DOI：10.14005/j.cnki.issn1672-7673.20210315.002

射电望远镜控制系统中的数据传输序列化分析*

李 军[1,2]，王 娜[1,3]，刘志勇[1,3]，宋祎宁[1,2]，杨 垒[1,2]，王吉利[1]

（1. 中国科学院新疆天文台，新疆 乌鲁木齐 830011；2. 中国科学院大学，北京 100049；
3. 中国科学院射电天文重点实验室，江苏 南京 210033）

摘要：控制系统能衔接、集成和管理射电望远镜的软硬件系统。控制系统的序列化工具可以将射电望远镜的不同设备、操作系统、编程语言和网络之间传输的信息进行编码和解码，增强系统之间数据的传输效率。分析和比较了 3 款二进制序列化工具 Msgpack，Protobuf 和 Flatbuffers 的编码原理及特性，并通过一个实例测试了它们的序列化数据大小、序列化时间和中央处理器利用率。结果表明，Msgpack 的综合性能优于 Protobuf 和 Flatbuffers，适用于周期长、需求易变的射电望远镜系统之间传输信息的编码和解码。

关键词：射电望远镜；二进制序列化工具；控制系统；编码；解码

中图分类号：TP311 **文献标识码**：A **文章编号**：1672-7673(2022)01-0008-08

射电望远镜是射电天文研究的基础，它由天线、接收机、终端、监测和控制等系统组成，其中，具有连接、集成和管理功能的控制系统是射电望远镜的重要组成部分[1]。数据交换是控制系统的基本功能之一，在望远镜控制与多终端数据交换过程中，需要保证稳定可靠的同时兼备高效和通用。对于将要建设的新疆奇台 110 m 射电望远镜(QiTai Radio Telescope，QTT)，各设备之间通信数据大小由观测波段和观测模式决定，它们之间的单次数据通信量一般小于 1 kB。天线伺服控制的数据通信最频繁，数据交换频率约为 20 Hz，单次数据交换大小约 200 B，其他设备的数据交换频率约 1 Hz。主动面运行时，数据通信量约 10 kB，电磁监测的数据通信量一般为 10~100 kB。110 m 射电望远镜控制系统拟采用分布式架构，各子系统之间的数据交换包含多种方式，如 Linux，Windows，VxWorks，Unix 和嵌入式等系统之间的信息传输；网络的大端模式与机器的小端模式之间的信息传输；C++与 Python 之间的信息交换等。其中，大端模式的机器与小端模式的机器传输信息时，long 类型数据的前后字节互换。为了解决系统之间传输信息的数据格式问题，序列化工具将射电望远镜系统之间的传输信息编码为统一格式。因此，序列化工具作为控制系统传输信息格式的基础，可以实现射电望远镜软硬件系统之间的信息传输。

现有的射电望远镜控制系统大多使用序列化技术[2]。如阿塔卡马大型毫米阵列(Atacama Large Millimeter Array，ALMA)的控制系统架构结合文本序列化工具 XML(Extensible Markup Language)作为控制系统传输信息编码和解码的基础[3]；澳大利亚平方公里矩阵探路者(Australian Square Kilometre Array Pathfinder，ASKAP)的监控系统使用网络通信引擎(Internet Communications Engine，ICE)提供的序列化技术，完成不同射电望远镜软硬件系统之间的信息传输[4]；巨型米波射电望远镜(Giant Metrewave Radio Telescope，GMRT)监控系统以 Tango 控制系统为基础，结合 XML 实现系统传输信息的编码和解码[5]。为了解决望远镜控制系统之间信息传输的格式问题，文[6]提出了以通信中间件 ZeroMQ 和文本序列化工具 JSON(JavaScript Object Notation)为望远镜自动控制系统的通信框架。

* 基金项目：国家重点研发计划（2018YFA0404603）；中国科学院天文台站设备更新及重大仪器设备运行专项经费；中国科学院西部之光项目（XBQN-A-1）资助.

收稿日期：2021-01-28；修订日期：2021-02-18

作者简介：李 军，男，博士研究生. 研究方向：大口径射电望远镜相关技术. Email：lijun@xao.ac.cn

通信作者：刘志勇，男，副研究员. 研究方向：射电天文、大口径射电望远镜相关技术. Email：liuzhy@xao.ac.cn

序列化工具中，XML 和 JSON 是文本型序列化工具，广泛应用于互联网软件系统，以及早期射电望远镜控制系统的应用层与服务层之间的数据交换。在射电望远镜控制系统的使用过程中，我们发现 XML 和 JSON 存在一些不足，如内存使用率高，数据类型精度易缺失，难以实现底层设备驱动程序与服务之间的数据交换[7]。于是实验物理装置、射电望远镜等底层与服务层之间的通信逐渐被二进制序列化工具 Msgpack，Protobuf 和 Flatbuffers[8-9]替代，它们可以更好地解决数据精度缺失、底层与服务层之间的数据交换效率等问题。本文着重分析 Msgpack，Protobuf 和 Flatbuffers 3 款二进制序列化工具的编码原理、特性，通过测试、比较和分析它们的序列化数据大小、序列化时间和中央处理器利用率，兼顾底层、服务层和应用层，选择适合射电望远镜控制系统的序列化工具，以提高控制系统的信息传输效率，保证射电望远镜系统信息传输格式的统一性和兼容性。

1　序列化工具

序列化工具由编码(又称序列化)和解码(又称反序列化)构成。序列化是将结构化数据(或对象)编码为字节流；反序列化则是将字节流还原成原始的结构化数据(或对象)。

使用序列化工具构建射电望远镜控制系统时，我们需要分析它的编码原理和特性。不同的编码方式影响序列化数据大小、序列化时间、中央处理器利用率等。本节后续部分以图 1 的 JSON 数据为例分析 Msgpack，Protobuf 和 Flatbuffers 的编码原理。

```
{
    "names" : "zhang",
    "num" : 1331,
    "descript" : "inthefirstnicks"
}
```

图 1　JSON 格式示例

Fig. 1　An example of JSON schema

1.1　Msgpack

Msgpack 是一款支持多语言、跨平台、具有动态编译的二进制序列化工具，编码对象(或结构化数据)之后的字节流具有紧凑、简洁的特点。字节流由头字节、前缀字节和数据字节构成。头字节表示之后紧跟的数据类型和类型个数；前缀字节表示其后的数据类型；数据字节表示对象的内容，如基本类型 bin，float 和 uint，结构化类型 str，array 和 map，扩展类型 ext 和 fixext 等。其中，字符串 str 不使用任何标记(或任何转义字符)表示内容结束。

Msgpack 的编码方式包括两种：第 1 种方式使用 Msgpack 编码 key-value 值(这种方式简写为 MSGP-M)，需要先编码 key 值，再编码 value 值。图 2 和图 3 分别表示 MSGP-M 编码图 1 的 JSON 数据之后得到的逻辑图和字节流图。逻辑图为编码之后的数据表示形式；字节流图则是编码之后各字节的先后顺序，如 83 为第 1 个字节。字节流由 7 部分组成：(1)第 1 个字节(83)的前 4 位(1000)表示编码的数据类型为 map，后 4 位(0011)表示后续包含 3 个 map 对象。(2)第 2 个字节为第 1 个 map 对象中 key 值的前缀字节(A5)，表示后续包含 5 个 str 对象；第 3～第 7 个字节以 ASCII 码表示 map 对象中的 key 值“names”。(3)第 8 个字节(A5)表示后续包含 5 个字符串；第 9～第 13 个字节使用 ASCII 码表示字符串“zhang”。(4)第 14 个字节表示第 2 个 map 对象中 key 值的前缀字节(A3)，表明后续包含 3 个字符串；第 15～第 17 个字节以 ASCII 码表示字符串“num”。(5)第 18 个字节为第 2 个 map 对象 value 值的前缀字节(CD)，表明其后紧跟 2 个字节的无符号整数；第 19～第 20 个字节则以大端模式表示数字“1331”。(6)第 21 个字节表示第 3 个 map 对象的 key 值前缀字节(A8)，表示后续包含 8 个字符串；第 22～第 29 字节使用 ASCII 表示字符串“descript”。(7)第 30 个字节(AF)为第 3 个 map 对象的 value 值的前缀字节(A5)，表示后续包含 15 个字符串；第 31～第 45 字节表示字符串“inthefirstnicks”。

第 2 种使用 Msgpack 编码 JSON 格式中的 value 值(序列化结果以数组表示)。这种方式的 Msgpack 缩写为 MSGP-D。MSGP-D 编码图 1 的 JSON 数据之后得到的逻辑图和字节流图如图 4 和图 5，包括 4 部分：(1)第 1 个字节(0X93)表示后续包含 3 个 array 对象。(2)第 2 个字节(0XA5)表示后续包含 5 个字符串；第 3～第 7 字节使用 ASCII 表示字符串“zhang”。(3)第 8 个字节(0XCD)表明后续包含一个 16 位无符号整数；第 9～第 16 字节以大端模式的二进制形式表示数字“1331”。(4)第 11 字节(0XAF)后续包含 15 个字符串；第 12～第 26 字节用 ASCII 表示字符串“inthefirstnicks”。

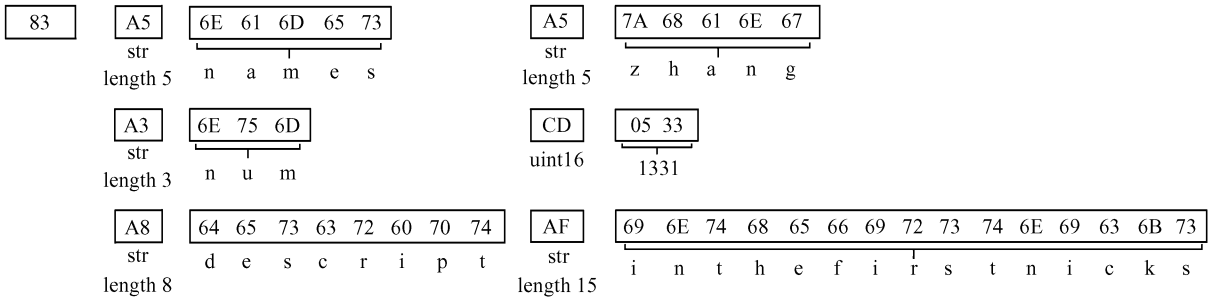

图 2 MSGP-M 对 JSON 格式中 key-value 值编码之后的逻辑图

Fig. 2 Logic diagram after MSGP-M encodes the key-value in JSON format

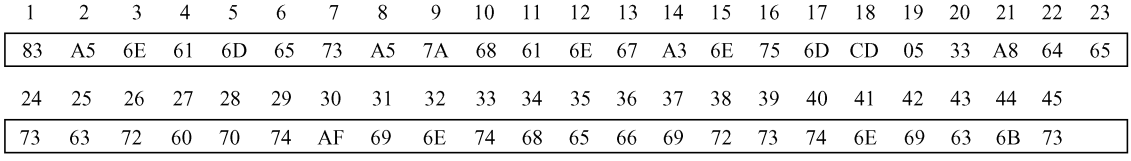

图 3 MSGP-M 对 JSON 格式中 key-value 值编码之后的字节流图

Fig. 3 MSGP-M encodes byte stream after the key-value in the JSON format

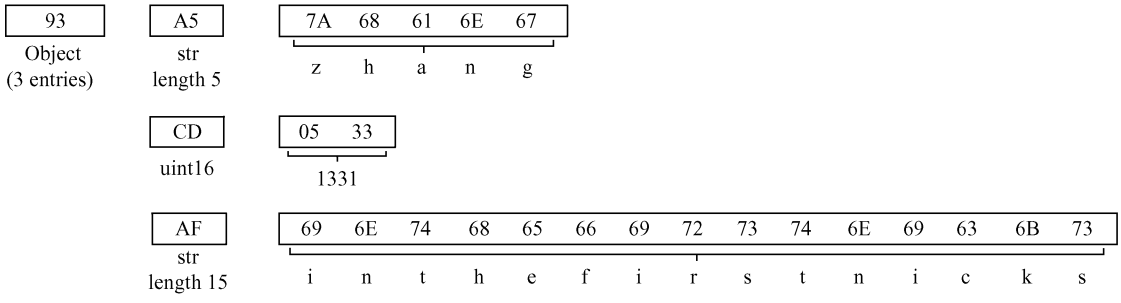

图 4 MSGP-D 对 JSON 格式中 value 值编码后的逻辑图

Fig. 4 Logic diagram after MSGP-D encodes value in JSON format

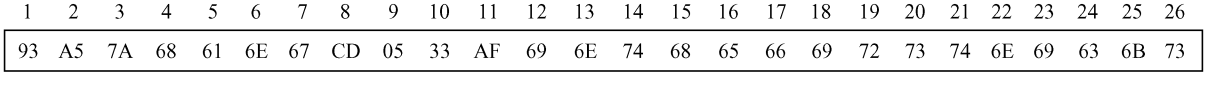

图 5 MSGP-D 对 JSON 格式中 value 值编码后的字节流图

Fig. 5 MSGP-D encodes byte stream of the value in the JSON format

1.2 Protobuf

Protobuf(PB)是一款开源、支持多语言、跨平台、提供接口描述语言(Interactive Data Language, IDL)的二进制序列化工具。PB 编码传输信息时，需定义 IDL 的键和字段，以生成指定编程语言代码，如 C++，Python 等。使用 PB 编码数据后，得到的字节流由键、前缀字节和数据字节组成。键分为标记数字和标记类型，标记数字将常用元素标记为 1~15，不常用元素标记为 16~2047；标记类型包括 string，float 等。键可以对 IDL 文件中的数据类型进行唯一标记，标记后的数据类型不能更改。

图 6 和图 7 分别为 PB 编码图 1 的 JSON 数据之后得到的逻辑图和字节流图。其中，字节流占用空间为 27 字节，由 3 部分组成：(1)第 1 个字节(0A)中的第 2 位到第 5 位(0001)为数字标记,后 3 位(010)表示数据类型为 string；第 2 个字节(05)表示后续包含 5 个 string 对象；第 3~第 7 字节以 ASCII 的形式表示字符串“zhang”。(2)第 8 个字节(10)表示后续包含一个整型的数据；第 9~第 10 个字节是以小端模式表示 16 位的无符号整数“1331”。(3)第 11 个字节(10)表示后续包含字符串；第 12 个字节(0F)则表示后续包含 15 个字符串；第 13~第 27 字节表示字符串“inthefirstnicks”。

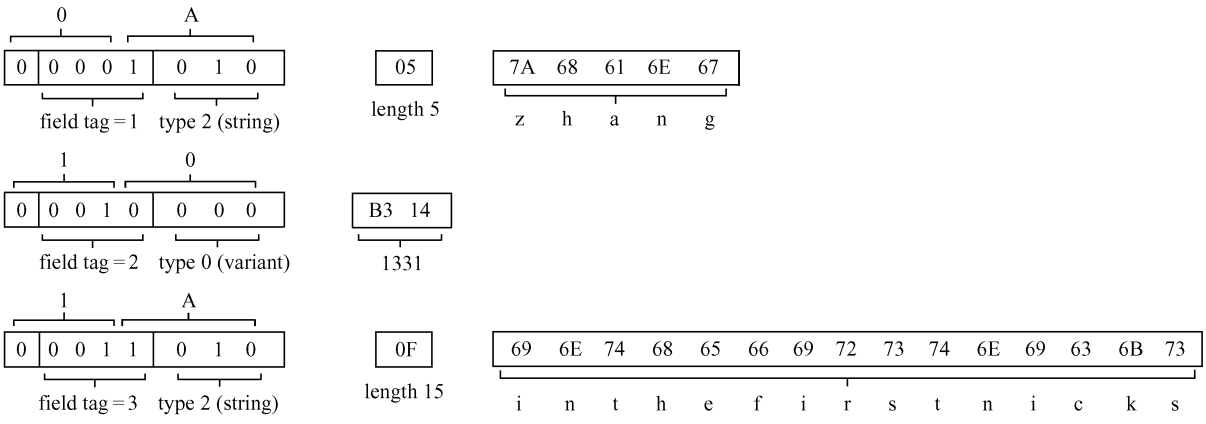

图 6　Protobuf 对 JSON 格式中 value 值编码后的逻辑图

Fig. 6　Logic diagram after Protobuf encodes value in JSON format

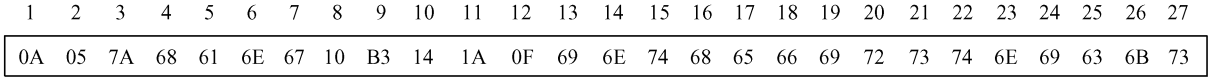

图 7　Protobuf 对 JSON 格式中 value 值编码后的字节流图

Fig. 7　Protobuf encodes the result of value in JSON format

1.3　Flatbuffers

Flatbuffers(FB)是一款支持多语言、跨平台、提供 IDL 的二进制序列化工具。FB 具备良好的兼容性，如系统添加新功能时，新字段只能在 IDL 文件末尾添加，且旧字段仍会正常读取；数据在内存中的格式与编码格式一致；反序列化过程支持零拷贝，便于快速读取数据。FB 序列化字节流包括 int，string 等标量和 struct，table 等矢量。标量由固定长度的以小端模式表示的整型(8~64 位)和浮点型构成；矢量由字符串和数组构成，开头必须是一个 32 位长度的 VECTOR SIZE 来指明矢量长度(不包括'\0'和本身占用空间大小)。其中，字符串和数组的唯一区别是字符串包含一个结束符"\0"。

图 8 为 FB 对 JSON 格式中 value 值编码后的字节流图。字节流占用空间大小为 62 字节，由 3 部分组成：(1)第 1~第 4 字节 root offset(10 00 00 00)为根偏移量，偏移 16 个字节之后为编码数据；第 5~第 6 字节 align(00 00)具有填充作用。(2)第 7~第 8 字节 vtable size(0A 00)为表示 vtable 的字节大小，包括 vtable size，object size，offset num，offset descript 和 offset names 占用的空间大小；第 9~第 10 字节 object size(10 00)表示在表中存储数据占用的空间偏移大小，包括 vtable offset，int offset，1331 和 string offset；第 11~第 12 字节 offset num(04 00)表示 num 在字节流中的位置，offset 只需移动 4 个字节便能找到 num 的偏移量；第 13~第 14 字节 offset descript(08 00)为 descript 的位置，通过 vtable offset 和 int offset 便能找到 descript 的位置；第 15~第 16 字节 offset name(0C 00)表示 vtable offset，int offset 和 string offset 之后为 name 对象。(3)第 17~第 20 字节 vtable offset(0A 00 00 00)与 vtable size 具有相同的大小，唯一的区别是后者占 2 个字节；第 21~第 28 字节分别表示 num 的前缀和以小端模式表示的数字"1331"；第 29~第 32 字节(0F 00 00 00)表示后续类型为 string；第 33~第 36 字节(0F 00 00 00)表明后续包含 15 个字符串；第 37~第 52 字节中包含 15 个以 ASCII 表示的"inthefirstnicks"和一个结束字符"\0"；第 53~第 62 字节与第 33~第 52 字节的编码原理相同。

1.4　编码原理分析与对比

对于 3 款二进制序列化工具，Msgpack 不使用 IDL 预先设置数据结构，可以手动编写字段，具有两种编码方式；Protobuf 和 Flatbuffers 在 IDL 中定义传输的信息字段，使用 IDL 编译器生成对应的编程语言接口，且只有一种编码方式。3 款二进制序列化工具的编码原理不同，编码之后的字节流占用空间大小不同。Msgpack 编码的字节流只有头字节，以及一一对应的前缀字节和数据字节。Protobuf 的字节流中包含一一对应的键、前缀字节和数据字节，其中，只有数据类型使用键和数据字节。Flatbuffers 序

列化之后的字节流与数据在内存中的存储格式一致，Flatbuffers 字节流中不仅包括前缀字节和数据字节，还包括 root offset，object size 和 vtable offset 等不能表示内容的字节。对同一数据编码格式，MSGP-M 序列化 JSON 格式的全部数据，占用空间大。MSGP-D 编码之后的字节流占用空间最小，字节流不用于表示信息的只有头字节和前缀字节。Protobuf 占用的空间稍大，字节流中不能表达数据信息，只包含一一对应的键和前缀字节。Flatbuffers 占用空间大，字节流中包含大量不能表达数据的信息。

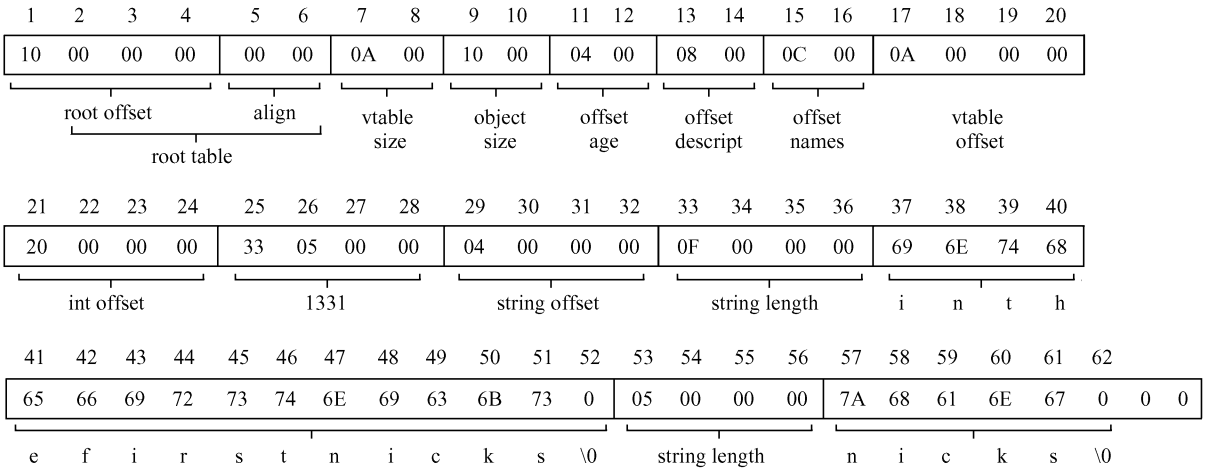

图 8 Flatbuffers 对 JSON 格式中 value 值编码后的字节流图

Fig. 8 Flatbuffers encodes the byte stream of the value in JSON format

2 实验结果与分析

Msgpack，Protobuf 和 Flatbuffers 不仅可以在 Linux，Windows 等操作系统上运行，还支持 C，C++ 和 Python 等编程语言。然而，控制系统开发往往使用多种编程语言，其中，C 和 C++用于底层驱动程序开发和通信；Python 用于服务端的开发以及数据处理等。因此，对于 3 款二进制序列化工具的测试，测试环境的中央处理器为 2.0 GHz 的 Intel Core i7-4750，内存为 8 GB，操作系统为 Ubuntu 16.04。编译环境的 GCC 版本为 5.3.1，Python 版本为 3.7.3。Msgpack，Flatbuffers 和 Protobuf 的版本分别为 1.2.1，1.1.0 和 3.7.1。

图 9 为射电望远镜系统之间的数据传输格式，所传输的信息为控制系统中射电望远镜的状态信息编码。TelStatus 为射电望远镜的状态信息，主要包括天线方位标签 AzFlag、天线俯仰标签 ElFlag、子系统时间标签 TimeFlag、望远镜状态标签 ServerFlag 和天线位置标签 PositionFlag 等。Weather 表示天文台站周围的气象信息，如气象设备状态、日期、风塔、气象仪器和风玫瑰图。WindTower 表示气象仪器的温度、气压、湿度、风速和风向等要素。Plot 表示风玫瑰图用于统计台站周围一段时期内的风向、风速等。控制系统只对传输的数据进行一次编解码。由于传输的数据中包含浮点型和双精度型数据，编码之后的数据在控制系统中不能以 ASCII 编码传输。下面以图 9 为例，使用 C++和 Python 测试 3 款二进制序列化工具的序列化数据大小、序列化时间和中央处理器利用率。

2.1 序列化数据大小

Msgpack 有两种编解码方式，既能编码和解码 JSON 格式中的 key-value 值，又能编码和解码 JSON 格式中的 value 值。我们使用 3 款二进制序列化工具分别测试图 9 的数据，得到 MSGP-M，MSGP-D，Protobuf 和 Flatbuffers 的字节流大小分别为 713 B，460 B，520 B 和 794 B。因此，序列化数据大小与编码原理密切相关。MSGP-M 较 MSGP-D 占用空间大，是因为 MSGP-D 只编码图 1 中的 key 值。MSGP-D 的字节流表示为一个头字节、一一对应的前缀字节和数据字节，而 Protobuf 的字节流包含一一对应的键、前缀字节和数据字节。Flatbuffers 占用空间大是因为其不仅编码 key-value 中的 value 值，还包括非数据值，如 root offset，int offset 和 float offset 等。因此，MSGP-D 比 Protobuf 和 Flatbuffers 输

出格式更紧凑，占用空间更小。


```
{
    "TelStatus": {
        "AzFlag": 0,  "ElFlag": 0, "TimeFlag": 0, "ServerFlag": 0, "PositionFlag": 0, "Az":
        54.321345,  "El": 44.321345, "AzRotation": 1,   "ElRotation": 1,
        "AzSpeed": 0.4234, "ElSpeed": 0.2345, "AzDiff": 1.23455, "ElDiff": 2.12456,
        "AzLimit": 0, "ElLimit": 0, "Track": 0 },
    "Weather": {
        "Status": 1,
        "Date": "2020-10-01 12:23:34.2000",
        "WindTower": {"one": [0.232, 230],"two":[0.3321, 230], "three": [0.4332, 230],"four":
        [0.4921, 230], "five":[0.6112, 230],"six":[0.7021, 230], "seven": [0.832, 230],
        "eight":[0.9431, 230], "nine": [1.232, 23],"ten":[1.5321, 230], "eleven": [1.862, 230]
        ,"twelve":[2.3321, 230]},
        "MeteoroInstrument": {
            "Temperature": 23.43,  "Pressure": 840.543, "Humidity": 0.3567, "Speed": 3.21,
            "Dir": 32.3213},
        "Plot": {
            "SpeedRose": [1.322689, 0.241007, 2.210769, 3.381295, 1.353786, 0.224426, 0.
            064588, 1.285891, 0.436123, 1.435216, 0.180438, 1.169811, 1.194245, 0.230022, 3.
            083807, 0.183219],
            "DirRose": [0.122689, 0.041007, 0.010769, 0.081295, 0.053786, 0.024426, 0.064588,
            0.085891, 0.036123, 0.035216, 0.080438, 0.069811, 0.094245, 0.030022, 0.083807, 0.
            03219]}
    }
}
```


图 9　射电望远镜系统之间的数据传输格式

Fig. 9　Data transmission format between radio telescope systems

2.2　序列化时间

Msgpack，Protobuf 和 Flatbuffers 的编解码原理不同，序列化时间和反序列化时间存在差异。以图 9 为例，3 款二进制序列化工具迭代 100 000 次之后，它们的单次平均序列化时间见图 10。MSGP-M(C++)的序列化时间为 22.425 μs，反序列化时间为 52.491 μs；Python 的序列化时间为 15.566 μs，反序列化时间为 10.896 μs。其中，C++的序列化时间比反序列化时间短，Python 的序列化时间比反序列化时间长，是因 C++的基本数据类型多，解码时间长；而 Python 基本类型少，能更好匹配 key-value 值，解码时间短。MSGP-M(C++)的序列化时间为 22.425 μs 较 MSGP-D(C++)的 13.321 μs 时间长，是因为 MSGP-M 需要编码 key-value，而 MSGP-D 只需编码 value 值。同理，解码与编码的原理相似。Protobuf(C++)序列化时间为 10.514 μs，反序列化时间为 17.163 μs；Python 的序列化时间为 86.506 μs，反序列化时间为 62.431 μs，两种编程语言各自的序列化和反序列化时间接近，是由 PB 的 IDL 决定。Flatbuffers(C++)序列化时间为 5.446 μs，反序列化时间为 0.344 μs；Python 的序列化时间为 203.719 μs，反序列化时间为 2.130 μs。Flatbuffers 的序列化时间比反序列化时间长，是因为编码之后的字节流与其在内存中的数据格式一致，解码不需要时间，只需读取输入输出的时间。

从图 10 可知，对于同一种二进制序列化工具的不同编程语言，Flatbuffers(C++)的序列化速度比 Python 的快 40 倍。Protobuf(Python)的序列化时间是 C++的 8 倍以上。MSGP-M 或 MSGP-D 同一种编码方式的 C++和 Python 的序列化时间与反序列化时间接近，不会因编程语言不同造成序列化和反序列时间的不平衡。

2.3　中央处理器利用率

中央处理器利用率的高低会影响程序运行。以图 9 展示的数据为例，测试 3 款二进制序列化工具得到表 1 的结果。由表 1 可知，无论是 C++还是 Python，Msgpack 编解码的中央处理器利用率均在 12.4% 左右，Protobuf 的中央处理器利用率也是 12.4%。然而，Flatbuffers 编码和解码的中央处理器利用率却存在差异。在编码时，Flatbuffers 的 Python 中央处理器利用率达到 25.9%，远高于 Msgpack 和 Protobuf 编码时 C++和 Python 的中央处理器利用率；而解码时，Flatbuffers 的中央处理器利用率相比 Msgpack 和 Protobuf 略低。因此，Msgpack 和 Protobuf 适用于服务端和客户端内存充足的场景，而 Flatbuffers 可以应

用于服务端内存充足、客户端内存不足的情况。然而，射电望远镜控制系统在实际应用中的服务端和客户端内存相似，在中央处理器利用率方面，Msgpack 和 Protobuf 的总体性能明显优于 Flatbuffers。

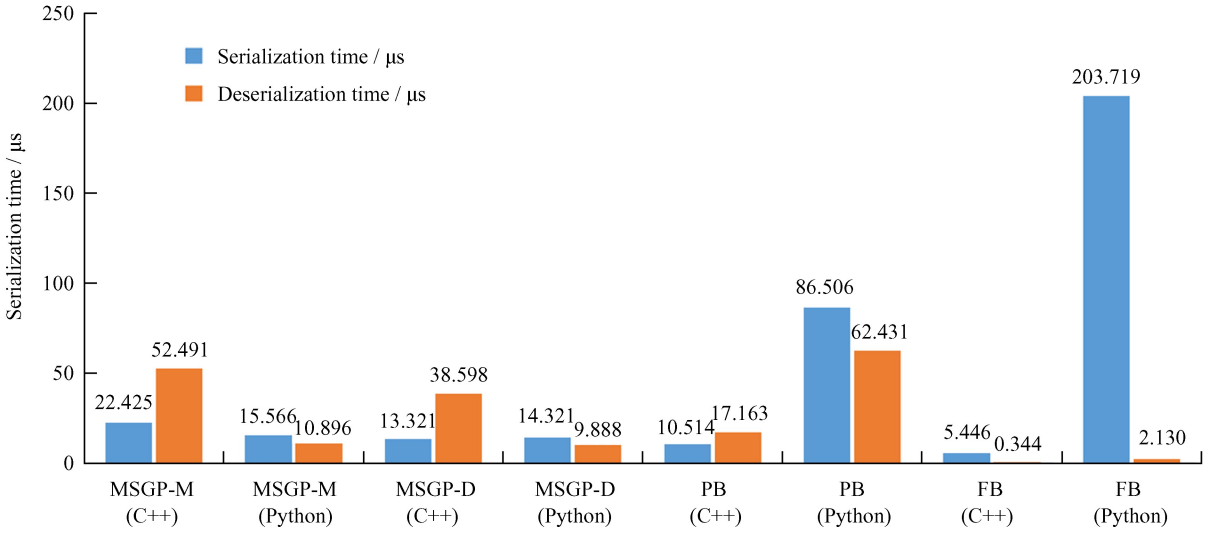

图 10 3 款二进制序列化工具的执行时间

Fig. 10 Execution time for three binary serialization tools

表 1 3 款二进制序列化工具的中央处理器利用率

Table 1 CPU utilization of three binary serialization tools

Name	MSGP-M		MSGP-D		Protobuf		Flatbuffers	
	C++	Python	C++	Python	C++	Python	C++	Python
encode CPU usage/%	12.45	12.44	12.38	12.36	12.37	12.57	12.48	25.9
decode CPU usage/%	12.5	12.37	12.48	12.4	12.47	12.49	11.82	12.21

3 总 结

本文分析比较了 Msgpack，Flatbuffers 和 Protobuf 的编码原理和特性，并对它们进行了测试。Msgpack 不需要 IDL，只需开发人员编写代码实现编解码的功能；而 Flatbuffers 和 Protobuf 使用 IDL 对传输的信息进行编解码。它们对同一信息编解码时，MSGP-D 字节流的大小和多语言的序列化时间优于 Protobuf，且明显优于 Flatbuffers。MSGP-M 对需求变化大的小数据编码具有优势，可以对同一数据以任意顺序的 key-value 数据进行编解码，但 Protobuf 和 Flatbuffers 却不能对这种方式进行解码。根据射电望远镜控制系统的开发情况，当通信的数据格式确定时，可使用 MSGP-D；而通信的数据格式变化较大时，可使用 MSGP-M。总之，通过分析 3 款二进制序列化工具，Msgpack 更适合射电望远镜控制系统的信息传输，有助于射电望远镜的硬件系统、软件系统、操作系统、编程语言和网络之间的信息交换，系统的扩展性好，移植性强。

参考文献：

[1] WANG J, LIU J J, TANG P Y, et al. A study on generic models of control systems of large astronomical telescopes [J]. Publications of the Astronomical Society of the Pacific, 2013, 125 (932): 1265.

[2] GÖTZ A, TAUREL E T, VERDIER P V, et al. TANGO-Can ZMQ replace CORBA? [C]// Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems. 2013: 964-968.

[3] SOMMER H, CHIOZZI G, FUGATE D, et al. Transparent XML binding using the ALMA Common Software (ACS) container/component framework [C]// Proceedings of the ASP Conference Series. 2004, 314: 81-84.

[4] GUZMAN J C, HUMPHREYS B. The Australian SKA Pathfinder (ASKAP) software architecture [C]// Proceedings of SPIE. 2010.

[5] KODILKAR J, UPRADE R, NAYAK S, et al. Developments of next generation monitor and control systems for radio telescopes [J]. Institute of Physics Conference Series, 2013, 44: 012026.

[6] 邓辉, 钟文杰, 付映雪, 等. 基于 ZeroMQ 的新一代望远镜自动控制系统的通信框架设计 [J]. 天文研究与技术, 2018, 15(3): 308-314.
DENG H, ZHONG W J, FU Y X, et al. The design of communication framework for a new generation of telescope autonomous control system based on ZeroMQ [J]. Astronomical Research & Technology, 2018, 15(3): 308-314.

[7] DARADKEH T, AGARWAL A, GOELY N, et al. Real time metering of cloud resource reading accurate data source using optimal message serialization and format [C]// 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). 2018: 476-483.

[8] RIGGI S, BECCIANI U, COSTA A, et al. The design of the local monitor and control system of SKA dishes [C]// Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems. 2016: 472-475.

[9] CLARKE M J, AKEROYD F A, ARNOLD O, et al. Live visualisation of experiment data at ISIS and the ESS [C]// Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems. 2017: 431-434.

Serialization Analysis of Data Transmission in Control System of Radio Telescope

Li Jun[1,2], Wang Na[1,3], Liu Zhiyong[1,3], Song Yining[1,2], Yang Lei[1,2], Wang Jili[1]

(1. Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China, Email: liuzhy@xao.ac.cn;
2. University of Chinese Academy of Sciences, Beijing 100049, China;
3. Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210033, China)

Abstract: The control system can connect, integrate and manage the software and hardware systems of the radio telescope. Serialization tool in the control system encodes and decodes the information transmitted between different devices, operating systems, programming languages, and networks in the radio telescope, enhancing the efficiency rate of data transmission between systems. This article analyzes and compares the coding principles and characteristics of the three binary serialization tools Msgpack, Protobuf and Flatbuffers, and tests their serialized data size, serialization time, and CPU utilization through an example. The results show that the overall performance of Msgpack is better than that of Protobuf and Flatbuffers, and it is suitable for encoding and decoding of transmission information between radio telescope systems with long periods and variable requirements.

Key words: radio telescope; binary serialization tool; control system; encode; decode