

Network Working Group
Internet-Draft
Updates: [6126](#) (if approved)
Intended status: Experimental
Expires: September 12, 2019

B. Jonglez
ENS Lyon
J. Chroboczek
IRIF, University of Paris-Diderot
March 11, 2019

**Delay-based Metric Extension for the Babel Routing Protocol
draft-jonglez-babel-rtt-extension-02**

Abstract

This document defines an extension to the Babel routing protocol that uses symmetric delay in metric computation and therefore makes it possible to prefer lower latency links to higher latency ones.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

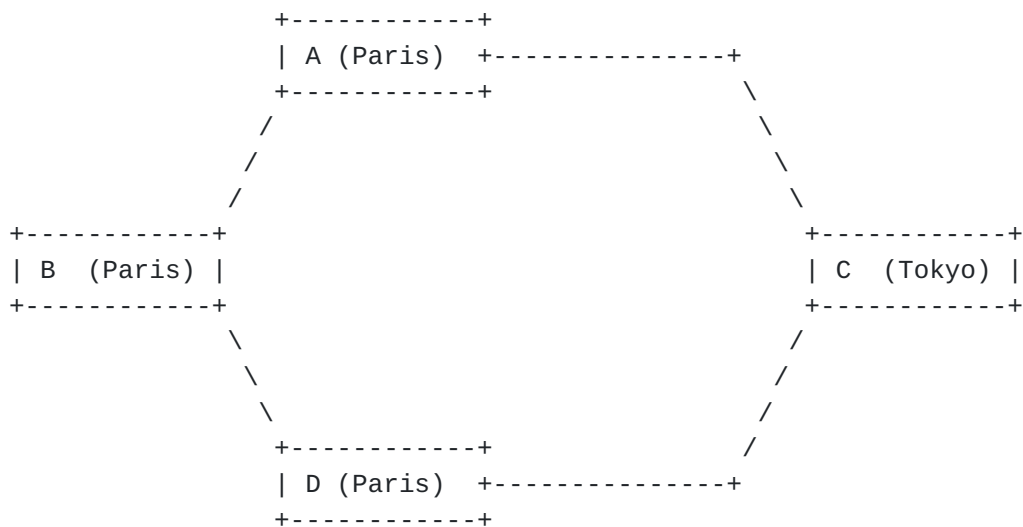
Table of Contents

- [1. Introduction](#) [2](#)
- [2. Protocol operation](#) [3](#)
 - [2.1. Delay estimation](#) [3](#)
 - [2.2. Metric computation](#) [5](#)
 - [2.3. Stability issues](#) [6](#)
 - [2.4. Backwards and forwards compatibility](#) [6](#)
- [3. Packet format](#) [6](#)
 - [3.1. Timestamp sub-TLV in Hello TLVs](#) [7](#)
 - [3.2. Timestamp sub-TLV in IHU TLVs](#) [7](#)
- [4. IANA Considerations](#) [8](#)
- [5. Security Considerations](#) [8](#)
- [6. References](#) [8](#)
 - [6.1. Normative References](#) [8](#)
 - [6.2. Informative References](#) [8](#)
- Authors' Addresses [9](#)

1. Introduction

The Babel routing protocol [[BABEL](#)] does not mandate a specific algorithm for computing metrics; existing implementations use a packet-loss based metric on wireless links and a simple hop-count metric on all other types of links. While this strategy works reasonably well in many networks, it fails to select reasonable routes in some topologies involving tunnels or VPNs.

Consider for example the following topology, with three routers A, B and D located in Paris and a fourth router located in Tokyo, connected through tunnels in a diamond topology.



When routing traffic from A to D, it is obviously preferable to use the local route through B, as this is likely to provide better service quality and lower monetary cost than the distant route through C. However, the existing implementations of Babel consider both routes as having the same metric, and will therefore route the traffic through C in roughly half the cases.

In this document, we specify an extension to the Babel routing protocol that enables precise measurement of the round-trip time (RTT) of a link, and allows its usage in metric computation. Since this causes a negative feedback loop, special care is needed to ensure that the resulting network is reasonably stable ([Section 2.3](#)).

We believe that this protocol may be useful in other situations than the one described above, such as when running Babel in a congested wireless mesh network or over a complex link layer that performs its own routing; the high granularity of the timestamps used (1ms) should make it easier to experiment with RTT-based metrics on this kind of link layers.

[2. Protocol operation](#)

The protocol estimates the RTT to each neighbour ([Section 2.1](#)) which it then uses for metric computation ([Section 2.2](#)).

[2.1. Delay estimation](#)

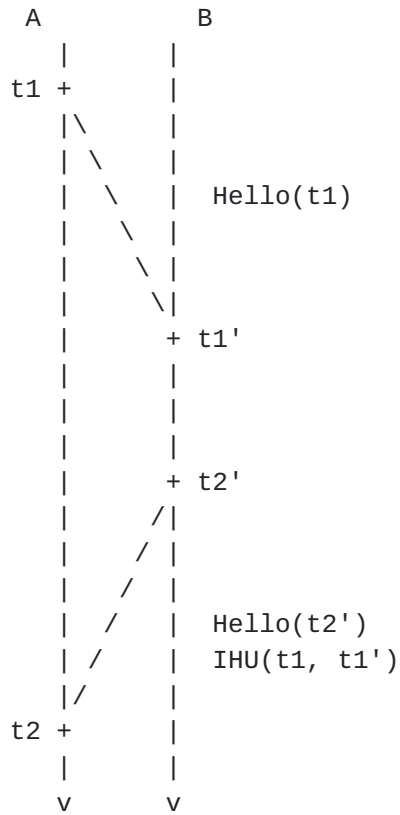
The RTT to a neighbour is estimated using an algorithm due to Mills [[MILLS](#)], originally developed for the HELLO routing protocol and later used in NTP [[NTP](#)].

A Babel speaker periodically sends a multicast Hello message over all of its interfaces (Section 3.4.1 of [[BABEL](#)]). This Hello is usually accompanied with a set of IHU messages, one per neighbour (Section 3.4.2 of [[BABEL](#)]).

In order to enable the computation of RTTs, a node A SHOULD include in every Hello that it sends a timestamp t_1 (according to A's clock). When a node B receives A's Hello, it records in its neighbour table the timestamp t_1 as well as the time t_1' according to its own (B's) clock at which it received the packet.

When B later sends an IHU to A, it SHOULD attach to the IHU the timestamps t_1 and t_1' which it has stored in its neighbour table. Additionally, it SHOULD ensure that the packet within which the IHU is sent contains a Hello TLV with an associated timestamp t_2' (according to B's clock). Symmetrically, A will record in its neighbour table the timestamp t_2' as well as the time t_2 (according

to A's clock) at which it has received the Hello. This is illustrated in the following sequence diagram:



A then estimates the RTT between A and B as $(t2 - t1) - (t2' - t1')$.

This algorithm has a number of desirable properties. First, since there is no requirement that $t1'$ and $t2'$ be equal, the protocol remains asynchronous -- the only change to Babel's message scheduling is the requirement that a packet containing an IHU also contains a Hello. Second, since only differences of timestamps according to a single clock are computed, it does not require synchronised clocks. Third, it requires very little additional state -- a node only needs to store the two timestamps associated with the last hello received from each neighbour. Finally, since it only requires piggybacking one or two timestamps on each Hello and IHU packet, it makes efficient use of network resources.

In principle, this algorithm is inaccurate in the presence of clock drift (i.e. when A's and B's clocks are running at different frequencies). However, $t2' - t1'$ is usually on the order of seconds, and significant clock drift is unlikely to happen at that time scale.

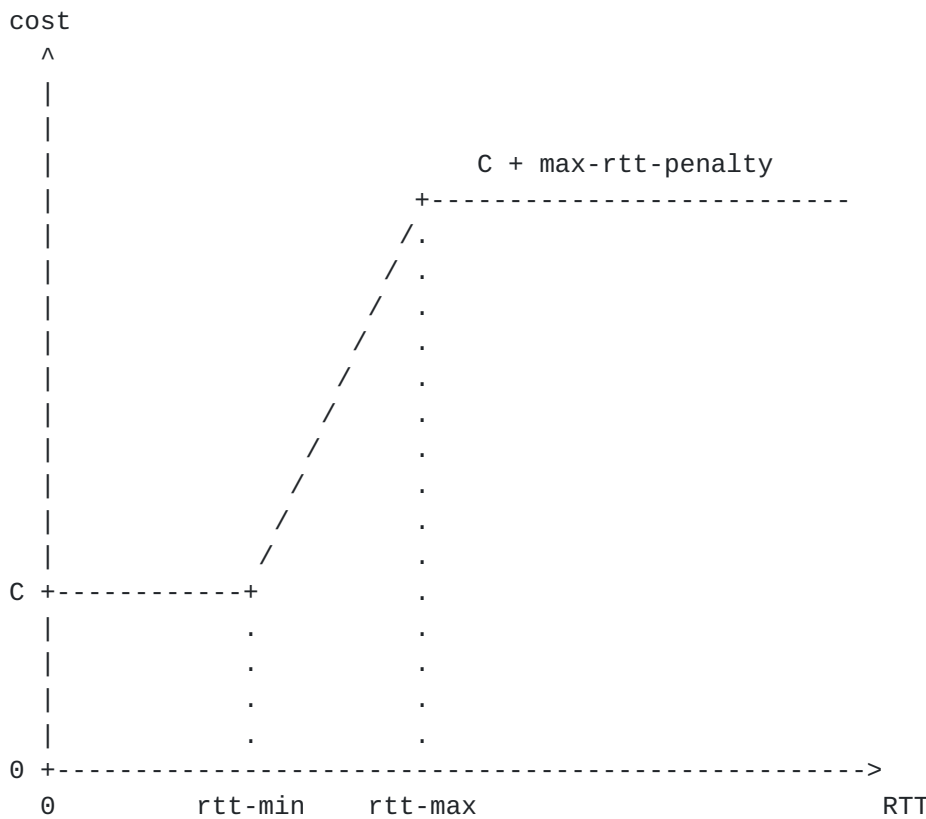
2.2. Metric computation

The algorithm described in the previous section allows computing an RTT to every neighbour. How to map this value to a link cost is a local implementation matter.

Obviously, the mapping should be monotonic (larger RTTs imply larger costs). In addition, in order to enhance stability ([Section 2.3](#)), the mapping should be bounded -- above a certain RTT, all links are equally bad.

2.2.1. Example metric computation

The current implementation of Babel uses the following function for mapping RTTs to link costs, parameterised by three parameters rtt-min, rtt-max and max-rtt-penalty:



For RTTs below rtt-min, the link cost is just the nominal cost of a single hop, C. Between rtt-min and rtt-max, the cost increases linearly; above rtt-max, the constant value max-rtt-penalty is added to the nominal cost.

2.3. Stability issues

Using delay as an input to the routing metric in congested networks gives rise to a negative feedback loop: low RTT encourages traffic, which in turn causes the RTT to increase. In a congested network, such a feedback loop can cause persistent oscillations.

The current implementation of Babel uses three techniques that collaborate to limit the frequency of oscillations:

- o the measured RTT is smoothed, which limits Babel's response to short-term RTT variations;
- o the mapping function is bounded, which avoids switching between congested routes;
- o a hysteresis algorithm is applied to the metric before route selection, which limits the worst-case frequency of route oscillations.

These techniques are discussed in more detail in [[DELAY-BASED](#)].

2.4. Backwards and forwards compatibility

This protocol extension stores the data that it requires within sub-TLVs of Babel's Hello and IHU TLVs. As discussed in Section 4 of [[BABEL-EXT](#)], implementations that do not understand this extension will silently ignore the sub-TLVs while parsing the rest of the TLVs that they contain. In effect, this extension supports building hybrid networks consisting of extended and unextended routers, and while such networks might suffer from sub-optimal routing, they will not suffer from blackholes or routing loops.

If a sub-TLV defined in this extension is longer than expected, the additional data is silently ignored. This provision is made in order to allow a future version of this document to extend the packet format with additional data.

3. Packet format

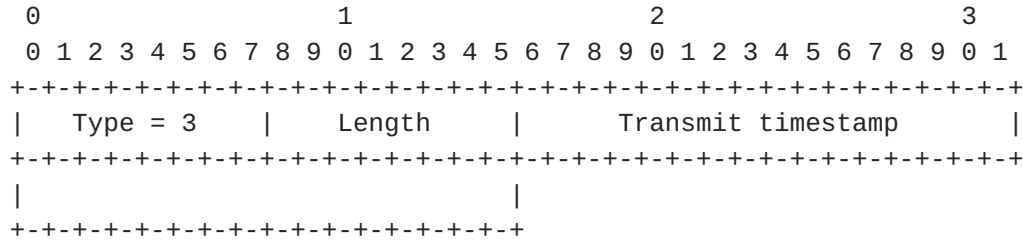
This extension defines the Timestamp sub-TLV [[BABEL-EXT](#)], whose Type field has value 3. This sub-TLV can be contained within a Hello sub-TLV, in which case it carries a single timestamp, or within an IHU sub-TLV, in which case it carries two timestamps.

Timestamps are encoded as 32-bit unsigned integers, expressed in units of one microsecond, counting from an arbitrary origin.

Timestamps wrap around every 4295 seconds, or slightly more than one hour.

3.1. Timestamp sub-TLV in Hello TLVs

When contained within a Hello TLV, the Timestamp sub-TLV has the following format:



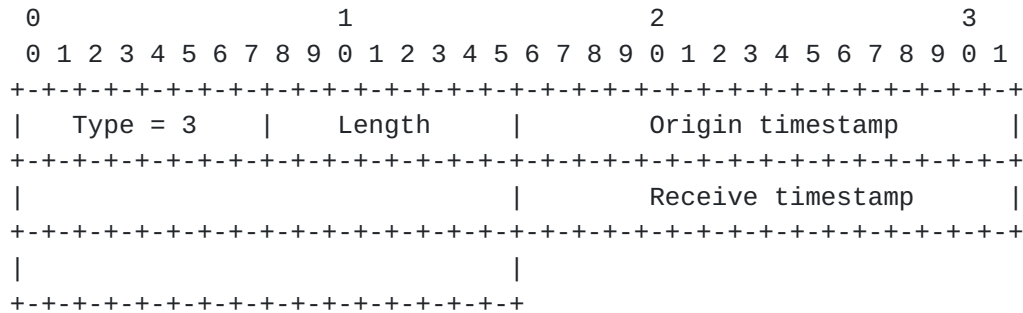
Fields :

- Type Set to 3 to indicate a Timestamp sub-TLV.
- Length The length of the body, exclusive of the Type and Length fields.
- Transmit timestamp The time at which the packet containing this sub-TLV was sent, according to the sender's clock.

If the Length field is larger than the expected 4 octets, the sub-TLV MUST be processed normally and any extra data contained in this sub-TLV MUST be silently ignored.

3.2. Timestamp sub-TLV in IHU TLVs

When contained in an IHU TLV destined for node A, the Timestamp sub-TLV has the following format:



Fields :

- Type Set to 3 to indicate a Timestamp sub-TLV.

Length The length of the body, exclusive of the Type and Length fields.

Origin timestamp A copy of the transmit timestamp of the last Timestamp sub-TLV contained in a Hello TLV received from node A.

Receive timestamp The time at which the last Hello with a Timestamp sub-TLV was received from node A according to the sender's clock.

If the Length field is larger than the expected 8 octets, the sub-TLV MUST be processed normally and any extra data contained in this sub-TLV MUST be silently ignored.

4. IANA Considerations

IANA is instructed to add the following entry to the "Babel Sub-TLV Types" registry:

Type	Name	Reference
3	Timestamp	(this document)

5. Security Considerations

This extension merely adds additional timestamping data to two of the TLVs sent by a Babel router, and does not significantly change the security properties of the Babel protocol.

6. References

6.1. Normative References

[BABEL] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, February 2011.

[BABEL-EXT] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", RFC 7557, May 2015.

6.2. Informative References

[DELAY-BASED] Jonglez, B. and J. Chroboczek, "A delay-based routing metric", March 2014.

Available online from <http://arxiv.org/abs/1403.3488>

- [MILLS] Mills, D., "DCN Local-Network Protocols", [RFC 891](#),
December 1983.
- [NTP] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network
Time Protocol Version 4: Protocol and Algorithms
Specification", [RFC 5905](#), June 2010.

Authors' Addresses

Baptiste Jonglez
ENS Lyon
France

Email: baptiste.jonglez@ens-lyon.org

Juliusz Chroboczek
IRIF, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13
France

Email: jch@irif.fr

