Hierarchical Multi-resource Fair Queueing for Network Function Virtualization

Chaoqun You*
*University of Electronic Science and Technology of China

***Abstract*—As the volume of traffic flows surges, providing Quality-of-Service (QoS) guarantees to flows by fair queueing has never been more challenging in Network Function Virtualization (NFV). There has been a recent effort in both industry and academia to develop fair queueing algorithms across multiple resources in NFV. However, all existing works fail to support hierarchical scheduling, a crucial feature that also provides QoS guarantees to grouped flows on tenant boundaries. In this paper, we present two new multi-resource fair queueing algorithms that support hierarchies, collapsed Hierarchical Dominant Resource Fair Queueing (collapsed H-DRFQ) and dove-tailing H-DRFQ, both of which provide hierarchical share guarantees. Through formal analysis, we find that the dove-tailing H-DRFQ outperforms collapsed H-DRFQ by providing a smaller delay bound. However, according to the simulation results, both algorithms have their pros and cons. Dove-tailing H-DRFQ benefits to the flows with more complex hierarchies, while collapsed H-DRFQ is better for the flows with simpler attribution structures. Meanwhile, our simulation shows that both H-DRFQ algorithms can achieve near-perfect fairness.**

I. Introduction

Network functions (NFs), also known as middleboxes, are ubiquitous today [1], [2]. They are deployed to perform various processing functions, ranging from firewalling, proxies to WAN optimizers, on the traffic flows passing through them. An emerging and promising technology, Network Function Virtualization (NFV), moves packet processing from dedicated hardware middleboxes to Virtualized Network Functions (NFVs) running on commodity (*e.g.*, x86 based systems) servers [3], [4]. As the volume of traffic surges [5], it is increasingly important to provide Quality-of-Service (QoS) guarantees across traffic flows. Fair queueing is a fundamental tool to achieve this objective. With fair queueing, a scheduler determines the order in which packets from multiple flows are forwarded on a shared resource, allocating a prescribed fair share to each flow [6]. Achieving fair queueing in NFV, however, is particularly challenging, due to the diversity in traffic characteristics and the heterogeneity of traffic demands.

Usually, flows are grouped and organized on tenant boundaries in NFV use cases [7], [8]. A tenant represents some aggregate of flows that are grouped according to administrative affiliation, protocol, traffic type, or other criteria [9], [10]. In a multi-tenant environment, there exists a multi-level *hierarchy* (*i.e.*, a multi-level tree) specifying the organizational structure of traffic flows, with its leaves to represent individual flows and its internal nodes to represent tenants. For such workloads, it is desirable for the scheduler to provide QoS guarantees in multiple levels, individual flows as well as internal tenants.

In addition, flows are also characterized by a high degree of demand diversity across *multiple* resources. Different VNFs required by different flows consume vastly different amount of resources. For example, basic forwarding functionality uses more link bandwidth than other resources. IPSec encryption, on the other hand, is CPU-bound. Payload analyses on production traces from Intel, Google and Facebook [11], [12], [13] confirmed that flows may consume vastly different amount of resources of CPU, memory, I/O bandwidth and network bandwidth.

Despite the extensive studies in multi-resource fair queueing recently, no existing algorithm is designed to support hierarchies. Specifically, Ghodsi *et al*. [11] proposed the first work on multi-resource fair queueing, named Dominant Resource Fairness Queueing (DRFQ), in which the flows receive roughly the same packet processing time on their respective *dominant resources* — the resources they respectively require the most processing time on. From DRFQ, a string of follow-up papers [14], [15], [16], [17], [18] modified, developed and extended DRFQ. However, their focus has so far been primarily on *flat* or *non-hierarchical* flow scheduling. In fact, with hierarchies, even the packet orders of internal nodes are *unclear*: each internal node acts as a sub-scheduler and its order of packets is determined by the scheduling policy applied to its children. Worse, as we shall show in Section III-D, naive extensions of DRFQ to support hierarchies violate the *hierarchical share guarantees*, the prescribed fair share of resources that each node in the hierarchy at least gets.

In this paper, we propose the first rigorous study on the fair queueing algorithms that supports hierarchies in multi-resource environments such as VNFs. We propose two new multi-resource fair queueing algorithms, collapsed Hierarchical Dominant Resource Fair Queueing (collapsed H-DRFQ) and dove-tailing H-DRFQ, that support hierarchical scheduling by providing hierarchical share guarantees. The main idea of collapsed H-DRFQ is to convert the hierarchy into a flat tree and then use the flattened tree as a DRFQ scheduler, while dove-tailing H-DRFQ follows the thought to execute DRFQ within each set of sibling nodes (nodes that share the same parent). Although both H-DRFQ algorithms support hierarchies, their scheduling orders of packets are different. To quantify the differences between collapsed H-DRFQ and dove-tailing H-DRFQ, we mathematically formalize the delay bound of a packet in each of both H-DRFQ algorithms. Theoretical analysis shows that dove-tailing H-DRFQ *always*

outperforms collapsed H-DRFQ by providing a smaller delay bound. However, according to the experimental results, both algorithms have their pros and cons. The actual delays a packet experiences in collapsed H-DRFQ are smaller than those in dove-tailing H-DRFQ when the flow is in the higher levels, levels with smaller number of edges to the root node of the hierarchy, while dove-tailing H-DRFQ works better for flows in the lower levels.

To summarize, our contributions in this work are four-fold:

- We identify the problem of *hierarchical* multi-resource fair queueing.
- We propose two hierarchical multi-resource fair queueing algorithms, collapsed H-DRFQ and dove-tailing H-DRFQ, that provide hierarchical share guarantees. (Section IV)
- We bound the scheduling delays of packets in both algorithms, showing dove-tailing H-DRFQ works better. (Section V)
- We conduct extensive simulations to evaluate the performances of H-DRFQ. The results show that both H-DRFQ algorithms have their own advantages. (Section VI)

II. Related Work

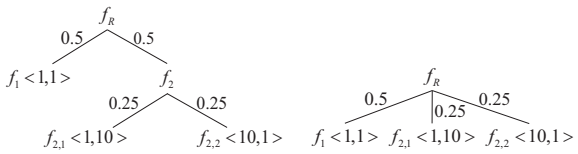
Traditionally, for a single resource, many classic fair queueing algorithms, such as WFQ [19], GPS [6], DRR [20], and SFQ [21], were proposed to provide share guarantees. In this section we present further details on most closely related works in the literature. We categorize these works between hierarchical and multi-resource fair queueing.

Hierarchical Fair Queueing: The hierarchical structures of traffic flows are common in any type of networks [22], [8]. To provide hierarchical share guarantees, several hierarchical scheduling algorithms are proposed in [23], [9], [10], [24]. These works extend the fair queueing problem from a one-level flat tree to a multi-level hierarchy. However, none of these works support flow scheduling across multiple resources. Our work bridges this gap and thus complements existing studies.

Multi-resource Fair Queueing: With the development of NFV, an increasing number of papers has studied the multi-resource fair queueing problem in a middlebox. Notably, Ghodisi *et al*. propose DRFQ that achieves DRF [12] in time domain. That is, flows in DRFQ receive roughly the same packet processing time on their respective dominant resources. We will review the mathematical details of DRFQ in Section III-D. Base on DRFQ, a string of flow-up papers [14], [15], [16], [17], [18] modified, developed and extended DRFQ. However, none of these works consider the hierarchical workloads submitted to a middlebox that supports multi-tenancy. Our work bridges this gap and thus complements existing studies.

III. Problem Formulation and Challenges

We begin by introducing the hierarchical fair queueing model in Section III-A and III-B, introducing the definition of hierarchical share guarantee in Section III-C, reviewing DRFQ in Section III-D and then arguing the inefficiencies of naive extensions of DRFQ to support hierarchies in Section III-E.



(a) Simple example of a hierarchy. (b) First Attempt

Fig. 1. Example of the naive extensions of memoryless DRFQ

A. Notations in Hierarchical Scheduling

The requirements of hierarchical workloads of traffic are specified with a weighted tree (see Fig. 1a). The root node represents a VNF scheduler and a leaf node represents a physical flow that ultimately being assigned resources. Each non-leaf node is connected to its parent by a logical queue.

A node in the tree is denoted as f_i, where i is a list of numbers that describe how the node can be found starting from the top of the tree and going down, left to right. For example, $f_{2,1}$ is found by starting at the root, picking its second (from left) child, and picking that child's first child. The root node is f_R (or R). Each flow i is associated with a weight ϕ_i. Without loss of generality, we assume that the sum of weights of all leaf nodes in the hierarchy to be equal to 1. The parent of flow i is denoted as $P(i)$, the set of children of flow i is denoted as $C(i)$, and the set of sibling nodes of flow i is denoted as $Sib(i)$. Then we have $\sum_{j\in C(i)} \phi_j = \phi_i$. For each flow i with H ancestors in a hierarchy, we write $P^h(i)$ to refer to its hth predecessor for $h = 0, 1, \ldots, H$, where $P^1(i) = P(i)$ and $P^H(i) = R$.

B. Design Objective

Let m be the number of resource types under consideration. Denote the kth packet of flow i as p_i^k. The *dominant resource* of a packet is defined as the one that requires the maximum packet processing time. Specifically, let $s_{i,r}^k$ be the packet processing time of p_i^k at resource r, the dominant resource of p_i^k is $r_{p_i^k} = \arg\max_r s_{i,r}^k$. We refer the processing time requirements $\{s_{i,r}^k\}$ $(r = 1, \ldots, m)$ of a packet p_i^k as its *packet profile*. And we denote μ_i^k as the processing time of p_i^k at its dominant resource (*i.e.*, $\mu_i^k = \max_r s_{i,r}^k$).

The main idea of an ideal hierarchical multi-resource fair queueing algorithm is to equalize the packet processing time between each pair of sibling nodes in a hierarchy on their respective dominant resources. If we denote $W_i(t_1, t_2)$ as the total processing time consumed by flow i on its dominant resource during time interval $[t_1, t_2]$, then we hope that

$$\frac{W_i(t_1, t_2)}{\phi_i} = \frac{W_j(t_1, t_2)}{\phi_j}, \tag{1}$$

between any pair of sibling nodes f_i and f_j. This objective is only achieved when packets in flows are infinitely divisible. Since in practice the packets are *not* divisible, we will design algorithms to approximate this ideal fairness.

In many workloads, packets within the same flow have similar packet profiles. Therefore, throughout, we assume that packets within the same *physical* flow have the same packet profiles and the dominant resource of a physical flow is the dominant resource of any of its packet. That is, for all leaves,

we have $s^k_{i,r} = s_{i,r}$, where $s_{i,r}$ denotes the packet processing time of any packet of leaf flow i at resource r. Notice that under this assumption, an internal node in a hierarchy may still experience packets with different packet profiles.

C. Hierarchical Share Guarantee

Hierarchical share guarantee is a *key* property that guarantees flow and tenant isolations in hierarchical scheduling. For non-hierarchical scheduling, DRFQ ensures that any active flow i with weight ϕ_i receives at least $\frac{\phi_i}{\phi_q}$ fraction of one of the resource it uses [11], where q denotes a VNF scheduler, and $\phi_q = \sum_{j \in C(P(i))} \phi_j$. We extend this guarantee to hierarchical scheduling as follows:

Hierarchical Share Guarantee: A node in a hierarchy is supposed to receive at least $\frac{\phi_i}{\phi_{P(i)}}$ fraction of one of the resources it uses from its parent node.

D. Review: Multi-Resource Fair Queueing for Packet Processing (DRFQ)

DRFQ makes non-hierarchical scheduling decisions for flows in a middlebox. Flows in DRFQ receive roughly the same packet processing time on their respective dominant resources. Specifically, according to the characteristics of the packet profiles, [11] introduced a tradeoff between memoryless DRFQ and dove-tailing DRFQ, which can be briefly illustrated as follows.

Memoryless DRFQ: Memoryless DRFQ applies to flows where different packets from the same flow have the same packet profiles. In memoryless DRFQ, a flow's current share of resources should not depend on its past share.

Consider two flows in a middlebox sharing two types of resources, f_1 sends packets with profile $\langle 2, 1\rangle$, f_2 sends packet with profile $\langle 1, 2\rangle$. Then the dominant resource of f_1 is r_1 while the dominant resource of f_2 is r_2. Memoryless DRFQ schedules two flows alternatively such that the processing time of f_1 on r_1 and the processing time of f_2 on r_2 are the same.

Dove-tailing DRFQ: Dove-tailing DRFQ applies flows where different packets from the same flow have different packet profiles. Dove-tailing allows the scheduler to have memory of past processing time given to a flow.

For example, a flow that sends a total of 10 packets, alternating in processing time requirements $\langle 2, 1\rangle$ and $\langle 1, 2\rangle$, respectively, is supposed to be treated the same as a flow that sends 5 packets, all with processing time $\langle 3, 3\rangle$.

E. Inefficiencies of Naive DRFQ Extensions to Support Hierarchies

First Attempt: Since the weight of a parent node is equal to the summation of the weights of its children, one intuition is to treat the hierarchy as a flat tree and the weight of each leaf node in the flat tree is exactly its weight in the hierarchy. Although this method *always* works well for a single resource [24], it turns out to be invalid for multi-resource scheduling. Specifically, it violates the hierarchical share guarantee.

Consider the hierarchy shown in Fig. 1a, our first attempt is to treat this hierarchy as the flat tree shown in Fig. 1b, then the scheduler can apply memoryless DRFQ to the three physical flows directly. After an initial start, DRFQ obtains a periodic pattern in which the ratio of packet numbers scheduled in the three flows is $40 : 1 : 1$, making the length of the period to be 51 time units. As a result, $f_{2,1}$ gets resource shares $\langle \frac{1}{51}, \frac{10}{51}\rangle$, while $f_{2,2}$ gets resource shares $\langle \frac{10}{51}, \frac{1}{51}\rangle$, leading their parent f_2 to get resource shares $\langle \frac{11}{51}, \frac{11}{51}\rangle$. Therefore, the dominant resource share of f_2 is $\frac{11}{51}$, much smaller than its guaranteed share 0.5, thus violates the hierarchical share guarantee.

Second Attempt: Since the packet profiles within the same physical flow are assumed to be the same, the second intuition is to apply memoryless DRFQ to each set of sibling nodes. Surprisingly, this method also works well for a single resource but violates the hierarchical share guarantee in multi-resource scheduling.

Still, consider the hierarchy shown in Fig. 1a, after applying memoryless DRFQ to $f_{2,1}$ and $f_{2,2}$, the logical queue on node f_2 can be regarded as a flow alternating $\langle 1, 10\rangle$ and $\langle 10, 1\rangle$. We then apply memoryless DRFQ to f_1 and f_2. After an initial start, DRFQ obtains a periodic pattern in which the ratio of packet numbers scheduled in the three flows is $20 : 1 : 1$, making the length of the period to be 31 time units. Similar to the first intuitive method, here f_2 gets a dominant share of $\frac{11}{31}$, smaller than its guaranteed share 0.5. Therefore, the second attempt still fails to support hierarchical fair queueing.

The failure of naive memoryless DRFQ extensions to the hierarchical fair queueing necessitates alternative scheduling mechanisms, which are the main theme of the next section.

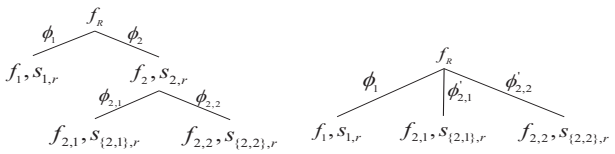
IV. Hierarchical Multi-resource Fair Queueing for Packet Processing

We begin by introducing collapsed H-DRFQ in Section IV-A and then dove-tailing H-DRFQ in Section IV-B.

A. Collapsed Hierarchical Multi-resource Fair Queueing

One well-known approach, which we call *collapsed hierarchies* [24], converts a hierarchical scheduler into a flat one. The idea is to take the hierarchy specification and compute what the corresponding weight would be for each leaf node if the hierarchy is flattened. As it was claimed in our first attempt in Section III-D, although this method works well for a single resource, it turns out to be non-trivial for multi-resource scheduling since the hierarchical share guarantees for internal nodes may be violated. We design an algorithm, collapsed H-DRFQ, that also flattens a hierarchy while still being able to provide the hierarchical share guarantee.

Given a hierarchy where all leaf flows are backlogged, we can always regard an internal node as a virtual backlogged flow that submits packets with *fixed* packet profiles. For example, in Fig. 1a, f_2 can be regarded as a flow submitting $\langle 11, 11\rangle$ continuously. We refer to this packet profile as *virtual packet profile*. In order to simplify the computation, we use the normalized packet profiles computed using $l^k_{i,r} = \frac{s^k_{i,r}}{\mu^k_i}$ to generate the virtual packet profile of internal nodes, where $l^k_{i,r}$ denotes the normalized packet processing time of p^k_i at resource r. At this



(a) Minimum general sub-hierarchy. (b) Transformation of Fig. 2a.

Fig. 2. Weight determination for the leaf node.

point, we are able to formally define the virtual packet profile of an internal node (*i.e.*, an internal scheduler) as following:

Definition IV.1 (Virtual Packet Profile). The virtual packet profile of a scheduler q on packet p_q^k is defined as the sum of the normalized packet processing time of all busy children of q on their respective resources. That is,

$$s_{q,r}^k = \sum_{i \in C(q)} l_{i,r}^k \phi_i. \tag{2}$$

Since all packets within the same leaf node submit demands with the same packet profiles, *i.e.*, $s_{i,r} = s_{i,r}^k$, each internal node will always keep a constant virtual packet profile during any of its backlogged period as long as its set of busy children does not change.

As a concrete example, consider an internal flow $\tilde{f}$ that has two children flow 1 and flow 2 with packet profiles $\langle 4, 2\rangle$ and $\langle 1, 2\rangle$ that share two resources. Thus the normalized packet profiles of flow 1 and flow 2 are $\langle \frac{4}{4}, \frac{2}{4}\rangle = \langle 1, 0.5\rangle$ and $\langle \frac{1}{2}, \frac{2}{2}\rangle = \langle 0.5, 1\rangle$, respectively. The weight of flow 1 is 0.4 and the weight of flow 2 is 0.2. Therefore, the packet processing time of $\tilde{f}$ on r_1 is $1 \times 0.4 + 0.5 \times 0.2 = 0.5$, while the value on r_2 is $0.5 \times 0.4 + 1 \times 0.2 = 0.4$. This defines the virtual packet profile of $\tilde{f}$ to be $\langle 0.5, 0.4\rangle$.

Now we compute the weight for each leaf node in the flattened tree. Consider a minimum general sub-hierarchy, which is able to form any hierarchical tree through iteration (see Fig. 2a). Let n_i be the number of packets scheduled in f_i from the hierarchy shown in Fig. 2a and n_i' be the number of packets scheduled in the same node f_i but from the flat tree shown in Fig. 2b. Assume all the flows are backlogged throughout the example. Then achieving DRFQ allocation between each pair of sibling nodes in Fig. 2a gives us

$$\frac{n_{2,1}\mu_{2,1}}{\phi_{2,1}} = \frac{n_{2,2}\mu_{2,2}}{\phi_{2,2}} \tag{3a}$$

$$\frac{n_1\mu_1}{\phi_1} = \frac{n_2\mu_2}{\phi_2} \tag{3b}$$

$$n_2 s_{2,r} = n_{2,1} s_{\{2,1\},r} + n_{2,2} s_{\{2,2\},r}, \quad (r = 1, \ldots, m) \tag{3c}$$

Combining (2) and (3c), we have

$$n_{2,1} = \frac{\phi_{2,1}}{\mu_{2,1}} n_2; \quad n_{2,2} = \frac{\phi_{2,2}}{\mu_{2,2}} n_2. \tag{4}$$

Meanwhile, achieving DRFQ allocation between each pair of sibling nodes in Fig. 2b we have

$$\frac{n_1'\mu_1}{\phi_1} = \frac{n_{2,1}'\mu_{2,1}}{\phi_{2,1}'} = \frac{n_{2,2}'\mu_{2,2}}{\phi_{2,2}'}. \tag{5}$$

In order to convert the hierarchy into a flat one, we hope that the allocations received by corresponding leaves in Fig. 2a and Fig. 2b are exactly the same. That is, $n_1 = n_1'$, $n_{2,1} = n_{2,1}'$ and $n_{2,2} = n_{2,2}'$. Combining (3b), (4) and (5), we have

$$\phi_{2,1}' = \frac{\phi_2 \phi_{2,1}}{\mu_2}; \quad \phi_{2,2}' = \frac{\phi_2 \phi_{2,2}}{\mu_2}. \tag{6}$$

Therefore, in a general case where the hierarchy has an arbitrary tree topology, the weight for each leaf node in the flattened tree is

$$\phi_i' = \phi_i \prod_{h=1}^{H-1} \frac{\phi_{P^h(i)}}{\mu_{P^h(i)}}. \tag{7}$$

More formally, we present the pseudo-code of collapsed H-DRFQ in Algorithm 1. Each time flows are added, removed, or change their demand status, the weights are recomputed.

Algorithm 1: Collapsed H-DRFQ Pseudo-code

```
Function Collapsed H-DRFQ(f_i):
    for each leaf node f_i do
        Calculate φ'_i
        Insert f_i in the flat tree with weight φ'_i
    end
    Apply memoryless DRFQ to the flattened tree
```

Algorithm 2: Dove-tailing H-DRFQ Pseudo-code

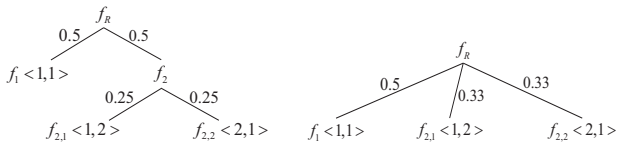
```
Function Dove-tailing H-DRFQ(f_i):
    if f_i is a leaf node then
        return
    end
    if not each child of f_i is visited then
        Choose an unvisited child f_i' of f_i
        Run Dove-tailing H-DRFQ(f_i')
    end
    Apply dove-tailing DRFQ to f_i
    Enqueue the scheduling result to f_i's parent's logical
     packet queue
    Mark f_i as visited
```

B. Dove-tailing Hierarchical Multi-resource Fair Queueing

Besides the collapsed H-DRFQ that firstly transforms a hierarchy into a flat tree, an alternative algorithm is to schedule sibling nodes directly in a DRFQ manner. In this algorithm, each non-leaf node can be regarded as a logical queue with packets in the order submitted to that node from its children, which we refer to as the *logical packet profile*. Formally, we give the definition as following,

Definition IV.2 (Logical Packet Profile). At any given time t, the logical packet profile of a packet of a scheduler q is defined as the head-of-line packet profile of its child that is currently being scheduled by the scheduler.

The logical packet profile provides another way to consider the packet processing time requirements of internal queues other than the virtual packet profile. Packets submitted to an internal node from different children may be heterogenous, leading the internal node to behave as a logical queue over time, with multiple packets that have different packet profiles. Therefore, it is desirable for the scheduler to have memory of



(a) Example of a hierarchy (b) Flattened tree of Fig. 3a

Fig. 3. Example hierarchy

the past packets submitted to each internal queue. This feature is referred to as *dove-tailing* in [11]. As a result, our second H-DRFQ algorithm uses dove-tailing DRFQ between each pair of sibling nodes. We refer to this method as dove-tailing H-DRFQ.

As a concrete example (see Fig. 3a), implementing dove-tailing DRFQ to the sub-scheduler f_2 makes its logical queue a flow with packet profiles $\langle 1, 2\rangle$ and $\langle 2, 1\rangle$ alternatively. Thereafter, implementing dove-tailing DRFQ to f_R makes its logical queue a flow with a periodic pattern in which 3 packets from f_1, one packet from $f_{2,1}$ and one packet from $f_{2,2}$ are scheduled.

More formally, we present the pseudo-code for dove-tailing H-DRFQ in Algorithm 2.

V. Delay Analysis of H-DRFQ

In this section, we study the differences between the two H-DRFQ algorithms by comparing their delay bounds. We begin by using a simple example to intuitively show the discrepancies in scheduling orders of packets between the two algorithms in Section V-A. Then we mathematically describe and compare their delay bounds in Section V-B and V-C.

A. A Simple Example Showing the Discrepancy

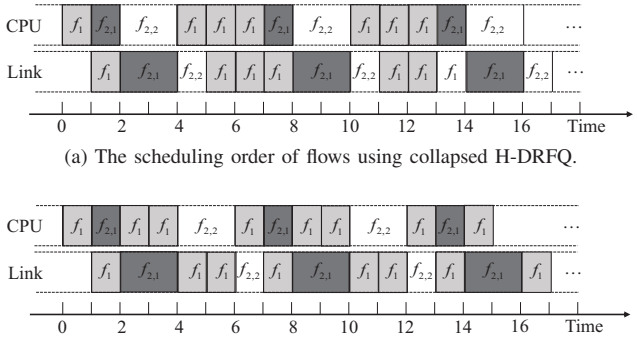
We use the example in Fig. 3a to coarsely illustrate the discrepancies between collapsed H-DRFQ and dove-tailing H-DRFQ. Assume all the three flows are backlogged throughout the example.

Fig. 4a shows how collapsed H-DRFQ works with the example hierarchy, and Fig. 4b shows how dove-tailing H-DRFQ works with the example. Collapsed H-DRFQ firstly transforms the hierarchy into a flat tree shown in Fig. 3b, and then use the flattened tree as a memoryless DRFQ scheduler. As shown in Fig. 4, after an initial start, the lengths of the periodic patterns of both algorithms are the same. That is, in each periodic pattern, three packets from f_1, one packet from $f_{2,1}$ and one packet from $f_{2,2}$ are scheduled. However, their packet orders are different from each other. The scheduling order of flows using collapsed H-DRFQ is $f_1, f_1, f_1, f_{2,1}, f_{2,2}$ in each period, while the scheduling order of flows using dove-tailing H-DRFQ is $f_1, f_{2,1}, f_1, f_1, f_{2,2}$ in each period.

The differences in scheduling orders may lead to different delays experienced by the same packet. Therefore, next we will analyse the scheduling delay packets experience to illustrate the differences between collapsed H-DRFQ and dove-tailing H-DRFQ.

B. Multi-resource WFI

To quantify the delay experienced by a packet, a metric called Worst-case Fair Index (WFI) was introduced in [9] to



(a) The scheduling order of flows using collapsed H-DRFQ.

(b) The scheduling order of flows using dove-tailing H-DRFQ.

Fig. 4. A scheduler that implements H-DRFQ algorithms.

characterize non-hierarchical fair queueing schedulers for a single resource (*e.g.*, link sharing), which can be generalized as following.

Definition V.1 (T-WFI)**.** A scheduler q is said to guarantee a Time Worst-case Fair Index (T-WFI) of $\mathcal{A}_{i,q}$ for flow i, if for any time t, the delay of a packet arriving at t is bounded above by the following,

$$d_i^k - a_i^k \leq \frac{Q_i(a_i^k)}{c_i} + \mathcal{A}_{i,q}, \tag{8}$$

where a_i^k and d_i^k are the arrival and departure times of p_i^k, respectively. c_i is the rate guaranteed to flow i, which is $\frac{\phi_i}{\phi_q}$ of the total rate of the link c_q. $Q_i(a_i^k)$ is the number of bits in the queue at time t.

Intuitively, $\mathcal{A}_{i,q}$ represents the maximum time a packet coming to an empty queue need to wait before receiving its guaranteed service share. That is, $\mathcal{A}_{i,q}$ represents the delay bound of flow i. An important observation is that in an ideal fluid system in which the packets are infinitely divisible and multiple flows can receive service simultaneously, $\mathcal{A}_{i,q}$ is 0 since f_i can receive its guaranteed share immediately after its arrival. However in a packet scheduling system, $\mathcal{A}_{i,q}$ must be larger than zero as a new arrival packet has to wait until the first arrivals finish in any one of the resources.

By multiplying c_i on both sides of the inequality in (8), it is easy to transform the inequality into the form of following,

$$W_i(a_i^k, d_i^k) \geq \frac{\phi_i}{\phi_q} W_q(a_i^k, d_i^k) - \alpha_{i,q}, \tag{9}$$

where $W_j(a_j^k, d_j^k)$ is the total amount of bits served during $[a_j^k, d_j^k]$, and $W_j(t_1, t_2) = c_j(t_2 - t_1)$. $\frac{\phi_i}{\phi_q}$ is the service share guaranteed to f_i. $\alpha_{i,q}$ represent a measurement of WFI in unit of bits instead of seconds.

However, if we extend (9) to multiple resource settings, the ideal service share received by f_i may not be $\frac{\phi_i}{\phi_q}$. We next propose a new theorem to rigorously quantify the ideal service share f_i received in DRFQ.

Theorem V.1. *In DRFQ, the ideal service share received by flow i from its scheduler q is*

$$\frac{W_i(t_1, t_2)}{W_q(t_1, t_2)} = \frac{\phi_i}{\mu_q}, \tag{10}$$

Proof: According to the definition of virtual packet profile in Definition V.1, for any multi-resource scheduler q, we have $W_q(t_2,t_2) = \mu_q n_q$. Meanwhile, the processing time of flow i on its dominant resource is $W_i(t_1,t_2) = \mu_i n_i$. According to (4), we have $n_i = \frac{\phi_i}{\mu_i} n_q$. Therefore, the ideal service share received by flow i from its scheduler q should be

$$\frac{W_i(t_1,t_2)}{W_q(t_1,t_2)} = \frac{\mu_i n_i}{\mu_q n_q} = \frac{\mu_i \phi_i n_q}{\mu_i \mu_q n_q} = \frac{\phi_i}{\mu_q}. \tag{11}$$

■

At this point, we are able to introduce a new definition of WFI in Definition V.2 that further applies to multi-resource packet processing.

Definition V.2 (M-WFI)**.** A server q is said to guarantee a Multi-resource Worst-case Fair Index (M-WFI) of $\alpha_{i,q}$, if for any interval $[t_1,t_2]$ within its backlogged period, the following holds

$$W_i(a_i^k, d_i^k) \geq \frac{\phi_i}{\mu_q} W_q(a_i^k, d_i^k) - \alpha_{i,q}, \tag{12}$$

where $W_j(t_1,t_2)$ is the total processing time served by f_j on its dominant resource during the time interval $[t_1,t_2]$. $\frac{\phi_i}{\mu_q}$ is the ideal service share that f_i supposed to receive in DRFQ.

In M-WFI, $\alpha_{i,q}$ represents the maximum processing time a packet p_i^k coming to an idle flow i need to wait before receiving its deserved service share. That is, $\alpha_{i,q}$ represents the processing time delay bound of flow i in DRFQ.

C. Delay Bounds of H-DRFQ

DRFQ has already rigorously bounded the delay of a packet that arrives when a flow is idle [11], which is shown in the next lemma. In this subsection, $s_{i,r}^{\uparrow}$ denotes $\max_k s_{i,r}^k$.

Lemma V.2. *Assume packet p_i^k of flow i arrives at t, and assume flow i is idle at time t. Assume all packets have non-zero demand on every resource. Then the maximum delay to start serving packet p_i^k, $\alpha_{i,q}$, is bounded by*

$$\alpha_{i,q} \leq \max_r \Big(\sum_{j=1, j\neq i}^{n} s_{j,r}^{\uparrow} \Big). \tag{13}$$

We next bound the delay of a packet in H-DRFQ. The delay bounds of collapsed H-DRFQ and dove-tailing H-DRFQ are shown in Theorem V.3 and V.4, respectively. We follow the same assumption with [11] that each packet has a non-zero demand for every resource.

Theorem V.3. *The collapsed H-DRFQ guarantees a delay bound $D^c(p_i^k)$ to start serving packet p_i^k by*

$$D^c(p_i^k) \leq \max_r \Big(\sum_{j=1, j\neq i}^{n} s_{j,r} \Big). \tag{14}$$

Proof: In collapsed H-DRFQ, the scheduler firstly transforms the hierarchical structure into a flat tree and then schedules the flows according to DRFQ. Therefore, the packet delay bound of collapsed H-DRFQ can be computed using the same method as DRFQ. Meanwhile, as shown in (14), for a flat scheduler, the packet delay bound has nothing to do with the flows' weights. Therefore the packet delay bound of collapsed H-DRFQ is the same as that of DRFQ since collapsed H-DRFQ only changes the weights of physical flows after flatting the hierarchy. Moreover, the packet profiles within the same leaf are assumed to be the same in collapsed H-DRFQ, *i.e.*, $s_{i,r}^{\uparrow} = s_{i,r}$. Thus the packet delay bound is $\max_r(\sum_{j=1,j\neq i}^{n} s_{j,r})$. ■

Theorem V.4. *The dove-tailing H-DRFQ guarantees a delay bound $D^t(p_i^k)$ to start serving packet p_i^k by*

$$\begin{aligned} D^t(p_i^k) \leq & \sum_{w=1}^{H-1} \prod_{h=0}^{w-1} \max_r \Big(\sum_{j \in \mathcal{U}(Sib(P^w(i)))} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} s_{j,r} \Big) + \\ & \max_r \Big(\sum_{j \in Sib(i)} s_{j,r} \Big), \end{aligned} \tag{15}$$

where $\mathcal{U}(i)$ denotes the set of leaves that share the same ancestor f_i.

Proof: Since node $P^{h+1}(i)$ is worst-case fair with the logical queue at node $P^h(i)$, the following holds for

$$W_{P^h(i)}(a_i^k, d_i^k) \geq \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} W_{P^{h+1}(i)}(a_i^k, d_i^k) - \alpha_{P^h(i)}, \tag{16}$$

where $W_{P^h(i)}(a_i^k, d_i^k)$ is the amount of service received by flow $P^h(i)$ in $[a_i^k, d_i^k]$, and $h = 0, \ldots, H-1$. Then we have

$$\begin{aligned} W_i(a_i^k, d_i^k) \geq & \frac{\phi_i}{\mu_{P(i)}} W_{P(i)}(a_i^k, d_i^k) - \alpha_i \\ \geq & \frac{\phi_i}{\mu_{P(i)}} \Big[\frac{\phi_{P(i)}}{\mu_{P^2(i)}} W_{P^2(i)}(a_i^k, d_i^k) - \alpha_{P(i)} \Big] - \alpha_i \\ \geq & \ldots \\ \geq & \prod_{h=0}^{H-1} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} W_R(a_i^k, d_i^k) - \\ & \Big(\sum_{w=1}^{H-1} \prod_{h=0}^{w-1} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} \alpha_{P^w(i)} + \alpha_i \Big). \end{aligned} \tag{17}$$

Therefore, we have

$$D^t(p_i^k) = \sum_{w=1}^{H-1} \prod_{h=0}^{w-1} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} \alpha_{P^w(i)} + \alpha_i. \tag{18}$$

Meanwhile, Lemma V.2 indicates that

$$\alpha_{P^h(i)} \leq \max_r \Big(\sum_{j \in Sib(p^h(i))} s_{j,r}^{\uparrow} \Big). \tag{19}$$

For each internal node $f_{P^h(i)}$ $(0 < h < H)$, we have $s_{P^h(i),r}^{\uparrow} = \max_{j \in C(P^h(i))}(s_{j,r}^{\uparrow}) \leq \sum_{j \in C(P^h(i))} s_{j,r}^{\uparrow}$. Consequently, the right side of (18) can be further bounded as flowing,

$$\begin{aligned} D^t(p_i^k) \leq & \sum_{w=1}^{H-1} \prod_{h=0}^{w-1} \max_r \Big(\sum_{j \in \mathcal{U}(Sib(P^w(i)))} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} s_{j,r} \Big) + \\ & \max_r \Big(\sum_{j \in Sib(i)} s_{j,r} \Big). \end{aligned} \tag{20}$$

■

A direct corollary (see Corollary 1) derived from (17) and (7) provides a rigorous formulation of the ideal service share received by each leaf node in a hierarchy.

Corollary 1. *The ideal dominant share received by each physical flow f_i in H-DRFQ is*

$$\prod_{h=0}^{H-1} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}}. \quad (21)$$

Given this ideal service share received by a flow in a hierarchy, we can rigorously prove that H-DRFQ can provide hierarchical share guarantees.

Theorem V.5 (Hierarchical Share Guarantee)**.** *H-DRFQ satisfies the Hierarchical Share Guarantee property.*

Proof: According to the method of generating virtual packet profile of internal queues shown in Equation (2), we have

$$s_{q,r}^k = \sum_{i \in C(q)} l_{i,r}^k \phi_i = \sum_{i \in C(q)} \frac{s_{i,r}^k}{\mu^k} \phi_i \leq \sum_{i \in C(q)} \phi_i = \phi_q. \quad (22)$$

Thus, $\mu_q = \mu_q^k = \max_r(s_{q,r}^k) \leq \phi_q$. Therefore, for each item in (21), we have

$$\frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}} \geq \frac{\phi_{P^h(i)}}{\phi_{P^{h+1}(i)}}, \quad (23)$$

as $\mu_{P^{h+1}(i)} \leq \phi_{P^{h+1}(i)}$ for $h = 0, \ldots, H-1$. Since $\frac{\phi_{P^h(i)}}{\phi_{P^{h+1}(i)}}$ denotes the hierarchical share guarantee we defined in Section III-B, (23) indicates that H-DRFQ satisfies hierarchical share guarantee property. ∎

The next corollary compares the delay bounds exhibited by collapsed H-DRFQ and dove-tailing H-DRFQ.

Corollary 2. *The delay bound of dove-tailing H-DRFQ $D^t(p_i^k)$ always outperforms the delay bound of collapsed H-DRFQ $D^c(p_i^k)$. That is,*

$$\max(D^t(p_i^k)) \leq \max(D^c(p_i^k)). \quad (24)$$

Proof: We observe that the upper bound of $D^t(p_i^k)$ (see (20)) can be regarded as a linear transformation of the delay bound of $D^c(p_i^k)$ (see (14)). Specifically, expect for the last term, each item of the bound of $D^t(p_i^k)$ is equivalent to the product of its corresponding item of the bound of $D^c(p_i^k)$ and a parameter $\prod_{h=0}^{w-1} \frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}}$. Moreover, this parameter is less than or equal to 1, as $\frac{\phi_{P^h(i)}}{\mu_{P^{h+1}(i)}}$ denotes the ideal dominant share of $f_{P^h(i)}$ and is always less than or equal to 1. And the last term of both bounds are the same. Therefore, we have $\max(D^t(p_i^k)) \leq \max(D^c(p_i^k))$. ∎

D. Discussion

Although the delay bound of dove-tailing H-DRFQ is always less than or equal to the delay bound of collapsed H-DRFQ (*i.e.*, $\max(D^t(p_i^k)) \leq \max(D^c(p_i^k))$), this does not necessarily indicate that dove-tailing H-DRFQ always outperforms collapsed H-DRFQ in packet delays. We will verify this point in Section VI-B through extensive simulations. In this subsection, we only give qualitative theoretical analysis.

The delay bound calculated in $D^c(p_i^k)$ may not be tight. Intuitively, the reason that a packet comes to an idle flow may have to wait for a long time is that some packets *related to it* have received *more* service than expected in a previous time period. In the case of non-hierarchical fair queueing, these packets must belong to the same flow i. In the case of hierarchical fair queueing, these packets may belong to flows that share the same ancestors with flow i. Therefore, WFI does not bound delay tightly using a flat or non-hierarchical scheduler since it does not take into account the fact that packets from the same flow may receive more service in a previous time period. However, WFI is more important in bounding the delay using a hierarchical scheduler because the extra service received in the previous time period may have been received by a flow other than the one being considered [9].

TABLE I
LINEAR MODEL FOR CPU PROCESSING TIME IN 3 MIDDLEBOX MODULES. MODEL PARAMETERS ARE BASED ON THE MEASUREMENT RESULTS. REPORTED IN [11].

Module	CPU processing time (μs)
Basic Forwarding	0.00286 × PacketSizeInBytes + 6.2
Statistical Monitoring	0.0008 × PacketSizeInBytes + 12.1
IPSec Encryption	0.015 × PacketSizeInBytes + 84.5

VI. PERFORMANCE EVALUATION

In this section, we present simulation experiments to (1) show the resource shares of the flow over time and confirm that both H-DRFQ algorithms offer hierarchical share guarantees, (2) show the difference in scheduling delay between collapsed H-DRFQ and dove-tailing H-DRFQ. For the purpose of verifying, we had two independent simulations, each corresponding to a hierarchical structure.

A. Dynamic Hierarchical Allocation

In this section, we demonstrate the dynamic hierarchical allocations on a simple hierarchy shown in Fig. 2a. To congest middlebox resources, we initiate 3 UDP flows, each sending 25,000 1300-byte packets per second. We configure the flows such that: (1) f_1 only undergoes basic forwarding, which is bandwidth bound, (2) $f_{2,1}$ requires statistical monitoring, which is also bandwidth-bound but uses slightly more CPU than basic forwarding, (3) $f_{2,2}$ undergoes IPSec, which is CPU-bound.

On the one hand, according to the measurement results in [11], the CPU processing time follows a simple linear model based on packet size and are summarized in Table I. On the other hand, the link transmission time is proportional to the packet size, and the output bandwidth of the middlebox is set to 200 Mbps, the same as [11].

Fig. 5 shows the dynamic resource share allocated to the three physical flows over time using the two H-DRFQ algorithms. Although the scheduling orders of packets in the two algorithms are not exactly the same, it provides the same results of resource shares allocated to each flow viewed from a long time scale perspective. Since f_1 is bandwidth-bound and is the only active flow during time interval $[0, 5]$, it receives 20% of the CPU share and all the bandwidth. In $[5, 10]$, f_1 and $f_{2,1}$ are active. Although the weight of $f_{2,1}$ is 0.25, at this point it has no sibling nodes. Therefore, the resources that are supposed to be allocated to $f_{2,2}$ are allocated to $f_{2,1}$. That is,

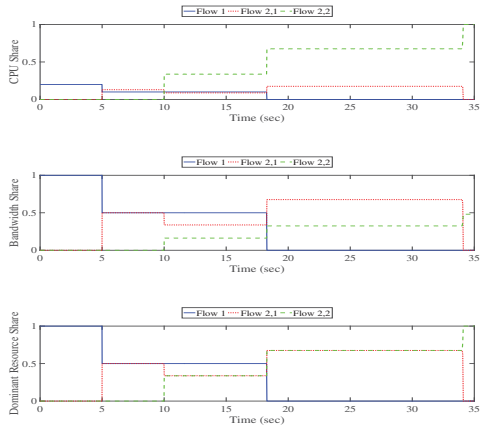


Fig. 5. Shares of three competing hierarchical flows arriving at different times. Flow 1, Flow 2,1 and Flow 2,2 respectively undergo basic forwarding, statistical monitoring and IPSec.

the weight of $f_{2,1}$ during $[5, 10]$ can be regarded as 0.5. As a result, the dominant shares of f_1 and $f_{2,1}$ in $[5, 10]$ are the same. Meanwhile, their bandwidth shares are also the same as they are both bandwidth bottleneck. Later, when $f_{2,2}$ becomes active after 10s, all the three flows are backlogged during the time interval $[10, 17]$. Since $f_{2,1}$ and $f_{2,2}$ is a pair of sibling nodes, their dominant shares are supposed to be the same, as the figure indicates. Since $f_{2,2}$ is CPU-bound, it grabs only 33.3% of the bandwidth allocated to f_2 (*i.e.*, 16.7% of the total bandwidth), yet is allocated 66.7% of the CPU allocated to f_2. Meanwhile, since the arrival of $f_{2,2}$ only leads to the reallocation of the resources allocated to f_2, the CPU share, bandwidth share, dominant share of f_1 remain to be the same until f_1 finishes in around 17s. Similar DRFQ allocations are observed between each pair of sibling nodes in subsequent time intervals. Through the whole process, we see that H-DRFQ algorithms can quickly adapt to traffic dynamics, providing predictable hierarchical share guarantees cross flows.

B. Delay Comparison

We next compare the packet delay under three schedulers, collapsed H-DRFQ, dove-tailing H-DRFQ and DRFQ, using a more complicated hierarchy shown in Fig. 6. By using the DRFQ scheduler we remove the hierarchical structure of the hierarchy shown in Fig. 6 but remain the weights of the leaves, thus the DRFQ scheduler schedules the leaves using their weights shown in the figure. We generate 150 UDP flows that start in serial every 0.1s, and these flows are divided into 15 groups to be submitted to each leaf node of the hierarchy. A flow randomly chooses one of the three middlebox modules to pass through, and the order of flows submitted to the leaves keeps the same across different algorithms. Let L denote the level of a hierarchy, which is the number of edges from a node to its root. To congest the VNF resources, the flow is set to send 10,000 UDP packets per second and the packet sizes uniformly drawn from 200 B to 1400 B, which are the typical setting for Ethernet. For each packet, we record its scheduling delay,

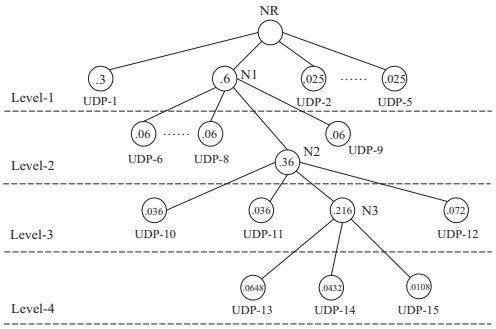


Fig. 6. Hierarchy used in Section VI-B

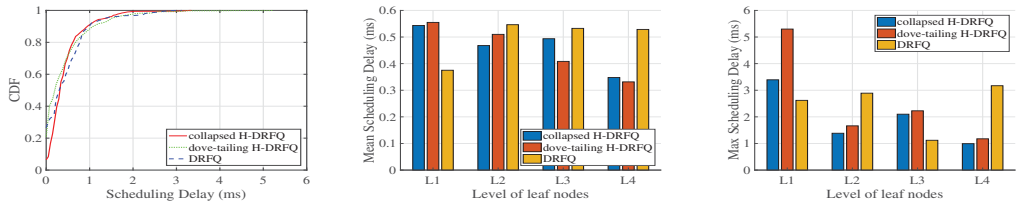
using collapsed H-DRFQ, dove-tailing H-DRFQ and DRFQ, respectively.

Fig. 7a depicts the CDF of the scheduling delay a packet experiences, from which we see that all packets can be scheduled in less than 6ms after their arrivals, among which nearly 80% of the them can be scheduled in 0.8ms. Further investigation reveals, comparing with collapsed H-DRFQ, that more packets in dove-tailing H-DRFQ suffer shorter delays (0ms-0.5ms), however, this is achieved at the expense of more packets to in dove-tailing H-DRFQ suffer longer delays (3ms-5ms). A detailed statistics breakdown is given in Fig. 7b and Fig. 7c showing the average and maximal delays a packet experiences in different levels. We notice that the mean scheduling delays of nodes from L_2, L_3 and L_4 using DRFQ are longer than those using the H-DRFQ algorithms, however, the delays of packets from L_1 show the opposite results. This indicates that physical nodes benefit from having longer distances to their root. We also observe that in general the lower the level is, the shorter the delay a packet experiences. Particularly, packets using collapsed H-DRFQ behaves better in higher levels (*i.e.*, L1 and L2), while dove-tailing H-DRFQ behaves better in lower levels (*i.e.*, L3 and L4).

Fig. 8 changes the weights of the nodes and shows the mean scheduling delay a flow experiences with respective to its weight in each of the four levels in the hierarchy. We observe that delay decreases as the weight increases. Still, collapsed H-DRFQ suffers shorter delays in L1 and L2, while dove-tailing H-DRFQ suffers shorter delays in L3 and L4. However, with the increase of weights, the scheduling delays experienced by the same packets tend to be the same using the two H-DRFQ algorithms. Interestingly, the delay bound of the collapsed H-DRFQ remains to be the same as the weight increases. However, the delay bound of the dove-tailing H-DRFQ decreases as the weight increases, showing the same variation tendency with the actual delays packets experience. As the qualitative theoretical analysis in Section V-D indicates, the upper bound of $D^c(p_i^k)$ is not tight. Each of the two H-DRFQ algorithms has its own advantage with respect to the level of flows in the hierarchy.

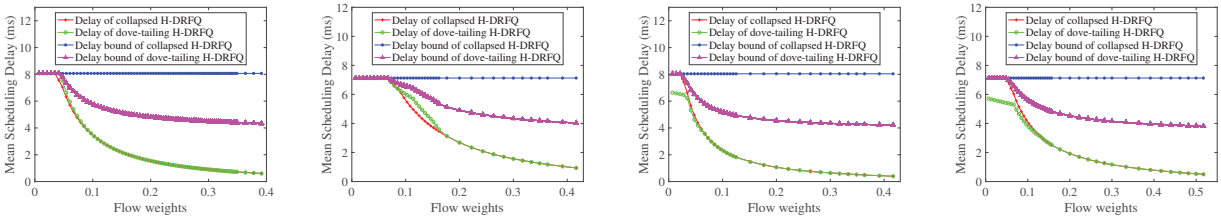
VII. Conclusions

In this paper, we designed two new hierarchical multi-resource fair queueing algorithms, collapsed H-DRFQ and



(a) CDF of the scheduling delay (b) Per-level mean delay comparison (c) Per-level maximum delay comparison

Fig. 7. Scheduling delay comparison between collapsed H-DRFQ and dove-tailing H-DRFQ of the hierarchy shown in Fig. 6.



(a) Mean scheduling delay in L1 (b) Mean scheduling delay in L2 (c) Mean scheduling delay in L3 (d) Mean scheduling delay in L4

Fig. 8. Scheduling delay comparison between collapsed H-DRFQ and dove-tailing H-DRFQ with respect to weights.

dove-tailing H-DRFQ. Collapsed H-DRFQ converted the hierarchy into a flat tree and then used the flat tree as a DRFQ scheduler, while dove-tailing H-DRFQ applied dove-tailing DRFQ between each pair of sibling nodes. Both algorithms provided hierarchical share guarantees. We studied the differences between the two H-DRFQ algorithms through comparing their scheduling delays. Theoretical analysis indicated that dove-tailing H-DRFQ worked better by providing a smaller delay bound. However, experimental results showed each of the two H-DRFQ algorithms had its own advantage with respect to the level of flows in the hierarchy.

VIII. Acknowledgements

This work is partially supported by the NSFC Fund (61671130, 61271165, 61301153), the National Basic Research Program (China's 973 program) (2013CB329103).

References

[1] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: Enabling innovation in middlebox deployment," in *HotNets Workshop*, 2011.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *SIGCOMM CCR*, 2012.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.

[4] G. Liu, Y. Ren, M. Yurchenko, K. Ramakrishnan, and T. Wood, "Microboxes: High performance NFV with customizable, asynchronous TCP stacks and dynamic subscriptions," in *SIGCOMM*, 2018.

[5] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP?" in *SIGCOMM*, 2011.

[6] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 3, pp. 344–357, 1993.

[7] P. Quinn and T. Nadeau, "Service function chaining problem statement," *draft-ietf-sfc-problem-statement-07*, 2014.

[8] M. Tufail, S. Majee, C. Captari, S. Homma *et al.*, "Service function chaining use cases in data centers," *draft-ietf-sfc-dc-use-cases-04*, 2016.

[9] J. C. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 5, no. 5, pp. 675–689, 1997.

[10] I. Stoica, H. Zhang, and T. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 2, pp. 185–199, 2000.

[11] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queueing for packet processing," *SIGCOMM CCR*, 2012.

[12] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types." in *NSDI*, 2011.

[13] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *SoCC*, 2012.

[14] W. Wang, B. Liang, and B. Li, "Multi-resource generalized processor sharing for packet processing," in *IWQoS*, 2013.

[15] W. Wang, B. Li, and B. Liang, "Multi-resource round robin: A low complexity packet scheduler with dominant resource fairness," in *ICNP*, 2013.

[16] W. Wang, B. Liang, and B. Li, "Low complexity multi-resource fair queueing with bounded delay," in *INFOCOM*, 2014.

[17] X. Li and C. Qian, "Low-complexity multi-resource packet scheduling for network function virtualization," in *INFOCOM*, 2015.

[18] C. Chen, W. Wang, S. Zhang, and B. Li, "Cluster fair queueing: Speeding up data-parallel jobs with delay guarantees," in *INFOCOM*, 2017.

[19] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM CCR*, 1989.

[20] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *SIGCOMM CCR*, 1995.

[21] P. Goyal, H. M. Vin, and H. Chen, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," in *SIGCOMM CCR*, 1996.

[22] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica, "Hug: Multi-resource fairness for correlated and elastic demands." in *NSDI*, 2016.

[23] P. Goyal, X. Guo, and H. M. Vin, "A hierarchical CPU scheduler for multimedia operating systems," in *OSDI*, 1996.

[24] A. Chandra and P. Shenoy, "Hierarchical scheduling for symmetric multiprocessors," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 19, no. 3, pp. 418–431, 2008.