A Comparison of Data Serialization Formats For Optimal Efficiency on a Mobile Platform

Audie Sumaray
Christopher Newport University
Newport News, Virginia, USA
audiesumaray@gmail.com

S. Kami Makki
Lamar University
Beaumont, Texas, USA
kami.makki@lamar.edu

ABSTRACT

Because of the increase in easily obtainable internet-connected mobile devices and their unique characteristics, choosing the proper data serialization format has become increasingly difficult. These devices are resource scarce and bandwidth limited. In this paper, we compare four different data serialization formats with an emphasis on serialization speed, data size, and usability. The selected serialization formats include XML, JSON, Thrift, and ProtoBuf. XML and JSON are the most well known text-based data formats, while ProtoBuf and Thrift are relatively new binary serialization formats. These data serialization formats are tested on an Android device using both text-heavy and number-heavy objects.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: General – *Distributed applications.*

General Terms

Design, Performance, Reliability

Keywords

Android, Dalvik, Data serialization, JSON, ProtoBuf, Thrift, XML.

1. INTRODUCTION

With modern advances in mobile computing, phones have evolved from simple devices for sending and receiving calls into more advanced smart-phones. These phones are essentially web enabled mini-computers that are almost always able to connect to the Internet. A range of system platforms and application programming environments have been created for the smartphones like Apple's iPhone, RIM's Blackberry line, PalmOS, Windows Mobile and many well-developed device and product lines.

Google, with the other members of Open Handset Alliance, released an open source mobile software stack named Android that includes an operating system, middleware, and a number of other applications. Android supports a subset of Java API, and uses Java as its programming language. It has broad customization support, has built-in graphical user interface components, and comes with a set of core applications accessible by third-party developers [1, 2]. Android operating system is used in Smartphones like Motorola, HTC, Samsung, LG, Sony Ericsson, and many more.

1.1 Android platform

The Android platform is a software stack for mobile devices that includes an operating system, middleware, and number of applications [3, 15]. There are four levels in Android architecture as shown in Figure 1.

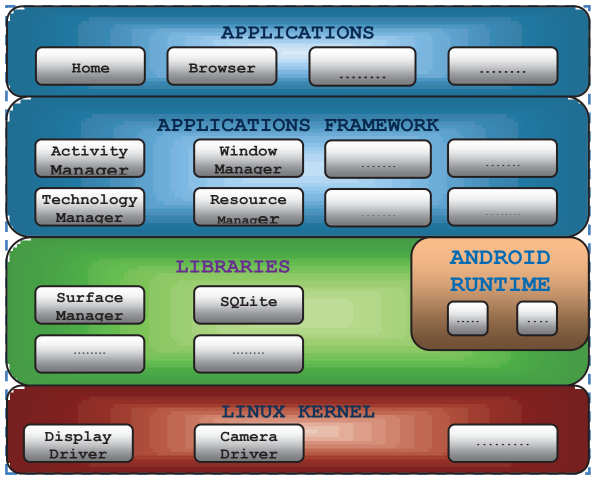

Figure 1 - Android System Architecture

Android uses Linux Kernel as a hardware abstraction layer and uses the Linux kernel's hardware drivers for memory management, networking, and managing processes. However, the Android applications will not make direct Linux calls; they always need to use the Android Virtual Machine (Dalvik)[1].

[1]This material is based upon work supported by the National Science Foundation under Grant No. 0851912. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This VM takes the generated Java class files and combines them into one or more executable files in order to be suitable for systems that are constrained in terms of memory and processor speed. (These executable files are identified by ".dex" extension and are called Dalvik executable file.)

Android includes a set of C/C++ libraries. These libraries are provided to the developers through the Android application framework. Some of the important libraries are surface manager, SQLite, 3D libraries, media codec, etc.

1.1.1 Applications and Widgets
In this layer, a number of applications are provided for the mobile device, such as email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written in Java. The overall system architecture is depicted in Figure 1.

1.2 Android Software Development Kit (SDK)
The Android SDK consists of several tools to help Android application development. This includes Eclipse IDE plug-in, Android emulator, debugging tools, visual layout builder, log monitor, and more.

The Android SDK also includes a mobile device emulator, a virtual mobile device that runs on the computer. The emulator lets us develop and test Android applications without using a physical device. When the emulator is running, we interact with the emulated mobile device just as we would do with an actual mobile device, except that we use our mouse pointer to touch the touch screen and can use some keyboard keys to invoke certain keys on the device. Figure 2 is the screen shot of the android emulator plug-in.

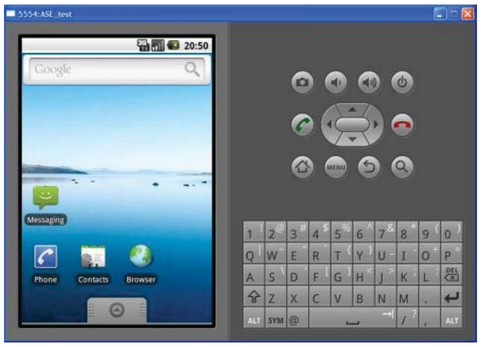

Figure 2 - Android Emulator

2. Data Serialization
There are many applications revolve around sending and receiving data over the network. The increase in data exchange over the Internet has made the selection of a proper data serialization format increasingly important.

Data serialization is the process of writing the state of an object to a stream, and is the process of rebuilding the stream back into an object [11]. The two most common modern data serialization formats are eXtensible Markup Language (XML) [7] and JavaScript Object Notation (JSON) [4]. They are both widely used and well documented, but were developed before the onset of smartphones. More recently, binary data serialization formats are being used in lieu of these common text-based formats. The two binary formats we will focus on in this paper are Protocol Buffers [10], which was developed and used by Google, and Apache Thrift [9], which was developed and used by Facebook. Both of these formats were developed to address shortcomings in XML and JSON.

2.1 XML
XML is a profile of Standard Generalized Markup Language (SGML) [7] that was developed by members of the W3C. It was designed to be easy to use over the Internet, supports a wide variety of applications, and is human-legible and easy to create [7]. It has a large user base, and is used widely in various web services. There is also a multitude of XML-based languages available. Despite its widespread use, XML is often criticized for being overly verbose and not well suited for mobile environments. In fact, the official XML specifications state that terseness in XML markup is of minimal importance [7].

2.2 JSON
JSON was developed by Douglas Crockford, who used it for several years before launching json.org. Soon afterwards both Yahoo and Google began to offer web services in JSON. JSON is often touted as a lightweight and a more efficient alternative to XML, as it does not have as much markup overhead. Recently, many companies are beginning to offer their web services in JSON, and it is often considered as the go-to language for mobile devices. However, JSON has a number of shortcomings, such as extensibility drawbacks, and its lack of namespace support and input validation [14].

2.3 Binary
While JSON and XML are the most widely used data serialization formats, binary format is quickly gaining momentum. The two binary formats that are currently growing very fast are Apache Thrift and Google's Protocol Buffers (ProtoBuf). Both were designed to be extremely lightweight, and fast to serialize and deserialize. Because both JSON and XML are text-based, they will always need to be parsed character by character, thus imposing a limit on deserialization speed. These binary formats are not subject to the same disadvantage as the JSON and XML formats. They both make use of 'positional binding' which allows the storing of the 'name' part of the name-value pairs in a separate file

('.proto' for ProtoBuf and '.thrift' for Thrift). These files do not have to be sent over the Internet, which can greatly decrease the size of the data to be communicated. Since these files have to be compiled before being included in a program, there are some restrictions based on what languages each protocol supports. ProtoBuf supports C++, JAVA, and Python [10], while Thrift supports C++, C#, Erlang, Haskell, Java, Objective C/Cocoa, OCaml, Perl, PHP, Python, Ruby, and Squeak [9]. One should note that because Apple iOS apps are programmed primarily in Objective C, ProtoBuf is not the best choice to use when programming for Apple iOS devices.

2.4 Mobile Serialization

Since the goal of this paper is to compare different data serialization formats on a mobile platform, it is necessary to consider the most important aspects for this environment, such as data size, serialization speed, and ease of use.

2.4.1 Data Size

Many mobile devices have limited amounts of internal storage, and may or may not have external storage. Because of this, serialized objects should use as little space as possible. A small serialized size is also important because many smartphone users have a limited amount of allotted data usage per month, so reducing the size of serialized objects helps to reduce the amount of data that has to be sent over the network. A reduced data size also has the benefit of decreasing the transfer time of the serialized object.

2.4.2 Serialization Speed

While mobile devices are steadily becoming more powerful, they still lack the processing speed of desktop PCs. Despite this fact, it is essential that the chosen data serialization format allows fast serialization and deserializion of an object. Speed is especially important on smartphones that emphasize the user interface, such as the Apple iPhone. Users of these devices expect smooth interaction with their phones, and waiting for serialization or deserialization to complete is not acceptable.

2.4.3 Ease of Use

With the popularity of application stores (app stores) on all the major smartphones, smartphones tend to attract a lot of new or non-programmers. This makes it important for a data serialization format to be relatively easy to implement. Due to the diversity of available web services and platforms, it is also important for a data serialization format to be interoperable with multiple web services and support multiple platforms.

3. Similar Work

There have been a number of papers that investigate data serializations, but to our knowledge none have focused on data serialization formats specifically in a mobile environment.

Kangasharju and Tarkoma propose binary XML as a good alternative to XML [12]. While binary XML offers a slight serialization speed and size improvement over the vanilla implementation of XML, it loses its interoperability with XML based web services, and it is no longer human readable.

Nurseitov et al., [14] compared XML and JSON formats for a client-server environment, and they showed that JSON is faster and uses fewer resources than XML for this environment.

Eriksson and Hallberg compared JSON with YAML (Yet Another Markup Language) [8]. While they showed that YAML is more functional and readable than JSON, JSON is much faster than YAML for both serialization and deserialization [13]. This fact, combined with the lack of web services using YAML, disqualifies YAML from being a recommended format for data serialization on mobile platforms.

The thrift-protobuf-compare project [6] is the most comprehensive comparison of data serializers to this date. This open source project currently has compared the speed of serialization and deserialization for the following data serializers: ProtoBuf, Thrift, Java, Scala, Stax, Binary XML, JSON, Xtream, Javolution, Hessian, Avro binary, JSON Marshaller, and Kryo. It should be noted that these comparisons deal with the actual serializers and parsers, and not individual data serialization formats. Although, many of these serializers use the same data format.

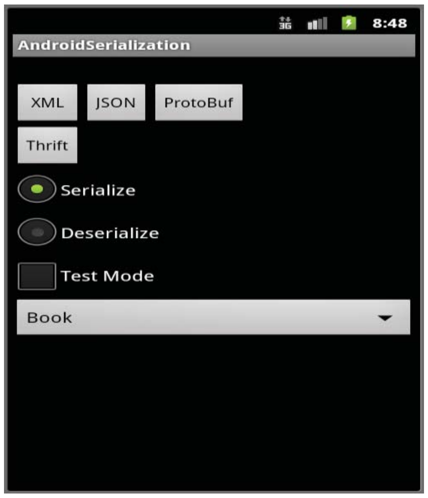

Figure 3: The Test Application

4. The Developed Test Application

In order to test each data serialization format in the most accurate and realistic way, it was necessary to develop an application to run on an Android smartphone. The goal of this application was threefold:

1. Accurately test both serialization and deserialization speed on an actual mobile device.
2. Allow easy batch testing, so that many iterations of each test can be performed quickly and easily.
3. Store all data and results to an easy to access location for future reference and for easy visualization.

As seen in Figure 3, the finished testing application contained a single button for each serialization format, two radio buttons to switch between data serialization and deserialization, and a check box to enable batch testing. When a test is performed, the results of each test are saved in a line-delimited format so that they are easily usable.

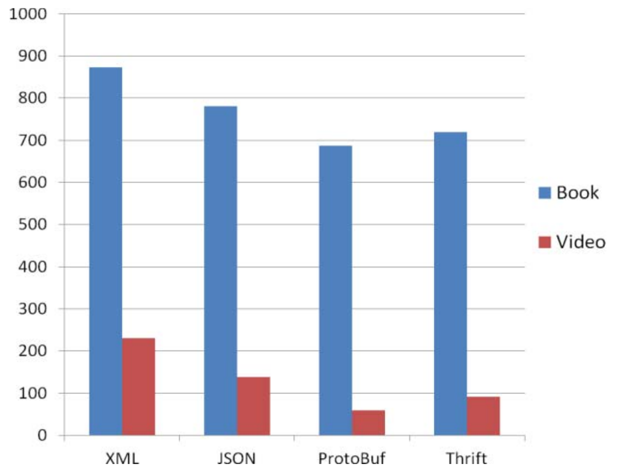

Figure 4: Serialized data size

5. Experiment and Results

5.1 Data and Serializer Specifics

There are two test objects used for this experiment as shown in Figure 4. The first object, "Book", is designed to mimic a book object that might be used in a book store application, and the structure is as follows:

```
public class Book {
  public enum Type{Paperback,
                   Hardcover,Digital
  }
  private  String title, author,
                  description, image;
  private  int  numInStock;
  private  double  price;
  private  Type  type;
  private  boolean  onSale;
  ..............
}
```

The Book object was designed to be heavy on text, and light on numerical values. The second object, "Video", was designed to mimic a video object that might be used with a streaming video website, such as YouTube, and is structured as follows:

```
public class Video {
  private String uri, title format;
  private int height, width, length,
              views;
  private Boolean enabled;
  ..............
}
```

This Video object was designed to be light on text and focuses on testing the efficiency of the serializers when dealing with an object that is made up of predominately numbers.

For XML serialization the Simple framework [5] was used, since it is considered to be both easy and fast. For JSON, the Streaming API contained in the Jackson framework [3] was used. Jackson is widely considered to be the fastest JSON serializer and parser available, which makes it the optimal choice for serialization on a mobile platform. The latest released versions of ProtoBuf and Thrift were used (2.4.1 and 0.6.1, respectively). The resulting serialized data from using XML or JSON is stored in text files. The serialized ProtoBuf or Thrift data is stored in a byte array.

5.2 Data Size Comparison

After the data was serialized into each format and stored in their respective file, the file sizes were compared. From Table 1, we can see that XML is the largest, followed by JSON, then Thrift, with ProtoBuf being the smallest. Because of their text-based nature, it is unsurprising that XML and JSON serialize larger than the binary formats.

Table 1: Serialized size in bytes

	XML	JSON	ProtoBuf	Thrift
Book	873	781	687	720
Video	231	139	59	92

The largest difference in serialization sizes becomes apparent when looking at the serialized video class. Since this class consists of more numbers than strings, the binary serializers were able to deliver extremely small serialized data.

5.3 Speed Comparison

Upon first running the serialization/ deserialization tests, it was found that all serialization times were slightly slower than expected, but when a second set of one-thousand iterations was completed, the times for all formats improved. For future tests, we performed two sets of serializations/deserializations directly after each other, and used the smaller of the two numbers.

Based on one-thousand iterations of both serialization and deserialization for each object type, data serialization is shown to be the costlier operation. On average, serialization takes more than double the amount of time it takes for deserialization.

Table 2: Average Serialization Time in ms

	XML	JSON	ProtoBuf	Thrift
Book	22.842	4.177	2.339	2.315
Video	17.884	4.097	1.800	1.747

Table 3: Average Deserialization Time in ms

	XML	JSON	ProtoBuf	Thrift
Book	7.908	1.199	0.298	0.732
Video	6.742	0.755	0.197	0.310

As shown in Table 2, XML is the slowest of the formats for serializing, and takes more than five times the amount of time to serialize than it takes the other formats. JSON is the second slowest format, but its serialization times are a massive improvement upon XML format. The binary formats (Thrift and ProtoBuf) are the fastest, taking roughly half the time required for JSON for their serialization. Thrift was found to be slightly faster than ProtoBuf, but the difference is essentially negligible. This is shown in Figure 5.

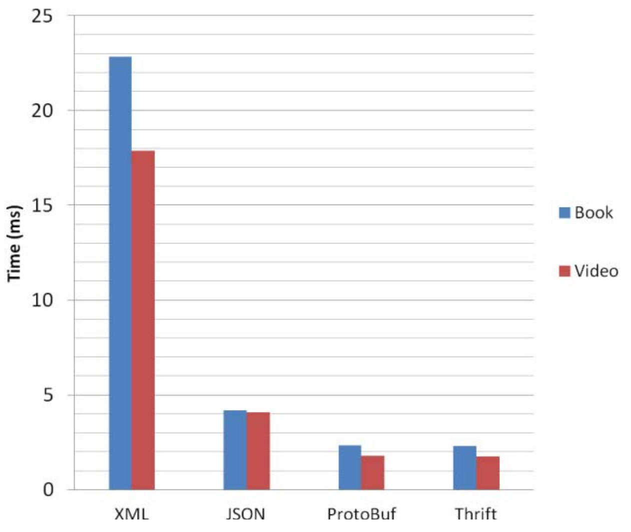

Figure 5: Average serialization Time

As shown in Table 3, the findings for deserialization mirror those of serialization, with XML being slowest, JSON being the next slowest but still much improved over XML, and the binary formats (Thrift and ProtoBuf) being the fastest. The main difference is that ProtoBuf is faster to deserialize than Thrift, however even though the speed difference is more significant than the one found while serializing, it is still a very slight increase in speed. This is shown in Figure 6.

As shown in Figures 5 and 6, the number-heavy video object was the fastest to serialize and deserialize across all data formats.

5.4 Usability

While usability can be hard to quantify, there are certain aspects to each serialization format that should be examined to help determine the usefulness of the format for mobile platforms. One of the main issues is ”human readability”. If a format is ”human readable”, then a developer should be able to view the serialized data out-of-context, and still be able to tell what data is represented. This can be essential in the debugging process. While data serialization formats were not designed to be manually written, if a format is human readable it is possible to hand-write data in that format as well. Both text based formats (XML and JSON) are human readable, while the binary formats are not.

Another important aspect of mobile development is platform compatibility. Since XML and JSON are text based, they are both language agnostic, meaning they can be created with any programming language. Because .proto and .thrift files need to first be compiled and imported, this is not the case with Binary languages.

Another factor to take into consideration when looking into data serialization formats is the available documentation and user base.

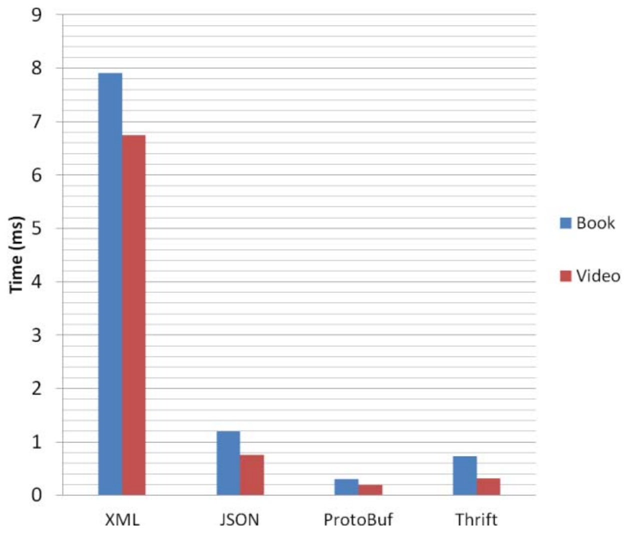

Figure 6: Average deserialization Time

6. Conclusion

In this paper, we tested a number of data serialization formats, which each has a number of advantages and disadvantages. The experiments showed that the XML data format is largely inferior to the other serialization formats: JSON, Thrift, and ProtoBuf. It also has the largest serialized size and the slowest speed. JSON, by comparison, manages to avoid the negatives of XML while maintaining many of the advantages. It has a large user base, is human readable, and is used in a large number of web services. In contrast, the binary formats (Thrift and ProtoBuf) are faster and smaller than the text-based formats, but are not as adaptable since data sent in a binary format cannot be parsed unless the receiver has the ".proto" or ".thrift" file and the other necessary include files.

Therefore, 1) XML should be avoided unless necessary, as JSON is a superior alternative; 2) When dealing with existing web services, JSON should be used when possible since it is a text-based format and can be written and parsed on any platform; 3) When serializing data for storage purposes, or when designing a new web service from the ground up, it is advantageous to use one of the binary formats due to their superior speed and size; 4) The performance differences between Thrift and ProtoBuf are essentially negligible, but Thrift can be compiled in more languages, and supports Objective-C, the language of choice for iOS devices.

7. References

[1] Stephen Ashmore, S. Kami Makki, IMISSAR: An Intelligent, Mobile Middleware Solution for Secure Automatic Reconfiguration of Applications, Utilizing a Feature Model Approach, In Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication (ACM SIGKDD SIGAPP ICUIMC 2011), 21-23 February, 2011, Seoul Korea.

[2] Narasimha Srirangam, Venkata Aiswarya, S. Kami Makki, Shui Yu, Utilizing Intelligent Middleware for Reconfiguration of Applications on Android, In Proceedings of International Conference on Convergence and Hybrid Information Technology, G. Lee, D. Howard, and D. Ślęzak (Eds.): ICHIT 2011, Lecture Notes in Artificial Intelligence (LNAI) Vol. 6935, pp. 81-89. Springer, Heidelberg (2011), September 23rd - 25th, 2011, Daejeon, Korea.

[3] Jackson high-performance JSON processor, http://jackson.codehaus.org/. Accessed: 7/26/2011.

[4] Json, http://www.json.org. Accessed: 7/26/2011.

[5] Simple, http://simple.sourceforge.net, accessed: 7/26/2011.

[6] thrift-protobuf-compare, http://code.google.com/p/thrift-protobuf-compare, accessed: 7/26/2011.

[7] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and Francois Yergeau, Extensible markup language (XML) 1.0 (fifth edition), World Wide Web Consortium, Recommendation REC-xml-20081126, November 2008

[8] Clark C. Evans, Yaml 1.2. http://yaml.org/, accessed: 7/26/2011.

[9] Apache Software Foundation, Thrift, http://thrift.apache.org/, accessed: 7/26/2011.

[10] Google, Protocol buffers, http://code.google.com/p/protobuf/, accessed: 7/26/2011.

[11] Marjan Hericko, Matjaz B. Juric, Ivan Rozman, Simon Beloglavec, and Ales Zivkovic, Object Serialization Analysis and Comparison in Java and .NET. SIGPLAN No. 38:44–54, August 2003.

[12] Jaakko Kangasharju and Sasu Tarkoma, Benefits of Alternate XML Serialization Formats in Scientific Computing. In Proceedings of the workshop on Service oriented computing performance: aspects, issues, and approaches, SOCP '07, pages 23-30, New York, NY, USA, 2007.

[13] Malin Eriksson and Victor Hallberg, Comparison between JSON and YAML for data serialization. Bachelor's thesis 2011, KTH Royal Institute of Technology, Stockholm University, Sweden.

[14] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta, Comparison of JSON and XML data interchange formats: A case study. ISCA 22nd International Conference on Computers and Their Applications in Industry and Engineering, pp. 157-62, 2009.

[15] Google Inc. Android developer's guide - What is Android, http://developer.android.com/guide/basics/what-is-android.html, Collected 2009-03-18, accessed 12/19/2011.