

Kodo - Cross-platform Network Coding Software Library

Morten V. Pedersen - Aalborg University /
Steinwurf ApS
mvp@es.aau.dk

Background

Academia

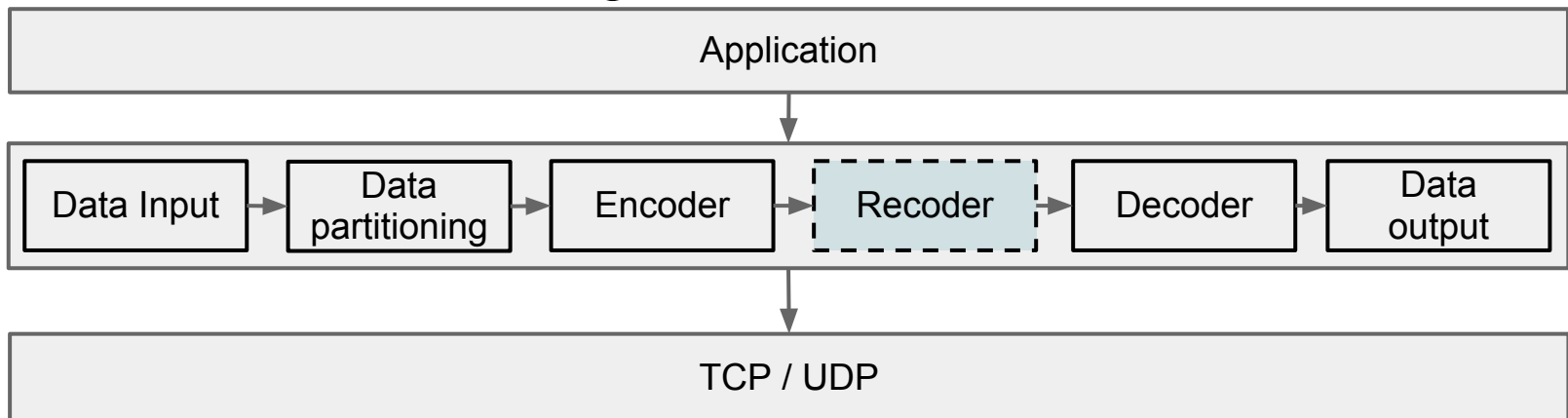
- Network coding key enabler for efficient user cooperation (p2p).
- Kodo developed during a 3 year **research project** CONE (COoperation and NEtwork Coding). *Concluded 2012.*

Industry

- On campus start-up **Steinwurf ApS** founded in 2011.
- Taking over the rights for Kodo and development.
- Library **source code** fully available. Licenses:
 - a. Free for Research / Educational
 - b. Paid Commercial

Kodo's Position

- Many different requirements
 - Deterministic vs. random, inter- vs. intra-flow, physical to application / transport layer.
- Current versions of Kodo implement
 - Software & Digital **Random Linear Network Coding** (RLNC)
 - Suitable for **transport / application layer** protocol implementations
 - Focus on the coding



Kodo (the library)

- C++11 (*staying compatible with major compilers*).
- Designed to allow for **easy experimentation** and a high degree of **code reuse**.
- Very flexible design technique used called "**mixin-layers**" or "**parameterized inheritance**" using C++ templates.
- Low-level = ample ways of shooting yourself in the foot. With **API specs**, we try to mitigate this.
- **High Performance** - code generated by compiler comparable to single monolithic implementation.
- Helper libraries.
 - Resource management
 - Finite Fields

Network Coding Algorithms (Kodo v7)

Don't pay for what you don't use!

```
1.  /// A basic RLNC encoder. This type of RLNC encoder
2.  /// transmits the entire encoding vector as part of the
3.  /// encoded payload. It therefore allows recoding at
4.  /// intermediate nodes in a network.
5.  template<class Field>
6.  class full_rlnc_encoder
7.      : public payload_encoder<
8.          systematic_encoder<
9.              zero_symbol_encoder<
10.                 full_vector_encoder<
11.                    linear_block_vector_generator<block_cache_lookup_uniform,
12.                    linear_block_encoder<
13.                    finite_field_math<fifi::default_field_impl,
14.                    symbol_storage_shallow_partial<
15.                    has_bytes_used<
16.                    has_block_info<
17.                    final_coder_factory_pool<full_rlnc_encoder<Field>, Field>
18.                        > > > > > > > > >
19.  {};
```

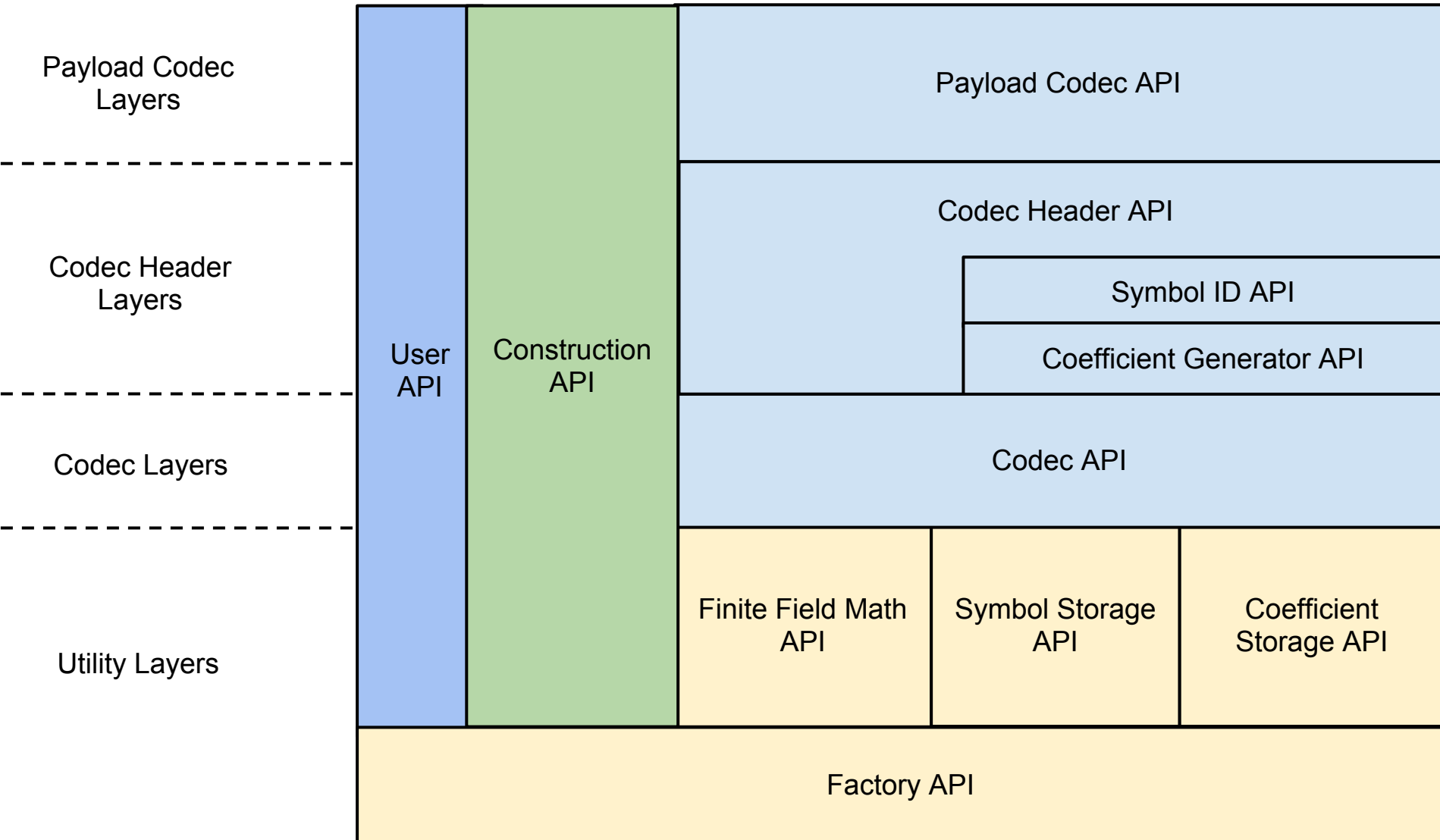
Reed Solomon Encoder/Decoder (Kodo v7)

```
1.  template<class Field>
2.  class rs_encoder
3.      : public payload_encoder<
4.          zero_symbol_encoder<
5.              reed_solomon_encoder<vandermonde_matrix,
6.                  linear_block_encoder<
7.                      finite_field_math<fifi::default_field_impl,
8.                          symbol_storage_shallow_partial<
9.                              has_bytes_used<
10.                                  has_block_info<
11.                                      final_coder_factory_pool<rs_encoder<Field>, Field>
12.                                          > > > > > > >
13.  {};
```

Only added layer is on **line 5** - everything else is reuse!

Kodo v8

Typical Codec Stack



Kodo v8 Documentation

Manual

Kodo documentation — Kodo master documentation - Google Chrome

https://kodo.readthedocs.org/en/latest/

Kodo master documentation

next Index

Project Versions

- latest
- 6.0.0
- 5.0.0
- 4.0.0
- 3.0.1
- 2.0.1

RTD Search

Full-text doc search.

Table Of Contents

- Kodo documentation
- Documentation

Next topic

Introduction

This Page

- Show Source
- Show on GitHub

Kodo documentation

Welcome to the Kodo Network Coding library documentation. Kodo is a C++ library for Network Coding, intended to be used for commercial applications and for research on implementation of Network Coding. The library enable researchers to implement, new codes and algorithms, perform simulations, and benchmark the coding operations on any platform where a C++ compiler is available. The library provide a multitude of build blocks and parameters that can be combined in order to create codes. To ensure ease of use several codes are predefined, and high level API's provided.

Documentation

- Introduction
 - Features
 - Platforms
 - Tools Needed
 - Download the sources
 - Waf (build system)
 - Quick Start (building Kodo examples and unit tests)

Kodo: API Layers - Google Chrome

file:///home/mvp/dev/steinwurf/kodo/doxygen/html/group_api_layer.html

Kodo

Main Page Related Pages Modules Namespaces Classes Files Examples Search

API Layers

Provides an overview of the APIs implemented by different components in Kodo. More...

Modules

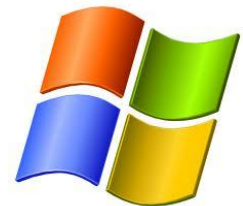
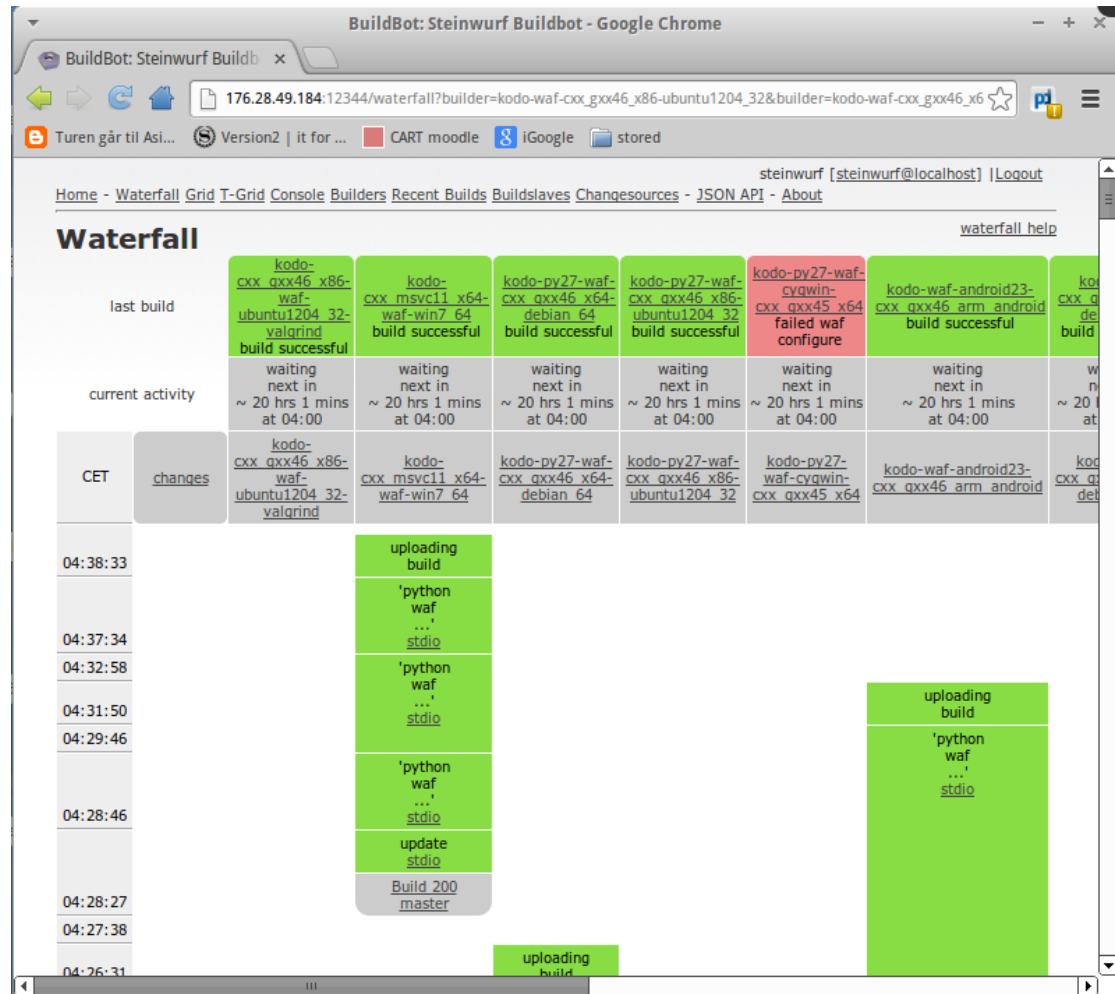
- Factory**
The factories are responsible for construction and initialization of encoders and decoders.
- Finite Field Math**
The finite field math layers perform computations on symbols and symbol coefficients.
- Coefficient Generator**
Responsible for generating the coding coefficients.
- Symbol IDs**
The symbol id describes how an encoded symbol has been produced.
- Coding Info**
Information about encoders and decoders.
- Symbol Storage**
Handles storage of encoding and decoding symbols.
- Coefficient Storage**
Handles storage of coding coefficients.
- Codec**
Implements encoding and decoding algorithms.
- Codec Header**
Implements header information to the coding symbols.
- Payload Codec**
The payload layer provides users of an encoder or decoder with a convenient API.

Generated on Mon Mar 11 2013 01:55:56 for Kodo by [doxygen](#) 1.8.3.1

API

Kodo Testing

- **Continuous Integration** (build on every commit)
- Different platforms & compilers
- Core part of our **release management**



Kodo Performance

Main thing:

- Measure **raw coding speed**.
- Catch **performance regressions**
- Research / Experimentation
 - Memory access patterns
 - Finite field operations
- Prove / test a clever algorithm

```
Terminal -.mvp@mvp-t410: ~/dev/steinwurf/kodo/build/linux/benchmark/throughput - + x
File Edit View Terminal Go Help
.mvp@mvp-t410:~/dev/steinwurf/kodo/build/linux/benchmark/throughput$ ./kodo_throughput
[ RUN      ] FullRLNC.Binary (5 runs)
[ DONE    ] FullRLNC.Binary (55.600000 iterations per run)
[ CONFIG  ] symbol_size=1600, symbols=16, type=encoder
[ RUNS    ] Average result: 1393.468223 MB/s
                Max: 1399.589964 MB/s (+6.121741 MB/s / +0.439317 %)
                Min: 1376.882456 MB/s (-16.585766 MB/s / -1.190251 %)
[ RUN      ] FullRLNC.Binary (5 runs)
[ DONE    ] FullRLNC.Binary (453.400000 iterations per run)
[ CONFIG  ] symbol_size=1600, symbols=16, type=decoder
[ RUNS    ] Average result: 1355.655070 MB/s
                Max: 1361.776024 MB/s (+6.120954 MB/s / +0.451513 %)
                Min: 1349.588397 MB/s (-6.066673 MB/s / -0.447509 %)
[ RUN      ] FullRLNC.Binary (5 runs)
[ DONE    ] FullRLNC.Binary (15.000000 iterations per run)
[ CONFIG  ] symbol_size=1600, symbols=32, type=encoder
[ RUNS    ] Average result: 736.873388 MB/s
                Max: 745.196973 MB/s (+8.323584 MB/s / +1.129581 %)
                Min: 729.552579 MB/s (-7.320809 MB/s / -0.993496 %)
[ RUN      ] FullRLNC.Binary (5 runs)
[ DONE    ] FullRLNC.Binary (143.800000 iterations per run)
[ CONFIG  ] symbol_size=1600, symbols=32, type=decoder
[ RUNS    ] Average result: 756.623446 MB/s
                Max: 763.538446 MB/s (+6.915001 MB/s / +0.913929 %)
                Min: 748.899028 MB/s (-7.724418 MB/s / -1.020906 %)
[ RUN      ] FullRLNC.Binary (5 runs)
[ DONE    ] FullRLNC.Binary (4.000000 iterations per run)
[ CONFIG  ] symbol_size=1600, symbols=64, type=encoder
[ RUNS    ] Average result: 380.051161 MB/s
```

Kodo and the IRTF NWCRG

- Provides a solid building block for
 - Protocol development.
 - Experimentation with different code variants.
- It is well tested.
- It has traction:
 - New features
 - Supported platforms
 - Several University projects using it.
- Version 8 soon to be released bring much better documentation.

The End

- Questions?
- Contributions + bug fixes please
 - Simple procedure with sign-off
- Feedback / comments / questions are all very welcome!

Morten V. Pedersen
mvp@es.aau.dk

Getting started

- Code
 - <http://github.com/steinwurf/kodo>
 - See example of encode/decode in the examples folder
- Documentation (we are working on it)
 - <http://readthedocs.org/docs/kodo/en/latest/>
- Status buildbot: <http://176.28.49.184:12344/>