DOI 10.34185/1562-9945-3-122-2019-03
UDC 004.931

O.O. Kavats, A.O. Kostenko

ANALYSIS OF CONNECTION METHODS OF TELEGRAM ROBOTS WITH SERVER PART

Abstract. The paper analyzes the methods of interaction of robotic applications with Telegram servers. A comparison was made between the standard polling method (Long Polling) and Webhook, both from the speed of application interaction with the end user and the complexity of the installation from the point of view of the developer. The interaction mechanism of telegrams-bot with Webhook-enabled telegram servers, which significantly improves the performance of the program as a whole, saving the user's query execution time and increasing fault tolerance.
The purpose of the study is to compare the methods of interaction between the application-work Telegrams written in Python, as well as the implementation of these methods in practice, in order to identify the complexity of writing both solutions.
The Webhook method is a way to deliver real-time data to applications. Unlike traditional APIs, where you need to specify data more often to get information in real time, Webhook sends data immediately.
It is proposed to consider the two most common communication options work and the Telegram server. The most common option is to periodically poll the Telegram servers for new information. All this is done through Long Polling, that is, the use opens for a short time and all updates immediately arrive bot.
In the work, an alternative communication option was proposed for the application to work with Telegram servers using Webhook. During the work on changing the data exchange method from standard polling (Long Polling) to Webhook, its indisputable advantage in loaded applications, namely on the number of incoming requests over a thousand, was proved (Long Polling).
Keywords. Telegram, Polling, Webhook, pyTelegramBotAPI, Python, Long Polling, API, testing, performance, SSL certificate.

Formulation of the problem. Sooner or later, any software developer faces the problem of expanding his product, as well as writing software code for telegram bots, and this direction is no exception. In the transition from testing the program to a full product developer, the developer may face the effect of the "narrow bottle neck," which role is played by the method of Telegram-bot and telegram servers' communication called "Long Polling", which queues the requests of the application users, since the process of Telegram-bot and Telegram servers' communication is periodic in the survey of servers for the incoming messages [1]. This system is suitable for use in unloaded applications or applications with a small number of users (no more than 1000 simultaneous interactions) [2], as well as for debugging and testing the program. In this paper, the mechanism of Telegram-bot with Telegram servers interaction with support of Webhook will be considered, which significantly raises the overall program performance, saving the user's request time and increasing fault tolerance. In this paper, we shall consider these methods in more detail on specific examples and draw conclusions for how the use of these methods affects the work of applications.

Analysis of recent research. Webhook is a way to deliver real-time application data. Unlike traditional APIs, where you often need data to get real-time information, Webhook sends data immediately. Webhook is a kind of analogue of push messages on your mobile phone. Instead of burning the battery on your phone, extracting information (polling) from applications to receive updates, Webhook sends data automatically based on events triggers. Like push messages, Webhook is less resource-intensive.

Webhook is much more effective than polls in terms of resources and communication. Zapier conducted a study of 30 million requests for surveys done through their service, and found that 98.5% of the polls were wasted and they spent 66 times more resources on the poll [3].

Always fresh data. The very nature of Webhook and the fact that they are usually triggered by events means that they provide you with information in real-time. In this regard, if you want the information to be as close as possible to the real time, you should choose the use of Webhook instead of the survey (Long Polling) [4].

Webhooks surpass the survey in terms of data relevance, communication efficiency, and infrastructure costs.

Formulating the goals of the article. The purpose of the study is to compare the methods of interaction of the application Telegram work written in the language Python, as well as the implementation of these methods in practice, in order to identify the complexity of writing the two solutions.

Main part. In the course of work, both methods of trying the robot program to use the Telegram servers: Long Polling and Webhook - were used, the basic schema of which is shown in Figure 1.

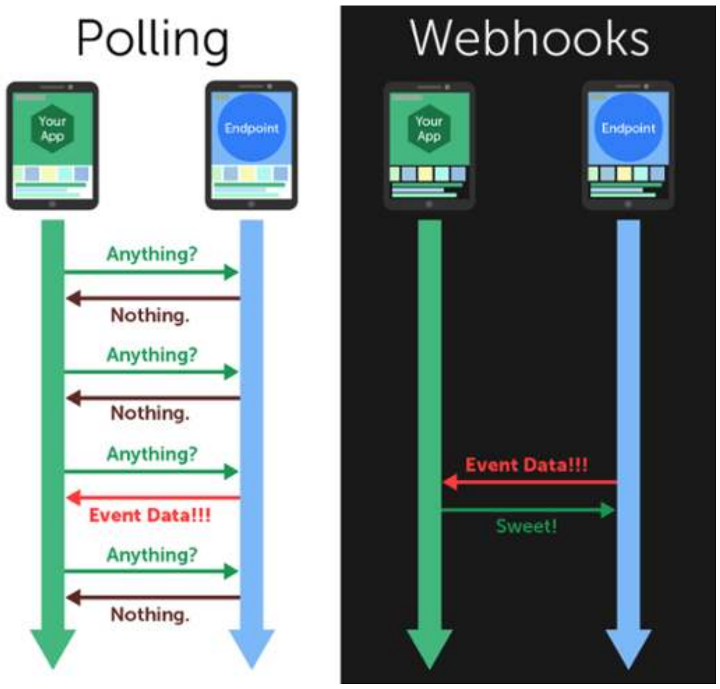

Figure 1 - Schematic diagram of Long Polling and Webhook.k

The paper considers two of the most common variants of communication between the robot and the Telegram server. The first and easiest option is to periodically poll Telegram servers for the availability of new information. All this is done through Long Polling, that is, the connection is opened for a short time and all updates are immediately flown by the bot. It is simple, but not very reliable. First, Telegram servers periodically begin to return the 504 (Gateway Timeout) error, which makes some bots stingy that even the pyTelegramBotAPI library created to write telegrams in Python can not always survive the following [5].

Second, if several bots are running at the same time, the probability of encountering errors increases. It's a shame if the bots themselves are not used very often [5].

Webhook works a bit differently. By installing Webhook, you would say to Telegram servers: "If anyone sends me write here - (link)". There is no need to periodically poll servers themselves, thus, the unpleasant reason for the fall of search engines' spamming disappears. However, for this you have to install a full web server on the machine on which it is planned to launch Telegram robots. What's even unpleasant, you need to have your own SSL certificate, because Webhook in Telegram works only on HTTPS [6] Fortunately, on one glorious day, support for self-signed certificates has appeared.

The ultimate goal of any API integration is to effectively exchange data between the application and the server in order to provide more performance for your users. To speed up the program's work, integration should provide a method for detecting changes as well as events occurring in the user's application. Currently, two of the most popular event management tools are the Long Polling (Web Polling) and web hooks (Webhook). Let's consider these two methods in more detail, on the example of the simple application of the telegrams "Schedule of classes", which is based on the reference to the MySQL database, written in Python, using the interface of the Telegram program by the end user. During the writing of the bot, the standard method of interviewing Telegram servers using the API was used, but during the testing of the program, immediately revealed the negative aspects of such interaction between the program and servers, namely, the rate of response of the program to the end-user request, and with increasing number of requests and users (testers of the application), the response rate for incoming requests was significantly reduced, thereby forcing the search field to be a productive method of communicating with the servers; it became a method called Webhook [7]. Trend of increased speed of the application on a user's query is shown in figure 1.

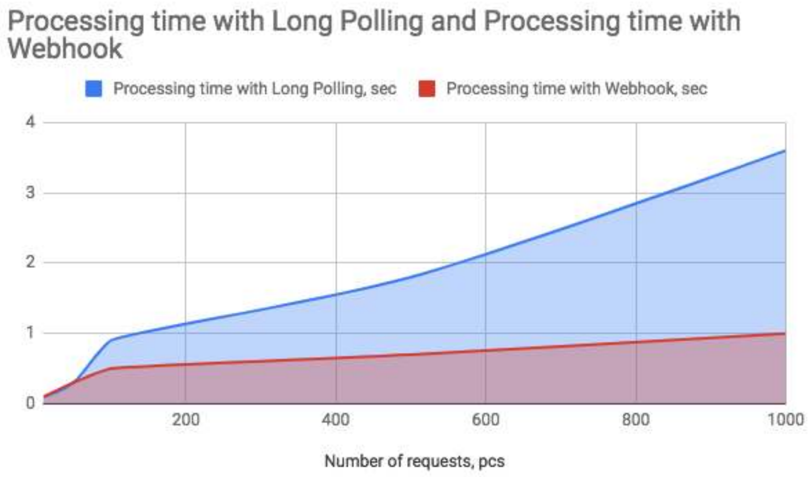

Chart 1 - The trend of increasing the response rate
of the application to the user's request

To study this particular questioning of the servers, ten testers of the Schedule of Occupations application were used, during the testing, the data was collected without using and using Webhook. The essence of the test was the simultaneous loading created by the testers in addition which had artificially removed the mechanisms of protection against DDoS attacks, namely, the limits of access to the same requests set in a certain unit of time were removed. After the experiment, the results of measurements were put in Table 1, then the data from Table 1 was plotted 1.

Table 1

Measurement of the peak load created by the testers in addition to the «Schedule of occupations»

Number of requests, pcs	10	50	100	500	1000
Processing time with Long Polling, sec	0,1	0,3	0,9	1,8	3,6
Processing time with Webhook, sec	0,1	0,3	0,5	0,7	1

Conclusions. In this paper, an alternative version of the robot application and Telegram servers' communicating with the help of Webhook was suggested. In the process of changing the method of data exchange from the standard poll (Long Polling) to Webhook was proved its undeniable advantage in loaded applications, namely on the number of inquiries more than a thousand, Webhook showed three times more performance than the standard poll (Long Polling) This undoubtedly justifies its more labor-intensive debugging process (SSL certificates, etc.).

REFERENCES

1. Downey, Allen B. (May 2012). Think Python: How to Think Like a Computer Scientist (Version 1.6.6 ed.). ISBN 978-0-521-72596-5.
2. "RFC6202 - Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP".
3. Server-Push Documents (HTML & XHTML: The Definitive Guide) Archived 2008-04-17 at the Wayback Machine. O'Reilly book explaining server-push.
4. "What is push notifications & How do push notifications work?". www.push-maze.com. Retrieved 2018-10-25.
5. M. Thomson, E. Damaggio and B. Raymor (October 22, 2016). "Generic Event Delivery Using HTTP Push". Internet Draft. Internet Engineering Task Force. Retrieved October 28,2016.
6. Lutz, Mark (2013). Learning Python (5thed.). O'Reilly Media. ISBN 978-0-596-15806-4.
7. Pilgrim, Mark (2009). Dive Into Python 3. Apress. ISBN 978-1-4302-2415-0. Archived from the original on 2011-10-17.

Received 06.02.2019.
Accepted 12.02.2019.

Анализ методов связи роботов Телеграм с серверной частью

В работе проведен анализ методов взаимодействия приложений-роботов с серверами Телеграмм. Проведено сравнение стандартного метода опроса (Long Polling) и Webhook как со стороны скорости взаимодействия приложения с конечным пользователем, так и трудо-емкости установление с точки зрения разработчика. Механизм взаимодействия телеграмм-бота с серверами телеграмм с поддержкой Webhook, что существенно поднимает производительность программы в целом, экономя время выполнения запроса пользователя и поднимая отказоустойчивость.

Целью исследования является сравнение методов взаимодействия приложения-работа Телеграмм, написанного на языке Python, а также реализация этих методов на практике, с целью выявления трудоемкости написания обоих решений.

Webhook - это способ доставки данных в реальном времени приложений. В отличие от традиционных API, где вам нужно чаще указывают данные, чтобы получить информацию в режиме реального времени, Webhook отправляют данные немедленно.

Предлагается рассмотрение двух наиболее распространенных варианта связи работа и сервера Телеграмм. Наиболее распространенный вариант заключается в периодическом опросе серверов Телеграмм на предмет наличия новой информации. Все это осуществляется через Long Polling, то есть открывается употребление непродолжительное время и все обновления тут же прилетают боту.

В работе был предложен альтернативный вариант общения приложении работу с серверами Телеграмм, с помощью Webhook. В ходе работы по изменению метода обмена данными со стандартного опроса (Long Polling) на Webhook было доказано его неоспоримое преимущество в нагруженных приложениях, а именно на количестве входящих запросов более тысячи, Webhook показал в три раза больше производительность по сравнению со стандартным опросом (Long Polling).

Аналіз методів зв'язку роботів Телеграм з серверної частиною

В роботі проведено аналіз методів взаємодії додатків-роботів з серверами Телеграм. Проведено порівняння стандартного методу опитування (Long Polling) і Webhook як з боку швидкості взаємодії додатка з кінцевим користувачем, так і трудомісткості налагодження з точки зору розробника. Механізм взаємодії телеграм-бота з серверами телеграм з підтримкою Webhook, що істотно піднімає продуктивність програми в цілому, економлячи час виконання запиту користувача і піднімаючи відмовостійкість.

Метою дослідження є порівняння методів взаємодії додатка-робота Телеграм, написаного на мові Python, а також реалізація цих методів на практиці, з метою виявлення трудомісткості написання обох рішень.

Метод Webhook - це спосіб доставки даних в реальному часі додатків. На відміну від традиційних API, де вам потрібно найчастіше вказують дані, щоб отримати інформацію в режимі реального часу, Webhook відправляють дані негайно.

Пропонується розгляд двох найбільш поширених варіанта зв'язку робота та сервера Телеграм. Найбільш розповсюджений варіант полягає в періодичному опитуванні серверів Телеграм на предмет наявності нової інформації. Все це здійснюється через Long Polling, тобто відкривається єднання на нетривалий час і всі оновлення тут же прилітають боту.

У роботі був запропонований альтернативний варіант спілкування додатку роботу з серверами Телеграм, за допомогою Webhook. В ході роботи по зміні методу обміну даними зі стандартного опитування (Long Polling) на Webhook було доведено його незаперечну перевагу в навантажених додатках, а саме на кількості вхідних запитів більше тисячі, Webhook показав в три рази більше продуктивність в порівнянні зі стандартним опитуванням (Long Polling).

Кавац А.А. – к.т.н., доцент, Национальной металлургической академии Украины.
Костенко А.А. – магистр, Национальной металлургической академии Украины.

Кавац О.О. - к.т.н., доцент, Національної металургійної академії України.
Костенко А.О. – магістр, Національної металургійної академії України.

Kavats E.A. - Candidate of Technical Sciences, Associate Professor, National Metallurgical Academy of Ukraine.
Kostenko A.A. - the Master National Metallurgical Academy of Ukraine.