Serializing Data with Protocol Buffers

Vinicius Vielmo Cogo
Smalltalks, DI, FC/UL. February 12, 2014.

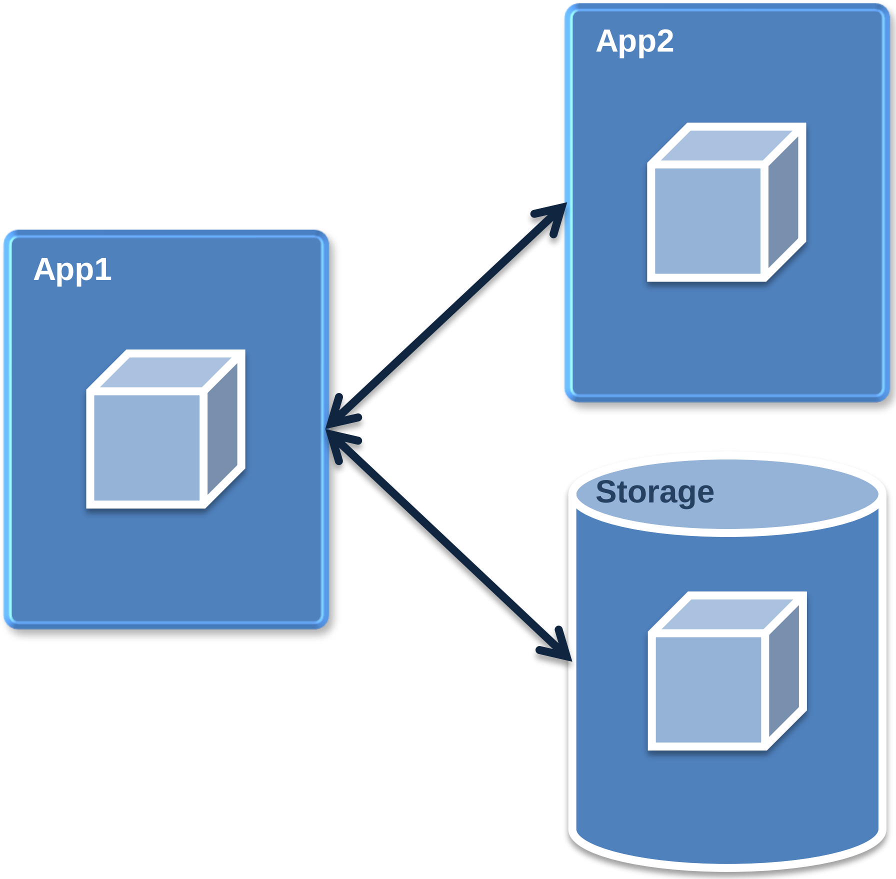
U LISBOA | UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DE LISBOA

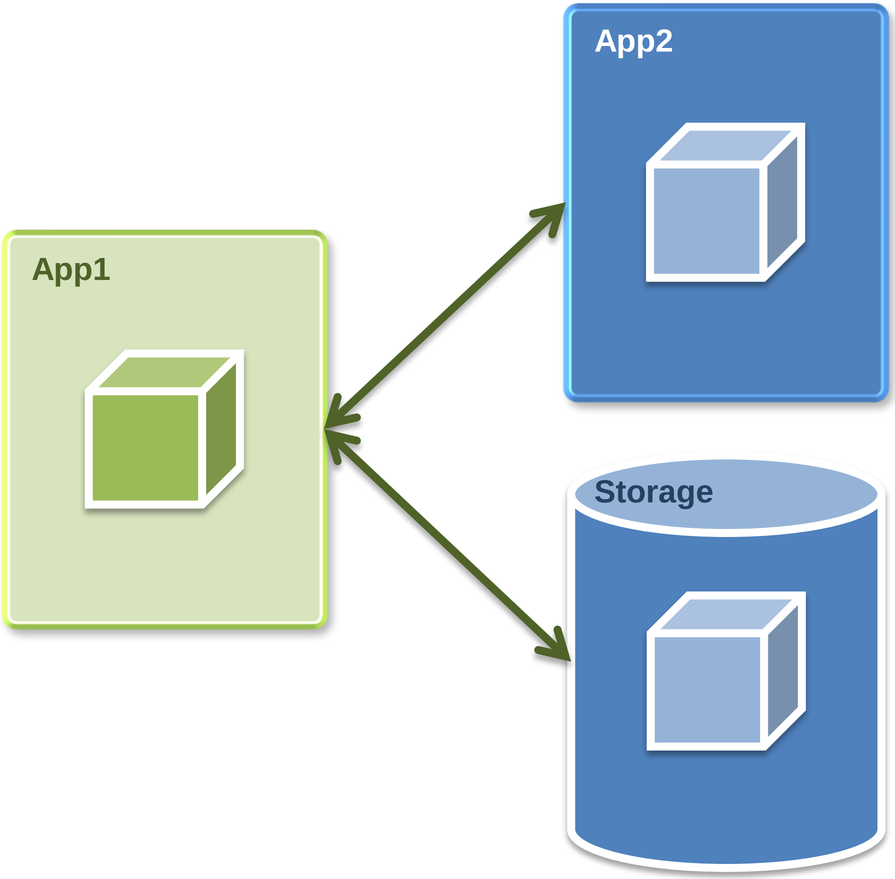
LASIGE
Large-Scale Informatics Systems Laboratory

Navigators

Problem statement



Problem statement



- Correct object
- Efficient (time and space)
- Language interoperable
- Easy to use

Context

- Data serialization (in Java):

1. Java built-in serialization
   - Ex.: ***writeObject/readObject(aObject)***
   - Easy to use, but inefficient in terms of space (extra fields)
   - No language interoperability

2. Manual binary encoding
   - Ex.: 4 ints as *“12:3:-23:67”* and ***aObject.getBytes()***
   - Space efficient, but time efficiency depends on parsing methods
   - Difficult for complex objects
   - Language interoperable

3. Human-readable formats
   - Ex.: Using **XML**, **JSON**, DOM, SAX, STAX, JAXB, JAXP, etc.
   - Inefficient (space and time w/ human readable format)
   - Language interoperable

Protocol Buffers

- Protocol Buffers (Protobuf) is a solution to data serialization:
  - A description language
  - A compiler
  - A library

- Easy-to-use, efficient automatic binary encoding
- Created by and in production at Google Inc.
- Publicly launched in 2008.

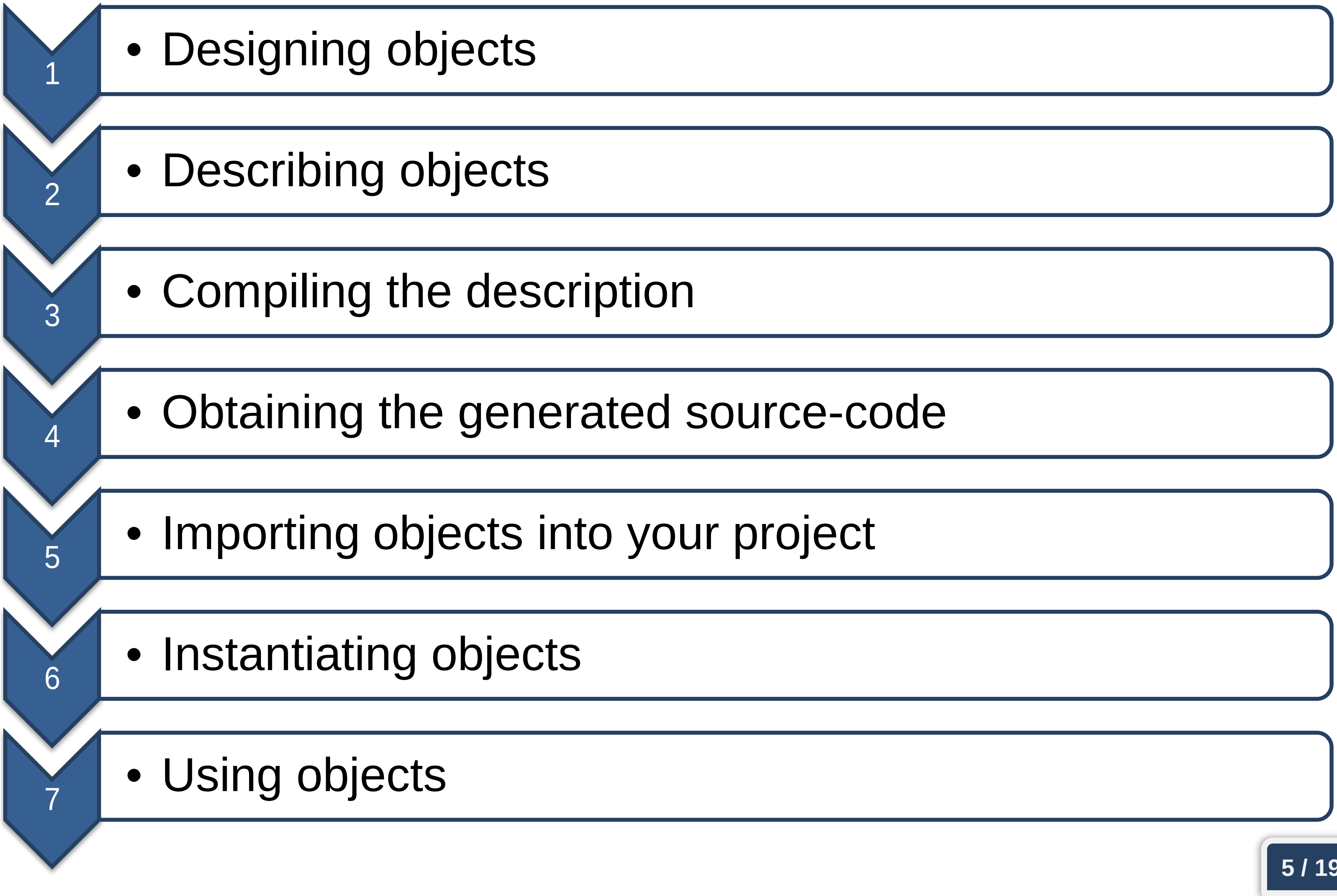
- Language interoperable:
  - Officially: Java, C++, and Python
  - Unofficially: C, C#, Erlang, Perl, PHP, Ruby, etc.

- Download and install the Protocol Buffers

Google Developers

Available at:
https://developers.google.com/protocol-buffers/

Generic workload



1. Designing objects
2. Describing objects
3. Compiling the description
4. Obtaining the generated source-code
5. Importing objects into your project
6. Instantiating objects
7. Using objects

1

- Designing objects

Person:

Id
Name
Age
Email
Phone(s)
…

Protocol Message:

ClientId
Sequence
Operation
Signature
…

Sentence:

Id
Language
Subject
Predicate
Verb(s)
Pronoun(s)
Object
…

Protein:

Id
Organism
Function(s)
Sequence
…

2

- Describing objects

Person:

```
required int32 id
required string name
optional string email
repeated string phone
…
```

Protocol Message:

```
required int32 clientId
required int32 sequence
optional string operation
optional string signature
…
```

Sentence:

```
required int32 id
optional string language
optional string subject
optional string predicate
repeated string verbs
repeated string pronouns
optional string object
```

Protein:

```
required int32 id
optional string organism
repeated string function
optional string sequence
…
```

• Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

2

• Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";
```

Initial configuration

```
message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

2

• Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

message = object

2

- Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

Basic fields

2

• Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

Enumerations

2

- Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

Internal objects

2

Describing objects

addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

Default values

2

• Describing objects

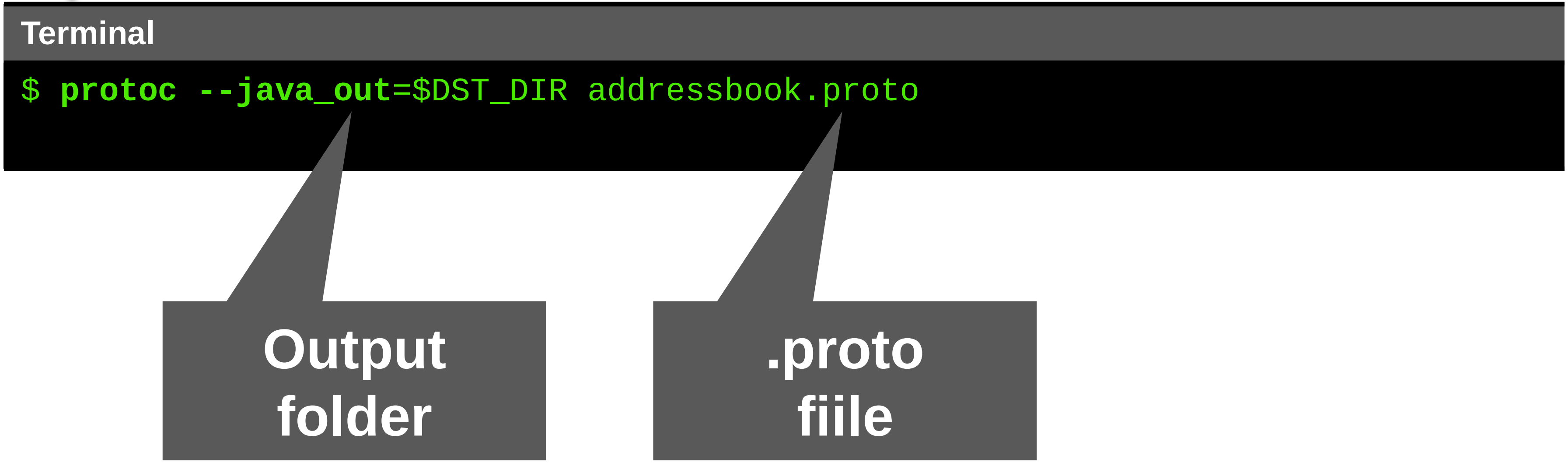
addressbook.proto

```
package tutorial;
option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2 [default = HOME];
        }
        repeated PhoneNumber phone = 4;
}

message AddressBook {
        repeated Person person = 1;
}
```

List of objects

3

- Compiling the description



Terminal

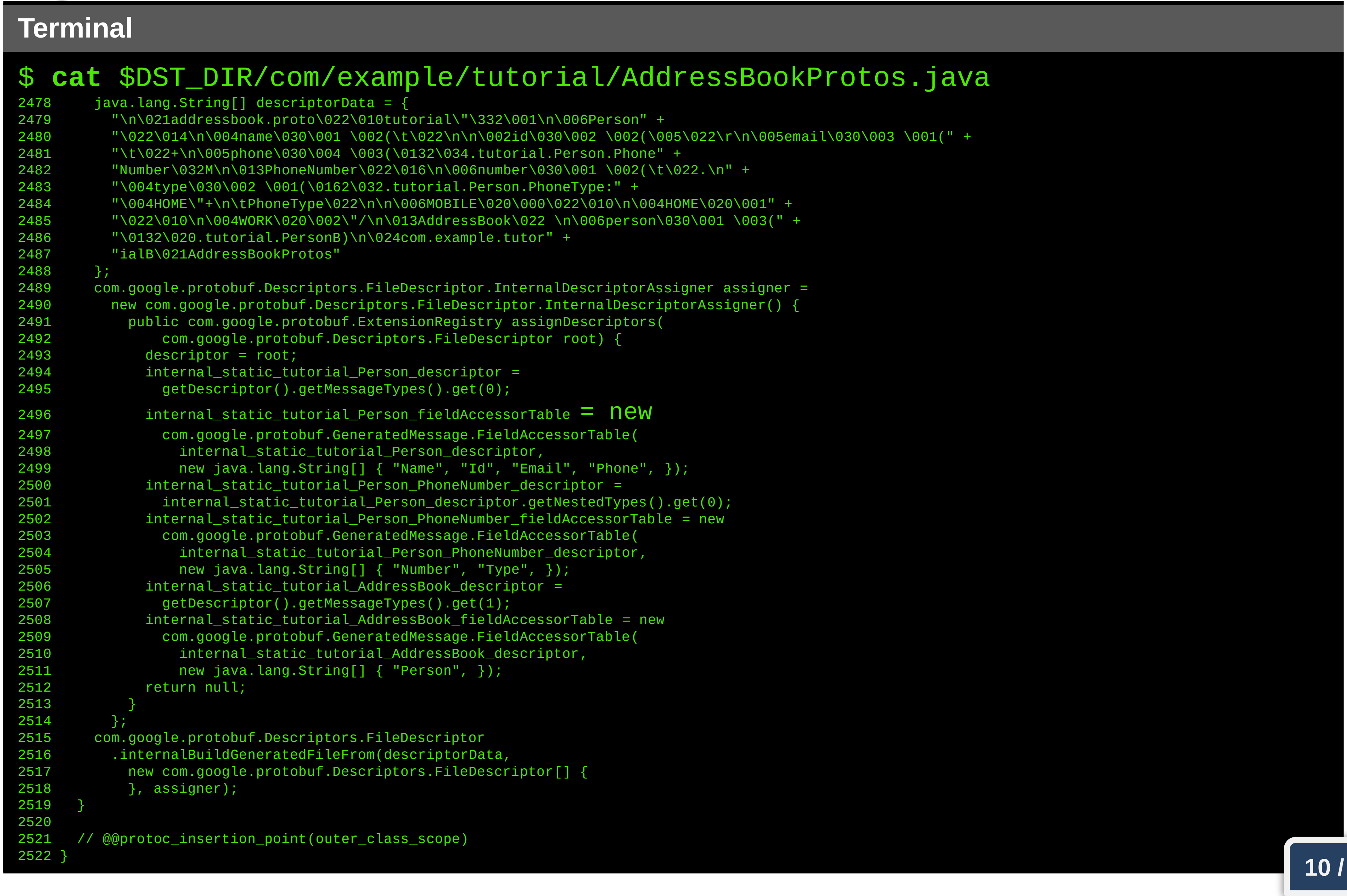
```
$ protoc --java_out=$DST_DIR addressbook.proto
```

Output folder

.proto fiile

4

• Obtaining the generated source-code



Terminal

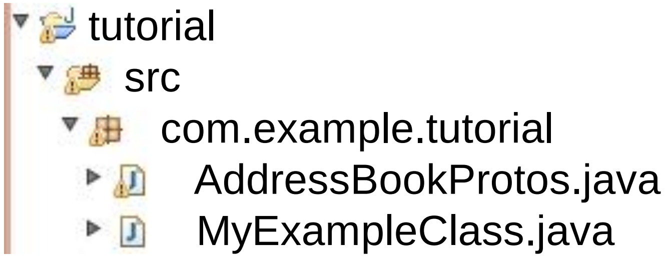
```
$ cat $DST_DIR/com/example/tutorial/AddressBookProtos.java
    java.lang.String[] descriptorData = {
      "\n\021addressbook.proto\022\010tutorial\"\332\001\n\006Person" +
      "\022\014\n\004name\030\001 \002(\t\022\n\n\002id\030\002 \002(\005\022\r\n\005email\030\003 \001(" +
      "\t\022+\n\005phone\030\004 \003(\0132\034.tutorial.Person.Phone" +
      "Number\032M\n\013PhoneNumber\022\016\n\006number\030\001 \002(\t\022.\n" +
      "\004type\030\002 \001(\0162\032.tutorial.Person.PhoneType:" +
      "\004HOME\"+\n\tPhoneType\022\n\n\006MOBILE\020\000\022\010\n\004HOME\020\001" +
      "\022\010\n\004WORK\020\002\"/\n\013AddressBook\022 \n\006person\030\001 \003(" +
      "\0132\020.tutorial.PersonB)\n\024com.example.tutor" +
      "ialB\021AddressBookProtos"
    };
    com.google.protobuf.Descriptors.FileDescriptor.InternalDescriptorAssigner assigner =
      new com.google.protobuf.Descriptors.FileDescriptor.InternalDescriptorAssigner() {
        public com.google.protobuf.ExtensionRegistry assignDescriptors(
            com.google.protobuf.Descriptors.FileDescriptor root) {
          descriptor = root;
          internal_static_tutorial_Person_descriptor =
            getDescriptor().getMessageTypes().get(0);
          internal_static_tutorial_Person_fieldAccessorTable = new
            com.google.protobuf.GeneratedMessage.FieldAccessorTable(
              internal_static_tutorial_Person_descriptor,
              new java.lang.String[] { "Name", "Id", "Email", "Phone", });
          internal_static_tutorial_Person_PhoneNumber_descriptor =
            internal_static_tutorial_Person_descriptor.getNestedTypes().get(0);
          internal_static_tutorial_Person_PhoneNumber_fieldAccessorTable = new
            com.google.protobuf.GeneratedMessage.FieldAccessorTable(
              internal_static_tutorial_Person_PhoneNumber_descriptor,
              new java.lang.String[] { "Number", "Type", });
          internal_static_tutorial_AddressBook_descriptor =
            getDescriptor().getMessageTypes().get(1);
          internal_static_tutorial_AddressBook_fieldAccessorTable = new
            com.google.protobuf.GeneratedMessage.FieldAccessorTable(
              internal_static_tutorial_AddressBook_descriptor,
              new java.lang.String[] { "Person", });
          return null;
        }
      };
    com.google.protobuf.Descriptors.FileDescriptor
      .internalBuildGeneratedFileFrom(descriptorData,
        new com.google.protobuf.Descriptors.FileDescriptor[] {
        }, assigner);
  }

  // @@protoc_insertion_point(outer_class_scope)
}
```

5

• Importing objects into your project

Terminal

```
$ cp $DST_DIR/com/example/tutorial/AddressBookProtos.java
~/workspace/tutorial/src/com/example/tutorial/
```



```
▼ tutorial
  ▼ src
    ▼ com.example.tutorial
      ▶ AddressBookProtos.java
      ▶ MyExampleClass.java
```

MyExampleClass.java

```
package com.example.tutorial;

import com.example.tutorial.AddressBookProtos.AddressBook;
import com.example.tutorial.AddressBookProtos.Person;


…
```

6

- Instantiating objects

MyExampleClass.java

```
…
Person   john = Person.newBuilder()
                .setId(12345)
                .setName(“John Foo Bar")
                .setEmail(“john@foobar.pt")
                .addPhone(Person.PhoneNumber.newBuilder()
                                .setNumber("+351 999 999 999")
                                .setType(Person.PhoneType.HOME)
                                .build())
                .build();
…
```

7

• Using objects: Storage

MyStorageExample.java

```
// Writing data to a file

FileOutputStream  aOutput = new FileOutputStream(“theFilename”);

Person  aPerson = Person.newBuilder().set… //instantiate a Person

aPerson.writeTo(aOutput);

aOutput.close();

// Reading data from a file

Person  aPerson = Person.parseFrom(new FileInputStream(“theFilename”));

// Do something with the received Person
```

7

• Using objects: TCP Communication

MyTCPCommunicationExample.java

```
// Server-side

ServerSocket  aServerSocket = new ServerSocket(10000);

Socket  aConnection = aServerSocket.accept();

Person  aPerson = Person.parseDelimitedFrom(aConnection.getInputStream());

// Do something with the received Person

aPerson.writeDelimitedTo(aConnection.getOutputStream());

// Client-side

Person  aPerson = Person.newBuilder().set… //instantiate a Person

Socket  aSocket = new Socket("127.0.0.1", 10000);

aPerson.writeDelimitedTo(aSocket .getOutputStream());

Person  aPerson = Person.parseDelimitedFrom(aConnection . getInputStream());
// Do something with the received Person
```

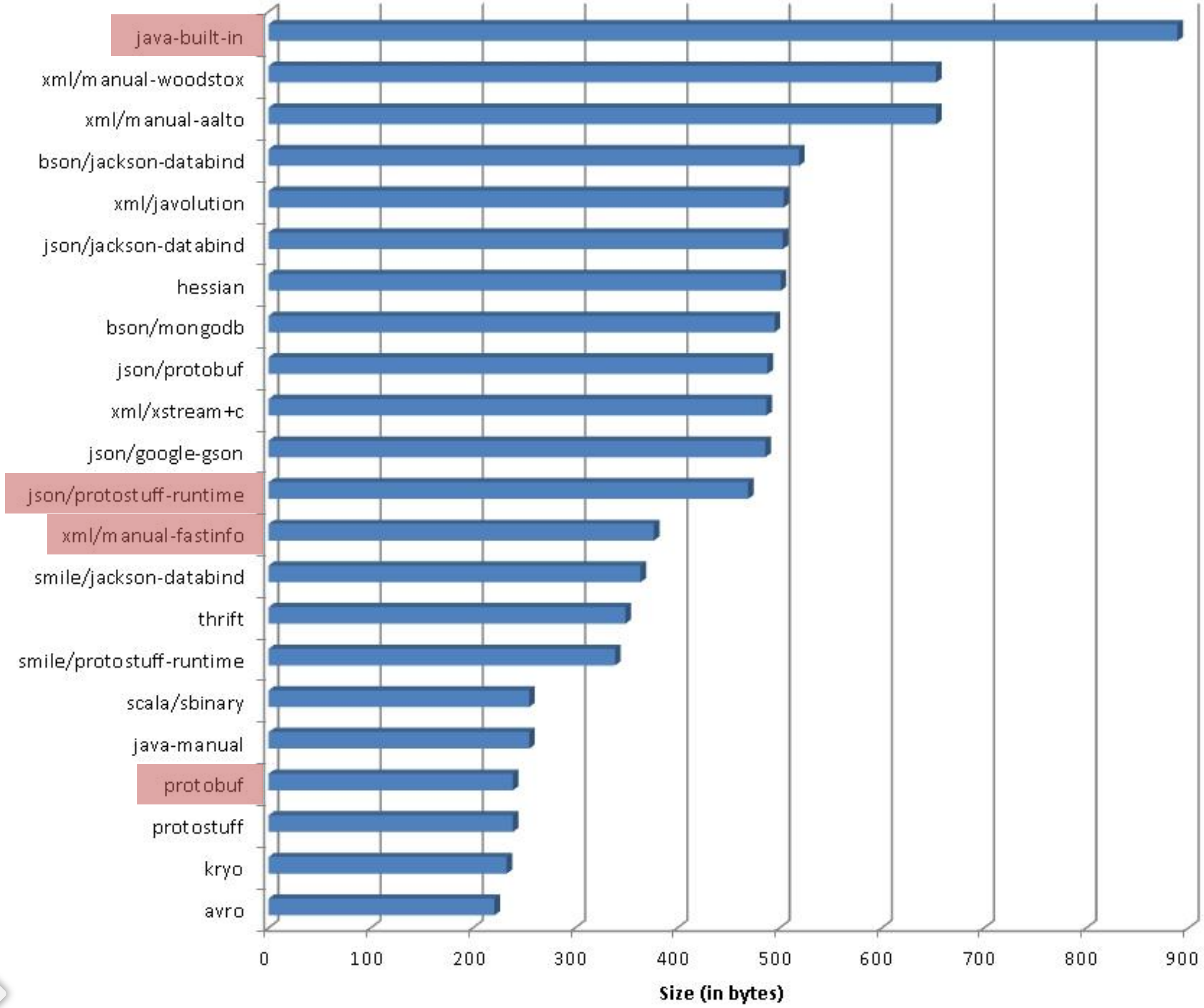
7

• Using objects: UDP Communication

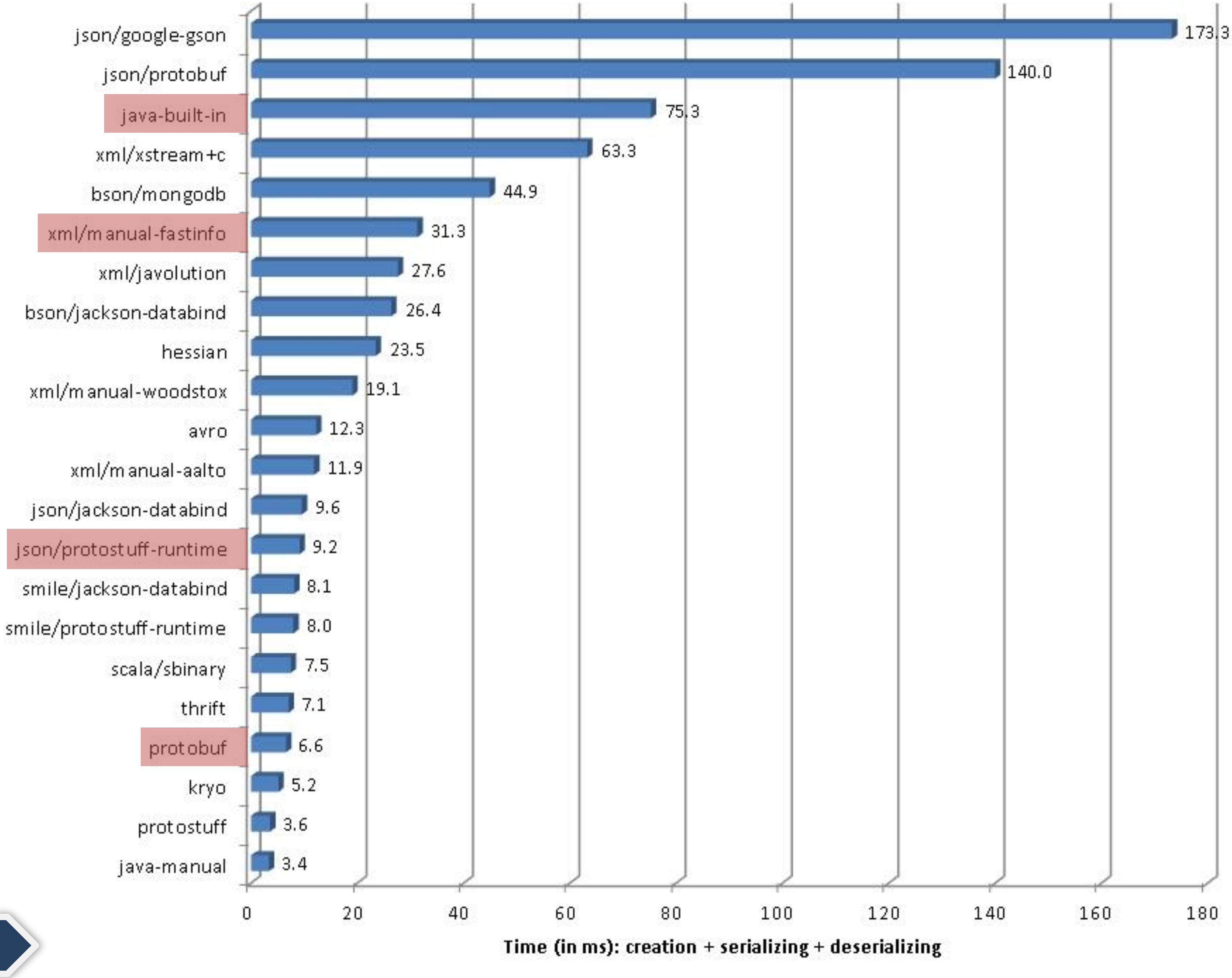
MyUDPCommunicationExample.java

```
// Receive a packet
DatagramSocket   aServerSocket = new DatagramSocket(10000);
byte[]   aReceiveData = new byte[1024];
DatagramPacket   aReceivePacket = new DatagramPacket(aReceiveData, aReceiveData.length);
aServerSocket.receive(aReceivePacket);
ByteArrayInputStream   aInput = new ByteArrayInputStream(aReceiveData);
Person   aPerson = Person.parseDelimitedFrom(aInput);
// Do something with the received Person
// Send a packet
DatagramSocket   aClientSocket = new DatagramSocket(10001);
ByteArrayOutputStream   aOutput = new ByteArrayOutputStream(1024);
Person   aPerson = Person.newBuilder().set… //instantiate a Person
aPerson .writeDelimitedTo(aOutput);
byte   aSendData[] = aOutput.toByteArray();
InetAddress   aIp = InetAddress.getLocalHost(); // or aReceivePacket.getAddress();
DatagramPacket   aSendPacket = new DatagramPacket(aSendData, aSendData.length, aIp, 10001);
aClientSocket.send(aSendPacket);
```

Performance: Size



Performance: Time



Final remarks

- Protobuf focuses on:
  - Efficiency (space and time)
  - Language interoperability
  - Usability

- It is a good alternative for built-in serialization

- But there are other available solutions:
  - Avro (http://avro.apache.org/)
  - Thrift (http://thrift.apache.org/)
  - Kryo (https://github.com/EsotericSoftware/kryo)

Thank You!

Vinicius Vielmo Cogo
Smalltalks, DI, FC/UL. February 12, 2013.

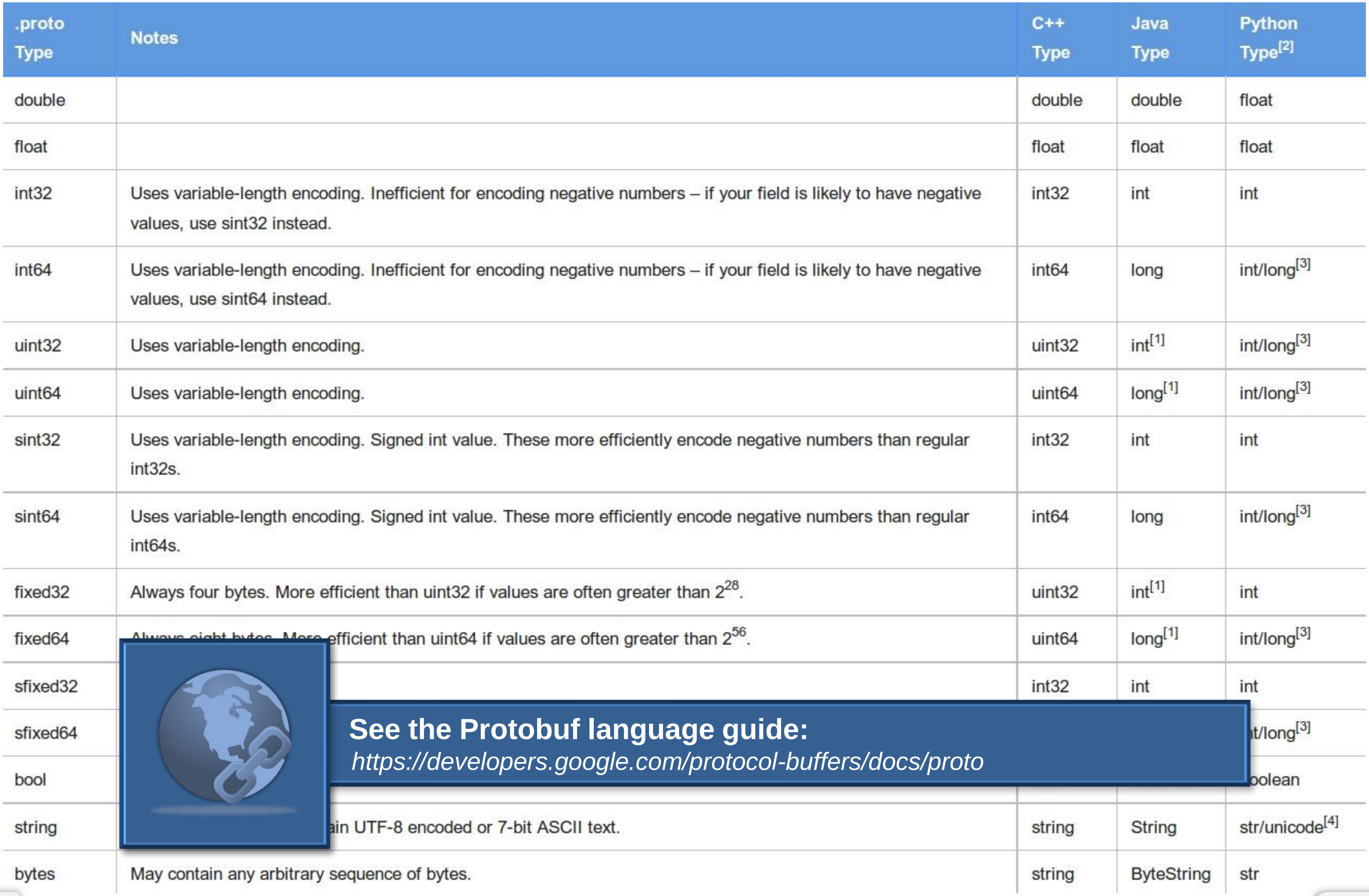
U LISBOA | UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DE LISBOA

LASIGE
Large-Scale Informatics Systems Laboratory

Navigators

Extra 1: .proto types



.proto Type	Notes	C++ Type	Java Type	Python Type[2]
double		double	double	float
float		float	float	float
int32	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint32 instead.	int32	int	int
int64	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint64 instead.	int64	long	int/long[3]
uint32	Uses variable-length encoding.	uint32	int[1]	int/long[3]
uint64	Uses variable-length encoding.	uint64	long[1]	int/long[3]
sint32	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int32s.	int32	int	int
sint64	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int64s.	int64	long	int/long[3]
fixed32	Always four bytes. More efficient than uint32 if values are often greater than 2^{28}.	uint32	int[1]	int
fixed64	Always eight bytes. More efficient than uint64 if values are often greater than 2^{56}.	uint64	long[1]	int/long[3]
sfixed32		int32	int	int
sfixed64				t/long[3]
bool				oolean
string	ain UTF-8 encoded or 7-bit ASCII text.	string	String	str/unicode[4]
bytes	May contain any arbitrary sequence of bytes.	string	ByteString	str

See the Protobuf language guide:
https://developers.google.com/protocol-buffers/docs/proto

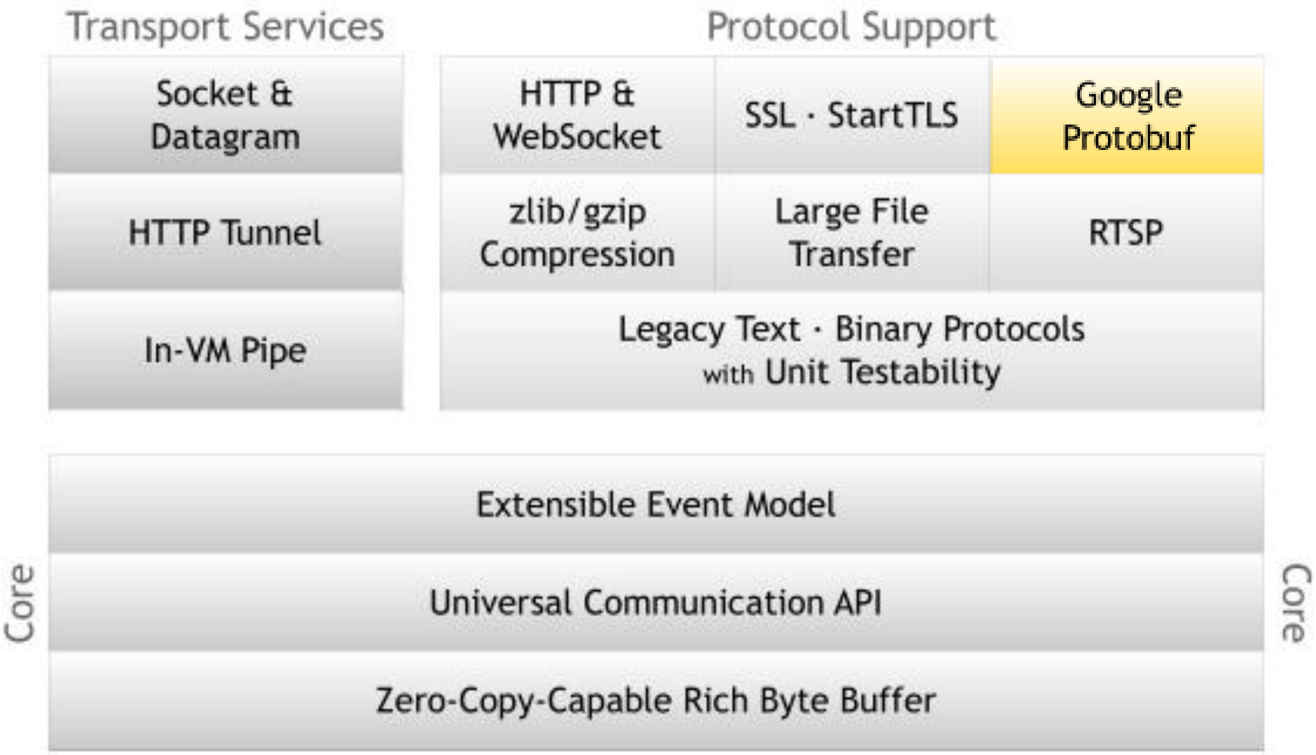
Extra 2: Polymorphism

```
message BaseType
{
    // Reserve field numbers 100 to 199 for extensions.
    extensions 100 to 199;
    // All other field numbers are available for use here.
    required string name = 1;
    optional uint32 quantity = 2;
}

extend BaseType
{
    // This extension can only use numbers 100 to 199.
    optional float price = 100;
}
```

```
message Cat
{
    optional bool declawed = 1;
}
message Dog
{
    optional uint32 bones_buried = 1;
}
message Animal
{
    required float weight = 1;
    optional Dog dog = 2;
    optional Cat cat = 3;
}
```

See the tutorial about Protocol Buffers Polymorphism:
http://www.indelible.org/ink/protobuf-polymorphism/

Extra 3: Netty and Protocol Buffers



See the following Netty packages and examples:
io.netty.handler.codec.protobuf
io.netty.example.worldclock

Extra 4: To delimit or not?

writeTo(OutputStream o)	writeDelimitedTo(OutputStream o)
• Serializes the message • Writes the message to output • No flush() • No close()	• Writes the size of the message • Serializes the message • Writes the message to output

parseFrom(InputStream i)	parseDelimitedFrom(InputStream i)
• Reads the InputStream until EOF • No close()	• Reads the size of message • Reads the message data

Extra 5: Performance

```
record Image = {
   uri: String
   title: String?
   width: Int32
   height: Int32
   size: Size

   enum Size = { SMALL, LARGE, }
}
```

```
record Media = {
   uri: String
   title: String?
   width: Int32
   height: Int32
   format: String
   duration: Int64
   size: Int64
   bitrate: Int32?
   persons: List<String>
   player: Player
   copyright: String?
   enum Player = { JAVA, FLASH, }
}
```

```
record MediaContent = {
   images: List<Image>
   media: Media
}
```

See the following links:

http://code.google.com/p/thrift-protobuf-compare/wiki/BenchmarkingV2
https://groups.google.com/forum/#!forum/java-serialization-benchmarking

Extra 6: General Tips

- Updating a .proto file can be compatible with previous versions

- Prefer optional fields than required (required is forever)

- Generic messages with one “TYPE” field and multiple optional specific messages fields is a good solution for complex protocols

- Same object with different structure are supported through optional fields