Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2019

L. Qiang, Ed.
B. Liu
T. Eckert, Ed.
Huawei
L. Geng
China Mobile
March 8, 2019

Large-Scale Deterministic Network
draft-qiang-detnet-large-scale-detnet-03

Abstract

This document presents the overall framework and key method for Large-scale Deterministic Network (LDN). LDN can provide bounded latency and delay variation (jitter) without requiring precise time synchronization among nodes, or per-flow state in transit nodes.

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 9, 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

This document explores the DetNet forwarding over large-scale network. In contrast to TSN that deployed in LAN, DetNet is expected to be deployed in larger scale network that has the following features:

- a large number of network devices

- the distance between two network devices is long

- a lot of deterministic flows on the network

These above features will bring the following challenges to DetNet forwarding:

- difficult to achieve precise time synchronization among all nodes

- long link propagation delay may introduce bigger jitter

o per-flow state is unscalable

Motivated by these challenges, this document presents a Large-scale Deterministic Network (LDN) mechanism. As [draft-ietf-detnet-problem-statement] indicates, deterministic forwarding can only apply on flows with well-defined traffic characteristics. The traffic characteristics of DetNet flow has been discussed in [draft-ietf-detnet-architecture], that could be achieved through shaping at Ingress node or up-front commitment by application. LDN assumes that DetNet flows follow some specific traffic patterns accordingly.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology & Abbreviations

This document uses the terminology defined in [draft-ietf-detnet-architecture].

TSN: Time Sensitive Network

PQ: Priority Queuing

CQF: Cyclic Queuing and Forwarding

LDN: Large-scale Deterministic Network

DSCP: Differentiated Services Code Point

EXP: Experimental

TC: Traffic Class

T: the length of a cycle

H: the number of hops

2. Overview

2.1. Summary

In LDN, nodes (network devices) have synchronized frequency, and each node forwards packets in a slotted fashion based on a cycle identifiers carried in packets. Ingres nodes or senders have a function called gate to shape/condition traffic flows. Except for this gate function, the LDN has no awareness of individual flows.

2.2. Background

This section motivates the design choices taken by the proposed solution and gives the necessary background for deterministic delay based forwarding plane designs.

2.2.1. Deterministic End-to-End Latency

Bounded delay is delay that has a deterministic upper and lower bound.

The delay for packets that need to be forwarded with deterministic delay needs to be deterministic on every hop. If any hop in the network introduces non-deterministic delay, then the network itself can not deliver a deterministic delay service anymore.

2.2.2. Hop-by-Hop Delay

Consider a simple example shown in Figure 1, where Node X has 10 receiving interfaces and one outgoing interface I all of the same speed. There are 10 deterministic traffic flows, each consuming 5% of a links bandwidth, one from each receiving interface to the outgoing interface.

Node X sends 'only' 50% deterministic traffic to interface I, so there is no ongoing congestion, but there is added delay. If the arrival time of packets for these 10 flows into X is uncontrolled, then the worst case is for them to all arrive at the same time. One packet has to wait in X until the other 9 packets are sent out on I, resulting in a worst case deterministic delay of 9 packets serialization time. On the next hop node Y downstream from X, this problem can become worse. Assume Y has 10 upstream nodes like X, the worst case simultaneous burst of packets is now 100 packets, or a 99 packet serialization delay as the worst case upper bounded delay incurred on this hop.

To avoid the problem of high upper bound end-to-end delay, traffic needs to be conditioned/interleaved on every hop. This allows to create solutions where the per-hop-delay is bounded purely by the

physics of the forwarding plane across the node, but not the accumulated characteristics of prior hop traffic profiles.

```
              +--+  +--+      ---                      ---
              |A1|  |A0|     -   -                    -   -
              +--+  +--+    -     -                  -     -
          ---------------->-       -                -       -
            +--+    +--+   -       -                -       -
            |B1|    |B0|   -       -                -       -
            +--+    +--+   -       -Interface I     -       -
          ---------------->-Node X ---------------> -Node Y ----->
                +--++--+   -       -                -       -
                |C1||C0|   -                        -
                +--++--+   -       -                -       -
          ---------------->-       -                -       -
                    .      -       -                -       -
                    .       -     -                  -     -
                    .        -   -                    -   -
                              ---                      ---
```

Figure 1: Micro-burst and micro-burst iteration

2.2.3. Cyclic Forwarding

The common approach to solve that problem is that of a cyclic hop-by-hop forwarding mechanism. Assume packets forwarded from N1 via N2 to N3 as shown in Figure 2. When N1 sends a packet P to interface I1 with a Cycle X, it must be guaranteed by the forwarding mechanism that N2 will forward P via I2 to N3 in a cycle Y.

The cycle of a packet can either be deduced by a receiving node from the exact time it was received as is done in SDN/TDMA systems, and/or it can be indicated in the packet. This document solution relies on such markings because they allow to reduce the need for synchronous hop-by-hop transmission timings of packets.

In a packet marking based slotted forwarding model, node N1 needs to send packets for cycle X before the latest possible time that will allow for N2 to further forward it in cycle Y to N3. Because of the marking, N1 could even transmit packets for cycle X before all packets for the previous cycle (X-1) have been sent, reducing the synchronization requirements between across nodes.

```
 P sent in          P sent in         P sent in
cycle(N1,I1,X)    cycle(N2,I2,Y)     cycle(N3,I3,Z)
+---------+       +---------+        +---------+
| Node N1|------->| Node N2|-------->| Node N3|------>
+---------+I1     +---------+I2      +---------+I3
```

Figure 2: Cyclic Forwarding

2.2.4. Co-Existence with Non-Deterministic Traffic

Traffic with deterministic delay requirements can co-exist with traffic only requiring non-deterministic delay by using packet scheduling where the delay incurred by non-deterministic packets is deterministic for the deterministic traffic (and low). If LDN is deployed together with such non-deterministic delay traffic than such a scheme must be supported by the forwarding plane. A simple approach for the delay incurred on the sending interface of a deterministic node due to non-deterministic traffic is to serve deterministic traffic via a strict, highest-priority queue and include the worst case delay of a currently serialized non-deterministic packet into the deterministic delay budget of the node. Similar considerations apply to the internal processing delays in a node.

2.3. System Components

The Figure 3 shows an overview of the components considered in this document system and how they interact.

A network topology of nodes, Ingress, Core and Egress support a method for cyclic forwarding to enable LDN. This forwarding requires no per-flow state on the nodes, and tolerates loss time synchronization.

Ingress edge nodes may support the (G)ate function to shape traffic from sources into the desired traffic characteristics, unless the source itself has such function. Per-flow state is required on the ingress edge node. LDN should work with some resource reservation methods, that will be not discussed in this document.

```
        /--\.     +--+        +--+      +--+        +--+.     /--\
       | (G)+-----+GS+--------+ S+------+ S+--------+ S+-----+    |
        \--/      +--+        +--+      +--+        +--+      \--/

       Sender    Ingress      Core      Core       Egress   Receiver
                Edge Node     Node      Node     Edge Node
```

Figure 3: Overview of LDN

3. LDN Forwarding Mechanism

DetNet aims at providing deterministic service over large scale network. In such large scale network, it is difficulty to get precise time synchronization among numerous devices. To reduce requirements, the forwarding mechanism described in this document assumes only frequency synchronization but not time synchronization across nodes: nodes maintain the same clock frequency 1/T, but do not require the same time as shown in Figure 4.

```
         <-----T----->                       <-----T----->
         |           |           |           |           |           |
 Node A  +-----------+-----------+   Node A  +-----------+-----------+
                     T0                                  T0

         |           |           |              |           |           |
 Node B  +-----------+-----------+   Node B     +-----------+-----------+
                     T0                                     T0

     (i) time synchronization           (ii) frequency synchronization

     T: length of a cycle
     T0: timestamp
```

Figure 4: Time Synchronization & Frequency Synchronization

IEEE 802.1 CQF is an efficient forwarding mechanism in TSN that guarantees bounded end-to-end latency. CQF is designed for limited scale networks. Time synchronization is required, and the link propagation delay is required to be smaller than a cycle length T. Considering the large scale network deployment, the proposed LDN Forwarding mechanism permits frequency synchronization and link propagation delay may exceed T. Besides these two points, CQF and the asynchronous forwarding of LDN are very similar.

Figure 5 compares CQF and LDN through an example. Suppose Node A is the upstream node of Node B. In CQF, packets sent from Node A at cycle x, will be received by Node B at the same cycle, then further

be sent to downstream node by Node B at cycle x+1. Due to long link propagation delay and frequency synchronization, Node B will receive packets from Node A at different cycle denoted by y in the LDN, and Node B swaps the cycles carried in packets with y+1, then sends out those packets at cycle y+1. The cycle mapping relationship (e.g., x->y+1) exists between any pair of neighbor nodes. With this kind of cycle mapping, the receiving node can easily figure out when the received packets should be sent out, the only requirement is to carry the cycle identifier of sending node in the packets.

```
       |  cycle x  | cycle x+1 |          |  cycle x  | cycle x+1 |
Node A +-----------+-----------+   Node A +-----------+-----------+
          \                                           \
           \packet                                      \packet
            \receiving                                   \receiving
             \                                            \
       |      V    | cycle x+1 |                 |         V |  cycle y+1|
Node B +-----------+-----------+      Node B     +-----------+-----------+
          cycle x      \packets                     cycle y      \packets
                        \sending                                  \sending
                         \                                         \
                          \                                         \
                           V                                         V

              (i) CQF                                (ii) LDN
```

Figure 5: CQF & LDN

3.1. Cyclic Queues

In CQF each port needs to maintain 2 (or 3) queues: 1 receiving queue is used to buffer newly received packets, 1 sending queue is used to store the packets that are going to be sent out, one more queue may be needed to avoid output starvation [scheduled-queues]. In LDN, at least 3 cyclic queues (2 receiving queues and 1 sending queue) are maintained for each port on a node. A cyclic queue corresponds to a cycle.

As Figure 6 illustrated, the downstream Node B may receive packets sent at two different cycles from Node A due to the absence of time synchronization. Following the cycle mapping (i.e., x --> y+1), packets that carry cycle identifier x should be sent out by Node B at cycle y+1, and packets that carry cycle identifier x+1 should be sent out by Node B at cycle y+2. Therefore, two receiving queues are needed to store the received packets, one is for the packets that carry cycle identifier x, another one is the packets that carry cycle identifier x+1. Plus one sending queue, each port needs at least 3

cyclic queues in LDN. In order to absorb more link delay variation (such as on radio interface), more queues may be necessary.

```
                      |  cycle x  | cycle x+1 |
              Node A  +-----------+-----------+
                               \      \
                                \      \packet
                                 \      \receiving
                                | V      V  |           |
                     Node B     +-----------+-----------+
                                   cycle y    cycle y+1
```

Figure 6: An example illustrates for 2 receiving queue in LDN

3.2. Cycle Mapping

When this packet is received by Node B, some methods are possible how the forwarding plane could operate. In one method, Node B has a mapping determined by the control plane. Packets from (the link from) Node A indicating cycle x are mapping into cycle y+1. This mapping is necessary, because all the packets from one cycle of the sending node need to get into one cycle of the receiving node. This is called "configured cycle mapping".

Instead of configuring an explicit cycle mapping such as cycle x -> cycle y+1, the receiving Node B could also have the intelligence in the forwarding plane to recognize the first packet from (the link from) Node A that has a new cycle x number, and map this cycle x to a cycle after the current cycle y. We call this option "self synchronized cycle mapping".

3.2.1. Cycle Identifier Carrying

In self synchronized cycle mapping, cycle identifier needs to be carried in the LDN packets, so that an appropriate queue can be selected accordingly. That means 2 bits are needed in the three queues model of LDF, in order to identify different cycles between a pair of neighboring nodes. There are several ways to carry this 2 bits cycle identifier. This document does not yet aim to propose one, but gives an (incomplete) list of ideas:

o DSCP of IPv4 Header

o Traffic Class of IPv6 Header

o TC of MPLS Header (used to be EXP)

- EtherType of Ethernet Header
- IPv6 Extension Header
- UDP Option
- SID of SRv6
- Reserved of SRH
- TLV of SRv6
- TC of SR-MPLS Header (used to be EXP)
- Three labels/adjacency SIDs for SR-MPLS

4. Performance Analysis

4.1. Queueing Delay

Figure 7 describes one-hop packet forwarding delay, that mainly consisted of A->B link propagation delay and queuing delay in Node B.

```
                     |cycle x |
              Node A +--------\+
                               \
                                \
                                 \
                                 |\ cycle y|cycle y+1|
              Node B             +V--------+--------\+
                                 :                   \
                                 :   Queueing Delay  :\
                                 :...=2*T ............ V
```

Figure 7: Single-Hop Queueing Delay

As Figure 7 shows, cycle x of Node A will be mapped into cycle y+1 of Node B as long as the last packet sent from A->B is received within the cycle y. If the last packet is re-sent out by B at the end of cycle y+1, then the largest single-hop queueing delay is 2*T. Therefore the end-to-end queueing delay's upper bound is 2*T*H, where H is the number of hops.

If A did not forward the LDN packet from a prior LDN forwarder but is the actual traffic source, then the packet may have been delayed by a gate function before it was sent to B. The delay of this function is outside of scope for the LDN delay considerations. If B is not

forwarding the LDN packet but the final receiver, then the packet may not need to be queued and released in the same fashion to the receiver as it would be queued/released to a downstream LDN node, so if a path has one source followed by N LDN forwarders followed by one receivers, this should be considered to be a path with N-1 LDN hops for the purpose of latency and jitter calculations.

4.2. Jitter

Considering the simplest scenario one hop forwarding at first, suppose Node A is the upstream node of Node B, the packet sent from Node A at cycle x will be received by Node B at cycle y as Figure 8 shows.

- The best situation is Node A sends packet at the end of cycle x, and Node B receives packet at the beginning of cycle y, then the delay is denoted by w;

- The worst situation is Node A sends packet at the beginning of cycle x, and Node B receives packet at the end of cycle y, then the delay= w + length of cycle x + length of cycle y= w+2*T;

- Hence the jitter's upper bound of this simplest scenario= worst case-best case=2*T.

```
       |cycle x |                        |cycle x |
Node A +-------\+                 Node A +\-------+
               :\                           \     :
               : \                           -------------\
               :  \                               :        \
               :w |\        |                     :w|       \ |
Node B         :  +V--------+     Node B          : +--------V+
                    cycle y                             cycle y

    (a) best situation               (b) worst situation
```

Figure 8: Jitter Analysis for One Hop Forwarding

Next considering two hops forwarding as Figure 9 shows.

- The best situation is Node A sends packet at the end of cycle x, and Node C receives packet at the beginning of cycle z, then the delay is denoted by w';

- The worst situation is Node A sends packet at the beginning of cycle x, and Node C receives packet at the end of cycle z, then the delay= w' + length of cycle x + length of cycle z= w'+2*T;

- Hence the jitter's upper bound = worst case-best case=2*T.

```
                 |cycle x |
          Node A +-------\+
                          \
                          :\| cycle y |
          Node B          : \---------+
                          :  \
                          :   \--------\
                          :             \         |
          Node C          ......w'......+V--------+
                                          cycle z

                     (a) best situation


                  |cycle x |
           Node A +\-------+
                    \      :
                     \     : | cycle y |
           Node B     \    : +---------+
                       \   :
                        ---:-------------------\
                           :             |       \ |
           Node C          :......w'.....+--------V+
                                           cycle z

                     (b) worst situation
```

Figure 9: Jitter Analysis for Two Hops Forwarding

And so on. For multi-hop forwarding, the end-to-end delay will increase as the number of hops increases, while the delay variation (jitter) still does not exceed 2*T.

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

Security issues have been carefully considered in [draft-ietf-detnet-security]. More discussion is TBD.

7. Acknowledgements

TBD.

8. Normative References

[draft-ietf-detnet-architecture]
"DetNet Architecture", <https://datatracker.ietf.org/doc/draft-ietf-detnet-architecture/>.

[draft-ietf-detnet-dp-sol]
"DetNet Data Plane Encapsulation", <https://datatracker.ietf.org/doc/draft-ietf-detnet-dp-sol/>.

[draft-ietf-detnet-problem-statement]
"DetNet Problem Statement", <https://datatracker.ietf.org/doc/draft-ietf-detnet-problem-statement/>.

[draft-ietf-detnet-security]
"DetNet Security Considerations", <https://datatracker.ietf.org/doc/draft-ietf-detnet-security/>.

[draft-ietf-detnet-use-cases]
"DetNet Use Cases", <https://datatracker.ietf.org/doc/draft-ietf-detnet-use-cases/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[scheduled-queues]
"Scheduled queues, UBS, CQF, and Input Gates", <http://www.ieee802.org/1/files/public/docs2015/new-nfinn-input-gates-0115-v04.pdf>.

Authors' Addresses

Li Qiang (editor)
Huawei
Beijing
China

Email: qiangli3@huawei.com

Bingyang Liu
Huawei
Beijing
China

Email: liubingyang@huawei.com

Toerless Eckert (editor)
Huawei USA - Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Liang Geng
China Mobile
Beijing
China

Email: gengliang@chinamobile.com