DetNet Working Group
Internet-Draft
Intended status: Informational
Expires: 17 September 2025

J. Joung
Sangmyung University
J. Ryoo
T. Cheung
ETRI
Y. Li
Huawei
P. Liu
China Mobile

16 March 2025

Asynchronous Deterministic Networking Framework for Large-Scale Networks
draft-joung-detnet-asynch-detnet-framework-06

Abstract

This document describes various solutions of Asynchronous Deterministic Networking (ADN) for large-scale networks. The solutions in this document do not need strict time-synchronization of network nodes, while guaranteeing end-to-end latency or jitter. The functional architecture and requirements for such solutions are specified.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on 17 September 2025.

Copyright Notice

Table of Contents

1. Introduction

Deterministic Networking (DetNet) provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low data loss rates and bounded latency within a network domain. The architecture of DetNet is defined in RFC 8655 [RFC8655], and the overall framework for DetNet data plane is provided in RFC 8938 [RFC8938]. Various documents on DetNet IP (Internet Protocol) and MPLS (Multi-Protocol Label Switching) data planes and their interworking with Time-Sensitive Networking (TSN) have been standardized. Technical elements necessary to extend DetNet to a large-scale network spanning multiple administrative domains are identified in [I-D.liu-detnet-large-scale-requirements].

This document considers the problem of guaranteeing both latency upper bounds and jitter upper bounds in large-scale networks with any type of topology, with random dynamic input traffic. The jitter is defined as the latency difference between two packets within a flow, not a difference from a clock signal or from an average latency, as is summarized in RFC 3393 [RFC3393].

In large-scale networks, end-nodes join and leave, and a large number of flows are dynamically generated and terminated. Achieving satisfactory deterministic performance in such environments would be challenging. The current Internet, which has adopted the DiffServ architecture, has the problem of the burst accumulation and the cyclic dependency, which is mainly due to FIFO queuing and strict priority scheduling. Cyclic dependency is defined as a situation wherein the graph of interference between flow paths has cycles [THOMAS]. The existence of such cyclic dependencies makes the proof of determinism a much more challenging issue and can lead to system instability, that is, unbounded delays [ANDREWS][BOUILLARD]. The Internet architecture does not have an explicit solution for the jitter bound as well. Solving the problem of latency and jitter as a joint optimization problem would be even more difficult.

The basic philosophy behind the framework proposed in this document is to minimize the latency bounds first by taking advantage of the work conserving schedulers with regulators or stateless fair queuing schedulers, and then minimize the jitter bounds by adjusting the packet inter-departure times to reproduce the inter-arrival times, at the boundary of a network. We argue that this is simpler than trying to minimize the latency and the jitter at the same time. The direct benefit of such simplicity is its scalability.

For the first problem of guaranteeing latency bound alone, IEEE asynchronous traffic shaping (ATS) [IEEE802.1Qcr], the flow-aggregate interleaved regulators (FAIR) [FAIR][Y.3113] frameworks, the port-based flow aggregate regulators (PFAR) [ADN], and the work conserving stateless core fair queuing (C-SCORE) are proposed as solutions. The key component of the ATS and the FAIR frameworks is the interleaved regulator (IR)), which is described in [RFC9320]. IR has a single queue for all flows of the same class from the same input port. Head of Queue (HOQ) is examined if the packet is eligible to exit the regulator. To decide whether it is eligible, IR is required to maintain the individual flow states. The key component of the PFAR framework is the regulators for flow aggregates (FA) per port per class, which regulates the FA based on the sum of average rates and the sum of maximum bursts of the flows that belong to the FA. The key component of C-SCORE is the packet state that is carried as metadata. C-SCORE does not need to maintain flow states at core nodes, yet it works as one of the fair queuing schedulers, which is known to provide the best flow isolation performance. The metadata to be carried in the packet header is simple and can be updated during the stay in the queue or before joining the queue.

For the second problem of guaranteeing jitter bound, it is necessary to assume that the first problem is solved, that is, the network guarantees latency bounds. Furthermore, the network is required to specify the value of the latency bound for a flow. The end systems at the network boundary, or at the source and destination nodes, then can adjust the inter-departure times of packets, such that they are similar to their inter-arrival times. In order to identify the inter-arrival times at the destination node, or at the network edge near the destination, the packets are required to specify their arrival times, according to the clock at the source, or the network edge near the source. The clocks are not required to be time-synchronized with any other clocks in a network. In order to avoid a possible error due to a clock drift between a source and a destination, they are recommended to be frequency-synchronized.

In this document, strict time-synchronization among network nodes is avoided. It is not easily achievable, especially over a large area network or across multiple DetNet domains. Asynchronous solutions described in this document can provide satisfactory latency bounds with careful design without complex pre-computation, configuration, and hardware support usually necessary for time synchronization.

2. Terminology

2.1. Terms Used in This Document

2.2. Abbreviations

3. Conventions Used in This Document

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

4. Framework for Latency Guarantee

4.1. Problem Statement

In Section 4, we assume there are only two classes of traffic. The high priority traffic requires latency upper bound guarantee. All the other traffic is considered to be the low priority traffic, and be completely preempted by the high priority traffic. High priority (HP) traffic is our only focus.

It is well understood that the necessary conditions for a flow to have a bounded latency inside a network, are that;

* a flow entering a network conforms to a prescribed traffic specification (TSpec), including the arrival rate and the maximum burst size, and

* all the network nodes serve the flow with a service rate which are greater than or equal to the arrival rate.

These conditions make the resource reservation and the admission control mandatory. These two functions are considered given and out of scope of this document.

Here, the notion of arrival and service rates represent sustainable or average values. A short-term discrepancy between these two rates contributes to the burst size increment, which can be accumulated as the flow passes through the downstream nodes. This results in an increase in the latency bound. Therefore, the value of accumulated burst size is a critical performance metric.

The queuing and scheduling of a flow plays a key role in deciding the accumulated burst size. Ideally, the flows can be queued in separate queues and the queues are scheduled according to the flow rates. In this case a flow can be considered isolated. With practical fair schedulers, such as Deficit Round Robin (DRR), an isolated flow still can be affected by the other flows as much as their maximum packet lengths.

If we adopt a separate queue per flow at an output port, and assume identical flows from all the input ports, then the maximum burst size of a flow out of the port, Bout, is given as the following:

$$Bout < Bin + (n-1)L*r/C,$$

where Bout is the outgoing flow's maximum burst size, Bin is the incoming flow's maximum burst size, n is the number of the flows, L is the maximum packet size, r is the average rate of the flow, and C is the link capacity. This approach was taken in the integrated services (IntServ) framework [RFC2212].

The separate queues in the aforementioned case can be too many to be handled in real time, especially at the core of large-scale networks. The common practice therefore is to put all the HP flows in a single queue, and serve them with higher priority than best effort traffic. It is also well known that, with a resource reservation, a proper scheduling scheme such as the strict priority (SP) scheduling can guarantee service rates larger than the arrival rates, therefore the latency can still be guaranteed. With such a single aggregate queue the flows are not considered isolated, however. In this case a flow's burst size in a node can be increased proportionally to the sum of maximum burst sizes of the other flows in the queue. That is,

$$Bout < Bin + (n-1)Bin*r/C.$$

The second term on the right-hand side represents the amount of increased maximum burst. It is dominated by the term (n-1)Bin, which is the maximum total burst from the other flows.

Moreover, this increased burst affects the other flows' burst size at the next node, and this feedforward can continue indefinitely where a cycle is formed in a network. This phenomenon is called a cyclic dependency of a network. It is argued that the burst accumulation can explode into infinity, therefore the latency is no longer guaranteed.

As such, a flow is required to be isolated to a certain level, from the other flows' bursts, such that its burst accumulations are kept within a necessary value. By doing so, the other flows are also

isolated. The regulators or the fair queuing schedulers are described as solutions in this document for such isolation. They can decrease the accumulated burst into a desirable level and can isolate flows from others. In case of the regulators, however, if the regulation needs a separate queue per flow, then the scalability would be harmed just like the ideal IntServ case. In this document solutions with the IR or the regulations on flow aggregates are described.

Meanwhile, Fair Queuing (FQ) limits interference between flows to the degree of maximum packet size. Packetized generalized processor sharing (PGPS) and Weighted Fair Queuing (WFQ) are representative examples of this technique [PAREKH]. In this technique, the key is to record the service status of the flow in real time to provide exactly the assigned amount of service. The worst delay at each node with FQ is proportional to the maximum packet length divided by service rate. However, for this purpose, the complexity of managing and recording a large amount of state information for each flow is a problem, so it is not usually applied in practice.

To solve this problem, the flow entrance node can create state information for each packet, including the finish time (FT) and other necessary information, and record it in the packet. Subsequent nodes infer the exact states of the packet at the node based on these records without managing flow state information. This method provides a scalable solution that enables isolation between flows by modifying the FT based on local and initial information as needed. In this document the method of such stateless FQ implementation in core nodes is described.

4.2. Asynchronous Traffic Shaping (ATS)

The first solution in this document for latency guarantee is IEEE TSN TG's ATS technology. Essentially it is a combined effort of the flow aggregation per node per input/output ports pair per class, and the interleaved regulator per flow aggregate (FA). The IR examines the HOQ, identifies the flow the packet belongs to, and transfers the packet only when it is eligible according to the TSpec including the maximum burst size and arrival rate of the flow. Having the flows regulated as their TSpecs, the flow's burst size in a node can be increased proportionally to the sum of initial maximum burst sizes of the other flows in the queue, denoted by B.

$$Bout < B + (n-1)B*r/C.$$

The initial maximum burst size refers to the maximum burst size specified in TSpec.

This solution can have only one queue per FA, but suffers from having to maintain each individual flow state. The detailed description on ATS can be found in [IEEE802.1Qcr].

4.3. Flow Aggregate Interleaved Regulators (FAIR)

4.3.1. Overview of FAIR

In the FAIR framework, a network can be divided into several aggregation domains (ADs). HP flows of a same path within an AD are aggregated into an FA. IRs per FA are implemented at the boundaries of the ADs. An AD can consist of an arbitrary number of nodes. An FA can be further subdivided based on the flow requirements and characteristics. For example, only video flows of the same path are aggregated into a single FA.

Figure 1 shows an example architecture of the FAIR framework. IRs at AD boundaries suppress the burst accumulations across the ADs with the latency upper bounds intact as they do in IEEE TSN ATS, if the incoming flows are all properly regulated, and the AD guarantees the FIFO property to all the packets in the FA [LEBOUDEC]. It is sufficient to put every FA into a single FIFO queue in a node, in order to maintain the FIFO property within an AD. However, in this case, if cycles are formed, the burst accumulations inside an AD can be accumulated indefinitely. If the topology does not include a cycle and the latency bound requirement is not stringent, then FIFO queues and SP schedulers would be allowable. Otherwise, FAs are recommended to be treated with separated queues and fair-queuing schedulers for flow isolation.

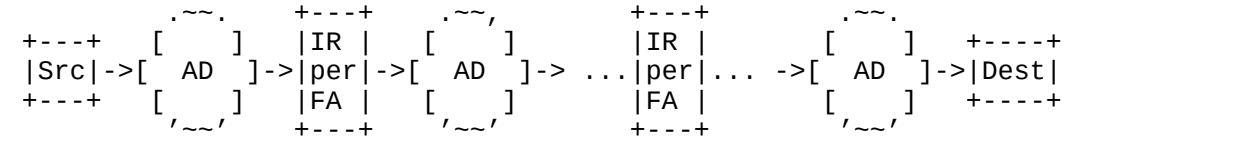

Figure 1: FAIR Framework

4.3.2. The performance of FAIR

FAIR guarantees an end-to-end delay bound with reduced complexity compared to the traditional flow-based approach. Numerical analysis shows that, with a careful selection of AD size, FAIR with DRR schedulers yields smaller latency bounds than both IntServ and ATS [FAIR].

ATS can be considered as a special case of FAIR with FIFO schedulers, where all the ADs encompass only a single hop. IntServ can also be considered as an extreme case of FAIR with fair schedulers and queues per FA, with an AD corresponding to an entire network; therefore, regulators are unnecessary.

4.4. Port-based Flow Aggregate Regulators (PFAR)

IR in ATS and FAIR suffers from two major complex tasks; the flow state maintenance and the HOQ lookup to determine the flow to which the packet belongs. Both tasks involve real-time packet processing and queue management. As the number of flows increases, the IR operation may become burdensome as much as the per- flow regulators. Without maintaining individual flow states, however, the flows can be isolated to a certain level, as is described in this section.

Let us call the set of flows sharing the same input/output ports pair the port-based FA (PFA). The only aggregation criteria for a PFA are the ports and the class. The port-based flow aggregate regulators (PFAR) framework puts a regulator for each PFA in an output port module, just before the class-based queuing/scheduling system of the output port module. The PFAR framework sees a PFA as a single flow with the "PFA-Tspec", {the sum of the initial maximum bursts; and the sum of the initial arrival rates} of the flows that are the elements of the PFA; and regulates the PFA to meet its PFA-Tspec.

If we assume identical flows in a network with ideal symmetrical topology, then the maximum burst size of an arbitrary set of flows within an PFA, Bout, is given as the following:

$$Bout < Bin + (p-1)B*r/C,$$

where Bin is the sum of maximum burst sizes of the flows within the FAin, B is the sum of initial maximum burst sizes of the flows within the FAin, and p is the number of the ports in the node.

PFARs can be placed at the output port of a node before the output SP scheduler. The architecture is similar to that suggested in IEEE ATS, except that in ATS, IRs are placed instead of PFARs.

Note that Bout is affected mostly by B; in other words, the burst size out of a node is affected mostly by the initial maximum burst sizes of the other PFAs from different input ports of the node. This property makes the Bout does not increase exponentially even in the existence of cyclic dependencies. The regulator in PFAR increases the worst latency, as much as (Bin - B)/r, while IR does not.

With PFAR, the HOQ flow identification process is unnecessary, and only the PFAs' states, instead of individual flows' states, must be maintained at a node. In this respect, the complexity of PFAR is reduced compared to that of IR in ATS or FAIR.

In a recent study [ADN], it was also shown, through a numerical analysis with symmetrical networks with cycles, that PFAR, when implemented at every node, can achieve comparable latency bounds to IEEE ATS technique.

The ATS, FAIR, and PFAR frameworks maintain regulators per FA. FAs in these frameworks are composed of the flows sharing the same ingress/egress ports of an AD. ADs can encompass a single hop or multiple hops. The regulators can be IRs or the aggregate regulators. There can be other combinations of AD and regulator type, which could be further investigated and compared to the frameworks introduced in this document.

4.5. Work conserving stateless core fair queuing (C-SCORE)

4.5.1. Framework

PGPS [PAREKH], WFQ, Virtual Clock (VC), and similar other schedulers utilize the concept of Finish Time (FT) that is the service order assigned to a packet. The packet with the minimum FT in a buffer is served first. We will call these works collectively as Fair Queuing (FQ).

As an example of FQ, the VC scheduler [ZHANG] defines FT to be

$$F(p) = \max\{F(p-1), A(p)\} + L(p)/r, \qquad (1)$$

where (p-1) and p are consecutive packets of the flow under observation, A(p)is the arrival time of p, L(p) is the length of p, and r is the flow service rate. The flow index is omitted.

The key idea of FQ is to calculate the service finish times of packets in an imaginary ideal fluid service model and use them as the service order in the real packet-based scheduler.

While having the excellent flow isolation property, FQ needs to maintain the flow state, F(p-1). For every arriving packet, the flow it belongs to has to be identified and its previous packet's FT should be extracted. As the packet departs, the flow state, F(p), has to be updated as well.

We consider a framework for constructing FTs for packets at core nodes without flow states. In a core node, the following conditions on FTs SHOULD be met.

C1) The 'fair distance' of consecutive packets of a flow generated at the entrance node has to be kept in the core nodes. That is; Fh(p) >= Fh(p-1) + L(p)/r, where Fh(p) is the F(p) at core node h.

C2) The order of FTs and the actual service order, within a flow, have to be kept. That is; Fh(p) > Fh(p-1) and Ch(p) > Ch(p-1), where Ch(p) is the actual service completion time of packet p at node h.

C3) The time lapse at each hop has to be reflected. That is; Fh(p) >= F(h-1)(p), where F(h-1)(p) is the FT of p at the node h-1, the upstream node of h.

In essence, (1) has to be approximated in core nodes. There can be many possible solutions to meet these conditions. We describe a generic framework with requirements for constructing FTs in core nodes that meet the conditions, without flow state, in the following.

Definition: An active period for a flow is a maximal interval of time during a node busy period, over which the FT of the most recently arrived packet of the flow is greater than the virtual time (equivalently the system potential). Any other period is an inactive period for the flow.

Requirement 1: In an entrance node, it is REQUIRED to obtain FTs with the following equation. 0 denotes the entrance node of the flow under observation.

$$F0(p) = \max\{F0(p-1), A0(p)\}+L(p)/r.$$

Note that if FTs are constructed according to the above equation, the fair distance of consecutive packets is maintained.

Requirement 2: In a core node h, it is REQUIRED to increase FT of a packet by an amount, d(h-1)(p), that depends on the previous node and the packet.

$$Fh(p) = F(h-1)(p) + d(h-1)(p).$$

Requirement 3: It is REQUIRED that dh(p) is a non-decreasing function of p, within a flow active period.

Requirements 1, 2, and 3 specify how to construct FT in a network. By these requirements Conditions C1), C2), and C3) are met. The following requirements 4 and 5 specify how FT is used for scheduling.

Requirement 4: It is REQUIRED that a node provides service whenever there is a packet.

Requirement 5: It is REQUIRED that all packets waiting for service in a node are served in the ascending order of their FTs.

We call this framework the work conserving stateless core fair queuing (C-SCORE), which can be compared to the existing non-work conserving scheme [STOICA].

4.5.2. Selection of delay factor for latency guarantee

For C-SCORE to guarantee E2E latency bound, the dh(p) is RECOMMENDED to be defined as in the following.

$$dh(p) = SLh. \quad (2)$$

The service latency of the flow at node h, denoted by SLh, is given as follows.

$$SLh = Lh/Rh + L/r, \quad (3)$$

where Lh is the max packet length in the node h over all the flows that are transmitted from the output port under observation, Rh is the link capacity of the node h, and L is the max packet length in the flow. The service latency was first introduced in the concept of Latency-rate server model [STILIADIS-LRS], which can be interpreted as the worst delay from the arrival of the first packet of a new flow until its service completion.

Consider the worst case: Right before a new flow’s first packet arrives at a node, the transmission of another packet with length Lh has just started. This packet takes the transmission delay of Lh/Rh. After the transmission of the packet with Lh, the flow under observation could take only the allocated share of the link, and the service of the packet under observation would be completed after L/r. Therefore, the packet has to wait, in the worst case, Lh/Rh + L/r.

The reason to add the service latency to F(h-1)(p) to get Fh(p) is to meet Condition C3) in a most conservative way without being too excessive. Intuitively, when every packet’s FT is updated with the flow’s own worst delay, then a packet that experienced the worst delay gets a favor. Thus its worst delay will not get any worse, while the delay differences among flows are reflected.

When dh(p) is decided by (2), then it can be proved that

$$Dh(p) <= (B-L)/r + SL0 + SL1 + ... + SLh, \quad (4)$$

where Dh(p) is the latency experienced by p from the arrival at the node 0 to the departure from node h; B, L, and r are the max burst of, max packet length of, and allocated rate to the flow under observation that p belongs to, respectively [KAUR].

Note that the latency bound in (4) is the same to the network where every node has a stateful FQ scheduler, including VC. The parameters in the latency bound are all intrinsic to the flow, except Lh/Rh.

On the other hand, dh(p) may not be a function of p, and dependent only on the nodes. Then it could be denoted as dh. For example, dh can be the minimum or maximum observed latency at the node h.

4.5.3. Network configuration for latency guarantee

A source requests an E2E latency bound for a flow, specifying its arrival rate, maximum packet length, and maximum burst size. If the E2E latency bound can be met, the network admits the flow. The network reserves the links in the path such that the sum of allocated service rates to the flows does not exceed the link capacity.

In the process of admission decision, the service rate allocated to a flow can be decided according to the requested latency bound of the flow. The detailed operational procedure for such admission and reservation is out of scope of this document.

4.5.4. Role of entrance node for generation and update of FT

It is assumed that the packet length information is written in the packet header. Entrance node maintains the flow state, e.g. FT of packet (p-1) at node 0 (F0(p-1)), the maximum packet length of the flow (L), and the service rate allocated to the flow (r). It operates a clock to identify the arrival time of a packet. It collects the link information such as the maximum packet length of all the flow (L0) and link capacity (R0) to calculate the delay factor at node 0.

Upon receiving or generating packet p, it obtains FT of packet p at node 0 (F0(p)), according to the VC algorithm and uses it as the service order in the entrance node. If the queue is not empty, then it puts p in a priority queue, in which the packets are sorted according to their FT. It also obtains FT of packet p at node 1 (F1(p)) before or during p is in the queue. It writes F1(p), L, and r in the packet as metadata for use in the next node 1. Finally, it updates the flow state information F0(p-1) to F0(p).

4.5.5. Role of core node for update of FT

A core node h collects the link information such as Lh and Rh. As in an entrance node, Lh is a rather static value, but still can be changed over time. Upon receiving packet p, it retrieves metadata Fh(p), L, and r, and uses Fh(p) as the FT value of the packet. It puts p in a priority queue. It obtains Fh+1(p) by updating Fh(p) with adding the delay factor and updates the packet metadata Fh(p) with Fh+1(p) before or during p is in the queue.

4.5.6. Mitigation of the complexity of entrance node

Flow states still have to be maintained in entrance nodes. When the number of flows is large, maintaining flow states can be burdensome. However, this burden can be mitigated as follows.

The notion of an entrance node can be understood as a various edge device, including a source itself. FT of a packet is decided based on the maximum of F0(p-1) and A0(p); and L(p)/r. These parameters are flow specific. There is no need to know any other external parameters. The arrival time of p to the network, A0(p), can be defined as the generation time of p at the source. Then F0(p) is determined at the packet generation time and can be recorded in the packet. In other words, the entrance node functionality can reside in the source itself.

Therefore, we can significantly alleviate the complexity of the proposed framework. The framework is scalable and can be applied to any network.

4.5.7. Compensation of time difference between nodes

We have assumed zero propagation delays between nodes so far. In reality, there are time differences between nodes, including the differences due to the propagation delays and due to the clock mismatches. This time difference can be defined as the difference between the service completion time measured at the upstream node and the arrival time measured at the current node.

FT does not need to be precise. It is used just to indicate the packet service order. Therefore, if we can assume that the propagation delay is constant and the clocks do not drift, then the time difference is constant for all the packets in a flow. In this case the delay factor in (2) can be modified by adding the time difference value. The E2E latency bound in (4) increases as much as the sum of propagation delays from node 0 to h.

The notion of an entrance node can be understood as a various edge device, including a source itself. FT of a packet is decided based on the maximum of F0(p-1) and A0(p); and L(p)/r. These parameters are flow specific. There is no need to know any other external parameters. The arrival time of p to the network, A0(p), can be defined as the generation time of p at the source. Then F0(p) is determined at the packet generation time and can be recorded in the packet. In other words, the entrance node functionality can reside in the source itself.

Therefore, we can significantly alleviate the complexity of the proposed framework. The framework is scalable and can be applied to any network.

Moreover, the time difference can be updated only once in a while. By the time difference compensation, the nodes become aware of the global clock discrepancies using a periodic quantification of the local clock discrepancies between adjacent nodes. Link by link, this ends up producing awareness of the discrepancies between the clocks of all the nodes, which is then included in the computation of FTs in core nodes. It is not synchronization in a strict sense because it does not involve the re-alignment of the clocks, only the quantification of their differences.

Even with the clock differences and propagation delays, the C-SCORE framework does not need global time synchronization.

4.6. Non-work conserving stateless core fair queuing

Stateless fair queuing techniques can also be implemented as non-work conserving. Core jitter virtual clock (CJVC) is a well known such scheduler [STOICA]. CJVC also enables isolation between flows by updating FT based only on nodal parameters and initial FT information as needed. The node at the edge of the network creates state information for each packet, including FT and other necessary information, and records them in the packet. Subsequent core nodes infer the exact flow states at the node based on these records without managing flow state information. CJVC mandates a packet to wait until an eligible time for a service start, thus a non-work conserving service.

In essence, CJVC calculates a packet's eligible time and finish time at a core node h as follows.

$$E0(p) = A0(p),$$
$$Eh(p) = Ah(p) + G(h-1)(p) + th(p), \quad (5)$$
$$Fh(p) = Eh(p) + L(p)/r.$$

Gh(p) is the difference between FT and the actual service completion time at h. th(p) is the delay that forces Eh(p) to be always greater than E(h-1)(p). This term guarantees the service order preservation between packets in a flow.

Consequently, Eh(p) can be expressed as a function of Ah(p) and other parameters that can be stored as metadata in the packet.

CJVC has the same E2E latency bound as that of PGPS and C-SCORE. However, it is shown that the jitter of CJVC is not better than the one of C-SCORE [C-SCORE].

Editor's note: A non-work conserving stateless core fair queueing, other than CJVC can be devised. Instead of (5), the following equations can be used.

$$E0(p) = \max\{F0(p-1), A0(p)\},$$
$$Eh(p) = E(h-1)(p) + L/r + Lh/Rh,$$
$$Fh(p) = Eh(p) + L(p)/r.$$

By using the above equations, it can be proved that the E2E latency of a flow is both upper and lower bounded. The upper bound of the E2E latency is identical to that of C-SCORE and PGPS. The jitter has also improved. This will be elaborated in the future versions.

5. Framework for Jitter Guarantee

5.1. Problem statement

The problem of guaranteeing jitter bounds in arbitrarily sized networks with any type of topology with random dynamic input traffic is considered.

There are several possible solutions to guarantee jitter bounds in packet networks, such as IEEE TSN's cyclic queuing and forwarding (CQF) [IEEE802.1Qch], its asynchronous variations [I-D.yizhou-detnet-ipv6-options-for-cqf-variant], and Latency-based Forwarding (LBF) [LBF].

CQF requires time-synchronization across every node in the network including the source. It is not scalable to a large network with significant propagation delays between the nodes. The asynchronous CQFs are scalable, but they may not satisfy applications' jitter requirements. This is because their jitter bounds cannot be controlled as desired, but are only determined by the cycle time, which should be large enough to accommodate all the traffic to be forwarded.

The systems with slotted operations such as CQF and its variations turn the problem of packet scheduling into the problem of scheduling flows to fit into slots. The difficulty of such a slot scheduling is a significant drawback in large scale dynamic networks with irregular traffic generations and various propagation delays.

LBF is a framework of the forwarding action decision based on the flow and packet status, such as the delay budget left for a packet in a node. LBF does not specify the actions to take according to the status. It suggests a packet slow down or speedup by changing the service order, by pushing packets into any desirable position of a first out queue, as a possible action to take. In essence, by having latency budget information of every packet, LBF is expected to maintain the latency and jitter within desired bounds. The processing latency required in LBF includes times 1) to lookup the latency budget information on every packet header, 2) to decide the queue position of the packet, 3) to modify the queue linked list, and 4) to update the budget information on the packet upon transmission. This processing latency, however, can affect the scalability especially in high speed core networks.

ATS, FAIR, and PFAR utilize the regulation function to proactively prevent the possible burst accumulation in the downstream nodes. It is not clear whether LBF can take such preventive action. If so, LBF can also act as a regulator and yield a similar latency bound.

5.2. Buffered network (BN)

The BN framework in this document for jitter bound guarantee is composed of

* a network that guarantees latency upper bounds;

* a timestamper for packets with a clock that is not necessarily synchronized with the other nodes, which resides in between, including the source and the network ingress interface; and

* a buffer that can hold the packets for a predetermined interval, which resides in between, including the destination and the network egress interface.

Figure 2 depicts the overall architecture of the BN framework for jitter-bound guarantees [BN]. Only a single flow is depicted between the source and destination in Figure 2. The arrival (an), departure (bn), and buffer-out (cn) times of the nth packet of a flow are denoted. The end-to-end (E2E) latency and the E2E buffered latency are defined as (bn-an) and (cn-an), respectively.

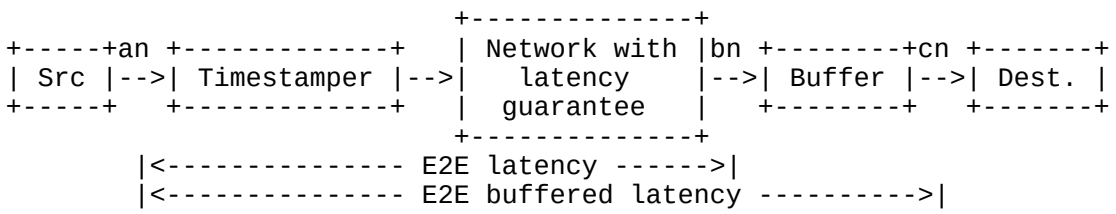

Figure 2: Buffered Network (BN) Framework for Jitter Guarantee

The buffer supports as many as the number of the flows destined for the destination. The destination shown in Figure 2 can be an end station or another deterministic network. The buffer holds packets in a flow according to predefined intervals. The decision of the buffering intervals involves the time-stamp value within each packet.

The network in between the time-stamper and the buffer can be of arbitrarily sized network. The input traffic can be dynamic. It is required that the network be able to guarantee and identify the E2E latency upper bounds of the flows. The network is also required to let the buffer be aware of the E2E latency upper bounds of the flows it has to process. It is recommended that the E2E latency lower bound information is provided by the network as well. The lower bound may be contributed from the transmission and propagation delays within the network.

The time-stamper marks on the packets their arrival times. The time-stamping function can use Real-time Transport Protocol (RTP) over User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). Either the source or network ingress interface can stamp the packet. In the case where the source stamps, the timestamp value is the packet departure time from the source, which is only a propagation time away from the packet arrival time to the network. The source and destination do not need to share a synchronized clock. All we need to know is the differences between the time stamps, that is, the information about the inter-arrival times.

5.3. Properties of BN

Let the arrival time of the nth packet of a flow be an. Similarly, let bn be the departure time from the network of the nth packet. Then, a1 and b1 are the arrival and departure times of the first packet of the flow, respectively. The first packet of a flow is defined as the first packet generated by the source, among all the packets that belong to the flow. Further, let cn be the buffer-out time of the nth packet of the flow. Let us define m as the jitter control parameter, which will be described later in detail.

Since buffers can be without cut-through capability, the processing delay within a buffer has to be taken into account. Let gn be the processing delay within the buffer of the nth packet of the flow. The gn includes the time to look up the timestamp and to store/ forward the packet. However, it does not include an intentional buffer-holding interval. By definition, cn - bn >= gn. Let max_n(gn)=g, the maximum processing delay for the flow in the buffer. It is assumed that a buffer can identify the value of g. Let U and W be the latency upper and lower bounds guaranteed to the flow by the network. Let m be the jitter control parameter, W+g <= m.

The rules for the buffer-holding interval decision are given as follows:

* c1=(b1+m-W),

* cn=max{(bn+g), (c1+an-a1)}, for n > 1.

The second rule governing the cn states that a packet should be held in the buffer to make its inter-buffer-out time, (cn-c1), equal to the inter-arrival time, (an-a1). However, when its departure from the network is too late, the inter-buffer-out time should be larger than the inter-arrival time, then hold the packet as much as the maximum processing delay in the buffer, that is, cn=bn+g. The buffer does not need to know the exact values of an or a1. It is sufficient to determine the difference between these values, which can be easily obtained by subtracting the timestamp values of the two packets.

The following theorems hold [ADN].

Theorem 1 (Upper bound of E2E buffered latency). The latency from the packet arrival to the buffer-out times (cn-an), is upper bounded by (U-W+m).

Theorem 2 (Lower bound of E2E buffered latency). The latency from the packet arrival to the buffer-out times (cn-an), is lower bounded by m.

Theorem 3 (Upper bound of jitter). The jitter is upper bounded by max{0, (U+g-m)}.

By setting m=(U+g), we can achieve zero jitter. In this case, the E2E buffered latency bound becomes (2U+g-W), which is roughly twice the E2E latency bound. In contrast, if we set m to its minimum possible value W+g, then the jitter bound becomes (U-W), which is roughly equal to U, while the E2E buffered latency bound becomes U, which is the same as the E2E latency bound.

The parameter m directly controls the holding interval of the first packet. It plays a critical role in determining the jitter and the buffered latency upper bounds of a flow in the BN framework. The larger the m, the smaller the jitter bound, and the larger the latency bound. With a sufficiently large m, we can guarantee zero jitter, at the cost of an increased latency bound.

5.4. Frequency synchronization between the source and the buffer

Clock drift refers to phenomena wherein a clock does not run at exactly the same rate as a reference clock. If we do not frequency-synchronize the clocks of different nodes in a network, clock drift is unavoidable. Consequently, jitter occurs owing to the clock frequency difference or clock drift between the source (timestamper) and the buffer. Therefore, it is recommended to frequency-synchronize the source (timestamper) and the buffer.

5.5. Omission of the timestamper

For isochronous traffic whose inter-arrival times are well-known fixed values, and the network can preserve the FIFO property for such traffic, then the timestampers can be omitted.

Otherwise the FIFO property cannot be guaranteed, then a sequence number field in the packet header would be enough to replace the timestamper.

5.6. Mitigation of the increased E2E buffered latency

The increased E2E buffered latency bound by the proposed framework, from U to almost 2U, can be mitigated by one of the added functionalities given as follows.

1) First, one can measure the E2E latency of a flow’s first packet exactly, and buffer it to make its E2E buffered latency be U. Then, by following the rules given in Section 5.3, every subsequent packet will experience the same E2E buffered latency, which is U, with zero jitter. An example of the exact latency measurement may be performed

by time-synchronization between the source (timestamper) and the buffer. However, how to measure the latency is for further investigation.

2) Second, one can expedite the first packet's service with a special treatment, to make its latency lower, compared to the other packets of the flow. If we can make the first packet's latency to be a small value d, then every packet will experience the same buffered latency d+U, with zero jitter. Considering that the E2E latency bound is calculated from the worst case in which rare events occur simultaneously, however, the first packet's latency is likely to be far less than what the bound suggests. Therefore, the special treatment to the first packet may be ineffective in real implementations.

5.7. Multi-sources single-destination flows' jitter control

The BN framework can also be used for jitter control among multiple sources' flows having a single destination. When a session is composed flows from more than one source, physically or virtually separated, the buffer at the boundary can mitigate the latency variations of packets from different sources due to different routes or network treatments. Such a scenario may arise in cases such as

1) that a central unit controls multiple devices for a coordinated execution in smart factories, or

2) multi-user conferencing applications, in which multiple devices/users physically separated can have a difficulty in real-time interactions.

The sources, or the ingress boundary nodes of the network, need to be synchronized with each other in order for the time-stamps from separated sources to be able to identify the absolute arrival times.

6. IANA Considerations

There are no IANA actions required by this document.

7. Security Considerations

This section will be described later.

8. Acknowledgements

9. Contributor

10. References

10.1. Normative References

[I-D.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T. T., Xiong, Q., Ryoo, J., zhushiyin, and X. Geng, "Requirements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-liu-detnet-large-scale-requirements-05, 20 October 2022, <https://datatracker.ietf.org/doc/html/draft-liu-detnet-large-scale-requirements-05>.

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

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

[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/info/rfc8655>.

[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/info/rfc8938>.

[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/info/rfc9320>.

10.2. Informative References

[ADN] Joung, J., Kwon, J., Ryoo, J., and T. Cheung, "Asynchronous Deterministic Network Based on the DiffServ Architecture", IEEE Access, vol. 10, pp. 15068-15083, doi:10.1109/ACCESS.2022.3146398, 2022.

[ANDREWS] Andrews, M., "Instability of FIFO in the permanent sessions model at arbitrarily small network loads", ACM Trans. Algorithms, vol. 5, no. 3, pp. 1-29, doi: 10.1145/1541885.1541894, July 2009.

[BN] Joung, J. and J. Kwon, "Zero jitter for deterministic networks without time-synchronization", IEEE Access, vol. 9, pp. 49398-49414, doi:10.1109/ACCESS.2021.3068515, 2021.

[BOUILLARD] Bouillard, A., Boyer, M., and E. Le Corronc, "Deterministic network calculus: From theory to practical implementation", in Networks and Telecommunications. Hoboken, NJ, USA: Wiley, doi: 10.1002/9781119440284, 2018.

[C-SCORE] Joung, J., Kwon, J., Ryoo, J., and T. Cheung, "Scalable flow isolation with work conserving stateless core fair queuing for deterministic networking", IEEE Access, vol. 11, pp. 105225 - 105247, doi:10.1109/ACCESS.2023.3318479, 2023.

[FAIR] Joung, J., "Framework for delay guarantee in multi-domain networks based on interleaved regulators", Electronics, vol. 9, no. 3, p. 436, doi:10.3390/electronics9030436, March 2020.

[I-D.yizhou-detnet-ipv6-options-for-cqf-variant] Li, Y., Ren, S., Li, G., Yang, F., Ryoo, J., and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants", Work in Progress, Internet-Draft, draft-yizhou-detnet-ipv6-options-for-cqf-variant-02, 26 April 2023, <https://datatracker.ietf.org/doc/html/draft-yizhou-detnet-ipv6-options-for-cqf-variant-02>.

[IEEE802.1Qch] IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding", IEEE 802.1Qch-2017, DOI 10.1109/IEEESTD.2017.7961303, 28 June 2017, <https://doi.org/10.1109/IEEESTD.2017.7961303>.

[IEEE802.1Qcr] IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping", IEEE 802.1Qcr-2020, DOI 10.1109/IEEESTD.2020.9253013, 6 November 2020, <https://doi.org/10.1109/IEEESTD.2020.9253013>.

[KAUR] Kaur, J. and H.M. Vin, "Core-stateless guaranteed rate scheduling algorithms", in Proc. INFOCOM, vol.3, pp. 1484-1492, 2001.

[LBF] Clenm, A. and T. Eckert, "High-precision latency forwarding over packet-programmable networks", NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, April 2020.

[LEBOUDEC] Le Boudec, J., "A theory of traffic regulators for deterministic networks with application to interleaved regulators", IEEE/ACM Trans. Networking, vol. 26, no. 6, pp. 2721-2733, doi:10.1109/TNET.2018.2875191, December 2019.

[PAREKH] Parekh, A. and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case", IEEE/ACM Trans. Networking, vol. 1, no. 3, pp. 344-357, June 1993.

[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/info/rfc2212>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <https://www.rfc-editor.org/info/rfc3393>.

[STILIADIS-LRS] Stiliadis, D. and A. Anujan, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms", IEEE/ACM Trans. Networking, vol. 6, no. 5, pp. 611-624, 1998.

[STOICA] Stoica, I. and H. Zhang, "Providing guaranteed services without per flow management", ACM SIGCOMM Computer Communication Review, vol. 29, no. 4, pp. 81-94, 1999.

[THOMAS] Thomas, L., Le Boudec, J., and A. Mifdaoui, "On cyclic dependencies and regulators in time-sensitive networks", in Proc. IEEE Real-Time Syst. Symp. (RTSS), York, U.K., pp. 299-311, December 2019.

[Y.3113] International Telecommunication Union, "Framework for Latency Guarantee in Large Scale Networks Including IMT-2020 Network", ITU-T Recommendation Y.3113, February 2021.

[ZHANG] Zhang, L., "Virtual clock: A new traffic control algorithm for packet switching networks", in Proc. ACM symposium on Communications architectures & protocols, pp. 19-29, 1990.

Authors' Addresses

Jinoo Joung
Sangmyung University
Email: jjoung@smu.ac.kr

Jeong-dong Ryoo
ETRI
Email: ryoo@etri.re.kr

Taesik Cheung
ETRI
Email: cts@etri.re.kr

Yizhou Li
Huawei
Email: liyizhou@huawei.com

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com