Implementation and evaluation of the CBOR protocol

Pavel Kalvoda
Supervisor: David Bednářek, Ph.D.

Motivation
Concise Binary Object Representation (CBOR, RFC 7049) is a new binary schema-less data serialization format. How does it compare to the alternatives in terms of:
- speed
- efficiency
- functionality?
Since there was no stable and feature-complete implementation available, creating one was necessary.

libcbor for C

libcbor, our C implementation of CBOR, exposes the features of CBOR either through the high-level API or through the streaming API.

The high-level API enables effortless encoding and decoding while also providing powerful manipulation routines and a convenient reference-counting memory management scheme.

When dealing with CBOR on this layer, all the protocol details are handled automatically and the user is presented with a logical data model where data items can be built, modified, encoded, and decoded.

The streaming API offers flexible incremental encoding and decoding capabilities. Owing to the omission of memory representation and an optimized implementation, it provides high throughput without sacrificing safety or robustness.

In order to use the streaming parser, one has to simply register the callback functions that will then be invoked as the matching structures are identified in the input data. This mechanism also allows creating custom CBOR decoders.

Conclusions - libcbor

Overall, libcbor is a stable, robust, production-ready implementation of CBOR. It has proven itself to be more than sufficient for investigating CBOR and comparing it to similar alternatives.

Based on the benchmarks, our implementation is significantly faster than the relevant alternatives. However, one key finding is that many of the formats and implementations remain mutually incomparable due to the fundamentally different philosophies and design goals.

The Ruby binding has exceeded the expectations and is almost trivial to use, as shown in the example. We believe it will be of significant value to professionals seeking to experiment with CBOR.

libcbor has already become fairly popular with the wider public and has even been praised as the most complete implementation yet by C. Bormann, the author of CBOR.

Future work will therefore undoubtedly include continued support and improvements of libcbor in collaboration with the OSS community.

libcbor for Ruby

A libcbor binding for the Ruby programming language has also been implemented as a part of our work. It enables seamless access to all the libcbor features from Ruby.

Both the high-level and the streaming APIs are exposed to the client. As the primary goal for the Ruby binding is to be as easy to use as possible, an additional adaption layer has been introduced. It provides a more idiomatic Ruby interface and renders libcbor remarkably easy to use.

The structure of the FFI mechanism and the relation between Ruby objects and the native data structures is illustrated in Figure 1.

Conclusions - CBOR

Based on both the theoretical inquiry and the experimental measurements, we have arrived at the following conclusions:
- CBOR can be implemented in a variety of ways, with performance comparable to that of msgpack-c.
- The encoding is between 13% and 48% more efficient compared to JSON. The main efficiency gain is likely to be obtained from the ability to embed binary resources (hence decrease network communication overload) in networking scenarios. This alone might not justify migration to CBOR for existing applications.
- Not all the design goals outlined by the standard specification have been met. Improvements to the format specification have been proposed. These mostly concern the unnecessary complexity and the lack of functionality in several areas.
- CBOR will likely be used in emerging domains such as IoT and mobile computing.

On the whole, CBOR does offer an innovative combination of properties that do set it apart from the existing alternatives. However, those properties will not always be of significant value to existing applications. The future of CBOR thus remains largely unpredictable.

Both the analyses and the implementation provide a solid foundation for further investigations, be they theoretical or experimental.