A NEW APPROACH TO STOCHASTIC SCHEDULING IN DATA CENTER NETWORKS

Tingqiu Tim Yuan, Tao Huang, Cong Xu and Jian Li

Huawei Technologies, China

ABSTRACT

The Quality of Service (QoS) of scheduling between latency-sensitive small data flows (a.k.a. mice) and throughput-oriented large ones (a.k.a. elephants) has become ever challenging with the proliferation of cloud-based applications. In light of this mounting problem, this work proposes a novel flow control scheme, HOLMES (HOListic Mice-Elephants Stochastic), which offers a holistic view of global congestion awareness as well as a stochastic scheduler of mixed mice-elephants data flows in Data Center Networks (DCNs). Firstly, we theoretically prove the necessity for partitioning DCN paths into sub-networks using a stochastic model. Secondly, the HOLMES architecture is proposed, which adaptively partitions the available DCN paths into low-latency and high-throughput sub-networks via a global congestion-aware scheduling mechanism. Based on the stochastic power-of-two-choices policy, the HOLMES scheduling mechanism acquires only a subset of the global congestion information, while achieves close to optimal load balance on each end-to-end DCN path. We also formally prove the stability of HOLMES flow scheduling algorithm. Thirdly, extensive simulation validates the effectiveness and dependability of HOLMES with select DCN topologies. The proposal has been in test in an industrial production environment. An extensive survey of related work is also presented.

KEYWORDS

Data center network, flow scheduling, network partition, stochastic scheduling model

1. INTRODUCTION

The wide adoption of diverse cloud-based applications and services exacerbates the challenges the design and operation of Data Center Networks (DCNs). In a multi-tenant mode, long-lasting elephant and short-lived mouse flows share on DCN paths [15, 46, 60]. According to the results shown in [7], the sizes of the numerous short-lived flows are usually less than 10KB, and the average load of these mouse flows is typically less than 5% [7, 45, 46]. However, east-west traffic in the data center, between 75% and 95%, tends to require very low latency. On the other hand, the long-lasting heavy DC flows are typically much larger than 100KB; although the number of these large flows is extremely small compared to that of the small flows [64, 65]. These elephant flows account for more than 80% of bandwidth in DCNs [45, 102, 110, 112].

To provide high bisection bandwidth, DCN topologies are often multi-rooted topologies, e.g. Fat-Tree, Leaf-Spine, characterized by a large degree of multipath [40, 41, 96]. There are multiple routes between any two DCN endpoints [1, 2, 105, 110]. However, a critical issue in such network topologies is to design an efficient scheduling mechanism to balance the load among multiple available paths, while satisfying different application requirements defined in the Service Level Agreements (SLAs).

The defector DCN flow scheduling scheme Equal Cost Multi-Path (ECMP [3]) cannot meet such dynamic performance requirements in data centers. Hash collisions in ECMP can cause congestions, degrading throughput [4-6, 105-107] as well as tail latency [7-9, 109-112] of DCN flows. To balance the load between DCN switches and paths, stateful schedulers have been proposed, e.g. Conga [10], Hedera [4], etc. They monitor the congestion state of each path and direct flows to less congested paths, hence more robust to asymmetry network without control plane reconfigurations [11, 12]. Since maintaining global congestion information at scale is challenging, local congestion-aware or stochastic scheduling schemes are proposed, e.g. Expeditus [12], Drill [13], Hula [14], etc. Using simple or local information collection, these mechanisms are more efficient and applicable for complex DCN architectures, e.g. 3-tier Clos topologies. However, the majority of these scheduling mechanisms focus on balancing the loads of DCN according to the congestion information, without any consideration of cloud applications or data center traffic patterns.

TABLE I
SUMMARY OF KEY NOTATIONS AND DEFINITIONS

Notations	Definitions
b	Number of packets acknowledged by a received ACK
p	Probability that a packet in a flow is lost
$E[W]$	Expected average TCP window size
$L(w)$	Probability that a packet is lost when the window size is w
$W(t)$	Window size of a flow at time t
$Q(t)$	Queue length of an end-to-end path at time t
P_{SS}	Steady-state probability of the slow start period
P_{TD}	Steady-state probability of the convergence avoidance period
$B(p)$	Average data sending rate with the average probability of packet loss p
C	Bottleneck end-to-end link capacity of an end-to-end path
K	Marking threshold; when queue length exceeds this threshold, a congestion signal will be triggered
W_m	Maximum window size of a TCP flow
T_C	Duration of one TCP convergence avoidance period
T_S	Duration of one TCP slow start period
Q_{max}	Maximum queue size of an end-to-end path
N	Total amount of DC flows
D	Amplitude of oscillation in window size of a single flow
A	Amplitude of queue oscillations of an end-to-end path
$P_{M\text{-}E}$	Probability that the mouse flows affect the elephant flows
$P_{E\text{-}M}$	Probability that the elephant flows affect the mouse flows

Existing solutions to scheduling the mice-elephants hybrid DCN traffic fall into two categories. The first category deploys unified schedulers for both mice and elephants on shared paths, in spite of the competing performance requirements of the two. Based on the analysis of DC traffic patterns, these studies design novel scheduling algorithms or congestion signals [16, 17] and strike at the right balance between throughput and latency on shared DCN paths. The main challenge though is the interference between the elephant and mouse flows. The second category deploys network partition schemes that transfer the two types of flows over separate paths. [18, 60-63, 100, 101, 103, 104] By isolating elephant and mouse flows, network partition solutions avoid the aforementioned interference. This is particularly attractive as hardware and system cost continues to drop. Nonetheless, new policies are required to adaptively partition the DCN paths, given dynamic DC traffic patterns and varied DC architectures.

This paper focuses on the second category, the network partition solutions. Using a stochastic performance model, we first theoretically prove that the interference between mice and elephants are inevitable under the unified scheduling mechanism, indicating that network partition is a more appropriate solution for handling such hybrid DC traffic. We then propose a novel scheduling scheme, HOLMES, for such hybrid traffic in data centers. HOLMES architecture partitions a DCN into high-throughput and low-latency sub-networks, decouples the two competing scenarios and eliminates the interference in the hybrid traffic. HOLMES further deploys a stochastic and global congestion-aware load balancing algorithm that optimizes the performance of each sub-network. The stability of the HOLMES flow scheduling algorithm is also proved and validated in this paper.

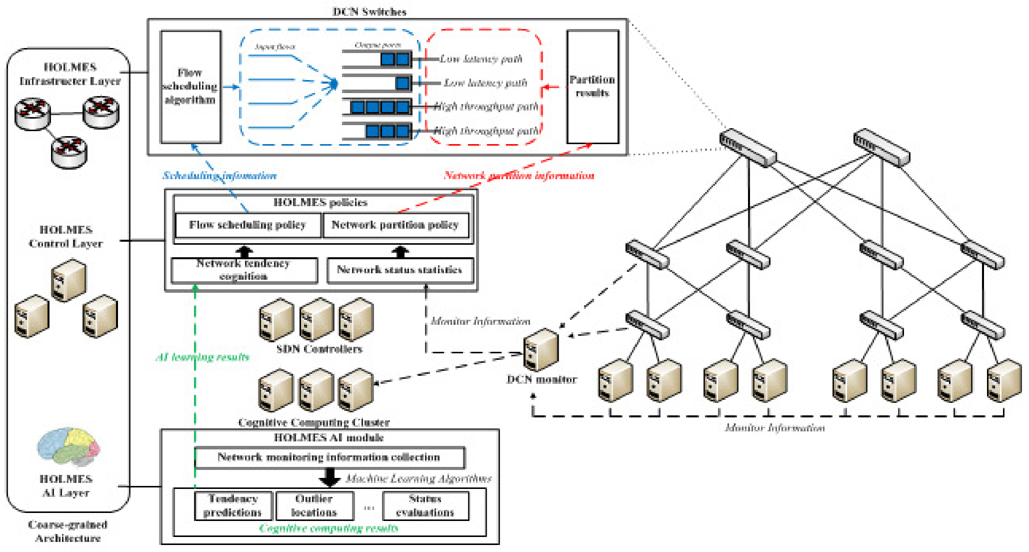


Fig. 1. HOLMES architecture: Based on the real-time monitor information, HOLMES AI module first analyzes the status or tendency of the DCN using machine learning models, and generates the learning results. Next, according to the analysis results, HOLMES control layer designs a network partition policy and a corresponding flow scheduling policy, and the policies are generated in the SDN controllers. Finally, the network partition as well as the flow scheduling operations will be executed on the DCN switches or hosts, under the guidance of the SDN controllers.

2. HOLMES ARCHITECTURE

Fig. 1 shows the HOLMES architecture in three layers: AI layer, control layer and the infrastructure layer. The HOLMES AI layer contains a cognitive computing cluster to implement the software AI module. The AI module collects the DCN state information from the DCN monitors, and applies the machine learning algorithms to generate some analysis results, e.g. networking tendency predictions, network outlier locations, etc. These learning results generated by the AI module provide a more comprehensive view of DCN behaviors. They will be then used for network partition and flow scheduling operations.

The HOLMES control layer is responsible for generating the network partition, congestion control, and local balancing policies, based on the monitoring information as well as the learning results generated by the AI module. The policies generated in the SDN are decomposed into a series of fine-grained partition and scheduling orders, which are transmitted to the DCN switches and end hosts for execution. Without deploying the HOLMES AI module, the functions provided by the control layer are the same as the traditional SDN controllers.

The HOLMES infrastructure layer executes the specific network partition as well as the flow scheduling operations. It is responsible for storing, forwarding and processing data packets. The detailed operation orders are transmitted and configured on the DCN switches. The DCN switches the first map each link to the high throughput network or the low latency sub-network, according to the network partition policies. When the elephant and mouse flow to arrive at a DCN switch, their packets are scheduled to the pre-partitioned paths separately. This process is managed by the HOLMES control layer.

A. Application Scenarios for HOLMES Architecture

Compared with the commonly used SDN architectures, a prominent feature of HOLMES is the implementation of the AI module and its machine learning algorithms. Machine learning methods have been widely used in network management [72-74, 95] and DC scheduling policy generation [75, 76, 96] operations. Those continuing learning and analysis results provide a comprehensive understanding of network features and behaviors, which benefits the designing of the corresponding network partition and flow scheduling policies. Therefore, the deep analysis and accurate AI prediction provided by the AI module enable the HOLMES architecture to perform more complex and intelligent operations.

One typical application scenario for HOLMES architecture is the deployment of application driven-networks (ADN) [77], where a physical network is sliced into logically isolated sub-networks to serve different types of applications. Each network slice in ADN can deploy its own architecture and corresponding protocols, to satisfy the requirements of the applications it serves. The key operations when implementing ADN are: (1) Constructing an application abstraction model to formulate the resource requirements of the applications; (2) mapping the distinct properties of applications to respective network resources. It is shown that the complexity and performance of these operations can be improved when some application features are pre-known [8]. Hence, the HOLMES AI module benefits the analysis of application features as well as the prediction of resource requirements, which further alleviate the complexity of application abstractions and mapping operations. Moreover, the design and implementation of network slicing mechanisms can also be realized by the cooperation of the control layer and the infrastructure layer.

Similarly, HOLMES architecture is also applicable for some other intelligent or complex application scenarios, which demand a deep understanding of network or application features such as Internet of Vehicles (IoV) [78], co flow scheduling [79, 80] and some other network architecture based on network slicing or network partitions. With the immense proliferation of complex and diverse cloud-based applications, we expect such architecture to be the development trend in the future.

B. Out-of-order vs. Efficiency

While elephants contribute to the majority volume of DCN traffic, mice account for 80% of the number of instances. [45, 102] Out-of-order scheduling done at host side may sacrifice efficiency. In addition, compatibility with legacy hardware has to be ensured. Therefore, one may still consider deploy conventional, sometimes oblivious, ECMP scheduling for in-order scheduling of mice.

3. HOLMES SCHEDULING ALGORITHMS

Once the hybrid DC traffic is separated into different sub-networks, scheduling algorithms affect the performance of each sub-network. In this section, we discuss the aforementioned global congestion-aware scheduling algorithm and prove the stability of its stochastic policy.

A. Deploying Stochastic Scheduling Algorithms

Compared with the other state-aware flow scheduling algorithms, stochastic flow scheduling algorithms are more applicable for large-scale data centers according to the following reasons:

1. Simplification of computing complexity

One of the key factors that degrade the performance of the traditional ECMP mechanism is the lack of global congestion information. To overcome this limitation, a group of studies has designed new flow scheduling policies based on a global "macroscopic" view of the entire DCN, e.g. CONGA [10]. However, in large-scale and heavily loaded data centers, the global macroscopic load balancing algorithms introduce unacceptable computing complexity to deal with the massive information, and the control loops in these scenarios are much slower than the duration of the majority of congestion incidents in data centers [13]. Therefore deploying the stochastic algorithms to achieve micro load balancing is a more viable solution. The micro load balancing solutions require only limited congestion information, which simplifies the computing complexity and enables instant reactions to load variations in large-scale data centers.

2. Optimization of storage complexity

In data centers, 8 and 16-way multipathing are common, while there is growing interest in multipathing as high as 32 or even 64. Specifically, with 40 servers in a rack, there will be 40 uplinks. Each flow can use a different subset of the 40 links, leading to 2^{40} possible subsets. Keeping the state of each path in this scenario requires unacceptable storage resources, which is difficult to be implemented. On the contrary, stochastic scheduling algorithms are effective to cope with the optimization of storage complexity, as well as the small number of register reads. Edsall *et al* [26] deploy the stochastic power-of-two-choices hashing solution for balancing loads of DC routers. The storage complexity of such a stochastic solution is logarithmically reduced.

3. Better adaptability for heterogeneous DCNs

A typical flow scheduling method in multi-rooted DCNs is equal traffic splitting based on hashing, as used in the traditional ECMP approach. However, the uniform hashing approach cannot achieve optimal load balance without the assumption of symmetric and fault-free topology [5, 10, 45, 46], which is not generally true in heterogeneous data centers. To provide better adaptability for heterogeneous DCNs, weighted traffic distribution methods have been widely adopted in the global macro load balancing solutions [11, 67]. In order to correct the imbalance caused by the even distribution approach and enable fair resource allocation, the weighted approaches distribute the traffic among available paths in proportion to the available link capacity of each path. The global weighted traffic distribution solutions have shown good adaptability to dynamically changing network topologies. However, these solutions still need real-time state information collection of all the paths, which introduces additional computing and storage complexity.

Stochastic flow scheduling algorithms can reduce the computing and storage overhead of weighted traffic distribution mechanisms, while maintaining the adaptability to heterogeneous DCN topologies. Consider the stochastic Power-of-Two-Choices: The algorithm only needs to record the states of the two randomly chosen paths; therefore, the storage and computing complexity are dramatically reduced. Moreover, the algorithm compares the load conditions of these two chosen paths, select the better one, hence performs a weighted operation in another form. Stochastic load balancing solutions have also been proved to be applicable for heterogeneous DCNs [13, 26]. Based on these justifications, we extend stochastic flow scheduling algorithms to our HOLMES mechanism.

B. Flow Scheduling Algorithm in HOLMES

We consider a stochastic scheduling policy, (d, m) policy: The input port chooses d random end-to-end paths out of all the possible paths. It finds the path with the minimum occupancy among all the d samples and m least loaded samples from the previous time slot. It then schedules the input packet to the selected end-to-end path.

Increasing the value of d and m to >>2 and >>1 will degrade the performance since a large number of random samples makes it more likely to cause the burst of packet arrivals on the same path [13]. As a result, we set m=1 and d=2 in our scheduling model. The detailed flow scheduling procedure is shown in Alg. 1.

Using global congestion information, the algorithm reacts rapidly to the link or node failures. Moreover, the limited information used in the algorithm improves the scheduling efficiency while avoids the traffic bursts on the same switch ports.

Algorithm 1 Flow scheduling policy in HOLMES

```
1: Input: Load condition of each end-to-end path at time t:
          {load(1), load(2), load(3),…}
       Path number of the selected path at time t -1: $s^{(t-1)}$
       Output ports of the TOR and aggregate switches on each path:
          $\{\{P^A_{(1)}, P^T_{(1)}\}, \{P^A_{(2)}, P^T_{(2)}\}, ...\}$
2: Output: Path number of the selected path at time t -1: s
       Output ports TOR and aggregate ports on the selected path:
          $\{P^A_{(s)}, P^T_{(s)}\}$
3: Initialize: $m = 2$, $d = 1$;
4: Initialize: $load_{OPT} = load(s^{(t-1)})$;
5: Random select m end-to-end paths {Path(1), Path(2), …, Path(m)}
6: Construct candidate set: $L \leftarrow \{Path(1), Path(2), …, Path(m)\} \cup \{s^{(t-1)}\}$
7: for each path i ($1 \le i \le m+1$) in the candidate set L do
8:   if $load(Path(i)) \le load_{OPT}$ then
9:     $load_{OPT} = load(Path(i))$;
10:  end for
11:  Assign value: s = Path number of the path with load $load_{OPT}$:
12:  return { $p^A_{(s)}$, $pT_{(s)}$ } and s
```

C. Stability Analysis of HOLMES's Scheduling Algorithm

We prove the stability of this stochastic global congestion-aware scheduling algorithm in a two-tier Clos DCN topology. We abstract an end-to-end path in a Clos network (Fig. 2A) as a serial queuing system consists of a series of queues (Fig. 2B). As a result, the whole Clos DCN topology is abstracted as a queuing network. We then evaluate the performance of a specific leaf-to-spine DCN path using a stochastic queuing network model.

We focus on analyzing the stability of the scheduling process from when a packet arrives at a TOR switch to when the packet reaches the spine switch. The packet experiences two queuing processes, at the TOR and the aggregate switch port, respectively. The entire path from the TOR node to the spine node can also be modelled as a large queue.

Based on the results of [53-56] and with a similar method shown in [57-59], we prove that HOLMES's scheduling algorithm is stable for all uniform and non-uniform independent packet arrivals. Some key notations and definitions used in the scheduling model are illustrated in Table II.

TABLE II
SUMMARY OF KEY NOTATIONS AND DEFINITIONS IN SCHEDULING MODEL

Notations	Definitions
λ_i	Average data arrival rate in the *i*thleaf-to-spine path
α_i	Average data processing rate of the TOR port in the *i*thleaf-to-spine path
w_i	Average data arrival rate of the aggregate port in the *i*thleaf-to-spine path
β_i	Average data processing rate of the aggregate port in the *i*thleaf-to-spine path
N	The number of the TOR and aggregate switches in a pod
N_A	The number of spine switches connected by each aggregate switch
$Q_k(t)$	The number of accumulated packets in the buffer of the *k*thleaf-to-spine path at time *t*
$\tilde{Q}_i(t)$	The number of accumulated packets in the buffer of the end-to-end path chosen by the *i*th input port using HOLMES at time *t*
$Q^*(t)$	The number of the accumulated packets in the global least loaded leaf-to-spine path at time *t*
$Q_{iT}(t)$	The number of the accumulated packets on the TOR port of the *i*thleaf-to-spine path, at time *t*
$Q_{iA}(t)$	The number of the accumulated packets on the aggregate port of the *i*thleaf-to-spine path, at time *t*
$Q^*_T(t)$	The number of the accumulated packets on the TOR port of the global least loaded leaf-to-spine path at time *t*
$Q^*_A(t)$	The number of the accumulated packets on the aggregate port of the global least loaded leaf-to-spine path at time *t*

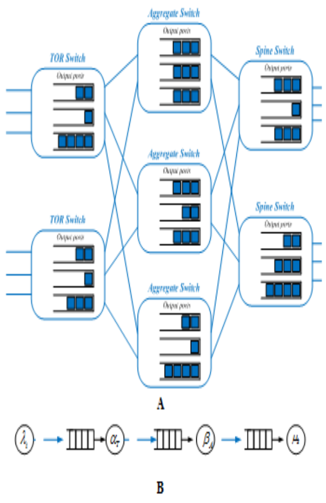


Fig. 2. Abstraction of a leaf-to-spine path in a Close network (A) to a serial queuing system (B)

We prove that the global (1, 1) policy is stable for all admissible parallel arrival process. We construct a Lyapunov function *L* as follows:

$$L(t)=\sum_{i=1}^{NN_A}(\tilde{Q}_i(t)-Q^*(t))^2+\sum_{i=1}^{NN_A}Q_i^2(t)$$

To prove the algorithm is stable, we show that there is a negative expected single-step drift in the Lyapnuov function, i.e.,

$$E[L(t+1)-L(t)\mid L(t)]\leq \varepsilon L(t)+k \quad (\varepsilon,k>0)$$

We divide the Lyapunov function into two sub functions as:

$$L_1(t)=\sum_{i=1}^{NN_A}(\tilde{Q}_i(t)-Q^*(t))^2 \qquad L_2(t)=\sum_{i=1}^{NN_A}Q_i^2(t)$$

Based on the above formulation, we prove that there exists a negative expected single-step drift in the Lyapnuov function in each possible case. Therefore, the global (1, 1) policy is stable. Based on the (*d*, *m*) policy, the HOLMES's scheduling algorithm is also stable. Please see details of the proof in Appendix B.

4. HOLMES PERFORMANCE EVALUATION

We evaluate HOLMES using simulation based on OMNET++ [99]. We construct a test-bed simulation platform to simulate the data transmission process in symmetric and asymmetric fat-tree DCNs. Similar to [10, 46, 52, 60, 62, 63], a heavy-tailed distribution is used to generate DC flow of different sizes. The hosts of the DCN run TCP applications. The flow request rate of each TCP connection satisfies the Poisson process.

A. Evaluation of HOLMES Network Partition

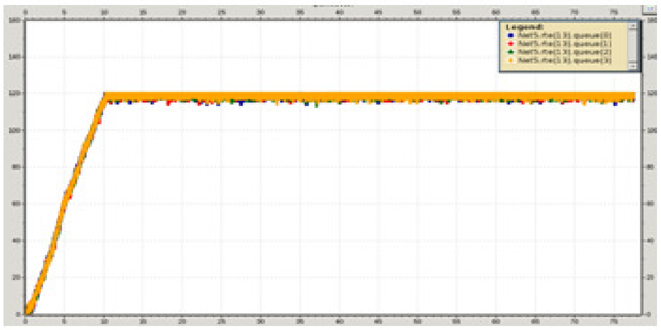
We evaluate the network partition policy of HOLMES in a scenario that hybrid elephant and mouse flows are scheduled in the same DCN with different scheduling schemes. The DCN topology deployed in this experiment is a Clos network with 2 and 4 leaf and spine switches respectively. We generate elephant and mouse flows to leaf switches with average sizes of 100KB and 1KB, respectively. The queue lengths of the switch ports are used as performance indicators.

Notations	Definitions
λ_i	Average data arrival rate in the *i*th leaf-to-spine path
α_i	Average data processing rate of the TOR port in the *i*th leaf-to-spine path
w_i	Average data arrival rate of the aggregate port in the *i*th leaf-to-spine path
β_i	Average data processing rate of the aggregate port in the *i*th leaf-to-spine path
N	The number of the TOR and aggregate switches in a pod
N_A	The number of spine switches connected by each aggregate switch
$Q_k(t)$	The number of accumulated packets in the buffer of the *k*th leaf-to-spine path at time *t*
$\tilde{Q}_i(t)$	The number of accumulated packets in the buffer of the end-to-end path chosen by the *i*th input port using HOLMES at time *t*
$Q^*(t)$	The number of the accumulated packets in the global least loaded leaf-to-spine path at time *t*
$Q_{iT}(t)$	The number of the accumulated packets on the TOR port of the *i*th leaf-to-spine path, at time *t*
$Q_{iA}(t)$	The number of the accumulated packets on the aggregate port of the *i*th leaf-to-spine path, at time *t*
$Q_T^*(t)$	The number of the accumulated packets on the TOR port of the global least loaded leaf-to-spine path at time *t*
$Q_A^*(t)$	The number of the accumulated packets on the aggregate port of the global least loaded leaf-to-spine path at time *t*

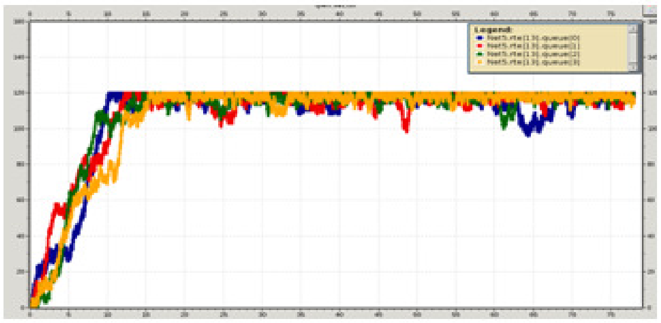
Since the scale of the DCN in the simulation is not very large, HOLMES deploys the static policy that partitions the two sub-networks in advance. The buffer sizes of all the switch ports are set to be the same. When the buffer of a switch port is full, all the upcoming input packets to that port will be dropped. We compare HOLMES against two start-of-the-art unified load balancing schemes: CONGA [10] and queue length gradient based scheduling. Similar to the delay gradient based congestion control policy used in [16, 66], we deploy the queue length gradient as the indicator and schedule the arrived packet to the port with the minimum length gradient.

Figs. 3A-3C show the queue length variation of the four ports of a spine switch under the three scheduling schemes. The X-axis indicates the time period and the Y-axis denotes the queue length. We can see from Fig. 3A that using CONGA, the buffers of all the four ports are full after a period of time, indicating the throughput of the switch has been maximized, which benefits the transmission of the elephant flows. However, when a mouse flow arrives, all the packets in that flow have to wait for a long queuing time since all the output port are of heavy loads. Consequently, the latency of the mouse flow will increase and degrade the overall performance of the hybrid DC flows.

Similarly, as shown in Fig. 3B, the buffers of the four output ports are also almost full after a period of time using the length gradient based policy. The results indicate that the length gradient based load balancing policy still suffers from the interference between the elephant flows and the mouse flows.



A



B

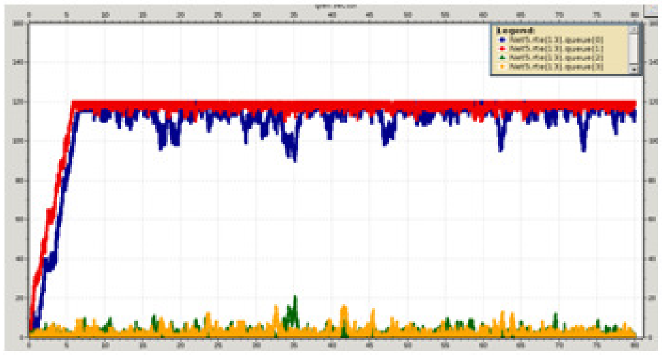

C

Fig. 3. Queue length changing trends of a DCN spine node's ports under policies CONGA (A), length gradient based policy (B) and HOLMES (C)

Comparing Fig. 3A and Fig. 3B, we find that the load balancing condition of the gradient-based scheme is a little worse than CONGA. The reason is that the gradient-based scheme schedules the DC flow according to the changing trend but not the current state of the DCN. Finally, Fig. 3C shows that HOLMES has successfully isolated the elephant and mouse flows. Two of the ports have been partitioned to the low latency sub-network and used for transmitting the mouse flows. Fig. 3C shows that the buffers of the two ports are almost empty during the entire transmission procedure. Thus, packets in the mouse flows do not need to wait for additional queuing delays, and the low latency of the mouse flow is ensured. Moreover, the buffers of the other high throughput ports are also full filled, which satisfies the throughput requirements of elephant flows. Hence, by isolating the mixed traffic, HOLMES network partition policy successfully eliminated the interference of the elephant flows to the mouse flows.

The main shortcoming of the network partition solution is the inefficient use of network resources. Although the isolation of the hybrid traffic avoids the interactions of the elephant and mouse flows, the spared network resource in the low latency paths has not been fully used since the buffers of these paths are almost empty. An effective solution is to improve the buffer allocation by limiting the buffer size of the low latency sub-network and assigning the spared buffers to the high throughput sub-network. This policy has been implemented in [61].

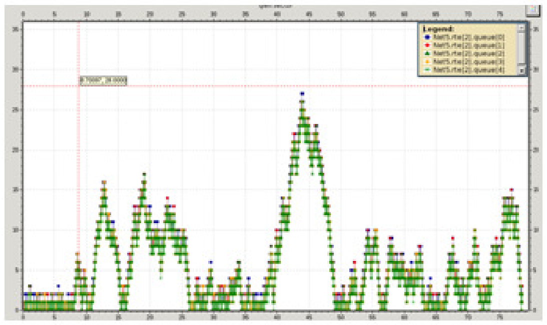
B. Stability Validation of HOLMES Scheduling Algorithm

We evaluate the stability of HOLMES flow scheduling algorithm. We simulate the scenario that DC traffic are scheduled in a DC with asymmetric network topology. We combine two different sized Clos networks, and construct an asymmetric DCN architecture. One of the Clos network consists of 2 leaf switches and 4 spine switches. The other is a Clos network with 5 and 4 leaf and spine switches, respectively. 10 hosts are attached to each leaf switch. We concentrate on validating the stability of HOLMES flow scheduling algorithm, rather than the network partition mechanism in this scenario. Thus, we do not deploy the HOLMES network partition mechanism in the experiment and only execute the HOLMES flow scheduling algorithm.

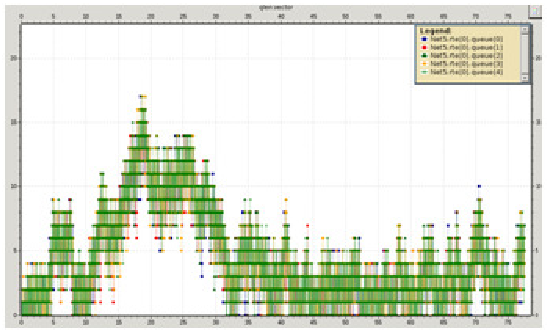
CONGA [10] and DRILL [13] are two load balancing solutions proven to be stable. Therefore, we compare them against HOLMES. All DC traffic is scheduled with the granularity of packet, and we focus on analyzing the stability of the three scheduling algorithms. When using the Power of Two Choices selections, we uniformly set d=2 and m=1. Similarly to the previous experiments, we deploy queue length as the load balance indicator to evaluate the overall load balancing condition of the DCN.

Figs. 4A-4C show the queue length changing trend of a specific leaf switch's ports, under load balancing policies CONGA, DRILL and HOLMES. We can see from Fig. 4A that the queue length changing trends of all the ports in a leaf switch are almost overlapped under CONGA, indicating that the queue lengths of all the switch ports are almost the same at each time unit. Therefore, the load balancing condition under CONGA is optimal among all the three mechanisms, since CONGA makes each scheduling decision based on the global congestion information. Without considering the time used for obtaining congestion information, CONGA obtains the global optimal load balancing result.

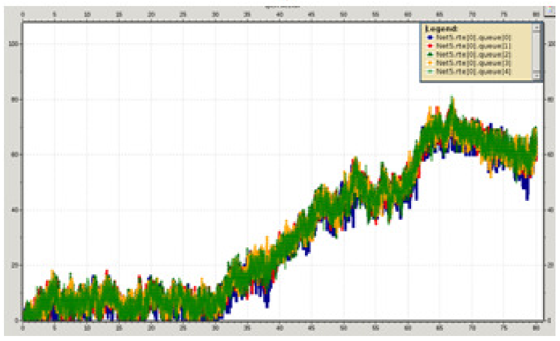
Fig. 4B shows the queue length changing trends of the same leaf switch ports under DRILL. Different with the former results, we find fluctuations in the queue length changing curve. In other words, the length difference of the longest queue and the shortest queue is clear. The reason is that the use of (*d*, *m*) policy in DRILL reduces the scale of the solution space, and the local optimal solutions affect the load balancing condition of the DCN.



A



B



C

Fig. 4. Queue length changing trends of a DCN leaf node's ports under load balancing policies CONGA (A), DRILL (B) and HOLMES (C)

Fig. 4C shows the queue length changing trends of the same leaf switch ports under HOLMES scheduling algorithm. The fluctuations also exist in the curve of Fig. 4C, where the amplitude of the fluctuation is more obvious. This phenomenon is also caused by the use of (d, m) policy. Compared with DRILL, the global (d, m) policy used in HOLMES further limits the solution space, and exacerbates the fluctuations. However, although the fluctuations are more obvious when executing HOLMES flow scheduling algorithm, we can also find an upper bound (about 10 packets) of the fluctuation amplitude, indicating that the length difference of the shortest and the longest queue is not infinite in HOLMES. Hence, our HOLMES flow scheduling algorithm is stable during the whole scheduling period. Moreover, limiting the solution exploration space reduces the time used to obtain the congestion information and make HOLMES more efficient and applicable for large-scale data centers.

C. Adaptability for Heterogeneous

As discussed earlier, both the stochastic flow scheduling algorithm and the weighted traffic splitting solutions can adapt to heterogeneous congestion states. We now evaluate the adaptability of the two solutions.

We first theoretically compare the approximate adaptability of the two solutions, as shown a simple leaf-spine DCN topology with N paths available between two racks as shown in Fig. 5.

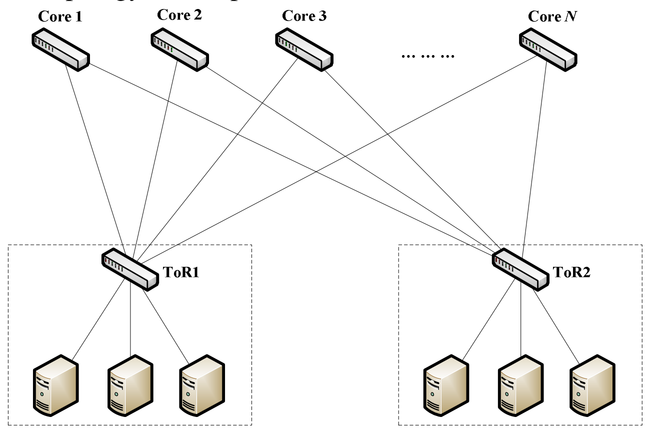


Fig. 5. Simple leaf-spine DCN topology for adaptability evaluations

We show that when the load conditions of the DC paths are heavily heterogeneous, the (d, m) policy also needs to maintain plenty of load status information to keep its adaptability as good as the weighted traffic splitting solutions. The stochastic scheduling mechanism does not show obvious advantages in this scenario.

Similar to the experiment in [97], we simulate the execution process of the coordinate approach as well as the Power-of-Two-Choices algorithm on a same switch. Fig. 6 shows the changing trend of the overall switch load as the modeling factor (d) increases.

The load distribution of the switch ports is initialized exponentially in this experiment. We see from Fig. 6 that, as the value of d increases, the load condition of the switch is improved under the power of two choices policy ((d, m) policy); since a larger value of d increases the probability of choosing the lightest loaded output port. As we increase the value of d from 2 to 5, the power of two choices policy performs almost as well as the theoretical load optimal policy (d = 10), which validates our previous modeling results. On the contrary, when using the coordinated approach, the switch attains optimal performance when the value of d is small ($d = 2$) and the load state of the switch is almost as good as the theoretical load optimal policy. This simulation result is in accordance with the analysis in [97]. However, as the value of d increases, the load state of the switch becomes worse: when assigning $d = 10$, the load balancing condition of the switch under the coordinated policy is even worse than the (2, 1) policy.

Although the stochastic flow scheduling outperforms the weighted traffic splitting solution in most cases, it still has some limitations. The weighted traffic splitting solution maintains the load status of all the paths. It dynamically adjusts the value of each weight according to the current load status of each path (the static weight configuration has proven to be not applicable in [10]). However, when deploying the (d, m) policy, the value of d and m are constant after the initializations. Thus, when the values are not appropriately assigned, (d, m) policy will not perform as well as the weighted traffic splitting solutions. Hence, the HOLMES AI module is responsible for analyzing the overall heterogeneity degree of a DCN, and guiding the flow scheduling algorithm to set appropriate values of the algorithm factors (d and m). The detailed design and implementation of the HOLMES AI module is our future work.

D. Technical Challenges in Hardware Implementations

Some technical challenges need to be considered to implement HOLMES in real-world data centers. We now summarize these challenges in hardware implementations.

1. Handling the packet reordering

The flow scheduling algorithm in HOLMES can be implemented with different data granularities: per packet scheduling, per flow scheduling or some intermediate data sizes, e.g. flow cell [24], flow let [10], etc. When using the TCP transmission protocol and implementing the per packet (or flow cell) scheduling, some studies have shown that this fine-grained traffic splitting techniques cause packet reordering and lead to severe TCP throughput degradations [23]. Therefore, the packet reordering problem needs to be considered when implementing the fine-grained HOLMES traffic scheduling algorithm. A viable solution is to deploy the JUGGLER [68] network stack in data center traffic transmissions. JUGGLER exploits the small packet delays in datacenter networks and the inherent traffic bursts to eliminate the negative effects of packet reordering while keeping state for only a small number of flows at any given time. This practical reordering network stack is lightweight and can handle the packet reordering efficiently.

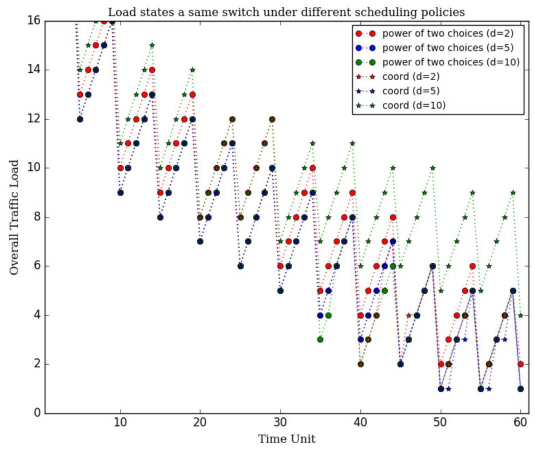


Fig. 6. Changing trend of a switch's overall traffic load under different policies

2. Design of DCN forwarding and state tables

The design of the forwarding and state tables is also a noteworthy challenge. An appropriate approach should cope with the small time budget as well as the small register usage. We now propose a viable design to implement the per-flow scheduling algorithms of HOLMES.

As shown in Fig. 7, a TOR switch maintains a flow table and a state table. The two tables work together to execute the load balancing policy attained from the SDN controller. Specifically, when a packet arrives, its flow ID is hashed to map the packet to a flow table entry. If the table entry is valid, the packet is dispatched to the path indicated by the stored hash applied to the packet's flow ID. On the contrary, if the packet's flow ID is not maintained in the flow table, the TOR switch will look up the destination TOR ID in the state table. After that, the (*d*, *m*) policy is applied to compare the load states of the three candidate end-to-end paths to the destination TOR (r1_metric, r2 metric and r3_metric), and assign the packet to the optimal path. Two of the three end-to-end paths are randomly selected. The third one is the optimal path from the last selection. Finally, the flow ID and the hash of the chosen path will be inserted into the flow table.

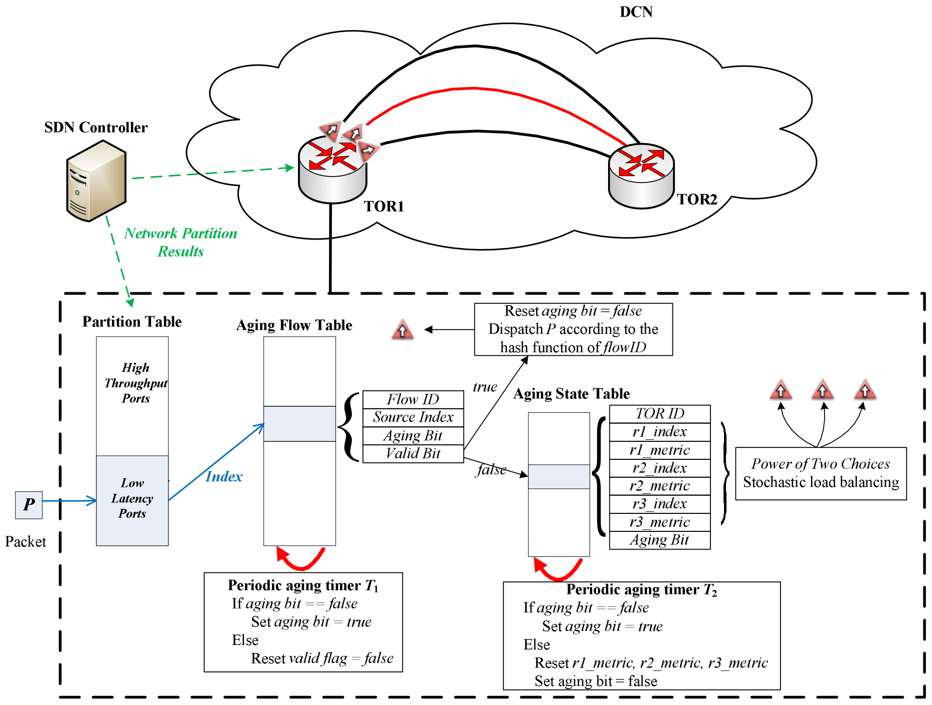


Fig. 7. Overview of HOLMES forwarding and state tables: A new flow table entry is set up by applying the (2, 1) policy in the state table. The period timer of each table is triggered every time period T_1 and T_2 to age out inactive flows and update the load status of each candidate end-to-end path.

The information of each table needs to be updated periodically to keep the real-time status of the traffic and paths. Thus, we associate an aging bit with each table entry. The aging bit of the flow table is responsible for marking inactive or idle flows: when a packet's flow ID maps the information in the forwarding table, the aging bit is cleared to indicate that the flow entry is active. A timer process visits every table entry every aging timeout T_1. When the timer process visits a table entry, it either turns on the aging bit or invalidates the entry if the aging bit is already on. In other words, T_1 is the timeout threshold to age out inactive flows, which is proportional to the size of the scheduling unit e.g. per-flow, per-flow cell, etc. If the packet's flow ID is not maintained in the flow table, the TOR switch will execute the (*d*, *m*) policy on the state table. Thus, T_1 can also be considered as the time period to trigger the execution of the HOLMES scheduling algorithm. On the other hand, another timer process runs together with the aging bit of the state table to update the load status of each candidate end-to-end path. The timeout threshold to age out the old status information in the state table is set to T_2. To ensure that the latest load state information is used when executing the (d, m) policy, the value of T_1 and T_2 should satisfy: $T_1 \geq T_2$. Moreover, in most cases, the global congestion control signals deployed in the flow scheduling algorithms are the feedback signals from the receivers of the end-to-end paths. Thus, we further get: $T_1 \geq T_2 \geq RTT$. Key *et al* [93] have suggested that the policy that periodically sampling a random path and retaining the best paths may perform well.

The periodically sampling of path congestion states in the state table makes the real-time collection of status information becomes a technical challenge. The state collection operations should not introduce obvious transmission overheads and performance penalties. Especially in the TCP in cast scenarios [69, 70] where multiple source nodes transmit traffic at the same time to a common destination node, the state collection operations introduce additional traffic and are prone to cause DCN throughput collapse [71]. A viable solution for collecting the real-time congestion status is deploying RepSYN as the signal to detect the load conditions of the multiple paths, as shown in [33]: before transmitting data among multi-rooted paths, multiple TCP connections are established; however, traffic is only transmitted using the first established connection and the other connections are ended immediately. The delay experienced by an SYN reflects the latest congestion condition of the corresponding path, and thus the congestion states can be collected. Moreover, this solution only replicates SYN packets to probe the network, which does not aggravate the TCP in cast in a DCN.

The state table only needs to periodically maintain the congestion states of two randomly chosen paths and the congestion-optimal path in the latest time unit. Compared with some other congestion-aware solutions e.g., CONGA [10], RepFlow [32], the storage complexity has been dramatically optimized. In order to make the scheduling results of the (d, m) policy more effective, we choose the disjoint end-to-end paths (paths with different intermediate aggregate or spine switches) to avoid the scenario that the same local congestions is shared by multiple end-to-end paths. This implementation is applicable for more complex multi-tier Leaf-Spine topologies or asymmetric DCN topologies.

3. Dealing with the stale information

When implementing HOLMES scheduling algorithm with packet granularity, the transmission latency of a packet is so small that the information refresh rate in the state table cannot catch up with. Correspondingly, the load balancing algorithm has to use the stale information to make the scheduling decisions [94, 98] have pointed out that the delayed information leads to a herd behavior of the scheduling results: data will herd toward a previously light loaded path for much longer time than it takes to fulfill the path. Thus, another technical challenge is to deal with the stale information used in the load balancing algorithms. (More detail including Figure 8 is omitted due to limited space allowed.)

Overall, the simulation experimental results validate the modeling results. They show that HOLMES load balancing algorithm is stable and adaptable in heterogeneous DCNs.

5. Related Work

Latency and throughput optimization for DCN has attracted increasing attention. A series of solutions have proposed to improve the performance of the scale-out multipath data center topologies, such as Clos networks [1, 2, 19], Flattened Butterfly [20], HyperX [21], DragonFly [22], etc. In general, the performance optimization mechanisms can be classified into two categories: temporal solutions (i.e. congestion control mechanisms) and spatial solutions (i.e. load balancing mechanisms). Specifically, one can classify the existing solutions according to Fig. 8.

A. Congestion Control Mechanisms – Temporal Solutions

DC congestion control is a deep studied topic. Generally, the congestion control mechanisms adjust the traffic transmission rates or the packet dropping strategies according to the feedback congestion signals. The control mechanisms can be implemented on either the end hosts or the in-network equipments.

1) Host-based congestion control mechanisms

The optimization of the transport protocols are usually host-based solutions. Those newly proposed transport protocols are customized for DCNs. The host-based control mechanisms can be implemented on either the sender [7, 16] or the receiver of a transportation path [87]. Jain *et al* [88] study the general patterns of response time and throughput of a network as the network load increases. They describe the changing trend of network performance curve using two factors: cliff point and knee point. As shown in Fig. 2 of [88], the point of congestion collapse is defined as a cliff due to the fact that the throughput falls after this point (packets start getting lost); and the point after which the increase in throughput is small (buffers of a path start to be filled) is described as a knee point. Correspondingly, the host-based congestion control policies can also be categorized using the two factors.

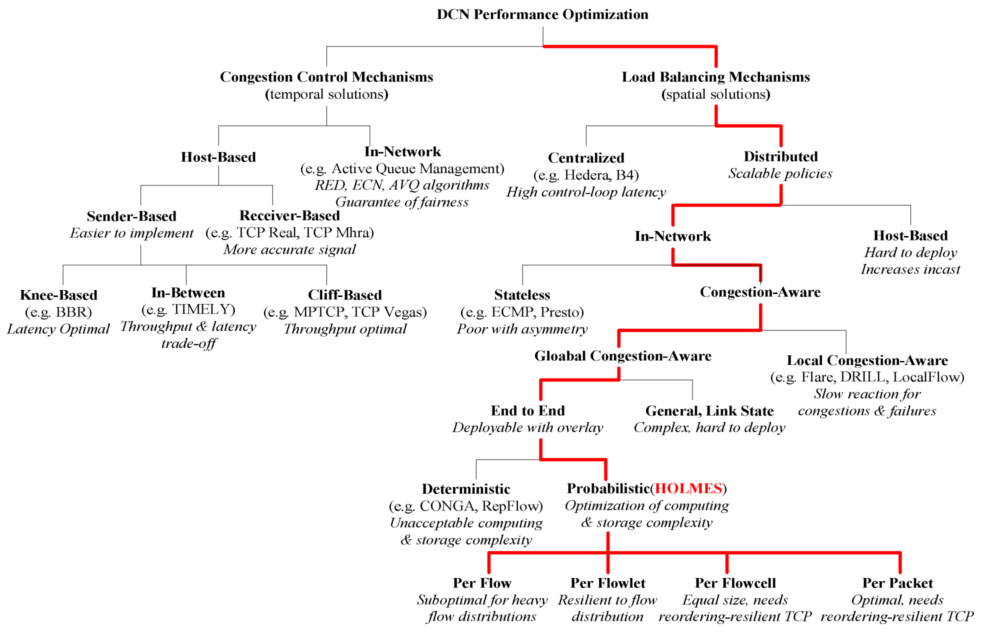

DCN Performance Optimization
Congestion Control Mechanisms (temporal solutions)
Load Balancing Mechanisms (spatial solutions)
Host-Based
In-Network (e.g. Active Queue Management) RED, ECN, AVQ algorithms Guarantee of fairness
Centralized (e.g. Hedera, B4) High control-loop latency
Distributed Scalable policies
Sender-Based Easier to implement
Receiver-Based (e.g. TCP Real, TCP Mhra) More accurate signal
In-Network
Host-Based Hard to deploy Increases incast
Knee-Based (e.g. BBR) Latency Optimal
In-Between (e.g. TIMELY) Throughput & latency trade-off
Cliff-Based (e.g. MPTCP, TCP Vegas) Throughput optimal
Stateless (e.g. ECMP, Presto) Poor with asymmetry
Congestion-Aware
Gloabal Congestion-Aware
Local Congestion-Aware (e.g. Flare, DRILL, LocalFlow) Slow reaction for congestions & failures
End to End Deployable with overlay
General, Link State Complex, hard to deploy
Deterministic (e.g. CONGA, RepFlow) Unacceptable computing & storage complexity
Probabilistic(HOLMES) Optimization of computing & storage complexity
Per Flow Suboptimal for heavy flow distributions
Per Flowlet Resilient to flow distribution
Per Flowcell Equal size, needs reordering-resilient TCP
Per Packet Optimal, needs reordering-resilient TCP

Fig. 8. Classification of DCN congestion control and load balancing mechanisms; the design space for HOLMES flow scheduling algorithm

Cliff-based mechanisms: Most of the modified transportation protocols based on the traditional TCP protocol are cliff-based mechanisms, such as MPTCP [89], DCTCP [7], D2TCP [90], etc. The cliff-based mechanisms are loss-based solutions, which interpret packet losses as congestions and attempt to fulfill the buffers of the TCP paths while avoiding the occurrence of packet losses [66]. These solutions deploy different types of feedback information from the last time point as the congestion signals to guide the traffic control in current time point. For example, DCQCN [49] combines Explicit Congestion Notification (ECN [7]) markings with a QCN [50, 53-55] inspired rate-based congestion based control algorithm to control DCN flows. The cliff-based mechanisms are usually the throughput-optimal solutions.

Knee-based mechanisms: Cardwell *et al* [91] point out that the operating point of the cliff-based mechanisms is not optimal. They argue that when the scale of a DCN is large enough, there will be quantities of packets accumulated in the buffers of a path. Thus, compared with the data transmission time in the network links, the queuing time in the buffers tend to dominate the overall data transmission latency. Thus, the cliff-based solutions are not applicable for the optimization of the data transmission latency. Furthermore, they propose a novel congestion control mechanism BBR, which adjust the operating point from the cliff point to the knee point, to optimize the data transmission latency of a DCN. Therefore, the cliff-based congestion control mechanisms are usually the latency-optimal solutions.

In-between: Different with the above two types of solutions that optimize the throughput or latency of a DCN respectively, a few mechanisms focus on handling the trade-off between latency and throughput, and attempt to find the right balance of the two conflicting factors. Hayes *et al* [17] propose a delay gradient algorithm for TCP congestion control of wide-area networks. Similarly, taking inspiration from Compound [47] and FAST [48], TIMELY [16] also deploys delay gradient as the congestion signal and proposes a gradient-based algorithm to jointly optimize the latency and throughput of a DCN in different time periods. TIMELY mechanism works during the time period between the knee and cliff points, which dynamically adjusts the importance of the throughput and latency issues.

2) In-network congestion control mechanisms

Network congestion usually occurs in the in-network devices, e.g. switches and routers. Thus, compared with the end host based solutions, the in-network congestion control mechanisms achieve more accurate congestion information and react more quickly to congestions and failures. Taking this advantage into account, many researchers migrate some status monitoring and flow control functions from end hosts to in-network devices. Correspondingly, a series of in-network congestion control mechanisms have been proposed.

Most of the in-network congestion control protocols adjust the congestion window size by managing the queues of the DCN routers. Quantities of Active Queue Management (AQM) algorithms have been proposed to generate congestion signals according to the real-time queue lengths in DCN routers, such as Adaptive RED [81, 82], Adaptive Virtual Queue (AVQ) [83], BLUE [84], etc. Hollot *et al* [85] apply classical control system techniques to design novel controllers that are better suited for AQM. Similarly, Firoiu *et al* [86] model the AQM RED algorithm as a feedback control system and discover fundamental laws governing the traffic dynamics in TCP/IP networks. pFrabic [51] preemptively schedules flows using packet forwarding priorities in switches and Detail [9] deploys a similar mechanism that give priorities to latency-sensitive mouse flows; however this simple control mechanism makes a mismatch between injection traffic and network capacity, results in packet loss and bandwidth wastage. These in-network solutions react quickly to the real-time congestions and failures; moreover, they can also generate feedback congestion signals to the end hosts, and cooperate with the host-based control mechanisms.

B. Load Balancing Schemes – Spatial Solutions

Different with the congestion control mechanisms, the load balancing schemes try to improve the DCN performance from a spatial aspect. This kind of solutions is especially applicable for the traditional multipath DCN topologies.

Similarly to the traditional traffic engineering techniques, some studies deploy the centralized scheme to schedule the DC traffic. SWAN [27] and B4 [28] collect statistical information from switches to a central controller, and push forwarding rules to balance the load for inter-datacenter WANs. Fastpass [42] deploys a centralized scheduling algorithm to ensure that the queues stay small while the queuing delay remains near optimal. Hedera [4] and MicroTE [29] also apply the centralized scheduling scheme and focus on the load balancing across multiple DC paths.

The main shortcoming of these centralized solutions is that they suffer from high control-loop latency in large-scale data centers, which are not applicable for handling highly volatile DC traffic in time [14]. Addressing this issue, quantities of scalable distributed load balancing schemes have been proposed. One can further categorize these solutions as stateless solutions and congestion-aware solutions.

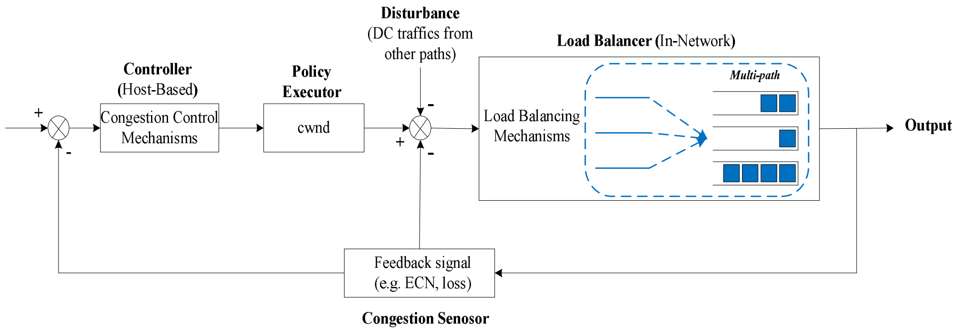


Fig. 9. Correlations between the DCN congestion controller and load balancer in a control theoretic model

1) Stateless load balancing schemes

ECMP [3] is a simple hash-based load balancing scheme that is widely used as the de facto scheme in switch ASICs today. The coarse-grained per-flow load balancing and the congestion agnostic hash collisions in ECMP have shown to cause performance degradation in asymmetric DCN topologies [10, 25, 26], during link failures.

To overcome the above-mentioned shortcomings in ECMP, quantities of solutions have been proposed to improve the traffic splitting granularity or the load balancing algorithm. PLTS [23] and DRB [5] is per-packet load balancing schemes that schedule DC traffic with the granularity of the packet. Presto [24] splits traffic into 64KB sized TSO (TCP Segment Offload) segments. Round robin fashion [6] is deployed in DRB and Presto to spray DC packets or flow cells. Based on Valiant Load Balancing (VLB [37]), some other solutions have been put forwarded to improve the failure tolerance of homogeneous and heterogeneous network topologies [38, 39].

None of the above solutions are state-aware, which causes performance degradation during link failures.

2) Congestion-ware load balancing schemes

The main drawback of the stateless schemes is causing performance degradation during link failures. Addressing this issue, a series of congestion-aware load balancing schemes have been proposed. Based on global or local congestion information, the congestion-aware solutions are more applicable for asymmetric topologies or link/switch failure scenarios.

Global congestion aware schemes: Global congestion-aware load balancing schemes deploy the end-to-end congestion signal as the feedback metric to schedule the DC traffic among multiple paths. TexXCP [30] and MATE [31] are adaptive traffic-engineering proposals that balance the load across multiple ingress-egress paths in the wide-area network, using the per-path congestion statistics. CONGA [10] proposes similar solutions for datacenters, by spraying DC traffic among multi-rooted networks based on the congestion state of each end-to-end DC path. RepFlow [32] and RepNet [33] replicate each mouse flow to opportunistically use the less congested path, and reduce the overall flow completion time in DCNs. Inspired by the Minimum Delay Routing [34], HALO [35] studies load-sensitive adaptive routing and implements its solution in the router software. These solutions are aware of the overall congestion status of the DCN and react fast for local failures or congestions.

Local congestion aware schemes: Using the global congestion-ware schemes to make load balancing decisions faces scalability challenges. Although the distributed architecture can improve the scalability of the scheduling schemes, they require coordination between switches or hosts. In large-scale data centers with high transmission rates, the continuously reacts to each congestion information introduce additional latencies and degrade the overall flow scheduling performance [13]. Consequently, the local congestion-aware solutions have drawn large interests. Local Flow [25] and Flare [36] study the switch-local solutions that balance the load on switch ports, without taking the global congestion information into account. Based on the Power-of-Two-Choices model [43, 44], Ghorbani *et al* [13] propose a stochastic switch-local scheduling scheme that further reduces the polling range of local solutions, and improves the execution rate of the flow scheduling algorithm.

As an improvement of CONGA[10], HULA [14] tracks the next hop for the best path and its corresponding utilization for a given destination, instead of maintaining per-path utilization congestion information. This novel strategy makes the load balancing scheme applicable for more complex DCN topologies, besides the two-tier Leaf-Spine topologies. Based on limited congestion information, the local congestion-aware solutions provide suboptimal routing decisions, while improve the overall policy execution rate. However, the stability and convergence of these switch-local solutions need to be ensured; worse more, as aforementioned, the local congestion-ware scheduling policies have been proved to react slowly to link failures [10, 14] and are prone to form congestion trees.

The implementation of the deterministic congestion-aware load balancing schemes requires recording the real-time load status of all the available paths or links, to make the global or local optimal choices. As previously discussed in Section V, keeping the status information and calculating the optimal solution in large-scale datacenters introduce unacceptable storage and computing complexity. Taking inspiration from this issue, we deploy a probabilistic global congestion-aware load balancing algorithm in HOLMES, to optimize the storage complexity while improve the execution rate of the load balancing algorithm.

C. Correlations between the Temporal and Spatial Solutions

Both the temporal congestion control mechanisms and the spatial load balancing mechanisms aim to optimize the throughput or latency of a DCN. Next, we try to describe the correlations of the two types of solutions. Hollot *et al* [85], apply the classical control system techniques to design controllers and analyze the stability of the same network system under different congestion control mechanisms. Since the publication of the first seminal paper [92] by Kelly *et al*, the framework of Network Utility Maximization (NUM) has been widely applied in network resource allocation algorithms as well as the congestion control protocols. Inspired by these solutions, we design a control theoretic model to describe the correlations between the congestion control mechanisms and the load balancing mechanisms.

As shown in Fig. 9, the closed-loop based control mechanisms are usually host-based mechanisms, which are generated in an end controller and will be executed later in the in-network devices. A congestion control mechanism typically operates on a single end-to-end path, and it concentrates on the performance optimization of one path. The main disturbance during the policy execution process is the traffic from other multi-rooted paths.

The load balancing mechanisms focus on the multi-path scenarios, and schedule the traffic of multiple paths to improve the overall performance of all the end-to-end paths. Either solution needs a feedback signal to guide the traffic scheduling in the upcoming control loop. Thus, the feedback signal will be transmitted to both the congestion controller and the load balancer after each transmission loop. The load balancing mechanisms are often implemented together with the congestion control policies. For example, MPTCP enables the parallelized TCP transmissions among multiple paths using its load balancing policies. It still realizes the traffic congestion control using the traditional TCP congestion avoidance algorithms. Therefore, it can be considered as part of the closed-loop control system in Fig. 9.

D. Architecture Improvements

Some other researchers also try to improve the architecture of the scheduling schemes based on the DC traffic patterns or application features. Freeway [18] dynamically partitions the multiple DCN paths into low latency and high throughput paths, and schedules the elephant and mouse flows separately. DevoFlow [52] uses multipath and changes the design of OpenFlow switches to enable easier flow aggregation, improving DCN latency and throughput. ADN [77] devolves into the application level, which concentrates on serving the up layer applications. It deploys a novel architecture that slices the whole DCN into logically isolated sub-networks to serve different types of applications. The architecture improvements can be implemented together with the aforementioned load balancing and congestion control mechanisms, to provide a more comprehensive performance optimization scheme for DCNs [106, 108].

6. Conclusion

This paper presents HOLMES, a novel DCN flow scheduling scheme, which tackles mixed (mice vs. elephants) data center traffic. Using a stochastic performance model, we first prove the necessity of isolating mice and elephants with a closed form. We then present the HOLMES architecture that partitions a DCN into high-throughput and low-latency sub-networks. We further design a stochastic and global congestion-aware load balancing algorithm that schedules the corresponding DC traffic to each sub-network. Simulation results show that HOLMES network partition policy can successfully eliminate the interference between the mouse and elephant data flows. Finally, we prove that HOLMES flow scheduling algorithm is stable and scalable for large-scale data centers.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," In ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63-74.

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, P. Patel, and S. Sengupta. "VL2: A Scalable and Flexible Data Center Network," In ACM SIGCOMM Computer Communication Review, 2009, 39(4): 51-62.

[3] C. Hopps. Analysis of an Equal-Cost Multi-Path algorithm. RFC 2992, Network Working Group, 2000.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," In Proc. USENIX NSDI, 2010, Vol. 10, pp. 19-19.

[5] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet load-balanced, low-latency routing for Clos-based data center networks," In Proc. ACM CoNEXT, 2013, pp. 49-60.

[6] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," In Proc. IEEE INFOCOM, 2013, pp. 2013-2018.

[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," In ACM SIGCOMM Computer Communication Review, 2010, 40(4): 63-74.

[8] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. M. B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," In ACM SIGCOMM Computer Communication Review, 2013, 43(4): 435-446.

[9] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," In ACM SIGCOMM Computer Communication Review, 2012, 42(4): 139-150.

[10] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," In ACM SIGCOMM Computer Communication Review, 2014, 44(4): 503-514.

[11] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted cost multipathing for improved fairness in data centers," In In Proc. ACM EuroSys, 2014.

[12] P. Wang, H. Xu, Z. Niu and D. Han, "Expeditus: Distributed congestion-aware load balancing in clos data center networks," In Proc. ACM CoNEXT on Student Workshop, 2014, pp. 1-3.

[13] S. Ghorbani, B. Godfrey, Y. Ganjali and A. Firoozshahian, "Micro load balancing in data centers with DRILL," In Proc. ACM HotNets, 2015, pp. 17.

[14] N. Katta, M. Hira, C. Kim, A. Sivaraman and J. Rexford, "HULA: Scalable load balancing using programmable data planes," In Proc. ACM SOSR, 2016.

[15] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," In IEEE Transactions on Services Computing, 2010, 3(4): 266–278.

[16] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall and D. Zats, "TIMELY: RTT-based congestion control for the datacenter," In ACM SIGCOMM Computer Communication Review, 2015, 45(4): 537-550.

[17] D. A. Hayes and G. Armitage, "Revisiting TCP congestion control using delay gradients," In Proc. IFIP Networking, 2011.

[18] W. Wang, Y. Sun, K. Salamatian and Z. Li, "Adaptive path isolation for elephant and mouse flows by exploiting path diversity in datacenters," In IEEE Transactions on Network and Service Management, 2016, 13(1): 5-18.

[19] V. Liu, D. Halperin, A. Krishnamurthy and T. Anderson, "F10: a fault-tolerant engineered network," In Proc. USENIX OSDI, 2010.

[20] J. Kim, W. J. Dally, S. Scott and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," In ACM SIGARCH Computer Architecture News, 2007, 35(2): 126-137.

[21] J. H. Ahn, N. Binkert, A. Davis, M. McLaren and R. S. Schreiber, "HyperX: topology, routing and packaging of efficient large-scale networks," In Proc. ACM SC, 2009, p. 41.

[22] J. Kim, W. J. Dally, S. Scott and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," In ACM SIGARCH Computer Architecture News, 2008, 36(3): 77-88.

[23] A. Dixit, P. Prakash and R. R. Kompella, "On the Efficacy of Fine-Grained Traffic Splitting Protocols in Data Center Networks," In ACM SIGCOMM Computer Communication Review, 2011, 41(4): 430-431.

[24] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," In ACM SIGCOMM Computer Communication Review, 2015, 45(4): 465-478.

[25] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," In Proc. ACM CoNEXT, 2013, pp. 151-162

[26] T. Edsall, A. Fingerhut, T. Lam, R. Pan, and G. Varghese, "Flame: Efficient an robust hardware load balancing for data center routers," [Department of Computer Science and Engineering], University of California, San Diego, 2012.

[27] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," In ACM SIGCOMM Computer Communication Review, 2013, 43(4): 15-26.

[28] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," In ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3-14.

[29] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," In Proc. ACM CoNEXT, 2011, pp. 8:1-8:12.
[30] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," In ACM SIGCOMM Computer Communication Review, 2005, 35(4): 253-264.
[31] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "Mate: Mpls adaptive traffic engineering," In Proc. IEEE INFOCOM, 2001, pp. 1300-1309.
[32] H. Xu and B. Li, "RepFlow: Minimizing flow completion times with replicated flows in data centers," In Proc. IEEE INFOCOM, 2014, pp. 1581-1589.
[33] S. Liu, W. Bai, H. Xu, K. Chen and Z. Cai "RepNet: Cutting tail latency in data center networks with flow replication," In Computer Science, 2014.
[34] R. Gallager, "A minimum delay routing algorithm using distributed computation," In IEEE Transactions on Communications, 1977, 25:73-85.
[35] N. Michael and A. Tang, "Halo: Hop-by-hop adaptive link-state optimal routing," In IEEE/ACM Transactions on Networking, 2015, 23(6): 1862-1875.
[36] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," In ACM SIGCOMM Computer Communication Review, 2007, 37(2): 51-62.
[37] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," In Proc. ACM STOC, 1981, pp. 263-277.
[38] R. Zhang and N. McKeown, "Designing a fault-tolerant network using valiant load balancing," In Proc. IEEE INFOCOM, 2008.
[39] R. Zhang and N. McKeown, "Designing a predictable internet backbone with valiant load-balancing," In Proc. IEEE/ACM IWQoS, 2005, pp. 178-192.
[40] D. Wischik, C. Raiciu, A. Greenhalgh , and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," In Proc. USENIX NSDI, 2011, Vol. 11, pp. 8-8.
[41] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," In ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63-74.
[42] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," In ACM SIGCOMM Computer Communication Review, 2014, 44(4): 307-318.
[43] M. Mizenmacher, "The power of two choices in randomized load balancing," In IEEE Transactions on Parallel and Distributed Systems, 2001, 12(10): 1094-1104.
[44] Y. T. He and D. G. Down, "Limited choice and locality considerations for load balancing," In Performance Evaluation, 2008, 65(9): 670-687.
[45] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," In Proc. ACM IMC, 2010, pp. 267-280.
[46] S. Kandula, S. Sengupta, A. G. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," In Proc. ACM IMC, 2009, pp. 202-208.
[47] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," In Proc. IEEE INFOCOM, 2006.
[48] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," In IEEE/ACM Transactions on Networking (ToN), 2006, 14(6): 1246-1259.
[49] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RMDA deployments," In ACM SIGCOMM Computer Communication Review, 2015, 45(4): 523-536.
[50] IEEE. 802.1Qau - Congestion Notification. http://www.ieee802.org/1/pages/802.1au.html.
[51] M. Alizadeh, S. Yang, S. Shanif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFrabic: Minimal near-optimal datacenter transport," In ACM SIGCOMM Computer Communication Review, 2013, 38(4): 435-446.
[52] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," In ACM SIGCOMM Computer Communication Review, 2011, 41(4): 254-265.
[53] D. G. Kendall, "On the generalized birth-and-death process," In the annals of mathematical statistics, 1948: 1-15.
[54] R. M. Dudley, "Uniform central limit theorems," Cambridge: Cambridge university press, 1999, Vol. 23.
[55] J. D. C. Little, and S. C. Graves, "Little's law," In Building intuition. Springer US, 2008, pp. 81-100.

[56] P. Kumar, and S. Meyn, “Stability of queuing networks and scheduling policies,” In IEEE Transactions on Automatic Control, 1995, 40(2): 251-260.
[57] A. Mekkittikul, and N. McKeown, “A practical scheduling algorithm to achieve 100% throughput in input-queued switches,” In Proc. IEEE INFOCOM, 1998.
[58] H. C. Lin, and C. S. Raghavendra, “An approximate analysis of the join the shortest queue (JSQ) policy,”, In IEEE Transactions on Parallel and Distributed Systems, 1996, 7(3): 301-307.
[59] R. D. Foley, and D. R. McDonald, “Join the shortest queue: stability and exact asymptotics,” In Annals of Applied Probability, 2001: 569-607.
[60] C. Xu, J. Yang, K. Yin, and H. Yu, “Optimal construction of virtual networks for Cloud-based MapReduce workflows,” In Elsevier Computer Networks, 2017, 112: 194-207.
[61] W. Cheng, F. Ren, W. Jiang, K. Qian, T. Zhang, and R. Shu, “Isolating Mouse and Elephant in Data Centers,” arXiv preprint arXiv: 1605.07732, 2016.
[62] J. Padhye, V. Firoiu, D. F. Towsley, J. F. Kurose, “Modeling TCP Reno performance: a simple model and its empirical validation,” In IEEE/ACM Transactions on Networking (ToN), 2000, 8(2): 133-145.
[63] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny. “Workload analysis of a large-scale key-value store” In ACM SIGMETRICS Performance Evaluation Review, 2012, 40(1): 53-64.
[64] A. Shaikh, J. Rexford, and K. G. Shin, “Load sensitive routing of long-lived ip flows,” In Proc. ACM SIGCOMM, 1999.
[65] T. Benson, A. Anand, A. Akella, and M. Zhang, “Understanding data center traffic characteristics,” In ACM SIGCOMM Computer Communication Review, 2010, 40(1): 92-99.
[66] J. Gettys, and K. Nichols, “Bufferbloat: Dark buffers in the internet,” In Queue 2011, 9 (11): 40.
[67] J. Zhang, K. Xi, L. Zhang, and H. J. Chao, “Optimizing network performance using weighted multipath routing,” In Proc. IEEE 21st International Conference on Computer Communications and Networks (ICCCN), 2012, pp 1-7.
[68] Y. Geng, V. Jeyakumar, A. Kabbani A, and M. Alizadeh, “J uggler: a practical reordering resilient network stack for datacenters,” In Proc. ACM Eurosys’16, 2016, pp. 20.
[69] J. Zhang, F. Ren, C. Lin, “Modeling and understanding TCP incast in data center networks,” In Proc. IEEE INFOCOM, 2011, pp. 1377-1385.
[70] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, “Understanding TCP incast throughput collapse in datacenter networks,” In Proc. 1st ACM workshop on Research on enterprise networking, 2009, pp. 73-82.
[71] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, “Safe and effective fine-grained TCP retransmissions for datacenter communication,” In ACM SIGCOMM Computer Communication Review, 2009, 39(4): 303-314.
[72] A. W. Moore, and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” In ACM SIGMETRICS Performance Evaluation Review, 2005, 33(1): 50-60.
[73] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification,” In Proc. ACM IMC, 2004, pp. 135-148.
[74] L. Bernaille, R. Teixeira, I. Akodjenou, “Traffic classification on the fly,” In ACM SIGCOMM Computer Communication Review, 2006, 36(2): 23-26.
[75] V. Jalaparti, P. Bodik, I. Menache, R. Rao, K. Makarychev, and M. Caesar, “Network-aware scheduling for data-parallel jobs: Plan when you can,” In ACM SIGCOMM Computer Communication Review, 2015, 45(4): 407-420.
[76] B. Wang, J. Jiang, and G. Yang, “ActCap: Accelerating mapreduce on heterogeneous clusters with capability-aware data placement,” In Proc. IEEE INFOCOM, 2015, pp. 1328-1336.
[77] Y. Wang, D. Lin, C. Li, J. Zhang, P. Liu, C. Hu, and G. Zhang, “Application driven network: providing on-demand services for applications,” In Proc. ACM SIGCOMM, 2016, pp. 617-618.
[78] W. Sun, “Internet of vehicles,” In Advances in Media Technology, 2013: 47.
[79] M. Chowdhury, Y. Zhong, I. Stoica, “Efficient coflow scheduling with varys,” In ACM SIGCOMM Computer Communication Review, 2014, 44(4): 443-454.
[80] S. Luo, H. Yu, Y. Zhao, S. Wang, S. Yu, and L. Li, “Towards practical and near-optimal coflow scheduling for data center networks,” In IEEE Transactions on Parallel and Distributed Systems, 2016, 27(11): 3366-3380.
[81] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, “Techniques for eliminating packet loss in congested TCP/IP networks,” In Ann Arbor, 1997, 1001: 63130.
[82] M. May, T. Bonald, and J. C. Bolot, “Analytic evaluation of RED performance,” In Proc. IEEE INFOCOM, 2000, 3: 1415-1424.

[83] S. Kunniyur, and R. Srikant, “Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management,” In ACM SIGCOMM Computer Communication Review, 2001, 31(4): 123-134.
[84] W. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, “The BLUE active queue management algorithms,” In IEEE/ACM transactions on networking, 2002, 10(4): 513-528.
[85] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, “On designing improved controllers for AQM routers supporting TCP flows,” In Proc. IEEE INFOCOM, 2001, 3: 1726-1734.
[86] V. Firoiu, and M. Borden, “A study of active queue management for congestion control,” In Proc. IEEE INFOCOM, 2000, 3: 1435-1444.
[87] C. Zhang, V. Tsaoussidis, “TCP-real: improving real-time capabilities of TCP over heterogeneous networks,” In Proc. ACM 11th international workshop on Network and operating systems support for digital audio and video, 2001, pp. 189-198.
[88] R. Jain, K. K. Ramakrishnan, and D. M. Chiu, “Congestion avoidance in computer networks with a connectionless network layer,” In arXiv preprint cs/9809094, 1998.
[89] C. Raiciu, M. Handley, and D. Wischik, “Coupled congestion control for multipath transport protocols,” RFC 6356 (Experimental), 2011.
[90] B. Vamanan, J. Hasan, and T. N. Vijaykumar, “Deadline-aware datacenter tcp (d2tcp),” In ACM SIGCOMM Computer Communication Review, 2012, 42(4): 115-126.
[91] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, “BBR: Congestion based congestion control,” In Queue, 2016, 14(5): 50.
[92] F. P. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: Shadow prices, proportional fairness and stability,” In J. Oper. Res., 1998, 49(3): 237-252.
[93] P. Key, L. Massoulie, and D. Towsley, “Path selection and multipath congestion control,” In Communications of the ACM, 2011, 54(1): 109-116.
[94] M. Mitzenmacher, A. W. Richa, and R. Sitaraman, “The power of two random choices: a survey of techniques and results,” In Combinatorial Optimization, 2001, 9: 255-304.
[95] A. Fox, S. Gribble, Y. Chawathe, E. Brewer, and P. Gauthier, "Cluster-Based Scalable Network Services." In Proceedings of the 16th ACM Symposium on Operating Systems Principles, 1997, pp. 78-91.
[96] P. Key, L. Massoulié, and D. Towsley, “Combined multipath routing and congestion control: a robust internet architecture,” In TechReport MSR-TR.–Microsoft Research, 2005, 361.
[97] P Key, L. Massoulié, D. Towsley, “Multipath routing, congestion control and dynamic load balancing,” In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, 2007, 4: IV-1341-IV-1344.
[98] M. Dahlin, “Interpreting stale load information,” In IEEE Transactions on parallel and distributed systems, 2000, 11(10): 1033-1047.
[99] OMNET++ discrete event simulator, https://omnetpp.org/.
[100] Wang, G., Andersen, D.G., Kaminsky, M., Papagiannaki, K., Ng, T.S., Kozuch, M. and Ryan, M., 2010, August. c-Through: Part-time optics in data centers. In ACM SIGCOMM Computer Communication Review (Vol. 40, No. 4, pp. 327-338). ACM.
[101] Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H.H., Subramanya, V., Fainman, Y., Papen, G. and Vahdat, A., 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. ACM SIGCOMM Computer Communication Review, 40(4), pp.339-350.
[102] Roy, A., Zeng, H., Bagga, J., Porter, G. and Snoeren, A.C., 2015, August. Inside the social network's (datacenter) network. In ACM SIGCOMM Computer Communication Review (Vol. 45, No. 4, pp. 123-137). ACM.
[103] Hussein A. Mohammed, Adnan Hussein Ali, Hawraa Jassim Mohammed. The Affects of Different Queuing Algorithms within the Router on QoS VoIP application Using OPNET. International Journal of Computer Networks & Communications (IJCNC) Vol.5, No.1, January 2013.
[104] Faiza Al-Salti, N. Alzeidi, Khaled Day, Bassel Arafeh and Abderezak Touzene. Grid-based Priority Routing Protocol for UWSNS. International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.6, November 2017.
[105] Deepa Dasarathan and P. Nirmal Kumar. Multicasting Based Enhanced Proactive Source Routing in MANETS. International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.6, November 2017.

[106] Torsten Teubler, Dennis Pfisterer and Horst Hellbrück. Memory Efficient Forwarding Information Base for Content-Centric Networking. International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.3, May 2017.
[107] Raghavendra M. and Pallapa Venkataram. ECA Model Based QoS AODV Routing for MANETS. International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.3, May 2017.
[108] Shinichi Kuribayashi. Flexible Virtual Routing Function Deployment in NFV-based Network with Minimum Total Network Cost. International Journal of Computer Networks & Communications (IJCNC) Vol.8, No.5, September 2016.
[109] Tripti Sharma1 and Vivek Kumar. Congestion Aware Link Cost Routing for MANETS. International Journal of Computer Networks & Communications (IJCNC) Vol.8, No.3, May 2016.
[110] Hamzah M A Hijawi and Mohammad M. N. Hamarsheh. Performance Analysis of Multi-Path TCP Network. International Journal of Computer Networks & Communications (IJCNC) Vol.8, No.2, March 2016.
[111] Amnah El-Obaid. Broadcast Wormhole-routed 3-D Mesh Networks. International Journal of Computer Networks & Communications (IJCNC) Vol.7, No.4, July 2015.
[112] Ahmed Y. Hamed and Ghazi Al-Naymat. A Generic Algorithm for Constructing Broadcast Trees with Cost and Delay Constraints in Computer Networks. International Journal of Computer Networks & Communications (IJCNC) Vol.7, No.1, January 2015.

AUTHORS

Mr. Tim Tingqiu Yuan (ytq@huawei.com) is a Vice President of Central Research Institute, Huawei Technologies. He has been responsible for research and development as well as technology planning for over 20 years at Huawei. His research interests include, but not limited to: packet-based networking, information centric networking, software-defined networking, future Internet, future terminal devices, artificial intelligence and machine learning.

Mr. Tao Huang (huangtao@huawei.com) is a Senior Expert of Central Research Institute, Huawei Technologies. Over 20 years, he has been instrumental to many R&D projects and products at Huawei.

Dr. Cong Xu(xucong@huawei.com) received his Ph.D. degree in Computer Science from Tsinghua University, Beijing, P.R.China, in 2015. He is now a researcher at Huawei Technologies. His research interests include data center, cloud computing, big data, block chain, etc.

Dr. Jian Li (jian.li1@huawei.com) works on research technology planning at Huawei Technologies. Before joining Huawei, he was an executive architect with IBM as well as a research scientist with IBM Research. He earned a Ph.D. in electrical and computer engineering from Cornell University.