DetNet
Internet-Draft
Intended status: Standards Track
Expires: 16 July 2026

Shaofu. Peng
ZTE
Peng. Liu
China Mobile
Kashinath. Basu
Oxford Brookes University
Aihua. Liu
ZTE
Dong. Yang
Beijing Jiaotong University
Guoyu. Peng
Beijing University of Posts and Telecommunications
Junfeng. Zhao
CAICT

12 January 2026

# Timeslot Queueing and Forwarding Mechanism
# draft-ietf-detnet-packet-timeslot-mechanism-00

## Abstract

IP/MPLS networks use packet switching (with the feature store-and-forward) and are based on statistical multiplexing. Statistical multiplexing is essentially a variant of time division multiplexing, which refers to the asynchronous and dynamic allocation of link timeslot resources. In this case, the service flow does not occupy a fixed timeslot, and the length of the timeslot is not fixed, but depends on the size of the packet. Statistical multiplexing has certain challenges and complexity in meeting deterministic QoS, and its delay performance is dependent on the used queueing mechanism. This document further describes a generic time division multiplexing scheme for layer-3 in an IP/MPLS networks, called timeslot queueing and forwarding (TQF) mechanism. TQF is an enhancement based on TSN TAS and allows the data plane to create a flexible timeslot mapping scheme based on available timeslot resources. It defines a cyclic period consisting of multiple timeslots where a flow is assigned to be transmited within one or more dedicated timeslots. The objective of TQF is to better handle large scaling requirements.

## Status of This Memo

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

This Internet-Draft will expire on 16 July 2026.

## Copyright Notice



## Table of Contents

## 1. Introduction

IP/MPLS networks use packet switching (with the feature store-and-forward) and are based on statistical multiplexing. The discussion of supporting multiplexing in the network was first seen in the time division multiplexing (TDM), frequency division multiplexing (FDM) and other technologies of telephone communication network (using circuit switching). Statistical multiplexing is essentially a variant of time division multiplexing, which refers to the asynchronous and dynamic allocation of link resources. In this case, the service flow does not occupy a fixed timeslot, and the length of the timeslot is not fixed, but depends on the size of the packet. In contrast, synchronous time division multiplexing means that a sampling frame (or termed as time frame) includes a fixed number of fixed length timeslots, and the timeslot at a specific position is allocated to a specific service. The utilization rate of link resources in statistical multiplexing is higher than that in synchronous time division multiplexing. However, if attempting to provide deterministic end-to-end delay in packet switched networks based on statistical multiplexing, the difficulty is greater than that in synchronous time division multiplexing. The main challenge is to obtain a deterministic upper bound on the queueing delay, which is closely related to the queueing mechanism used in the network.

In addition to IP/MPLS network, other packet switched network technologies, such as ATM, also discusses how to provide corresponding transmission quality guarantee for different service types. Before service communication, ATM needs to establish a connection to reserve virtual path/channel resources, and use fixed-

length short cells and timeslots. The advantage of short cell is small interference delay, but the disadvantage is low encoding efficiency. The mapping relationship between ATM cells and timeslots is not fixed, so it still depends on a specific cells scheduling mechanism (such as [ATM-LATENCY]) to ensure delay performance. Although the calculation of delay performance based on short and fixed-length cells is more concise than that of IP/MPLS networks based on variable length packets, they all essentially depend on the queueing mechanism.

[TAS] introduces a synchronous time-division multiplexing method based on gate control list (GCL) rotation in Ethernet LAN. Its basic idea is to calculate when the packets of the service flow arrive at a certain node, then the node will turn on the green light (i.e., the transmission state is set to OPEN) for the corresponding queue inserted by the service flow at that time duration, which is defined as TimeInterval between two adjacent items in gating cycle. The TimeInterval is exactly the timeslot resource that can be reserved for service flow. A set of queues is controlled by the GCL, with round robin per gating cycle. The gating cycle (e.g, 250 us) contains a lot of items, and each item is used to set the OPEN/CLOSED states of all traffic class queues. By strictly controlling the release time of service flow at the network entry node, multiple flows always arrive sequentially during each gating cycle at the intermediate node and are sent during their respective fixed timeslot to avoid conflicts, with extremely low queueing delay. However, the GCL state (i.e., items set, and different TimeInterval value between any two adjacent items) is related with all admitted flows that passes through the node. Calculating and installing GCL states separately on each node has scalability issues.

[CQF] introduces a synchronous time-division multiplexing method based on fixed-length cycle in Ethernet LAN. [ECQF] is a further enhancement of the classic CQF. CQF with 2-buffer mode or ECQF with x-bin mode only uses a small number of cycles to establish the cycle mapping between a port-pair of two adjacent nodes, which is independent of the individual service flow. The cycle mapping may be maintained on each node and swaped based on a single cycle id carried in the packet during forwarding ([I-D.eckert-detnet-tcqf]), or all cycle mappings are carried in the packet as a cycle stack and read per hop during forwarding ([I-D.chen-detnet-sr-based-bounded-latency]). According to [ECQF], how many cycles (i.e., x-bin mode) are required depends on the proportion of the variation in intra-node forwarding delay relative to the cycle size. If the proportion is small, 3-bin is enough, otherwise, more than 3 bins needed. Compared to TAS, CQF/ECQF no longer maintains GCL on each node, but instead replaces the large number of variable length of timeslots related to service flows in

GCL with a small number of fixed length cycles unrelated to service flows. Thus, CQF/ECQF simplifies the data plane, but leaves the complexity to the control plane, by calculating and controlling the release time of service flow at the network entry, to guarantee no conflicts between flows in any cycle on any intermediate nodes.

In order to meet the large scaling requirements, this document describes a scheduling mechanism for enhancing TAS. It defines a cyclic period consisting of multiple timeslots that share limited buffer resources, and a flow is assigned to be transmited within one or more dedicated timeslots. It does not rely on time synchronization, but needs to detect and maintain the phase difference of cyclic period between adjacent nodes. It further defines two scheduling modes: on-time or in-time mode. This mechanism is named Timeslot Queueing and Forwarding (TQF), as a supplement to IEEE 802.1 TSN TAS. In TQF, the selected length of the cyclic period (i.e., gating cycle of TAS) depends on the length of the supported service burst interval. Note that TQF is a layer-3 solution and can operate over different types of QoS sensitive layer-2 including TSN but is not an alternative to TSN.

Similar to TAS and CQF/ECQF, TQF is also TDM based scheduling mechanisms. However, their differences are as below:

* Compared to classic TAS, TQF may use round robin queues corresponding to the count of timeslots during gating cycle, while TAS only maintains queues corresponding to the number of traffic classes and one of them is used for the Scheduled Traffic (i.e., DetNet flows). That means TQF need more queues than TAS (i.e., multiple timeslot queues vs single traffic class queue). However, TAS needs to use other complex methods to control the arrival order of all flows sharing the same traffic class queue to isolate them (so that each flow faces almost zero queuing delay), while TQF’s timeslot queue naturally isolates flows by timeslot id of gating cycle. TAS can be seen as a special case of TQF (where M = 1). And, TQF with in-time scheduling mode may use a single Push-in First-out (PIFO) queue to approximate the ultra-low delay of TAS.

* Compared to CQF/ECQF, TQF on-time scheduling maintains round robin queues corresponding to the count of timeslots during gating cycle, while CQF/ECQF maintains extra tolerating queues depending on the proportion of the variation in intra-node forwarding delay relative to the cycle size. There is no gating cycle with its timeslot resources designed by CQF/ECQF, it needs to use additional flow interleaving method to control the arrival order of flows sharing the same cycle queue to isolate flows (or alternatively tolerate overprovision), while TQF’s timeslot queue

naturally isolates flows by timeslot id of gating cycle. This is also the semantic difference between cycle id and timeslot id, where the former is used to indicate the NO. of the aggregated queues such as sending, receiving, or tolerating queue that share the same resources per CQF instance, rather than indicating the individual timeslot resource within the gating cycle like the later. After defining timeslot resources in IP/MPLS network, TQF does not limit the implementations of the data structure type corresponding to timeslot resources on the data plane, which may be round robin queues, or a single PIFO queue.

## 1.1. Requirements Language

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

# 2. Terminology

The following terminology is introduced in this document:

Timeslot: The unit of TQF scheduling. It needs to design a reasonable value, such as 10us, to send at least one complete packet. Different nodes can be configured with different length of timeslot.

Timeslot Scheduling: The packet is stored in the buffer zone corresponding to a specific timeslot id, and may be sent before (in-time mode) or within (on-time mode) that timeslot. The timeslot id is always a NO. from the orchestration period.

Service Burst Interval: The traffic specification of DetNet flow generally follows the principle of generating a specific burst amounts within a specific length of periodic burst interval. For example, a service generates 1000 bits of burst per 1 ms, where 1 ms is the service burst interval.

Orchestration Period: The orchestration period is a cyclic period and used to instantiate timeslot resources on the link. The selection of orchestration period length depends on the service burst interval of DetNet flows, e.g., the Least Common Multiple of service burst intervals of all flows. It is actually the gating cycle in TAS, just with different queue allocation rules. It contains a fixed count (termed as N and numbered from 0 to N-1) of timeslots. For example, the orchestration period include 1000 timeslots and each timeslot length is 10 us. Multiple

orchestration period length may be enabled on the link, but all nodes included in the DetNet path must interoperate based on the same orchestration period length. A specific orchestration period length can be used to indicate the corresponding TQF scheduling instance.

Ongoing Sending Period: The orchestration period which the ongoing sending timeslot belongs to.

Scheduling Period: The scheduling period of a TQF scheduling instance depends on the hardware's buffer resources that is supported by the device. Its length reflects the count of the timeslot resources (termed as M and numbered from 0 to M-1) with related buffer resources that is actually instantiated on the data plane, which is limited by hardware capabilities. Scheduling period length may be less than or equal to orchestration period length in the case of on-time mode, or larger than or equal to orchestration period length in the case of in-time mode. Packets belonging to a specific timeslot (in orchestration period) will be mapped and stored in the buffer zone of the corresponding timeslot of the scheduling period.

Incoming Timeslot: For the headend of the DetNet path, the current timeslot of UNI at which a flow arriving and after being policing is the incoming timeslot. For any intermediate node of the DetNet path, the timeslot contained in the packet received from the upstream node (i.e., the outgoing timeslot of the upstream node) is the incoming timeslot. An incoming timeslot id is the timeslot in the context of the orchestration period.

Outgoing Timeslot: When sending a packet to the outgoing port, according to resource reservation or certain rules, it chooses to send packet in the specified timeslot of that port, which is the outgoing timeslot. An outgoing timeslot id is the timeslot in the context of the orchestration period.

Ongoing Sending Timeslot: When the end of the incoming timeslot to which the packet belongs reaches a specific port, the timeslot currently in the sending state is the ongoing sending timeslot of that port. Note that the ongoing sending timeslot is different with the outgoing timeslot. An ongoing sending timeslot id is the timeslot in the context of the orchestration period.

## 3. Overview

This scheme introduces the time-division multiplexing scheduling mechanism based on the fixed length timeslot in the IP/MPLS network. Note that the time-division multiplexing here is a L3 packet-level scheduling mechanism, rather than the TDM port (such as SONET/SDH) implemented in L1. The latter generally involves the time frame and the corresponding framing specification, which is not necessary in this document. The data structure associated with timeslot resources may be implemented using round robin queues, or a single PIFO queue, etc.

Figure 1 shows the TQF scheduling behavior implemented by the intermediate node P through which a deterministic path passes.

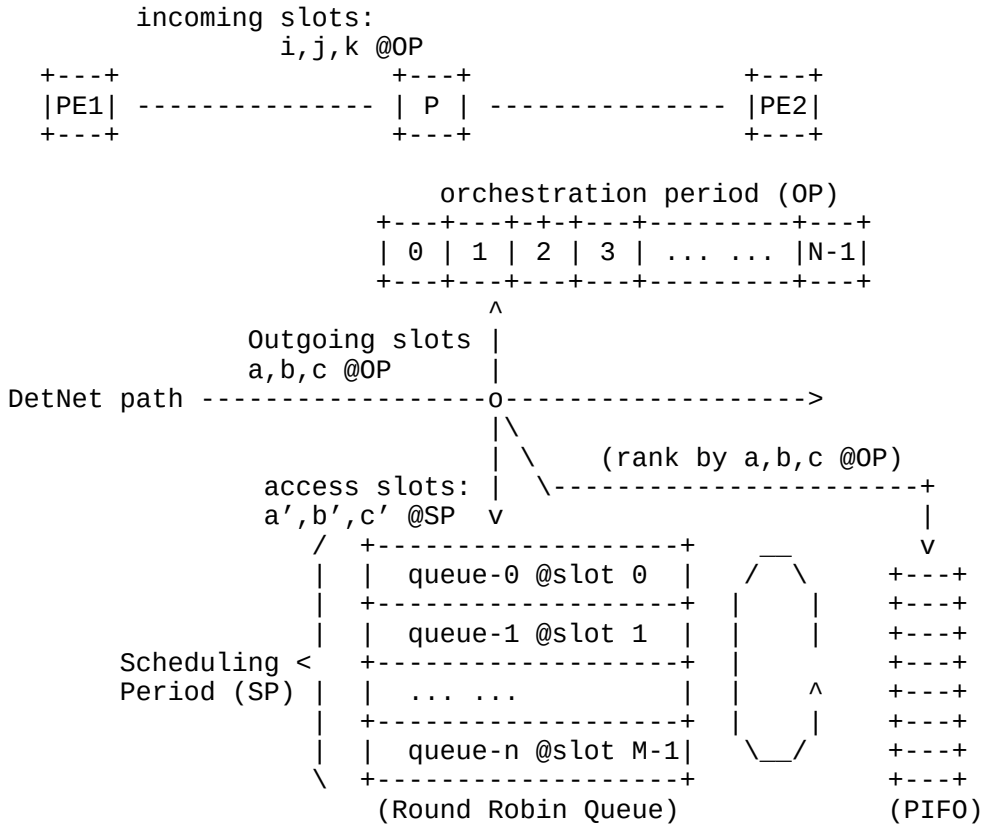


Figure 1: Timeslot Based Scheduling Mechanism

Where, both the orchestration period and the scheduling period consist of multiple timeslots, the number of timeslots supported by orchestration period is related to the length of the service burst interval, while the number of timeslots supported by scheduling period is limited by hardware capabilities, and it may be instantiated by a Round Robin queue or PIFO.

A TQF enabled link may configure multiple TQF scheduling instances each with specific orchestration period length. Nodes communicate with each other based on the same instance.

Each TQF scheduling instance related to a specific orchestration period length may configure its service rate, and the sum of service rates of all instances must not exceed the port bandwidth. For a TQF scheduling instance, the total amount of bits that can be consumed in each timeslot is generally not exceeding the result of the service rate multiplied by the timeslot length.

For each orchestration period length, all nodes in the network does not require phase alignment. The phase difference of orchestration period between adjacent nodes should be detected.

For a specific orchestration period configured on different links in the network, these links may configure different timeslot lengths because their capabilities vary, for example, the link capability of the edge nodes is weaker than that of core nodes.

In Figure 1, a DetNet path consumes timeslots i, j, k from the orchestration period of the link PE1-P, and a, b, c from the orchestration period of the link P-PE2 respectively. From node P's perspective, i, j, k are incoming timeslots, while a, b, c are outgoing timeslots. The cross connection between an incoming timeslot and an outgoing timeslot will result in the corresponding residence delay, which depends on the offset between the incoming and outgoing timeslots based on the phase difference of the orchestration periods of link PE1-P and P-PE2.

An outgoing timeslot in the orchestration period will finally access the mapped timeslot in the scheduling period. There is a mapping function from the timeslot z in the orchestration period to the timeslot z' in the scheduling period, i.e., z' = f(z). For example, the mapping function may be z' = z, z' = z + offset, z' = z % M, and z' = random(z), etc. Which function to use depends on the queue allocation rule and data structure instantiated for timeslot resources. In this document, two mapping functions are mainly introduced: z' = z % M (in the case of round robin queue), and z' = z (in the case of PIFO).

The controller plane function used for TQF, to calculate a DetNet path with the allocated timeslots that meets the flow requirements, can be found in [I-D.peng-detnet-tqf-controller-plane].

## 4. Timeslot Mapping Relationship

In order to determine the offset between the incoming timeslot and the outgoing timeslot in the context of specific TQF scheduling instance that is identified by orchestration period length, it is necessary to first determine the ongoing sending timeslot that the incoming timeslot falls into, i.e., the mapping relationship between the incoming timeslot and the ongoing sending timeslot.

Two methods are provided in the following sub-sections to determine the mapping relationship between the incoming timeslot and the ongoing sending timeslot.

### 4.1. Deduced by BTM

Figure 2 shows that there are three nodes U, V, and W in turn along the path. All nodes are configured with the same orchestration period length (termed as OPL), which is crucial for establishing a fixed timeslot mapping relationship between the adjacent nodes.

* Port_u2 has timeslot length L_u2, and an orchestration period contains N_u2 timeslots.

* Port_v1 has timeslot length L_v1, and an orchestration period contains N_v1 timeslots.

* Port_v2 has timeslot length L_v2, and an orchestration period contains N_v2 timeslots.

Hence, L_u2*N_u2 = L_v1*N_v1 = L_v2*N_v2. In general, the link bandwidth of edge nodes is small, and they may be configured with a larger timeslot length than the aggregated/backbone nodes.

It has been mathematically proven that if the least common multiple of L_u# and L_v# is LCM, OPL is also a multiple of LCM.

Node U may send a detection packet from the end (or head, the process is similar) of an arbitrary timeslot i of port_u2 connected to node V. After a certain link propagation delay (D_propagation), the packet is received by the incoming port of node V, and i is regarded as the incoming timeslot by V. At this time, the ongoing sending timeslot of port_v1 is j, and there is time T_ij left before the end of the timeslot j.

This mapping relationship is termed as:

* <instance OPL, port_u2 slot i, port_v1 slot j, T_ij>

To avoid confusion, this mapping relationship is called the base timeslot mapping (BTM), as it is independent of the DetNet flows. Instead, another mapping relationship that depends on the specific flow, e.g., between the outgoing timeslot of port_u2 and the outgoing timeslot of port_v2, is called the forwarding timeslot mapping (FTM).

BTM is generally maintained by node V when processing probe message received from node U. However, node U may also obtain this information from node V, e.g, by an ACK message. The advantage of maintaining BTM by node U is that it is consistent with the unidirectional link from node U to V, so it is more appropriate for node U (rather than V) to advertise it in the network. A BTM detection method can be found in [I-D.xp-ippm-detnet-stamp], and the advertisement method can be found in [I-D.peng-lsr-deterministic-traffic-engineering].

Note that this document does not recommend directly detecting and maintaining BTM between the outgoing timeslot of port_u2 and the ongoing sending timeslot of port_v2 (i.e., the outgoing port of downstream node V), as this is too trivial. In fact, as shown above, maintaining only BTM between the outgoing timeslot of port_u2 and the ongoing sending timeslot of port_v1 (i.e., the incoming port of downstream node V) is sufficient to derive other mapping relationships.

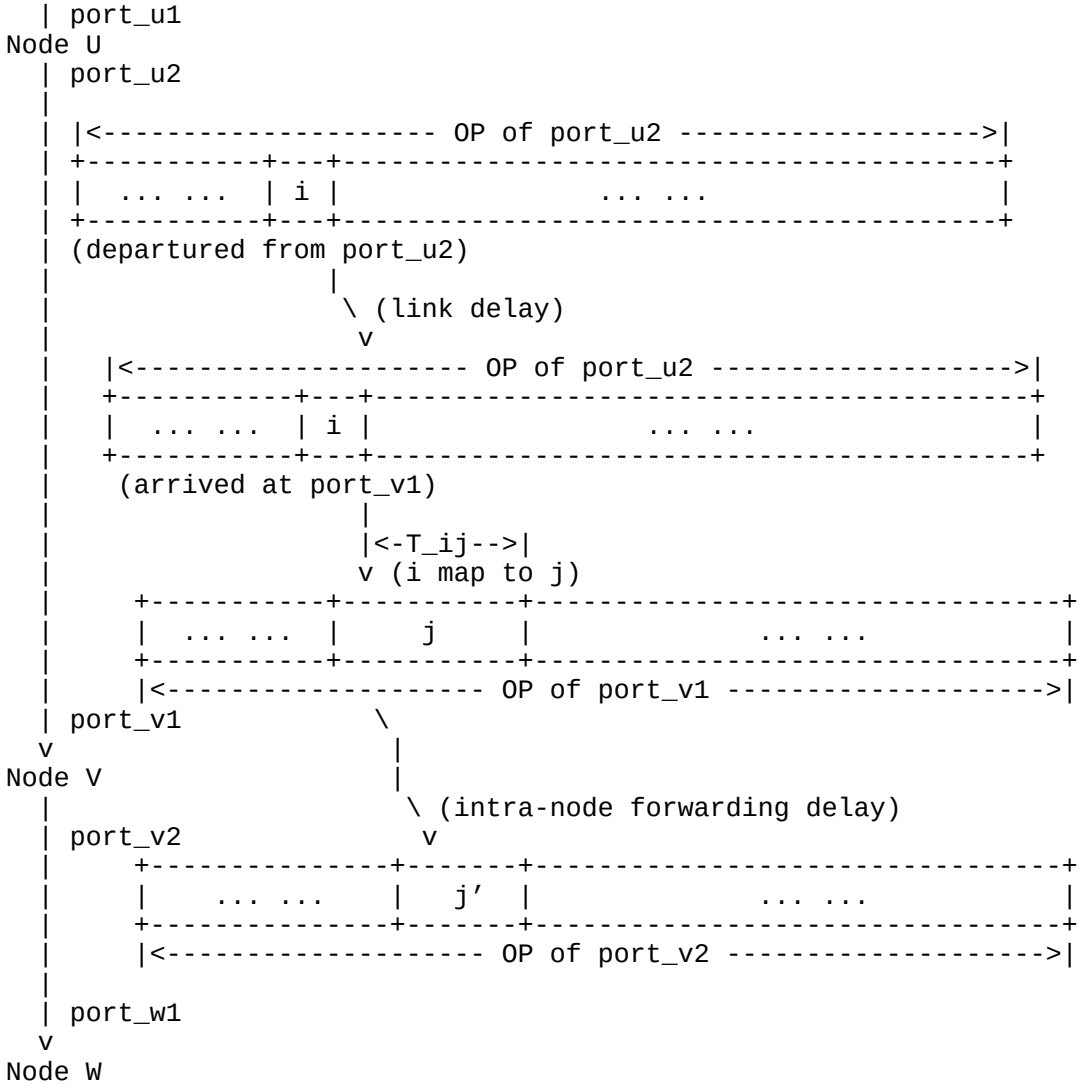


Figure 2: BTM Detection

Based on the above detected BTM, and knowing the intra-node forwarding delay (F) including parsing, table lookup, internal fabric exchange, BTM between any outgoing timelot x of port_u2 and the ongoing timeslot y of port_v2 can be derived.

Let t is the offset between the end of the timeslot x of port_u2 and the beginning of the orchestration period of the port_v2.

* t = ((j'+1)*L_v1 - T_ij' + OPL + (x-i)*L_u2 + F) % OPL

Then,

* y = [t/L_v2]

And the time T_xy left before the end of the timeslot y is:

* T_xy = (y+1)*L_v2 - t

This document recommends that the time of each port within the same node must be synchronized, that is, all ports of a node share the same local system time, which is easy to achieve. It is also recommended that the begin time of the orchestration period for all ports within the same node be the same or differ by an integer multiple of OPL, e.g, maintaining a global initial time as the logical begin time for the first round of orchestration period for all ports. Whether node restart or port restart, this initial time should continue to take effect to avoid affecting the timeslot mapping relationship between each node. Depending on the implementation, considering that the initial time may be a historical time that is too far away from the current system time, regular updates may be made to it (e.g, self increasing k*OPL, where k is a natural number) to be closer to the current system time.

## 4.2. Deduced by BOM

Figure 3 shows that there are three nodes U, V, and W in turn along the path. Similar to Section 4.1, it still has L_u2*N_u2 = L_v1*N_v1 = L_v2*N_v2.

Node U may send a detection packet from the head (or end, the process is similar) of the orchestration period of port_u2 connected to node V. After a certain link propagation delay (D_propagation), the packet is received by the incoming port of node V. At this time, there is time P_uv left before the end of the ongoing sending period of port_v1.

This mapping relationship is termed as:

* <instance OPL, port_u2, port_v1, P_uv>

This mapping relationship is called the base orchestration-period mapping (BOM), which it is independent of the DetNet flows.

BOM is generally maintained by node V when processing probe message received from node U. However, node U may also obtain this information from node V, e.g, by an ACK message. The advantage of

maintaining BOM by node U is that it is consistent with the unidirectional link from node U to V, so it is more appropriate for node U (rather than V) to advertise it in the network. A BOM detection method can be found in [I-D.xp-ippm-detnet-stamp], and the advertisement method can be found in [I-D.peng-lsr-deterministic-traffic-engineering].

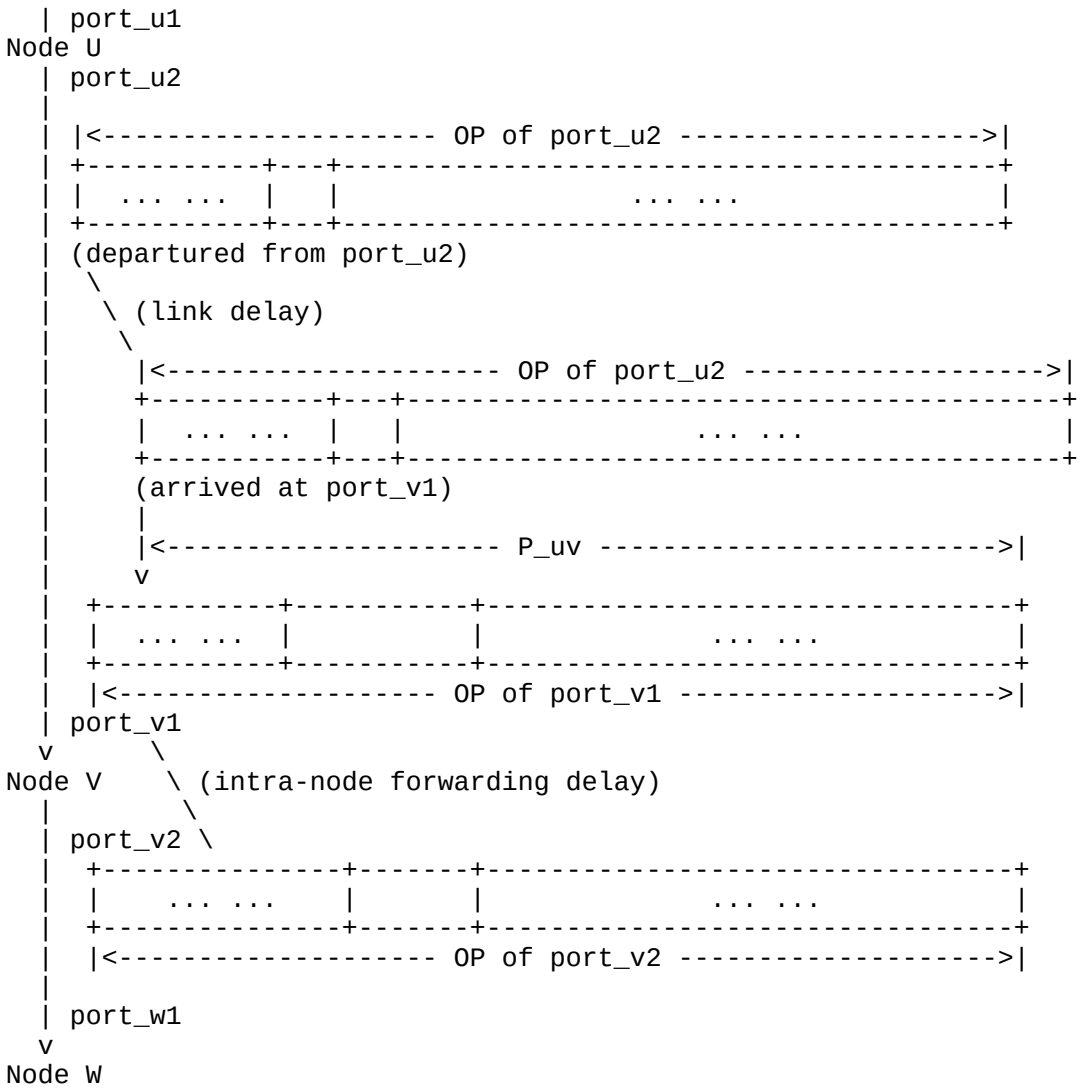


Figure 3: BOM Detection

Based on BOM, and knowing the intra-node forwarding delay (F), the mapping relationship between any outgoing timelot x of port_u2 and the ongoing timeslot y of port_v2, can be derived.

Let t is the offset between the end of the timeslot x of port_u2 and the beginning of the orchestration period of the port_v2.

* t = ((x+1)*L_u2 + OPL - P_uv + F) % OPL

Then,

* y = [t/L_v2]

And the time T_xy left before the end of the timeslot y is:

* T_xy = (y+1)*L_v2 - t

## 5. Resources Used by TQF

The operation of TQF scheduling mechanism will consume two types of resources:

* Bandwidth: Each TQF scheduling instance may configure its service rate that is a dedicated bandwidth resource from the outgoing port, and the sum of service rates of all instances must not exceed the port bandwidth.

* Burst: The burst resources of a specific TQF scheduling instance can be represented as the corresponding bit amounts of all timeslots included in the orchestration period. The total amount of bits that can be consumed by flows in each timeslot is generally not exceeding the result of the service rate multiplied by the timeslot length.

## 6. Arrival Postion in the Orchestration Period

Generally, a DetNet flow has its TSpec, such as periodically generating traffic of a specific burst size within a specific length of burst interval, which regularly reaches the network entry. The headend executes traffic regulation (e.g, setting appropriate parameters for leaky bucket shaping), which generally make packets evenly distributed within the service burst interval, i.e, there are one or more shaped sub-burst in the service burst interval. There is an ideal positional relationship between the regulated time (the moment when each sub-burst leaves the regulator) and the orchestration period of UNI port, that is, each sub-burst corresponds to an ideal incoming timeslot of UNI port. Based on the ideal incoming timeslot, an ideal outgoing timeslot of NNI port may be

selected and consumed by the sub-burst.

For example, if a DetNet flow distributes m sub-bursts during the orchestration period, the network entry should maintain m items for that flow:

* <OPL, ideal incoming slot i_1, ideal outgoing slot z_1>

* <OPL, ideal incoming slot i_2, ideal outgoing slot z_2>

* ... ...

* <OPL, ideal incoming slot i_m, ideal outgoing slot z_m>

However, the packets arrived at the network entry are not always ideal, and the regulated time may not be in a certain ideal incoming timeslot. Therefore, an important operation that needs to be performed by the network entry is to determine the ideal incoming timeslot i based on the regulated time. This can first determine the actual incoming timeslot based on the regulated time, and then match an item with the ideal incoming timeslot that is closest to and not earlier than the actual incoming timeslot. Alternatively, another match operation is to match an item with the ideal outgoing timeslot that is closest to and not earlier than the actual incoming timeslot. Note that all sub-bursts of the same flow must use the same match operation, otherwise, inconsistent operations may cause them to conflict in the same outgoing timeslot.

Figure 4 shows, for some typical DetNet flows, the ideal incoming timeslots in the orchestration period of UNI, as well as the ideal outgoing timeslots of NNI consumed by these DetNet flows.

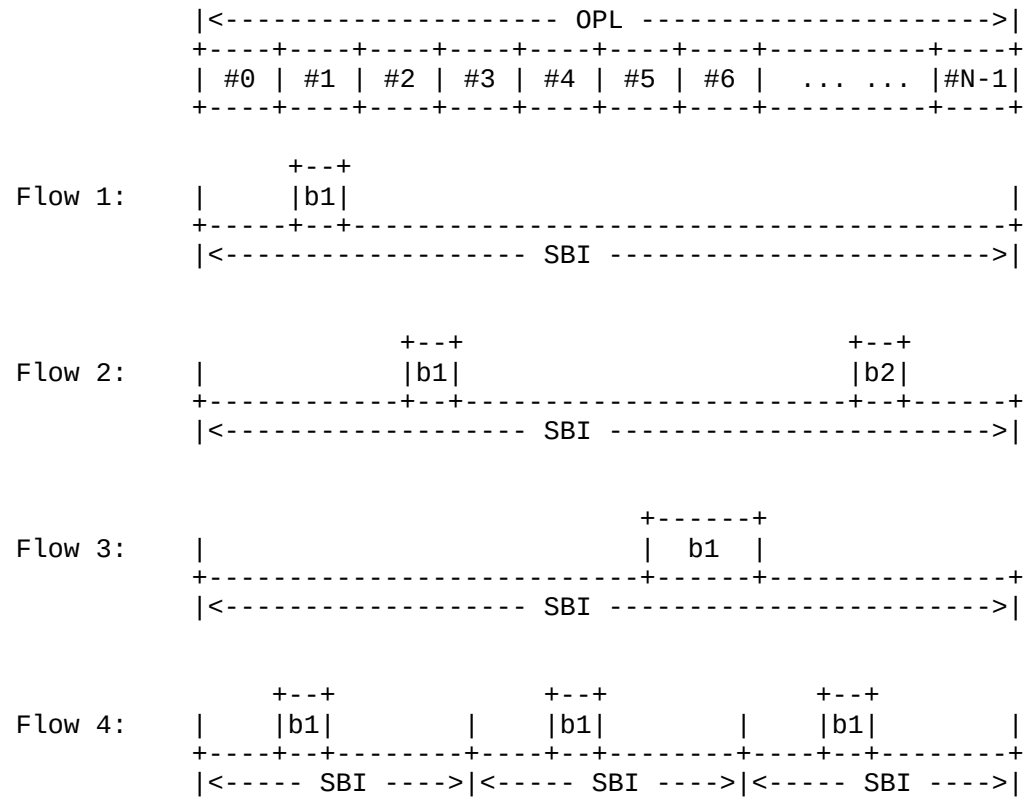


Figure 4: Relationship between SBI and OP

As shown in the figure, the service burst interval of flows 1, 2, 3 is equal to the orchestration period length, while the service burst interval of flow 4 is only 1/3 of the orchestration period length.

* Flow 1 generates a small single burst within its burst interval, which may consume timeslot 2 or other subsequent timeslot of NNI;

* Flow 2 generates two small discrete sub-bursts within its burst interval and also be shaped, which may respectively consume slots 4 and N-1 of NNI;

* Flow 3 generates a large single burst within its burst interval but not be really shaped (due to purchasing a larger burst resource and served by a larger bucket depth), which may also be split to multiple back-to-back sub-bursts and consume multiple consecutive timeslots, such as 8 and 9 of NNI.

* The service burst interval of flow 4 is only 1/3 of the orchestration period. Hence, construct flow 4’ with 3 occurrence of the flow 4 within an orchestration period. So flow 4’ is similar to flow 2, generating three separate sub-bursts within its burst interval. It may consume timeslots 3, 7, and N-1 of NNI.

# 7. Residence Delay Evaluation

## 7.1. Residence Delay on the Ingress Node

On the headend H, the received flow corresponds to an ideal incoming timeslot i of UNI port. Although there is actually no timeslot mapping relationship established between the end-system and the headend, it can still be assumed that the end-system applies the same orchestration period as UNI, and the BOM with phase aligned is detected. Then, according to Section 4, for the above incoming timeslot i, the ongoing sending timeslot j of NNI port, as well as the remaining time T_ij of timeslot j , can be deduced.

An outgoing timeslot z of NNI, which offset o (>=1) timeslots from j, can be selected for the flow. That is, z = (j+o)%N_h2, where N_h2 is the number of timeslots in the orchestration period of NNI.

Thus, on the headend H the residence delay obtained from the outgoing timeslot z is:

Best Residence Delay = F + T_ij + (o-1)*L_h2

Worst Residence Delay = F + T_ij + L_h1 + o*L_h2

Average Residence Delay = F + T_ij + (L_h1 + (2o-1)*L_h2)/2

where, L_h1 is the timeslot length of UNI port, L_h2 is the timeslot length of NNI port.

The best residence delay occurs when the flow arrived at the end of the ideal incoming timeslot i, and sent at the head of the outgoing timeslot z.

The worst residence delay occurs when the flow arrived at the head of the ideal incoming timeslot i, and sent at the end of the outgoing timeslot z.

The delay jitter within the headend is (L_h1 + L_h2). However, the jitter of the entire path is not the sum of the jitters of all nodes.

Note that there is a runtime jitter, as mentioned in Section 6, which depends on the deviation between the regulated time t0 and the begin time of the ideal incoming timeslot i, called the arrival deviation, i.e., i.begin - t0. A maximum arrival deivation can be set for each sub-burst of the flow. The maximum arrival deviation can be set as the interval between the adjacent ideal incoming timeslots. The scheduling period length also affect the value of maximum arrival deviation (refer to Section 13). Even if t0 is in the ideal incoming timeslot i, there will usually be a negative value of arrival deivation. Further operations to eliminate jitter (refer to Section 7.4) must consider any possible positions of t0 uniformly.

### 7.2. Residence Delay on the Transit Node

On the transit node V, according to Section 4, for any given incoming timeslot i, the ongoing sending timeslot j of the outgoing port (port_v2), as well as the remaining time T_ij of timeslot j, can be deduced.

An outgoing timeslot z of the outgoing port, which offset o (>=1) timeslots from j, can be selected for the flow. That is, z = (j+o)%N_v2, where N_v2 is the number of timeslots in the orchestration period of port_v2.

Thus, on the transit node V the residence delay obtained from the outgoing timeslot z is:

Best Residence Delay = F + T_ij + (o-1)*L_v2

Worst Residence Delay = F + T_ij + L_u2 + o*L_v2

Average Residence Delay = F + T_ij + (L_u2+(2o-1)*L_v2)/2

where, L_u2 and L_v2 is the timeslot length of port_u2 and port_v2 respectively.

The best residence delay occurs when the flow is received at the end of the incoming timeslot i and sent at the head of the outgoing timeslot z.

The worst residence delay occurs when the flow is received at the head of the incoming timeslot i and sent at the end of the outgoing timeslot z.

The delay jitter within the node is (L_u2 + L_v2). However, the jitter of the entire path is not the sum of the jitters of all nodes.

### 7.3. Residence Delay on the Egress Node

Generally, for the deterministic path carrying the DetNet flow, the flow needs to continue forwarding from the outgoing port of the egress node to the client side, and also faces the issues of queueing. However, the outgoing port facing the client side is the part of the overlay routing. It is possible to continue supporting TQF mechanism on that port. In this case, the underlay DetNet path will serve as a virtual link of the overlay path, providing a deterministic delay performance.

Therefore, for the underlay deterministic paths, the residence dalay on the egress node is only contributed by the forwarding delay (F) including parsing, table lookup, internal fabric exchange, etc.

### 7.4. End-to-end Delay and Jitter

Figure 5 shows that a path from headend P1 to endpoint E, for each node Pi, the timeslot length of the outgoing port is L_i, the intra-node forwarding delay is F_i, the remaining time of the mapped ongoing sending timeslot is T_i, the number of timeslots offset by the outgoing timeslot relative to the ongoing sending timeslot is o_i, especially on node P1 the timeslot length of UNI is L_h, then the end to end delay can be evaluted as follows (not including link propagation delay):

Best E2E Delay = sum(F_i+T_i+o_i*L_i, for 1<=i<=n) - L_n + F_e

Worst E2E Delay = sum(F_i+T_i+o_i*L_i, for 1<=i<=n) + L_h + F_e

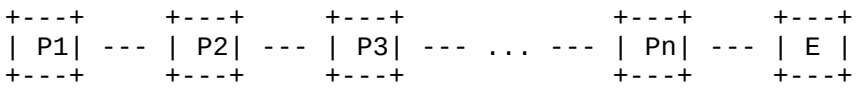


Figure 5: TQF Forwarding Path

The best E2E delay occurs when the flow arrived at the end of the ideal incoming timeslot and sent at the beginning of the outgoing timeslot of each node pi. The worst E2E delay occurs when the flow arrived at the beginning of the ideal incoming timeslot and sent at the end of the outgoing timeslot of each node Pi. The E2E delay jitter is (L_h + L_n).

Additional operation can be taken to further eliminate jitter. A latency deviation (E) can be included in the packet for this purpose. In order to let all packets of the same flow experience the same end-to-end delay that equals to the summary of the maximum arrival

deviation and the worst E2E delay, the flow entrance node can set E as maximum arrival deviation minus arrival deviation, and the flow exit node can update E as E plus departure deviation plus timeslot length, where departure deviation equals to the beginning of the outgoing timeslot minus the actual departure time from the scheduler. The departure deviation is also used to locate the right outgoing timeslot z in the case of in-time scheduling mode (refer to Section 10.2).

## 8. Flow States in Data-plane

The headend of the path needs to maintain the timeslot resource information with the granularity of sub-burst of each flow, so that each sub-burst of the DetNet flow can access the mapped timeslot resources. However, the intermediate node does not need to maintain states per flow, but only access the timeslot resources based on the timeslot id carried in the packets.

[I-D.pb-6man-deterministic-crh], [I-D.p-6man-deterministic-eh] defined methods to carry the stack of timeslot id in the IPv6 packets.

## 9. Queue Allocation Rule of Round Robin Queue

The buffer resources may be organized in the form of fixed number of round robin queues. In this case, only on-time scheduling mode is supported. In-time scheduling mode may cause urgent and non urgent packets to be stored in the same queue.

The number of round robin queues should be designed according to the number of timeslots included in the scheduling period. Each timeslot corresponds to a separate queue, in which the buffered packets must be able to be sent within a timeslot.

The length of the queue, i.e., the total number of bits that can be sent for a timeslot, equals to the allocated bandwidth of the corresponding TQF instance (see Section 12) multiplied by the timeslot length.

Figure 1 shows that the scheduling period actually instantiated on the data plane is not completely equivalent to the orchestration period. The scheduling period includes M timeslots (from 0 to M-1), while the orchestration period includes N timeslots (from 0 to N-1). N is an integer multiple of M. In the orchestration period, from timeslot 0 to M-1 is the first scheduling period, from timeslot M to slot 2M-1 is the second scheduling period, and so on. Therefore, it is necessary to convert the outgoing timeslot of the orchestration period to the target timeslot of the scheduling period, and insert the packet to the round robin queue corresponding to the target timeslot for transmission.

A simple conversion method is:

* target scheduling timeslot = outgoing timeslot % M

This is safe when o < M is always followed, where o is the number of offset timeslots between the outgoing timeslot z and the ongoing sending timeslot j (please refer to Section 7).

Next, a brief demonstration was provided that the sub-burst that arrives at the outgoing port during the ongoing sending timeslot (j) can be safely inserted into the corresponding queue in the scheduling period mapped by the outgoing timeslot z, and that queue will not overflow.

Assuming that each timeslot in the orchestration period has a virtual queue, for example, termed the virtual queue corresponding to the outgoing timeslot z as queue-z, the packets that can be inserted into queue-z may only come from the following flows:

During the ongoing sending timeslot j = (z-M+1+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = M-1.

During the ongoing sending timeslot j = (z-M+2+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = M-2.

... ...

During the ongoing sending timeslot j = (z-1+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = 1;

The total consumed burst resources of all these flows does not exceed the burst resource of the outgoing timeslot (z).

Then, when the ongoing sending timeslot changes to z, queue-z will be sent and cleared. In the following time, starting from timeslot z+1 to the last timeslot N-1, there are no longer any packets inserted into queue-z. Obviously, this virtual queue is a great waste of queue resources. In fact, queue-z can be reused by the subsequent outgoing timeslot (z+M)%N. Namely:

During the ongoing sending timeslot j = (z+1)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = M-1.

During the ongoing sending timeslot j = (z+2)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = M-2.

... ...

During the ongoing sending timeslot j = (z+M-1)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = 1.

The total consumed burst resources of all these flows does not exceed the burst resource of the outgoing timeslot (z+M)%N.

It can be seen that queue-z can be used by any outgoing timeslot (z+k*M)%N, where k is a non negative integer. By observing (z+k*M)%N, it can be seen that the minimum z satisfies 0<= z< M, that is, the entire orchestration period actually only requires M queues to store packets, which are the queues corresponding to M timeslots in the scheduling period. That is to say, the minimum z is the timeslot id in the scheduling period, while the outgoing timeslot (z+k*M)% N is the timeslot id in the orchestration period. The latter obtains the former by moduling M, which can then access the queue corresponding to the former. In short, the reason why a queue can store packets from multiple outgoing timeslots without being overflowed is that the packets stored in the queue earlier (more than M timeslots ago) have already been sent.

For example, if the total length of all queues supported by the hardware is 4G bytes, the queue length corresponding to a timeslot of 10us at a port rate of 100G bps is 1M bits, then a maximum of 32K timeslot queues can be provided, and TQF scheduler can use some of the queue resources, e.g., M may be 1K queues to construct a scheduling periold with 10 ms, and the corresponding orchestration period may be several 10ms.

## 10. Queue Allocation Rule of PIFO Queue

The buffer resources may also be organized in the form of PIFO queue. In this case, both in-time and on-time scheduling mode can be easily supported, because packets with different rank always ensure scheduling order.

### 10.1. PIFO with On-time Scheduling Mode

In the case of on-time mode, the buffer cost of PIFO queue is the same as that of round robin queues. It can directly use the begin time of the outgoing timeslot z as the rank of the packet and insert the packet into the PIFO for transmission.

* rank = z.begin

Here, the outgoing timeslot z refers to the outgoing timeslot z that is after the arrival time at the scheduler and closest to the arrival time.

The rule of the on-time scheduling mode is that if the PIFO is not empty and the rank of the head of queue is equal to or earlier than the current system time, the head of queue will be sent; otherwise, not.

### 10.2. PIFO with In-time Scheduling Mode

In the case of in-time mode, the buffer cost of PIFO queue is generally larger than that of on-time mode due to burst accumulation. [SP-LATENCY] provides guidance for evaluating excess buffer requirements.

Similar to Section 10.1, it can directly use the begin time of the outgoing timeslot z as the rank of the packet and insert the packet into the PIFO for transmission. However, due to in-time scheduling behavior, the expected outgoing timeslot z may not be the current outgoing timeslot z that is closest to and after the arrival time at the scheduler, since there are repeated z in different rounds of orchestration period. The expected outgoing timeslot z may be far away from the arrival time.

A departure deviation may be carried in the packet to help locate the expected outgoing timeslot z.

On the headend node:

* Match the ideal incoming timeslot i and outgoing timeslot z based on the regulated time.

* rank = z.begin

* When the packet leaves the headend, the departure deviation is set to z.begin minus the actual departure time from the scheduler. The departure deviation is carried in the sending packet.

On the intermediate node:

* Obtain departure deviation from the received packet.

* Use the result of the arrival time plus the departure deviation as the ideal arrival time, to locate the expected outgoing timeslot z that is after and closest to this result.

* rank = z.begin

* When the packet leaves the intermediated node, the departure deviation is updated to z.begin minus the actual departure time from the scheduler. The updated departure deviation is carried in the sending packet.

The rule of the in-time scheduling mode is that as long as the PIFO is not empty, packets are always obtained from the head of queue for transmission.

In summary, the in-time scheduling with the help of departure deviation, can suffer from the uncertainty caused by burst accumulation, and it is recommended only deployed in small networks, i.e., a limited domain with a small number of hops, where the burst accumulation issue is not serious; The on-time scheduling is recommended to be used in large networks.

## 11. Global Timeslot ID

The outgoing timeslots discussed in the previous sections are local timeslots style for all nodes. This section discusses the situation based on global timeslot style.

Global timeslot style refers to that all nodes in the path are identified with the same timeslot id, which of course requires all nodes to use the same timeslot length. There is no need to establish FTM for the DetNet flow on each node or carry FTM in packets. The packet only needs to carry the unique global timeslot id. However, the disadvantage is that the latency performance of the path may be large, which depends on BOM between the adjacent nodes. Another disadvantage is that the success rate of finding a path that matches the service requirements is not as high as local timeslot style.

Global timeslot style requires that the orchestration period is equal to the scheduling period, mainly considering that arrival packets with any global timeslot id can be successfully inserted into the corresponding queue without overflow. However, as the ideal design goal is to keep the scheduling period less than the orchestration period, further research is needed on other methods (such as basically aligning orchestration period between nodes), to ensure that packets with any global timeslot id can queue normally when the scheduling period is less than the orchestration period.

Compared to the local timeslot style, global timeslot style means that the incoming timeslot i must map to the outgoing timeslot i too.

As the example shown in Figure 6, each orchestration period contains 6 timeslots. Node V has three connected upstream nodes U1, U2, and U3. During each hop forwarding, the packet accesses the outgoing timeslot corresponding to the global timeslot id and forwards to the downstream node with the global timeslot id unchanged. For example, U1 sends some packets with global slot-id 0, termed as g0, in the outgoing timeslot 0. The packets with other global slot-id 1~5 are similarly termed as g1~g5 respectively. The figure shows the scheduling results of these 6 batches of packets sent by upstream nodes when node V continues to send them.

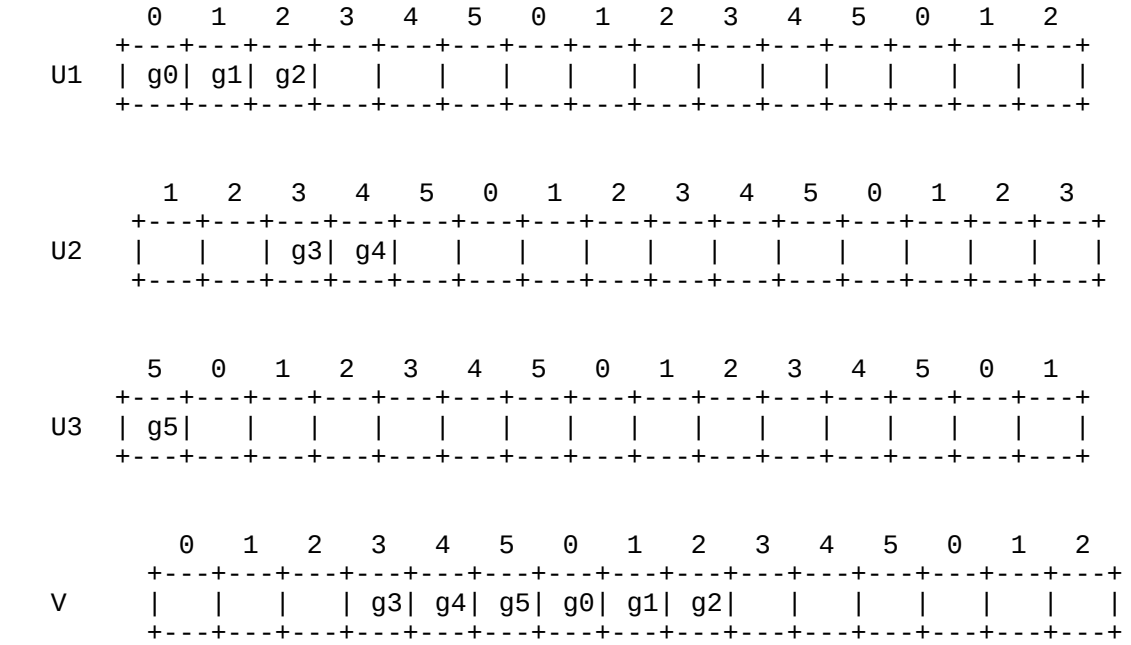

Figure 6: Global Timeslot Style Example

In this example:

* BTM between the outgoing timeslot of U1 and the ongoing sending timeslot of V is i -> i, so the global outgoing timeslot for the incoming timeslot i is i+6 (i.e., belongs to next round of orchestration periold).

* BTM between the outgoing timeslot of U2 and the ongoing sending timeslot of V is i -> i-1, so the global outgoing timeslot for the incoming timeslot i is i (i.e., belongs to current round of orchestration periold).

* BTM between the outgoing timeslot from U3 and the ongoing sending timeslot of V is i -> i+1, so the global outgoing timeslot for the incoming timeslot i is i+6 (i.e., belongs to next round of orchestration periold).

It can be seen that packets from U1 and U3 has large residency delay in the node V, while packets from U2 has small residency delay in the node V.

It should be noted that if round robin queue is used, for the orginal BTM i -> i (example of U1), or i -> i+1 (example of U3), the packets need to be stored in a buffer prior to the TQF scheduler (such as the buffer on the input port side) for a fixed latency (such as serveral timeslots) and then released to the scheduler. Otherwise, directly inserting the queue may cause jitter, i.e., part of the packets belonging to the same incoming timeslot i can be sent in the outgoing timeslot i, while the other part of the packets has to be delayed to be sent in the next round of timeslot i. This fixed-latency buffer is only introduced for specific upstream nodes. It can be determined according to the initial detection result of BTM between the adjacent nodes. If the original BTM is i -> i or i -> i+1, it needed, otherwise not. After the introduction of fixed-latency buffer, the new detection result of BTM will no longer be i -> i, or i -> i+1.

If PIFO queue is used, there is no need to introduce a fixed-latency buffer because in this case, rank = i.begin + OPL, and it will not be scheduled to be sent in the current outgoing timeslot i, but in the next round. However, in this case the PIFO itself serves as a fixed-latency buffer.

For the headend, the residence delay is similar to Section 7.1. For a flow which has the ideal incoming timeslot i, it may select a global outgoing timeslot z based on the BTM i -> j, where, j is the ongoing sending timeslot of the ougtoing port, and the timeslot offset o equals (N+z-j)%N.

For transit nodes, the residence delay is similar to Section 7.2, in addition to considering possible latency contributed by the above fixed-latency buffer. For a flow with the global incoming timeslot z, it still select the global outgoing timeslot z based on the BTM z -> j, where, j is the ongoing sending timeslot of the ougtoing port, and the timeslot offset o equals (N+z-j)%N.

The end-to-end delay equation is similar to Section 7.4, in addition to considering possible cumulated latency contributed by the above fixed-latency buffer.

## 12. Multiple Orchestration Periods

A single orchestration period may not be able to cover a wide range of service needs, such as some with a burst interval of microseconds, while others have a burst interval of minutes or even larger. When using a single orchestration period to simultaneously serve these services, the timeslot length may be microseconds, resulting in the need to include a large number of timeslots in the orchestration period. The final result is a proportional increase in the buffer size required for the scheduling period.

Multiple orchestration periods each with different length may be provided by the network. A TQF enabled link can be configured with multiple TQF scheduling instances each corresponding to specific orchestration period length. For simplicity, the orchestration period length itself can be used to identify a specific instance.

For example, one orchestration period length is 300 us, termed as OPL-300us, which is the LCM of the burst interval of the set of flows served. Another orchestration period length is 100 ms, termed as OPL-100ms, which is the LCM of the burst interval of another set of flows served. Each orchestration period instance has its own timeslot length. The timeslot length of a long orchestration period instance should be longer than that of a short orchestration period instance, and the former is an integer multiple of the latter. But the long orchestration period itself may not necessarily be an integer multiple of the short orchestration period.

As shown in Figure 7, both link-a and link-b are configured with n orchestration period instances, with the corresponding orchestration period lengths OPL_1, OPL_2, ..., OPL_n in ascending order. For each

orchestration period length OPL_i, the dedicated bandwidth resource is BW_U_i for node U (or BW_V_i for node V), and the timeslot length is TL_U_i for node U (or TL_V_i for node V). For each TQF enabled link, the sum of dedicated bandwidth resources of all TQF scheduling instances must not exceed the total bandwidth of the link.

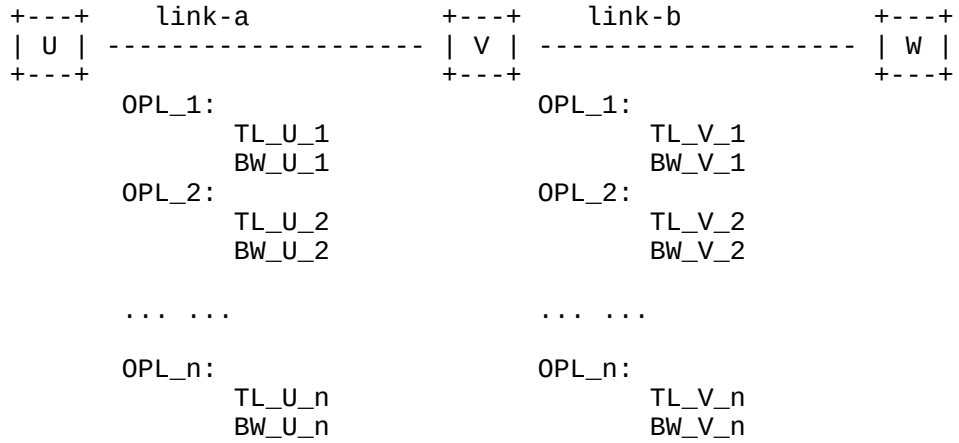


Figure 7: Multiple TQF Instances

Due to the fact that long orchestration periods serve DetNet flows with large burst intervals, for a given burst size, the larger the burst interval, the less bandwidth consumed by the DetNet flow. Therefore, it is recommended that the bandwidth resources of the long orchestration period is less than that of the short orchestration period, which is beneficial for reducing the buffer required for long orchestration period.

Interworking between different nodes is based on the same orchestration period. That means that the timeslot mapping described in Section 4 should be maintained in the context of the specific orchestration period. The orchestration period length should be carried in the forwarding packets to let the DetNet flow to consume the timeslot resources corresponding to the TQF scheduling instance.

If round robin queues are used, each TQF scheduling instance has its own separate queue set. Time division multiplexing scheduling is based on the granularity of the minimum timeslot length of all instances. Within each time unit of this granularity, the queues in the sending state of all instances are always scheduled in the order of OPL_1, OPL_2, ..., OPL_n.

If PIFO queue is used, all TQF scheduling instances may share a single PIFO queue. An implementation may use rank (i.e., the beginning of the outgoing timeslot z) plus timeslot length to determine the insertion order of two packets from different instances, so that the packet from the short orchstration period inserted at the front.

## 13. Admission Control on the Headend

On the network entry, traffic regulation must be performed on the incoming port, so that the DetNet flow does not exceed its T-SPEC such as burst interval, burst size, maximum packet size, etc. This kind of regulation is usually the shaping using leaky bucket combined with the incoming queue that receives DetNet flow. A DetNet flow may contain discrete multiple sub-bursts within its periodic burst interval. The leaky bucket depth should be larger than the maximum packet size, and should be consistent with the reserved burst resources required for the maximum sub-burst.

The scheduling mechanism described in this document has a requirement on the arrival time of DetNet flows on the network entry. It is expected that the distribution of sub-bursts (after regulation) of the DetNet flow will always appear in an ideal incoming timeslot of UNI port. However, a maximum arrival deviation is permitted. Based on the ideal incoming timeslot, an ideal outgoing timeslot is selected. For a single DetNet flow, the network entry may maintain multiple forwarding states each containing <ideal incoming timeslot, ideal outgoing timeslot>, due to many sub-bursts within the service burst interval. It is possible to set different maximum arrival deviation for different sub-bursts of the same flow, to eliminate the jitter between them.

For example, the network entry may maintain up to 3 sub-burst forwarding states for a flow. Ideally, all packets of this flow are split into 3 sub-bursts after regulation, each sub-burst matching one of the states. Here, 3 is the maximum sub-bursts for this flow, and it does not always contain so many bursts within the burst interval during each communication.

The arrival deviation should not exceed o-1 for late arrival case, or M-o-1 for early arrival case, where o is the offset between the outgoing timeslot and ongoing sending timeslot as mentioned above. Intuitively, large o can tolerate large late arrival deviations, while small o (or large M even for large o) can tolerate large early arrival deviations.

Restricted arrival deviation is beneficial for the design goal that scheduling period is smaller than the orchestration period, and packets can always be successfully inserted into the scheduling queue (RR or PIFO) without overflow. An unrestricted arrival deviation may contain one or more scheduling periods, and therefore there is an overflow risk when inserting packets into the queue associated with the ideal outgoing timeslot z at the regulated time.

Otherwise, for randomly arriving DetNet flows, it can be supported by taking a large M (or even M = N) (option-1) to accommodate random arrival, or it can be supported by introducing an explicit buffer put before the scheduler on the network entry to let the arrival time always meet the arrival deviation limitation (option-2).

* Note that due to randomness of arrival time, the packet may just miss the scheduling (or arrive too earlier) and need to wait in the scheduling queue (in the case of option-1) or the explicit buffer (in the case of option-2) for the next orchestration period.

Note that the arrival deviation on the ingress node, as well as the departure deviation on the egress node, may cause runtime jitter during forwarding. A latency deviation (E) calculated by arrival deviation and departure deviation can be used to eliminate jitter at the network egress on demand. This will achieve zero jitter performance metrics.

## 14. Frequency Synchronization

The basic explanation for frequency synchronization is that the crystal frequency of the hardware is consistent, which enables all nodes in the network to be in the same inertial frame and have the same time lapse rate. This is a prerequisite for TQF mechanism. The related frequency synchronization mechanisms, such as IEEE 1588-2008 Precision Time Protocol (PTP) [IEEE-1588] and synchronous Ethernet (syncE) [syncE], are not within the scope of this document.

Sometimes, people also refer to the frequency asynchrony as the timeslot rotation frequency difference caused by different node configurations with different timeslot lengths. This document supports the interconnection between nodes with this type of frequency asynchrony.

## 15. Evaluations

### 15.1. Large Scaling Requirements Matching Degree

This section gives the evaluation results of the TQF mechanism based on the requirements that is defined in [I-D.ietf-detnet-scaling-requirements].

| Requirements | Evaluation | Notes |
|---|---|---|
| 3.1 Tolerate Time Asynchrony | Yes | No time sync needed, only need frequency sync (3.1.3). |
| 3.2 Support Large Single-hop Propagation Latency | Yes | The timeslot mapping covers any value of link propagation delay. |
| 3.3 Accommodate the Higher Link Speed | Partial | The higher the service rate, the more buffer needed for the same timeslot length. |
| 3.4 Be Scalable to the Large Number of Flows and Tolerate High Utilization | Yes | Multiple OPL instance, each for a set of serivce flows, without overprovision. Utilization may reach 100% link bandwidth. The unused bandwidth of the timeslot can be used by best-effot flows. Calculating paths is NP-hard. |
| 3.5 Tolerate Failures of Links or Nodes and Topology Changes | N/A | Independent of queueing mechanism. |
| 3.6 Prevent Flow Fluctuation | Yes | Flows are isolated from each other through timeslots. |
| 3.7 Be scalable to a Large Number of Hops with Complex Topology | Yes | E2E latency is liner with hops, from ultra-low to low latency by multiple OPL. E2E jitter is low by on-time mode. Calculating paths is NP-hard. |
| 3.8 Support Multi-Mechanisms in Single Domain and Multi-Domains | N/A | Independent of queueing mechanism. |

Figure 8: Evaluation for Large Scaling Requirements

### 15.2. Taxonomy Considerations

[I-D.ietf-detnet-dataplane-taxonomy] provides criteria for classifying data plane solutions.

For performance, the per hop latency dominant factor of TQF is the offset between incoming timeslot and outgoing timeslot that is asigned to the flow.

For functional characteristics, TQF is periodic, flow level for traffic granularity, and bouned for time bounds.

* Periodic: Periodicity of TQF contains two characteristics, the first is that there is a time period P (i.e., orchestration periold) containing multiple time slots, and the second is that a flow is assigned repeatly to a particular set of time slots in that time period P.

* Flow level: TQF enhances TAS by extending the number of queues for Scheduling Traffic (i.e., from 1 to N), and may allocate one queue (represented by timeslot id) to each flow or a flow aggregate. Customizing different timeslots for different flows will make them interleaved.

* Bounded: The transmission completion of a packet will be in the outgoing timeslot to which the packet belongs, i.e., after the beginning of the timeslot and before the end of the timeslot.

[I-D.ietf-detnet-dataplane-taxonomy] also specifies the suitable categories of solutions for DetNet. According to the above functional characteristics, TQF will map to flow level periodic bounded category.

### 15.3. Examples

This section describes the example of how the TQF mechanism supports DetNet flows with different latency requirements.

#### 15.3.1. Heavyweight Loading Example

This example observes the service scale and different latency bound supported by the TQF mechanism in the heavyweight loading case.

Figure 9 provides a typical reference topology that serves to represent or measure the multihop jitter and latency experience of a single "flow i" across N hops (in the figure, N=10). On each of the N outgoing interfaces (represented by circles in the figure), "flow i" has to compete with different flows (represented by different

symbols on each hop). Especially, the competed flows arrive simultaneously at multiple incoming ports, with the same starting time when measuring their respective residence time. The characteristic of this reference topology is that every link that "flow i" passes through may be a bottleneck link with 100% network utilization, causing "flow i" to achieve the worst-case latency on each hop.

As shown in Figure 9:

* Network transmission capacity: each link has rate 10 Gbps. Assuming the service rate of TQF scheduler allocate the total port bandwidth.

* TSpec of each flow, maybe:

  - burst size 1000 bits, SBI 1 ms, and average arrival rate 1 Mbps.

  - or, burst size 1000 bits, SBI 100 us, and average arrival rate 10 Mbps.

  - or, burst size 1000 bits, SBI 10 us, and average arrival rate 100 Mbps.

* RSpec of each flow, maybe:

  - E2E latency 100us, and E2E jitter less than 10us or 100us.

  - or, E2E latency 200us, and E2E jitter less than 20us or 200us.

  - or, E2E latency 300us, and E2E jitter less than 30us or 300us.

  - or, E2E latency 400us, and E2E jitter less than 40us or 400us.

  - or, E2E latency 500us, and E2E jitter less than 50us or 500us.

  - or, E2E latency 600us, and E2E jitter less than 60us or 600us.

  - or, E2E latency 700us, and E2E jitter less than 70us or 700us.

  - or, E2E latency 800us, and E2E jitter less than 80us or 800us.

  - or, E2E latency 900us, and E2E jitter less than 90us or 900us.

  - or, E2E latency 1000us, and E2E jitter less than 100us or 1ms.

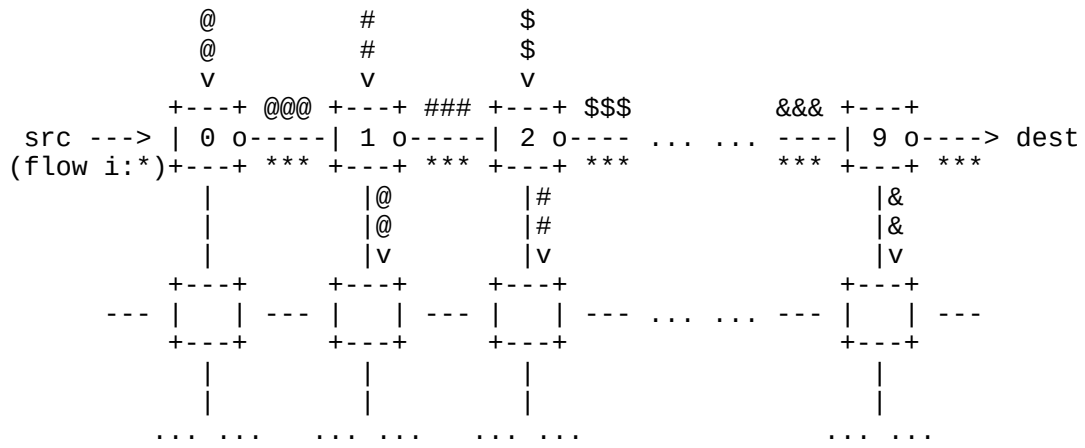


Figure 9: Heavyweight Loading Topology Example

For the observed flow i (marked with *), its TSpec and RSpec may be any of the above. Assuming that the path calculated by the controller for the flow i passes through 10 nodes (i.e., node 0~9). Especially, at each hop, flow i may conflict with other competitive flows, also with similar TSpec and RSpec as above, originated from other sources, e.g, competing with flow-set "@" at node 0, competing with flow-set "#" at node 1, etc.

For each link along the path, it may configure OPL-10ms instance with dedicated bandwidth 10 Gbps, containing 1000 timeslots each with length 10us. Assuming no link propagation delay and intra node forwarding delay, if flow i consumes outgoing timeslot by o=1, it can ensure an E2E latency of 100us (i.e., o * TL * 10 hops), and jitter of 20us(on-time mode) or 100us (in-time mode). The consumption by other o values is similar.

The table below shows the possible supported service scales. As flows arrived synchronously, the consumption of each timeslot in the orchestration period may be caused by any value of o. For example, if the ideal incoming timeslots of all flows are perfectly interleaved, then they can all consume timeslots by o=1 to get per-hop latency 10us, or all consume timeslots by o=2 to get per-hop latency 20us, etc. However, due to the fixed length of OPL, after all timeslot resources are exhausted by specific o value, it means that there are no timeslot resources used by other o values. Another example is that the ideal incoming timeslots of all flows are the same, then some of them consume timeslots by o=1, some consume timeslots by o=2, and so on. In either case, the total service scale is OPL * C / burst_size, that is composed of sum(s_i), where s_i is the service scale for o = i. The table provides the total scale and the average scale corresponding to each o value.

Note that in the table each row only shows the data where all flows served based on all o values have the same TSpec (e.g, in the first row, TSpec per flow is burst size 1000 bits and arrival rate 1 Mbps), while in reality, flows served based on different o values generally have different TSpec. It is easy to add rows to describe various combinations.

| | o=1 | o=2 | o=3 | o=4 | o=5 | o=6 | o=7 | o=8 | o=9 | o=10 |
|---|---|---|---|---|---|---|---|---|---|---|
| TSpec: 1000 bits SBI 1 ms | total = 10000 | | | | | | | | | |
| | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| TSpec: 1000 bits SBI 100 us | total = 1000 | | | | | | | | | |
| | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| TSpec: 1000 bits SBI 10 us | total = 100 | | | | | | | | | |
| | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

Figure 10: Timeslot Reservation and Service Scale Example

### 15.3.2. Lightweight Loading Examples

The following examples observe how the preset service scale is supported and mapped to different timeslots by the TQF mechanism in the lightweight loading case.

In these examples, the network only contains a small number of bottleneck links with low network utilization, and it can be considered as the lightweight loading case of Figure 9. Lightweight loading usually means having a smaller calculated worst-case latency per hop, or the actual latency experienced doesn't reach the worst-case latency.

#### 15.3.2.1. Grid Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes a grid topology (Figure 11) with partial mesh. Three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services. Among them, audio and CC flows consume less bandwidth (1.6 Mbps per flow and 0.48 Mbps per flow respectively) but both require lower E2E latency (5ms), while video flows consume more bandwidth (11 Mbps per flow) but can tolerate larger E2E latency (10ms).

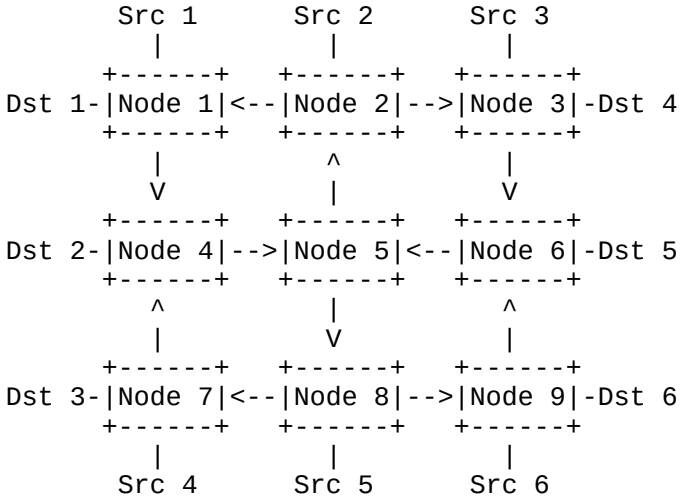


Figure 11: Lightweight Loading Topology Example

According to the preset rules that generate a unique route for every source and destination pair, the details of all paths are as follows:

```
Src1-1-Dst1
Src1-1-4-Dst2
Src1-1-4-5-8-7-Dst3
Src1-1-4-5-2-3-Dst4
Src1-1-4-5-8-9-6-Dst5
Src1-1-4-5-8-9-Dst6
```

```
Src2-2-1-Dst1
Src2-2-1-4-Dst2
Src2-2-3-6-5-8-7-Dst3
Src2-2-3-Dst4
Src2-2-3-6-Dst5
Src2-2-3-6-5-8-9-Dst6

Src3-3-6-5-2-1-Dst1
Src3-3-6-5-2-1-4-Dst2
Src3-3-6-5-8-7-Dst3
Src3-3-Dst4
Src3-3-6-Dst5
Src3-3-6-5-8-9-Dst6

Src4-7-4-5-2-1-Dst1
Src4-7-4-Dst2
Src4-7-Dst3
Src4-7-4-5-2-3-Dst4
Src4-7-4-5-8-9-6-Dst5
Src4-7-4-5-8-9-Dst6

Src5-8-7-4-5-2-1-Dst1
Src5-8-7-4-Dst2
Src5-8-7-Dst3
Src5-8-7-4-5-2-3-Dst4
Src5-8-9-6-Dst5
Src5-8-9-Dst6

Src6-9-6-5-2-1-Dst1
Src6-9-6-5-2-1-4-Dst2
Src6-9-6-5-8-7-Dst3
Src6-9-6-5-2-3-Dst4
Src6-9-6-Dst5
Src6-9-Dst6
```

Where, flows to destination Dst1 and Dst6 are audio flows, flows to destination Dst2 and Dst5 are CC flows, and flows to destination Dst3 and Dst4 are video flows. Each path carries 10 flows, e.g., the path "Src1-1-Dst1" carries 10 audio flows. It can be seen that the longest path contains 7 hops, and the bottleneck link involves link (2-3) and link (8-7), both of which have 10 audio flows, 60 video flows, and 10 CC flows.

Grid topology in this example only contains a small number of bottleneck links with low network utilization, and it can be considered as the lightweight loading case of Figure 9. Lightweight loading usually means having a smaller calculated worst-case latency per hop, or the actual latency experienced cannot reach the worst-case latency.

According to the longest path and the expected E2E latency, the per-hop latency bound for each type of flow can be estimated, i.e., 700us for audio and CC flows, 1400us for video flows. This means that the TQF mechanism needs to provide appropriate timeslots, and the offset between the incoming timeslots and outgoing timeslots mapped by flow audio, CC, and video, cannot be larger than 700us, 700us, 1400us, respectively.

A TQF instance with OPL-5ms may be configured on each link in the network. The reason for choosing OPL-5ms is that it is approximately the Least Common Multiple of the packet intervals of the three type of flows. In this example, the regulated packet interval (i.e., packet_size / service_rate) for flows audio, video, and CC are 1.25 ms, 1.1 ms, and 5 ms, respectively. So, within each OP, it contains 4 audio packets per flow, 5 video packets per flow, and 1 CC packet per flow. Taking an audio flow as an example, its four discrete packets will occupy four ideal positions in OP, and these packets can be denoted as four sub-bursts, namely sub-burst 1, 2, 3, and 4. Similarly, a CC flow will occupy one ideal position in OP, by sub-burst 1. And, a video flow will occupy five ideal position in OP, by sub-burst 1, 2, 3, 4, 5, respectively.

Considering the maximum packet size is 12000 bits (from video flow) and the link capacity is 1 Gbps, timeslot length 100us is selected to accommodate at least such a single packet. Intuitively, if the link capability is larger, such as 10 Gbps, the timeslot length can be chosen to be smaller, such as 10us.

Although the number of flows on different links varies, to simplify the description, the collection of various types of flows on bottleneck and non bottleneck links is taken as the number of flows on each link to calculate the timeslot resource requirements. So the load is slightly increased and it is assumed that the number of each type of flows on a link reached 60.

Figure 12 shows a possible timeslots mapped by flows on a link. Note that it assumes that the first sub-burst of all flows (i.e., 60 CC flows, 60 audio flows, and 60 video flows) arrive concurrently at the same time T0 (nearly before timeslot #0) and some packets have to experience larger per-hop latency than others, and this is the worst case. In fact, all flows may arrive in different timeslots

interleaved, so that the consumed outgoing timeslots are also naturally interleaved and each packet may experience smaller per-hop latency (e.g., one timeslot length).

| Slot | Bursts | Services Mapped |
|---|---|---|
| #0 | 100 Kbits | 40 CC flows(sub-burst 1), 96 Kbits<br>2 audio flows(sub-burst 1), 4 Kbits |
| #1 | 100 Kbits | 20 CC flows(sub-burst 1), 48 Kbits<br>26 audio flows(sub-burst 1), 52 Kbits |
| #2 | 100 Kbits | 26 audio flows(sub-burst 1), 52 Kbits<br>4 video flows(sub-burst 1), 48 Kbits |
| #3 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>2 audio flows(sub-burst 1), 4 Kbits |
| #4 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>2 audio flows(sub-burst 1), 4 Kbits |
| #5 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>2 audio flows(sub-burst 1), 4 Kbits |
| #6 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>remaining 4 Kbits |
| #7 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>remaining 4 Kbits |
| #8 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>remaining 4 Kbits |
| #9 | 100 Kbits | 8 video flows(sub-burst 1), 96 Kbits<br>remaining 4 Kbits |
| #10 | 100 Kbits | |
| #11 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #12 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #13 | 100 Kbits | 50 audio flows(sub-burst 2), 100K bits |

| | | |
|---|---|---|
| #14 | 100 Kbits | 10 audio flows(sub-burst 2), 20 Kbits<br>4 video flows(sub-burst 2), 48 Kbits<br>remaining 32 Kbits |
| #15 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #16 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #17 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #18 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #19 | 100 Kbits | 8 video flows(sub-burst 2), 96 Kbits<br>remaining 4 Kbits |
| #20 | 100 Kbits | |
| #21 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #22 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #23 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #24 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #25 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #26 | 100 Kbits | 50 audio flows(sub-burst 2), 100 Kbits |
| #27 | 100 Kbits | 10 audio flows(sub-burst 3), 20 Kbits<br>4 video flows(sub-burst 3), 48 Kbits<br>remaining 32 Kbits |
| #28 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |
| #29 | 100 Kbits | 8 video flows(sub-burst 3), 96 Kbits<br>remaining 4 Kbits |

| #30 | 100 Kbits | |
|---|---|---|
| #31 | 100 Kbits | 4 video flows(sub-burst 4), 48 Kbits<br>remaining 52Kbits |
| #32 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #33 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #34 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #35 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #36 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #37 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #38 | 100 Kbits | 8 video flows(sub-burst 4), 96 Kbits<br>remaining 4 Kbits |
| #39 | 100 Kbits | 50 audio flows(sub-burst 4), 100 Kbits |
| #40 | 100 Kbits | 10 audio flows(sub-burst 4), 20 Kbits<br>remaining 80 Kbits |
| #41 | 100 Kbits | |
| #42 | 100 Kbits | 4 video flows(sub-burst 5), 48 Kbits<br>remaining 52Kbits |
| #43 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits<br>remaining 4 Kbits |
| #44 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits<br>remaining 4 Kbits |
| #45 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits<br>remaining 4 Kbits |
| #46 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits<br>remaining 4 Kbits |

| #47 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits |
| | | remaining 4 Kbits |
| #48 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits |
| | | remaining 4 Kbits |
| #49 | 100 Kbits | 8 video flows(sub-burst 5), 96 Kbits |
| | | remaining 4 Kbits |

Figure 12: Timeslot Resource Pool and Service Mapped in Grid Topology

Each CC flow contributes only one sub-burst within the OP:

* The first sub-bursts of all 60 CC flows, totaling 144 Kbits, arrived nearly before timeslot #0, consume timeslots #0, #1. The worst-case per-hop latecy for the last packet sent is 200 us.

Each audio flow contributes four sub-burst within the OP:

* The first sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #0, consume timeslots #0, #1, #2, #3, #4, #5. The worst-case per-hop latecy for the last packet sent is 600 us.

* The second sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #13, consume timeslot #13, #14. The worst-case per-hop latecy for the last packet sent is 200 us.

* The third sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #26, consume timeslots #26, #27. The worst-case per-hop latecy for the last packet sent is 200 us.

* The fourth sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #39, consume timeslots #39, #40. The worst-case per-hop latecy for the last packet sent is 200 us.

Each video flow contributes five sub-burst within the OP:

* The first sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #0, consume timeslots #2, #3, #4, #5, #6, #7, #8, #9. The worst-case per-hop latecy for the last packet sent is 1000 us.

* The second sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #11, consume timeslots #11, #12, #14, #15, #16, #17, #18, #19. The worst-case per-hop latecy for the last packet sent is 900 us.

* The third sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #21, consume timeslots #21, #22, #23, #24, #25, #27, #28, #29. The worst-case per-hop latecy for the last packet sent is 900 us.

* The fourth sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #31, consume timeslots #31, #32, #33, #34, #35, #36, #37, #38. The worst-case per-hop latecy for the last packet sent is 900 us.

* The fifth sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #41, consume timeslots #42, #43, #44, #45, #46, #47, #48, #49. The worst-case per-hop latecy for the last packet sent is 900 us.

NOTE:

* In the above process of resource allocation, the 10 flows carried on each path are individually allocated burst resources. This is the most general case, that is, although the 10 flows share the same path, they are assumed to be independent of each other. However, in some cases, if these 10 flows are treated as a macro flow and policing is executed at the network entrance node for the macro flow (the leaky bucket depth is still the maximum packet size, but the leack bucket rate is the aggregation rate), and resources are reserved for the macro flow instead of the member flow, then it is still necessary to allocate timeslot resources for the same total amounts of bursts, which will be more evenly distributed (i.e., more interleaved) within the OP. For example, a macro CC flow (including 10 CC member flows) contribute 10 sub-bursts within the OP (5 ms), and each sub-burst size is 2400 bits.

* Video flows have 30 back-to-back packets per single burst, and are being regulated on the flow entrance node, to support 60 video flows on each link. Operators may increase the bucket depth for video flows to make the shaped pattern and the original arrival pattern as consistent as possible, but this will be harmful to service scale. There is a tradeoff between burstiness, policing, and service scale.

#### 15.3.2.2. Ring-Mesh Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes another hierarchical Ring-Mesh topology (Figure 13), where, node 1~9 are core routers, and each leaf group consists of 10 Ring networks. Each Ring network (Figure 14) has 8 nodes, with one node connected to the core by a separate inter-domain link.

The capacity of all the links in the core network is 10 Gbps. The capacity of all the links in the leaf network, including the inter-domain link, is 1 Gbps.

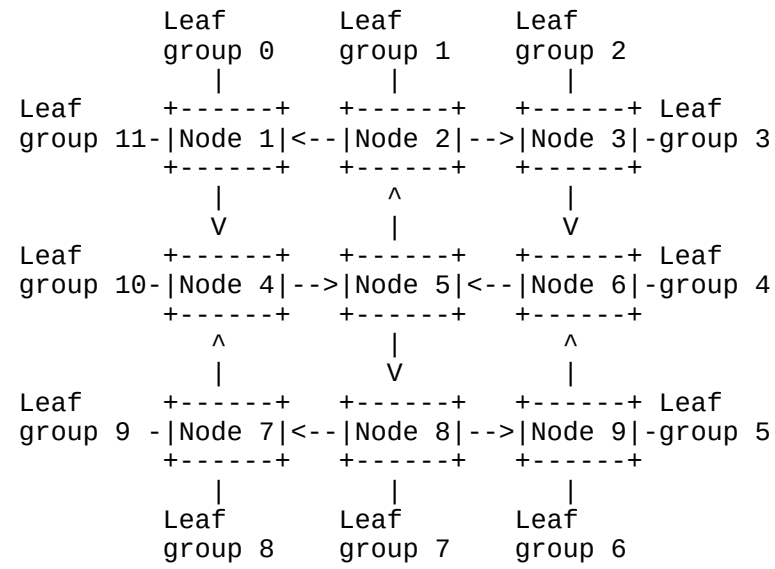


Figure 13: Hierarchical Ring-Mesh Topology

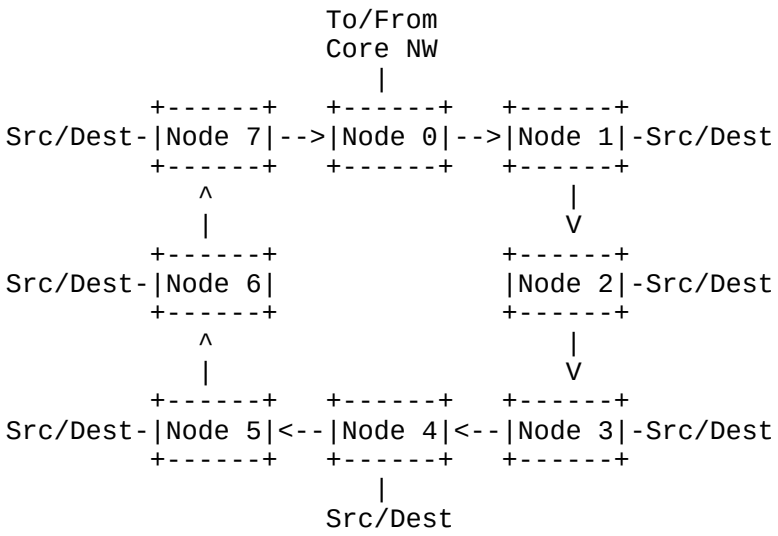


Figure 14: Ring Network

Again, three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services, whose TSpec and RSpec are consistent with the above Grid example.

A flow-set is defined that includes 7 audio flows, 7 video flows, and 32 CC flows, all of which share the same DetNet path. For example, node 1 in the source ring may send a flow-set to node 7 in the destination ring, and the DetNet path may be, 1-2-3-4-5-6-7-0 (source ring), inter-domain link, core, inter-domain link, 0-1-2-3-4-5-6-7 (destination ring).

The longest DetNet path may be 20 hops, where, 7 hops in the source ring, 7 hops in the destination ring, 2 inter-domain hops, and 4 hops in the core.

The preset routing of DetNet path is that, every flow-set in a ring network travels from node i to node (i+7)mod8, and each leaf group i sends n flow-sets (e.g., n = 10, if a leaf group contains 10 ring networks) to the leaf group (i+6)mod12.

Take a flow-set from the source ring to the destination ring as the observed flow-set. The observed flow-set will compete with other 6 flow-sets in the ring, and compete with more flow-sets (coming from other leaf groups) in the core. Note that there is no competition on the inter-domain link.

In this example, it is no longer assumes that every packet of all flow-sets, including the observed flow-set and the competed flow-sets, arrives simultaneously. Although assuming extremely high concurrency can accommodate any topology with some actual concurrency, it underestimate the service scale that can be admitted. In fact, in the ring network, the concurrency at each hop is that only two input interfaces compete for one output interface. For inter-domain links, concurrency is even zero. In the core network, concurrency is also limited. By utilizing the knowledge of concurrency, more reasonable allocation of slots can be applied for all flows.

A TQF instance OPL-5ms with timeslot length 100us is configured on each link in both ring and core networks.

In the ring network, a bad flow interleaving is that each node generates a flow-set at the same time (e.g., nearly before timeslot #0). Assuming that each node prioritizes obtaining a smaller timeslot offset for locally generated flow-set, i.e., each node allocates timeslots for the local flow-set starting from #0, while allocates later timeslots for the flow-sets received from upstream nodes. When a local flow-set, e.g., f1 generated by Node 1, is sent

to the downstrem Node 2, it will conflict with another local flow-set, e.g., f2 generated by Node 2, and has to be assigned a later timeslot (i.e., obtaining a larger timeslot offset). Since that f1 is assigned a later outgoing timeslot on Node 2 means that the incoming timeslot on Node 3 is also later, it does not cause f1 to face more queuing delay on node 3 (and also other downstream nodes). Therefore, as the number of hops increases, a perfect flow interleaving is formed. For example, on Node 1, the interleaved flow-sets received may be {f0, f7, f6, f5, f4, f3, f2}, among which f0 (generated by Node 0) arrived the earliest and f2 (generated by Node 2) arrived the latest. Note that on Node 1, f2 will be terminated and not compete for outgoing port.

Figure 15 shows a possible timeslots mapped by flows on link-1-2 in the ring domain. Other links are similar to link-1-2. For a flow-set, try to allocate smaller timeslot offsets for CC, then audio, and at last video, as video flows have loose latency requirements. For simplicity, assuming no link propagation delay and intra node forwarding delay.

| Slot | Bursts | Services Mapped |
|---|---|---|
| #0 | 100 Kbits | 32 CC flows of f1<br>9 CC flows of f0 |
| #1 | 100 Kbits | 23 CC flows of f0<br>18 CC flows of f7 |
| #2 | 100 Kbits | 14 CC flows of f7<br>27 CC flows of f6 |
| #3 | 100 Kbits | 5 CC flows of f6<br>32 CC flows of f5<br>4 CC flows of f4 |
| #4 | 100 Kbits | 28 CC flows of f4<br>13 CC flows of f3 |
| #5 | 100 Kbits | 19 CC flows of f3<br>each 7 audio flows of f1, f0, f7 (round 1)<br>6 audio flows of f6 (round 1) |
| #6 | 100 Kbits | 1 audio flows of f6 (round 1)<br>each 7 audio flows of f5, f4, f3 (round 1)<br>4 video flows of f1 (round 1) |

| | | |
|---|---|---|
| #7 | 100 Kbits | 3 video flows of f1 (round 1)<br>5 video flows of f0 (round 1) |
| #8 | 100 Kbits | 2 video flows of f0 (round 1)<br>6 video flows of f7 (round 1) |
| #9 | 100 Kbits | 1 video flows of f7 (round 1)<br>7 video flows of f6 (round 1) |
| #10 | 100 Kbits | 7 video flows of f5 (round 1)<br>1 video flows of f4 (round 1) |
| #11 | 100 Kbits | 6 video flows of f4 (round 1)<br>2 video flows of f3 (round 1) |
| #12 | 100 Kbits | 5 video flows of f3 (round 1)<br>3 video flows of f1 (round 2) |
| #13 | 100 Kbits | each 7 audio flows of f1, f0, f7~f3 (round 2) |
| #14 | 100 Kbits | 4 video flows of f1 (round 2)<br>4 video flows of f0 (round 2) |
| #15 | 100 Kbits | 3 video flows of f0 (round 2)<br>5 video flows of f7 (round 2) |
| #16 | 100 Kbits | 2 video flows of f7 (round 2)<br>6 video flows of f6 (round 2) |
| #17 | 100 Kbits | 1 video flows of f6 (round 2)<br>7 video flows of f5 (round 2) |
| #18 | 100 Kbits | 7 video flows of f4 (round 2)<br>1 video flows of f3 (round 2) |
| #19 | 100 Kbits | 6 video flows of f3 (round 2) |
| #20 | 100 Kbits | |
| #21 | 100 Kbits | 7 video flows of f1 (round 3)<br>1 video flows of f0 (round 3) |
| #22 | 100 Kbits | 6 video flows of f0 (round 3)<br>2 video flows of f7 (round 3) |
| #23 | 100 Kbits | 5 video flows of f7 (round 3)<br>3 video flows of f6 (round 3) |

| | | |
|---|---|---|
| #24 | 100 Kbits | 4 video flows of f6 (round 3)<br>4 video flows of f5 (round 3) |
| #25 | 100 Kbits | 3 video flows of f5 (round 3)<br>5 video flows of f4 (round 3) |
| #26 | 100 Kbits | each 7 audio flows of f1, f0, f7~f3 (round 3) |
| #27 | 100 Kbits | 2 video flows of f4 (round 3)<br>6 video flows of f3 (round 3) |
| #28 | 100 Kbits | 1 video flows of f3 (round 3) |
| #29 | 100 Kbits | |
| #30 | 100 Kbits | |
| #31 | 100 Kbits | 7 video flows of f1 (round 4)<br>1 video flows of f0 (round 4) |
| #32 | 100 Kbits | 6 video flows of f0 (round 4)<br>2 video flows of f7 (round 4) |
| #33 | 100 Kbits | 5 video flows of f7 (round 4)<br>3 video flows of f6 (round 4) |
| #34 | 100 Kbits | 4 video flows of f6 (round 4)<br>4 video flows of f5 (round 4) |
| #35 | 100 Kbits | 3 video flows of f5 (round 4)<br>5 video flows of f4 (round 4) |
| #36 | 100 Kbits | 2 video flows of f4 (round 4)<br>6 video flows of f3 (round 4) |
| #37 | 100 Kbits | 1 video flows of f3 (round 4) |
| #38 | 100 Kbits | |
| #39 | 100 Kbits | each 7 audio flows of f1, f0, f7~f3 (round 4) |
| #40 | 100 Kbits | |
| #41 | 100 Kbits | 7 video flows of f1 (round 5)<br>1 video flows of f0 (round 5) |
| #42 | 100 Kbits | 6 video flows of f0 (round 5)<br>2 video flows of f7 (round 5) |

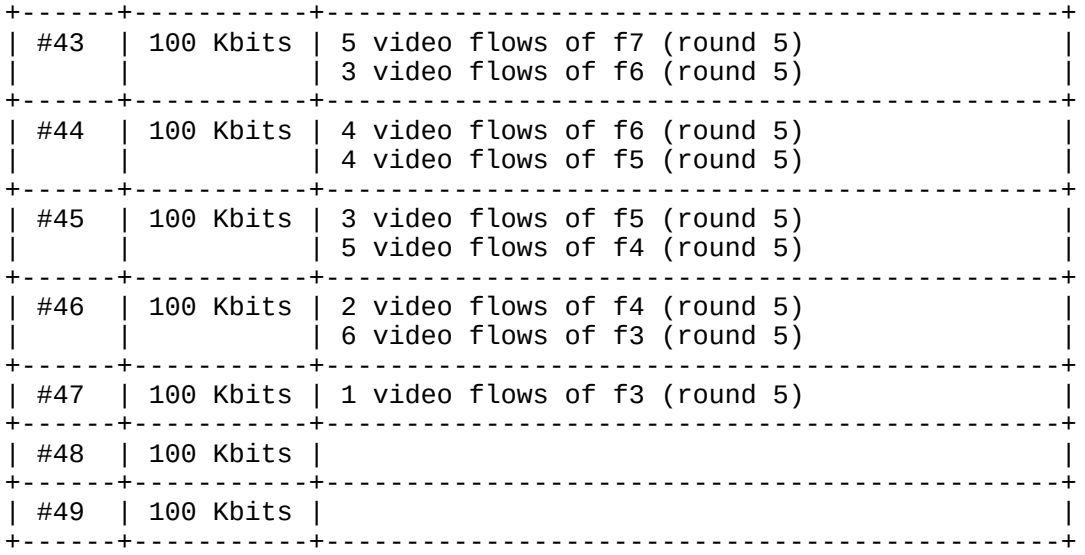


Figure 15: Timeslot Resource Pool and Service Mapped in Ring Domain

Each CC flow contributes only one round within the OP:

* The first round of all 7*32 CC flows, totaling 537.6 Kbits, arrive sequentially in the duration from timeslot #0 to #5, consuming timeslots #0 to #5 respectively. The worst-case per-hop latecy is 100 us.

Each audio flow contributes four sub-burst within the OP:

* The first round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration from timeslot #5 to #6, except that the local generated 7 audio flows arrive at timeslot #0, consuming timeslots #5 to #6 respectively. The worst-case per-hop latecy is 500 us for the local generated 7 audio flows, and 100 us for other audio flows.

* The second round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #13, consuming timeslots #13 respectively. The worst-case per-hop latecy is 100 us.

* The third round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #26, consuming timeslots #26 respectively. The worst-case per-hop latecy is 100 us.

* The fourth round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #39, consuming timeslots #39 respectively. The worst-case per-hop latecy is 100 us.

Each video flow contributes five sub-burst within the OP:

* The first round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #6 to #12, except that the local generated 7 video flows arrive at timeslot #0, consuming timeslots #6 to #12 respectively. The worst-case per-hop latecy is 600 us for the local generated 7 video flows, and 200 us for other audio flows.

* The second round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #12 to #19, except that the local generated 7 video flows arrive at timeslot #11, consuming timeslots #12 to #19 respectively. The worst-case per-hop latecy is 300 us for the local generated 7 video flows, and 200 us for other audio flows.

* The third round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #21 to #27, consuming timeslots #21 to #27 respectively. The worst-case per-hop latecy is 200 us.

* The fourth round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #31 to #37, consuming timeslots #31 to #37 respectively. The worst-case per-hop latecy is 200 us.

* The fifth round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #41 to #47, consuming timeslots #41 to #47 respectively. The worst-case per-hop latecy is 200 us.

In the core network, the details of all DetNet paths are as follows:

```
group0 -1-4-5-8-9- group6
group1 -2-1-4-5-8- group7
group2 -3-6-5-8-7- group8
group3 -3-6-5-8-7- group9
group4 -6-5-2-1-4- group10
group5 -9-6-5-2-1- group11
group6 -9-6-5-2-1- group0
group7 -8-7-4-5-2- group1
group8 -7-4-5-2-3- group2
group9 -7-4-5-2-3- group3
group10 -4-5-2-3-6- group4
group11 -1-4-5-8-9- group5
```

Where, the bottleneck link-4-5 will carry 70 flow-sets, in which, 10 flow-sets each from separate inter-domain link, 30 flow-sets from node 1, and 30 flow-sets from node 7.

Another bottleneck link-5-2 will also carry 70 flow-sets,in which, 30 flow-set from node 6, and 40 flow-sets from node 4.

On the bottleneck link-4-5, a bad flow interleaving is that there are 12 bursts competing for the outgoing interface. Their sizes are 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 30, and 30 flow-sets respectively. Assuming that each incoming flow-set arrives nearly before timeslot #0. A simple timeslot resource allocation method similar to Section 15.3.2.1 can be used (assuming extreme concurrency), see Figure 16. In fact, all flows may arrive in different timeslots interleaved, so that the consumed outgoing timeslots are also naturally interleaved and each packet may experience smaller per-hop latency (e.g., one timeslot length), that is similar to the method in ring domain.

| Slot | Bursts | Services Mapped |
|---|---|---|
| #0 | 1 Mbits | 416 CC flows |
| #1 | 1 Mbits | 416 CC flows |
| #2 | 1 Mbits | 416 CC flows |
| #3 | 1 Mbits | 416 CC flows |
| #4 | 1 Mbits | 416 CC flows |
| #5 | 1 Mbits | 160 CC flows<br>308 audio flows (round 1) |

| | | |
|---|---|---|
| #6 | 1 Mbits | 182 audio flows (round 1)<br>53 video flows (round 1) |
| #7 | 1 Mbits | 83 video flows (round 1) |
| #8 | 1 Mbits | 83 video flows (round 1) |
| #9 | 1 Mbits | 83 video flows (round 1) |
| #10 | 1 Mbits | 83 video flows (round 1) |
| #11 | 1 Mbits | 83 video flows (round 1) |
| #12 | 1 Mbits | 22 video flows (round 1)<br>61 video flows (round 2) |
| #13 | 1 Mbits | 490 audio flows (round 2) |
| #14 | 1 Mbits | 83 video flows (round 2) |
| #15 | 1 Mbits | 83 video flows (round 2) |
| #16 | 1 Mbits | 83 video flows (round 2) |
| #17 | 1 Mbits | 83 video flows (round 2) |
| #18 | 1 Mbits | 83 video flows (round 2) |
| #19 | 1 Mbits | 14 video flows (round 2) |
| #20 | 1 Mbits | |
| #21 | 1 Mbits | 83 video flows (round 3) |
| #22 | 1 Mbits | 83 video flows (round 3) |
| #23 | 1 Mbits | 83 video flows (round 3) |
| #24 | 1 Mbits | 83 video flows (round 3) |
| #25 | 1 Mbits | 83 video flows (round 3) |
| #26 | 1 Mbits | 75 video flows (round 3)<br>50 audio flows (round 3) |
| #27 | 1 Mbits | 440 audio flows (round 3) |

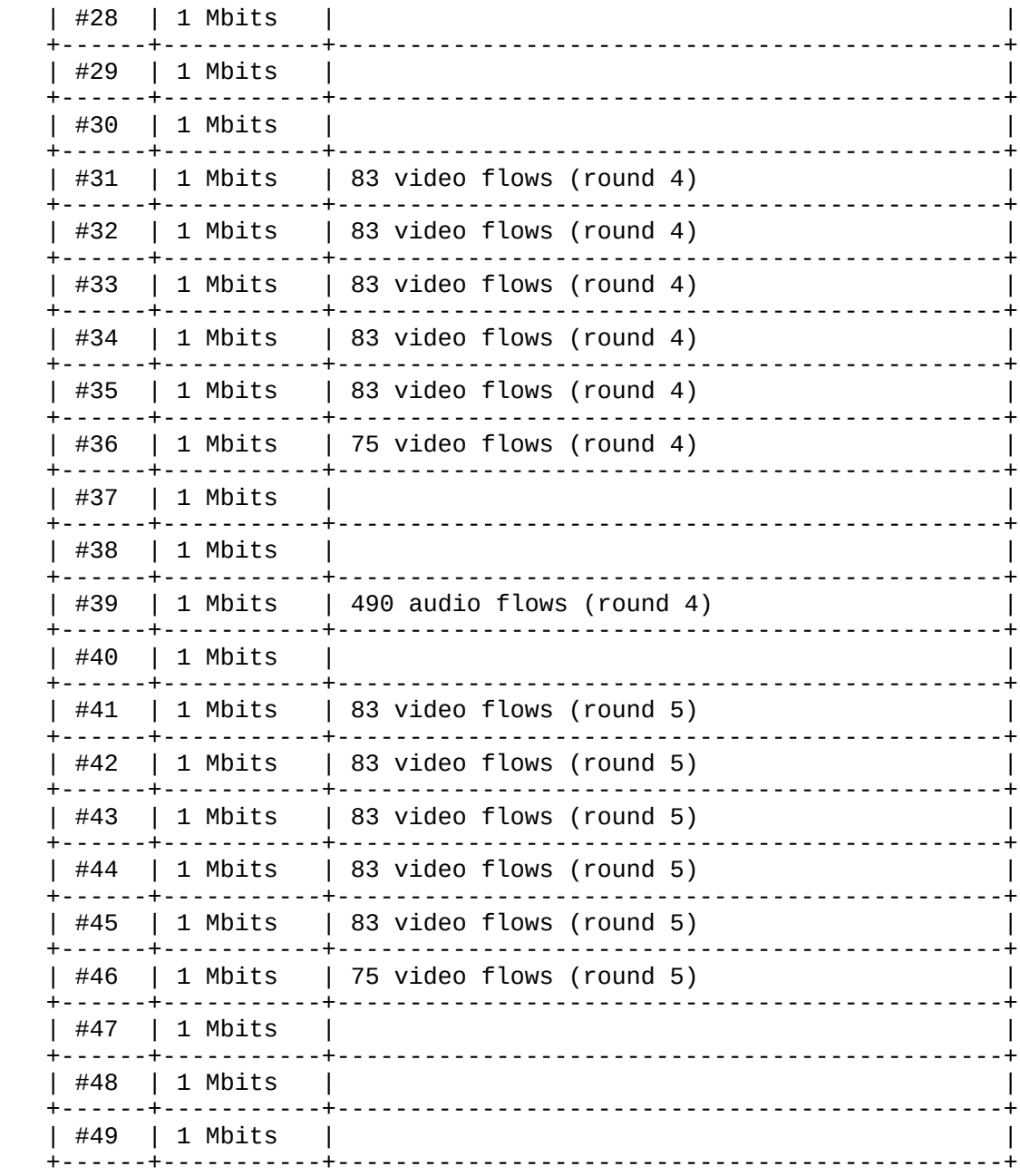


Figure 16: Timeslot Resource Pool and Service Mapped in Core Domain

According to the above table, the worst-case per-hop latecy is 600 us for CC, 700 us for audio, 1.3 ms for Video.

Other explanations are similar to the previous example of Grid reference topology.

## 16. IANA Considerations

There is no IANA requestion for this document.

## 17. Security Considerations

Security considerations for DetNet are described in detail in [RFC9055]. General security considerations for the DetNet architecture are described in [RFC8655]. Considerations specific to the DetNet data plane are summarized in [RFC8938].

Adequate admission control policies should be configured in the edge of the DetNet domain to control access to specific timeslot resources. Access to classification and mapping tables must be controlled to prevent misbehaviors, e.g, an unauthorized entity may modify the table to map traffic to an unallowed timeslot resource, and competes and interferes with normal traffic.

## 18. Acknowledgements

The authors appreciate David Black, Jinoo Joung, Toerless Eckert, Xuesong Geng, Florian Kauer, Bin Tan, Feng Liu and Xiangyang Zhu, for their insightful comments and productive discussion that helped to improve the document.

## 19. References

### 19.1. Normative References

[I-D.chen-detnet-sr-based-bounded-latency]
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>.

[I-D.eckert-detnet-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>.

[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.ietf-detnet-scaling-requirements]
Liu, P., Li, Y., Eckert, T. T., Xiong, Q., Ryoo, J., zhushiyin, and X. Geng, "Requirements for Scaling Deterministic Networks", Work in Progress, Internet-Draft, draft-ietf-detnet-scaling-requirements-09, 7 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-scaling-requirements-09>.

[I-D.p-6man-deterministic-eh]
Peng, S., "Deterministic Source Route Header", Work in Progress, Internet-Draft, draft-p-6man-deterministic-eh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-p-6man-deterministic-eh-01>.

[I-D.pb-6man-deterministic-crh]
Peng, S. and R. Bonica, "Deterministic Routing Header", Work in Progress, Internet-Draft, draft-pb-6man-deterministic-crh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-pb-6man-deterministic-crh-01>.

[I-D.peng-detnet-tqf-controller-plane]
Peng, S., Liu, P., Basu, K., Liu, A., Yang, D., and G. Peng, "Timeslot Queueing and Forwarding (TQF) Control Plane", Work in Progress, Internet-Draft, draft-peng-detnet-tqf-controller-plane-00, 20 June 2024, <https://datatracker.ietf.org/doc/html/draft-peng-detnet-tqf-controller-plane-00>.

[I-D.peng-lsr-deterministic-traffic-engineering]
Peng, S., "IGP Extensions for Deterministic Traffic Engineering", Work in Progress, Internet-Draft, draft-peng-lsr-deterministic-traffic-engineering-04, 8 January 2026, <https://datatracker.ietf.org/doc/html/draft-peng-lsr-deterministic-traffic-engineering-04>.

[I-D.xp-ippm-detnet-stamp]
Min, X., Peng, S., and X. hexiaoming, "STAMP Extensions for DetNet", Work in Progress, Internet-Draft, draft-xp-ippm-detnet-stamp-03, 11 December 2025, <https://datatracker.ietf.org/doc/html/draft-xp-ippm-detnet-stamp-03>.

[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>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

## 19.2. Informative References

[ATM-LATENCY]
"Bounded Latency Scheduling Scheme for ATM Cells", 1999, <https://ieeexplore.ieee.org/document/780828/>.

[CQF] "Cyclic queueing and Forwarding", 2017, <https://ieeexplore.ieee.org/document/7961303>.

[ECQF] "Enhancements to Cyclic Queuing and Forwarding", 2023, <https://1.ieee802.org/tsn/802-1qdv/>.

[IEEE-1588]
"IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008, <https://standards.ieee.org/findstds/standard/1588-2008.html>.

[SP-LATENCY] "Guaranteed Latency with SP", 2020, <https://ieeexplore.ieee.org/document/9249224>.

[syncE] "Timing and synchronization aspects in packet networks", 2013, <https://www.itu.int/rec/T-REC-G.8261>.

[TAS] "Time-Aware Shaper", 2015, <https://standards.ieee.org/ieee/802.1Qbv/6068/>.

## Authors' Addresses

Shaofu Peng
ZTE
China
Email: peng.shaofu@zte.com.cn

Peng Liu
China Mobile
China
Email: liupengyjy@chinamobile.com

Kashinath Basu
Oxford Brookes University
United Kingdom
Email: kbasu@brookes.ac.uk

Aihua Liu
ZTE
China
Email: liu.aihua@zte.com.cn

Dong Yang
Beijing Jiaotong University
China
Email: dyang@bjtu.edu.cn

Guoyu Peng
Beijing University of Posts and Telecommunications
China
Email: guoyupeng@bupt.edu.cn

Junfeng Zhao
CAICT
China
Email: zhaojunfeng@caict.ac.cn