שאחרי כל הוספה של שכן צריך לעשות ל BGP סוג של רענון כדי שנתונים אלו יכנסו לטבלה שלו באופן תקין, לאחר כמה פעמים של רענון עדיין ראיתי שהמצב נשאר IDLE והתחלתי לבדוק לעומק ומצאתי שיש חוק שאומר ש EBGP חייב להיות מחובר ישירות ובמקרה שלנו נכון שיש חיבור ישיר אבל אנחנו משתמשים בכתובת לוגית בשכנות שבין שני הנתבים וזה לא מחובר ישירות, לכן נדרש להגדיר עוד שני פקודות מסויימות על הנתב מול השכן הזה כדי שנוכל ליצור שכנות

```
neighbor 6.6.6.6 remote-as 400
neighbor 6.6.6.6 ebgp-multihop 2
neighbor 6.6.6.6 disable-connected-check
neighbor 6.6.6.6 update-source Loopback3
neighbor 6.6.6.6 next-hop-self
```

מדובר על שני פקודות, הראשונה היא EBGP-MULTIHOP2 פקודה זו אומרת לנתב שכדי להגיע לשכן הזה יש לעבור שני מסוכות, הפקודה השניה היא -CONNECTED-CHECK-DISABLE הפקודה הזאת אומרת לא לבדוק שנתב זה מחובר ישירות אלינו, הפעלתי סניפר בזמן ביצוע הפקודה וראיתי את החבילות הבאות

BGP

172 570.519786	c2:02:6b:bb:0...	c2:02:6b:bb:0...	LOOP	60 Reply
173 574.347078	2.2.2.2	4.4.4.4	TCP	60 35422 → 179 [SYN] Seq=0 Win=16384 Len=0 MSS=536
174 574.367337	4.4.4.4	2.2.2.2	TCP	60 179 → 35422 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=536
175 574.377352	2.2.2.2	4.4.4.4	TCP	60 35422 → 179 [ACK] Seq=1 Ack=1 Win=1048576 Len=0
176 574.387497	2.2.2.2	4.4.4.4	BGP	99 OPEN Message
177 574.397649	4.4.4.4	2.2.2.2	BGP	99 OPEN Message
178 574.407689	2.2.2.2	4.4.4.4	BGP	73 KEEPALIVE Message
179 574.407764	4.4.4.4	2.2.2.2	BGP	73 KEEPALIVE Message
180 574.417816	4.4.4.4	2.2.2.2	BGP	73 KEEPALIVE Message
181 574.417817	2.2.2.2	4.4.4.4	BGP	73 KEEPALIVE Message
182 574.427908	4.4.4.4	2.2.2.2	BGP	73 KEEPALIVE Message
183 574.427936	2.2.2.2	4.4.4.4	BGP	73 KEEPALIVE Message
184 574.639243	4.4.4.4	2.2.2.2	TCP	60 179 → 35422 [ACK] Seq=103 Ack=103 Win=1042048 Len=0
185 574.639301	2.2.2.2	4.4.4.4	TCP	60 35422 → 179 [ACK] Seq=103 Ack=103 Win=1042048 Len=0

חשוב לזכור שבהודעות KEEPALIVE שנראה לאחר מכן יהיה TTL של 2 כפי שהגדרנו את מספר הקפיצות

BGP

