CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:5735–5748
Published online 13 October 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3650

SPECIAL ISSUE PAPER

Congestion-free routing strategy in software defined data center networks

Yan Li[1], Wenxin Li[2], Honghui Chen[1,*,†], Deke Guo[1,*,†], Tian Zhang[3] and Ting Qu[1]

[1]*Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China*
[2]*School of Computer Science and Technology, Dalian University of Technology, Dalian City 116024, China*
[3]*School of Information Management, Wuhan University, Wuhan 430072, China*

SUMMARY

Large-scale online services and distributed execution engines (i.e., MapReduce and Dryad) generate large volumes of traffic in data center networks. As a consequence, significant congestion can occur in the data center network. To the best of our knowledge, most existing approaches either focus on local congestion-aware mechanisms, which have only a poor ability to handle asymmetry or use explicit congestion notification packets, which are difficult to implement directly in switch hardware. These methods are insufficient to solve the congestion problem. In this paper, we focus on a congestion-free routing strategy, resorting to the global view of the data center network in a software-defined networking controller. Specifically, a timeslot allocation was first conducted for the coming packets, and then the corresponding routing paths were computed for each packet. In view of the efficiency, the timeslot allocation algorithm follows a heuristic pattern, and the path selection is modeled as a bin-packing problem. Simulation results showed that the congestion-free routing strategy proposed here performs well in throughput, queuing, and end-to-end round-trip time. Copyright © 2015 John Wiley & Sons, Ltd.

Received 22 July 2015; Accepted 25 July 2015

KEY WORDS: congestion; data center network; SDN

1. INTRODUCTION

Data centers, the underlying infrastructure of today's cloud computing, not only provide different kinds of large-scale online services but also host various back-end computation tasks, such as MapReduce [1] and Dryad tasks [2]. Because these applications have a large volume of traffic to be transported among a large number of servers in data centers, this traffic is likely to lead to significant congestion in the data center network. Moreover, such congestion can become worse with the constant increasing demand of these applications.

Taking advantage of the queues in the switches, some existing congestion control-based approaches focus on low queue occupancy using explicit congestion notification [3], such as data center transmission control protocol (DCTCP), high-bandwidth ultra-low latency, and low-latency data center transport [4–6]. Others rely on local selection of no congestion paths for the arriving packets [7]. However, these methods are not suitable for a congestion-free data center network. Generally, the drawbacks of conventional approaches mainly involve the following two facts: First, explicit congestion notification is hard to be implemented in hardware. Second, locally selecting no congestion paths can be inefficient in an asymmetry network.

*Correspondence to: Honghui Chen and Deke Guo, Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.
†E-mail: 248856416@qq.com, guodeke@gmail.com

For these reasons, the present work focused on a congestion-free routing strategy in data center networks in this paper. To make an optimal routing strategy with no-queuing and no-waiting, global network control is needed. Fortunately, such central network control is becoming technically feasible with the success of software-defined networking (SDN) controller, which has been widely applied in the data center networks [8]. Therefore, aiming to eliminate congestion, a feasible congestion-free routing strategy can be realized by carefully making decisions regarding when to schedule each packet and when to select the corresponding routing path.

The SDN controller can be used to make decisions regarding timeslot allocation and routing path selection for each packet to prevent congestion in the data center network. In order to achieve this goal, two stages were considered. The first stage involves executing a timeslot allocation algorithm to find a maximal matching. All the selected endpoints can be used to establish communication when they satisfy all the input and output bandwidth constraints in one timeslot. The second stage is to assign routing paths for packets transported among the selected endpoints in each timeslot. Specifically, the path assignment problem was modeled as a bin-packing problem. Here, bandwidth resources are modeled as bins, and packets from m items of different sizes were modeled as objects. These multi-dimensional bin-packing problems were non-deterministic polynomial-time hard (NP-hard) [9]. It was here approximated using a greedy algorithm, which blended the advantages of both the next fit heuristic (NF) algorithm and simple genetic algorithm (SGA). Finally, comprehensive experiments were conducted to demonstrate the efficiency of the proposed congestion-free routing strategy.

In summary, the contributions of this paper were demonstrated as follows:

- Focusing on the congestion problem in a data center network, a congestion-free routing strategy was here proposed by taking advantage of an SDN controller. Such an SDN controller makes decisions for each packet regarding timeslot allocation and routing path selection.
- The underlying key stages of congestion-free routing strategy are timeslot allocation and path assignment. To efficiently solve these problems, heuristic algorithms were used to produce an approximate solution. For timeslot allocation, a fast data structure, that is, the bitmap table, was use to speed up the algorithm. For path assignment, both the NF algorithm and SGA were used to select a path.
- The algorithm was evaluated using detailed packet-level experiments in comparison with transmission control protocol (TCP) and DCTCP. The experimental results showed that the proposed congestion-free routing strategy performed better than DCTCP on reducing the switch queue size and round-trip time (RTT). The proposed method achieved a throughput similar to that of the TCP method.

The rest of this paper is organized as follows. In Section 2, an overview of the congestion-free routing design is presented. In Section 3, timeslot allocation is shown. In Section 4, path selection algorithms are presented. In Section 5, the experimental evaluation is discussed. Further discussion is provided in Section 6. Related work is presented in Section 7. Finally, this paper is concluded in Section 8.

2. OVERVIEW OF CONGESTION-FREE ROUTING STRATEGY

To design a congestion-free routing strategy, a discrete time-slotted system where the timeslot length can range from hundreds of milliseconds to minutes is considered [10, 11]. In each timeslot t, a number of flows generated by the applications arrive at the data center, and each packet of flows must be forwarded to the destination without causing any congestion in the data center. Hence, a congestion-free routing strategy is designed by taking advantage of an SDN controller to monitor the traffic and has a central network control [12, 13]. Consequently, the key design is to assign timeslots and routing paths for each packet. As shown in Figure 1, the SDN controller contains three main functions: timeslot allocation, path selection, and SDN communication. Through communicating with switches, the SDN controller determined overall network situations as well as informed switches of the network modification. Once an endpoint (source) called for a transmission, the operating system sent this demand to the SDN controller. The source provided the controller with two pieces

DOI: 10.1002/cpe

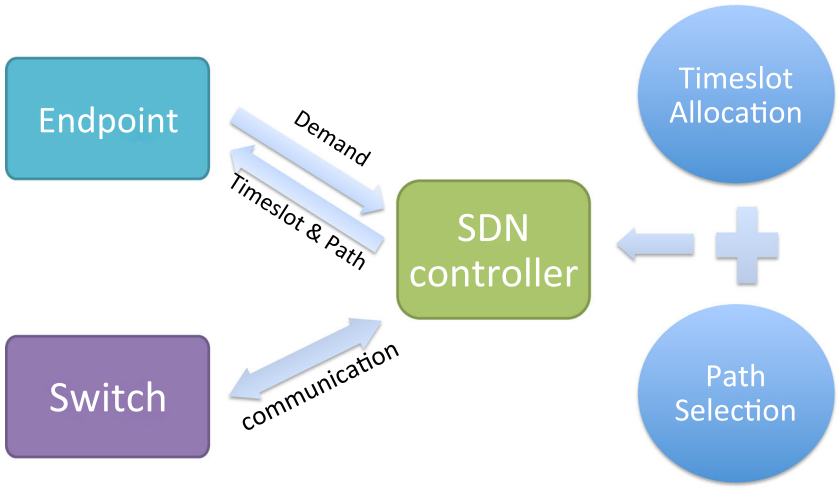


Figure 1. The overview of congestion-free routing function on software-defined networking (SDN) controller, which allocates timeslot to each flow, computes the routing path, and interacts with the source.

of information: (i) the flow size (FlowSize) and (ii) destination. Based on this, the controller allocated timeslots to each packet, computed the routing paths, and returned them to the corresponding sources.

As a general view, the most important design principle behind congestion-free routing strategy is precise transfer time control, and spare paths choose, which we divided into the following:

- Timeslot allocation. The SDN controller ran the timeslot allocation algorithm and assigned the requester a set of timeslots in which to transmit its data and also keep track of the source-destination pairs assigned to each timeslot.
- Path selection. After assigning timeslots, the controller selected available paths for each requester in every timeslot and returned this information to the source.

Given the aforementioned guidelines, the essential workings of congestion-free routing strategy are as follow. Every source periodically contacted the controller to retrieve its transmission time and paths and transmits its packets until all data are finished. Because the controller knew about all current and scheduled transfers, it could choose timeslots and paths to guarantee the transmission of data without delays and congestion in the data center network. One key benefit of congestion-free routing is that there is no need to make any changes to the switches. The main reason is that the endpoints' transmissions should occur at the times prescribed by the controller, and there should be some hardware support in Network Interface Card (NIC)'s of endpoints [14].

3. TIMESLOT ALLOCATION SCHEME

In this section, the design of the proposed timeslot allocation algorithm is described in detail, and then the performance of the algorithm is analyzed.

3.1. Timeslot allocation design

We consider a discrete time-slotted system where time is slotted. In timeslot t, each endpoint can transmit or receive at most one Maximum Transmission Unit (MTU)-sized packet. Thus, when a large amount of flow arrives at the data center network, the first task is to allocate transfer time for them to avoid congestion. Here, timeslot allocation was conducted at the packet level.

A congestion-free routing strategy design should ensure that traffic can only be routed in a congestion-free way and that it will satisfy both the input and output bandwidth constraints. Traditionally, today's data center networks are often organized into tiers, with tiers having various

DOI: 10.1002/cpe

bandwidth constraints, such that the total amount of bandwidth required by allocated matchings must not exceed both the input and output bandwidth capacities of each tier.

Here, the optimal timeslot allocation problem (TAP) was mathematically formulated using two or more tier resources and multiple packet classes. The objective of the TAP was to maximize the *matchings* between endpoints in a given timeslot with the bandwidth constraints satisfied, such that congestion cannot occur.

Consider a network with N endpoints where endpoints are denoted by $1, \ldots, N$. Each endpoint potentially has requests for any other endpoint. Let $D_{ij}(t)$ denote the total demand of packets waiting to be transferred from endpoint i to endpoint j at timeslot t. D_{ij} can be described using a data structure: (i, j, and $timeslots$' need). Figure 2 shows the demand requests of one timeslot with six active flows. For simplicity, a full bisection bandwidth network that is capable of supporting any traffic, where each node was paired to at most one other node per timeslot, was assumed.

The controller's timeslot allocation algorithm can be described as follows: Each D_{ij} is assigned a priority number based on some schedule strategy, that is, max-min fairness or minimizing flow completion time. In the beginning of timeslot t, the controller processes all D_{ij} in non-decreasing order based on priority numbers. Timeslot t can be assigned to pair (i, j) if there is no another packet starting from endpoint i or destined to endpoint j in this timeslot. Step by step, at the end of algorithm, maximal matching is achieved between endpoints that are allocated to the timeslot. As

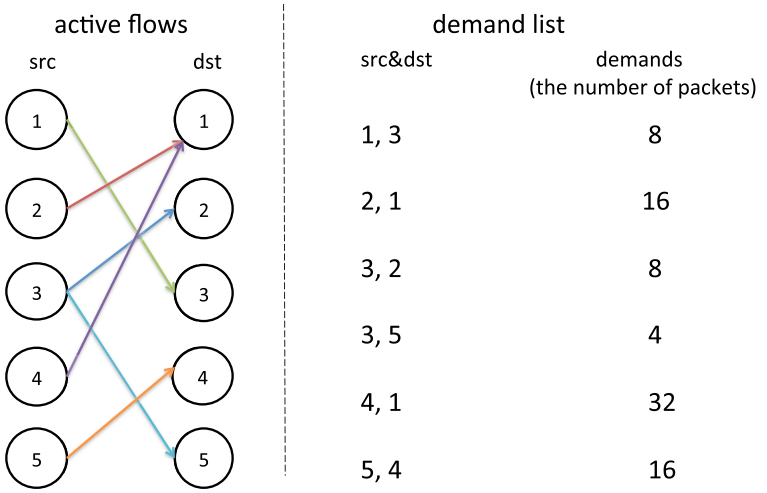


Figure 2. A case of demand needed in a simple data center with five endpoints.

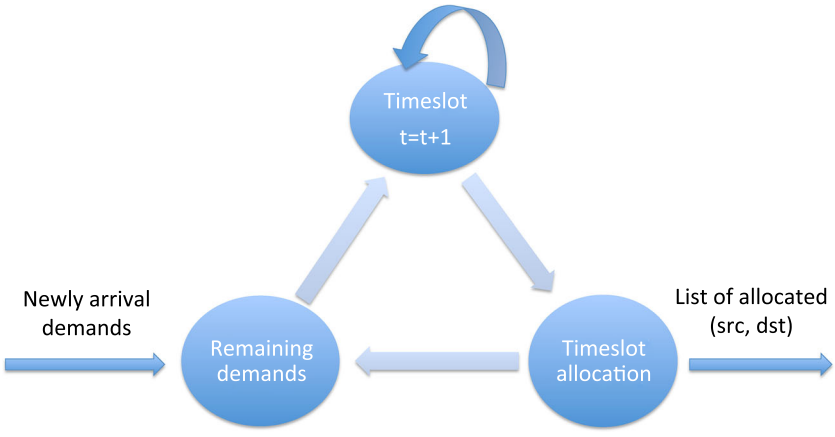


Figure 3. Procedure of timeslot allocation. The controller allocates current timeslot to remaining demands.

DOI: 10.1002/cpe

shown in Figure 3, the controller always maintains a list of requests from unassigned endpoints and allocates the current timeslot to the remaining flow demands. Note that the newly received demands are also merged to the remaining demand set.

Algorithm 1 Timeslot allocation design

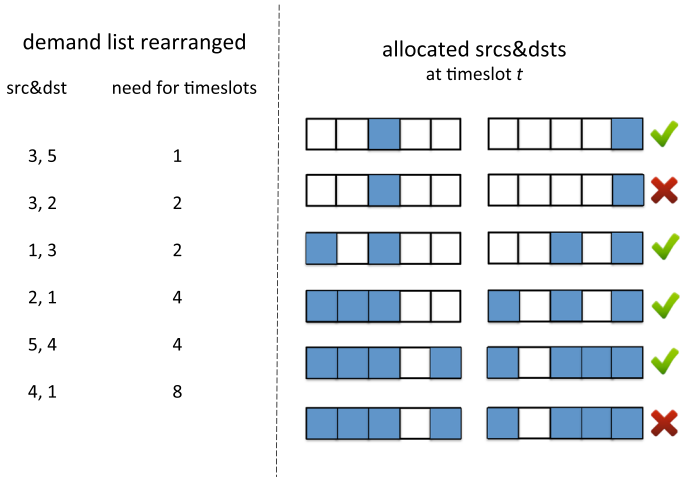
Input:

<RemainingDemand, NewlyDemand>

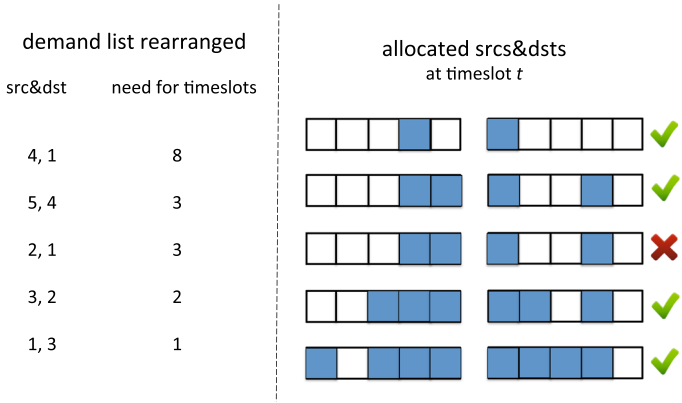
Output:

Allocated Matching

1: *DemandList=RemainingDemand+NewlyDemand*;
2: Order *DemandList* based on allocation policies and generate *DemandOrder*;
3: **while** $ADH+Demand[i]<C$ **do**
4: $AllocatedMatching=AllocatedMatching+(src,dst)_i$;
5: $ADH=ADH+Demand[i]$;
6: $i++$;
7: **end while**



(a) timeslot allocation of timeslot t



(b) timeslot allocation of timeslot t+1

Figure 4. A simple example of timeslot allocation, using min-max-min Maximum Transmission Unit (MTU)'s policy.

DOI: 10.1002/cpe

Algorithm 2 Timeslot allocation model

Input: *<RemainingDemand, NewlyDemand>*
Output: Allocated Matching
1: *DemandList=RemainingDemand+NewlyDemand*;
2: Order *DemandList* based on allocation policies and generate *DemandOrder*;
3: **while** $ADH+Demand[i]<C$ **do**
4: $AllocatedMatching=AllocatedMatching+(src,dst)_i$;
5: $ADH=ADH+Demand[i]$;
6: $i++$;
7: **end while**
8: *RemainingDemand=DemandList−AllocatedMatching*;
9: **return** *AllocatedMatching*;

3.2. Algorithm demonstration

At the first glance, this problem only needed to be solved in the conventional way, that is, shortest flow or data first (SF), by resorting to the global network view of the controller. Unfortunately, such a singsong scheduling strategy can potentially lead to some consequences, that is, SF is likely to starve some large flows. In the following, various timeslot allocation strategies are used in different timeslots.

In this paper, the controller was allowed to order the demands by first performing the fewest remaining MTUs and then the largest remaining MTUs, then the few remaining MTUs again, namely, min-max-min MTUs. As a result, large flows do not need to wait for a long time, but short flows can be finished rapidly. For simplicity, here, we assume one bit for each source and for each destination.

Figure 4 demonstrates the allocation of one timeslot in a simple network with five endpoints. The left panel of Figure 4 shows the demand list rearranged for min-max-min MTUs' purpose, while the right panel shows the bandwidth constraints that should be satisfied. Figure 4(a) is the allocation of timeslot t. The controller first allocates the pair of (3,5). Because source 3 and destination 5 are available, the pair can be allocated. However, the second pair (3,2) cannot be allocated because source 3 has already been allocated. In this similar way, maximal matching can be reached (3,5), (1,3), (2,1), and (5,4) at timeslot t.

Ideally, the controller can process the remaining demands as soon as it is produced. Figure 4(b) shows the timeslot allocation at timeslot $t+1$. However, contrary to the aforementioned sequence, the controller must reorder demands before allocating timeslots. Because the pair of 3 and 5 no longer requires any timeslot, it is removed. The procedure of allocation is the same as timeslot t. At last, we will get a matching of (4,1), (5,4), (3,2), and (1,3).

For min-max-min MTU policy, every time the controller chooses matching for a new timeslot, it should reorder the demands. To reduce the overhead of processing and reordering demands, demands can be kept in roughly the correct order, while the controller allocates a batch of four timeslots in one shot.

Speeding up the process: It is important for the controller to calculate the new assignment quickly in order to promptly adapt to network dynamics and meet low latency and high throughput requirements.

However, finding an allocation with a maximum matching is expensive. To reduce allocation time, a fast data structure, the $bitmap\ table$ was used to produce a fast operation in a heuristic way. In a bitmap table, '1' indicates that the pair cannot communicate in that timeslot, while a '0' indicates otherwise [14]. The algorithm was sped up using a 'find first set' operation.

4. PATH SELECTION

In this section, we first modeled our path assignment problem as a bin-packing problem and then present a hybrid genetic algorithm to solve this problem.

DOI: 10.1002/cpe

Algorithm 3 Path selection

Input: Allocated matching
Output: RouTing path

```
1: Compute the demand of each ToR switch;
2: Sort ToR switches in the decrease order of the corresponding demand;
3: Choose Agg and Core switches.
4: Select Agg switches.
5: Sort the Agg switch's bandwidths;
6: Initialize setNumAgg;
7: for each i in NumToR do
8:    Sequentially find an allocated Agg switch j with the spare space being larger than ToR
      demand i;
9:    if no spare space for all allocated Agg then
10:      allocate a new Agg switch;
11:      allocate the new AggID to this ToR demand;
12:   else
13:      allocate Agg switch j to this ToR demand;
14:   end if
15: end for
16: Select Core switches and operate as the same in Step 4-15.
```

4.1. Design of path selection process

A critical requirement for congestion-free network is to assign packets to *idle* paths. No path selection should assign one link to multiple packets in a single timeslot. In common data center topologies (e.g., multi-rooted trees), there are redundant paths between endpoints. The rearrangeably non blocking (RNB) property guarantees that once the timeslots are allocated, path selection assigns packets to available paths through the network [15] .

Modern data center networks are constructed into tiers. Typical architectures today consist of three-level trees of switches or routers (e.g., fat-tree): ToR, Agg, and Core. In the higher levels of the hierarchy, there is a larger switching capacity to aggregate traffic between the edges. For simplicity, it is here assumed that host to switch links were 1GigE, links between ToR and Agg were 2GigE, and links between Agg and Core were 4GigE.

The path selection problem is a variant of multi-dimensional hierarchy bin-packing problem, where the resources are the bins and the packets are the objects. Multi-dimensional bin-packing problems are NP-hard [9, 16]. The objective of path selection is to minimize resource usage and path length used by packet traffic. The notion of minimize resource usage and link length (MRL) is here defined. Two types of resources were used: switches and links. Typically, assigning packets to different switches will result in different MRL. For simplicity, each packet was first directly assigned to its one-hop ToR switch.

Algorithm sketch: A given set of packets were sorted by decreasing traffic volume, and an attempt was made to assign them one by one (i.e., packets with most traffic are assigned first). ToR, Agg, and Core switches are here modeled as bins of three different sizes. To assign a given packet, all switches were considered possible candidates to host this packet and pick the assignment that results in the smallest MRL. This was processed in a hierarchical way, that is, first, all the packets were loaded into ToR bins, then the ToR bins were loaded into Agg bins, and at last the Agg bins were loaded into Core bins.

MRL consideration: For each potential assignment, the additional resource usage and path length must be found. Considering the data center topology, the upper switches must be used as little as possible.

In this way, bins (switches) may have some spare space. This may waste some resources. However, experimental results indicate that a congestion-free routing strategy can observably reduce the tail of the packet delay distribution, which significantly improved network efficiency.

DOI: 10.1002/cpe

4.2. Design of hybrid genetic algorithm

To solve a multi-dimension bin-packing problem, the problem was first approximated with a greedy algorithm, which combined the advantages of both the NF algorithm and SGA. By emulating biological selection and reproduction, the algorithm can search the problem domain in a general, representation-independent manner very effectively. Here, a *sliding window* was used to identify highly fit sequences and identify $reduction$ for their preservation. When these sequences were optimal, the chromosome could be reduced and the sequences preserved by movement to the front of the chromosome. NF served as the objective function for the following reasons: First, NF is by far the simplest of the algorithms to implement; its $O(n)$ efficiency proves more cost-effective than the O ($n$log$n$) efficiency of the other methods. Second, because we do not know when a packet arrives and what its size may be, the bin packing is somehow an online problem, and NF can solve online problems better than other methods can. This idea has been applied with success to a genetic algorithm for bin packing.

5. PERFORMANCE EVALUATION

The performance of congestion-free routing strategy was evaluated through a series of simulations. Congestion-free routing's performance was compared to that of existing data center transports, TCP and DCTCP. These were compared with respect to throughput, queueing, and latency. The congestion-free strategy proposed here performed better than TCP and DCTCP in detailed packet-level simulations.

5.1. Experimental setup

The experimental settings, including the data center topology and traffic workloads, are here described, and the protocols are compared.

Data center topology: Here, three-tier topology comprising layers with 16 ToR switches, eight aggregation switches, and four core switches is used. The topology interconnects 64 hosts through 16 ToR switches that are divided into four parts, which are in turn interconnected via four core switches. In the baseline topology, the servers attached to the ToR switches with 1 Gbps links. The ToR switches connected to each Agg switch with two 2 Gbps uplinks, and each Agg switch connected to core switches with four 4 Gbps uplinks. Note that there was a 2:1 oversubscription at the ToR level, typical of today's data center deployments. Through the experiments, one server is set aside for running the controller.

Traffic workloads: A simple program was developed to generate the traffic. It was here assumed that flows arrived in a pattern of Poisson process and two servers were randomly chosen as its sender–receiver pairs. Because the goal of the study was to address network congestion, a Poisson arrival interval was set to 10 ms and timeslot to 1 s. Flow sizes were drawn from the interval (2 KB and 198 KB), and packet sizes were drawn from the interval (64 B and 1513 B). Both of them showed a uniform distribution.

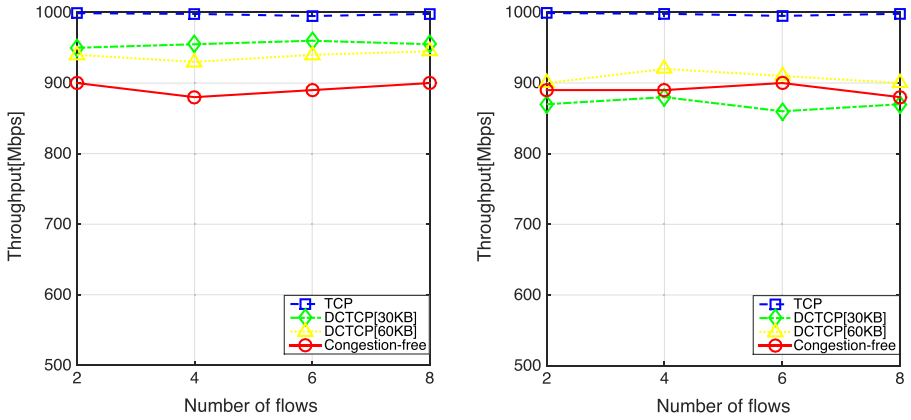
Protocols compared: Congestion-free strategy, TCP, and DCTCP are compared. TCP uses the endpoints to control the transmission rate and switches to choose the transmission paths. For TCP, all the flows and endpoints were treated the same way, meaning that all the flows were equally scheduled. The only difference between TCP and DCTCP was that DCTCP prevented queue buildup and so prevented timeouts. TCP and DCTCP were also implemented in the same data center topology described previously.

Performance metrics: For deadline-constrained traffic, application throughput served as a metric, here defined as the fraction of flows that met their deadlines. Meanwhile, for the queuing in switch, the queue length was used at the receiver's switch port. For the latency, RTT served as a performance metric.

5.2. Throughput evaluation

This experiment was performed to determine whether congestion-free strategy could reach the same throughput as TCP or DCTCP in the presence of long-lived flows. For simplicity, the number of

DOI: 10.1002/cpe



(a) Throughput with 1GB link (b) Throughput with 10GB link

Figure 5. Comparison of congestion-free with transmission control protocol (TCP) and data center transmission control protocol (DCTCP) in terms of application throughput.

flows was kept constant during the experiment, and the data of flows was sent at a constant speed. The number of flows ranged from two to eight. We also consider the N flows with identical RTTs, sharing a single bottleneck link of capacity C. Note that each flow was from a different server sending to the same receiver.

Three schemes were compared: (i) standard TCP, (ii) DCTCP with 30 KB marking threshold, and (iii) DCTCP with 60 KB marking threshold. For DCTCP, there was a simple active queue management scheme, the marking threshold, K. An arriving packet was marked with the Congestion Experienced (CE) code point if the queue occupancy was greater than K upon its arrival. Note that the K was set as 30 and 60 KB. Also, two kinds of values of C, for example, 1 and 10 Gbps, were used.

Analysis: The results are shown in Figure 5. The congestion-free strategy showed only slightly less throughput than TCP and DCTCP with 1 Gbps link, while both TCP and DCTCP achieved the maximum throughput of 0.945 Gbps. The throughput becomes gradually less effective as the number of flows increases. Then, the experiment was repeated with a 10-Gbps link. Figure 5(b) shows the throughput results. While TCP and congestion-free strategies maintained the throughput similar to that maintained with the 1 Gbps link, the DCTCP performance was sensitive to value of K. At 30 KB, the DCTCP performed worse than the congestion-free strategy.

5.3. *Queuing evaluation*

In congestion-free strategy, queuing may occur at a switch's port, while packets from multiple endpoints might arrive together. If the endpoints transmission time controlled by controller is not precise enough, the queuing becomes worse. In this section, a simple application was developed to funnel traffic to a single receiver. The server requests were generated as a Poisson process in an open loop fashion. That is, new requests were triggered independently of prior requests. Each server made two requests per second on average, while the senders established connections to the receiver and sent data as fast as they could. During the transfer, the queue length was sampled at the receiver's switch port every 200 ms. In this section, the congestion-free strategy was compared to standard TCP and DCTCP ($K = 20$ KB).

A huge difference in queue length was observed among TCP, DCTCP, and the congestion-free strategy. Figure 6 shows the variation of queue length over time. Results showed that both TCP and DCTCP ($K = 20$ KB) queue lengths varied widely, especially TCP. Both of them were maintained at high levels. On the contrary, the queue length of congestion-free strategy was remarkably small and did not vary widely, even so much as the optimized queue-based DCTCP with a low marking threshold. Meanwhile, the CDF in Figure 7 shows that congestion-free strategy queue length is

DOI: 10.1002/cpe

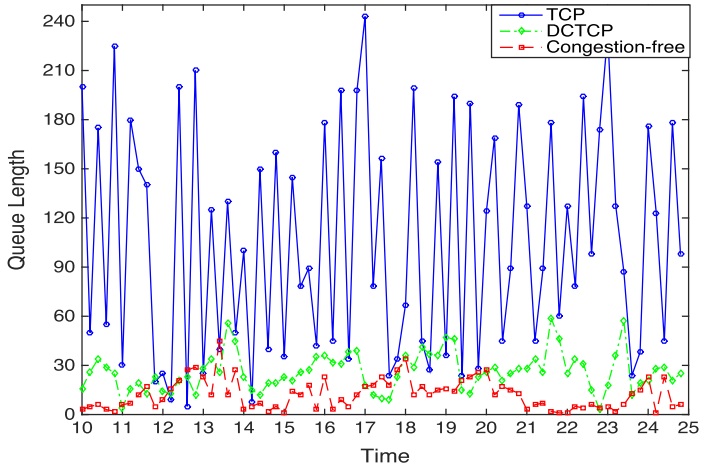


Figure 6. Switch queue lengths sampled at 200 ms intervals on the ToR switch. TCP, transmission control protocol; DCTCP, data center transmission control protocol.

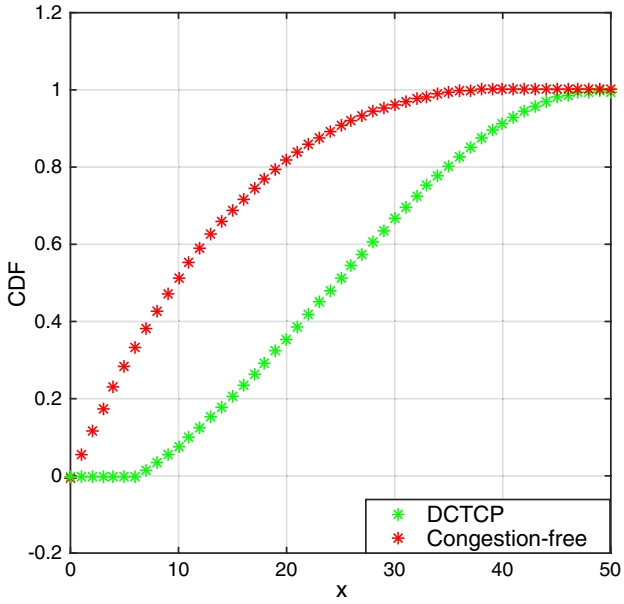


Figure 7. CDF of queue lengths with data center transmission control protocol (DCTCP) and congestion-free strategy.

stable around two packets while the DCTCP ($K = 20$ KB) queue length was around 20 packets and TCP queue length is 10 times larger than DCTCP. Considering the aforementioned statement, it is observable that the congestion-free method achieves a suitable throughput, even at very small queue length.

Analysis: Switch's dynamic buffer allocation policy[17] caused TCP and DCTCP to use queues to absorb data bursts. If packets arrived faster than the router (or switch) can process, the router (or switch) placed them into the queue (also called the buffer) until they could obtain enough resources to transmit them. The difference between TCP and DCTCP was that the DCTCP senders started reacting as soon as queue length at a given switch port exceeded K. This reduced queueing delays on congested switch ports, but it could not eliminate the queues. However, the congestion-free strategy used a centralized global controller to assign timeslots to senders, which in turn keeps queues relatively empty and greatly mitigated costly packet losses that can lead to timeouts.

DOI: 10.1002/cpe

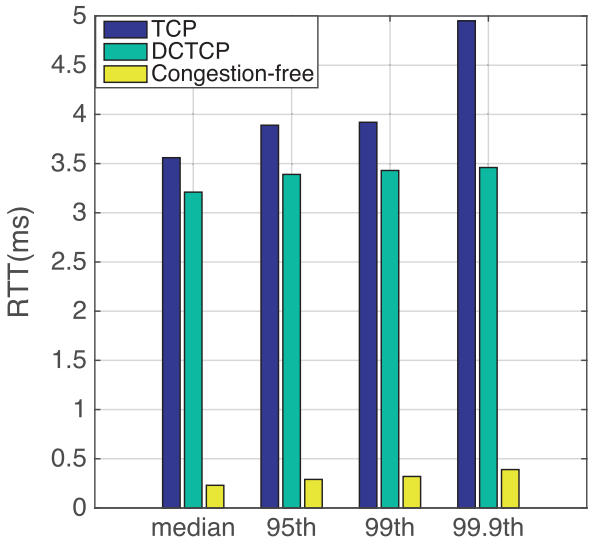


Figure 8. Congestion-free round-trip time (RTT) computed with a widely range of workloads in comparison with transmission control protocol (TCP). DCTCP, data center transmission control protocol.

5.4. *Latency evaluation*

Here, the RTT of interactive requests was used to evaluate the latency under high load. Network tools like ping tests and trace routes were used to measure latency by determining the time it takes a given network packet to travel from source to destination and back, the so-called RTT.

These experiments were performed using TCP and DCTCP. For DCTCP experiments, K was set to 20 on 1 Gbps links. The interactive requests (e.g., query) were time-sensitive, and the completion time was the most critical point, although the RTT was a key indicator. Figure 8 shows the median, 95th, 99th, and 99.9th percentiles of RTT (ms). The results showed that congestion-free strategy performs significantly better than TCP and DCTCP. It is here visible that, with TCP, the RTT of interactive requests is substantially worse at the tail of the distribution than at other points while congestion-free performance was unchanged and maintained significant reductions.

Analysis: Note that network delay included four parts: processing delay, queuing delay, transmission delay, and propagation delay. Queuing delay is a key component of network delay. So even with the added round-trips to the controller, end-to-end latency was substantially lower than TCP.

6. DISCUSSION

6.1. *Packet-based transmission*

In this paper, a congestion-free routing strategy on the granularity of packets is proposed. Existing switching methods used for data center can be classified as flow switching, packet switching, or flowlet switching. Flow switching is suboptimal for 'heavy' flow distributions with large flows. A flowlet is a burst of packets from the same flow followed by an idle interval. However, it is unclear whether the low intra-data center latencies can meet the timing requirements of flowlet bursts to prevent packet reordering and still achieve good performance. The gaps needed for flowlets may be rare in the very high bandwidth of internal data center.

For scalability and reliability, *packet* was selected as the schedule unit. Packet switching is optimal, but it required a reordering-resilient mechanism. To the extent that reordering occurs today, many technologies have been proposed. RR-TCP [18] is a reordering-robust TCP with TCP senders extended to distinguish between reordering and loss. This provides us a false fast retransmit-avoid and transmission improvement over paths that reorder or delay packets.

DOI: 10.1002/cpe

6.2. Explicit path control over allocated paths

Once the paths were allocated by the controller, the routing of packets to the paths allocated to them must be guaranteed. This requires explicit control of the routing path over the underlying topologies.

In software-defined data center networks, OpenFlow [8, 19] has been used in many recent proposals, such as Hedera [7], MicroTE [20], and ElasticTree [21] to enable explicit path control. OpenFlow can establish fine-grained explicit routing path by installing flow entries in the switches via the SDN controllers.

Current numerous companies and research institutions have presented their own controller solutions. Ryu is one of an Apache-licensed open source software for building software-defined networks. It offers software components with well defined API that make it easy for developers to create new network management and control applications. Ryu is fully written in Python.

7. RELATED WORK

Broadly speaking, the prior work on addressing congestion-aware load balancing can be classified as either centralized scheduling (e.g., Hedera [7]), in-network mechanisms (e.g., Flare [22]), or host-based transport protocols (e.g., multipath TCP (MPTCP) [23]). These approaches all have their limitations.

Traditional traffic engineering mechanisms for WAN use centralized ways of providing better load balance and network sharing, but they operate at coarse timescales (hours) based on long-term estimates of traffic matrices. Hedera collects flow information from constituent switches, computes non-conflicting paths for elephant flows, and instructs switches to re-route elephants accordingly [7]. Its goal was to maximize aggregate network utilization, and there is no solution that can reduce network latency. The data center time division multiple access (TDMA) Media Access Control (MAC) layer is implemented for commodity Ethernet hardware that allows link bandwidth and switch buffer space are partitioned to flows, while only schedule elephant flows and short flows are delegated to other means [24]. Recently, B4 and SWAN have been proposed, and the two have nearly ideal performance. They classify modern WAN traffic into three categories and operate over minutes, which is not suitable for highly volatile data center networks. The current method is similar to Fastpass in that it uses a centralized method to determine when and how a packet should be transferred, while Fastpass is suitable for two-tier data center [14]. The current method is more scalable for all kinds of data centers.

In-network mechanisms and host-based transport protocols have a distributed traffic schedule. Distributed approaches always use endpoints to make packet transmission decisions and switches to make path selection. These result in a loss of control over intra-data center network. Flare proposed multiple paths in the wide area on the granularity of local congestion-aware processes, which handles asymmetry only poorly [22]. TeXCP [25] and MATE [26] perform dynamic traffic scheduling across multiple paths using explicit congestion notification packets, which are hard to be implemented directly in switch hardware. CONGA [27] is conceptually similar to TeXCP in that it uses pathwise congestion metrics but it is significantly simpler. It splits TCP flows into flowlets. However, CONGA may not achieve the optimal traffic balance in networks with three or more tiers because it only controls the load balancing decision at the leaf switches.

The host-based transport protocol, MPTCP [23], splits each connection into multiple sub-flows and schedules them based on perceived congestion. While MPTCP is effective for load balancing, the sub-flows can also increase the burst of traffic, making the method difficult to implement.

8. CONCLUSION

A centralized traffic engineering mechanism using an SDN controller to determine when each packet should be transmitted and what path it should follow is here presented in the interest of improving network congestion-free performance. The congestion-free strategy design involves two key components: (i) *Timeslot assignment algorithm* uses a fast maximal matching to determine the time

DOI: 10.1002/cpe

at which each packet is transmitted, and it reaches specific objectives such as max-min fairness, minimizing the flow completion times and (ii) *path selection algorithm* at the SDN controller is constructed as a bin-packing problem to determine each packet's path. Extensive evaluation with a simulation experiment demonstrated that this congestion-free routing strategy performed well in throughput, queuing, and end-to-end RTT and can significantly reduce the queue length and median ping time while maintaining a throughput similar to that maintained by DCTCP.

ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation for Outstanding Youth under grant no. 61422214, the National 973 Basic Research Program under no. 2014CB347800, the Program for New Century Excellent Talents in University, and the Distinguished Young Scholars of National University of Defense Technology under grant no. JQ14-05-02.

REFERENCES

1. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM* 2008; **51**(1):107–113.
2. Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: distributed data-parallel programs from sequential building blocks. *Proceedings of the EuroSys 2007 Conference*, Lisbon, Portugal, 2007; 59–72.
3. Ramakrishnan K, Floyd S, Black D. RFC 3168. *The Addition of Explicit Congestion Notification (ECN) to IP* 2001.
4. Alizadeh M, Greenberg AG, Maltz DA, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M. Data center TCP (DCTCP). *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New Delhi, India, 2010; 63–74.
5. an Abdul Kabbani MA, Edsall T, Prabhakar B, Vahdat A, Yasuda M. Less is more: trading a little bandwidth for ultra-low latency in the data center. *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012*, San Jose, CA, USA, 2012; 253–266.
6. Munir A, Qazi IA, Uzmi ZA, Mushtaq A, Ismail SN, Iqbal MS, Khan B. Minimizing flow completion times in data centers. *Proceedings of the IEEE INFOCOM 2013 Conference*, Turin, Italy, 2013; 2157–2165.
7. Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: dynamic flow scheduling for data center networks. *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, USA, 2010; 281–296.
8. Lu G, an Yulong Li CG, an Tong Yuan ZZ, Wu H, Xiong Y, Gao R, Zhang Y. Serverswitch: a programmable and high performance platform for data center networks. *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, 2011; 2–2.
9. Chekuri C, Khanna S. On multi-dimensional packing problems. *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, 1999; 185–194.
10. Jia L, Fang-yuan J, Xiao-xiao X. A dynamic routing, wavelength and timeslot assignment algorithm for WDM-TDM optical networks. *2nd International Conference on Future Computer and Communication (ICFCC '10),* Vol. 1: IEEE, 2010; V1–533.
11. Eguchi Y. *Monitoring apparatus and monitored apparatus*, 2010.
12. Yu H, Li K, Li W, Tao X. Zebra: an east-west control framework for SDN controllers. *Proceedings of the 44th International Conference on Parallel Processing (IPCC)*, Beijing, China, 2015.
13. Li W, Qi H, Li K, Stojmenovic I, Lan J. Joint optimization of bandwidth for provider and delay for user in software defined data centers. *IEEE Transactions on Cloud Computing (TCC)* 2015.
14. Perry J, Ousterhout A, Balakrishnan H, Shah D, Fugal H. Fastpass: a centralized 'zero-queue' datacenter network. *Proceedings of the ACM SIGCOMM 2014 Conference*, Chicago, IL, USA, 2014; 307–318.
15. Duato J, Yalamanchili S, Ni LM. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann: San Mateo, CA, 2003.
16. Asplund M, Nadjm-Tehrani S, Zagar K. Middleware extensions that trade consistency for availability. *Concurrency and Computation: Practice and Experience* 2009; **21**(9):1181–1203.
17. Choudhury AK, Hahne EL. Dynamic queue length thresholds for shared-memory packet switches. *IEEE/ACM Transactions on Networking (TON)* 1998; **6**(2):130–140.
18. Zhang M, Karp B, Floyd S, Peterson LL. RR-TCP: a reordering-robust TCP with DSACK. *Proceedings of the 11th IEEE International Conference on Network Protocols*, Atlanta, GA, USA, 2003; 95–106.
19. McKeown N, Anderson T, Balakrishnan H, Parulkar GM, Peterson LL, Rexford J, Shenker S, Turner JS. Openflow: enabling innovation in campus networks. *Computer Communication Review* 2008; **38**(2):69–74.
20. Benson T, Anand A, Akella A, Zhang M. Microte: fine grained traffic engineering for data centers. *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies*, Tokyo, Japan, 2011; 8.

DOI: 10.1002/cpe

21. Heller B, Seetharaman S, Mahadevan P, Yiakoumis Y, Sharma P, Banerjee S, McKeown N. Elastictree: saving energy in data center networks. *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, USA, 2010; 249–264.
22. Kandula S, Katabi D, Sinha S, Berger A. Dynamic load balancing without packet reordering. *Computer Communication Review* 2007; **37**(2):51–62.
23. Raiciu C, Barré S, Pluntke C, Greenhalgh A, Wischik D, Handley M. Improving datacenter performance and robustness with multipath TCP. *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Communications*, Toronto, Canada, 2011; 266–277.
24. Vattikonda BC, Porter G, Vahdat A, Snoeren AC. Practical TDMA for datacenter Ethernet. *Proceedings of the 7th ACM European Conference on Computer Systems*, Bern, Switzerland, 2012; 225–238.
25. Kandula S, Katabi D, Davie BS, Charny A. Walking the tightrope: responsive yet stable traffic engineering. *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, Pennsylvania, USA, 2005; 253–264.
26. Elwalid A, Jin C, Low SH, Widjaja I. MATE: MPLS adaptive traffic engineering. *Proceedings of the IEEE INFOCOM 2001 Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the Communications Odyssey*, Alaska, USA, 2001; 1300–1309.
27. Alizadeh M, Edsall T, Dharmapurikar S, Vaidyanathan R, Chu K, Fingerhut A, Lam VT, Matus F, Pan R, Yadav N, Varghese G. CONGA: distributed congestion-aware load balancing for datacenters. *Proceedings of the ACM SIGCOMM 2014 Conference*, Chicago, IL, USA, 2014; 503–514.

DOI: 10.1002/cpe