

# Response Time Analysis of Mobile Application DNUN in New Relic Monitoring Platform

Karthik Reddy Nalla, Hosam El-Ocla\*

Department of Computer Science, Lakehead University, Thunder Bay, Canada

Copyright©2016 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

**Abstract** In this paper we present a case study using New Relic as a monitoring platform to analyse the performance of mobile applications. DNUN (Danger Notification and User Navigation) is the application assumed to be evaluated. DNUN is associated with a geolocation system to navigate to the location of an object for immediate/later use from anywhere on the globe. We use the response time as the metric that Application Performance Management uses to measure the reliability of the DNUN application. We install DNUN app into the New Relic program using gem and configuration files. The dashboard proves that the overall rating of the app is excellent based on the response time performance.

**Keywords** Response, Mobile, Application, Performance, New Relic

---

## 1. Introduction

The mobile applications (apps) market place is quite competitive all over the modern world particularly for young generation. Apps are getting increasing popularity owing to the evolution in the mobile technology and the huge growing in the number of their users. To guarantee a highly performance of their apps, designers should measure and provide data on how their app is behaving in congested areas. Challenges are augmented by the resource limitations ingrained in the wide range networks such as WILDNeT [1].

Studying the behaviour performance of the mobile app is a demanding task for its service users. In a previous work, authors have developed a new mobile app DNUN (Danger Notification and User Navigation) aimed to provide a rescue and navigation software using the geolocation process [2, 3]. DNUN provides an integration of mobile sensors and social network as a rescue plan that can be used in several military and commercial applications [4]. Also, DNUN is an additional application that supports the mobility of the smartphone users. Reliability of DNUN is quite important particularly for those users who are in danger situations.

Application Performance Management (APM) tools, such as Dynatrace, Quest PerformaSure, AppDynamics, and CA APM provide a more sophisticated view on the monitored system [5]. APM strives are to detect and diagnose apps performance problems to maintain an expected level of service.

New Relic is an APM solution primarily developed to monitor apps running in cloud, on-premise, or hybrid environments. The best part of the New Relic APM solution is that it is compatible with languages and frameworks such as PHP, Java, Ruby, Rails, .NET, Python, and Play2.0, which is not limited and can be extended for Drupal, Magento, CakePHP, Joomla, and so on [6].

We use the New Relic software as a monitoring platform to analyse the performance of DNUN. Response time in DNUN is a crucial factor in saving those individuals in danger. As a result, we work on studying and analyzing this factor to improve the DNUN design, if needed, through reducing the response time to its minimal. New Relic delivers real-time and crucial information about the performance of the web app from the minute that user starts browsing until the page completes its loading. After the user sign up for an account and install the New Relic program (known as agent) in the server, it begins to monitor and collect all network and database activities. Response time [7] is the metric that APM uses to measure the reliability of DNUN.

## 2. Methodology

We can add the app to New Relic by installing the agents on the server having DNUN as shown in figure 1. New Relic provides different agents for different programming languages. Since we developed the app using Ruby on Rails system [8, 9], we use New Relic Ruby agent in DNUN to measure its performance. Newrelic\_rpm is a gem which acts as a New Relic Ruby agent. This gem is added to the gem file of the app. New Relic provides a configuration file newrelic.yml, which we download and place in the config sub-directory of the app. Next, we change the default app

name into a meaningful name in the downloaded file [10]. Re-launch the app server and we can see the performance of the app in the dashboard. The dashboard gives us the basic information about the selected app including Apdex score, throughput (requests per minute), web transactions, and error rate [11]. In this paper, we focus on Apdex score and web transactions.

It should be noted that the response time is the time taken by a website server to respond to a user's or visitor's request [7]. *Apdex Score* is a measure of the response time based against a set threshold  $T$ . It measures the ratio of satisfactory to unsatisfactory response times referring to response time ranges defined by Nielsen [12]. The response time is measured from an asset request to completed delivery, and back to the requestor. All responses handled within  $T$  should satisfy the user. Apdex tracks three response counts [13]:

- Satisfied: The response time is less than or equal to  $T$ .
- Tolerating: The response time is greater than  $T$  and less than or equal to  $4T$ .
- Frustrated: The response time is greater than  $4T$ .

Apdex score [13, 14] is a ratio value of the number of satisfied and tolerating requests to the total requests made. Each satisfied request counts as one request, while each tolerating request counts as half as satisfied request.

Web transactions are the http requests given by the user to the app server. Header information of the http request includes: request origin, domain name of the server, content type, cookies, etc.

New Relic provides performance parameters of DNUN app including web transactions response time, throughput, and browser page load time.

Web transaction response is the frame received from the app server for a given request. The information contained in the response is the connection status, type, range, and length of the frame's content, etc.

### 3. Results and Discussion

Web Transaction response time is the time taken by the app server to respond to web transactions or requests made by the user. The response time of the app is shown in figure 1. The response time for each request can be obtained by adding the total time spent by both the request and response in all the internal and external components of the application server. These components include the Ruby language, rack

middleware, database, and the external web. The response times in figure 1 is calculated based on the user activity over a period of 30 minutes. On average, the response time of the app server is about 50 milliseconds (ms) and of the browser is about 12.9 seconds. The user has performed some activities during that time period such as login, adding contacts, sending danger notifications, saving locations, navigating, etc. Figure 1 shows the response time of all components of web transactions and display them in a single graph.

The response time for each component of the server is shown in the figures below where we discuss as:

1. **Middleware:** Rack is a kind of middleware that acts as a filter used to intercept a request and modify the response as a request has triggered an application. It handles all of the server-specific application program interface (API) calls, passes on the HTTP request and all the environment parameters in a hash, and gives the app's response back to the server. The total time spent by the request and response in the middleware is shown in figure 2. When a client connects, it takes a time in the range of tens ms before any data is received. It is better if the app could feed the client the service promptly so he/she can download external resources. However, the rack response time is generally satisfactory as it represents a normal distribution function at different ranges of times.
2. **Ruby:** Ruby is the core program of the app server. Figure 3 shows the total time spent in the ruby code. The total time is the time spent by the request, the response, and the return back to the middleware which redirects the response to the browser. The request's response is when the ruby code is executed, i.e., routing to the right controller as per the user request who calls the related view and/or model. Here, the response time is also satisfactory as it represents a multiple normal distributions but with shorter times. It is expected that the range of response times for ruby calls is shorter compared to the rack. As it is well known that the rack sits between the web app and the web server, the response time with a ruby call should be longer for the time spent in the rack request interception. However, this observation is sometimes not valid as it happens that the rack request would have a shorter response time than the ruby call.

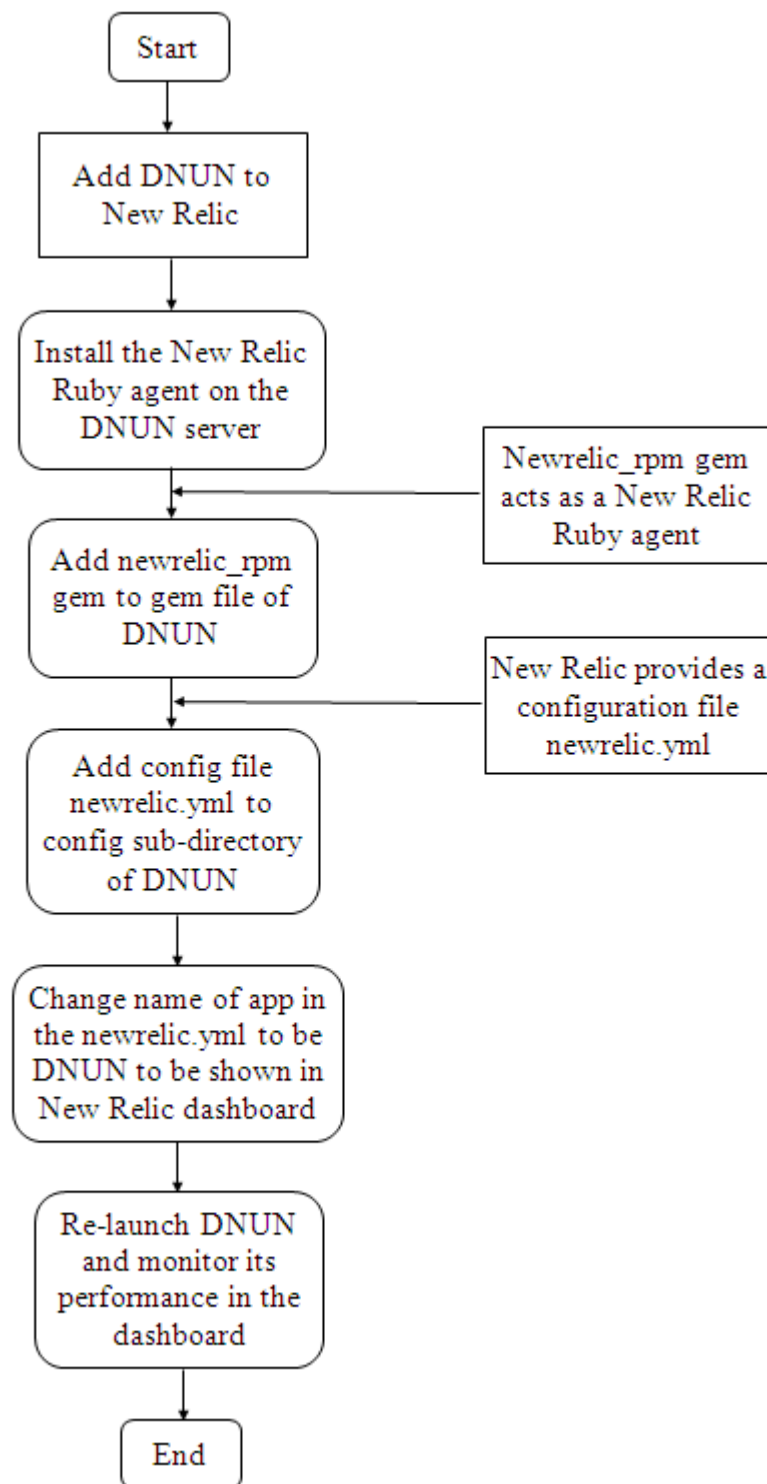


Figure 1. Flowchart showing New Relic installation

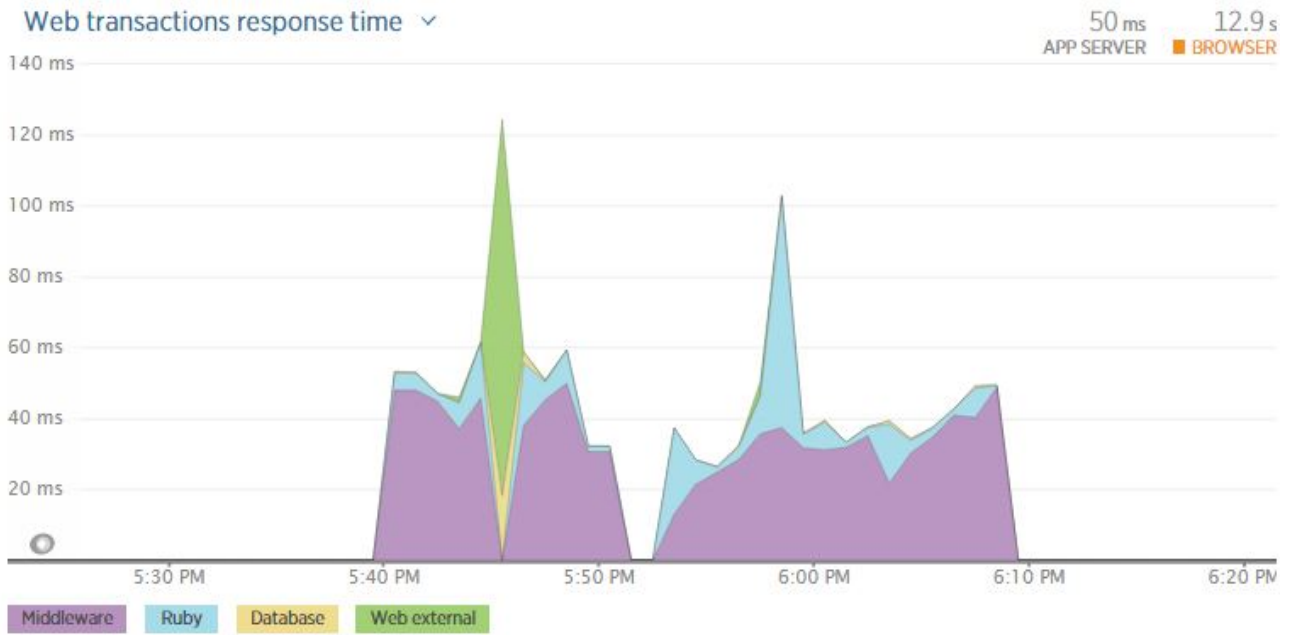


Figure 2. Web Transaction Response Time

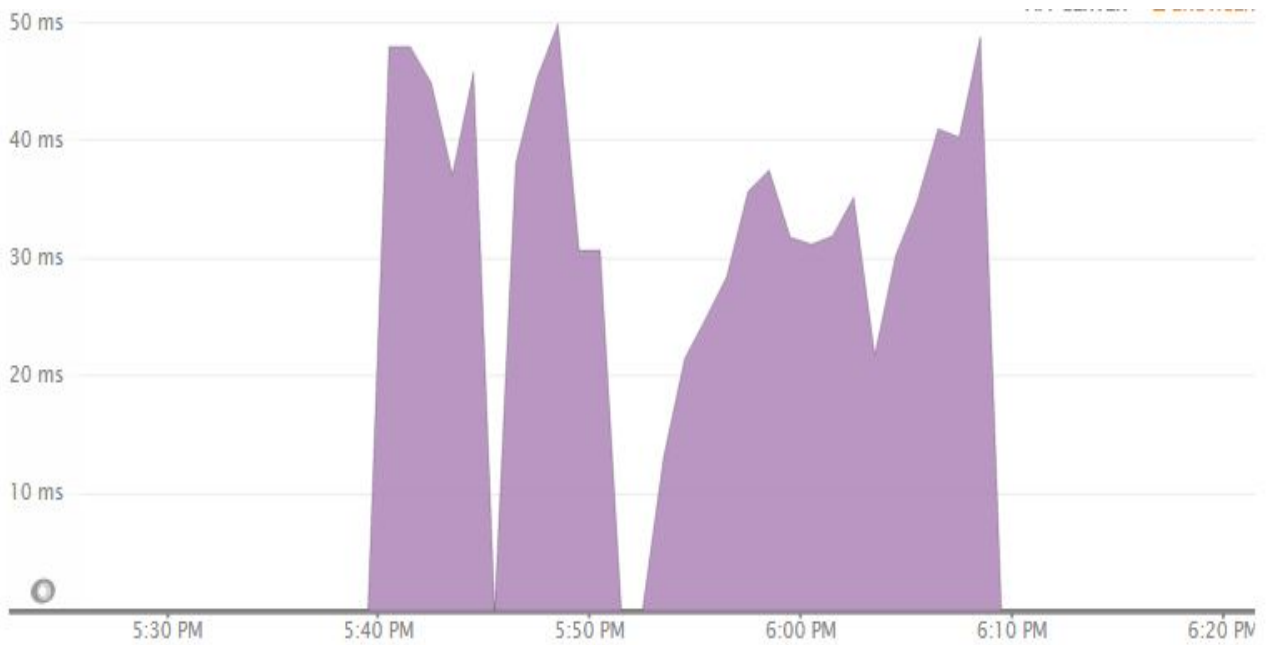


Figure 3. Response Time of Rack (Middleware)

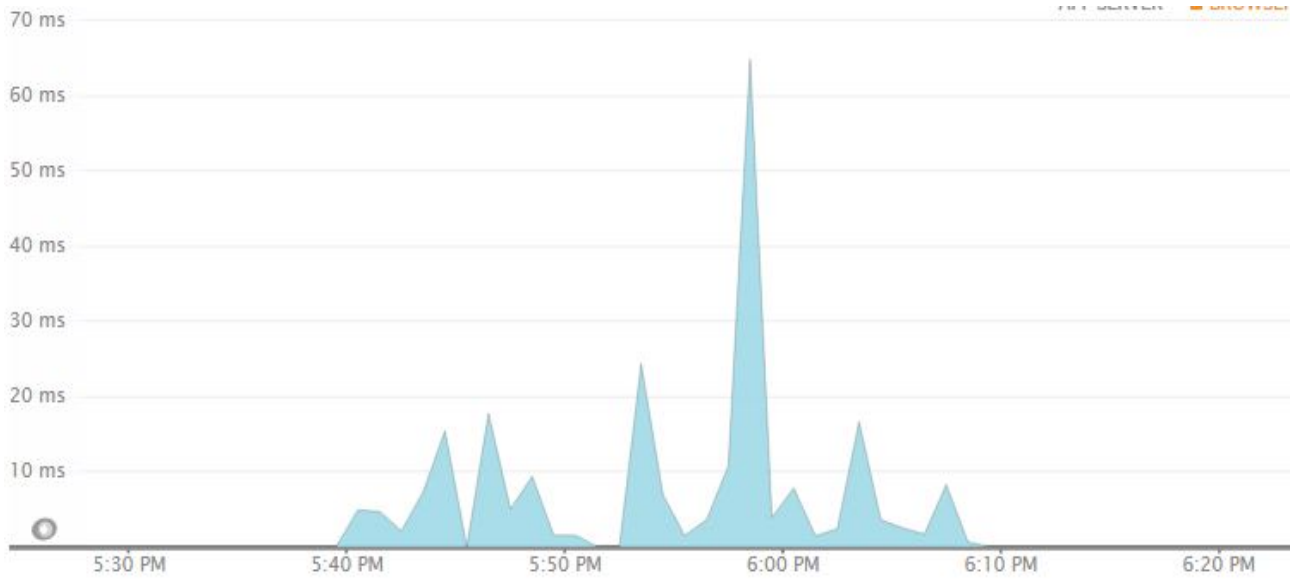


Figure 4. Response Time of Ruby

3. Database: Most of the requests will have something to be inserted or retrieved from the database. Database plays a key role in storing the data. The total time spent in accessing the database, depending on the request, is shown in figure 4. Response time has also here a normal distribution and it is shorter than the previous two cases. This is because that the list of user's contacts stored by DNUN is not a big database and as a result the database access response time is relatively short.

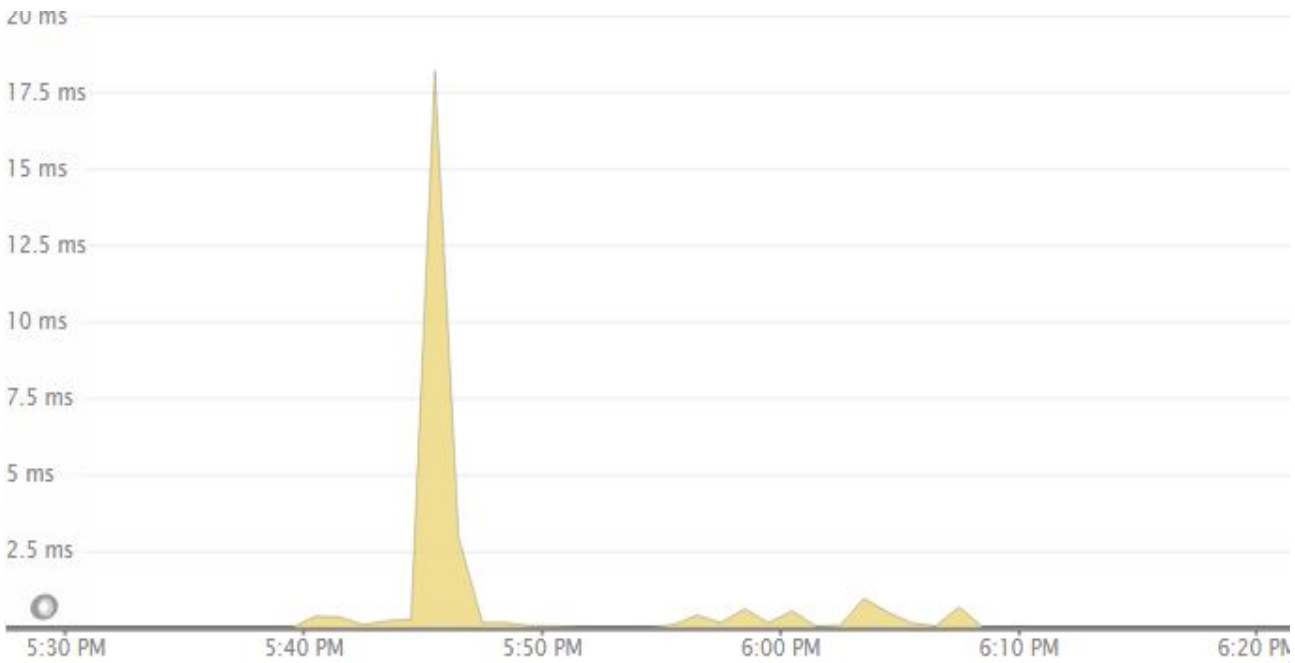


Figure 5. Response Time of the Database



Figure 6. Response Time of External Web

4. External web: This is the Internet access and the total time spent in the external web is shown in figure 5. Some APIs in the app such as Google Maps needs the Internet connectivity to display maps. The response time here is expectedly the longest compared to the above cases as the queuing time combined with the HttpDispatcher time would be obviously longer.

## 4. Conclusions

The New Relic APM generates a periodic report for DNUN application based on the user activity. According to the report, the overall rating of the app is *excellent* based on the response time performance. The apdex score of the DNUN app server is 0.95. This indicates that users of the app are never frustrated with the app loading time. The average response time of the app is about 83.7 milliseconds, which is very small compared to other apps. This reflects that DNUN is available with a short response time which is needed particularly in danger situations where DNUN is designed for.

## REFERENCES

- [1] Rabin Patra, Sergiu Nedeveschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, Eric Brewer, "WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks", Proc. Of 4th USENIX Symposium on Networked Systems Design & Implementation, pp. 87–100, Cambridge, MA, USA, 2007.
- [2] Ionescu, Daniel. "Geolocation 101: How It Works, the Apps, and Your Privacy". PCWorld. Retrieved March 29, 2010.
- [3] Jump up, Kevin F. King, "Geolocation and Federalism on the Internet: Cutting Internet Gambling's Gordian Knot", COLUM. SCIENCE & TECH. LAW REV., 41, 2010.
- [4] K. Nalla and H. El-Ocla, "Mobile DNUN: Danger Notification and User Navigation in Mobile Devices", IEEE IT Professional, (to appear).
- [5] Tilmann Rabl, Mohammad Sadoghi, Hans-Arno Jacobsen, "Solving Big Data Challenges for Enterprise Application Performance Management", Proc. of the VLDB Endowment, Vol. 5, No. 12, pp. 1724-1735, 2012.
- [6] Surendra Mohan, Administrating Solr, Packt Publishing, 2013.
- [7] M. Joseph and P. Pandya, "Finding response times in a realtime system", The Computer Journal (British Computer Society), Vol. 29, No. 5, pp. 390–395, 1986.
- [8] Wendy Liu, "Ruby on Rails", Jesse Garrett's Article, pp. 4-9, Fall 2007.
- [9] Rick Borup, "An Introduction to Ruby and Rails", Southwest Fox conference in Gilbert, Arizona in October, pp. 50-55, 2010.
- [10] Ruby agent installation, 2008–16 New Relic, Inc.
- [11] Getting started with New Relic: A newbie's guide to building killer apps with great performance, New Relic, Inc, 2013.
- [12] Jakob Nielsen, "User interface directions for the Web", Communications of the ACM, Vol. 42, No. 1, pp. 65-72, 1999.
- [13] Peter Sevcik, Defining The Application Performance Index, Business Commun. Review, pp. 8-10, 2005.
- [14] Peter Sevcik, Service Level Agreements for Business-Critical Applications, NetForecast Report 5091, NetForecast Inc., 2008.