



Utrecht
University

Exploring the iRODS Native protocol, a hidden gem

Ton Smeele
Utrecht University

Agenda

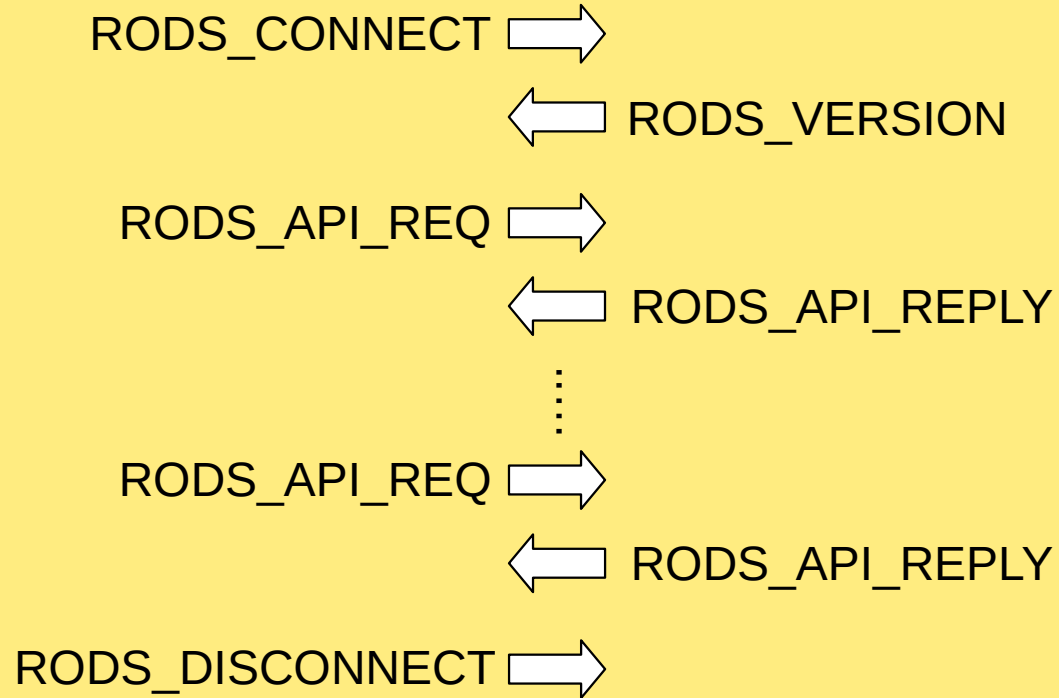
- **Context: the iRODS protocol**
- **Design & Implementation (MyRods) of Native protocol**
- **Performance Benchmark**
- **Application & Summary**

(See UGM proceedings for more in-depth information)

The iRODS communication protocol

- **Based on client/server protocol Postgres95**
 - Developed in 1986
 - Efficient SQL request/response communications
- **Extensively changed for use with SRB and iRODS**
 - 1995 thru 2008, by Michael Wan
 - Exchange of composed data structures
 - Distributed client/server, all grid communications
- **Still going strong!**
 - Flexible and efficient

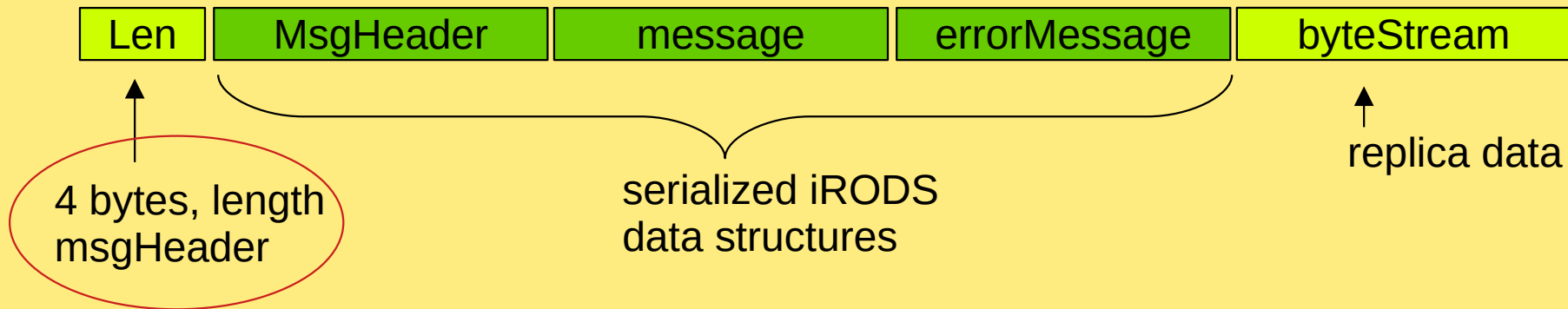
Request/response data packets



Does this log entry look familiar?

```
[SYS_HEADER_READ_LEN_ERR] errno []  
-- message [read 0 expected 4]
```

Common data packet structure



- **iRODS data structures:** `int`, `char`, `str`, `double`, `struct`, ...
 - Composed and named e.g. “`GenQueryOut_PI`” (similar to a schema)
 - Well known by client and server → requires compatible releases!

Exchanged data struct is serialized

- **Native protocol serialization variant**
 - Densely packed binary representation of element values
 - Implemented in the iRODS client library (used by iCommands)
 - IRODS specific, client library has C/C++ binding
- **XML protocol serialization variant**
 - Tagged string, comprises of element names + values
 - Used by all non-C clients (e.g. Jargon, PRC, GO, PRODS)
 - Can reuse existing XML parsing libraries

**How hard is it to implement the Native protocol
in languages other than C ?**

MyRODS layered architecture

use case patterns

- macros for common sequences of API calls

client API and
data structures

- direct request/response workflow
- transform data types <-> iRODS C structures

message
serialization

- serialize the message components
- support for Native and XML variants

client/server connection
and message exchange

- maintain TCP/IP socket connection
- SSL socket layered on top of regular socket
- exchange of serialized messages

Effort to realize Native protocol

- **Total development time (Java implementation)**
 - 60 hours initial design and prototype
 - 4 additional weeks to complete layers and use case patterns
- **A large portion of time was spent on**
 - Extensive iRODS source code inspection (iRODS API is documented inline)
 - Obtain a thorough understanding of protocol specifications (learn its implications via prototyping)

Conclusion: Implementation “client”-side of protocol in languages other than C is perfectly doable!!

Performance benchmark Native / XML

- **Benchmark test executes common API calls**
 - authenticate (native)
 - stat data object,
 - execute general query
 - download replica content
- **Finding: Native messages pack much smaller than XML**
 - Efficiency ratio increases with complexity of data structure (the number of data elements exchanged)
 - Most striking difference in General Query

Partial benchmark results: GenQuery

```
SELECT DATA_NAME, DATA_SIZE WHERE COLL_NAME LIKE "path"
```

Action: (client)	Header Native	Header XML	Message Native	Message XML	Pack ratio Native/XML
GenQuery request	133	134	81	423	38%
Response	142	142	2078	7787	28%
GenQuery Close req.	133	134	81	421	39%
Response	142	142	1116	3789	32%

all serialized component sizes are in bytes

Initial query request returns response with column headings and 100 rows
Close query request returns response with column headings and 0 rows

Partial benchmark results: “iget”

GET_RESOURCE_INFO_FOR_OPERATION, REPLICA OPEN, DATA OBJ READ, DATA OBJ CLOSE

Action: (client)	Header Native	Header XML	Message Native	Message XML	Pack ratio Native/XML
get resc	136	136	112	328	53%
(response)	140	141	76	150	74%
open	136	136	104	321	53%
(response)	142	142	1711	2345	75%
read	133	134	36	218	48%
(response)	141	141	0	0	100%
close	133	134	36	212	49%
(response)	139	139	0	0	100%

MyRods put to use: ipump

- **Use case: migrate data between nonfederated zones**
 - Recursively copies data objects
 - Resumable, command-line type application, runs in tmux
- **IRODS API-level programming**
 - Macro's for use case patterns help a lot (authenticate, data transfer)
 - Sufficiently convenient as basis for application development
 - Feels like a racing car - iRODS on steroids

Summary: Native is a hidden gem

- **Native protocol requires only 25-50% (!) of network bandwidth compared to the XML variant**
- **Deserves more language bindings besides C/C++!**
 - Implementation is a matter of weeks
- **Powerful, flexible and highly efficient**
 - The Native protocol is competitive, even decades after its inception!

MyRods source code available at: <https://github.com/tsmeele/MyRods>