gLBF: Per-Flow Stateless Packet Forwarding with Guaranteed Latency and Near-Synchronous Jitter

Toerless Eckert, Alexander Clemm
Futurewei, USA
Stewart Bryant
Institute for Communication Systems, University of Surrey, UK

***Abstract*—This paper contributes gLBF (guaranteed Latency-Based Forwarding), a new algorithm for leaky bucket defined traffic flows, feasible for low-cost, high-speed forwarding planes in LAN/RAN/WAN networks to support simply computed, per-hop bounded latency for high-precision communication services. gLBF combines benefits of earlier algorithms such as Urgency-Based Scheduling (UBS) and Cyclic Queuing and Forwarding (CQF) while avoiding their downsides. gLBF does not require per-hop, per-flow state such as shapers or (interleaved) regulators, thus avoiding control complexity and limiting hardware cost. It does not require strict network-wide time synchronization and therefore eliminates the opex and capex of deploying IEEE1588. It minimizes end-to-end jitter, thus eliminating the need for playout buffers in endpoints to compensate for network jitter.**

***Index Terms*—deterministic, latency, synchronous, guarantees, synchronous, gLBF, LBF, CQF, UBS, TSN, LDN, TCQF, NewIP**

I. Introduction

A growing number of applications require communication services that are able to forward traffic with guaranteed or deterministic latency bounds. Such services have been evolving for decades, mostly in limited physical domain networks such as inside vehicles or on factory floors for applications involving industrial machinery [18].

Standards for deterministic bounded maximum latency guarantees in packet forwarding networks were defined in the latter part of the 1990s via the IETF "Guaranteed Services" (GS) standard [17], but it was never adopted in significant deployments. Instead, deterministic latency services first saw wider deployments in Ethernet networks via vendor proprietary solutions that where replaced over time with several IEEE "Time Sensitive Networking" (TSN) standards including "Cyclic Queuing and Forwarding" (CQF) [10] and later "Asynchronous Traffic Shaping" based on the "Urgency Based Scheduling" (UBS) algorithm [18]. Both are precursors to the work presented here.

With TSN enjoying increasing adoption at Layer 2, there is interest in revisiting deterministic service also in the context of IP and for use across wide-area networks beyond the typically more limited physical scope of TSN networks. This resulted in the formation of the DetNet working group in the IETF. Many core use cases build around high-speed wide-area IP networks [9], which has led to a re-investigation of the algorithmic requirements to best support these services in such networks.

This paper proposes a novel algorithm called "guaranteed Latency Based Forwarding" (gLBF) for deterministic latency in large scale (tenths of thousands of flows), high speed (link speeds $\geq$ 100 Gbps) and wide-area (radius $\geq$ 10 km) networks, including IP and MPLS networks. Its benefits could also be useful for other networks.

gLBF eliminates what the authors considers to be problems of prior algorithms for this type of deployments: per-hop, per-flow state and needs for network-wide clock synchronization. gLBF provides tightly bounded jitter, which the authors consider crucial for many applications, and it provides the same simple latency calculus as UBS, therefore providing the same latencies as UBS, as well as its support of different guaranteed latencies for different flows across the same paths and compatibility with UBS Path Computation Engines (PCE) and resource reservation systems.

gLBF requires what today can be called more advanced queuing mechanisms than what UBS requires, such as "PIFO" (Push-In-First-Out") queues, which may take more time to get implemented into high-speed, low-cost forwarding hardware. A mechanism is described to support a subset of gLBF with queuing mechanisms equivalent to those required for UBS.

The rest of the paper is organized as follows: Section II discusses the current state of the art: Section II-A introduces a model for the network architecture in which gLBF and prior bounded latency mechanisms operate and Section II-B introduces the relevant prior art latency control mechanisms. Section III contributes an analysis of their problems for the purpose of bounded latency, Section IV describes the main contribution of this paper, gLBF and its different implementation options, Section V describes the authors experimental validation of gLBF, and Section VI concludes the paper and discusses desirable next steps.

II. Preliminaries

A. Common system model

Notwithstanding other options to build a complete network solution around gLBF or the other discussed prior algorithms, we describe Figure 1 as the common reference system model for the purpose of this paper.

The traffic of interest is controlled via an Admission Controller (AC) that tracks all flows and their parameters that are guaranteed to be delivered across the network with deterministic latency. Optionally, there may also be a component called

978-3-903176-36-2 ©2021 IFIP

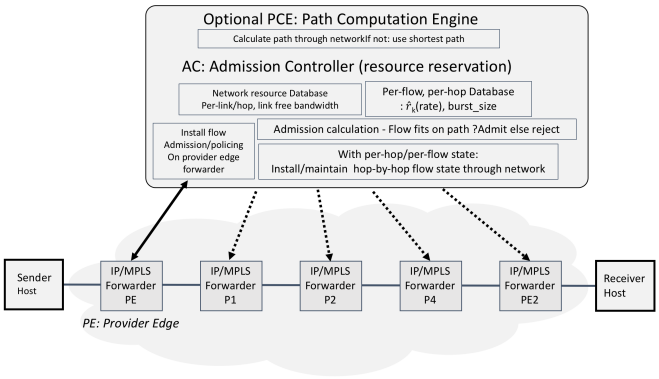

Fig. 1. Common Reference System Model

a Path Computation Engine (PCE) [4] which supports the AC function by calculating the optimum path through the network for each individual traffic flow.

Traffic enters the network via the network ingress or first-hop router, also called the Provider Edge (PE) router. The PE is responsible for ensuring that traffic does not exceed the bounds the AC determines that it should have. This means that the ingress PE will have per-flow state policing and/or will provide adjustment to traffic flows entering the network. Likewise, the PCE may instruct the PE to cause the traffic flow to be routed across a particular path. Traffic is then routed across zero or more Provider (Core) Routers (P routers) towards an egress PE. In "stateless" solutions, P routers will not have per-flow state. Per-flow state may exist purely to steer traffic and/or to perform per-flow shaping in support of bounded latency. The latter type of state has in the past been considered to be most expensive in high-speed networks.

B. Precursors and related technologies

This section explains and reviews technologies to the extent that they contributed to this papers work. A broader review and comparison of deterministic latency options can be found in [12]. A mathematical overview of the algorithms can be found in [7].

1) Urgency Based Scheduling: Urgency Based Scheduling [UBS] is the deterministic latency algorithm adopted for IEEE Time Sensitive Networking (TSN) Asynchronous Traffic Shaping (TSN-ATS) [11]. UBS relies on per-hop, per-flow processing. It is an improved mechanism compared to "Guaranteed Services" (GS) [17]. UBS relies on so-called interleaved-regulators compared to per-flow shaping and scheduling in GS, reducing the required queuing hardware cost/complexity.

Because gLBF uses the same latency calculus as UBS, and its explanation requires understanding it, we describe the UBS calculus here.

UBS traffic flows (i) are characterized by a burstiness $b(i)$ [e.g. bits] and a leak rate $r(i)$ [e.g. bits/sec]. These parameters define a constraint for the cumulative amount of data [e.g. bits] $w(d,i)$ over any period d called the leaky bucket constraint:

$$w(i,d) \leq b(i) + d * r(i) \tag{1}$$

To support different guaranteed latencies for different flows across the same interface, UBS uses a small number of strict priority queues $p = 1...N$ on an outgoing interface, for example $N = 8$. The buffers required for each queue p is $\sum_{i \in flows_p} b_i$, where $flows_p$ is the set of flows that should use priority queue p. The guaranteed maximum $latency(p)$ of a packet of $priority(p)$ through a forwarder hop is roughly (missing some fixed offset):

$$latency(p) = \sum_{j \in flows(q), q \leq p} b(j)/r_i \tag{2}$$

r_i is the serving rate of the interface, e.g. [bps]. Both N and the priority of a flow can be configured differently on every hop of the path. This allows the provision of finer grained end-to-end differences in latency across different flows than N, for example by assigning a flow to p and $p+1$ on different hops of its path.

Admission control in UBS needs to also ensure that the total amount of traffic does not exceed the link serving rate:

$$r_i \geq \sum_{j \in flows_i} flowrate_j \tag{3}$$

The per-hop calculations required for each packet of a flow to determine the earliest time it can be forwarded so the flow does not violate its leaky bucket constraint is the same as in a per-flow shaper. Instead of using a per-flow FIFO, calculating the head-of-fifo earliest departure time, and then performing appropriate scheduling across all FIFO queues with a head that is ready to be sent, UBS puts packets of all flows received from the same input interface k and destined for the same priority p into the same $FIFO(k,p)$. It then only calculates the earliest departure time for the head of each such $FIFO(k,p)$ based on the leaky bucket constraints of that flow and schedules packets across those $FIFOs(k,p)$ according to strict priority across p and between different k of the same p based on earliest departure time. These $FIFO(k,p)$ are called "Interleaved Regulators" because they interleave multiple flows and regulate the rate of their traffic. Interleaved regulators are the core novel implementation simplification that UBS contributes to the problem of deterministic latency based forwarding.

2) Cyclic Queuing and Forwarding: In IEEE TSN Cyclic Queuing and Forwarding [10] deterministic traffic packets are sent via a periodic time cycles. Received packets are assigned to a cycle purely by their receive time being within the cycle time. This requires strict time synchronization across forwarders and speed-of-light link propagation time must be known and packets only be sent so their last bit will arrive on the receiving forwarder before the cycle time ends there. The higher the link propagation latency, the lower the percentage of time during a cycle that packets can be sent. Typical CQF networks can therefore only span few kilometers in diameter.

3) Tagged Cyclic Queuing and Forwarding (TQCF) : [15] (updated by [16]) follows the model of CQF but attaches a cycle tag to the packet, therefore removing the size limitation of networks to support wide-area IP or MPLS networks.

Link propagation latency only has to be known but can be arbitrary large, and receivers can still appropriate map received packets into the desired cycle buffer. TCQF still requires clock synchronization, but its Mean Time Interval Error (MTIE) can be one or two orders of magnitude (factor 10..100) larger than required for CQF, therefore reducing the overall cost of clock synchronization significantly. In addition, by using a larger number of cycle buffers, such as 4, it is possible to also support links with variable propagation latency.

4) Latency Based Forwarding: Despite its similar name, Latency Based Forwarding (LBF) [3] started from a different set of goals than gLBF. LBF is based on the model that applications would indicate their Service Level Objective (SLO) for end-to-end latency in each data packet, specifying a desired [min..max] latency range. The network would then, without any form of upfront resource reservation, attempt to meet those demands against competing demands from other packets by matching QoS actions against an implied latency budget. (g)LBF is part of a larger effort to examine how better network packet header information can support better network services. This is also called NewIP.

The resulting forwarding algorithm tracks the actual latency of a packet hop-by-hop, predicting the remaining path latency through controller or routing protocols, and ultimately calculating on each hop a local queuing budget for the packet based on SLO, latency incurred, and remaining path information. The packet is then prioritized through the use of intelligent queuing, such that packets that have already been delayed on prior hops would be subjected to less queuing delay on this hop, all other requirements being equal. This approach opens up a range of new deployment options, such as hiding latency differences across different paths in a wide area network by forcing packets to be delayed on every hop by the minimum required latency. Integration of LBF with congestion control mechanisms from TCP and other transport layer protocols is future research work.

Because of the flexibility and complexity of the forwarding algorithms of LBF, there has so far been no calculus for guaranteeing latency for traffic of guaranteed bandwidth in conjunction with LBF forwarding. Upon examination of the UBS calculus and comparing the likely UBS forwarding plane implementations with the ones proposed for LBF, it became clear that a variation of LBF as presented in this paper can close this gap. This is the origin of the work presented in this paper.

III. Problem analysis

Existing bounded latency mechanism have issues in the face of constraints such as those of network equipment design, network deployment/operations and applications. Some of these issues resulted in the design and implementation of [15] and are described there. A more thorough problem description was contributed by the authors via [8]. gLBF builds on the experience with these prior mechanisms, combining their benefits and eliminating their remaining problems.

This section summarizes these problems and contributes the analysis for industrial control loop timing behavior to it.

Per-hop, per-flow state: GS and UBS require (interleaved) regulator (”shaper”) state for every flow on every hop where flows merge, which in typical topologies is every hop. While such state is common in e.g. lower speed (10 Gbps) in-building switches, it is uncommon and expensive to build at low-cost and for large number of flows in routers with 100 Gbps speed per interface or higher, which is the standard for wide area networks. When, as in GS/UBS, it is desired to not introduce any queuing latency in the absence of competing traffic, this so called ”regulator” state is unavoidable to compensate for uncontrollable latency on the prior hop that happens because of traffic from multiple interfaces colliding on the same outgoing interface.

Per-hop, per-flow state has to be signaled from a resource reservation system whenever a flow is added/deleted/changed. This creates highly undesirable churn/stress for intermediate hop routers for large scale networks with large number of flows. Network solutions such as ”Segment Routing” (SR-TE) [6] are today’s standard in wide-area networks specifically with the architectural goal not to have such per-flow, per-hop state and the need to signal it.

Packet scheduling is the biggest cost factor for hardware. GS requires scheduling across all flows (per-flow shaper). UBS requires scheduling only across O(#interfaces * #priorities) interleaved regulators. This makes UBS scheduling hardware less expensive than GS for switches with small #interfaces such as in industrial Ethernet switches where commonly #interfaces $\leq$ 24. For aggregation routers in wide area networks, #interface can be multiple hundreds, so the hardware improvement of UBS over GS is about a factor of 10 smaller.

Tightly bounded jitter: Path latency roughly is composed from queuing latency and non-queuing latency, such as the propagation latency of links (speed of light) and other, mostly fixed non-queuing propagation latencies in forwarders. The difference between the maximum and minimum latency is called jitter. Here we consider queuing jitter. In GS and UBS, queuing latency is zero in the absence of competing traffic. In the presence of the maximum permissible amount of competing traffic, queuing jitter is at maximum, resulting in the guaranteed (maximum) bounded latency. This means that queuing jitter in GS and UBS is maximum of any bounded latency solutions. In (T)CQF mechanisms, queuing jitter is to the largest extend independent of the amount of competing traffic, and queuing jitter is therefore tightly bounded.

Because the amount of competing traffic may change instantaneously, deterministic applications must be able to deal with packets arriving with maximum latency, even if all prior packets arrived with minimum latency. Many important type of applications can effectively only process data synchronously. In result, any packets arriving earlier than the maximum (guaranteed) bounded latency need to be buffered in the application before it can be processed. This includes many industrial control loop applications, but also most of media processing. Shorter latency in the absence of competing traffic is not a

benefit for these applications but an additional design cost that may extend (especially in low end IoT receiver devices) into the hardware design for buffers. Worse yet, the jitter in GS/UBS depends on the network size, such as number of router/switch hops and their maximum queue depths. Receiver devices therefore may need to be designed against specific assumptions about the type of network they can operate in. This is not required with (T)CQF mechanisms.

Clock synchronization: Network wide clock synchronization via IEEE1588 is required on forwarders using (T)CQF mechanisms. TSN-ATS was developed after CQF also to support bounded latency in networks, where such clock synchronization is an undesirable cost factor, both for its hardware requirements in switches, as well as operationally. Clock synchronization is common and acceptable in some networks, but it is uncommon in wide-area networks beyond mobile fronthaul network.

Not requiring clock synchronization for GS/TSN-ATS does not mean that it is not required on the hosts. Instead, applications may require clock synchronization, precisely because of how GS/TSN-ATS operate. For example, in industrial applications, Programmable Logic Controllers (PLC) do often operate on a local clock without external synchronization, periodically poll sensor data and send instructions to actors. Such periodic communication is synchronous to the accuracy of the network jitter. In TCQF, jitter across an arbitrary wide-area network path is always in the order of the cycle time, for example 10..100 usec. In GS/ATS across the same network, jitter can easily be multiple msec, or up to a factor of 100 worse than with tagged CQF.

When network jitter is too large for the application to operate in this fashion, the application itself may need clock synchronization of sensors and actors, and without other sources of clock synchronization, this would bring back clock synchronization into the network, even if it is not used by the deterministic mechanism of the network itself but only by hosts connected to it.

Problem summary: Per-hop, per-flow state of asynchronous deterministic latency solutions such as GS and UBS is highly undesirable in wide-area networks, solutions such as TCQF can provide bounded latency without this problem. TCQF still requires clock synchronization across the network. Eliminating the need for clock synchronization in the network such as required for (T)CQF is highly desirable, but for wide area networks it is not sufficient to do so at the cost of making jitter become order of magnitude larger and depending on network size. A deterministic latency mechanism not requiring per-hop, per-flow state without the need for clock synchronization and providing tightly bounded jitter would solve these problems. This is what gLBF provides.

IV. GLBF

A. gLBF concept

The root problem for deterministic latency guarantees is to avoid multi-hop packet clogging which happens when packets from multiple flows pass through the same interface(s). A packet P1 from one flow may end up behind a long interface queue, but the next packet P2 from the same flow, may arrive when the queue has already cleared and pass through the interface without any queuing delay. On the next hop, P1 and P2 are clogged together, and across 2 or more hops this ”(micro)burst-accumulation” makes calculation of latency guarantees a mathematically exponential problem.

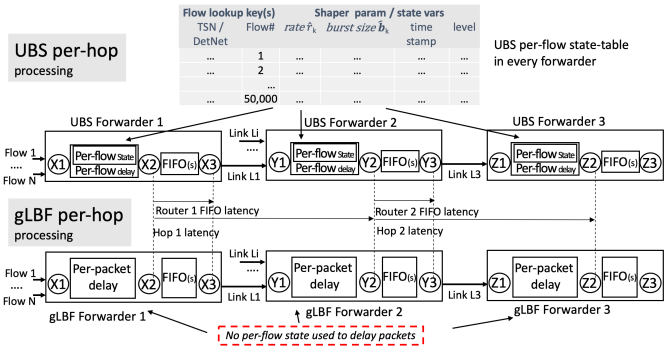

Fig. 2. UBS and gLBF per-hop processing

In gLBF, each hop measures the packets queuing latency through its interface, calculates the difference between this queuing latency and the guaranteed latency for the hop, signals this remaining latency value in the packet to the next hop, and that next hop then delays the packet by this value. This creates virtually zero-jitter guaranteed latency forwarding (quasi-synchronous) and operates without per-flow state and without network wide clock synchronization - both are not required because of the in-packet field for the remaining per-hop latency. gLBF also allows to use the same latency calculus as UBS and provides therefore the same guaranteed latency.

In comparison, GS and UBS do require per-hop, per-flow state because they do measure each flows burstiness on every hop and delay a packet when it would otherwise be processed too early (clogged) with respect to the prior packets of the same flow. This also introduces worst-case jitter: zero in the absence of competing traffic, and maximum in the presence of maximum competing traffic.

B. gLBF Mechanism

Using the names from the gLBF part of Fig. 2, gLBF operates as follows.

We assume that at X2, all packets comply with their flows leaky bucket condition because of prior operations. The maximum delay of a packet through Forwarder 2 FIFO is MAX_FIFO. This can be calculated from UBS calculus or simply as the latency of a bit through the full FIFO. MAX_LINK is the maximum serialization time of the largest permissible packet through link L1.

At X3, Forwarder 1 measures the packets delay through its FIFO as $FIFO\_latency$, $(<= \ MAX\_FIFO)$. It also calculates $packet\_serialization\_delay$ from the packet size and link L1 speed. It finally calculates $glbf\_delay \ = \ MAX\_FIFO \ + \ MAX\_LINK \ -$

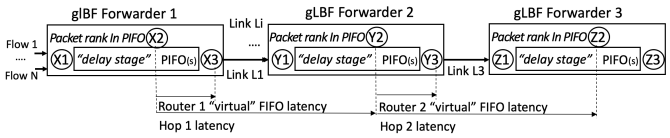

Fig. 3. gLBF with PIFO(s)

$FIFO\_latency - packet\_serialization\_delay$. This value is signaled as a new packet header field in the packet to Forwarder 2. Forwarder 2 delays the packet by this value before passing it to point Y2.

The result of this gLBF mechanism is that all packets will arrive at Y2 at a time $Hop1\_Latency = MAX\_FIFO + MAX\_LINK$ later than their arrival time at X2. Therefore they will all comply with their leaky bucket condition at Y2 if they did so at X2. This condition likewise applies to any further gLBF hops. When a forwarder receives packets from an untrusted sender and/or without the glbf_delay parameter, it needs to perform per-flow shaping so packets conform to their leaky bucket conditions before entering the gLBF network.

With a constant delay for every gLBF hop, all packets are forwarded with their maximum bounded latency as calculated by e.g. UBS calculus. Unlike UBS, gLBF will also ensure that packets are delivered with very tightly bounded jitter, which is determined solely by the latency through the last hop Forwarders FIFO, whereas the jitter incurred on any other hop across the gLBF network is compensated for by the gLBF delay operations. If the actual receiver application also understands gLBF, it can also implement the gLBF delay stage and effectively process packets with zero path jitter after that stage.

C. ASIC based gLBF

Sequential delay and FIFO stages will incur performance issues or additional delay in real-world ASIC implementations. The reason for this is that in the worst case, all packets may need to be passed at exactly the same time across X2, for example because each packet came from a different original sender and was sent at the same time. Both stages can be merged into a single Push-In-First-Out (PIFO) queue, in which the rank of each packet is its desired time of release from the delay stage (e.g. Y2 for Forwarder 2). This is depicted in Figure 3.

When the packet arrives at time $T1$ at Forwarder 2, Y1, it calculates $rank = T1 + glbf\_delay$ and enqueues the packet into the PIFO. When the packet is dequeued at Y3, time $T3$, it calculates $glbf\_delay = MAX\_FIFO + MAX\_LINK - (T3 - rank) - packet\_serialization\_delay$. $(T3 - rank)$ is the time that the packet had to wait longer in the PIFO than its desired $gLBF\_delay$, aka: the equivalent of $FIFO\_latency$.

D. Multi-priority gLBF

gLBF can be expanded to support multiple priorities and therefore latencies in the same fashion as UBS does by using a separate PIFO for each priority and dequeueing packets from the highest priority PIFO whose head of queue packet is ready, e.g. $(rank < current\_time)$. Each PIFOs has its own MAX_FIFO parameter and may have a different MAX_LINK parameter for purposes of gLBF calculations.

For per-flow stateless operations with multiple priorities, each gLBF hop needs to learn the packets priority from a packet header. This can be a single priority parameter for all hops, or a sequence of priority values, one per-hop, to allow different priorities across different hops, as UBS does as well.

In Segment Routing for IPv6 (SRv6) network technology, a sequence of (128 bit) IPv6 addresses is used to steer packets through the network. Therefore a similar type of header, requiring for example 4 bit per hop for up to 16 priorities per hop, would be a relatively insignificant impact to overall packet header size.

E. FIFO based gLBF

100 Gbps or faster PIFO are subject of research for high-speed ASIC/FPGA, but not available in shipping products. We describe how to utilize single stage FIFO, using the design and calculus experience from UBS single stage FIFO queuing.

In UBS, every (IIF,P) combination of Incoming InterFace (IIF) and Priority (P) has its own interleaved regulator (IR), which is a FIFO where packets are dequeued using per-flow (regulator) state on the head of queue packet. Only packets for the same (IIF,P) can share the same IR because only their arrival order is also their departure order. Likewise, in gLBF, each PIFO for a particular priority P can be replaced by a set of (IIF,P) FIFO combined with the logic to dequeue packets based on the earliest rank of the head of those FIFOs.

(IIF,P) FIFOs cannot support different priorities for a packet on different hops, because the FIFO into which a packet needs to get enqueued is the priority of the packet on the receive-side hop of the forwarder (up to point 2), but the dequeuing of the packet is based on the packets priority of the send-side hop of the forwarder (up to 3). Only when those two priorities of a packet are the same can it stay in the same single-stage FIFO, or else dequeue order is not the same as enqueue order. A single packet header priority parameter can support this model. Single priority deployments require no priority header field at all and just a FIFO per IIF to avoid PIFO implementation.

This per-path priority limitation can be resolved by a combination of (IIF,Input-Prio,Output-Prio) FIFOs and according dequeuing logic, but this would result in a logic more complex than UBS, and is therefore just one out of likely multiple options for implementing full PIFO queuing logic in longer-term hardware designs.

V. Validation

To validate gLBF, we developed a simple, purpose-built time discrete simulation tool in Perl [21] to understand and diagnose gLBF behavior with as little as possible software code to understand and. The goal was to understand the gLBF algorithm in operation, find cases where burst accumulation across a single hop cause latency to exceed the calculated bounds (in the absence of gLBF) and then to add gLBF and

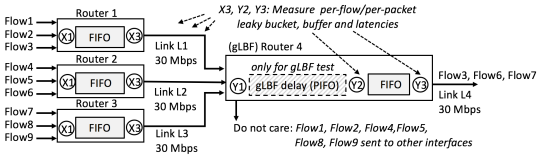

Fig. 4. Validation test setup

validate that it would not only fix that issue, but also achieve the synchronous latency across that single hop.

This tool operates at simulated 1 nsec times to allow bit-accurate measurements for link speeds up to 1 Gbps. For reasons of space in this paper, we can only summarize validation of the most conclusive test scenario we tested. In a first set of test runs, we validated the existence of the FIFO (micro)burst-accumulation problem (to verify the validation tool and problem statement). In a second set of test runs, we validate and quantify that gLBF solves these problems through the introduction of the gLBF latency step. We use example parameters that allow for virtual time test runs of 1 second duration.

Router 1, Router 2, and Router 3 are set up to each receive three rate-controlled traffic flows, each with a bitrate of 10Mbps, burst sizes of 3 packets, and slightly different packet sizes per flow: Router 1 = (900, 1000, 1100) bytes, Router 2 = (930, 1030, 113) bytes and Router 3 = (1370, 1170, 970) bytes. These values are chosen to maximize the problem of burst accumulation in the absence of gLBF. All nine flows arrive on Router 4, but only Flow 3 (from Router 1), Flow 6 (from Router 2) and Flow 7 (from Router 3) are forwarded across Router 4 FIFO to L4. The remaining flows are assumed to be routed by Router 4 to other interfaces that we do not care about. We use three routers (1, 2, 3) to feed Router 4 because if the output interface from Router 4 was fed by just a single interface (of the same speed), there would be no further burst accumulation on Router 4 in the absence of gLBF.

Upon origination (into Router 1,2,3), all three flows stay within their leaky bucket constraint:

$$w(i,d) \leq b(i) + d * r(i) \tag{4}$$

where $b(i)$ is the burst size of flow i, $r(i)$ is the rate of flow i (10 Mbps), d is a period of time, and $w(i,d)$ is the maximum permitted total amount of data of flow i across period d.

The validation tool measures compliance of a flows packets with the flows leaky bucket condition of [UBS] applied to the timestamp every packet of the flow is observed with:

```
// Initialization
for(i:I) { state[i].timestamp := 0;
           state[i].level := b(i); }

receive(p, i, t) { // packet p of flow i arrives at t
  state[i].timestamp = dt = t - state[i].timestamp;
  state[i].level += dt * state[i].rate;
  if(state[i].level > if(state[i].bsize)
    state[i].level = state[i].bsize;
  state[i].level -= 8 * p.length;
  if(state[i].level < 0)
    state[i].levelerrors++;
}
```

When a flow exceeds its leaky bucket condition, $level$ becomes negative. Packets of the flow during this period use more buffer space than allocated in a real system and they would be discarded.

In our first set of validation runs without gLBF, we observe level errors for the flows arriving from Router 2/3, and note that their latency's through Router 2/3 FIFO stay within the bounds of their calculated buffer / burst size limits. However, this changes when flows 3,6,7 go through the Router 4 FIFO. The maximum calculated buffer size for the admitted flows for the Router4 FIFO is 3 * (1100 + 1130 + 970) = 9600 bytes and the maximum guaranteed latency through Router4 FIFO is therefore 2.56 msec (9600 bytes at 30 Mbps). Because of burst accumulation, the validation shows a maximum buffer utilization of 11540 bytes or 1940 bytes above the target guaranteed maximum and therefore a maximum latency of 2.82 msec. This is the validation proof for the problem to be solved by deterministic forwarding solutions.

Our second set of validation runs observes how the same setup behaves with gLBF activated in Routers 1/2/3/4. For Routers 1/2/3 this means we do activate the calculation of $glbf\_delay$ at X3 and for Router 4 we active the gLBF delay component reacting on it, so ultimately, we do want to see that the Hop 1 buffer utilization and latency stays within the target bounds.

In result, in Y1, we observe FIFO hop latencies of 315445 nsec to 2808003 nsec and 1224 packets with leaky bucket violations, lowest level of -15232 bits. After gLBF delay on Router 4, observing packets at Y2, no packet has a leaky bucket condition violation, and the FIFO gLBF latencies are 2693334 nsec for all packets from Router 1, 2693334 nsec for all packets from Router 2 and 3101334 nsec for all packets from Router 3. The constant latency across all packets from a particular router proves the zero jitter that gLBF achieves, compensating for the jitter that was seen in the test runs before gLBF was enabled. The absolute delays are different across the Router 1, 2, 3 hops because the FIFO buffer size in Routers 1, 2, 3 are different due to the flows' divergent packet sizes. The latencies measured match the latencies expected by the UBS calculus.

Finally, we observe the FIFO latency on Router 4 after gLBF delay to show that the maximum FIFO buffer utilization is again within the limits of the worst-case buffer utilization under worst-case leaky-bucket conformant flows, as opposed to the earlier observation of overrunning these buffers with just FIFO forwarding. The maximum buffer utilization observed is 8630 bytes (out of the theoretical maximum of 9600 bytes for the three flows burst sizes), and a maximum FIFO latency of 2.25 msec, so all packets stay within the theoretical limits for the buffer size. In comparison, with gLBF delay removed, the validation shows 6 packets exceed the buffer limit with a maximum buffer utilization of 11540 bytes (20% beyond 9600 bytes) and a maximum FIFO latency of 2.82 msec (10% beyond the theoretical limit of 2.56 msec).

Note that our tests do not account for the speed of light propagation latency of Link 1, because it does not impact the gLBF algorithm. If there is significant propagation latency, as there would be in any target wide area network deployment, then the physically observed latencies between the X and Y measurement points would simply be larger by that propagation latency than the latencies validated here.

Even though only anecdotal through simple worst-case flow collision experiments, the validation shows what the authors think to be theoretically easy to grasp:

gLBF delay of packets reconstitutes packet flows to their prior-hop inter-packet spacing. If on that prior hop, packets of flows stay within their leaky bucket condition, then they too will do so after gLBF delay on the following hop. Therefore the non-exponential calculation of per-hop FIFO latency as done in rate-controlled service disciplines such as UBS can be reused in gLBF, but without the systematic challenges introduced by per-flow shaper/interleaved regulators or cyclic queuing as in prior mechanisms.

gLBF achieves per-hop zero-jitter forwarding relative to the nodes local clock, whereas (T)CQF would introduce jitter in the order of a cycle (e.g. 100usec), and UBS would introduce per-hop jitter in the order of the per-hop bounded latency.

VI. Conclusions and outlook

gLBF is a new mechanism that enables predictable, virtually synchronous per-hop-latency, enabling high-precision communication services that provide end-to-end latency guarantees as required by many time-sensitive networking applications. gLBF combines the time-synchronization free multi-priority bounded benefits of UBS with the low-jitter and per-flow-stateless operation of (T)CQF, allowing low-cost implementation and low operational complexity.

The principle of gLBF can be used to compensate for the variability of not only the queuing part of forwarding and the (packet size dependent) link serialization latency, but for any variable latency that can be predicted or measured. For example, in complex routers latency through ingress line card or fabric may be variable and/or line cards may not have a common time basis and appropriate adoption of gLBF across these components can be used to create virtually synchronous latency across such complex routers.

The exact behavior of gLBF under differences in clock speed across consecutive hops deserves further examination, similar to the errors introduced into clock differences in pre-existing asynchronous systems (that are equally not discussed by initial publications of those mechanisms). The authors believe that differences in clock speed below some N percent across hops will require worst case the need to overestimate per-hop latency and buffer utilization by N percent and to limit link utilization to (100-N)%, but experimental or theoretical validation of this is subject to future work.

References

[1] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. Rsvp-te: Extensions to rsvp for lsp tunnels. RFC 3209, RFC Editor, December 2001.

[2] Bob Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. RFC 2205, RFC Editor, September 1997. URL: http://www.rfc-editor.org/rfc/rfc2205.txt.

[3] Alexander Clemm and Toerless Eckert. High-Precision Latency Forwarding over Packet-Programmable Networks. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, April 2020.

[4] A. Farrel, Q. Zhao, Z. Li, and C. Zhou. An architecture for use of pce and the pce communication protocol (pcep) in a network with central control. RFC 8283, RFC Editor, December 2017.

[5] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer. Ipv6 segment routing header (srh). RFC 8754, RFC Editor, March 2020.

[6] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. Segment routing architecture. RFC 8402, RFC Editor, July 2018.

[7] Norm Finn, Jean-Yves Le Boudec, Ehsan Mohammadpour, Jiayi Zhang, Balazs Varga, and Janos Farkas. *draft-ietf-detnet-bounded-latency*, May 2021.

[8] Toerless Eckert and Stewart Bryant. *draft-eckert-detnet-bounded-latency-problems*, July 2021.

[9] E. Grossman. Deterministic networking use cases. RFC 8578, RFC Editor, May 2019.

[10] 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*, 2017.

[11] IEEE Time-Sensitive Networking (TSN) Task Group. *IEEE Std 802.1Qcr-2020: IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks — Amendment: Asynchronous Traffic Shaping.*, 2020.

[12] Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury. *TSN Algorithms for Large Scale Networks: A Survey and Conceptual Comparison* , 2019.

[13] Kathleen Nichols, Steven Blake, Fred Baker, and David L. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, RFC Editor, December 1998. http://www.rfc-editor.org/rfc/rfc2474.txt.

[14] Jon Postel. Internet protocol. STD 5, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt.

[15] Li Qiang, Xuesong Geng, Bingyang Liu, Toerless Eckert, Liang Geng, and Guangpeng Li. *draft-qiang-detnet-large-scale-detnet: Large-Scale Deterministic IP Network*, September 2019.

[16] Joanna Dang and Bingyang Liu. *draft-dang-queuing-with-multiple-cyclic-buffers: A Queuing Mechanism with Multiple Cyclic Buffers*, February 2021.

[17] Scott Shenker, Craig Partridge, and Roch Guerin. Specification of guaranteed quality of service. RFC 2212, RFC Editor, September 1997. URL: http://www.rfc-editor.org/rfc/rfc2212.txt, https://doi.org/10.17487/RFC2212.

[18] Johannes Specht and Soheil Samii. Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2016.

[19] JP. Vasseur and JL. Le Roux. Path computation element (pce) communication protocol (pcep). RFC 5440, RFC Editor, March 2009. http://www.rfc-editor.org/rfc/rfc5440.txt.

[20] John Wroclawski. The use of rsvp with ietf integrated services. RFC 2210, RFC Editor, September 1997. http://www.rfc-editor.org/rfc/rfc2210.txt.

[21] Toerless Eckert. Simple GLBF Validation tool, Jan 2021. https://github.com/network2030/glbf-validation/