Network Working Group B. Jonglez
Internet-Draft ENS Lyon
Updates: 6126 (if approved) J. Chroboczek
Intended status: Experimental PPS, University of Paris-Diderot
Expires: January 3, 2015 July 2, 2014

Delay-based Metric Extension for the Babel Routing Protocol
draft-jonglez-babel-rtt-extension-00

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and background

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.

```
                      +------------+
                      | A (Paris)  +----------------+
                      +------------+                \
                     /                               \
                    /                                 \
                   /                                   \
     +------------+                                     +------------+
     | B  (Paris) |                                     | C  (Tokyo) |
     +------------+                                     +------------+
                   \                                   /
                    \                                 /
                     \                               /
                      +------------+                /
                      | D (Paris)  +----------------+
                      +------------+
```

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 memo, 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. This Hello is usually accompanied with a set of IHU messages, one per neighbour.

In order to enable the computation of RTTs, a node A includes in every Hello that it sends a timestamp t1 according to A's clock. Additionally, a node B includes in the IHU it sends to A the timestamp t1 included in the last Hello received from A, and the timestamp t1' according to B's clock at which it received that Hello. Upon receiving B's combined Hello and IHU, node A records the timestamp t2 at which it received the combined packet, according to A's clock. This is described in the following sequence diagram:

```
    A          B
      |      |
   t1 +      |
      |\     |
      | \    |
      |  \   |  Hello(t1)
      |   \  |
      |    \ |
      |     \|
      |      + t1'
      |      |
      |      |
      |      |
      |      + t2'
      |     /|
      |    / |
      |   /  |
      |  /   |  Hello(t2')
      | /    |  IHU(t1, t1')
      |/     |
   t2 +      |
      |      |
      v      v
```

Node A then computes the RTT 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 that is required is to ensure that IHUs are always sent together with Hellos. Second, since only differences of timestamps according to a single clock are computed, it does not require synchronised clocks. Third, it is mostly stateless -- a node only needs to store the two timestamps associated with the last hello received from each neighbour. Finally, since it only requires piggybacking a couple of timestamps on each Hello and IHU packet, it makes efficient use of network resources.

In principle, this protocol is incorrect 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 drift is unlikely to happen at this time scale.

2.2. Metric computation

The algorithm described in the previous section allows computing an RTT to all neighbours. How to map this value to a link cost is left to the implementation.

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. Sample mapping

The sample 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:

```
     cost
       ^
       |
       |
       |                           C + max-rtt-penalty
       |                       +---------------------------
       |                      /.
       |                     / .
       |                    /  .
       |                   /   .
       |                  /    .
       |                 /     .
       |                /      .
       |               /       .
       |              /        .
       |             /         .
     C +------------+          .
       |            .          .
       |            .          .
       |            .          .
       |            .          .
     0 +----------------------------------------------------->
       0         rtt-min    rtt-max                          RTT
```

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; abover 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 sample 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].

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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 3    |     Length    |      Transmit timestamp       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 3    |    Length     |         Origin timestamp      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |         Receive timestamp     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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. References

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

[BABEL-EXT]
Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", Internet Draft draft-chroboczek-babel-extension-mechanism-01, June 2014.

[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
PPS, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13
France

Email: jch@pps.univ-paris-diderot.fr