Babel
Internet-Draft
Intended status: Informational
Expires: October 7, 2016

T. Hoeiland-Joergensen
Karlstad University

April 05, 2016

Experiences implementing the Babel routing protocol
draft-hoeiland-joergensen-babel-implementation-00

Abstract

This document describes the experiences creating independent implementations of the Babel routing protocol [RFC6126], based on the two independent implementations currently available.

The Babel routing protocol is easy to implement, and it is quite feasible for someone with no prior experience with routing protocols to create a working implementation. The few issues that needed clarification during the independent implementation work are documented, with the hope of aiding others seeking to implement Babel.

Table of Contents

1. Introduction 2
2. Overall experiences 3
3. Specific issues encountered 3
3.1. Port number 3
3.2. Metric functions 3
3.3. Sequence number space 3
3.4. Error handling on packet parsing 3
3.5. Router ID issues 4
3.6. The need for blackhole routing 4
3.7. Extra space at the end of TLVs 4
3.8. Imprecise language concerning IDs 4
4. In summary 4
5. References 5
5.1. Normative References 5
5.2. Informative References 5
Author's Address 5

1. Introduction

Babel [RFC6126] is a distance vector routing protocol especially suited for hybrid networks (with both stable and unstable parts) [I-D.chroboczek-babel-applicability]. The reference implementation (Babeld, available from http://www.pps.univ-paris-diderot.fr/~jch/software/babel/) is released as open source and has been deployed in several places. However, having separate clean-slate implementations of the protocol from the specification is an important part of the work to standardise the protocol. This document is an attempt to collect the experiences of implementing Babel from the RFC, to serve as information for others wanting to implement the protocol.

There currently exists two independent implementations of the protocol, Pybabel by Markus Stenberg (https://github.com/fingon/pybabel/) and a patch to the Bird routing daemon by Toke Hoeiland-Joergensen (http://trubka.network.cz/pipermail/bird-

users/2016-April/010300.html). This document is based on the experiences from these two implementations.

2. Overall experiences

Overall, the Babel protocol is remarkably easy to implement. The specification is just over 40 pages, of which around 30 (sections 3 and 4 of [RFC6126]) contain the details required for an implementation and the rest is background and supporting text. This means that it is possible to understand the protocol and keep it present in one's mind in its entirety. Indeed, a feeling of "really, that's it?" presents itself in the early stages of the work with the protocol.

The two independent implementations were created in a few days and a few weeks, respectively. And while none of them are as complete as the reference implementation (for instance, none of them currently implement any extensions, and both are IPv6-only), this attests to the ease with which a new implementation can be written.

3. Specific issues encountered

This section documents the issues that needed clarification during the implementation work. All of these issues were resolved on the public mailing lists during the work with the implementation, but they are collected and documented here to help future implementers.

3.1. Port number

The port number specified in [RFC6126] is wrong; an errata exists that changes the port number to the right one (6696), but this can easily be missed.

3.2. Metric functions

The base specification does not include any metric functions. Workable functions are specified in the RFC appendix, however, and using those leads to a workable and interoperable implementation.

3.3. Sequence number space

The handling of sequence number space is not specified in the RFC. The sequence numbers should be computed by module arithmetic as specified by [RFC1982], but no reference exists for this in [RFC6126].

3.4. Error handling on packet parsing

The Babel specification does not specify how to handle parsing errors when processing incoming packets. For instance, some TLVs have implicit dependencies on earlier TLVs in the same packet (such as

Update TLVs depending on earlier Router ID TLVs), and so continuing processing if the earlier TLV fails to parse is likely to lead to errors. Some care is needed when implementing this logic, and [RFC6126] does not offer a lot of guidance.

3.5. Router ID issues

The Babel specification includes a mechanism to infer router IDs from prefixes in an Update TLV; however, it does not specify how to do this for IPv4 addresses. In addition, while a mechanism to generate router IDs is specified that is unlikely to generate duplicates, this is optional and no mechanism for detecting or reacting to duplicate IDs is specified. Finally, when writing an implementation it is easy to assume that one can simply check for an empty router ID to discover if a router ID TLV has been seen; however, an all-zero router ID is not in fact disallowed by the specification, and so an extra explicit flag is needed in the parsing state.

3.6. The need for blackhole routing

When a route is retracted, Babel temporarily installs it as an explicit unreachable route in the kernel routing table. This is done to avoid routing loops when a more specific route fails. The reason for this mechanism is explained in section 2.8 of [RFC6126]. However, since this is in the background section of the document, one is not likely to refer to that after an initial reading. And since this practice is somewhat unusual, it can lead to confusion.

3.7. Extra space at the end of TLVs

The Babel specification explicitly allows extra space at the end of TLVs, and uses this for its extension mechanism (specified in [RFC7557]). However, this fact can easily be missed leading one to implement the protocol in a way that rejects TLVs with extra trailing data.

3.8. Imprecise language concerning IDs

In section 3.5 of [RFC6126] (specifying route table maintenance), there is some confusing use of 'id' in various contexts that may take a few readings to sort out.

4. In summary

The Babel routing protocol is easy to implement, as has been shown by two independent implementations created in a few days and weeks respectively. The specification is very readable with only a few omissions and unclear points as noted above. It is quite feasible

for someone with no prior experience with routing protocols to create a working implementation, and doing so is instructive for someone looking to understand both routing protocols in general and the Babel protocol in particular.

5. References

5.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <http://www.rfc-editor.org/info/rfc1982>.

[RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <http://www.rfc-editor.org/info/rfc6126>.

[RFC7557] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", RFC 7557, DOI 10.17487/RFC7557, May 2015, <http://www.rfc-editor.org/info/rfc7557>.

5.2. Informative References

[I-D.chroboczek-babel-applicability]
Chroboczek, J., "Applicability of the Babel routing protocol", draft-chroboczek-babel-applicability-01 (work in progress), February 2016.

Author's Address

Toke Hoiland-Joergensen
Karlstad University
Dept. of Computer Science
Karlstad 65188
Sweden

Email: toke.hoiland-jorgensen@kau.se