Big data small pipes

Dr. John Moore, Antonio D Kheirkhahzadeh, Jiva N Bagale
University of West London
moorejo@uwl.ac.uk

17th September 2012
My background and interests

- Hacker/tinkerer/hobbyist turned lecturer
- Been working with structured data for a number of years
- Interested in out-of-band communication
- Like the challenge of working with constrained networks and constrained devices
- The “Internet of Things”
The Internet of Things

Illustration for TIME by Christoph Niemann
This is what I do

The problem

space required to transport 60 people

car           bus           bicycle

(Poster by city of Münster Planning Office, August 2001) Credit: Press Office City of Münster, Germany
Carpool Kenny

http://www.prankplace.com
Carpool Kenny

http://www.prankplace.com
Working with data not people
Unstructured data

JohnPSmithDirector5119710917MaryTSmithRalphTSmith19571111SusanBJones19590717

You can figure out where most of the data begins and ends, but who is Ralph? We do not know the relationship between any of these values without guessing.

Computers are not very good at guessing.
Adding structure to the data

<?xml version="1.0" encoding="UTF-8" ?>
<personnel>
  <name>
    <givenName>John</givenName>
    <initial>P</initial>
    <familyName>Smith</familyName>
  </name>
  <title>Director</title>
  <number>51</number>
  <dateOfHire>19710917</dateOfHire>
  <nameOfSpouse>
    <givenName>Mary</givenName>
    <initial>T</initial>
    <familyName>Smith</familyName>
  </nameOfSpouse>
  <children>
    <ChildInformation>
      <name>
        <givenName>Ralph</givenName>
        <initial>T</initial>
        <familyName>Smith</familyName>
      </name>
      <dateOfBirth>19571111</dateOfBirth>
    </ChildInformation>
    <ChildInformation>
      <name>
        <givenName>Susan</givenName>
        <initial>B</initial>
        <familyName>Jones</familyName>
      </name>
      <dateOfBirth>19590717</dateOfBirth>
    </ChildInformation>
  </children>
</personnel>

Can you see who Ralph is now?

This is now understandable but we have added a huge amount of overhead to our data by adding on the structure using XML.

But what about data types?
Adding meaning to the data

We can use XML Schema to give our data extra meaning by specifying its data type as well as specifying constraints on values belonging to that type.
So what can we do with this Schema?

- We can validate our data
- We can use the extra knowledge we have to compress the data beyond standard statistical/dictionary based compression techniques
- We can communicate the Schema as a key to decrypt a bit stream
- I have been developing a tool to support all three called Packedobjects
The Packedobjects project

- A tool to encode structured data
- Consists of two components
- libpackedobjects - encoding library
- packedobjectsD - messaging framework built on top of libpackedobjects
- C library based on libxml2
- Can compress XML DOM very concisely
- Uses XML Schema for validation and compression
- Works on mobile and embedded computing devices such as the iPhone

![Diagram of binary numbers]
libpackedobjects: key features

- Very efficient encoding size
- Light-weight and fast
- Good choice of data types for target application domain
- Simple API with two main function calls
- Requires a simple subset of XML Schema to create network protocols
libpackedobjects: limitations

- Not a document compression tool but a data communication tool
- Requires a simple subset of XML Schema to create network protocols
libpackedobjects: application areas

- Internet of Things
- Sensor networks
- Emerging wireless networks
- Network management of the above
libpackedobjects: how it works from a high-level perspective

• Preprocess the XML Schema to allow efficient validation (libxml2)
• Transform the XML Schema to mirror the shape of the XML
• Traverse the XML and use XPath to obtain information from transformed Schema
• The combined data from both the XML and XML Schema is supplied to the low-level encoder
libpackedobjects: how it works from a low-level perspective

- All data to be encoded is mapped from the various data types to an integer within a specific range.
- From this we can calculate the number of bits required to represent this range and then encode the value within these bits.
- All data is encoded to bit-level without padding any byte boundaries.
- The process is an abstraction of Packed Encoding Rules which I refer to as Integer Encoding Rules.
Simple data types

- integer
- boolean
- numeric-string
- hex-string
- bit-string
- string
- octet-string

- null
- enumerated
- decimal
- currency
- ipv4-address
- unix-time
- utf8-string
A sequence is a way of logically grouping data within its own unique namespace. Every element in the sequence must be present and be in the correct order.
sequence with optionality

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="http://zedstar.org/.../packedobjectsDataTypes.xsd"/>
  <xs:complexType name="foobar">
    <xs:sequence>
      <xs:element name="boz" type="integer"/>
      <xs:element name="baz" type="integer" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

In this example only boz is required because baz is now an optional element.
sequence with repeating data

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="http://zedstar.org/.../packedobjectsDataTypes.xsd"/>
  <xs:complexType name="foobar">
    <xs:sequence>
      <xs:element name="boz" type="integer"/>
      <xs:element name="baz" type="integer" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

This is the same as the previous example but now we can have a repeating bar sequence.
The ability to structure data selectively is very powerful. You can easily implement simple network protocols where there is a choice between the different application messages.
Getting started

- git clone git://gitorious.org/libpackedobjects/libpackedobjects.git
- cd libpackedobjects
- autoreconf -i
- ./configure
- make
- make check
- sudo make install

Standard open source autoconf build produces a command line program for testing.
Command-line usage

**encode**

```
./packedobjects --schema foo.xsd --in foo.xml --out foo.po
```

**decode**

```
./packedobjects --schema foo.xsd --in foo.po --out foo.new.xml
```

If you want to examine the performance of the tool you can use the `--loop` command-line flag. This will loop everything including opening and closing files but will only run the initialisation function one time to mirror intended use.
You must first initialise the library using your XML Schema. Typical use would be one call to `init_packedobjects` at startup and then multiple calls to encode/decode based on your protocol. The interface to the `packedobjects_encode` function requires a libxml2 doc type. The `packedobjects_decode` function returns a libxml2 doc type.
Compression

libpackedobjects: size comparison
libpackedobjects: speed comparison

Compression time loop

- PO
- DTDPPM
- XMLPPM
- WBXML
- XMILL
- ZILB

Time (s)

temper-sens, router-qos, sensor, switch-config, purchase-order, router-disc, router-addnet, iprel-ethinfo, iprel-devinfo
libpackedobjects: speed comparison

Decompression time loop
libpackedobjects: size/speed ratio

![Graph showing the speed/size ratio for different objects](image-url)
libpackedobjects vs EXIficient

![Graph showing processing time and percentage of original for different datasets, comparing PO and EXI]
Efficient XML Interchange (EXI)

• Adopted as a recommendation by the World Wide Web Consortium (W3C)
• Similar approach to Packedobjects using a Schema
• Packedobjects is a domain-specific tool whereas EXI seems to target more general use
• IMHO, EXI seems overly complex which can result in a lack of decent FOSS tools
• IMHO, zlib performs well enough for some general cases
• zlib is already available on mobile and embedded platforms such as the iPhone
• Examine on a speed/size ratio graph to determine which performs best
• However, don’t underestimate the significance of the network
• Messaging framework built on ØMQ
• Nodes pub/sub via a broker
• Brokers are uniquely created depending on the schema
• A lookup server is used to manage brokers
• Infrastructure hidden from user - they just use a library
• Client API: init(schema), send/receive, free
Application: The Clashing Rocks (TCR)
Roadmap for Packedobjects

- Need a unit test framework
- User testing on students in October
- Need to fix some memory leaks and add exception handling
- Improve encoding/decoding speeds
- Improve preprocessing of XML Schema
- Release iOS and Android version
- Release TCR app for select platforms
Thanks for listening...

Get in touch via:

- Email: moorejo@uwl.ac.uk
- Twitter: http://twitter.com/jptmoore
- IRC: zedstar@freenode