الجمهورية العربية السورية

جامعة البعث

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة التحكم الآلي والحواسيب

عنوان البحث

دراسة تحليلية عن صيغ سَلسَلة البيانات الأكثر استخداماً باستخدام بايثون

الباحثان

طاهر صافي

د. بسيم عمران

Syrian Arab Republic

Al Baath University

Faculty of Mechanical and Electrical Engineering

Department of Automatic Control and Computers

Research title

Analytic Study of Most Used Data Serialization Formats Using Python

Researchers

Taher Safi

Dr. Bassim Oumran

دراسة تحليلية عن صيغ سَلسَلة البيانات الأكثر استخداماً باستخدام بايثون

ملخص:

إن ازدياد مقدار إرسال البيانات عبر شبكة الإنترنت وخاصة مع الانتشار الكبير لإنترنت الأشياء وشبكات الحساسات اللاسلكية جعل من اختيار صيغة مناسبة لسَلسَلة البيانات أمراً هاماً. تكون الأجهزة المتصلة على الشبكة في هذه التطبيقات ذات قدرات حوسبة محدودة ومتصلة بعرض حزمة قليل نسبياً. لقد تم في هذا البحث القيام بدراسة تحليلية عن صيغ سَلسَلة البيانات الأكثر استخداماً وفق ثلاثة معايير وهي حجم السِلسِلة الناتجة وزمن عمليتي السَلسَلة وإلغاء السَلسَلة. إن الصيغ التي تم اختيارها هي صيغة JSON وصيغة YAML وصيغة Bencode وصيغة Pickle نظراً لاستخدام كل منهم في مجال استخدام خاص به. تم اختبار هذه الطرق على وحدة Raspberry Pi باستخدام لغة Python.

الكلمات المفتاحية: سَلسَلة البيانات، JSON، YAML، Bencode، Pickle.

Analytic Study of Most Used Data Serialization Formats Using Python

Abstract:

The increase of sending data using the Internet, especially with the wide use of Internet of Things and Wireless Sensor Networks, makes the process of choosing a proper data serialization method an important thing. In these applications, connected devices are usually low-end devices with small computational power and uses a relatively small bandwidth. In this thesis, an analytical study of most used data serialization was done with respect to three criterion: the size of the result, time of serialization and time of deserialization. JSON, YAML, Bencode, and Pickle was chosen since they are in wide use, each within its own application. These formats were tested using a Raspberry Pi module using Python.

Keywords: data serialization, JSON, YAML, Bencode, Pickle

1. مقدمة

إن سَلسَلة البيانات (data serialization) هي عملية يتم فيها تحويل كائن بيانات ما (data object) إلى سِلسِلة من البيانات [1]. يمكن استخدام سِلسِلة البيانات الناتجة كوسيلة لحفظ كائن البيانات الأصلي على وسائل التخزين أو نقله عبر الشبكة. يمكن الحصول على كائن البيانات الأصلي انطلاقاً من البيانات المسلسلة مسبقاً من خلال عملية تعرف باسم إلغاء السَلسَلة (deserialization).

لقد شهدت السنوات الأخيرة زيادة كبيرة في استخدام شبكة الإنترنت وخاصة مع ظهور إنترنت الأشياء (Internet of Things) وأيضاً شبكات الحساسات اللاسلكية (Wireless Sensor Networks). إن طبيعة البيانات المرسلة في تطبيقات إنترنت الأشياء تتطلب استخدام سَلسَلة البيانات في إرسال القيم من وإلى الأشياء المتصلة بالإنترنت. هذا الأمر جعل من البيانات المسلسلة واحدة من أكثر أنواع الملفات المرسلة عبر شبكة الأنترنت [2].

لقد تم تحليل أداء عمل بعض صيغ سَلسَلة البيانات في دراسات سابقة وعلى منصات اختبار عدة. لقد قام [3] بمقارنة من ناحية الأداء والحجم لصيغة XML وللصيغة الثنائية المستخدمة في منصتي Java و.NET. وتوصل إلى أن عملية السَلسَلة للصيغة الثنائية في منصة Java أفضل منها في منصة .NET. بينما كانت عملية سَلسَلة صيغة XML أفضل منها في منصة .NET..

لقد قام [4] بمقارنة الفعالية بين برامج عدة تستخدم في تطبيقات الويب لسَلسَلة البيانات باستخدام XML و JSON وطرق ثنائية أخرى. تم اختيار برامج تستخدم خوارزميات مختلفة عند إجراء عملية السلسلة. لقد طبق الدراسة من أجل ثلاثة أنواع من الكائنات حيث ركز على الإنتاجية الكلية للصيغة عند المقارنة وبين أن اختيار الخوارزمية يؤثر على فعالية طريقة السَلسَلة وأن صيغة JSON ليست بالضرورة أن تعطي نتيجة أفضل من صيغة XML.

لقد قام [5] بالمقارنة بين طرق XML و JSON و Thrift و ProtoBuf للعمل في بيئة الهواتف المحمولة. لقد تمت المقارنة من ناحية سرعة عملية السلسلة والحجم وقابلية الاستخدام إما للإرسال عبر الشبكة أو للتخزين على وسائل التخزين. تم تنفيذ المقارنة على منصة Android باستخدام بيانات ذات حقول نصية كثيرة وبيانات ذات حقول عددية كثيرة. أظهرت الدراسة أن أداء صيغة XML سيء مقارنة بالطرق الأخرى المدروسة كما بينت الحالات التي يفضل عندها استخدام كل صيغة.

لقد قام [6] بمقارنة بين 12 مكتبة لسَلسَلة البيانات باستخدام صيغة XML و JSON وطرق ثنائية أخرى. لقد تمت المقارنة من ناحية الحجم وزمن المعالجة للمكتبات المذكورة وقد تبين عدم وجود طريقة فضلى حيث يكون أداء المكتبة جيداً في التطبيق الذي صممت من أجله.

2. هدف البحث

يهدف هذا البحث إلى تقديم دراسة تحليلية بين صيغ سَلسَلة البيانات الأكثر استخداماً وذلك بهدف استخدامها في التطبيقات التي تكون فيها قدرات الحوسبة محدودة. تشمل هذه التطبيقات تطبيقات عدة مثل إنترنت الأشياء (اختصاراً IoT) وشبكات الحساسات اللاسلكية (اختصاراً WSN).

3. معايير المقارنة

عند المقارنة بين صيغة من صيغ سَلسَلة البيانات وصيغة أخرى يمكن النظر إلى المعايير الآتية من أجل تحديد فيما إذا كانت صيغة ما من هذه الصيغ مناسبة لتطبيق معين أو غير مناسبة.

3.1 قدرة الإنسان على فهم الصيغة

إن الصيغة المقروءة من قبل الإنسان (human–readable format) هي الصيغة التي يمكن للإنسان قراءتها وفهمها بشكل مباشر دون الحاجة إلى وسيط، بينما تسمى الصيغ التي لا يمكن للإنسان قراءتها وفهمها بشكل مباشر باسم الصيغة المقروءة من قبل الآلة (machine–readable format) [2][7]. تستخدم الصيغ المقروءة من قبل الإنسان عادة محارف يميزها الإنسان كالأحرف والأرقام وعلامات الترقيم بينما تستخدم الصيغ المقروءة من قبل الآلة تمثيلاً خاصاً للبيانات لا يمكن للإنسان قراءته بشكل مباشر دون تحويله إلى تمثيل آخر.

3.2 أنواع البيانات المدعومة

تختلف صيغ سَلسَلة البيانات في نوع البيانات التي يمكن تمثيلها باستخدام الصيغة. تدعم معظم صيغ سَلسَلة البيانات أنواع بيانات بسيطة كالأعداد (numbers) والمحارف (characters) والنوع البولياني (أو المنطقي) (boolean) بالإضافة إلى أنواع بيانات مركبة من الأنواع البسيطة كالمصفوفات (arrays) التي تكون عبارة عن مجموعة متعاقبة من القيم، والكائنات (objects) والتي تعرف أيضاً باسم المصفوفات الترابطية (associative arrays) ويكون لها شكل يربط بين مفتاح (key) وقيمة له (value).

3.3 التمثيل الثنائي

يمكن أن يكون التمثيل الداخلي لصيغ سَلسَلة البيانات عبارة عن محارف بترميز ASCII أو Unicode، أو أن يكون تمثيلاً ثنائياً (binary representation) خاصاً [7]. إن الصيغة التي تسمح بالتمثيل الثنائي للقيم توفر إمكانية تمثيل بعض أنواع البيانات التي لا يمكن تمثيلها باستخدام المحارف فقط ومنها الصور مثلاً. ترتبط هذه الخاصية أيضاً بقدرة الإنسان على قراءة الصيغة إذ أن الصيغة التي تستخدم تمثيلاً ثنائياً تكون غير مقروءة من قبل الإنسان.

3.4 حجم السِلسِلة الناتجة

إن لحجم السِلسِلة الناتجة أهمية كبيرة [2] عند المقارنة بين صيغة من صيغ سَلسَلة البيانات وصيغة أخرى. من الواضح أن نقصان الحجم اللازم لتمثيل كائن ما أمر مرغوب لما لذلك من فوائد في تقليل السعة التخزينية اللازمة لحفظ السِلسِلة الناتجة وفي تقليل الزمن اللازم عند إرسالها أو استقبالها عبر الشبكة وفي تقليل عرض الحزمة اللازم للقيام بذلك على الشبكة.

3.5 تعقيد عمليتي السَلسَلة وإلغاء السَلسَلة

إن لتعقيد عمليتي السَلسَلة وإلغاء السَلسَلة دور كبير في فعالية استخدام صيغة ما ضمن تطبيق معين. إن زيادة تعقيد هاتين العمليتين سيؤدي إلى زيادة الزمن اللازم لتنفيذ تلك العمليتين وهو أمر قد يكون له تأثير كبير [2] في حال كان حجم البيانات التي يتم التعامل معها كبيراً نسبياً وخاصة في تطبيقات IoT و WSN.

3.6 مدى إمكانية تطبيق الصيغة

تكون بعض صيغ سَلسَلة البيانات مصممة لاستخدامها ضمن تطبيق معين أو ضمن لغة برمجة معينة ولا يمكن استخدامها خارج هذا التطبيق أو اللغة. إن إمكانية استخدام صيغة سَلسَلة البيانات في لغات برمجة عدة تسمح باستخدام التطبيق في عدة منصات وبالتالي تسهيل عملية إعادة استخدام نفس التطبيق بشكل مباشر في نواحٍ أخرى دون التقيد بلغة برمجة معينة.

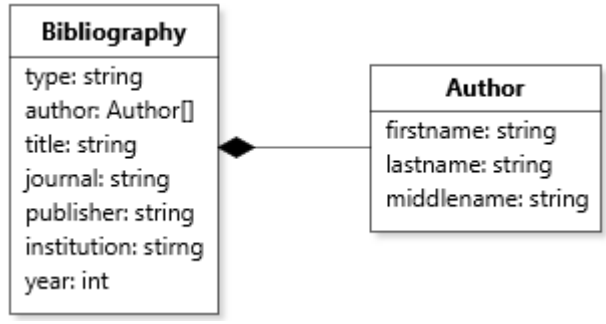
3.7 وجود معيار للصيغة

إن وجود معيار دولي متفق عليه لصيغة سَلسَلة البيانات يسمح بضمان إمكانية استخدام الصيغة في تطبيقات عدة وعلى منصات عدة دون حدوث أية مشاكل تتعلق بالتوافق بين منصة وأخرى.

4. مواد وطرق البحث

تم في هذا البحث دراسة أربع صيغ تستخدم في سَلسَلة البيانات في مجالات مختلفة. تم تطبيق برنامج على وحدة Raspberry Pi تعمل بنظام تشغيل Raspberry Pi OS باستخدام لغة البرمجة Python. يقوم البرنامج بتوليد كائن من النوع المتداخل (nested) بشكل عشوائي وثم تمثيله باستخدام صيغ سَلسَلة البيانات قيد الدراسة وقياس الحجم التخزيني الذي تستهلكه السِلسِلة الناتجة وزمن عملية السَلسَلة بالإضافة إلى زمن عملية إلغاء السَلسَلة.

تمثل البيانات العشوائية التي تم توليدها بيانات عن مراجع لمقالة علمية لها الشكل العام المبين في الشكل 1. إن البيانات المولدة عبارة عن مصفوفة من كائنات تمثل مراجع علمية من خمسة أنواع وهي مقالة، وكتاب، وأطروحة دكتوراة، وتقرير لمؤتمر، وتقرير تقني. لقد تم توليد البيانات بحيث تحتوي واحدة من كل نوع من هذه الأنواع وبترتيب عشوائي. تم توليد الحقول بشكل عشوائي بحيث تكون مشابهة للبيانات التي يتم مصادفتها في حالة العمل مع مثل هذا النوع من البيانات. إن بعض الحقول المبينة في الشكل اختيارية ويتم اختيار تضمينها من عدمه بالاعتماد على نوع المرجع فمثلاً الحقل `publisher` يوجد عندما تكون قيمة الحقل `type` هي `book` ولا يوجد عندما يكون النوع `article`. إن طول البيانات العشوائية المولدة يختلف بين عينة وعينة أخرى وذلك بسبب كون عدد الكلمات في كل حقل من حقول البيانات عشوائياً وكون بعض الحقول ضمنها اختيارية. لقد تم اختيار قيود لعدد الكلمات ولعدد الأحرف ومجال القيم للأعداد بناء على الحقل فمثلاً تم اختيار حقل العنوان للمرجع في البيانات المولدة ليكون بين 3 و 20 كلمة بطول بين 1 و 10 أحرف.



الشكل 1: مخطط UML للشكل العام للبيانات المستخدمة في الدراسة

5. فكرة عن الصيغ المستخدمة في الدراسة التحليلية

لقد تم اختيار صيغ سَلسَلة البيانات الآتية باعتبارها الأكثر شيوعاً واستخداماً في مجالات متعددة:

5.1 صيغة JSON

إن JSON (اختصاراً لـ JavaScript Object Notation) هي صيغة لتبادل البيانات معرفة في المعيار ECMA-404 [8]. إن صيغة JSON صيغة نصية فهي تستخدم محارف وأرقام ورموز وبالتالي يمكن للإنسان قراءتها وفهمها بسهولة كما ويمكن استخدامها في تطبيقات عدة فهي تستخدم ترميز UTF-8 الذي تدعمه جميع أجهزة الحوسبة الحديثة. تدعم صيغة JSON أنواع البيانات الشهيرة كالأعداد والنوع البولياني والنصوص والمصفوفات والكائنات. تدعم العديد من لغات البرمجة هذه الصيغة بشكل مباشر كلغة البرمجة JavaScript ما جعلها مستخدمة بكثرة في الوب. [9]

يبين ما يأتي مثالاً على صيغة JSON ناتجة عن تمثيل بيانات لمرجع.

```
{"bibliography":[{"type":"book","author":[{"f
irstname":"Jiri","lastname":"Soukup"},{"first
```

```
name":"Petr","lastname":"Mach\u00e1\u010dek"}
],"title":"Serialization and Persistent
Objects: Turning Data Structures into
Efficient Databases","publisher":"Springer-
Verlag Berlin
Heidelberg","year":2014,"isbn":"978-3-642-
39323-5"}]}
```


5.2 صيغة YAML

إن YAML (اختصاراً لـ YAML Ain't Markup Language) هي صيغة لتبادل البيانات تستخدم بكثرة لتخزين وإرسال ملفات الضبط للبرامج كما أنها مقروءة من قبل الإنسان فهي تستخدم محارف وأرقام ورموز. تستخدم صيغة YAML ترميز Unicode وهناك عديد من لغات البرمجة التي توفر دعماً لهذه الصيغة، وبالتالي فيمكن استخدامها على جميع الأجهزة التي تدعم هذا الترميز. تتميز صيغة YAML عن JSON في أن للفراغات بين أجزاء السِلسِلة أهمية ولا يمكن الاستغناء عنها. تدعم صيغة YAML أنواع البيانات الشهيرة كالأعداد والبولياني والنصوص والمصفوفات والكائنات. [10]

يبين ما يأتي مثالاً على صيغة YAML ناتجة عن تمثيل نفس البيانات السابقة.

```
bibliography:
- author:
  - firstname: Jiri
    lastname: Soukup
  - firstname: Petr
    lastname: "Mach\xE1\u010Dek"
  isbn: 978-3-642-39323-5
  publisher: Springer-Verlag Berlin
Heidelberg
  title: 'Serialization and Persistent
Objects: Turning Data Structures into
Efficient Databases'
  type: book
```

```
year: 2014
```

5.3 صيغة Bencode

تستخدم صيغة Bencode بشكل أساسي في BitTorrent وهو نظام لنقل البيانات وفق آلية الند للند (peer-to-peer). إن صيغة Bencode صيغة بسيطة تسمح بترميز أنواع البيانات العددية الصحيحة والمحارف والمصفوفات والكائنات فقط. تدعم هذه الصيغة أيضاً ترميز البيانات الثنائية على شكل نص ما يجعلها ملائمة لترميز بيانات لا يمكن تمثيلها باستخدام أنواع البيانات البسيطة. بما أن هذا الصيغة تقبل تمثيل البيانات بشكل ثنائي فهي تعتبر من الطرق التي لا يمكن قراءتها من قبل الإنسان بشكل كامل. [12] [11]

يبين ما يأتي مثالاً على صيغة Bencode ناتجة عن تمثيل نفس البيانات السابقة.

```
d12:bibliographyld6:authorld9:firstname4:Jiri
8:lastname6:Soukuped9:firstname4:Petr8:lastna
me10:Mach\xc3\xa1\xc4\x8dekee4:isbn17:978-3-
642-39323-59:publisher33:Springer-Verlag
Berlin Heidelberg5:title86:Serialization and
Persistent Objects: Turning Data Structures
into Efficient
Databases4:type4:book4:yeari2014eeee
```

5.4 صيغة Pickle

إن صيغة Pickle هي صيغة خاصة بلغة البرمجة Python وهي صيغة ثنائية وبالتالي فهي غير مقروءة من قبل الإنسان. تدعم هذه الصيغة أنواع البيانات المدعومة من قبل لغة البرمجة Python. إن كون هذه الصيغة محدودة بلغة برمجة محددة يجعل من غير الممكن تطبيقها إلا في المنصات التي تدعم هذه اللغة. [8]

يبين ما يأتي مثالاً على صيغة Pickle ناتجة عن تمثيل نفس البيانات السابقة.

\x80\x04\x952\x01\x00\x00\x00\x00\x00\x00}\x94\x8c\x0cbibliography\x94]\x94}\x94(\x8c\x04type\x94\x8c\x04book\x94\x8c\x06author\x94]\x94(}\x94(\x8c\tfirstname\x94\x8c\x04Jiri\x94\x8c\x08lastname\x94\x8c\x06Soukup\x94u}\x94(h\t\x8c\x04Petr\x94h\x0b\x8c\nMach\xc3\xa1\xc4\x8dek\x94ue\x8c\x05title\x94\x8cVSerialization and Persistent Objects: Turning Data Structures into Efficient Databases\x94\x8c\tpublisher\x94\x8c!Springer-Verlag Berlin Heidelberg\x94\x8c\x04year\x94M\xde\x07\x8c\x04isbn\x94\x8c\x11978-3-642-39323-5\x94uas.

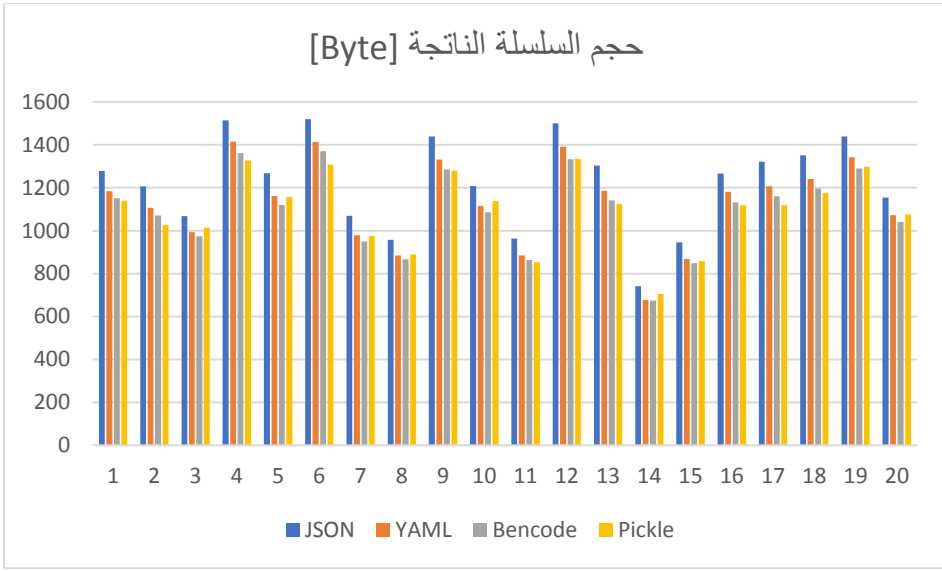
يبين الجدول 1 ملخصاً لمعايير المقارنة بين صيغ سَلسَلة البيانات قيد الدراسة. [8][9][10][11][12].

	JSON	YAML	Bencode	Pickle
قدرة الإنسان على فهم الصيغة	نعم	نعم	لا	لا
نوع البيانات المدعومة	جميع الأنواع الشائعة	جميع الأنواع الشائعة	لا تدعم الأعداد ذات الفاصلة	جميع الأنواع الشائعة
التمثيل الثنائي	لا	لا	نعم	نعم
مقدار دعم لغات البرمجة	واسع	واسع	واسع	فقط Python
وجود معيار	نعم	لا	نعم	نعم

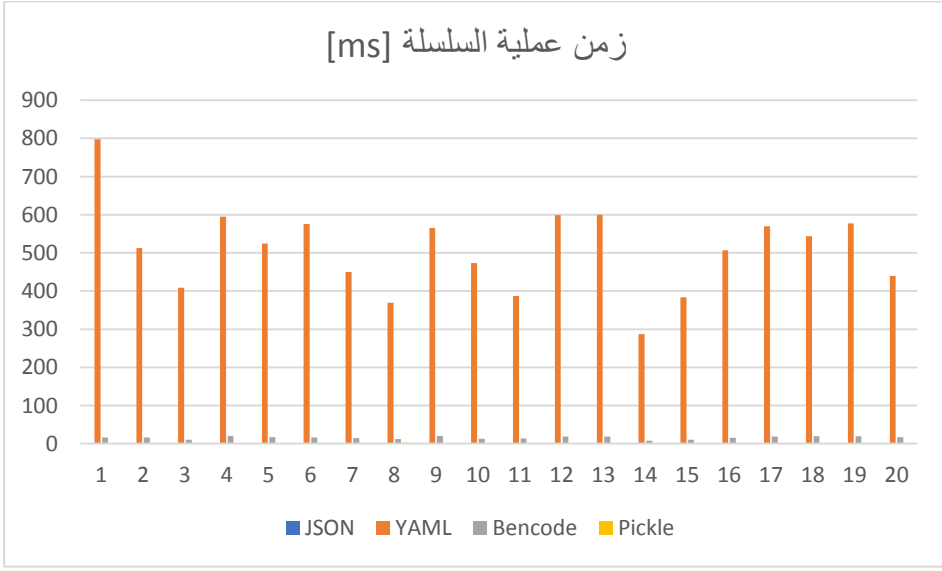
الجدول 1: ملخص المقارنة بين صيغ سَلسَلة البيانات

6. النتائج ومناقشتها

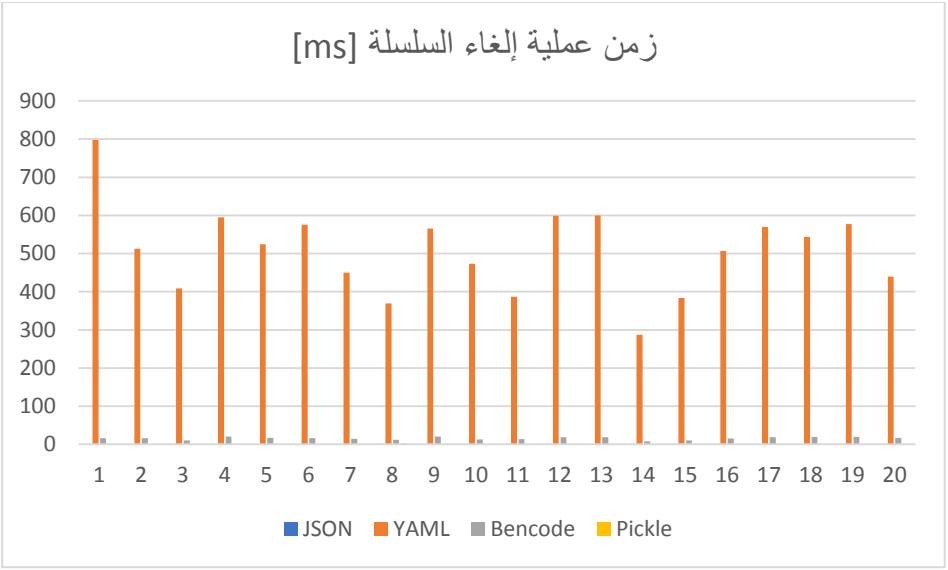
لقد تم تنفيذ الدراسة على وحدة Raspberry Pi باستخدام لغة البرمجة Python. تم كتابة برنامج يقوم بتوليد بيانات عشوائية ذات حقول مختلفة وبأطوال مختلفة ثم يقوم بتحويل هذه البيانات إلى كل صيغة من الصيغ المدروسة وقياس الزمن المستغرق خلال عملية التحويل مع حجم السِلسِلة الناتجة، وبعدها قياس زمن عملية إلغاء السَلسَلة التي تطبق على السِلسِلة الناتجة عن كل صيغة وتحويلها إلى كائن البيانات الأصلي. تم تكرار هذه العملية 1000 مرة وتم الحصول على النتائج من أجل كل مرة. إن حجم النتائج كبير ولا يتسع لعرضه هنا وسيتم الاكتفاء بعرض ملخص كما يأتي. تبين الأشكال من الشكل 2 إلى الشكل 4 على الترتيب حجم السِلسِلة الناتجة، وزمن عملية السَلسَلة، وزمن عملية إلغاء السَلسَلة وذلك من أجل عينة من القيم التي تم الحصول عليها.



الشكل 2: عينة من قيم حجم السِلسِلة الناتجة لصيغ سَلسَلة البيانات المدروسة



الشكل 3: عينة من قيم زمن عملية السَلسَلة من أجل صيغ سَلسَلة البيانات المدروسة



الشكل 4 : عينة من قيم زمن إلغاء السَلسَلة من أجل صيغ سَلسَلة البيانات المدروسة

يبين الجدول 2 متوسط القيم التي تم الحصول عليها من أجل حجم السِلسِلة الناتجة، وزمن عملية السَلسَلة، وزمن عملية إلغاء السَلسَلة من أجل طرق سَلسَلة البيانات الأربعة.

Pickle	Bencode	YAML	JSON	
1119.84	1117.14	1152.12	1244.44	حجم السِلسِلة الناتجة (B)
0.73	11.99	369.73	2.72	زمن السَلسَلة (ms)
0.55	15.7	500.1	1.25	زمن إلغاء السَلسَلة (ms)

الجدول 2: نتائج الحجم وزمن عمليتا السَلسَلة وإلغاء السَلسَلة

بالنظر إلى الأشكال والجداول يتبين أن حجم السِلسِلة الناتجة بين الطرق الأربعة متقارب نسبياً حيث كانت حجم السِلسِلة الناتجة عن صيغة JSON هو الأكبر مقارنة بالطرق

الأخرى. يتبين أيضاً بالنظر إلى زمن عمليتي السَلسَلة وإلغاء السَلسَلة أن الزمن من أجل صيغة YAML أكبر بكثير من الطرق الأخرى بينما يكون الزمن في صيغة Pickle صغيراً جداً وهذا أمر متوقع كون هذه الصيغة جزء من لغة البرمجة المستخدمة.

7. الاستنتاجات والتوصيات

يمكن تلخيص الاستنتاجات التي تم التوصل إليها بما يأتي:

- إن صيغة JSON تعطي أداءً جيداً كون زمن عملتي السَلسَلة وإلغاء السَلسَلة صغير كما أنها مدعومة من كثير من لغات البرمجة.
- تعاني صيغة YAML من كون زمن عمليتي السَلسَلة وإلغاء السَلسَلة كبير جداً مقارنة بالطرق المدروسة الأخرى مع أن حجم السِلسِلة الناتجة فيها أصغر بشكل عام من حجم السِلسِلة الناتجة عن صيغة JSON.
- إن صيغة Bencode تعطي أداءً جيداً أيضاً من ناحية حجم السِلسِلة الناتجة فهو أقل بشكل عام من حجم السِلسِلة الناتجة عن JSON إلا أنها لا تدعم أنواع بيانات ضرورية كالأعداد ذات الفاصلة.
- إن أداء صيغة Pickle ممتاز جداً ولكن دعمها محصور ضمن لغة Python كما أنها غير مقروءة من قبل الإنسان.

بناء على الاستنتاجات التي تم التوصل إليها تقترح الدراسة استخدام صيغة Bencode في تطبيقات أخرى بالإضافة إلى استخدامها الحالي في بروتوكول BitTorrent فقط مع إضافة دعم لأنواع البيانات الأخرى غير المدعومة في المعيار الحالي. تقترح الدراسة أيضاً استخدام صيغة YAML فقط من أجل عملية تخزين البيانات التي لا تتطلب وصولاً متكرراً كون الحجم التخزيني لها أقل من JSON وكونها مقروءة من قبل الإنسان.

8. المراجع

[1] SOUKUP, J., MACHÁČEK, P. 2014 – **Serialization and Persistent Objects**. Springer Heidelberg New York Dordrecht London, 263p.

[2] MCANLIS, C., HAECKY, A. 2016 – **Understanding Compression**. O'Reilly Media United States of America, 217p.

[3] HERICKO M., JURIC B.M., ROZMAN I., BELOGLAVEC S., ZIVKOVIC A. 2003 – **Object Serialization Analysis and Comparison in Java and .NET**, Vol. 38(8), 44–54.

[4] AIHKISALO T, PAASO T. 2011. **A Performance Comparison of Web Service Object Marshalling and Unmarshalling Solutions**. 2011 IEEE World Congress on Services, Washington, DC, USA. 122–129.

[5] SUMARAY A, KAMI MAKKI S. 2012. **A Comparison of Data Serialization Formats For Optimal Efficiency on a Mobile Platform.** ICUIMC' 12.

[6] MAEDA K. 2013. **Performance Evaluation of Object Serialization Libraries in XML, JSON and Binary Formats**, 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), Bangkok, Thailand. 177–182.

[7] https://isocpp.org/wiki/faq/serialization#serialize-decide-text-vs-binary. Retrieved on 3/29/2021

[8] ECMA International. 2017 – **The JSON Data Interchange Syntax, 2nd Edition**. Ecma International, Austria, 16p.

[9] https://www.json.org/json-en.html. Retrieved on 1/13/2021.

[10] Ben-Kiki, O., Evans, C., Ingy döt Net. 2009 – **YAML Ain't Markup Language (YAML™) Version 1.2**. 84p.

[11] https://wiki.theory.org/index.php/BitTorrentSpecification. Retrieved on 1/7/2021.

[12] http://www.bittorrent.org/beps/bep_0003.html. Retrieved on 1/7/2021.

[13] https://docs.python.org/3/library/pickle.html. Retrieved on 1/18/2021.