Recommendations for Resilient App Development

Leonardo Corbalán[1[0000-0001-9026-8059]], Juan Fernández Sosa[1[0000-0002-0482-3392]], Fernando Tesone[1[0000-0001-9499-3127]], Sebastián Dapoto[1[0000-0001-7593-0198]], Federico Cristina[1[0000-0003-3838-417X]], Pablo Thomas[1[0000-0001-9861-987X]] and Patricia Pesado[1[0000-0003-0000-3482]]

[1] Computer Science Research Institute LIDI (III-LIDI)
School of Computer Science, National University of La Plata,
La Plata, Buenos Aires, Argentina
{corbalan, jfernandez, ftesone, sdapoto, fcristina, pthomas, ppesado}@lidi.info.unlp.edu.ar

Abstract. Applications for mobile devices tend to face a greater number of adverse situations derived from the very nature of mobility. In this context, improving resilience becomes relevant. Here we present some recommendations, based on preliminary studies, to increase resilience to the performance drop and power consumption increase caused by increased computational load.

Keywords: Resilient Mobile Applications, Performance, Battery Consumption, 3D Mobile Applications, Mobile Applications with High Computational Load.

1 Introduction

Etymologically, the term resilience comes from the Latin *resilīre*, which means 'to jump back, to bounce back', 'to withdraw' [1]. Applied to different areas of knowledge and human activities, this term has been redefined in different ways, incorporating nuances specific to each discipline. In biology, it refers to the capacity of a living being to adapt to a disturbing agent or adverse situation. In physics, it describes the capacity of a material, mechanism or system to recover its initial state when the perturbation to which it had been subjected has ceased. In psychology, resilience is the capacity of a person to adapt positively to adverse situations, overcoming the trauma they may have caused.

In engineering disciplines, resilience is understood as the capacity of a system to resist and recover from a failure in some critical component in order to maintain its operation and services in an acceptable manner [2]. The amount of recovered functionality and the time invested in the recovery are relevant parameters to determine the degree of resilience of a system in a given adverse situation. These statements also apply to software systems.

Software failures can arise from unforeseen situations, adverse circumstances, random events or malicious attacks. In the latter case, the ability of a system to withstand a cyber-attack is known as cyber resilience. Regardless of the origin of the failures, resilient software will be able to recover to a degree and period of time previously determined according to the requirements and particularities of the system.

In the case of software applications for mobile devices (apps), the resilience analysis must consider the very nature of mobility. This confers certain distinctive features that expose apps to a set of potentially adverse situations that are not present in other types of applications.

In this short paper we present a set of preliminary results from which we derive a series of recommendations for the development of apps that are resilient to the drop in performance and the increase in energy consumption caused by the increase in computational load.

The rest of this paper is organized as follows: section 2 describes adverse scenarios and strategies to improve app resilience, section 3 presents some results on app performance and energy consumption, section 4 presents a series of recommendations to increase app resilience, and finally, section 5 presents conclusions and future work.

2 Resilience in Apps

The development of applications for mobile devices must consider how to increase the degree of resilience of apps in order to cope with a greater number of stress situations to which they are often subjected. Battery drain, total or partial loss of connectivity that comes with mobility, increased risk of cyber attacks, partly due to the typical behavior of mobile device users, are just a few examples of such stress situations.

It is possible to recommend the implementation of a series of strategies to increase the degree of resilience depending on the type of adversity to which an app may be subjected.

2.1 Resilience of Apps to Connection Loss

It is common for apps to have a strong dependency on Internet connection to access and communicate with services deployed in the cloud. To build resilient apps, connection failures should not be considered exceptional and the offline experience should be considered as just another state of the application. In this way, when a connection is lost, the app will continue to operate with some limitation, but avoiding crashes, blank screens or alert messages that generate frustration and abandonment by users [3].

Upon loss of connection, the app must be able to capture the user's intent, executing the required action as far as possible and completing it later when the connection has been re-established. This type of design is referred to as "offline-first" solutions [4].

While there is no connection to the network, in the absence of the most recent data, the local memory of the device will provide the data obtained in the last stable connection. PWA, Progressive Web Applications, are mobile web applications that use this methodology to generate an offline experience. For this purpose, web technologies such as Service Workers and caching, among others, are used [5].

2.2 Resilience of Apps in the Face of Connection Instability

In case the connection is unstable, resilient behavior can be achieved by implementing two well-known design patterns: the Retry Pattern [6] and the Throttling Pattern [7].

Retry Pattern allows handling transient failures in the connection to a network service by retrying a failed operation. This strategy is based on the assumption of two premises: i) failures are of short duration and ii) the previously failed request may succeed in a subsequent attempt. It is necessary to define the maximum number of attempts to be made and the waiting time between each of them, being able to opt for a fixed or incremental waiting time.

On the other hand, Throttling Pattern consists of relaxing some requirements to reduce the demand for resources while avoiding compromising response times or the overall performance of the app. This allows the system to continue to function, in a suboptimal manner, in the face of difficulty in fully accessing a resource. Some actions to take are: limit the number of requests per user, keep essential services in the first instance and prioritize the distribution of requests.

2.3 Resilience of Apps against Cyber Attacks

Resilience to cyber attacks is known as cyber resilience. It is defined as the ability to anticipate, resist, recover from, and adapt to attacks or compromises in systems that use cyber resources [8]. Compromised systems must absorb the impacts of attacks, and respond quickly and flexibly to ensure the operational continuity of their critical components.

In recent years, mobile devices have become an essential tool for the operation and activities of companies, but, at the same time, they represent a new medium through which criminals can attack and access a company's resources. Malware attacks, social engineering, access to insecure wireless networks, weak passwords, incorrect use or careless user behavior are just some of the potential risks associated with the use of mobile devices in companies. Loss or theft of the device is also a security issue that can facilitate access to company resources by unauthorized third parties.

Some recommendations for the development of cyber resilient mobile apps [9] are: i) web services communication should use SSL/TLS (Secure Sockets Layer / Transport Layer Security), ii) URLs launched by an application should use the HTTPS protocol, iii) encrypt the information handled by the application, iv) the number of login attempts should be limited and v) development frameworks should be updated.

2.4 Resilience of Apps to Increased Computational Load

A high computational load increases energy demand and affects battery life. In addition, depending on the computational capacity of the device, it can delay app response time, which is perceived as a performance drop that degrades the user experience. As an example, consider 3D mobile applications whose processing demands can easily exceed the capacity of the devices on which they run.

In order to increase resilience to performance loss and rapid battery depletion, it is necessary to monitor response speed and power consumption in real time. In this way, measures can be taken to reduce the computational requirement (Throttling Pattern), extending the device's battery life and improving the app's response time.

3 Study of Performance and Power Consumption in Apps with High Computational Load

For this analysis it is necessary to consider the development frameworks used because they impact the efficiency with which the apps consume or use the available resources. The performance and power consumption of apps built with the following development frameworks were studied: Android SDK (native), Cordova, Titanium, NativeScript, Xamarin, and Corona. [10]. The best response times and most efficient power consumption, during processing-intensive tests, were obtained by applications developed with Cordova, Titanium, and NativeScript (see Table 1).

Table 1. Intensive processing app

Framework	Power (mWh)		CPU charge (%)		Duration (s)	
	$\overline{E}$	S_E	$\overline{C}$	S_C	$\overline{T}$	S_T
Cordova	1.597	0.136	35.924	2.571	8.467	0.679
Titanium	1.692	0.096	37.480	2.395	8.355	0.643
NativeScript	1.792	0.176	33.357	2.217	9.109	1.789
Xamarin	3.036	0.185	32.072	1.768	17.891	0.973
Android SDK	3.463	0.149	32.468	1.332	18.568	2.938
Corona	7.304	0.189	44.347	54.793	38.877	1.492

With respect to 3D applications for mobile devices, 2 of the most currently used development frameworks were evaluated: Unity and Unreal Engine [11]. The characteristics of a 3D application that have the greatest impact on performance and energy consumption on a mobile device were identified. It was observed that applications developed with Unity consume less energy, but at the same time present a greater performance degradation, compared to Unreal Engine, when the complexity of the scene to be visualized increases drastically (see Table 2).

Clearly, the factors that most affect performance and power consumption in both frameworks are the display of complex objects (high number of polygons) and the simultaneous use of different particle systems (e.g. smoke, sparks, explosion, etc.).

Table 2. Performance and consumption of 3D mobile applications

Scene characteristics	Performance as scene complexity increases of the scene with increasing number of objects (FPS)		Consumption (mAh)	
	Unity	Unreal	Unity	Unreal
Simple objects	60 60 60 60 60 47 16 04	60 61 60 58 40 28 10 08	19.8	51.53
Complex objects	40 34 17 08 05 05 04 04	08 08 08 08 08 08 08 08	27.18	61.06
Lights and shadows	60 60 60 60 59 40 15 04	60 60 60 59 42 30 10 08	20.73	37.4
Textures	60 60 60 60 60 47 13 04	60 60 61 56 37 29 12 08	20.23	47.48
Particles	60 60 46 14 04 03 00 00	19 19 16 11 08 08 08 08	15.94	21.78
Physics	60 60 60 60 60 45 03 03	60 60 60 58 42 30 10 08	18.36	18.65

4 Recommendations. Preliminary Results

A series of recommendations are presented here to improve the resilience of apps to performance drops (longer response times) and increased energy consumption caused by increased computational load.

The first aspect that must be taken into account to improve resilience is the app's ability to offer resistance to the adversity it faces. For this purpose, development frameworks should be used to obtain apps that better manage energy consumption and computational load.

It is recommended to use the multiplatform development frameworks Cordova, Titanium and NativeScript for general-purpose applications with high computational load.

For the development of 3D applications with medium or low complexity scenes, it is recommended to use Unity. Unreal Engine can process more powerful and realistic graphics than Unity but with higher power consumption. For the case of very complex scenes, Unreal Engine should be considered because the performance of apps generated with this framework does not degrade as much as the performance of apps developed with Unity.

The second important aspect to improve resilience is to increase the capacity to recover from a stressful situation. In this sense, an effective strategy is to use the Throttling Pattern. For this it is necessary to monitor the response speed and energy consumption in real time. In 3D applications this can be achieved by reducing the complexity of the scene to be visualized. According to the data analyzed, and due to the impact they have on performance and energy consumption, it is recommended to reduce some or several of these factors that affect the complexity of the scene: i) Number of polygons in the scene, ii) particle systems to visualize, iii) dynamic lights and shadows of the objects and iv) complex textures or materials of the objects.

5 Conclusions and Future Work

In this short paper we have analyzed, from a resilience perspective, some experiments on the impact of development frameworks on the energy efficiency of the apps built.

From this analysis, a series of recommendations have been made to improve the resilience of apps to the performance drop and energy consumption increase caused by the increased computational load. Recommendations were made for the development of general-purpose applications and for 3D mobile applications.

As future work, we plan to extend the experiments to other frameworks, and also to study the behavior of the developed apps in the event of partial or total loss of connectivity. We also intend to incorporate the analysis of using local DBs with cloud synchronization to mitigate the loss of connectivity.

Finally, we will seek to expand the analysis of resilience aspects in 3D Apps, from the incorporation of Virtual Reality (VR).

References

1. Real Academia Española. Resiliencia. En Diccionario de la lengua española, 23.ª ed., versión en línea. Recuperado de https://dle.rae.es/resiliencia.
2. Murray, A., Mejias, M., & Keiller, P. (2017). Resilience methods within the software development cycle. In Proceedings of the International Conference on Software Engineering Research and Practice (SERP) (pp. 62-65). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
3. Lazar, J., Jones, A., Hackley, M., & Shneiderman, B. (2006). Severity and impact of computer user frustration: A comparison of student and workplace users. Interacting with Computers, 18(2), 187-207.
4. Vanhala, J. (2017). Implementing an Offline First Web Application.
5. Aguirre, V., Delía, L., Thomas, P., Corbalán, L., Cáseres, G., & Sosa, J. F. (2019, October). PWA and TWA: Recent Development Trends. In Argentine Congress of Computer Science (pp. 205-214). Springer, Cham.
6. Retry pattern - Azure Architecture Center | Microsoft Docs https://docs.microsoft.com/en-us/azure/architecture/patterns/retry.
7. Throttling pattern - Azure Architecture Center | Microsoft Docs https://docs.microsoft.com/en-us/azure/architecture/patterns/throttling
8. Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, R. (2021). Developing cyber resilient systems: a systems security engineering approach (No. NIST Special Publication (SP) 800-160 Vol. 2). National Institute of Standards and Technology. available free of charge from: https://doi.org/10.6028/NIST.SP.800-160v2r1
9. Jaramillo, H. D., Romero, G. K., & Ramos, C. (2021). Framework de seguridad para desarrollo de aplicaciones móviles y su aporte a la CiberResiliencia. Revista Ibérica de Sistemas e Tecnologias de Informação, (E42), 452-468.
10. Corbalán, L., Thomas, P., Delía, L., Cáseres, G., Sosa, J. F., Tesone, F., & Pesado, P. (2019, June). A study of non-functional requirements in apps for mobile devices. In Conference on Cloud Computing and Big Data (pp. 125-136). Springer, Cham.
11. Cristina, F., Dapoto, S. H., Thomas, P. J., Pesado, P. M., Perez Altamirano, J., & De la Canal Erbetta, M. (2020). Aplicaciones Móviles 3D: un estudio comparativo de performance y consumo de energía. In XXVI Congreso Argentino de Ciencias de la Computación (CACIC)(Modalidad virtual, 5 al 9 de octubre de 2020).