פרויקט גמר בקורס מערכות מבוזרות 2016

השוואה בין הפרוטוקולים HTTP -ל Web Socket באפליקציית צ׳ט

מהם הבדלים בין שני הפרוטוקולים מבחינת מספר ביטים, מספר פקטות וזמנים

מרצה: ד"ר יעקב אקסמן.

גיטהאב: https://github.com/Think-Smart/chat-Application-Compare-Protocols

הסטודנטים:

אורי אמיר	305399420	oriamir1@gmail.com
ברק תורגמן	305631293	barak.turgeman@gmail.com

הקדמה

נשאלת השאלה - מה היא הדרך העדיפה ביותר לבניית אפליקציית צ'ט. כמובן שיש את הפרוטוקול HTTP וכן את והפרוטוקול החדש (יחסית) Web Socket . את שני הפרוטוקולים נבחן בפרויקט זה.

איך עובדים הפרוטוקולים

HTTP - Hypertext Transform Protocol

HTTP הוא פרוטוקול תקשורת שנועד להעברת דפי HTML ואובייקטים.

התקשורת ב־HTTP מתחילה ביצירת תקשורת בין השרת ללקוח באמצעות פרוטוקול TCP, ונמשכת בסדרה של בקשות (requests) ותשובות (responses) שנשלחות על ידי הלקוח והשרת, בהתאמה. ראשית, הלקוח יוצר חיבור לכתובת ה-IP ולפורט שבו השרת נמצא. לאחר מכן נשלחת הבקשה, הכוללת את הכתובת של האובייקט המבוקש ופרטים נוספים על הבקשה ועל הלקוח. השרת קורא את הבקשה, מפענח אותה, שולח ללקוח תשובה בהתאם ולרוב מנתק את החיבור ללקוח כשהשליחה הסתיימה.

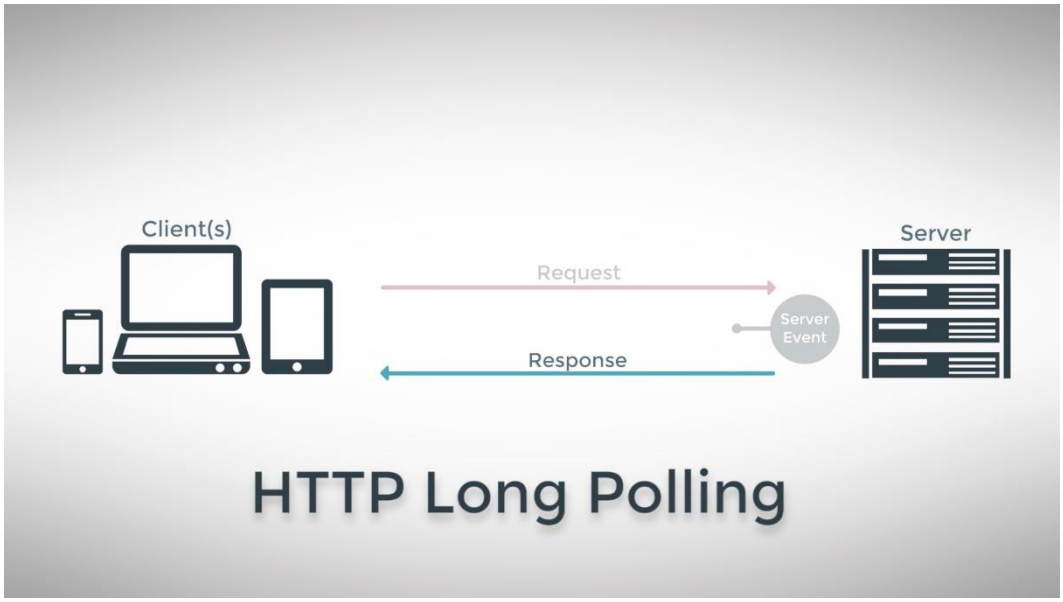
בפרוטוקול HTTP, אך ורק צד הלקוח רשאי ליזום בקשות, השרת תפקידו להגיב לאותן בקשות.

קיימות כמה שיטות לבנות אפליקציית צ'ט באמצעות הפרוטוקול HTTP. אחת מהם היא Long Polling שמשמעותה לשלוח בקשה לשרת לקבלת הודעה חדשה. השרת מתעכב עם התשובה ללקוח. ומחזיר תשובה לאחר שאחד מהמשתמשים באפליקציה שלח הודעה.

יש לא מעט חסרונות לשימוש בשיטה Long Polling:

- צריכת משאבים גבוהה שנובעת מכמות הבקשות הפעילות שאנחנו צריכים להחזיק.
- אנו חייבים לדאוג לזמינות התקשורת לדוג' במקרה שבקשות נופלות או שמתקבל Timeout.
- HTTP לא מיועד לספק תקשורת דו-כיוונית.

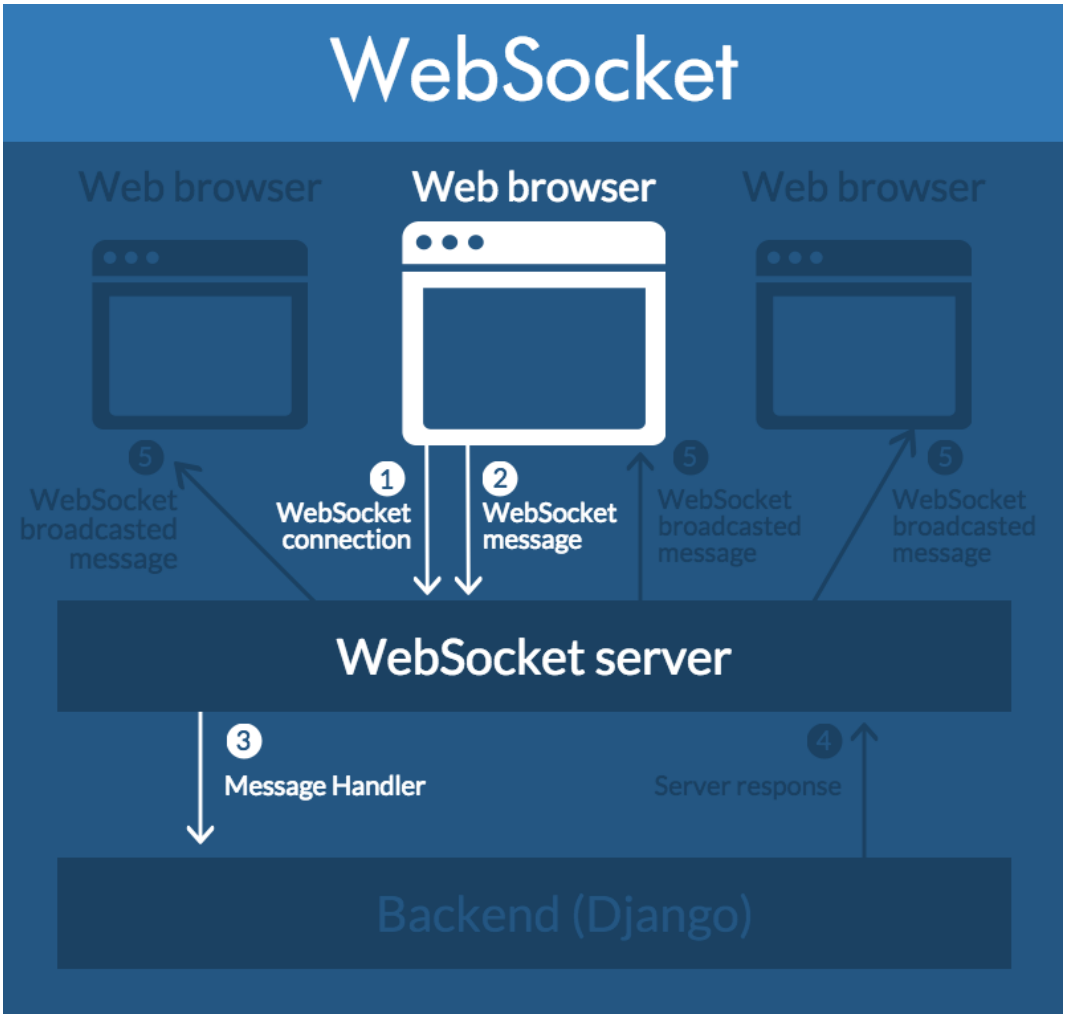
למרות כל החסרונות של השיטה הזו, במידה ונרצה לבנות אפליקציית צ'ט בהתבסס על פרוטוקול HTTP זוהי השיטה העדיפה.



Web Socket

ב-HTML5 הובאה לנו טכנולוגיה חלופית העונה לשם Web Sockets. זהו פרוטוקול המאפשר לבצע תקשורת דו-כיוונית על גבי צינור בודד שנשאר פתוח לכל אורך ההתקשרות ובו כל צד יכול לשלוח מידע לצד השני. גם כאן, הלקוח הוא זה שיוזם את ההתקשרות, ומרגע שהנתיב פעיל בין שני הצדדים, כל צד יכול לשלוח מידע באופן עצמאי לצד השני.

התקשורת ב-Web Socket מתחילה בתהליך שנקרא handshake אשר נועד לאמת ש-2 הצדדים יכולים באמת לתקשר ביניהם. לאחר שהתהליך עבר בהצלחה נוצר לנו נתיב שבאמצעותו הצדדים יכולים לתקשר אחד עם השני, המידע שנשלח ביניהם מועבר ביחידות הנקראות "messages" או "frames".



הסבר לתמונה:

1. הדפדפן מבצע חיבור Web Socket לשרת ה-Web Socket.
2. הדפדפן שולח באמצעות Web Socket הודעה לשרת ה-Web Socket.
3. בשרת יש אלגוריתם המטפל בהודעה.
4. האלגוריתם מחזיר תגובה לשרת ה-Web Socket.
5. שרת ה-Web Socket שולח הודעה למשתמשים המחוברים באמצעות Web Socket.

כלים

על מנת לבדוק את ההבדלים בין שתי הפרוטוקולים:

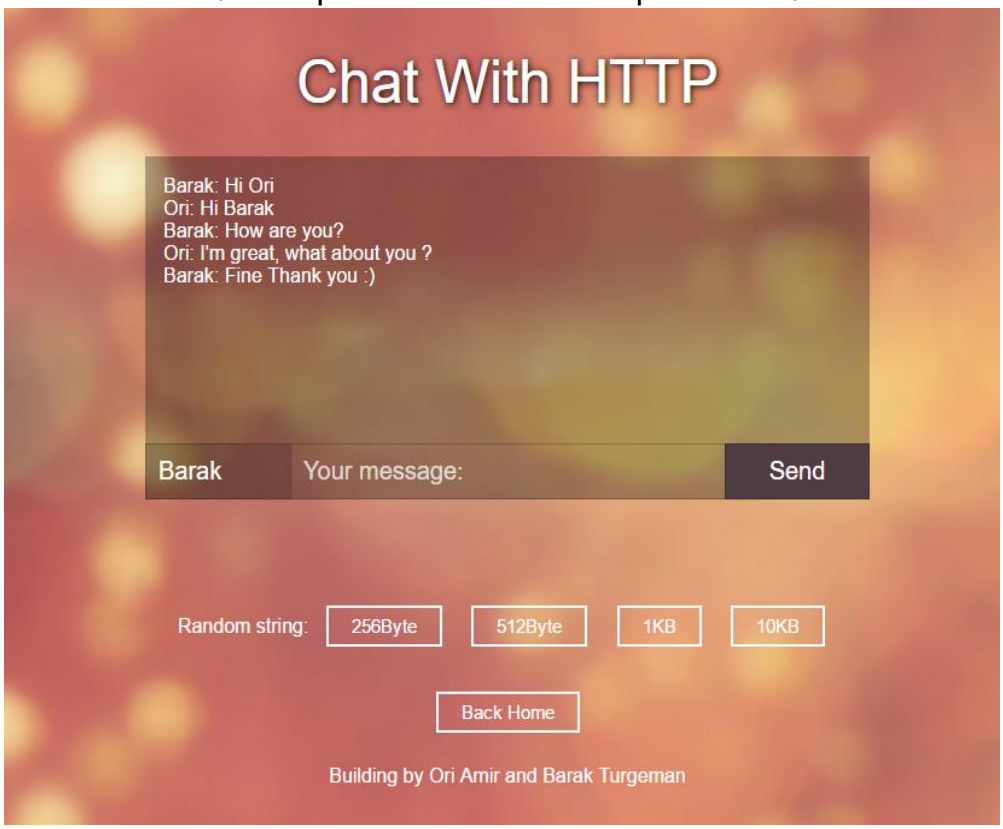
1. בנינו אפליקציית צ'ט המבוססת על הפרוטוקול HTTP. (הסבר נוסף בהמשך)
2. בנינו אפליקציית צ'ט המבוססת על הפרוטוקול Web Socket. (הסבר נוסף בהמשך)
3. האפליקציות מאוחסנות על שרת חזק לקבלת נתונים אמינים.
4. Wireshark – השתמשנו בתוכנה על מנת לנתח את הפקטות שנשלחו והתקבלו:
   א. בדיקת מספר פקטות
   ב. בדיקת מספר ביטים
   ג. בדיקת זמנים

איך בנינו את אפליקציות הצ'ט

אפליקציית צ'ט מבוססת על פרוטוקול **HTTP**:

כתובת האפליקציה: http://www.twix.co.il/http

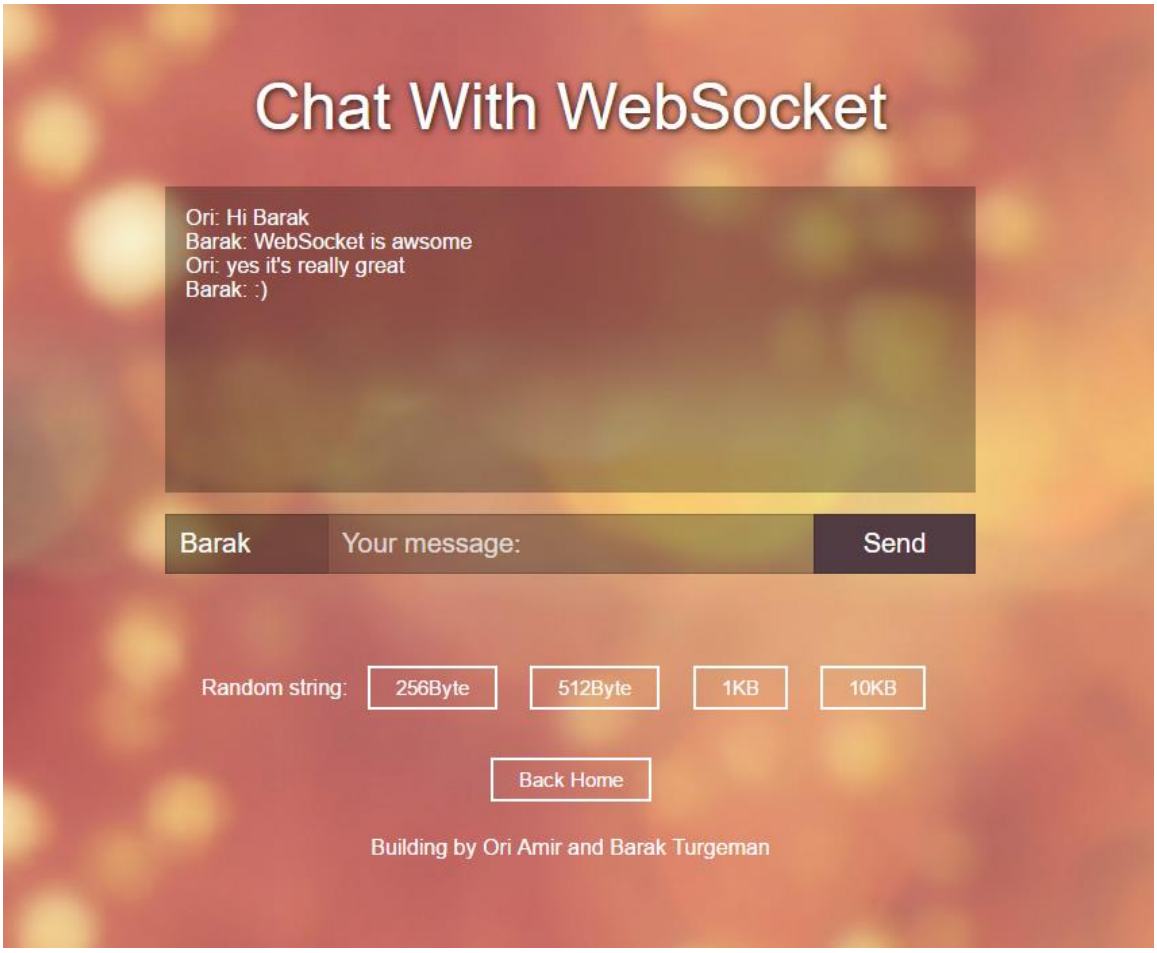
- האפליקציה בנויה בשפות: HTML5, JS ,PHP.
- באפליקציה מוצג חלון ובו השיחה וטופס לשליחת הודעה.
- בלחיצה על שלח מתבצעת שליחת ההודעה לשרת באמצעות AJAX.
- ברקע מתבצעת בקשת GET באמצעות AJAX לשרת ומתקבלת תשובה מהשרת רק לאחר שהתקבלה הודעה חדשה. ומיד לאחר קבלת ההודעה החדשה מתבצעת פעולת AJAX חדשה.
- הוספנו אופציה מיוחדת על מנת לבצע את הניסויים ובה ניתן לשלוח הודעה בגודל מסוים וזאת על מנת לבדוק את ההבדלים גם לפי אורך ההודעה שנשלחת.



אפליקציית צ'ט מבוססת על פרוטוקול **Web Socket**

כתובת האפליקציה: http://www.twix.co.il/webSocket

- האפליקציה בנויה בשפות: HTML5, JS ,NodeJS.
- פתחנו פורט מיוחד על השרת בשביל הWeb Socket. (ws://5.100.253.198:1337)
- התקנו על השרת את אפליקציית ה- NodeJS והגדרנו שהיא מאזינה על פורט 1337.
- באפליקציה מוצג חלון ובו השיחה וטופס לשליחת הודעה.
- מתבצעת התחברות באמצעות web Socket לשרת.
- שליחת הודעות מתבצעת באמצעות פונקציית send של web Socket.
- הודעה חדשה שמתקבלת באמצעות web Socket מתווספת לחלון השיחה
- הוספנו אופציה מיוחדת על מנת לבצע את הניסויים ובה ניתן לשלוח הודעה בגודל מסוים וזאת על מנת לבדוק את ההבדלים גם לפי אורך ההודעה שנשלחת.



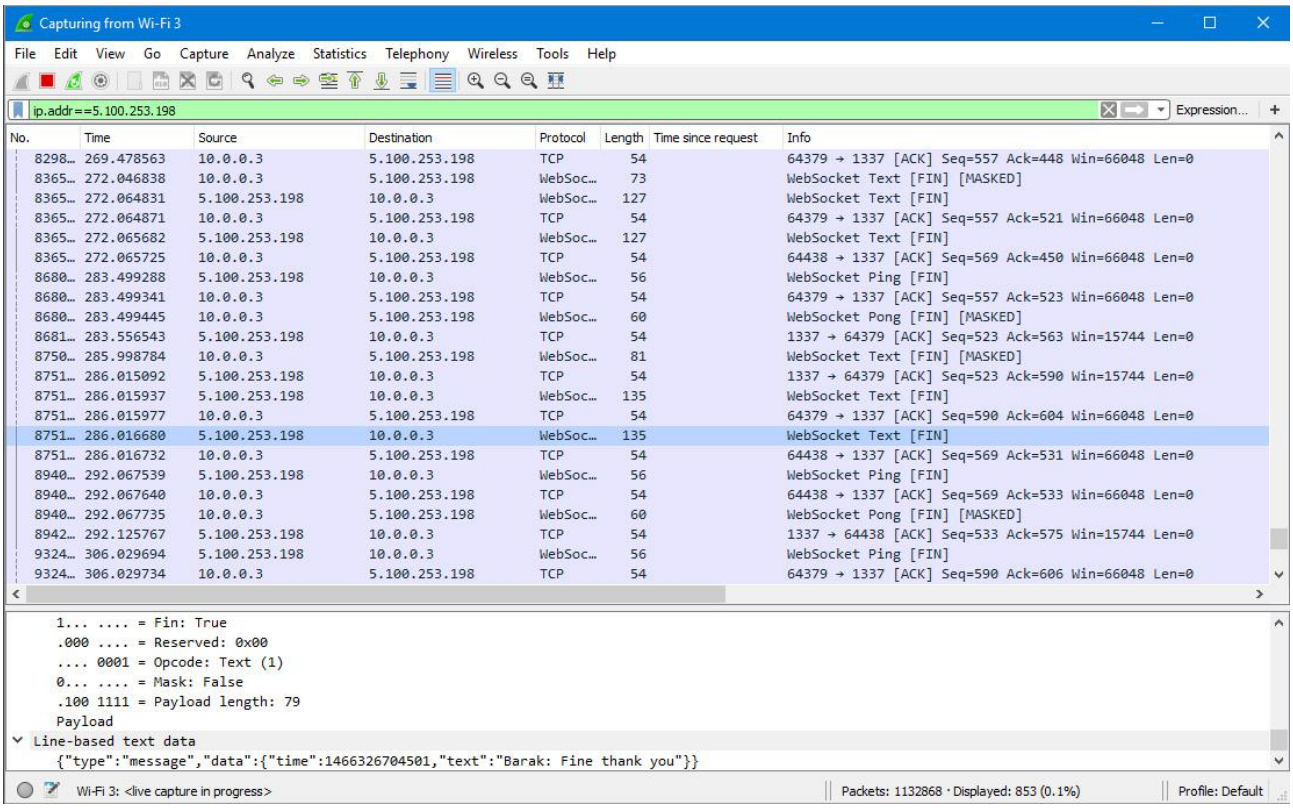
הניסויים שביצענו

ביצענו את הניסויים בעזרת כלי שמחולל מחרוזות בגדלים שונים על מנת לבחון את השפעת גודל ההודעה במקביל להשפעת הפרוטוקול שנבחן.
הניסוי נעשה במחשב נייח המריץ את מערכת ההפעלה Windows 10 ומחובר לרשת באמצעות חיבור אלחוטי במהירות 50 מגה בייט.
עבור 2 הפרוטוקולים HTTP ו- Web Socket ביצענו את המדידות הבאות בעזרת התוכנה Wireshark :

1. השוואת מספר הפקטות שנשלחו בכל פרוטוקול ביחס לגודל החבילה שנשלחה.
2. השוואת מספר הביטים שנשלחו בכל פרוטוקול ביחס לגודל החבילה שנשלחה.
3. השוואת הזמן הכולל מרגע השליחה ועד הקבלה בכל פרוטוקול ביחד לגודל החבילה שנשלחה.

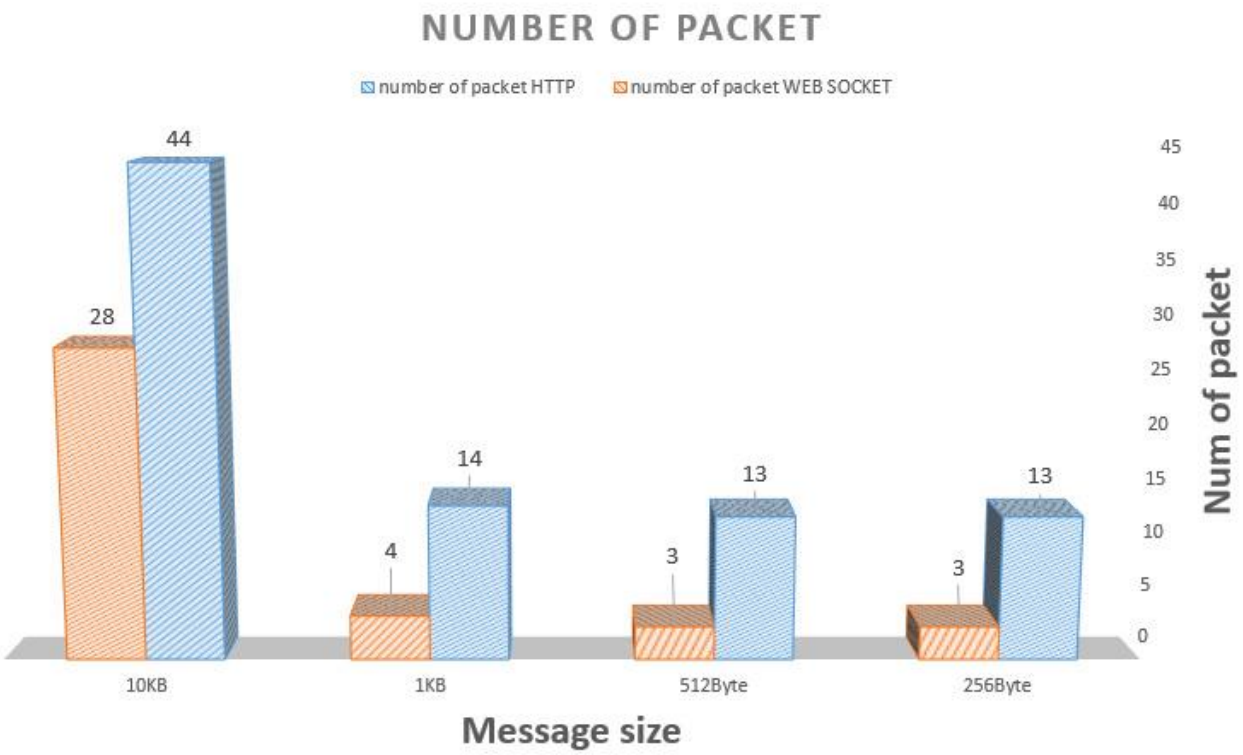
תוצאות צפויות

אנו מצפים לקבל שאפליקציית הצ׳ט שעובדת על בסיס פרוטוקול Web Socket תעבוד בצורה יעילה ומהירה יותר מאשר האפליקציה שעובדת על בסיס פרוטוקול HTTP .
כלומר , נצפה לראות שבפרוטוקול ה Web Socket :
1.פקטות- מספר הפקטות שישלחו יהיה קטן יותר .
2.ביטים -מספר הביטים שישלחו יהיה קטן יותר.
3. זמנים – נצפה לשיפור גדול יחסית בזמן הכולל שלוקח לחבילה להישלח ולהגיע ליעדה .



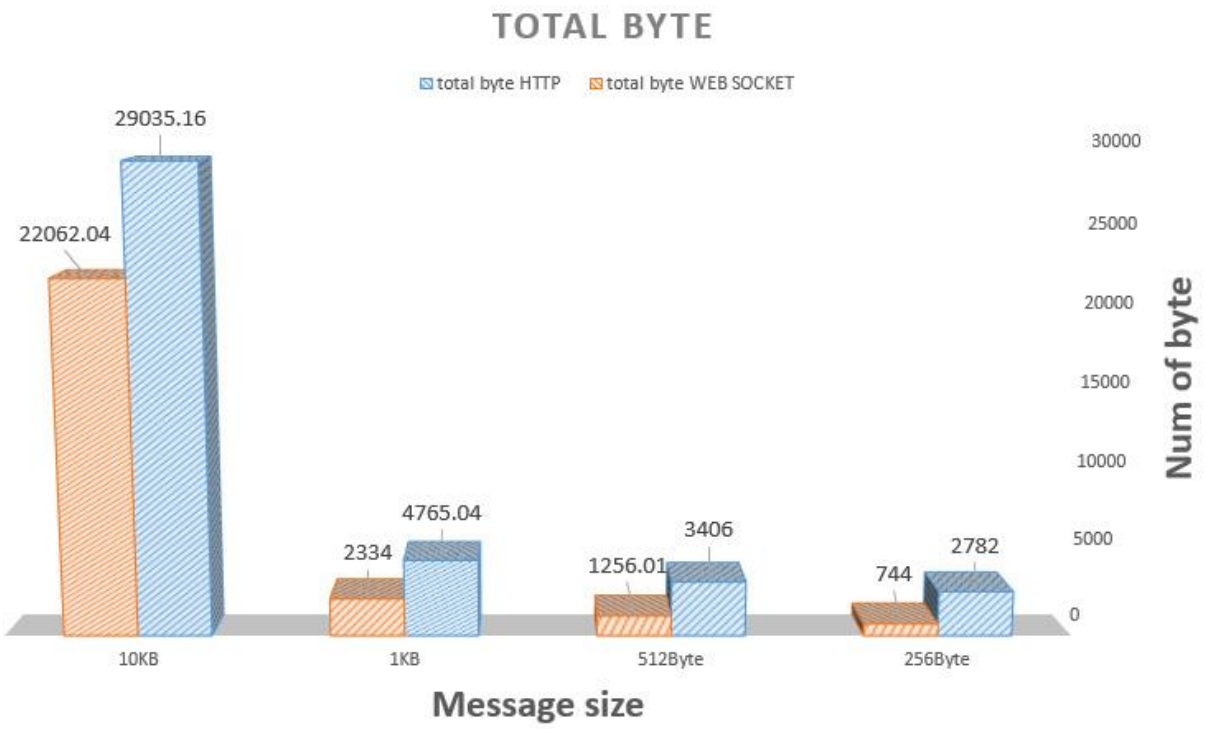
תוצאות שהתקבלו

1. בניסוי זה מדדנו את מספר הפקטות שנשלחו ביחס לגודל החבילה הנבחרת בשני האפליקציות השונות שכל אחת בנויה על פרוטוקול אחר.
ייצרנו 4 גדלים של חבילות בעזרת מחולל המחרוזות. גדלי החבילות שנבחנו היו:
256Byte, 512Byte , 1KB, 10KB ובכל פעם נשלחה חבילה בודדה ונבחנה בנפרד.



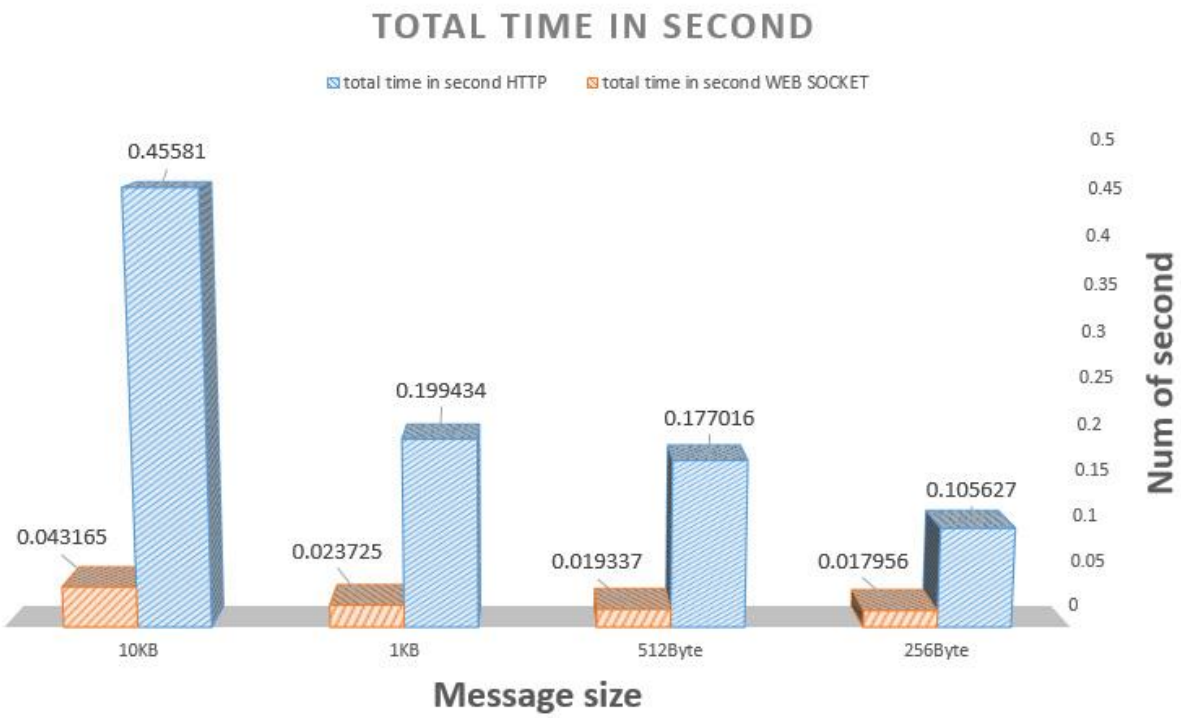
מניתוח הגרף ניתן לראות בבירור כי מספר הפקטות הנשלחות בפרוטוקול ה Web Socketקטן משמעותית מאשר מאשר אלה שנשלחות בפרוטוקול הHTTP , וזה יהיה יותר ויותר משמעותי כאשר גודל החבילה שתשלח יהיה גדול יותר.
בפרוטוקול ה HTTP נשלחות פקטות נוספות של של הקמת קשר, סיום קשר, בדיקה האם הקשר חי וכו׳ ולכן מספר הפקטות בפרוטוקול זה הוא גדול בהרבה מאשר ממספר הפקטות שנשלחות ב- Web Socket שם החיבור נשאר פתוח בין 2 הצדדים.

2. בניסוי זה מדדנו את מספר ה-byte שנשלחו ביחס לגודל החבילה הנבחרת בשני האפליקציות השונות שכל אחת בנויה על פרוטוקול אחר.
ייצרנו 4 גדלים של חבילות בעזרת מחולל המחרוזות. גדלי החבילות שנבחנו היו:
10KB ,1KB , 512Byte, 256Byte ובכל פעם נשלחה חבילה בודדה ונבחנה בנפרד.



מניתוח הגרף ניתן לראות בבירור כי מספר ה byte שנשלח בפרוטוקול ה- Web Socket קטן משמעותית מאלה שנשלחו בפרוטוקול ה- HTTP . ככל שמספר המשתמשים בצ׳ט ילך ויגדל כך גם מספר ההודעות שנשלחות בצ׳ט יגדל וכך מספר ה-byte שישלחו באפליקציה יגדל , במקרה כזה כמות ה- byte שתישלח כבר תהיה גדולה ומשמעותית בהרבה ולכן בחירה בפרוטוקול ה Web Socket תהיה חכמה יותר.

3. בניסוי זה מדדנו את הזמן הכולל בשניות שלוקח לחבילה להגיע מהרגע שנשלחה ועד שהגיעה ליעד וזאת ביחס לגודל החבילה הנבחרת . הזמן נמדד ב2 האפליקציות, האחת שבנויה על בסיס HTTP והשנייה על בסיס Web Socket.
ייצרנו 4 גדלים של חבילות בעזרת מחולל המחרוזות. גדלי החבילות שנבחנו היו:
256Byte, 512Byte , 1KB, 10KB ובכל פעם נשלחה חבילה בודדה ונבחנה בנפרד.



מניתוח הגרף ניתן לראות בבירור כי הזמן שלוקח להודעה להגיע ליעדה בפרוטוקול Web Socket מהיר בערך פי 10 מאשר הזמן שלוקח להודעה להגיע ליעדה בפרוטוקול ה-HTTP. זה מתחיל להיות ויותר משמעותי בחבילות אשר גודלן נהיה יותר ויותר גדול שכן הזמנים כבר מתחילים לגדול לשניות ולא נשארים מינוריים.

מסקנות

לאחר מחקר מעמיק בנושא ועריכה של מספר גדול של ניסויים ממוקדים הצלחנו להגיע למסקנות לגבי ההבדלים בין הפרוטוקולים HTTP ו Web Socket באפליקציית הצ׳ט :

1. אפליקציית הצ׳ט שעובדת על בסיס הפרוטוקול של ה- Web Socket עובדת מהר הרבה יותר מאשר זאת שבנויה על HTTP בכמעט פי 10,שזהו נתון מדהים. בימינו אחד הדברים העיקריים והחשובים בכל אפליקציה ובפרט באפליקציית צ׳ט הוא הזמן ובעניין זה אין עוררין לגבי זה שאם נרצה לבנות אפליקציית צ׳ט מהירה אנו נבחר בפרוטוקול ה- Web Socket .

2. מבחינת כמות הפקטות שנשלחות ישנו יתרון ברור לפרוטוקול ה Web Socket שכן כמות הפקטות הנשלחות בו היא קטנה מאלה שנשלחות בפרוטוקול ה- HTTP , מה שכמובן גורם לנו לחיסכון בתעבורה ברשת ולפחות חבילות שנשלחות בכל שליחת הודעה.

3. מבחינת כמות הביטים שנשלחים גם כאן יש יתרון ברור לפרוטוקול ה Web Socket שכן כמות הביטים הנשלחת בו היא קטנה יותר , מה שגורם לפחות עבודה לשרת ולפחות תעבורה ברשת.

4. **המסקנה העיקרית** שהגענו בסופו של דבר הוא שכדאי ואף רצוי לפתח אפליקציית צ׳ט בפרוטוקול ה Web Socket כיוון שהוא עובד בשיטה של צינור תקשורת שנשאר פתוח בין 2 המשתמשים ולכן הוא יעיל ומהיר בהרבה מפרוטוקול ה- HTTP המיושן המבוסס על בקשה-תגובה . כיוון שפרוטוקול הHTTP מבוסס על עיקרון זה העברת המסרים בין 2 המשתמשים הוא איטי וכבד יותר וזאת ראינו בניסויים.

ספרות

מתוך אתר Google scholar : http://www.file.scirp.org/Html/1-23101_25428.htm

Table 1. Data table used in test.

	Number of packets		Number of bits		Time (second)	
	HTTP	Web Socket	HTTP	Web Socket	HTTP	Web Socket
Client to server	83	5	33,662	372		
Server to client	77	8	45,600	7456		
Total	160	13	79,262	7828	~2.5	~0.25