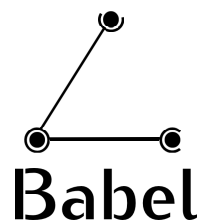
Mandatory sub-TLVs
draft-ietf-babel-rfc6126bis-03

17 July 2017



draft-ietf-babel-rfc6126bis

Babel was defined back then:

- RFC 6126, base protocol, 2011, Experimental;
- RFC 7557, extension protocol, 2015, Experimental.

One of the (current) goals of the Babel WG is to merge and improve these documents in order to produce rfc6126bis.

Changes in rfc6126bis

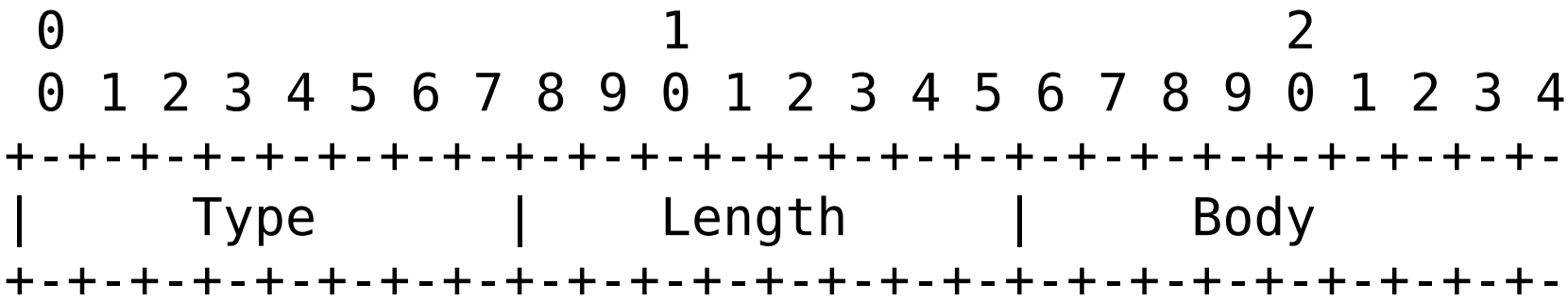
Rfc6126bis makes four substantial changes to Babel:

- mandatory sub-TLVs, implemented, incompatible (this talk);
- unicast Hellos, unimplemented?, incompatible (David's talk);
- relax route acquisition, implemented, compatible (please review);
- relax hold time, unimplemented, compatible (please review and implement).

Compatibility issues are discussed at the end of this talk.

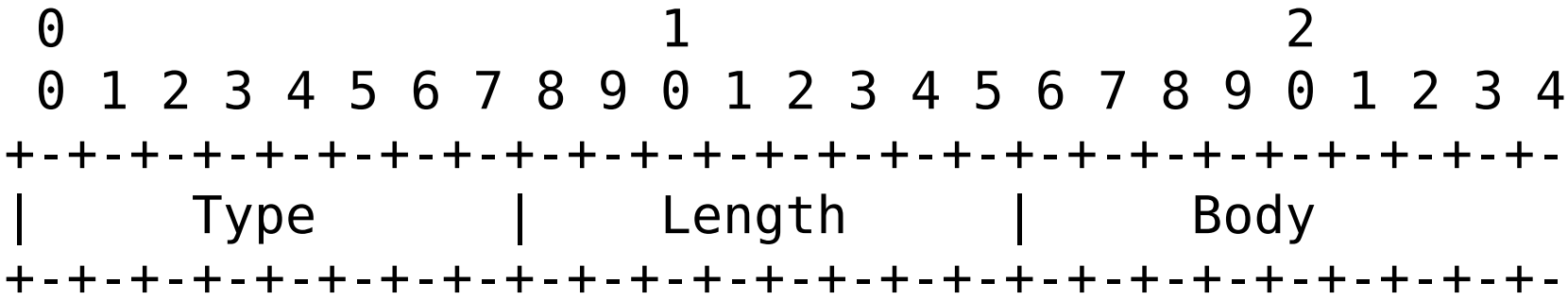
Sub-TLVs

RFC 6126 defines space for extension data in TLVs:



Any extra data at the end of the Body silently ignored: stick extension data there.

RFC 7557 defines the format of extension data: sub-TLVs:



If a sub-TLV is not understood, it MUST be silently ignored.

The need for mandatory sub-TLVs

Sometimes, if extension data is not understood, the whole enclosing TLV needs to be ignored. This is not possible with sub-TLVs.

To avoid the issue, the source-specific extension uses three new TLVs:

- source-specific update;
- source-specific request;
- source-specific seqno request.

Why not a new AE

Idea: use a new Address Encoding (AE) number, which will be silently ignored by existing implementations.

Received cautiously by the WG in Chicago.

To check whether I was right:

- Matthieu Boutier tried designing a new variant of source-specific routing using a new AE;
- Gwendoline Chouasne tried designing an extension for ToS-specific routing using a new AE.

It turned out that I was wrong: using a new AE is a mess. Gory details in Appendix C of rfc6126bis.

Mandatory sub-TLVs

In rfc6126bis, if a sub-TLV has a Type between 128 and 255 it is mandatory. (The most significant bit is called the mandatory bit.)

An unknown mandatory sub-TLV causes the whole enclosing TLV to be ignored.

Works very nicely for:

- source-specific routing (Matthieu Boutier);
- ToS-specific routing (Gwendoline Chouasne).

Both extensions are:

- implemented and
- written down in I-Ds.

Mandatory sub-TLVs and compression

Babel uses a stateful parser: parsing a TLV causes the parser state to be updated, and a later TLV may refer to the parser state.

Parsing a TLV unconditionally updates the parser state even if it was ignored due to a mandatory sub-TLV.

This is essential in order to have deterministic parsing. (Think tcpdump.)

Implementation complexity

Minor increase in implementation complexity:
an implementation MUST parse sub-TLVs in order to check the mandatory bit even if doesn't otherwise use sub-TLVs.

Example: sbabeld:

- source grew by 38 lines of C,
- binary size increased by 56 octets.

Compatibility with RFC 6126

Mandatory bits are incompatible with RFC 6126: old implementations will not ignore TLVs with unknown mandatory sub-TLVs. This could break your network. This is also true of unicast Hellos.

Two solutions:

- don't use the new extensions until you've upgraded all of your routers ("don't do that, then");
- bump the protocol version on packets that contain mandatory sub-TLVs (or unicast Hellos), but not on others
  (MUST accept both versions, MUST send version 3 if mandatory sub-TLV or unicast Hello, MAY send version 2 otherwise).

Conclusion

Mandatory sub-TLVs are a simple, elegant feature that drastically simplifies some extensions.

They are incompatible with RFC 6126, and might break your network if you enable the new extensions without upgrading old routers.

This could be avoided by bumping the protocol version.