Webology (ISSN: 1735-188X)
Volume 18, Number 2, 2021

HCCLBA: Hop-By-Hop Consumption Conscious Load Balancing Architecture Using Programmable Data Planes

V. Rajkumar[1] , Dr. V. Maniraj[2]

[1]Research Scholar, Dept. of Computer Science, AVVM Sri Pushpam College (Affiliated to Bharathidasan University, Tiruchirappalli), Poondi, Thanjavur, , ORCID ID: 0000-0002-8113-2616

[2]Research Advisor, PG and Research Dept. of Computer Science, AVVM Sri Pushpam College (Affiliated to Bharathidasan University, Tiruchirappalli)., Poondi, Thanjavur, Tamilnadu, India.

Abstract

Datacenter networks typically make use of multi-rooted topologies in order to deliver enormous bisection bandwidth (such as leaf spines and fat trees). In order to make the most of the available bisection band width, it is necessary to have a load-balancing mechanism for the data plane. This is due to the significant degree of multipathing that is present in these topologies. The most typical approach of load balancing is called equivalent-cost multi-path routing, or ECMP for short. This method moves traffic down a number of different routes simultaneously. Congestion Aware Load Balancing Techniques, such as CONGA, have been created as a solution to the restrictions that ECMP presents. Using these approaches does come with a few of limitations. To begin, the amount of congestion-tracking state that can be retained at the edge switches is restricted due to the memory limitations of the switches, which makes it hard to scale complicated topologies using these switches. The second drawback is that because they are incorporated into specialized hardware, they cannot be modified locally on the premises where they are being used. A load-balancing approach for the data plane in this research, HCCLBA is offered as a solution to overcome both of these concerns. As a first stage, rather of watching congestion on all possible pathways, each HCCLBA switch only tracks congestion on the most efficient path to the destination, which is typically through an adjacent switch. It is also feasible to run HCCLBA on these chipsets without the requirement for new hardware by designing it to work on programmable switches and programming it in P4. This would make it viable to run HCCLBA on these chipsets. The simulation shows that HCCLBA has a faster average flow completion time than a scalable extension to CONGA (1.6 times faster at 50 percent load and 3 times faster at 90 percent load).

Keywords: HCCLBA, Load Balancing, Software Defined Networking.

Introduction

Large bisection bandwidth is offered by the networks that are used in data centers that have multi-rooted topologies (Fat-Tree and Leaf-Spine). These topologies are distinguished by the presence of a significant number of pathways that may be used to connect any two nodes. It is vital to spread the traffic load over

the multiple data plane channels in order to make the most efficient use of the bandwidth that is available for the bisection. Because the whole network is abstracted into a single enormous output-queued switch when load balancing is used, it is much simpler to allocate bandwidth to various tenants, flows, or groupings of flows.

ECMP, which distributes traffic by randomly assigning each flow to one of numerous channels, is one of the approaches that is utilized the most frequently for the purpose of balancing the load on the data plane. ECMP suffers from decreased performance despite the fact that two long-running flows can be allocated to the same route at the same time. When ECMP is implemented, the network may become underutilized or overloaded due to asymmetric topologies as well as connection failures. The CONGA data plane load balancing solution has recently evolved as a response to the limitations imposed by ECMP. This solution addresses the limitations by making use of link utilization statistics in order to balance load across paths.

It is essential to keep in mind, however, that this responsiveness is associated with a significant cost of implementation. Implementing CONGA in custom silicon on a switching chip requires a significant amount of time spent on the hardware design and verification processes. As a direct consequence of this, the CONGA approach cannot be altered after it has been put into practice. Because of the limited amount of memory available on a switching chip, the congestion-monitoring method that CONGA employs at the leaf switches can only be employed in topologies that have a limited number of routes. It is for this reason that CONGA can only be utilized in a Leaf-Spine architecture, which severely restricts its capacity for scaling.

This research presents a new technique of load balancing for the data plane called HCCLBA (Hop-by-hop Consumption Conscious Load Balance Architecture). Its purpose is to address the challenges described above. When it comes to scalability, HCCLBA is superior to CONGA in a number of aspects, the most important of which are described here. In contrast to CONGA's leaf switches, which select the complete path, each HCCLBA switch just determines the next hop. This reduces the requirement that forwarding state be maintained for a large number of tunnels. In contrast to this, CONGA's leaf switches, which determine the course of the whole route. Congestion status need only be maintained for the best next hop that goes to a destination as opposed to keeping track of it for all of the pathways that lead to a destination since HCCLBA switches will only select the best next hops along the immediate best path.

In order to make advantage of HCCLBA, you will need a programmable switch architecture such as RMT, Flexpipe, or Xpliant. In order to provide evidence of this, the recently proposed P4 programming language specifically targets programmable data planes. Because of this, the network operator is given the opportunity to analyses and alter the HCCLBA algorithm without being constrained by silicon implementation.

To put it another way, the HCCLBA compiles information on global link utilization by making use of certain probes (not included in data packets). These probes are dispatched according to a predetermined timetable to guarantee that they cover the entirety of the network in order to accomplish load balancing. Each switch has its own database where this information is stored, and this database is accessed in order to calculate the best "next hop" to any given point. After that, every switch will send an updated HCCLBA probe to the other upstream switches, providing those switches with their view of the best path for the downstream traffic (defined as the one that uses the least amount of all links along a path).

Because of this, information on the most efficient routes is dispersed over the entirety of the network in a manner that is analogous to a protocol based on distance vectors. In order to prevent the reordering of packets, the HCCLBA performs load balancing at the level of individual bursts of packets that are spaced out over a significant amount of time.

In order to evaluate HCCLBA in relation to other current load-balancing strategies, we constructed it with the help of the network simulator ns-2. When compared to other systems, HCCLBA's three-tier design results in a reduction in switch states and an improvement in flow completion times. When one of the essential links is severed, we investigate how HCCLBA reacts to these changes in the network. According to the results of our tests, HCCLBA offers superior performance than its rivals in both symmetric and asymmetric topologies.

In a nutshell, the two most important contributions that we make are as follows:

Within the scope of this study is the proposal of HCCLBA, a data-plane load-balancing system that is scalable. According to our knowledge, HCCLBA is the first load balancing system that was built expressly for a programmable switch's data plane. This is what we perceive to be the case. HCCLBA beats the existing state-of-the-art congestion-aware load balancing algorithms by providing flow completion speeds that are between 1.6 and 3.3 times quicker under heavy network loads.

Design Challenges for HCCLBA

The Fat-Tree architecture, which consists of numerous tiers, is frequently utilized in the case of large datacenter networks. In these topologies, leaf-spine pods have two tiers each, and they are joined to one another by further tiers of spines. Depending on the bandwidth requirements of the data center, these extra layers linking the pods might go as deep as is necessary in order to meet the requirements. Load balancing in such expansive datacenter topologies is afflicted with three main issues due to the fact that the number of paths between any two Top of Rack switches (ToRs) grows at an exponential pace.

Large path utilization matrix: The number of paths that can link any two ToRs increases proportionally with the size of the radix at the center of a Fat-Tree topology. A transmitter ToR in a Fat-Tree topology with radix k has to be able to monitor link use on all desirable paths leading to the destination ToR in order to function properly.

Topology	# Routes connecting two different ToRs	# The maximum number of forwarding entries allowed per switch
Fat-Tree - 8	18 Pairs	945
Fat-Tree - 16	62 Pairs	15806
Fat-Tree - 32	250 Pairs	2,32,456
Fat-Tree - 64	1022 Pairs	43,21,452

Table 1: In 3-tier Fat-Tree topologies, the total number of possible pathways and forwarding entries

The following step that has to be taken is to make a list of all of the k2 routes that travel to each of the

ToRs. The number of entries that it has to keep track of will increase at an exponential rate if there are m of these leaf ToRs. The route usage information can be stored in CONGA's RAM, which has a total capacity of 48K bits. In a network configuration with 10000 TORs and 10000 pathways connecting each pair, the ASIC would need a memory capacity of 600,000,000 bits; this is an excessively large quantity. Every switch needs to be in a decreased congestion-tracking state if the ASIC is going to be profitable and if it is going to be able to grow to accommodate topologies of significant size.

Large forwarding state: In order to enable the leaf-to-leaf tunnel for each path over which the switch has a demand for packet routing, the methods that are now in use call for huge forwarding tables to be maintained in each switch. This is necessary since each switch has a requirement for packet routing. The Fat-Tree topology with radix 64 has a total of seventy thousand trees of records, as seen in Table 1, and each switch stores four million entries. There are also other topologies, such as VL2 and BCube, that are in the same difficult position. New technologies, like as Xpath, have implemented compression algorithms that take use of the symmetry of the network in an effort to find a solution to this challenge. The involvement of the control plane in updating and compressing the forwarding entries, which is typical in big topologies, leads these systems to react slowly to failures and topological imbalance. These problems are frequent in large topologies. Because of this, the systems are significantly slower to adjust to new circumstances.

Discovering uncongested paths: Even when the network is under heavy use, reactive load balancing methods have a hard time finding an uncongested channel for incoming traffic. Short flow completion times are impacted because the load balancer is unable to locate an uncongested channel for these flows. Consequently, it is advantageous to provide the sender with use information prior to the start of a brief flow.

Programmability: It is necessary to implement data-plane load-balancing systems in hardware for the aforementioned reasons, in addition to the fact that such systems need a significant amount of work in terms of both design and verification. As a consequence of this, network operators are compelled to install a singular device that is preset and cannot be modified in any way. The operator is required to wait until the subsequent product cycle in the event that a change to the load balancer or the addition of a new feature is wanted (which may take many years). One modification to load balancing that is based on queue occupancy rather than link utilization is an example of what is known as backpressure routing.

This design strategy presents a unique opportunity to examine and enhance its functionality in light of the recent explosion in the number of programmable packet-processing pipelines. A common programming language like as P4 may be used to configure these data-plane systems. This provides operators with the ability to conduct stateful data-plane packet processing at line rate. After a load balancing scheme has been built in P4, an operator is able to change the programmed in such a way that it matches to the deployment scenario that she is using, and then compile the programmed in order for it to operate on the underlying hardware. In the context of programmable data planes, the load-balancing method needs to be simple enough so that it can be compiled to the instruction set that is provided by a certain programmable switch. If it isn't simple enough, it won't be able to be compiled. In the event that it is not, the system will not function as intended.

HCCLBA Overview: Scalable, Proactive, Adaptive, and programmable

In addition to the fact that HCCLBA does not necessitate the use of tunnels, one of its advantages is that it provides scalable, adaptable, and distributed network routing. In the same way that traditional distance-vector routing works, HCCLBA makes use of periodic probes to proactively update network switches with information on the best route to each individual leaf ToR. Because of this, HCCLBA is able to accomplish the same level of performance as the more conventional distance-vector routing. These probes, in contrast to the manner in which routers handle control packets, can be handled on the data plane at line rates. In contrast to this, the way routers handle the processing of control packets is as follows: Switches are able to make timely and effective forwarding decisions for the varied datacenter traffic that they experience as a result of the continual monitoring of the global congestion on the network that they do. In addition, in contrast to conventional routing, switches divide flows into flowlets whenever they identify a break in the RTT (network round trip time) of a flow. This occurs whenever a flow is sent across the network. As a consequence of this, load balancing may be carried out on a very granular level. The amount of reordering of receive-side packets that is caused as a result of an HCCLBA switch's transmission of different flowlets on separate paths that were evaluated to be optimal at the time of their arrival is reduced. This is because the switch sends the flowlets on paths that were determined to be optimal prior to the arrival of the flowlets. Because of the fundamental approach taken by the HCCLBA, which comprises of probe-informed forwarding and flowlet switching, it is possible for us to possess the features that are listed below.

Maintaining compact path utilization: Instead of keeping a record of all of the possible routes to a destination ToR, an HCCLBA switch simply maintains a table that maps the destination ToR to the next hop that will provide the most benefit to the overall path. When a switch gets many probes coming from different paths leading to a destination ToR, it will choose the hop that saw the probe make the fewest number of paths uses and choose that hop as the winner. As a direct consequence of this, it shares this information with its immediate neighbors. As a consequence of this, HCCLBA does not need to keep track of the number of alternative paths that might lead to a specific ToR. Since the switch's utilization has been cut down to the order of the number of ToRs, the switch memory is no longer under stress from route explosion (instead of the number of ToRs times the number of pathways to these ToRs from the switch). HCCLBA is responsible for the dissemination of global congestion information in order to make scaled local routing practicable.

Scalable and adaptive routing: The need for separate source routing is removed by HCCLBA's best hop table, which enables the network to make advantage of many network paths simultaneously. Other source-routing systems, such as CONGA and XPath, vary from HCCLBA in that the sender ToR is not responsible for determining optimal data path routes. This is one of the ways in which HCCLBA sets itself apart. In order to get at the intended point, it is necessary for each switch to independently choose the best next hop to take. This indicates that switches need not require separate forwarding-table entries in order to keep track of tunnels, which is beneficial for routing systems that utilize sources. Instead, this switch memory might be used to support more ToRs in the HCCLBA best hop table. This would be an alternative use for this memory. Because the best-hop table is updated by probes often at data-plane

speeds, the packet forwarding in HCCLBA is able to quickly react to changes in the dynamics of the datacenter, such as flow arrivals and departures.

Automatic discovery of failures: To stay operational, the HCCLBA waits for probes to arrive from the switches that are immediately adjacent to it. When a switch in the vicinity does not get a probe from another switch for an extended length of time, the switch's network utilization of the hop will age. This eliminates any possibility of the hop being chosen as the optimal hop for any of the ToR destinations. As a result of the fact that the switch will transmit this information to the switches that are upstream from it, all of the necessary switches within an RTT will be informed about the damaged path. The same as what happens if the connection is broken, the hop will become a best hop candidate for the reachable destinations the next time it gets a probe on the link. This will happen regardless of whether or not the connection was broken. When contrasted with this lightning-fast, topology-independent alternative, slow control-plane-driven routing technologies are given a significant challenge to their market dominance.

Proactive path discovery: HCCLBA does not transmit probes by piggybacking on data packets; rather, it sends them independently. This enables congestion information to be transmitted on channels regardless of the flow of data packets, in contrast to alternatives such as CONGA, which prevent this from being possible. HCLBA will use this for periodic probes on paths that are not being occupied by any switch at the moment. It enables switches to select a path that is not congested upon the arrival of a fresh flowlet, rather than having to first search through paths that are congested. When HCCLBA is being utilized, a flowlet will be redirected to a connection that is less congested thanks to the switches that are located on the path that connects to the bottleneck link. In this approach, short flows may be directed to less busy pathways in a manner that is both speedy and efficient.

Programmability: A packet is processed in the packet processing pipeline of an HCCLBA switch by changing the switch state at line rate. This occurs during the processing of a packet. Processing a probe entail doing tasks such as updating the best hop table and disseminating the probe to other switches in the immediate area. When processing a data packet, one of the steps involves reading a table containing the best hops and, if required, updating a database containing flowlets.

Topology and transport oblivious: The HCCLBAs do not have topological independence. This rule does not impose any restrictions on the number of tiers present in the network design, nor does it restrict the number of hops or paths that can exist between any two particular ToRs. Because HCCLBA may be deployed to only some of the switches in a network or only some of the traffic that flows through the network, gradual adoption is a possibility. Because HCCLBA is oblivious of the end-host application transport layer, host TCP stack changes are not necessary.

HCCLBA Design: Probes and Flowlets

The proactive broadcast of network utilization statistics to all of the switches in the network is facilitated by the probes provided by HCCLBA. Leaf ToRs are responsible for sending out probes, which are then repeated across the network. The control plane will take control of the replication mechanism when it

has finished setting up a multicast group. When a probe is connected to an incoming port, switches modify the optimal route that should be taken by flowlets that are moving in the opposite direction. In addition to this, the probes help in the process of discovering and adapting to changes in the topology of the network. All of this is accomplished by HCCLBA with a little amount of additional probing work.

In the topology of the network, upstream and downstream switches are notions that are assumed to exist. The concept of having numerous layers of switches is already included into the network topologies of the vast majority of datacenters, which means that it may be utilized in an organic manner. Upstream switches are defined as switches that are directly linked to a tier-i switch, while downstream switches are defined as switches that are directly connected to one another in the same tier. Both of these types of switches fall under the category of tier-1 switches.

Origin and Replication of HCCLBA Probes

Data is gathered through the use of HCCLBA probes that are transmitted on each and every uplink that links ToR to the network of the datacenter. A switch data plane, the CPU of the ToR, or a server that is connected to the ToR might all be responsible for the production of probes, provided that the hardware supports this possibility. This information is transmitted at a frequency of Tp hertz once per p seconds; from this point forward, we will refer to this frequency as the probe frequency.

As soon as they reach A1, the probes are dispersed to all of the ToRs (T 2) located further downstream, in addition to all of the spines located further upstream (S1, S2). Spine S1 makes a duplicate of each aggregate switch that is farther downstream, and spine S1 also receives a copy of the probe that is received. A probe is transmitted from switch S3 to switch A4, which then transmits the probe upstream to all of the ToRs associated with switch A4. This ensures that each and every path across the network will be tested by the probes, which in turn ensures that the network will be secure. This provides an additional guarantee that the probe will not wind endlessly around itself. When a probe completes its mission by travelling to another ToR, this is referred to as a "successful completion."

For the purpose of making the replication of probes easier, a multicast group table is generated in the data plane by the control plane. Because this is a one-time procedure, there is no need to worry about any connection failures or recoveries occurring throughout this process. It is possible to quickly replicate multicast groups by including more switches in an already existing set of switches. As soon as a new switch is connected, all that is required to be done in order to set up multicast mechanisms on the new switch is to add the switch port to multicast groups on the switches that are nearby. This is in addition to the initial connection of the new switch.

S1 S2 S3 S4

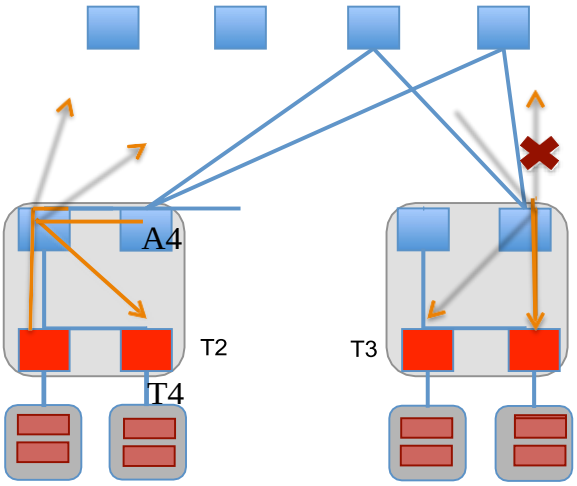

Figure 1: Logic for replicating the HCCLBA probe

Processing Probes to Update Best Path

A HCCLBA probe packet must be at least 64 bytes in size, and in addition to the standard Ethernet and Internet Protocol headers, it must additionally contain an HCCLBA header. The conventional Ethernet and Internet Protocol headers are supplemented by this additional header. The HCCLBA header consists of two distinct fields, which are as follows:

torID (24 bits): The position on the leaf TOR at which the probe was discovered for the first time. This is the target topology ring (ToR) for which the probe is communicating information on the usage of downstream routes, travelling in the opposite direction.

minUtil (8 bits): The utilization of the most efficient path in the case that the packet was moving in the opposite direction of the probe.

Data-Plane Adaptation to Failures

In addition to finding the best feasible forwarding routes, HCCLBA is able to learn about broken connections owing to the lack of probes. This information may be gleaned from the absence of probes. A procedure known as "ageing" is carried out on the entries of the bestHop table while they are being stored in the data plane of the network. The HCCLBA makes use of an update table to ensure that it is always up to date with the most current modification to bestHop. This indicates that if the destination's bestHop entry has not been refreshed within the previous Tf ail seconds, any other probe that has information about a ToR (from a different hop) will simply replace the bestHop and pathUtil entries for the ToR. This will take place regardless of whether the information was obtained from the same hop or a different hop. If the destination's bestHop entry has not been refreshed during the past TF-ail seconds, this is the result (a threshold for failure detection). If switches acquire this information about the change in the optimal path usage farther up the road, then they may pick a path that does not contain any links

at all in order to avoid the bottleneck link.

Without having to rely on the control plane, the HCCLBA is able to recognize when there has been an error and make the necessary adjustments to rectify the situation. Instead, the process of failure recovery in HCCLBA is much quicker than the recovery that is coordinated by the control plane, and it takes happen inside the timeframes for network RTT. This makes the HCCLBA method far more advantageous. Because the flowlets are rapidly directed to the next best alternative way rather than congestion-oblivious pre-installed backup paths, this strategy is superior to having a set of pre-programmed backup routes. This is because having a collection of pre-installed backup routes is preferable. This approach is better than other methodologies for a number of reasons, including this one. Consequently, flowlets are prevented from being transferred across faulty network channels, which leads to quicker completion times for both the network and the flow.

Programming HCCLBA in P4

Introduction to P4

P4 was built in order to accommodate programmable data-plane designs such as RMT, Intel Flexpipe, and Cavium Xpliant, amongst others. The cornerstone of the language is an abstract forwarding model that is also known as protocol-independent switch architecture (PISA). In this conception of a switch, the decoding of packets travelling over the wire is performed by programmable parsers. Following this, the packets are directed via an ingress pipeline that is equipped with a number of match-action tables. These tables modify the packets in accordance with whether or not the header fields of the packets match. The packets are switched over after they have arrived at the output ports in their entirety. The packets are processed by match-action tables in the egress pipeline before being serialized into bytes and transmitted. This occurs before the packets are sent.

In a P4 programme, the protocol header format, a parse graph for the various headers, the definitions of tables along with their match and action forms, and finally the control flow that determines the order in which these tables process packets are all specified. All of these components are referred to as "headers." "Headers" is the collective noun for each of these individual components. During the compilation process, this software is responsible for making the configuration choices for the hardware. During runtime, it is the responsibility of the control plane to populate the tables with entries. The rules that are created in the tables are what are utilized to determine how network packets should be handled. Writing apps that use the P4 platform involves adhering to a certain syntax that is outlined in the P4 standard.

In order for network operators to be able to compile HCCLBA for usage on any hardware target that is supported by P4, they are required to learn how to programme HCCLBA in P4. This is a prerequisite for network operators. Network operators do not have to make an investment in new hardware in order to tweak and recompile the P4 software that they use since it is viable for them to do so and allows them to change settings in addition to the core HCCLBA logic. HCCLBA P4 As a consequence of the increased interest in P4, a number of switch manufacturers are likely to embed P4 compilers into their hardware. This is one of the potential outcomes of this trend. Because of this, in the not-too-distant future, operators will be able to write HCCLBA on switches that are equipped with compilers of this

kind.

Evaluation

In this part, we show the efficacy of the HCCLBA load balancer by implementing it in the ns-2 discrete event simulator and comparing it with the following other load balancing schemes. ns-2 is a simulator that simulates discrete events.:

1. **ECMP:** The next hop for each flow is established by performing a hash operation on the flow's five-tuple structure (src IP, dest IP, src port, dest port, protocol).
2. **CONGA':** Congestion-aware data-plane load balancing may also be accomplished with CONGA, which is the most similar alternative to HCCLBA. CONGA, on the other hand, was developed exclusively for two-tier systems.

The architecture of the plant's leaves and spines. According to the authors of this research, CONGA should be applied to each pod, and ECMP should be applied to cross-pod traffic at the flowlet level. Both of these should be done in order to optimize performance. This strategy involves hashing the six-tuple tuple, which is comprised of the flow's five-tuple as well as the flowlet ID (which is incremented every time a new flowlet is detected at a switch). This hash is utilized by each and every switch in the network in order to calculate the next hop for each flowlet. CONGA is the name given to this load-balancing approach in the evaluation findings that we have compiled.

Parameters: In the experiment that we are doing, the behavior of the system is affected by two primary factors. As suggested in earlier research, the flowlet inter-packet gap is modified to be of the same order as the network RTT in order to cut down on the amount of packet reordering that occurs at the receiver. During our tests, we used a flowlet spacing of one hundred nanoseconds. In every one of our tests, the probe frequency was set at 200 seconds unless otherwise specified.

Conclusion

This paper presents a technique known as HCCLBA, which is an efficient load-balancing strategy designed for programmable data planes. Its name comes from the acronym for "high-capacity load-balancing architecture" (Hop-by-hop Consumption Conscious Load Balancing Architecture). When it comes to applying a distance-vector approach to the process of broadcasting information on network utilization to network switches, the HCCLBA makes use of periodic probes in order to achieve the desired results. Switches do not record information on congestion for each destination; rather, they record information regarding the next best way and the volume of traffic that flows along that path. In addition, HCCLBA does not require an additional source routing mechanism since all it does is find the next hop; it does not map out the full path that must be forwarded. This is because all it does is identify the next hop. It is feasible to avoid using routes that are no longer operational if the data on their use are immediately updated whenever problems develop.

We found that HCCLBA was both more effective at load balancing and more scalable than the other solutions when we compared it to comparable systems that are already in place. In addition to being

efficient, HCCLBA is simple enough to be carried out at line rate in the data plane on upcoming designs of programmable switches. This will be a significant advantage for these switches. Because of this, it is able to rapidly adjust to the unpredictable nature of the workloads in a data center. This research has looked at the empirical characteristics of the dynamic behavior of HCCLBA, and an inquiry into its optimality and stability will shed light on that behavior.

Reference

[1] Traffic classification and sifting to improve TDM-EPON fronthaul upstream efficiency: Yu Wu;Massimo Tornatore;Yongli Zhao;Biswanath Mukherjee, IEEE/OSA Journal of Optical Communications and Networking_2018.
[2] Joint Computation Offloading, Power Allocation, and Channel Assignment for 5G- Enabled Traffic Management Systems: Zhaolong Ning;Xiaojie Wang;Joel J. P. C. Rodrigues;Feng Xia, IEEE Transactions on Industrial Informatics_2019.
[3] Utility-Optimized Flow-Level Bandwidth Allocation in Hybrid SDNs: Xiaohong Huang;Tingting Yuan;Maode Ma, IEEE Access_2018.
[4] Joint optimal transceiver placement and resource allocation schemes for redirected cooperative hybrid FSO/mmW 5G fronthaul networks: Mahmoud A. Hasabelnaby;Hossam A.I. Selmy;Moawad I. Dessouky, IEEE/OSA Journal of Optical Communications and Networking_2018.
[5] Toward Traffic Patterns in High-Speed Railway Communication Systems: Power Allocation and Access Selection: Jiaxun Lu;Ke Xiong;Xuhong Chen;Pingyi Fan, IEEE Transactions on Vehicular Technology_2018.
[6] A Hybrid Downlink Scheduling Approach for Multi-Traffic Classes in LTE Wireless Systems: Moustafa M. Nasralla, IEEE Access_2020.
[7] Dynamic Resource Allocation With RAN Slicing and Scheduling for uRLLC and eMBB Hybrid Services: Lei Feng;Yueqi Zi;Wenjing Li;Fanqing Zhou;Peng Yu;Michel Kadoch, IEEE Access_2020.
[8] Energy-Efficient Resource Allocation With Hybrid TDMA–NOMA for Cellular-Enabled Machine-to-Machine Communications: Zeming Li;Jinsong Gui, IEEE Access_2019.
[9] Achieving Near-Optimal Traffic Engineering Using a Distributed Algorithm in Hybrid SDN: Cheng Ren;Shiwei Bai;Yu Wang;Yaxin Li, IEEE Access_2020.
[10] Resource Allocation and HARQ Optimization for URLLC Traffic in 5G Wireless Networks: Arjun Anand;Gustavo de Veciana, IEEE Journal on Selected Areas in Communications_2018.
[11] Optimization of MAC Frame Slots and Power in Hybrid VLC/RF Networks: Madiha Amjad;Hassaan Khaliq Qureshi;Syed Ali Hassan;Arsalan Ahmad;Sobia Jangsher, IEEE Access_2020.