DETNET
Internet-Draft
Intended status: Standards Track
Expires: 18 July 2026

T. Eckert, Ed.
Futurewei Technologies USA
A. Clemm
Sympotech
S. Bryant
Independent
S. Hommes
ZF Friedrichshafen AG

14 January 2026

Deterministic Networking (DetNet) Data Plane - guaranteed Latency Based Forwarding (gLBF) for bounded latency with low jitter and asynchronous forwarding in Deterministic Networks

draft-ietf-detnet-glbf-00

Abstract

This memo proposes a mechanism called "guaranteed Latency Based Forwarding" (gLBF) as part of DetNet for hop-by-hop packet forwarding with per-hop deterministically bounded latency and minimal jitter.

gLBF is intended to be useful across a wide range of networks and applications with need for high-precision deterministic networking services, including in-car networks or networks used for industrial automation across on factory floors, all the way to ++100Gbps country-wide networks.

Contrary to other mechanisms, gLBF does not require network wide clock synchronization, nor does it need to maintain per-flow state at network nodes, avoiding drawbacks of other known methods while leveraging their advantages.

Specifically, gLBF uses the queuing model and calculus of Urgency Based Scheduling (UBS, [UBS]), which is used by TSN Asynchronous Traffic Shaping [TSN-ATS]. gLBF is intended to be a plug-in replacement for TSN-ATN or as a parallel mechanism beside TSN-ATS because it allows to keeping the same controller-plane design which is selecting paths for TSN-ATS, sizing TSN-ATS queues, calculating latencies and admitting flows to calculated paths for calculated latencies.

In addition to reducing the jitter compared to TSN-ATS by additional buffering (dampening) in the network, gLBF also eliminates the need for per-flow, per-hop state maintenance required by TSN-ATS. This avoids the need to signal per-flow state to every hop from the controller-plane and associated scaling problems. It also reduces implementation cost for high-speed networking hardware due to the

avoidance of additional high-speed speed read/write memory access to retrieve, process and update per-flow state variables for a large number of flows.

Status of This Memo

This Internet-Draft will expire on 18 July 2026.

Copyright Notice

Table of Contents

1. Overview (informative)

1.1. Terminology

CaaS Control as a Service. Cloud (compute) and network services to enable control (loops) with network connected devices, for example cars.

CQF Cyclic Queuing and Forwarding. A queuing mechanism defined by annex T of IEEE802.1Q.

DT Dead Time. A term from CQF indicating the time during each cycle in which no frames can be sent because the the receiving node could not receive it into the desired cycle buffer.

node Term used to indicate a system that with respect to gLBF does not act as a host, aka: sender/receiver. This memo avoids the term router to avoid implying that this is an IP/IPv6 router, as opposed to an LSR (label switch router). Likewise, a node can also be an 802.1 bridge implementing gLBF.

TCQF Tagged Cyclic Queuing and Forwarding. The mechanism specified in this memo.

1.2. Summary

This memo proposes a mechanism called "guaranteed Latency Based Forwarding" (gLBF) for hop-by-hop packet forwarding with per-hop deterministically bounded latency and minimal jitter.

gLBF is intended to be useful across a wide range of networks and applications with need for high-precision deterministic networking services, including in-car networks or networks used for industrial automation across on factory floors, all the way to ++100Gbps country-wide networks.

At its foundation, gLBF addresses the problems of burst accumulation and jitter accumulation across multiple hops.

Burst accumulation is the phenomenon in which bursts of packets from senders in admission-controlled network will increase across intermediate nodes. This can only be managed with exponential complexity in admission control processing and significantly worst-case increase in end-to-end latency and/or lowered maximum utilization. What is needed for dynamic, large-scale or easy to manage admission control solutions are forwarding mechanisms without this problem, so that admission control for bandwidth and jitter/buffer-requirements can be linear: decomposed into solely per-hop calculations independent of changes in prior-hop traffic characteristics. Without forwarding plane solutions to overcome burst accumulation, this is not possible

Existing solutions addressing burst-accumulation do this by maintaining inter-packet gaps on a per-flow basis, such as in TSN Asynchronous Traffic Shaping (TSN-ATS). gLBF instead ensures inter-packet gaps are always maintained without the need for per-flow state. The basic idea involves assigning a specific queuing delay budget for any given node and class of traffic. This budget is pre-known from admission control calculus. As the packet is transmitted, the actual queuing delay that was experienced by the packet at the node is subtracted from that budget and carried in a new header field of the packet. Upon receiving the packet, the subsequent node subjects the packet to a delay stage. Here the packet needs to wait for the time specified by that parameter before the node proceeds with regular processing of the packet. This way, any queuing delay variations are absorbed and deterministic delay without the possibility of burst accumulation can be achieved.

By addressing burst-accumulation, gLBF also overcomes the problem of jitter-accumulation. This is the second core problem of mechanisms such as TSN-ATS: Depending on the amount of competing admitted traffic on a hop at any point in time packets of a flow may experience zero to maximum delay across the hop. This is the per-hop jitter. This jitter is additive across multiple hope, resulting in the inability for applications requiring (near) synchronous packet delivery to solely rely on such mechanisms. It likewise limits the ability of high utilization of networks with large number of bounded latency flows.

The basic principle on which gLBF operates was already proposed by researchers in the early 1990th and called "Dampers". These dampers where not pursued or adopted due to the lack of network equipment capabilities back then. Only with recent and upcoming improvements in advanced forwarding planes will it be possible to build these technologies at scale and cost.

Contrary to other proposals, gLBF does not require network wide clock synchronization. Likewise, it does not need to maintain per-flow state at network nodes, as delay budget and the queuing delay variations that are to be absorbed are carried as part of the packets themselves, making them "self-contained". This eliminate the for the per-flow, per-hop state maintenance required by TSN-ATS, which involves scaling problems of signaling this per-flow state to every hop from the controller-plane as well as the high-speed networking hardware implementation cost of high-speed speed read/write memory access to retrieve, process and update these per-flow state variables for large number of flows.

Opposed to other damper proposals, gLBF also supports the queuing model and calculus of Urgency Based Scheduling (UBS, [UBS]), which is used by TSN-ATS. gLBF is intended to be a plug-in replacement for TSN-ATN or as a parallel mechanism beside TSN-ATS because it allows to keeping the same controller-plane design which is selecting paths for TSN-ATS, sizing TSN-ATS queues, calculating latencies and admitting flows to calculated paths for calculated latencies.

While gLBF as presented here is intended for use with IETF forwarding protocols and to provide DetNet QoS for bounded latency and lower bounded jitter, it would equally applicable to other forwarding planes, such as IEEE 802.1 Ethernet switching - assuming appropriate packet headers are defined to carry the hop-by-hop and end-to-end metadata required by the mechanism.

1.3. Application scenarios and use cases

gLBF addresses the same use cases that are targeted by deterministic networking and high-precision networking services in general. Common requirements of those services involve the need to provide service levels within very tightly defined service level bounds, in particular very specific latencies without the possibility of congestion-induced loss. The ability to provide services with corresponding service level guarantees enables many applications that would simply not be feasible without such guarantees. The following describes some use cases and application scenarios.

The development towards autonomous driving vehicles leads to new requirements and a high demand of computational resources that are often not available within a car. A solution is to reduce the in-car processing to a minimum, and to offload more computational expensive tasks to the cloud environment. Due to the safety implications and use cases, a delay in the transport of message from the cloud to the car and vice versa is often not acceptable. This includes application scenarios such as trajectory planning for driverless vehicles, but also emergency notifications to surrounding cars that require a fast delivery of messages to prevent a delayed action in case of an accident. While the usage of TSN for in-vehicle networks is already investigated and more mature, the usage of a real-time communication with guaranteed latencies for vehicles with to the cloud is still an open challenge.

Another use case that is important for the automotive industry is to further optimise the manufacturing process by taking into account more data sources. Since most of the SCADA systems and PLCs cannot connect to large data lakes and perform more advanced computational jobs, a real-time communication to the shop floor from a cloud instance does have several benefits. First of all does it integrate more data into the decision making process, since the control algorithms to automate manufacturing sites located in a cloud environment can take into account more variables than single plant control systems. Another aspect is also to reduce the resources on a particular factory floor or plant by transferring complex, recurring and more resource computational jobs to a cloud provider where a lack of computational power is not the limiting factor. However, in such cases it is important that associated control can still occur in real-time and according to very precise timing constraints. Such a development is already foreseen by trends such as Industry 4.0 and Control-as-a-Service (CaaS).

1.4. Background

The following background introduces and explains gLBF step by step. It uses IEEE 802.1 "Time Sensitive Networking" (TSN) mechanisms for reference, because at the time of this memo, TSN bounded latency mechanisms are the most commonly understood and deployed mechanisms to provide bounded latency and jitter services.

All mechanisms compared here are as well as those used by TSN based on the overall service design that traffic flows have a well-defined rate and burstyness, which are tracked by the controller-plane and called here the "envelope" of the flow.

The traffic model used for gLBF is taken from UBS gives the flow a rate r [bps/sec] and a burst size b [bits] and the traffic envelope condition is that the total number of bits w(t) over a period t [sec] of for the flow must be equal to or smaller than (r * t + b). In one typical case, a flow wants to send a packet of size b one every interval of p [sec]. This translates into a rate r = b / p for the flow because after the flow has sent the first packet of b bits, it will take p seconds until (r * t) has a size of b again: (r * t) = ((b / p) * p).

The controller-plane uses this per-flow information to calculate for each flow a path with sufficient free bandwidth and per-hop buffers so that the bounded end-to-end latency of packets can be guaranteed. It then allocates bandwidth and buffer resources to the flow so that further flows will not impact it.

Within this framework, bounded latency mechanisms can in general be divided into "on-time" and "in-time" mechanisms.

1.4.1. In-time mechanisms

"In-time" bounded latency mechanisms forward packets in an (almost) work conserving manner.

When there is no competing traffic in the network, packets of traffic flows that comply to their admitted envelope are forwarded without any mechanism introduced queuing latency. When the maximum amount of admitted traffic is present, then packets of admitted flows will experience the maximum guaranteed, so-called bounded latency. In result, in-time mechanism introduce the maximum amount of jitter because the amount of competing traffic can quickly change and then the latency of packets will change greatly.

IEEE TSN Asynchronous Traffic Shaping is the prime example of an in-time mechanism. IETF [RFC2212], "Integrated Services" is an older mechanism based on the same principles. In-time mechanisms have the benefit of not requiring clock-synchronization between nodes to support their queuing.

1.4.2. On-time mechanisms

On-time bounded latency mechanisms do deliver packets (near) synchronously with zero or a small maximum jitter - significantly smaller than that of in-time mechanisms.

EEE TSN Cyclic Queuing and Forwarding (CQF) is an example of an on-time mechanism as is the Tagged Cyclic Queuing and Forwarding (TCQF) mechanism proposed to DetNet.
Unlike the before mentioned in-time mechanisms, these mechanisms require clock synchronization between router.

1.4.3. Control loops vs. in-time and on-time

One set of applications that require or prefer on-time (low jitter) delivery of packets are control loops in vehicles or industrial environments and hence low-speed and short-range networks.

Emerging or future use-cases such as remote PLC or remote driving extend these requirements also into metropolitan scale networks. In these environments, on-time forwarding is also called synchronous forwarding for synchronous control loops.

Typically, in synchronous control loops, central units such as Programmable Logic Controllers (PLC) do control a set of sensors and actors, polling or periodically receiving status information from sensors and sending action instructions to actors. When packet forwarding is on-time (synchronous), this central unit does exactly know the time at which sensors sent a packet and the time at which packets are received by actors and they can react on it.

These solutions do not require sensors and actors to have accurate, synchronised clocks. Instead, the central unit can control the time at which sensor and actors perform their operations within the accuracy of the (zero/low) jitter of the network packet transmission.

When bounded latency forwarding is (only) in-time, edge nodes in the network and/or sensors and actors need to convert the packets arriving with high jitter into an on-time arrival model to continue supporting this application required model.

This conversion is typically called a playout-buffer mechanism and involves the need to synchronizing time between senders and receivers, and hence most commonly the need for the network to support clock synchronization to support these edge devices and/or sender receivers.

In result, existing bounded latency mechanisms to support synchronous, on-time delivery of packets do require clock synchronization across the network. In the existing mechanisms like CQF for the forwarding mechanism itself, in the on-time mechanisms to synchronize edge-devices and hosts.

1.4.4. Challenges with network wide clock synchronization

While clock synchronization is a well understood technology, it is also a significant operational if not device equipment cost factor [TBD: Add details if desired].

Therefore, clock synchronization with e.g.: IEEE 1588 PTP is only deployed where no simpler solutions exist that provide the same benefits but without clock synchronization. Today this for example means that in mobile networks, only the so-called fronthaul deploys clock synchronization, but not the backhaul.

In result, it could be a challenge to introduce new applications such as the above mentioned remote PLC, driving applications if they wanted to rely on a bounded latency network service. But even for existing markets such as in-car or industrial networks, removal or reduction of the need for clock synchronization could be a significant evolution to reduce cost and increase simplicity of solutions.

2. gLBF introduction (informative)

2.1. Dampers

The principle of the mechanism presented here is the so-called "Damper" mechanism, first mentioned in the early 1990th, but never standardized back then, primarily because the required forwarding was considered to be too advanced to be supportable in equipment at the time. These limitations are not starting to be resolved, and hence it seems like a good time to re-introduce this mechanism.

The principle of damper based forwarding is easily explained: When a packet is sent by a node A, this node will have measured the latency L, how long the packet was processed by the node. The main factor of L is the queuing latency of the packet in A because of competing traffic sent to the same outgoing interface. Based on the admission control and queuing algorithms used in the node, there can be a known upper bound M(ax) for this processing latency though, and when the packet then arrives at the next-hop receiving node B, this node will simply further delay the packet by (B-L), and in result the packet will have synchronously been forwarded from A to B with a constant latency of M(ax).

```
+------------------------+      +------------------------+
| Node A                 |      | Node B                 |
|   +-+         +-+      |      |   +-+         +-+      |
|-x-|F|---------|Q|------|------|-x-|F|---------|Q|------|
|   +-+         +-+      | Link |   +-+         +-+      |
+------------------- ----+      +------------------------+
  |<--------- Hop A/B latency --->|
```

Figure 1: Forwarding without Damper

Figure 1 shows a single hop from node A to node B without Dampers.

A receives a packet. The F)orwarding module determines some outgoing interface towards node B where it is enqueued into Q because competing packets may already be in line to be sent from Q to B. This is because packets may be arriving simultaneously from multiple different input interfaces on A (not shown in picture), and/or the speed of the receiving interface on A is higher than the speed of the link to B.

Forwarding F can be L2 switching and/or L3 routing, the choice does not impact the considerations described here.

When the packet is finally sent from Q over link to B, B repeats the same steps towards the next (not shown) node.

(x) is the reception time of the packet in A and B, and the per-hop latency of the packet is xB - xA, predominantly determined by the time the packet has to wait in Q plus any other relevant processing latency, fixed or variable from F plus the propagation latency of the packet across link, which is predominantly determined by the serialization latency of the packet plus the propagation latency of the link, which is the speed of light in the material used, for example fiber, where the speed of light is 31 percent slower than across vacuum.

All the factors impacting the hop A/B latency other than Q in A are naturally bounded: A well defined maximum can be calculated or overestimated. The transmission of the packet from A to B is composed from the serialization latency of the packet which can be calculated from the packet length of the packet and per-packet-bit speed of the packet. The propagation latency through the link can be calculated from speed of light and material (speed of light is 31% slower through fiber than vacuum for example). And so on.

To have a guaranteed bounded latency through Q, an admission control system is required that tracks, accounts and limits the total amount of traffic through Q. Admission control mechanisms rely on knowing the maximum amount of bursts that each traffic flow can cause and adding up those bursts to determine the maximum amount of simultaneous packet data in Q that therefore impacts the maximum latency of each individual packet through Q.

With such an admission control system, one can therefore calculate a maximum hop A/B propagation latency MAX for a packet, but packets will naturally have variable hop A/B latency lower than MAX, based on differences in packet size and differences in competing traffic. In result, their relative arrival times xB in B will be different from their relative arrival times xA in A. This leads to so-called "burst-aggregation" and hence the problem that the admission control system can not easily calculate the maximum burst and latency through Q in B as it can do for Q in A.

If however packets could be made to be forwarded such that their Hop A/B latency would all be the same (synchronous, on-time), then the admission control system could apply the same calculus to B as it was able to apply to A. This is what damper mechanisms attempt to achieve.

```
+-------------------------+      +-------------------------+
| Node A                  |      | Node B                  |
|   +-+   +-+   +-+       |      |   +-+   +-+   +-+       |
|-x-|D|-y-|F|---|Q|-------|------|-x-|D|-y-|F|---|Q|-------|
|   +-+   +-+   +-+       | Link |   +-+   +-+   +-+       |
+-------------------------+      +-------------------------+
        |<- A/B in-time latency ->|
        |<--A/B on-time latency ------->|
```

Figure 2: Forwarding with Damper

Figure 2 shows the most simple to explain, but not implement, Damper mechanism to achieve exactly this. Node A measures the time at xA, sends this value in a packet header to B. B measures the time directly at reception of the packet in xB. It then delays the packet for a time (MAX-(xB-yA)). In result, the latency (yB-yA) will exactly be MAX, up to the accuracy of the damper.

The first challenge with simple approach is the need to synchronize the clocks between A and B, so that (xB-yA) can correctly be calculated.

```
+-------------------------+      +-------------------------+
| Node A                  |      | Node B                  |
|   +-+   +-+   +-+       |      |   +-+   +-+   +-+       |
|-x-|D|-y-|F|---|Q|----z-|------|-x-|D|-y-|F|---|Q|----z-|
|   +-+   +-+   +-+       | Link |   +-+   +-+   +-+       |
+-------------------------+      +-------------------------+
        |<- A/B in-time latency ->|
        |<--A/B on-time latency ------->|
```

Figure 3: Forwarding with Damper and measuring

Figure 3 shows how this can be resolved by also measuring the time at z. A calculates d = (MAX1-(zA-yA)) and sends d in a header of the packet to B. B then delays the packet in D by d relative to the also locally measured time xB. Because the calculation of d in A and the delay by d in D does not depend on clock synchronization between A and B anymore, this approach eliminate the need for clock synchronization.

But in result of this change, the measurement of latency incurred by transmitting the packet over link is also not included in this approach.

MAX1 in this approach is the bounded latency for all processing of a packet except for this link propagation latency P, equivalent to the speed of light across the transmission medium times the length of the medium, and MAX = MAX1 + P.

The serialization latencies latency across the sending interface on A and the receiving interface on B can be included in MAX1 though. It is only necessary for the measured timestamps zA and xB to be the logically for the same point in time assuming the link had a minimum length, e.g.: for a back to back connection. For example, the reference time is the time when the first bit of the packet can be observed on the shortest wire between A an B. A can most likely measure zA only some fixed offset o of time before r, hence needs to correct zA += o before calculating d. Likewise, B could for example only measure xB exactly after the whole packet arrived. this would be l * r later than the reference time, where r is the serialization rate of the receiving interface on B and l is the length of the packet. B would hence need to adjust d -= l * r to subtract this time from the time the packet needs to be delayed in D.

```
+-----------------------+      +-----------------------+
| Node A                |      | Node B                |
| +-+    +-+   +-+   +-+ |      | +-+    +-+   +-+   +-+ |
|-|F|--x-|D|-y-|Q|-z-|M|-|------|-|F|--x-|D|-y-|Q|-z-|M|-|
| +-+    +-+   +-+   +-+ |      | +-+    +-+   +-+   +-+ |
+-----------------------+      +-----------------------+
       |<- Dampened Q ->|             |<- Dampened Q ->|
             |<- A/B in-time latency ->|
             |<--A/B on-time latency ------->|
```

Figure 4: gLBF refined model

   Figure 4 shows a further refined model for simpler implementation. Larger routers/switches are typically modular, and forwarding happens on a different modular component (such as a linecard) than the queuing for the outgoing interface. Expecting clock synchronization even within such a large device is undesirable.

   In result, Figure 4 shows damper operation as occurring solely before enqueuing packets into Q and after dequeuing them. The Damper module measures the x timestamp, extracts d from the packet header, adjusts it according to the above described considerations and then delays the packet by that value and enqueues the packet afterwards. After the packet is dequeued, the Marking module measures the time z, adjusts it according to the previously described considerations calculates the value of d and overwrites the packet header before sending the packet.

2.2. Dampers and controller-plane

```
            Controller-plane:
           Path Control  and
            Admission Control
         ..      .      .      ..
        .        .       .       .
      +---+   +---+   +---+   +---+  Src---| A |---| B |---| C |---| D |
      +---+   +---+   +---+   +---+
        |       |       |       |
      +---+   +---+   +---+   +---+
      | E |---| F |---| G |---| H |--- Dst
      +---+   +---+   +---+   +---+
```

Figure 5: Path selection and gLBF example

   While the damper mechanism described so far can be realized with different queuing mechanisms and hence different calculus for every interface for which bounded latency can be supported, the role of the controller plane involves other components beside admission control, and those need to be possible to tightly be coupled with admission control for efficient use of resources.

Figure 5 shows the most common problem. The controller-plane needs to provide for a new traffic flow from Src to Dst a path through the network and reserve the resources. The most simple form for just bandwidth reservation is called Constrained Shortest Path First (CSPF). With the need to also provide end-to-end latency guarantees in support of bounded latency, not only does the path calculation becomes more complex, but many per-hop bounded latency queuing mechanisms support also to select more than one per-hop latency on the hop, and the controller-plane needs to determine for each hop of a possible hop, which one is best.

In result, the controller-plane needs to know exactly what parameters the bounded latency queuing mechanism on each hop/interface can have to perform these operations, and standardization of these parameters ultimately results in standardizing the bounded latency queuing mechanism - and not only the damper part.

3. gLBF specification (normative)

3.1. Damper with UBS queuing/calculus

guaranteed Latency Based Forwarding (gLBF) is using the queuing model of Urgency Based Scheduling (UBS), which is also used in TSN Asynchronus Traffic Shaping (TSN-ATS). This allows gLBF to re-use or co-develop one and the same controller-plane and its optimization algorithms for bandwidth and latency control for TSN-ATS or a DetNet L3 equivalent thereof and gLBF. Hence, gLBF should provide the most easily operationalised on-time solution for networks that want to evolve from an in-time model via TSN-ATS, or that even would want to operate both options in parallel.

Effectively, gLBF replaces the per-flow interleaves regulators of UBS with per-flow stateless damper operations. Where UBS only provides for in-time latency guarantees, gLBF provides in result in-time latency service, but with fundamentally the same calculus as UBS.

```
   +-------------------------------------------------+
   | UBS strict priority Queuing block               |
   |                                                 |
   |               +--------------+   +----------+  |
   | +-------+   /-| Prio 1 queue |---|          |  |
   | | Packet|  /  +--------------+   | Strict   |  |
-->| | Prio  |->        ...          -| Priority |  |--->
   | | enque |  \  +--------------+   | Schedule |  |
   | +-------+   \-| Prio 8 queue |---|          |  |
   |               +--------------+   +----------+  |
   +-------------------------------------------------+
```

Figure 6: Path selection and gLBF example

Strict Priority Scheduling removes packets always from the highest priority queue (1 is highest) that has a pending packet and forward it. UBS defines two options for the traffic model traffic flows. The more flexible one defines that each flow specifies a rate r and a maximum burst size b (in bits).

In result, the bounded latency of a packet in priority 1 is (roughly) the latency required to serialize the sum of the bursts of all the flows admitted into priority 1. The bounded latency of a packet in priority 2 is that priority latency plus the latency required to serialize the sum of the bursts of all the flows admitted into priority 2. And so on.

3.2. gLBF processing

The following text extends/refines the damper processing as necessary for gLBF.

```
   +------------------------+       +------------------------+
   | Node A                 |       | Node B                 |
   | +-+    +-+   +-+   +-+ |       | +-+    +-+   +-+   +-+ |
 --|-|F|--x-|D|-y-|Q|-z-|M|-|-------|-|F|--x-|D|-y-|Q|-z-|M|-|--->
IIF| +-+    +-+   +-+   +-+ |OIF IIF| +-+    +-+   +-+   +-+ |OIF
   +------------------------+       +------------------------+
          |<- Dampened Q ->|               |<- Dampened Q ->|
                |<- A/B in-time latency -->|
                |<--A/B on-time latency -------->|
```

Figure 7: refined gLBF processing diagram

The Delay module retrieves the delay from a packet header fields, requirements for that packet header field are defined in Section 3.4.

Because serialization speeds on the Incoming InterFace (IIF) will be different for different IIF, and because D will likely need to compensate the measured time x based on the packet length and serialization speed, an internal packet header should maintain this information. Note that this is solely an implementation consideration and should not impact the configuration model of gLBF.

The Queuing module is logically as described in UBS, except that the priority of a packet in gLBF is not selected based on per-flow state, but instead an appropriate packet header field of the packet is looked up in the Packet Prio Enque stage of Q and the packet accordingly enqueued based on that fields value. Possible options for indicating such a priority in a packet header field are defined below in Section 3.4.

Like Q, M also needs to look up the packet priority to know MAX1 for the packet and hence calculate d that it rewrites in the packet header.

The overall configuration data model to be defined for gLBF is hence the configuration data model for UBS, which is a set of priority queues and their maximum size, and in addition for gLBF for each of these queues a controller-plane calculated MAX maximum latency value to be used by M.

Given how the node knows the serialization speed of OIF, MAX1 for each of the queues could automatically be derived from the maximum length in bytes for each of the UBS priority queues, but this may not provide enough flexibility for fine-tuning (TBD), especially when packet downgrade as describe below is used.

TBD: paint a small yang-like data model, even though its trivial, like in the TCQF draft. This should primarily include additional diagnostic data model elements, such as maximum occupancy of any of the priority queus and number of overruns.

3.3. Error handling

A well specified standard for a damper mechanism such as described in this document needs to take care of error cases as well. The prime error condition is, when the sending node A of a gLBF packet recognizes that the packet was enqueued for longer than MAX1 and hence the to-be-calculated delay to be put into the packet would have to be < 0.

In this case, error signaling, such as ICMPv6/ICMP needs to be triggered (throttled !), and the packet be discarded - to avoid failure of admission control / congestion further down the path for other packets as well.

The data model (below) describes an optional data model that allows instead of discarding of the packet to signal an ECN like mechanism together with lower-priority forwarding of such packet to inform the receiver directly about the problem and allow the application to deal with such conditions better than through other error signaling.

3.4. Data Model for gLBF packet metadata

gLBF is specifically designated to support per-hop, per-flow stateless operations because it does not require any per-flow, but only per-packet metadata for its processing. While it is possible to

3.4.1. damper: 24 bit value, unit 1 usec.

This is a value potentially different for every packet in a flow and it is rewritten by every gLBF forwarder along the path.

gLBF requires a packet header indicating the delay that the damper on the next hop needs to apply to the packet. Dampening only needs to be accurate to the extent that synchronous delivery needs to be accurate. A unit of 1 usec is considered today to be sufficient for all purposes. The maximum size of delay is the maximum per-hop queuing latency. 65 msec is considered to be much more than ever desirable. Therefore, a 16 bit header field is sufficient.

Note that link propagation latency does not impact delay. Hence the size of delay does not create any constraint on the length of links.

3.4.2. end-to-end-priority: 3 bits

This is a field that needs to be the same for all packets of a flow so that they will be forwarded with the same latency processing and hence do not incur different latencies and therefore possible reordering. This field is written when the packet enters the gLBF path and only read.

8 Priorities and hence 8 different latencies per hop are considered sufficient on a per-hop basis. If prio is an end-to-end packet header field such as by using 8 different DSCP in IPv6/IP, this results in 8 different latency traffic classes.

3.4.3. hop-by-hop-priority: 3 bits (per hop)

This is a better, more advanced alternative to the end-to-end-priority. This data is optional. If it is present for a particular hop, then it supersedes the end-to-end-priority.

Because this data is per-hop, it should not be encoded in an end-to-end part of the packet header, but into a hop-by-hop part of the header:

Because DetNet traffic needs the resources of each flow to be controlled, re-routing of DetNet flows without the controller-plane is highly undesirable and could easily result in congestion. A per-hop, per-flow stateless forwarding mechanism such as SR-MPLS or SRv6 is therefore highly desirable to provide a per-hop steering field. The priority could easily be part of such a per-hop steering field by allocating dynamically, or on-demand up to 8 different SID (Segment IDentifiers), such as up to 8 MPLS labels, or using 3 bits of the parameter field of IPv6 SIDs.

When such per-hop priority is indicated, the controller-plane can support much more than 8 end-to-end latencies, simply by using different per-hop latencies. For example one flow may use priority 1-1-1-1-1 across four hops, the new slower flow may use 1-2-1-2 across 4 hops, the next one may use 2-2-2-2, and so on.

3.4.4. phop-prio: 3 bits

This field is re-written on every hop when hop-by-hop-priorities are used.

This is an optional field which may be beneficial in and end-to-end header if hop-by-hop priorities are used for forwarding in gLBF and the hop-by-hop-priority of the packet on the prior hop is not available from the hop-by-hop packet header anymore. This is typically the case in MPLS based forwarding, such as SR-MPLS because this information would have been removed. Note that this header field is only required in support of optional simplified high-speed forwarding implementation options.

3.4.5. Error handling data items

Di: one bit

Downgrade intent. This bit is set when the packet enters the gLBF domain and never changed.

Ds: one bit

Downgrade status. This bit is set to 0 when the packet enters the gLBF domain and potentially changed to 1 by one gLBF forwarder as described in the following.

When the Di bit is not set, and a gLBF forwarder recognizes that the packet can not be forwarded within its guaranteed latency, the packet is discarded and forwarding plane specific error signaling is triggered (such as IGMP/ICMPv6). Discarding is done to avoid causing latency errors further down the path because of this packet.

When this bit is set, the packet will not be discarded, but instead the Ds bit is set to indicate that the packet must not be forwarded with gLBF guaranteed latency anymore, but only with best or even lower-than-best effort.

Di and DS may be encoded into the two ECN bits for IP/IPv6 dataplanes. The required behavior should be backward compatible with existing ECN implementations should the packet unexpected pass a router that processes the packet not as gLBF.

3.4.6. Accuracy and sizing of the damper field considerations

This memo recommends that the damper field in packet headers has a size of 24 bits or larger and represents the dampening time with a resolution of 1 nsec. The following text explains the reasoning for this recommendation.

The accuracy needed for the damper value in the packet header as well as the internal calculations performed for gLBF depends on a variety of factors. The most important factor is the accuracy of the provided bounded latency as desired/required by the applications.

Even though Ethernet is defined as an asynchronous medium, the clock accuracy is required to be +- 100 ppm (part per million). For a 1500 byte packet across a 100 Mbps Ethernet the propagation latency difference between fastest and slowest clock is 24 nsec. If gLBF is to be supported also on such slower speed networks with multiple hops, then these errors may add up (?), and it is likely not possible to provide end-to-end propagation latency accuracy much better than 1 usec without requiring more transmission accuracy through mechanisms such as PTP - which gLBF attempts to avoid/minimize. For higher speed links, errors in short term propagation latency variation becomes irrelevant though.

In WAN network deployments, propagation latency is in the order of msec such as ca. 1 msec for 250 Km of fiber. Serialization latency on a 1gbps Ethernet is ca. 1 usec for a 128 byte packet. It is likely that an accuracy of propagation latency in the order of 1 usec is sufficient when the round-trip-time is potentially 1000 times faster.

In result of these two simple data points, we consider that the accuracy of end-to-end propagation latency of interest is 1 usec. To avoid introducing additive errors, the resolution of the damper value needs to be higher. This memo therefore considers to use 1 nsec resolution to represent damper values. This too is the value used in PTP.

The maximum value and hence the size of the damper field in packets depends on the maximum latency introduced in buffering on the sending node plus smaller factors such as serialization latency. This maximum is primarily depending on the slowest links to be supported. A 128 byte packet on a 100 Mbps link has ca. 10 usec propagation latency. With just 16 bit damper value with nsec accuracy this would allow only 6 packet buffers. This may be too low. Hence the damper field should at minimum be at least 24 bits.

3.5. Ingress and Egress processing

3.5.1. Network ingress edge policing

When packets enter a gLBF domain from a prior hop, such as a node from a different domain or a sending host, and that prior hop is sending gLBF packet markings, then the timing of the packet arrival as well as the markings in packet header(s) for gLBF MUST only be observed when that prior hop is trusted by the gLBF domain

If the prior hop can not be trusted for correct marking and/or timing of the packets, the first-hop gLBF node in the gLBF domain MUST implement a per-flow policer for the flow to avoid that packets from such a prior hop will cause problems with gLBF guarantees in the gLBF domain.

A policer is to be placed logically after the damper module of gLBF and before the queuing module.

Instead of a policer, the ingress edge node of the gLBF domain MAY instead use a per-flow shaper or interleaved regulator. When a prior hop sends for example packets of a flow with too high a rate, a shaper would attempt to avoid discard packets from such a flow until also the burst size of the flow is exceeded, by further delaying them. A policer would immediately discard any packets that does not meet their flows envelope.

3.5.2. gLBF sender types

3.5.2.1. Simple gLBF senders

Senders are expected to send their traffic flows such that their relative timing complies to the admitted traffic envelope. Senders that for example only send one flow, such as simple sensors or actors, typically will be able to do this.

When senders can actually do this (send packets for gLBF with the right timing), then these packets do not need to have gLBF packet header elements. Instead, this ingress network node can calculate the damper time locally from the following consideration to avoid any need for gLBF processing in the sender.

The maximum gLBF damper time in this case is the serialization time of the largest packet for gLBF received from the sender. Thus, when the sender sends smaller packets than the maximum admitted packet size, then the first hop network node needs to dampen packets based on that difference in serialization time.

3.5.2.2. Normal gLBF senders

When senders can not fully comply to send packets with their admitted envelope, then they need to implement gLBF and indicate in the packet header the gLBF damper value. For example, when the sender can not ensure that at the target sending time of a gLBF packet the outgoing interface is free, then that other still serializing packet will impact the timing of the following packet(s) for gLBF. Another reason is when the sender has to send packets from multiple flows and those can not or are are not generated in a coordinated fashion to avoid delay, then they could compete on the outgoing interface of the sender, introducing delay.

Normal gLBF senders simply need to implement the queue and marking stage of gLBF, but not the dampening stage, because that only applies to receivers/forwarders.

3.5.2.3. Non gLBF senders

When senders can not be simple gLBF senders but can also not implement the gLBF queuing and marking stage, then the following first hop node of the gLBF domain needs to treat them like untrusted prior hop but always use a shaper or interleaved regulator so as not to discard their packets.

3.5.3. receivers and gLBF

3.5.3.1. Normal gLBF receivers

Normal gLBF receivers can process packet gLBF markings and need to therefore implement the gLBF dampening block.

3.5.3.2. gLBF incapable receivers

When receivers can not implement the gLBF dampening stage, then the worst-case burst that may arrive at the receiver is to be counted as added jitter to the end-to-end service.

It is impossible to let the network eliminate such bursts without additional coordination and gating of packets across all senders and their network ingress nodes by gating of those packets. This is not a gLBF specific issue, but simply a limitation of the (near) synchronous service model offered by gLBF and equally exists in any delay and latency model with this type of service:

Consider a set of N senders conspire to generate a burst at the same receiver. They do know from the admission control model the latency each of them has to that receiver and can thus time the generation of packets such that the last-hop router would have to send all those N packets (one from each sender) in parallel on the interface. Which is obviously not possible.

With gLBF capable receivers, this possible burst is taken into account by including the latency introduced by such a worst-case burst into the end-to-end latency through the controller-plane, and indicating the actual latency that the packet needs to be dampened in the gLBF header field to the receiver. But when the receiver does not support such dampening, then that maximum last-hop burst-size simply turns into possible jitter.

3.5.4. Further considerations

A host may be a different type of gLBF sender than it is gLBF receiver. In a common case in cloud applications, a container or VM in a data center may the a sender/receiver, and such an application may be considered to be trusted by the gLBF domain if it is an application of the network operator itself (such as part of a higher level service of the network operator), but not when it is a customer application.

gLBF marking needs to happen after contention of packets from all applications, so it is something that can considered to happen inside the operating system. This operating system of such a host may not (yet) implement gLBF, so it may not be possible to correctly generate the gLBF marking as a sender. Instead, the application may generate the appropriate packet markings for steering and gLBF, but may need to leave the damper marking incorrect.

On the other hand, dampening received gLBF packets on a receiver can happen at the application level, so that the operating system does not need to do it. Such dampening is exactly the same functionality that in applications is normally called "playout buffering", except that it will be a much smaller amount of delay time because playout buffering needs to take the whole path delay into account, whereas gLBF dampening is only for the prior hop.

3.5.5. Summary

There is a non-trivial set of options for ingress/egress processing that could be beneficial to simplify and secure dealing with different type of sender and receivers.

Later versions of this memo can attempt to define specific profiles of edge behavior to limit the recommended set of implememtation options for sender/receivers and gLBF edge nodes.

4. Controller-plane considerations (informative)

4.1. gLBF versus UBS / TSN-ATS

By relying on [UBS] for both the traffic model as well as the bounded latency calculus, gLBF should be easy to operationalize by relying on controller-plane implementations for TSN-ATS: UBS/TSN-ATS and gLBF provide the same bounded latency and use the same model to manage bandwidth for different flows: By calculating which flow needs to go on which hop into which priority queues.

The main change to a UBS/TSN-ATS controller-plane when using gLBF is that when a flow is to be admitted into the network, removed, or rerouted. In UBS, each of these operations imply that the controller-plane needs to signal to each node along the path the traffic flow parameters so the forwarding plane can establish the per-hop,per-flow state for the flow. Depending on network configuration this will also imply configuration of the next-hop for the flow for the routing.

For gLBF hop-by-hop operations, the first hop needs to receive the per-path or per-hop priority information that is then imposed into an appropriate packet header for the flows packets.

4.2. first-hop policing

In gLBF, the controller-plane has to perform this action only against the ingress node to the gLBF domain. The traffic parameters such as rate and burst size are only relevant to establish a policer so that the flow can not violate the admitted traffic parameters for the flow. This "policing" on the first hop is actually a function independent of gLBF, UBS or any other method used across the path.

4.3. path steering

gLBF operated independently of the path steering mechanism, but the controller-plane will very likely want to ensure that gLBF (or for that matter any DetNet) traffic does not unexpectedly gets rerouted by in-networking routing mechanisms because normally it will or can not reserve resources for such re-routed flows on such arbitrary failure paths without significant additional effort and/or waste of resources.

5. IANA considerations

None yet.

6. Changelog

[RFC-editor: please remove]

00 Initial version.

01 Added use cases from Stefan

02 refresh only, waiting for progress in WG discussion

03 refresh, affiliation change of co-author

04 refresh, still waiting for more discuss in detnet

05 refresh, still waiting for more discuss in detnet

06 added Solution Scope, Taxonomy and Status for DetNet adoption considerations

07 Fixed author address (hommes)

7. References

7.1. Normative References

[RFC2210] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, DOI 10.17487/RFC2210, September 1997, <https://www.rfc-editor.org/rfc/rfc2210>.

[RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <https://www.rfc-editor.org/rfc/rfc2212>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/rfc/rfc2474>.

[RFC3270] Le Faucheur, F., Ed., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <https://www.rfc-editor.org/rfc/rfc3270>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/rfc/rfc8200>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/rfc/rfc8655>.

[RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <https://www.rfc-editor.org/rfc/rfc8964>.

7.2. Informative References

[BIER-TE] Eckert, T. T., Menth, M., and G. Cauchie, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, draft-ietf-bier-te-arch-13, 25 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-bier-te-arch-13>.

[CQF] IEEE Time-Sensitive Networking (TSN) Task Group., "IEEE Std 802.1Qch-2017: IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding (CQF)", 2017.

[DNBL] Finn, N., Le Boudec, J., Mohammadpour, E., Zhang, J., and B. Varga, "Deterministic Networking (DetNet) Bounded Latency", Work in Progress, Internet-Draft, draft-ietf-detnet-bounded-latency-10, 8 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-bounded-latency-10>.

[gLBF] Eckert, T., Clemm, A., and S. Bryant, "gLBF: Per-Flow Stateless Packet Forwarding with Guaranteed Latency and Near-Synchronous Jitter,", IEEE 2021 17th International Conference on Network and Service Management (CNSM), Izmir, Turkey, doi 10.23919/CNSM52442.2021.9615538, October 2021, <https://dl.ifip.org/db/conf/cnsm/cnsm2021/1570754857.pdf>.

[gLBF-Springer2023]
Eckert, T., Clemm, A., and S. Bryant, "High Precision Latency Forwarding for Wide Area Networks Through Intelligent In-Packet Header Processing (gLBF)", Springer Journal of Network and Systems Management, 31, Article number: 34 (2023), doi https://doi.org/10.1007/s10922-022-09718-9, February 2023.

[I-D.dang-queuing-with-multiple-cyclic-buffers]
Liu, B. and J. Dang, "A Queuing Mechanism with Multiple Cyclic Buffers", Work in Progress, Internet-Draft, draft-dang-queuing-with-multiple-cyclic-buffers-00, 22 February 2021, <https://datatracker.ietf.org/doc/html/draft-dang-queuing-with-multiple-cyclic-buffers-00>.

[I-D.eckert-detnet-bounded-latency-problems]
Eckert, T. T. and S. Bryant, "Problems with existing DetNet bounded latency queuing mechanisms", Work in Progress, Internet-Draft, draft-eckert-detnet-bounded-latency-problems-00, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-bounded-latency-problems-00>.

[I-D.eckert-detnet-flow-interleaving]
Eckert, T. T., "Deterministic Networking (DetNet) Data Plane - Flow interleaving for scaling detnet data planes

with minimal end-to-end latency and large number of flows.", Work in Progress, Internet-Draft, draft-eckert-detnet-flow-interleaving-03, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-flow-interleaving-03>.

[I-D.ietf-detnet-dataplane-taxonomy]
Joung, J., Geng, X., Peng, S., and T. T. Eckert, "Dataplane Enhancement Taxonomy", Work in Progress, Internet-Draft, draft-ietf-detnet-dataplane-taxonomy-05, 8 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-dataplane-taxonomy-05>.

[I-D.stein-srtsn]
Stein, Y. J., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://datatracker.ietf.org/doc/html/draft-stein-srtsn-01>.

[IEEE802.1Q]
IEEE 802.1 Working Group, "IEEE Standard for Local and Metropolitan Area Network —— Bridges and Bridged Networks (IEEE Std 802.1Q)", doi 10.1109/ieeestd.2018.8403927, 2018, <https://doi.org/10.1109/ieeestd.2018.8403927>.

[IEEE802.1Qbv]
IEEE Time-Sensitive Networking (TSN) Task Group., "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic (TAS)", 2015.

[IPV6-PARMS]
"Internet Protocol Version 6 (IPv6) Parameters", IANA , n.d., <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>.

[LBF] Eckert, T. and A. Clemm, "High-Precision Latency Forwarding over Packet-Programmable Networks", IEEE 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020), doi 10.1109/NOMS47738.2020.9110431, April 2020.

[LDN] Liu, B., Ren, S., Wang, C., Angilella, V., Medagliani, P., Martin, S., and J. Leguay, "Towards Large-Scale Deterministic IP Networks", IEEE 2021 IFIP Networking Conference (IFIP Networking), doi 10.23919/IFIPNetworking52078.2021.9472798, 2021, <https://dl.ifip.org/db/conf/networking/networking2021/1570696888.pdf>.

[LSDN] Qiang, L., Geng, X., Liu, B., Eckert, T. T., Geng, L., and G. Li, "Large-Scale Deterministic IP Network", Work in Progress, Internet-Draft, draft-qiang-detnet-large-scale-detnet-05, 2 September 2019, <https://datatracker.ietf.org/doc/html/draft-qiang-detnet-large-scale-detnet-05>.

[multipleCQF] Finn, N., "Multiple Cyclic Queuing and Forwarding", October 2021, <https://www.ieee802.org/1/files/public/docs2021/new-finn-multiple-CQF-0921-v02.pdf>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/rfc/rfc3209>.

[RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <https://www.rfc-editor.org/rfc/rfc4875>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/rfc/rfc8296>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/rfc/rfc8402>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/rfc/rfc8754>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/rfc/rfc8938>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/rfc/rfc8986>.

[RFC9016] Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D. Fedyk, "Flow and Service Information Model for Deterministic Networking (DetNet)", RFC 9016, DOI 10.17487/RFC9016, March 2021, <https://www.rfc-editor.org/rfc/rfc9016>.

[RFC9262] Eckert, T., Ed., Menth, M., and G. Cauchie, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", RFC 9262, DOI 10.17487/RFC9262, October 2022, <https://www.rfc-editor.org/rfc/rfc9262>.

[RFC9320] Finn, N., Le Boudec, J.-Y., Mohammadpour, E., Zhang, J., and B. Varga, "Deterministic Networking (DetNet) Bounded Latency", RFC 9320, DOI 10.17487/RFC9320, November 2022, <https://www.rfc-editor.org/rfc/rfc9320>.

[SCQF] Chen, M., Geng, X., Li, Z., Joung, J., and J. Ryoo, "Segment Routing (SR) Based Bounded Latency", Work in Progress, Internet-Draft, draft-chen-detnet-sr-based-bounded-latency-03, 7 July 2023, <https://datatracker.ietf.org/doc/html/draft-chen-detnet-sr-based-bounded-latency-03>.

[TCQF] Eckert, T. T., Li, Y., Bryant, S., Malis, A. G., Ryoo, J., Liu, P., Li, G., and S. Ren, "Deterministic Networking (DetNet) Data Plane - Tagged Cyclic Queuing and Forwarding (TCQF) for bounded latency with low jitter in large scale DetNets", Work in Progress, Internet-Draft, draft-eckert-detnet-tcqf-12, 15 December 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-tcqf-12>.

[TSN-ATS] Specht, J., "P802.1Qcr - Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping", IEEE , 9 July 2020, <https://1.ieee802.org/tsn/802-1qcr/>.

[UBS] Specht, J. and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", IEEE 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016.

Appendix A. High speed implementation considerations

A.1. High speed example pseudocode instead of reference pseudocode

This memo does not describe a normative reference code that aligns with the normative functional description above Section 3. The primary reason is that a naive 1:1 implementation of the functional description would lead to inferior performance. Nevertheless, the feasibility of high speed implementations is an important factor in the ability to deploy mechanisms like gLBF. Therefore, this appendix describes considerations for such high speed implementations including pseudocode for on possible option.

[gLBF] describes two approaches for possible optimized hardware implementation approaches for gLBF. One using "Push In First Out" (PIFO) queues, the other one times FIFO queues. The following is a representation and explanation of that second approach because it is hopefully more feasible for nearer term hardware implementations given the fact that scalable FIFO high-speed hardware implementations are still a matter for research.

However, the timed FIFO style algorithm presented here is also subject to possible scalability challenges for hardware because it requires for each outgoing interface O(IIF * priorities^2) number of FIFO queues, where IIF is the number of (incoming) interfaces on the node, and priorities is the number of priority levels desired (TSN allows up to 8). If however the priority of flows (packets) used is not per-hop but the same for every hop (per-path), then the number is just (IIF * priorities).

A.2. UBS high speed implementations

The algorithm for the pseudocode shown in this appendix further down in this section is based on the analysis of gLBF behavior done in and for [gLBF]. This analysis re-applies the same type of analysis and algorithm that was done in [UBS] and can be used to implement TSN-ATS in high speed switching hardware. Therefore, this appendix will start by explaining the UBS mechanisms.

UBS (see [UBS], Figure 4), logically consists of two separate stages. The first stage is a per priority, per incoming interface (IIF) interleaved regulator. An interleaved regulator is a FIFO where dequeuing of the queue head packet (qhead) is based on the per-flow state of qhead.

A target departure time for that qhead is calculated from the flow state of the packet maintained on the router across packets and once that departure time is reached, the packet is passed to one of the egress strict priority queues. This is called interleaved regulator, because the FIFO will have packets from multiple flows (all from the same incoming interface and outgoing priority), hence 'interleave', and regulator because of the delaying of the packet up to the target departure time.

The innovation of interleaved regulators as opposed to the prior per-flow shapers as used in [RFC2210] is the recognition and mathematical proof that the arrival and target departure times of all packets received from the same IIF (prior hop node) and the same egress priority will be in order and that they can be enqueued into a single FIFO where the per-flow processing only needs to happen for the qhead. Without (anything but negligible) added latency vs. per-flow shapers.

With this understanding, the interleaved regulator and following strict priority stage can easily be combined into a single queuing stage with appropriate scheduling. In this "flattened" approach, each logical egress strict priority queue consists of the per-IIF interleaved regulator (FIFO queue) for that priority.

Scheduling of packets on egress does now need to perform the per-queue processing of the per-flow state of the qhead, and then take the target departure time of that packet into account, selecting the interleaved regulator with the highest priority and earliest target departure time qhead.

A.3. gLBF compared to UBS

In gLBF, the very same line of thoughts as in UBS can be applied and are the basis for the described algorithm and shown pseudocode.

In glBF, the order of packets in their arrival and target departure times depends on the egress priority, the IIF, but also on the priority the packet had on the prior hop (pprio). All packets with the same (IIF,prio,prio) will arrive in the same order in which they need to depart, and in which it is therefore possible to use a FIFO to enqueue the packets and only do timed dequeuing of the qhead.

Both prio and pprio are of interest, because when gLBF uses an encapsulation allowing per-hop priority indications, its priority on each hop can be different. In UBS, priority can equally be different across hops for the same packet, but the prior hop priority is not necessary to put packets into separate interleaved regulators because UBS targets in-time delay, where the interleaved regulator does (only) need to compensate for burst accumulation, but in gLBF the delay to be introduced depends on the damper value which depends on the prior hop priority and in result, the order in which in gLBF packets should depart (absent any burst accumulation) depends also on the prior hop priority.

Because gLBF does not deal with per-flow state on the router as UBS does in its interleaved regulators, the FIFOs for gLBF are called timed FIFOs in this memo and not interleaved regulators

A.4. Comparison to normative behavior

Whereas the normative description described D, Q and M blocks, where the D block performs the dampening, and the Q block performs the priority queuing block, in the high speed hardware optimization, these are replaced by a single DQ block where the packets are enqueued into a single stage per-(IIF,pprio,prio) FIFO (similar to UBS) and then the dampening happens (similar to UBS) on dequeuing from that stage by the damper time but scheduling/prioritizing packets based on their prio.

```
   +-------------------------+       +-------------------------+
   | Node A                  |       | Node B                  |
   | +-+    +-------+   +-+ |       | +-+    +-------+   +-+ |
 --|-|F|--x-| DQ y  |-z-|M|-|-------|-|F|--x-| DQ y  |-z-|M|-|--->
IIF| +-+    +-------+   +-+ |OIF IIF| +-+    +-------+   +-+ |OIF
   +-------------------------+       +-------------------------+
          |<- Dampened Q ->|               |<- Dampened Q ->|
                 |<- A/B in-time latency ->|
                 |<--A/B on-time latency -------->|
```

Figure 8: High Speed single stage gLBF

In the normative definition, the damper value to be put into the packet depended on the time y, where the packet left the damper stage D and entered the priority queue stage Q. Simplified, the damper value is (MAX1 - (z - y)).

In the high-speed implementation, the packets are not passed from a damper state queue to a priority queue. Instead they stay in the same queue. This is on one hand one of the core reasons why this approach can support high speed - it does not require two-stage enqueuing/dequeing/ But it eliminates also the ability to take a time stamp at time y.

To therefore be able to replicate the normative gLBF behavior with a single stage, it is necessary in the marking stage M (where the new damper value is calculated), to somehow know y.

Luckily, y is exactly the target departure time of the packet for the D and therefore also the DQ stage, so already needs to be calculated on entry to DQ by calculation: y = (x + damper - <details>), as explained below.

The pseudocode consists of glbf_enqueue() describing the enqueuing into the DQ stage, and glbf_dequeue() describing the dequeuing from the DQ stage. glbf_dequeue() (synchronously) calls glbf_send() which represents the marking stage M.

A.5. Pseudocode

A.5.1. glbf_enqueue()

```
void glbf_enqueue(pak,oif) {
  tdamp = pak.header.tdamp // damper value from packet
  prio  = pak.header.prio  // this hops packet prio
  pprio = pak.header.pprio // previous hops packet prior
  iif   = pak.context.iif  // incoming interface of packet

  ta = adj_rcv_time(now(),                   \[1]
                    pak.context.iif.speed,
                    pak.context.length)

  td = now() + ta // (dampened earliest) departure time
  pak.context.max1 = max1\[oif,prio]
  enqueue(pak, td, q(oif,iif,prio,pprio))
}
```

Figure 9: Reference pseudocode for glbf_enqueue()

pak.header are parsed fields from the received gLBF packet. tdamp is the damper value. pak.context is a node local header of the packet. iif is the interface on which the packet was received.

prio and pprio are current prior hop priority from the packet header. If for example SRH ([RFC8754]) is used in conjunction with gLBF, and [RFC8986] formatting of SIDs is used, then the priority for each hop could be expressed as a 4-bit SID ARG field (see [RFC8986], section 3.1) to indicate 16 different priorities per hop.

Parsing the prior hop priority is not a normative requirement of gLBF, but a specific requirement of the high speed algorithm presented here to allow the use of single stage timed FIFOs instead of PIFOs. This type of parsing is not required for SRH, but would be a benefit of a packet header which like SRH keeps the prior hops information, opposed to the SR-MPLS approach where past hop SD/Labels are discarded.

now() takes a timestamp from a local (unsynchronised) clock at the time of execution.

[1] pak.context.ta (time of arrival) is the calculated time at which the first bit of the packet was entering the incoming interface of the node. adj_rcv_time() is defined in the next subsection. td is the target departure time calculated from the current time and the damper value from the packet. It simply convert the relative damper value to an absolute timestamp.

max1[oif,prio] is the only gLBF specific interface level state maintained (read-only) across packets. It is the maximum time calculated by admission control in the controller-plane for each priority. It is transferred into a context header field here even though it will only be used further down in dequeuing because of the assumption that access to state information is not desirable at the time critical stage of dequeuing/serialization, whereas such state lookup is very common for many forwarding plane features before enqueuing.

Finally, the packet is enqueued into a per-oif,iif,prio,pprio timed FIFO queue.

A.5.2. adj_rcv_time()

To calculate the reference time against which the damper value in the packet is to be used, this specification assumes the time when the first bit of the packet is seen on the media connecting to the incoming interface.

```
time_t adj_rcv_time(now, speed, length) {
  time_t now
  int speed, length

  return now - length * 8 / speed - FUDGE
}
```

Figure 10: Example pseudocode for adv_rcv_time()

In most simple node equipment, the primary variable latency introduced between that (first bit) measurement point and the time when the time parameter t for adj_rcv_time() can be taken is the deserialization time of the packet. This can easily be calculated from the packet length as assumed to be known from pak.context.length and the bitrate of the incoming interface known from pak.context.iif.speed plus fixed measured or calculated other latency FUDGE through internal processing. The example pseudocode, those are assumed to be static.

If other internal factors in prior forwarding within the node do create significant enough variation, then those may need compensation through additional mechanisms. In this case, instead of as shown in glbf_enqueue(), the reference time for calculation should be taken directly upon receipt of the packet, before entering the forwarding stage F, and then used as the now parameter for adj_rcv_time(). This is explicitly not shown in the code so as to highlight the most simple implementation option that should be sufficient for most node types.

A.5.3. glbf_dequeue()

```
void glbf_dequeue(oif) {
  next_packet: while(1) {
    tnow = now()
    ftd = tnow + 1  // time of first packet to send
    fq = NULL      // queue of first packet to send
    foreach prio in maxpriority...minpriority {
      foreach iif in iifs {
        foreach pprio in priorities}
          // find qhead with earliest departure time
          if (q = q(oif,iif,prio,pprio).qhead) {
            td = q.qhead.td   // target (departure) time
            if td < ftd
              ftd = td
              fq = q
            }
          }
        }
      }
      if fq { // found packet
        pak = dequeue(q,oif)
        glbf_send(oif,tnow,pak)
        break next_packet
      }
    }
  }
}
```

Figure 11: Pseudocode for glbf_dequeue()

glbf_dequeue() finds the gLBF packet to send as the enqueued packet with the highest priority and the earliest departure time td (as calculated in glbf_enqueue()), where td must also not in the future, because otherwise the packet still needs to be dampened.

The outer loops across this hops prio will find the packet with exactly those properties, dequeues and sends it. For this, the strict priority is expressed through the term maxpriority...minpriority, which first searches from maximum to minimum priority..

The two inner loops simply look for the packet with the earliest target departure time across all the (IIF,pprio) timed FIFO queues for the same prio and OIF.

In ASIC / FPGA implementations, finding this packet is not a sequential operation as shown in the pseudocode, but can be a single clock-cycle parallel compare operation across the td values of all queues qheads - at the expense of chip space, similar to how TCAMs work.

A.5.4. glbf_send()

```
void glbf_send(oif,tnow,pak) {
  qtime = tnow - pak.context.td          // \[1]
  td = pak.context.max1 - qtime - FUDGE2 // \[2]
  pak.header.tdamp = td
  serialize(oif,pak)
}
```

Figure 12: Reference pseudocode for glbf dequeue

Sending packet pak represents the marking block M in the specification. This needs to calculate the new damping value t and update it in the packet header before sending the packet.

td needs to be the maximum time max1 that the packet could have stayed in priority queuing minus the time qtime that it actually did stay in the queue and minus any additional time FUDGE2 that the packet will still needs to be processed by the router before its first bit will show up on the sending interface. This is calculated as shown in in [1] and [2] of Figure 12.

The primary factor impacting FUDGE2 is whether or not this calculation and updating of the packet header td field can be done directly before serialization starts, or whether it potentially would need to be done while there is still another packet in the process of being serialized on the interface. Taking the added latency of a currently serializing packet into account can for example be implemented by remembering the timestamp of when that packet started to be serialized and its length - and then taking that into account to calculate FUDGE2.

A.6. Performance considerations

IEEE 802.1 PTP clock synchronization does need to capture the accurate arrival time of PTP packets. It is also measuring the residency time of PTP packets between receiving and sending the packet with an accuracy of nsec and add this to a PTP packet header field (correctionField). Given how this needs to be a hardware supported functionality, it is unclear whether there is a relevant difference supporting this for few PTP signaling packets as compared to a much larger number of gLBF data packets - or whether it is a good indication of the feasibility packet processing steps gLBF requires.

Appendix B. History and comparison with other bounded latency methods

Preceding this work, the best solution to solve the requirements outlined in this document, where [TCQF] and [SCQF], which are proven to be feasible for high speed forwarding planes, because of vendor high-speed forwarding plane implementation using low-cost FPGA for cyclic queuing. On the other hand, it was concluded from implementation analysis, that [UBS] was infeasible for the same type of high-speed, low-cost hardware due to the need for high-speed flow-state operations, especially read/write cycles to update the state for every packet at packet processing speeds.

While TCQF and [SCQF] are very good solution proven to work at high-speed, low cost, available for deployment and ready for standardization, the need for network wide and hop-by-hop clock synchronization (albeit at lower accuracy than required for other mechanisms feasible at high-speed, low-cost) as well as the need to find network wide good compromise clock cycle times makes planning and managing them in dynamic, large-scale and/or low-cost network solutions still more complex to operationalize than what the authors wished for.

In parallel to short term operationalizable solutions [TCQF] and [SCQF], [LBF] was researched, which is exploring a wide range of options to signal and control latency through advanced per-hop processing and in-packet latency related new header elements. Unfortunately, with the flexibility of [LBF] it was impossible to find a calculus for simple admission control. Ultimately, [gLBF] was designed out of the desire to have a mechanism derived from [LBF] that also provides guaranteed bounded latency, has the flexibility benefits of [UBS] with respect to fine-grained and per-hop independent latency management but does hopefully still fits the limits of what can be implemented in high-speed, low-cost forwarding/ queuing hardware.

This high-speed hardware forwarding feasibility of gLBF has yet to be proven. Here are some comsiderations, why the authors think that this should be well feasible:

The total number of FIFO that a platform needs to support is comparable to the number of queues already supportable in the same type of devices today, except that service provider core nodes may not all implement that many, because absent of DetNet services, there is no requirement for them.

The need to implement timed FIFOs (if the proposed high-speed implementation approach is used) is equal or less complex to shapers, which by now have also started to appear on high-speed (100Gbps and faster) core routers, primarily due to high-speed virtual leased line type of services.

The re-marking of packet header fields derived from the calculated latency of the packet experienced in the node is arguably something where iOAM type functionality likely provides also prior evidence of feasibility in high speed hardware forwarding planes. Similarily, processing of PPT timing protocol packets also have similar requirements.

Even when implementation challenges at high-speed, low-cost make gLBF a longer term option in those networks, implementation at low-speed (1/10 Gbps) such as in manufacturing or cars may be an interesting option to explore given how it has the no-jitter benefits of [CQF] without the need for cycle management or clock-synchronization: Best of both worlds [CQF] and TSN-ATS ??

Appendix C. Solution Scope, Taxonomy and Status for DetNet adoption considerations

C.1. Solution scope

gLBF is proposed as a solution not standalone, but in conjunction with [I-D.eckert-detnet-flow-interleaving] as a per-flow mechanism to be used on the ingress gLBF router to allow more flexible per-flow management of admission of flow bursts into cycles than presented in this document. The reason for splitting up the proposal into two documents is because [I-D.eckert-detnet-flow-interleaving] can be combined with all "mostly-synchronous" per-hop mechanisms, for example beyond gLBF also (of course) CQF and [TCQF].

C.2. Taxonomy

Using the section number/titles of the taxonomy document [I-D.ietf-detnet-dataplane-taxonomy] the TCQF technology can be classified as follows. In summary, gLBF is very much like TCQF except that it removes cycles and replaces them with Dampers, hence even further reducing the end-to-end jitter, reducing clock accuracy requirements (only phase synchronization, not clock synchronization required) and removed the complexity of end-to-end cycle management. In comparison to older Damper based proposals, gLBF re-uses ATS calculus, which allows to re-use ATS admission control and delaycalculations and to use simple FIFO with timed scheduler based implementation (like ATS).

* 4.1 Per Hop Dominant Factor for Latency Bound

  The per-hop dominant latency bound is Category 3. The per-hop latency for each class is determined by the configured burst size cycle size.

* 5.1 Periodicity

  gLBF is not periodic. Its fundamental mode of operation is not based on cycles to manage bursts but instead by using per-hop per-packet synchronous delivery to allow maintaining the burst sizes across all hops end-to-end.

* 5.2. Traffic Granularity

  Factually, gLBF is class-level granularity. However, the description in [I-D.ietf-detnet-dataplane-taxonomy], revision 04 claiming that all packets within a class are treated the same is questionable for gLBF (and in general all Damper solutions). In Damper solutions like gLBF, packets latency is calculate per-packet (within the same class) based on prior experienced per-hop latency - to compensate it.

* 5.3. Time Bounds

  gLBF is per-hop bounded. It is Right Bounded by the configured burst size for the hop and class the packet is in. It is Left bounded by the signalled remaining latency to achieve synchronous forwarding. If observation of bound was not taken on e.g.: the wire but instead in a specific forwarding step in the router(after reception, before enqueuing), then the per-hop bound would be extremely tight bounded, limited only by implementation accuracy.

* 5.4 Service Order

gLBF service order is solely arrival based.

* 1. Suitable Categories for DetNet

  TCQF is "6.5. Class level non-periodic bounded category" according to the categorization.

C.3. Status

gLBF has primarily an open source simulation level validation implementation as described in [gLBF]. Additional implementations where done in Linux kernel by University (TBD reference) and a second university in conjunction with a Prolog based Admission Control Server implementation (TBD reference).

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies USA
2220 Central Expressway
Santa Clara, CA 95050
United States of America
Email: tte@cs.fau.de

Alexander Clemm
Sympotech
CA
United States of America
Email: ludwig@clemm.org

Stewart Bryant
Independent
United Kingdom
Email: sb@stewartbryant.com

Stefan Hommes
ZF Friedrichshafen AG
Germany
Email: stefan.hommes@zf.com