2
Related Work

The recent trend of cloud computing emphasizes the need for the designing of scalable and efficient data center which is capable enough to handling the present as well as sustain for the forthcoming requirements. To meet the diverse requirements of applications, a conscious design of the data center network infrastructure and its protocols is a must. As highlighted previously, the performance of a distributed application depends on the network. Therefore the researchers are interested in building the infrastructure as well as optimally utilize these resources of the data center network. Since the traffic in data center is bursty and unpredictable, a static allocation of network resources is not sufficient. The requirements of designing data center and the Internet are different. The data centers are designed for business purposes which impose some financial restriction in terms of reasonable price. Contrary to this, the Internet was designed for interconnecting different networks and to act as a bit-pipe between the networks. Hence, recently a significant amount of research is being conducted on various aspects of the data center networks. The focus of the thesis is on reducing the latency of flows in data center networks. The state of the art research from literature addresses this problem through transport and/or network layer mechanisms. The flow completion time can be reduced by taking the inherent features of the topologies (like multiple paths, hop count) and scheduling and prioritizing the traffic at network switches. Hence, this chapter has been organized broadly into three sections, namely Data center topologies, Transport layer protocols, and Multi-path load-balancing protocols. Further each section is divided into two subsections, wherein the first subsection discusses a mechanism to classify the proposals and the second subsection discusses the proposals in detail.

2.1 TOPOLOGIES FOR DATA CENTER NETWORK

The topology plays a significant role in the performance of data centers, since it acts as a back-plane of the data center infrastructure. The role of the topology within the core infrastructure of data center is similar to that of the nervous system in human body. The network topology is a bedrock for any data center, since the interconnection of all computing elements can be achieved through it. The design of topology of a data center is very crucial for satisfying business needs. Hence it is important to discuss and understand the requirements of data center topology. Few of the basic demands of data center topologies are scalability, cost effectiveness and be capable of providing high bisectional bandwidth. In order to meet the future requirements with minimum modifications, a data center topology must be scalable. Which means that adding or removing some computational resource should not require many changes and should not affect the existing topology. In a scalable network, as the number of hosts increases, both the bisectional bandwidth delivered and the associated cost of infrastructure should also be increasing linearly. The bisectional bandwidth is very important for the performance of any application. In lower bisectional bandwidth networks, traffic of one service impacts the availability and performance of other co-located services. If all servers associated with an application can communicate with each other at their maximum interface bandwidth, the performance of the application will be high. On the other hand, the application is network restricted, if servers associated with the application are not able to achieve the desired bandwidth.

In order to increase the reliability and security, the servers must be allocated and re-located, as per the need, at different locations in the data centers. For seamless relocation of these servers, without hampering the performance of the applications, addresses associated with them should be location independent. In order to perform location independent server placement, it is advisable to have high bisectional bandwidth. In other words, the servers in data center can be placed anywhere if the bandwidth between them does not depend on their location. Hence, it is the responsibility of the topology designer to design a high bandwidth network for supporting location independent server placement and seamless relocation of servers. For better understanding of topologies in data center, few necessary definitions are provided here.

Bisectional Width: If a network is divided into two equal parts by using smallest cut, the minimum number of links connecting these parts, forms the bisectional width of the network. In other words the minimal number of links required for partitioning a network into two parts of equal size is called bisectional width. With the large bisectional width, a network is more resilient against failures.

Bisectional Bandwidth : If a network is divided into two equal parts the minimum achievable bandwidth between them is referred to bisectional bandwidth. In other words, bisectional bandwidth can be defined as the sum of the bandwidth of links defined by bisectional width. It is the measure of worst case network capacity.

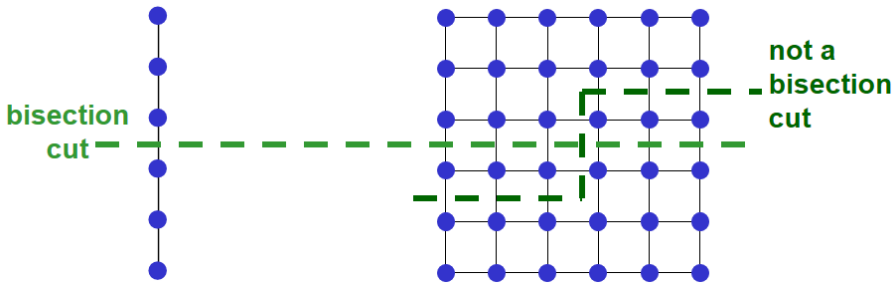


Figure 2.1. : Bisectional width of network

An example of bisectional bandwidth is shown in Figure 2.1. The figure shows, a line and grid topologies that are divided into two equal parts. For line topology a single link can partition the network. For the grid topology there are minimum six links should be removed to partition the network into two equal parts. Hence, bisectional width of the line and grid topologies, which are shown in the figure are, one and six respectively. Similarly bisectional bandwidth is the sum of the bandwidth of these links.

Oversubscription factor: An oversubscription factor is defined as the ratio of bandwidth at the server or host layer and the total bandwidth required at the core (i.e., the top most) layer. In other words, it is the ratio of bandwidth between the servers connected to the same ToR switch and the bandwidth between the servers connected to different ToR switches [Lebiednik et al., 2016]. The oversubscription is a way to reduce the cost of the infrastructure of the network. The basic assumption behind the oversubscription is that all servers do not transmit the data simultaneously. Based on this intuition, the designers of data center topology reduce the bandwidth at the top layer of the network.

In a topology with oversubscription of 1:1, a host may communicate with any other host with full of interface bandwidth. For example all servers connected through just a ToR switch always has 1:1 oversubscription ratio. In a topology with oversubscription of 5:1, only 20% of

the interface bandwidth is available to all hosts for some communication activity. The unequal bandwidth between the nodes in data center can lead to an entangled application design. It is also possible that due to high oversubscription ratio, the data center may be fragmented into server pools and may create a congestion hot-spot although there exists a spare capacity in the network.

VLAN: In computer networking, Virtual LAN (VLAN) is a technique through which a single Layer-2 broadcast network is divided into multiple distinct small sized broadcast domains [Greenberg et al., 2008]. These sub domains are mutually isolated and routers are responsible for passing packets between them. This is required as the broadcast nature of Layer-2 technology limits the size, and performance of the network. VLAN technology can help to increase the performance by creating multiple sub networks of smaller size. In VLANs, a 12 bit VLAN-ID is used for tagging the traffic. In static VLAN configuration, ports of a switch are mapped to a VLAN-ID, whereas in dynamic configuration the source MAC addresses are mapped to a set of ports. However, VLAN itself is a broadcast domain. Hence, in order to keep the broadcast domain small (increase scalability), VLAN must be setup between small number of racks. Since a large number of VLANs increase the overhead of resource allocation and configuration at the switch, the number of VLANs in the network must be kept small.

2.1.1 Classification of data center topologies

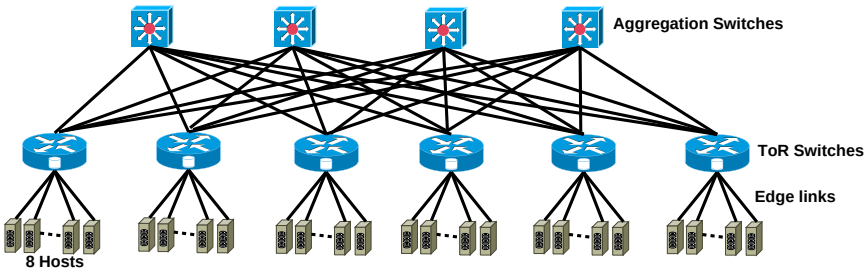
The topologies used in data center can be classified in different ways. In literature, there exist six ways of classifying data center networks. These are on the basis of the number of layers, the technology being used, the transmission media being used, the mobility of elements, the routing capabilities, and the base structure. Based on these ways the topologies can be grouped together with one another.

2-tier and 3-tier topology

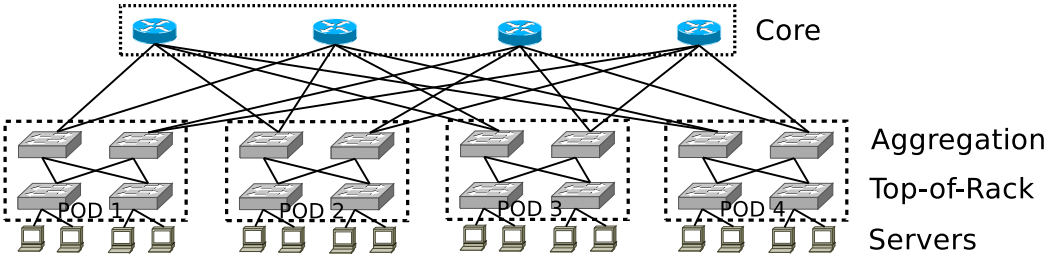
In 2-tier topology there are only two layers for connectivity in the network. These layers are Aggregation layer and Edge (Top of Rack) layer as can be seen in Figure 2.2(a). Generally people design small size networks with 2-tier topology. In other words, the number of servers in a 2-tier topology is less than the number of servers in 3-tier topology. With the help of Core layer, multiple 2-tier topologies are added together to form a 3-tier topology as shown in Figure 2.2(b). At the higher tier of topology switches with higher speeds are required, which tend to be more expensive and power hungry.

Layer-2 and Layer-3 topology

Based on technology, the data center topology can be implemented on either Layer-2 or Layer-3 switches, where each type of switches has its own advantages and disadvantages. Layer-2 switches are less expensive and require less administration. Further, these are self configurable and the forwarding tables are populated automatically. Switches in Layer-2 network use spanning tree protocol for avoiding the forwarding loops. This spanning tree protocol limits the performance of the network. Due to the spanning tree protocol, end hosts cannot utilize the multiple paths that are available in the topology. Layer-2 switches work, based on broadcast transmission, hence the scalability is a big issue. The functions related to filtering and QoS, are also not supported by Layer-2 switches. On the other hand, Layer-3 switches are comparatively expensive and impose administrative burden. Adding a new switch requires manual configuration. The scalability is not an issue with Layer-3 networks as the broadcast traffic is confined to a network. The migration of VM from one cluster to another cluster is a problem in the network of Layer-3 switches. As the addresses of VMs in Layer-3 network are associated with network address, migration from one cluster to another requires changing the addresses of VMs. In order to reduce the scalability problem in Layer-2 network, VLANs are used as a mid-way solution between Layer-2 and Layer-3



(a) 2-tier topology



(b) 3-tier topology

Figure 2.2. : An example of 2 and 3-tier topologies

switches. The VLANs also use spanning tree protocol (can not use multipath), and switches in the VLANs also need to do bookkeeping of all hosts which increases the overhead.

Electrical and Optical

Based on transmission media, data center network can be divided into electrical packet switching or optical circuit switching. c-Through [Wang et al., 2010] and Helios [Farrington et al., 2010] are examples of the data center topology that uses optical circuit switching. The optical circuit switching provides higher bandwidth compared to packet switching, but at the granularity of a packet it can not provide full bisectional bandwidth. The conversion of electrical signal to light and light to electrical signal in optical switching are down side of this technology. Due to physical rotation of mirrors, switching in optical technology (with MEMS switches) also takes a long time (order of few milliseconds). Optical transmissions are less affected by electromagnetic noise and the optical switches consume low energy as well.

Recursive and non-recursive

The topology which can define itself in its own form, is called recursive topology. In other words, if a topology can increase its size by combining its multiple smaller topologies, the topology can be defined as recursive topology. Dcell [Guo et al., 2008] and Bcube [Guo et al., 2009] are the few examples of recursive topology in data center network. On the other hand a non-recursive topology cannot be defined in its basic form. The recursive topologies are easily scalable, as they just need to add many basic structures to increase the size.

Fixed and Flexible Topology

In fixed topology structures, all elements of the network are fixed. In case of flexible topology structures, some network elements like servers, switches, and links connecting switches, and servers are packed in a container. In order to increase the size of the network such containers can be added to the existing network topology. Since these containers are removable, there is no need to be tied

to a fixed location. An organization can re-locate the containers based on the requirement, and the whole network topology can also be mobile.

Switch and Server-Centric

In server-centric topology, the servers not only act as end hosts but also as relay nodes [Chen et al., 2011]. Whereas in switch-centric topology, the servers act only as end hosts, and switches are responsible for forwarding the packets of the servers. In a server-centric topology, a server must require multiple network interface cards for future expansions and scaling. Servers in switch-centric topology have only one network interface card; switches are equipped with multiple interfaces, further switches are responsible for connecting the servers. A server should remained powered at all times (even no processing at server) in server-centric topology, since a powered-off server impacts on the network connectivity. Due to forwarding responsibility at server, such a topology increases per-packet latency.

2.1.2 Proposals in literature

The significance and the requirement of designing an efficient topology in data center has been discussed earlier. For fulfilling these needs, researchers from different backgrounds come up with various proposals. In this section, some of the proposed topology structures for data center network are described.

Tree-based

Topologies in older data centers are based on a tree-based hierarchical structure with 2 or 3-tiers (as per the need) [Wang et al., 2014]. Tree-based topology structures are oversubscribed at higher layers. Hence, the hosts under one ToR switch can communicate at full bandwidth, but moving up in the hierarchy limits the bandwidth of topology. The bisectional width of tree topology is 1. The problems with this topology structure are low bisectional bandwidth, poor scalability, and single point of failure.

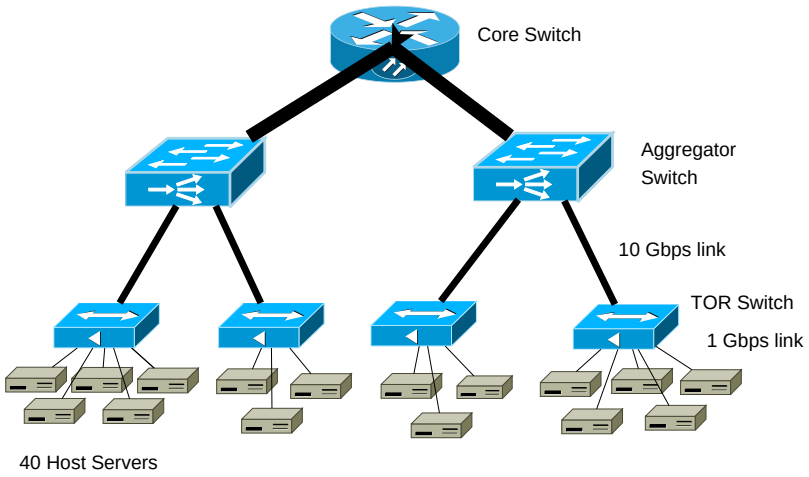


Figure 2.3. : An example of Tree-based topology

Fat-Tree

Fat-Tree topology [Al-Fares et al., 2008] consists of identical k-port Ethernet switches. With the help of Fat-Tree topology a large port switch can be constituted, through the multistage topolo-

gies of small port switches. In Fat-Tree topology pod is a basic replicable block. This topology consists of k pods (same as the number of ports of a switch) where each pod containing two layers, each layers having k/2 switches. All lower layer switches in the pod are connected to k/2 hosts. The remaining k/2 ports of these lower layer switches are connected to k/2 switches of the aggregation layer. There are $k^2/4$, core switches are required to connect these pods. One port of every core switch is connected to each of the k pods. A Fat-Tree topology can connect $k^3/4$ hosts using $5k^2/4$ commodity switches. This topology provides $k^2/4$ different equal cost paths between any two end hosts that are in different pods.

Fat-Tree is a non-blocking structure which provides a full end-to-end bisection bandwidth. In other words oversubscription ratio of Fat-Tree topology is 1:1. Fat-Tree topology consists of identical and inexpensive commodity switches. The cabling in Fat-Tree topology is complex.

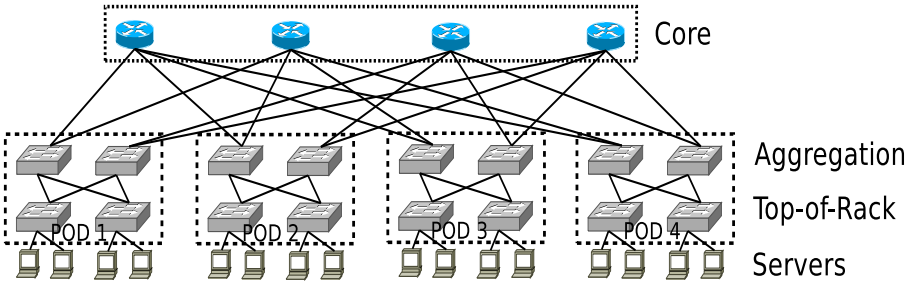


Figure 2.4. : An example of k=4 Fat-Tree topology

BCube

BCube [Guo et al., 2009] is an example of flexible topology architecture with modular data center (MDC) using commodity switches. This topology is based on the server-centric approach, hence a server is responsible for generating, consuming and forwarding the data packets. In BCube, the routing intelligence is placed on the servers. BCube works with mini-switches. Servers in BCube topology will have multiple ports and switches connecting these servers. In this topology a switch is responsible for connecting the servers, and direct links between the servers do not exist. The structure of BCube is recursively defined, in $BCube_0$ there are simply n servers connected by an n-port switch. $BCube_k$ is constructed with n $BCube_{k-1}$ structures. This $BCube_k$ network structure will have n^k switches. Every switch connecting the same index server from all the $BCube_{k-1}$ structures. In $BCube_k$ maximum hamming distance between any two servers is k+1. In other words, the diameter, which is the longest shortest path distance between all the server pairs of a $BCube_k$ is k+1. In BCube topology, it is possible to connect 4096 servers with 8-port mini-switches in BCube3. Figure 2.5 shows a 2 level $BCube_1$ structure with n=4. Source routing is used in BCube. While forwarding a packet an intermediate node must ensure that hamming distance between the forwarding node and the destination node is being decreased. For finding optimal path and to deal with failures in the network, a periodic search is performed.

DCell

DCell [Guo et al., 2008] is also a server-centric and recursive topology structure. In DCell topology all servers will have multiple Network Interface Card (NIC). In the basic structure of DCell which is called level-0 structure, n servers are connected to a commodity network switch. This switch is used to connect the servers within a DCell of level-0. Higher level structures in DCell contain multiple lower level DCells structures. The level-1 of DCell topology is built by n + 1, DCells of level-0 structures. One NIC of each server in every level-0 DCell is connected with a server in another level-0 DCell. If a level-0 DCell is treated as a virtual node, the whole topology

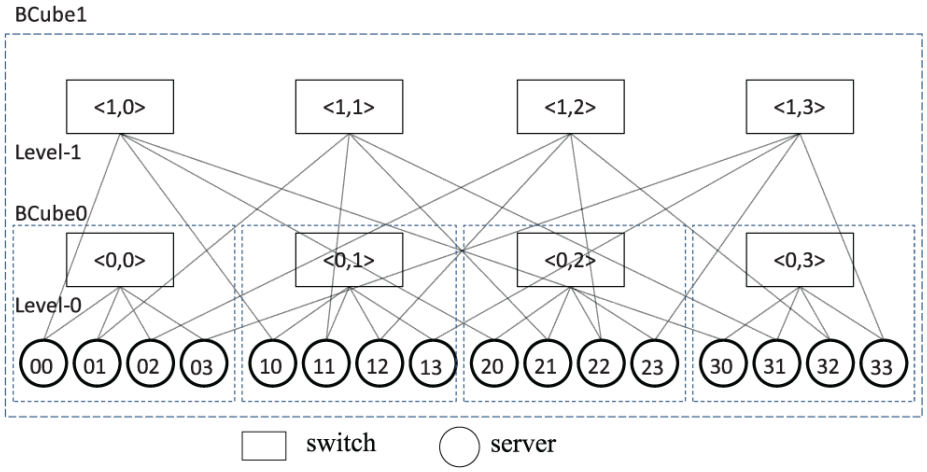


Figure 2.5. : Example of n=4, Bcube1 topology [Guo et al., 2009]

becomes a complete graph. An example of level-1 DCell with n = 4 is shown in Figure 2.6. DCell is highly scalable, i.e., the number of servers in this topology increases double-exponentially with the number of NIC ports in a server.

With Dcell topology structure more than 3.26 million servers with an average diameter of less than 10 (level=3, n=6) can be constructed. For routing in a DCell network, a packet needs to traverse from source to common ancestor DCell, a link connecting the previous level DCells and the destination. Finally, DCell topology is highly scalable and it provides high network capacity and fault-tolerant topology structure.

VL2

VL2 [Greenberg et al., 2009] uses low-end commodity switches to build tree-based topology. The difference between VL2 and Fat-Tree topologies is in the way in which the core and aggregation switches are connected. In VL2, core switches and aggregation switches form a complete bipartite graph as shown in Figure 2.7. The number of cables in VL2 is less compared to that in Fat-Tree. VL2 uses lower speed link for server to switch and higher speed links for switch to switch connections, e.g., 1 Gbps for server to switch links and 10 Gbps for switch to switch links. It uses directory system for enforcing access control in the network. This directory system secures a service from being flooded by other services. An additional IP address is used for mapping within the directory system through IP in IP encapsulation. For routing, VL2 uses both VLB and ECMP routing.

PortLand

PortLand [Niranjan Mysore et al., 2009] is a scalable, fault-tolerant Layer-2 routing and forwarding protocol for data center. It is a 3-tier architecture with Clos topology like Fat-Tree. It uses logically centralized fabric manager, which is responsible for maintaining configuration information of the topology. Switches in PortLand use additional pseudo MAC (PMAC) address which is based on a hierarchical addressing. This address helps to decide the next hop. The source calculates the routing path on behalf of the switches. Since PortLand uses MAC layer addresses for addressing the hosts, VM migration is possible.

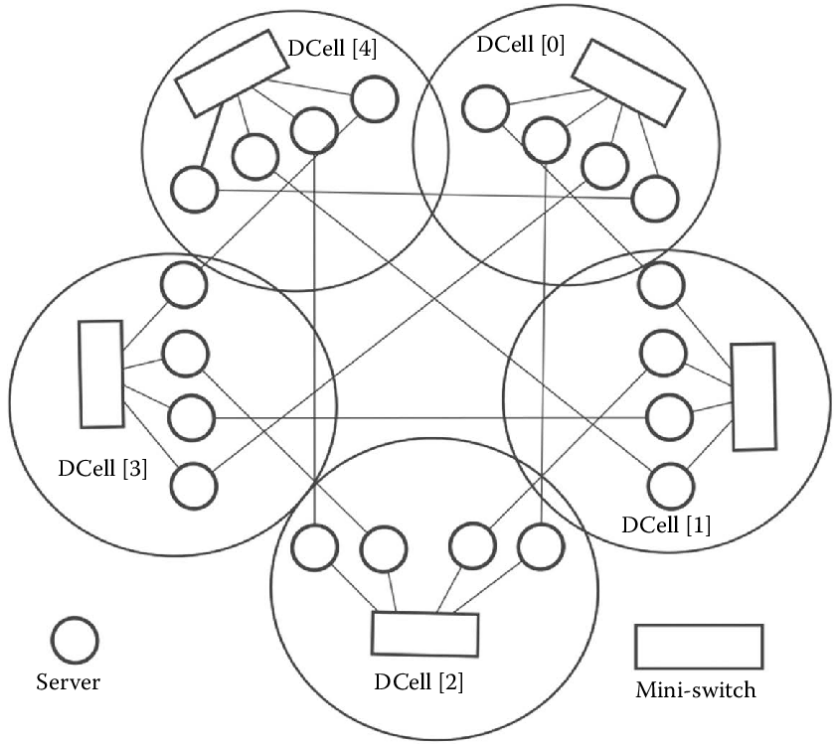


Figure 2.6. : Example of n=4, DCell topology [Guo et al., 2008]

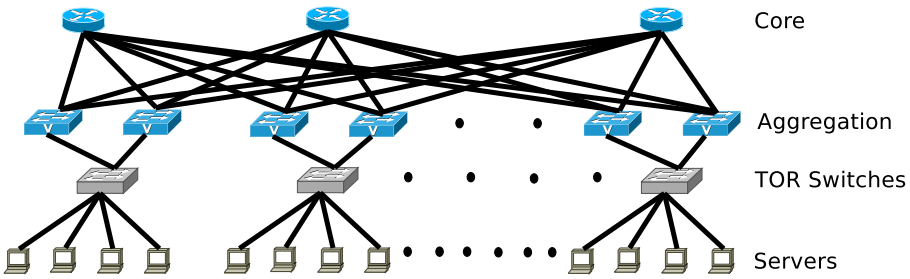


Figure 2.7. : An example of VL2 topology

SPAIN

In order to reduce the broadcast domain in Layer-2 data center and use multiple paths, SPAIN [Mudigonda et al., 2010] disjoints the network into multiple VLANs. In SPAIN, for announcing their identity, capabilities and neighbors, switches use Link Layer Discovery Protocol (LLDP). The LLDP collects the topology information by using Simple Network Management Protocol (SNMP). On the basis of this topology information, SPAIN computes the VLAN assignment. The computed VLAN configurations are installed into the switches via SNMP. Since SPAIN spreads traffic across multiple VLANs at flow-level, it achieves high bisection bandwidth and increases fault tolerance.

c-Through

c-Through [Wang et al., 2010] is based on HyPaC (Hybrid Packet and Circuited network), which integrates optical circuit switching technology into traditional packet switched technology. c-Through is a tree-based topology. Further, in c-Through ToR switches are connected by optical links. c-Through uses large TCP socket buffers for buffering the traffic. When this buffered traffic reaches the threshold and the queue size variations also reach at some other threshold (1 MB), the centralized manager is informed in c-Through. The manager figures out optical path configuration for this traffic. c-Through uses VLAN to differentiate the electrical and optical networks.

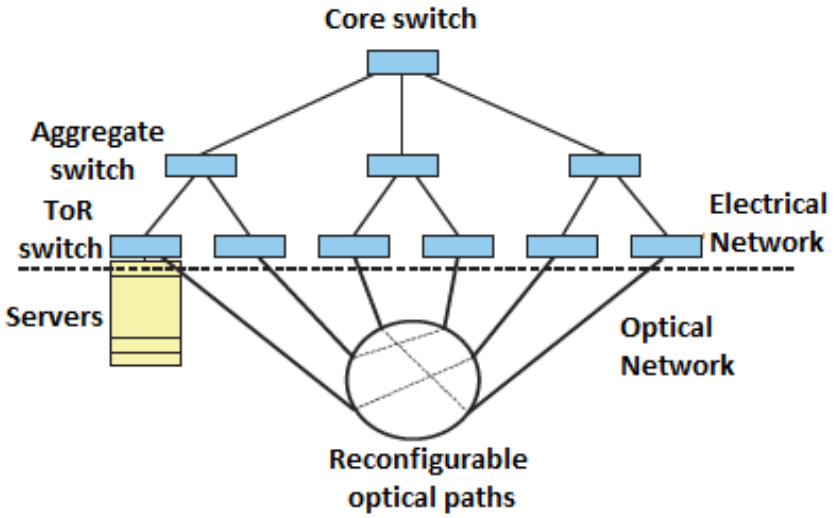


Figure 2.8. : Example of c-Through topology [Wang et al., 2010]

Helios

Helios [Farrington et al., 2010] can be defined as a 2-level multi-rooted tree of pod switches and core switches. It is also based on HyPaC like c-Through. Core switches are of both types, electrical switches and optical switches. In order to carry optical signals of different wavelengths on a single fiber, multiplexers ("Mux") are used in Helios. On pod switches Helios measures traffic flows and estimates bandwidth demand through max-min fairness. Similar to c-Through, Helios also calculates optical path configuration, which is based on max-weighted matching. Unlike c-Through, in which both the electrical and the optical switches are dynamically configured based on the computed configuration, Helios is not dynamically configured.

JellyFish

Jellyfish [Singla et al., 2012] is a degree-bounded random regular graph at the Top of Rack layer. It provides high bandwidth and flexibility network topology. It is built by n-port switches, in which r ports are used to connect the other switches (like random graph) and the remaining $n-r$ ports are used to connect the hosts. At the time of adding a new link, a pair of not directly connected switches (with one unused port) are selected randomly, for connecting them through a link. This topology is built on the basis of random graphs. Hence, it provides low average path length and higher throughput compared to other symmetric topologies like Fat-Tree. Although, for practical use of this topology, problems like routing, packaging need to be investigated.

At the end of discussion on topology structures, Table 2.1 summarizes the existing proposals on topology structures in data center.

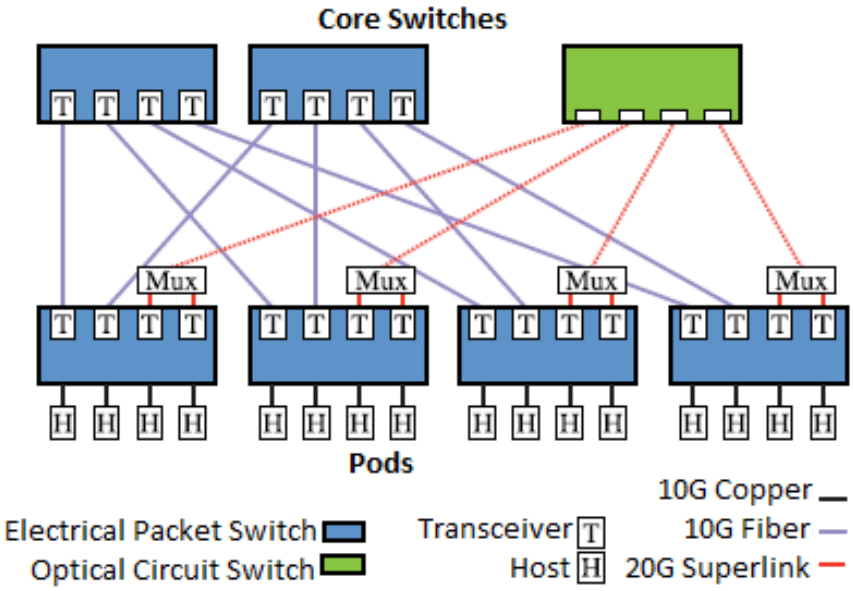


Figure 2.9. : Example of Helios topology [Farrington et al., 2010]

Proposal	Layers	Technology	Media	Structure	Mobility	Forwarder
Tree-based	2-tier & 3-tier	L-2 & L-3	Electrical	Non-recursive	Fixed	Switch
Fat-Tree	3-tier	L-3	Electrical	Non-recursive	Fixed	Switch
BCube		L-3	Electrical	Recursive	Flexible	Server
Dcell		L-3	Electrical	Recursive	Fixed	Server
VL2	3-tier	L-3	Electrical	Non-recursive	Fixed	Switch
PortLand	3-tier	L-2	Electrical	Non-recursive	Fixed	Switch
SPAIN		L-2	Electrical	Non-recursive	Fixed	Switch
c-Through	3-tier	L-3	Electrical & Optical	Non-recursive	Fixed	Switch
Helios	2-tier	L-3	Electrical & Optical	Non-recursive	Fixed	Switch
Jellyfish		L-2& L-3	Electrical	Non-recursive	Fixed	Switch

Table 2.1. : Summary of different topological structures proposed for data center networks

2.2 DATA TRANSPORT PROTOCOLS

As discussed in Chapter 1, the transport layer protocols used in the Internet are not suitable for data center networks. Unlike in the Internet, data center networks have different communication patterns, time scale of RTTs and requirements of deadline. These challenges in data center enforce researchers to rethink about transport layer solutions. In last few years, deadline aware protocols are designed to meet the deadline of delay sensitive flows. However, even with deadline aware solutions challenges remain as challenges [Krevat et al., 2007] for the following reasons.

- In the Internet, the underlying network is a best effort and unreliable and the packet is a basic unit to transfer the data. The deadlines can only be associated with flows but not to individual packets.

- Deadline sensitive flows are small in size and can be completed in few RTTs; the reservation-based schemes take more time as they require an additional RTT for reservation.

- All workers are synchronized and deadlines of them are approximately the same. This causes fan-in burst in aggregation/partition communication pattern.

In addition to these problems, RTT time scale in data center is in the order of microseconds. Due to the time scale of RTT, a delay based protocol faces problem as it can not infer the reason for packet loss. It may be possible that, in a delay based approach a small amount of delay may falsely be inferred as packet loss, and can be a cause of triggering the congestion avoidance mechanism at the sender [Vasudevan et al., 2009]. In case of dealing with latency, buffering of packets at intermediate switch increases the latency of the flow. That is low latency requires low buffering at the switch. However, buffering is helpful for achieving higher throughput and cope up with temporary oversubscription in the network. In aggregation/partition structure, as multiple workers send data to a master concurrently, the intermediate switches need to be oversubscribed.

2.2.1 Classification of data center protocols

Data center protocols can be classified as deadline aware and deadline agnostic protocols in terms of dealing with deadline information. Deadline aware solutions are not compatible with traditional solutions [Wilson et al., 2011] [Vamanan et al., 2012] [Rojas-Cessa et al., 2015]. These solutions perform better compared to deadline agnostic, since they can use the deadline information. Few protocols use flow size (the amount of data to be transferred by flow) information if deadline information is not available. The deadline aware solutions require modification in the interface between the application and transport layers. In literature, the solutions are classified into five main categories [Liu et al., 2013], which are based on minimizing queue length, rate base, priority scheduling, multipath connectivity and on-path aggregation.

Reducing Queue length at switch

Queuing time is the major factor in the latency of a flow as compared to the propagation and transmission times. Reducing queue length for delay sensitive flows help to achieve their deadlines. For reducing queuing time, proposals like DCTCP[Alizadeh et al., 2010], HULL[Alizadeh et al., 2012], NDP [Handley et al., 2017], L^2TCP[Munir et al., 2013], D^2TCP[Vamanan et al., 2012], use Explicit Congestion Notification (ECN) to inform the source about the congestion in the network. In response to the ECNs, the source reduces its congestion window. These approaches are reactive and mitigate the congestion only after it happens. Such schemes take more time to handle the situation, as congestion information reaches at the source after it comes back through the receiver. This delay of congestion information propagation is almost equal to the RTT. In order to handle aggressive traffic, in this duration, a large headroom is required at the switch in such schemes.

Resource reservation-based schemes

As discussed in Chapter 1, additive increase and multiplicative decrease takes multiple RTT to find the share of a flow in the network. These many RTTs will increase the latency of the flow. Hence in resource reservation schemes, a source request the network for the desired bandwidth according to its demand [Chen et al., 2013]. The network replies to the request based on some parameters. In this scheme a source directly comes to know the available bandwidth in the network without going through multiple RTTs. PDQ[Hong et al., 2012], PASE[Munir et al., 2014], D^3[Wilson et al., 2011] are resource reservation-based schemes. These schemes are too heavy

for small delay sensitive flows, which require few RTTs to finish. Such schemes require accurate information of the number of flows, the flow size and the path of the flow to calculate the rate precisely. Further, these schemes perform better with static and constant rate traffic, like real-time applications, whereas traffic in data center is dynamic with varying work loads. Due to pausing and proceeding the flows, the rate reservation-based schemes impose high flow switching overhead.

Prioritization of flows

In data center networks, elephant flows and bursty flows are responsible for increasing the queuing size. This increased queuing size further increases the latency for delay sensitive mice flows. A smart prioritization scheme may reduce this queuing time for mice flows by giving high priority for mice flows compared to elephant flows. However, how to assign priority to packets at the sender without having the knowledge of the deadline of other flows is a reasonable question. pFabric[Alizadeh et al., 2013], PASE[Munir et al., 2014], and NDP [Handley et al., 2017] are based on priority scheduling. Such a scheme imposes the requirement of multiple queues at the switch.

Multipath approach

Most of the data center topologies support the multiple paths between end hosts, and load on each path is statistically independent with regard to the other paths. With a careful transmission and receiving mechanism, these paths can be used for parallel transmission of data or reducing the latency of a flow. RepFlow[Xu and Li, 2013], DeTail[Zats et al., 2012], MPTCP [Raiciu et al., 2011] use multipath approach to reduce flow completion time.

On-path Aggregation

Middle boxes are in the trends of networking nowadays. These middle boxes have enough processing capability and are connected to intermediate devices. For reducing the processing at destination hosts, data is aggregated and processed at intermediate switches with the help of middle boxes. NetAgg[Mai et al., 2014] is based on, on-path aggregation approach at the switches. Such an approach requires additional hardware and violate end-to-end semantics.

2.2.2 Proposals in literature

The requirement for designing optimized latency sensitive protocols in data center has been discussed previously. The problems associated with traditional protocols in data center environment are highlighted in Chapter 1. For fulfilling the demands of data center applications, researchers have proposed several protocols for fully exploring the network capabilities. This section describes some of the protocols that have been proposed specifically for data center network.

DCTCP

DCTCP[Alizadeh et al., 2010] is a variant of traditional TCP and is based on reducing the queue size at the switch. It uses ECN to send back congestion information to the source. Nowadays, all switches used in the data centers are ECN enabled. In this proposal, if the queue length at a switch reaches some predefined threshold (K), the switch starts marking the packets with the Congestion Enable (CE) codepoint. Marking in DCTCP is based on instantaneous queue length, which helps the DCTCP to detect and react to the congestion before it happens. These CEs are sent back by the receiver in ACKs. The sender counts the number of CE marked packets for each window and applies moving average. The sender reduces its congestion window based on the extent of congestion at the switch. In DCTCP a source receives multi bit feedback from single bit CE marked packets and reduces the congestion window in proportion to the congestion experienced in the network; unlike TCP which halves its congestion window when it receives a CE marked packet. DCTCP is a deadline agnostic solution, hence it treats elephant and mice flows in

the same manner and reduces the congestion window for all flows equally.

$$Window_{new} = Window_{old} \times (1 - \frac{\text{Fraction of ECN marked packets}}{2})$$

DCTCP is a pioneering work in the latency of data center networks, which opened the research on specialized transport protocols for data center networks. DCTCP provides high throughput for elephant flows and low latency for mice flows simultaneously. It requires the modification at end hosts and the changes to the configuration at switches. DCTCP is implemented in Microsoft windows server 2012.

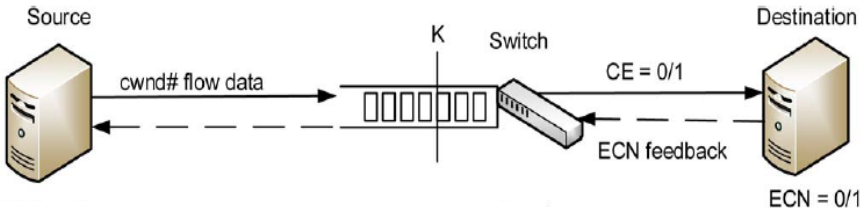


Figure 2.10. : DCTCP congestion control

HULL

HULL[Alizadeh et al., 2012] is also a solution which is based on reducing the queue size at the switch. The congestion control part of HULL is the same as that of DCTCP. It uses a phantom queue at the egress port of the switch for ECN marking, instead of the real queue as shown in Figure 2.11. These phantom queues are drained out slowly, compared to the original queues. HULL requires the packet pacer at the end hosts to curtail the negative effect of packet bursts. The basic idea of HULL design is to trade bandwidth for buffer space, which contributes to significant increase in latency by the use of phantom queue. HULL is based on the idea of renunciation of small amount of bandwidth for reducing average and tail latencies. Results show that, HULL can reduce the average and 99^{th} percentile packet latencies compared to DCTCP and TCP by sacrificing a small amount of effective bandwidth, which is a trade-off. Due to packet pacing and phantom queues, HULL requires modification at the end hosts and switches.

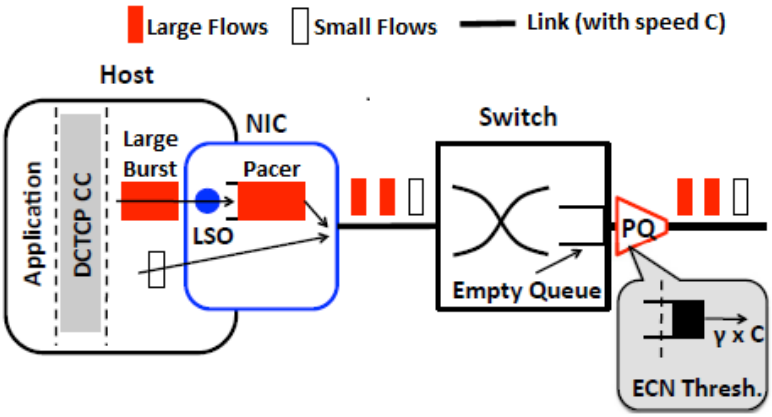


Figure 2.11. : HULL architecture [Alizadeh et al., 2012]

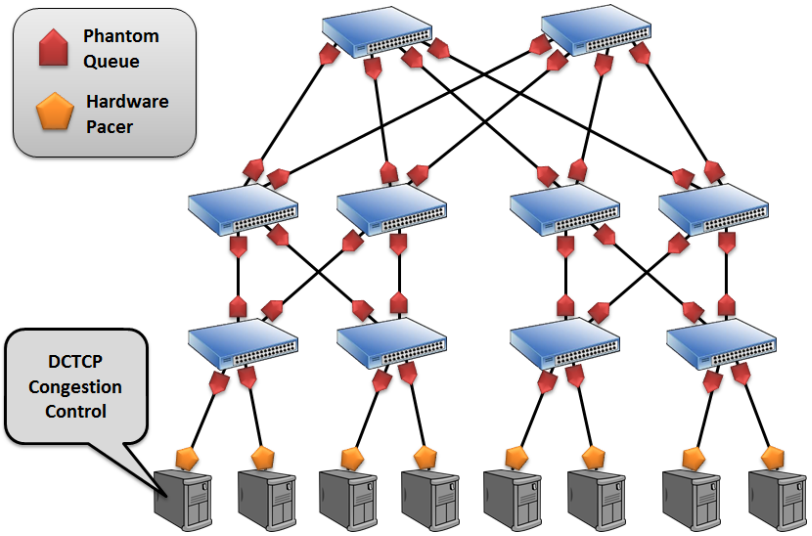


Figure 2.12. : HULL topology

D²TCP

D^{2}TCP [Vamanan et al., 2012] is based on DCTCP. It tries to reduce the queue size at the switch for latency sensitive flows. This is a deadline aware approach to reduce the latency in data center. D^{2}TCP takes advantage of deadline awareness, and at the time of reducing congestion window, it also takes care of deadline of the flow. It uses a gamma-correction function as shown in Figure 2.13, to decide the penalty of a flow, and further to reduce the congestion window. The penalty (p) of a flow is computed based on the ECN feedback and the urgency of flow (d). This urgency of flow is a function of flow size, current rate and remaining time of flow to achieve the deadline. Flows with far deadline back-off more compared to the flows having near deadline. Hence flows can achieve their deadline easily by giving up some amount of bandwidth for the flows that are in trouble. Further, the coordination among the flows is missing in D^{2}TCP, which can lead to under utilization of the network.

$$penalty(p) = (\text{Fraction of ECN marked packets})^d$$

where

$$d = \frac{\text{time needed for a flow to complete transmitting all its data with current rate}}{\text{time remaining until its deadline expires}}$$

The urgency of flow (d) captures the tightness of deadline.

L²TCP

L^{2}TCP [Munir et al., 2013] also treats flows differently by using gamma-correction function same as in D^{2}TCP. However it is a deadline agnostic approach and does not use deadline information for modifying congestion window. Instead of deadline information, L^{2}TCP assigns a weight to every flow and this weight is decreased as the flow progresses. When a flow is started, the weight of the flow is maximum and this weight decays with the transmitted data of the flow. The size of elephant flows is larger than the mice flows and the elephant flows persist for longer time. As the elephant

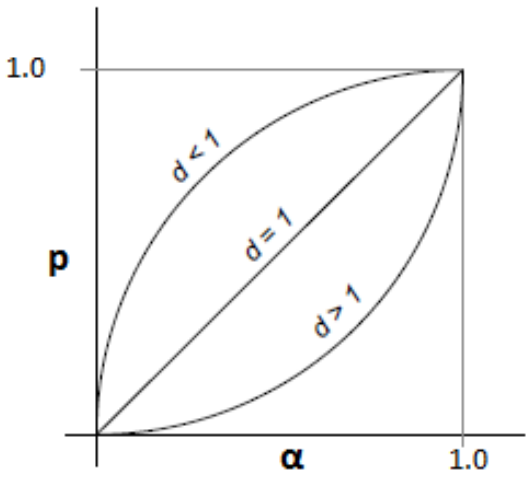


Figure 2.13. : Gamma-correction function [Vamanan et al., 2012]

flows spend longer time in the network, their weights will be low compared to those of mice flows. Further, elephant flows will back-off more compared to mice flows at the time of congestion. Due to this, mice flows complete faster than elephant flows. This approach of L^2TCP implicitly prioritizes the mice (new) flows over the elephant (older) flows with the help of weight. L^2TCP also ensures that, congestion window of mice flows increases faster compared to that of elephant flows in the absence of congestion.

PDQ

PDQ[Hong et al., 2012] is a resource reservation-based scheme. It is a preemptive scheduling scheme at flow-level, which enables flow preemption and uses deadline information for scheduling the flows. In this approach, a source requests the network for bandwidth by providing deadline information in packet header. The network allocates bandwidth to flows in descending order of the criticality. The network replies in the fields of protocol header as shown in Figure 2.14. A flow is accepted if all switches along the path accepts it, otherwise the flow will be paused. In PDQ a flow is terminated by a source, if it is not possible for the flow to achieve the deadline. By terminating unproductive flows a source avoids the wastage of bandwidth. PDQ is implemented with a single queue at the router and paused flows will wait at the sender. It uses earliest deadline first (EDF) for scheduling the flows, and if requests of two flows with same deadline arrive, then it goes with shortest job first (SJF) scheduling. Since switches need to be aware of the active flows going through them, they need to do the book-keeping of active flows. PDQ needs a shim layer between the network and transport layers. Also, it is not compatible with existing network hardware and software. PDQ is an expensive solution and can not scale well.

D^3

D^3[Wilson et al., 2011] is another resource reservation-based scheme. Authors of D^3 suggest that fair share policy of original TCP is not a good idea to achieve the deadline as shown in Figure 2.15. This fair share scheme may be a reason for missing the deadline of multiple flows. In a deadline aware network, at the time of congestion, it is possible that by slowing down some flows, we can ensure that the remaining flows do meet their deadlines. In other words, by assigning bandwidth for flows properly, it is possible to increase the number of flows which can achieve the deadline. D^3

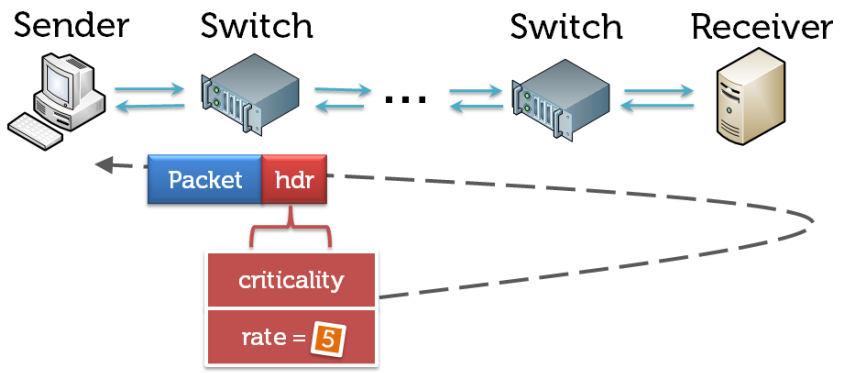


Figure 2.14. : Resource reservation-based scheme (PDQ)

uses deadline information for explicit bandwidth assignment and the bandwidth assignment is as per the need of the flows. In this scheme a source requests for the desired rate from the switches along the path to the destination, and intermediate switches satisfy incoming request as First Come First Serve (FCFS) manner, aiming to fulfill as many deadlines as possible. If enough bandwidth is not available to satisfy the request, a base rate is allocated for the flow, which allows the flow to send a header-only packet per RTT for the purpose of probing the network bandwidth in the future. D^3 is a centralized solution as compared to PDQ which is a distributed one. Due to FCFS rate allocation in D^3, the priority of requests may be inverted, as far deadline flows arriving slightly ahead allocated first than near deadline flows. The bandwidth of a flow depends on the order of its arrival and there are n! possible arrival orders for n flows. This approach imposes significant operational overhead and requires changes to applications, end hosts, and network switches.

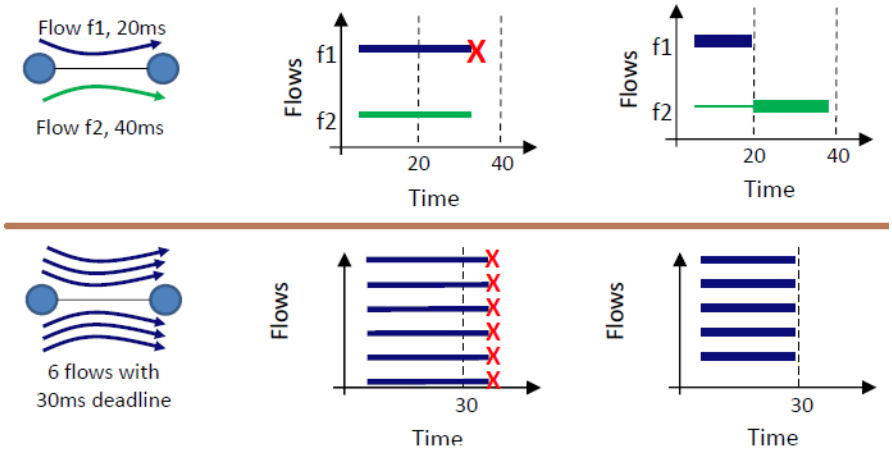


Figure 2.15. : Disadvantages of fair share policy for delay sensitive traffic

PASE

PASE[Munir et al., 2014] also uses resource reservation approach, which depends on in-network prioritization. This is also a centralized scheme; it satisfies all requests only at the entry level switch (ToR). In order to reduce the number of priority queues at the switch, PASE uses dynamic prioritization to map flows to limited number of priority queues. In PASE, a flow is

prioritized relatively based on its demand, and after prioritization, the flow is mapped to a limited number of priority queues. With the help of a few number of priority queues at the switch, PASE is able to reduce flow switching time. For example in Figure 2.16, 4 flows are mapped to 2 priority queues at time t_1 and at time t_2 when flow 1 is completed flow 2 is switched to a higher priority queue (reducing flow switching time). However, dynamic prioritization works well only with single hop flows. It uses delegation mechanism for inter-rack flows, in which total bandwidth of higher layer switch is divided into lower layer switches and these lower layer switches are responsible for the assignment of rate on behalf of the higher layer switches. For rate control PASE uses rate control mechanism of DCTCP.

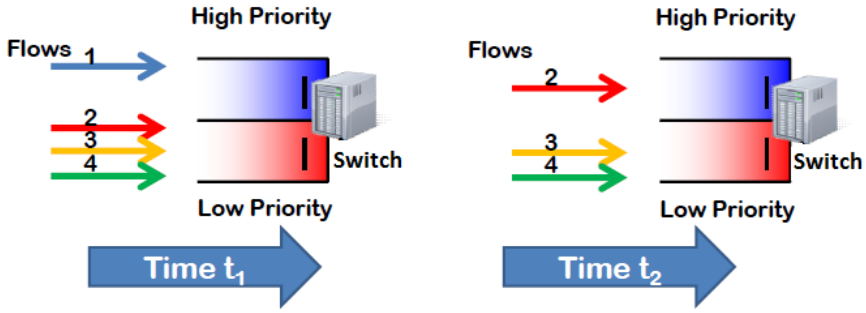


Figure 2.16. : Dynamic mapping of 4 flows to 2 queue and its advantage

Fastpass

Fastpass [Perry et al., 2014] is a centralized approach in which a global controller is responsible for admission control and routing of packets. The controller decides the time at which packets of a flow are to be inserted (or transmitted) in the network. The controller implements Max-Min fairness policy for allocating the time slot, in which time slots are allocated in least recently allocated first manner as shown in Figure 2.17. In order to handle the failure of centralized controller, Fastpass uses replication strategy. Fastpass tries to remove in-network queuing. However, due to the single centralized controller, requests are queued for allocation.

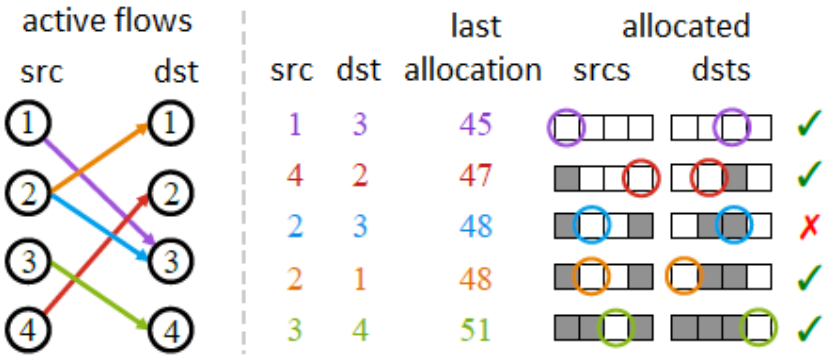


Figure 2.17. : Max-min timeslot allocation [Perry et al., 2014]

pFabric

pFabric[Alizadeh et al., 2013] is based on the priority scheduling of flow, and it decouples the flow scheduling from rate control. pFabric gives high priority to small flows by using remaining size of the flow for scheduling. In pFabric, the sender sends remaining size of the flow in the packet

header, and switch schedules the packets according to the shortest remaining size first policy as shown in Figure 2.18. In case of congestion packets are dropped based on their priority. The lower priority (higher in number) packets are dropped among all packets (incoming packet and existing packets in the buffer). In pFabric all flows are started at maximum rate, and if they observe high and persistent loss, they decrease their sending rate.

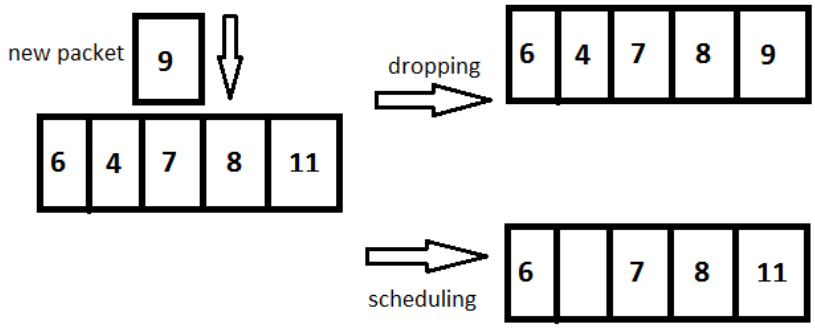


Figure 2.18. : Priority based scheduling/dropping in pFabric

Figure 2.18 shows the queue of a switch in which buffer is full with packets of different priority. When a new packet (priority 9) is received at the switch, it simply drops the lower priority packet (priority 11) and inserts the new packet (priority 9). For scheduling a packet, it selects the highest priority packet (priority 4).

RepFlow

RepFlow[Xu and Li, 2013] works on top of the transport layer and it follows a multipath approach. In RepFlow, with the help of ECMP a deadline sensitive flow is replicated over multiple paths. This is not like other multipath schemes like MPTCP or RPS, which split the flows. The replicated flow in RepFlow is exactly the same as its original one. Only its destination port is one more than original flow's destination port. Application at the receiver end uses the data of the flow whichever finishes first. The basic intuition behind RepFlow is that the congestion levels along different paths are statistically independent, and the probability of multiple paths experiencing a long queuing delay is much smaller. As the size of the deadline sensitive flow is small, overhead of replication is negligible. Hence, RepFlow opportunistically utilizes less congested path with the help of replication, without being able to reduce the queue length at a switch.

NetAgg

NetAgg[Mai et al., 2014] tries to solve buffer overflow problem with on-path aggregation approach at switches. In this scheme, instead of performing data aggregation at the edge servers, NetAgg uses middleboxes, connected by high bandwidth links to network switches, for data aggregation as shown in Figure 2.19. It specifies two requirements namely, associative and commutative for on-path data aggregation. The basic idea of NetAgg is that the size of the aggregated result is a small fraction of the intermediate data that is generated. Therefore the amount of network traffic is reduced at each hop. However, this approach depends on the application specific aggregation function at network devices.

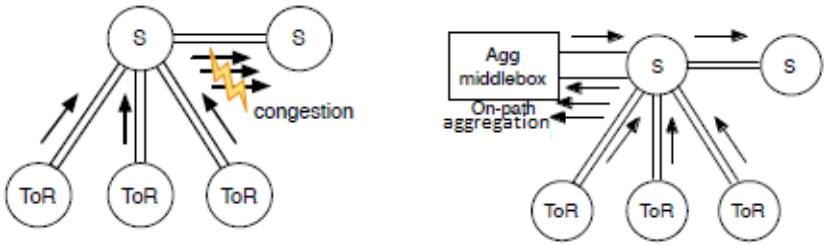


Figure 2.19. : On-path data aggregation [Mai et al., 2014]

DX

DX [Lee et al., 2015] uses delay as a congestion signal for high throughput, and low latency data center networks. It estimates accurate queuing delay in a data center through adding TCP options and modifying both sender and receiver as shown in Figure 2.20. DX assumes that with accurate latency feedback a sender can calculate the maximum number of packets in a network. This approach assumes that minimum RTT implies no queuing in the network. DX is a window based proposal, in which an additive increase and multiplicative decrease algorithm is performed based on average queuing delay. In DX congestion response is based on the ratio of the measured average queuing delay and calculation of the number of competing flows. In the circumstance of latency feedback being zero, it increases the congestion window. This approach is not compatible with the existing network software.

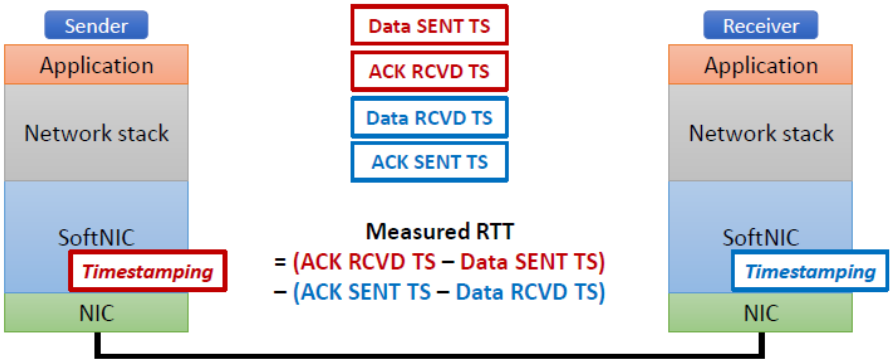


Figure 2.20. : Base RTT estimation in DX

QJUMP

QJUMP [Grosvenor et al., 2015] uses both rate limiting and prioritization of flow. It classifies traffic in different classes based on their requirements. A traffic with low latency requirement is assigned a high priority value with limited rate. On the other hand, a traffic requiring high throughput is assigned a low priority without any rate limited. QJUMP prioritizes the traffic based on the trade-off between the network latency and the throughput. In QJUMP a high priority packet can jump the packet of lower priority flows as shown in Figure 2.21. In other words, the packets of latency sensitive flows can jump the queue over the packets of throughput sensitive flows. However, QJUMP provides guaranteed latency for very low bandwidth traffic only.

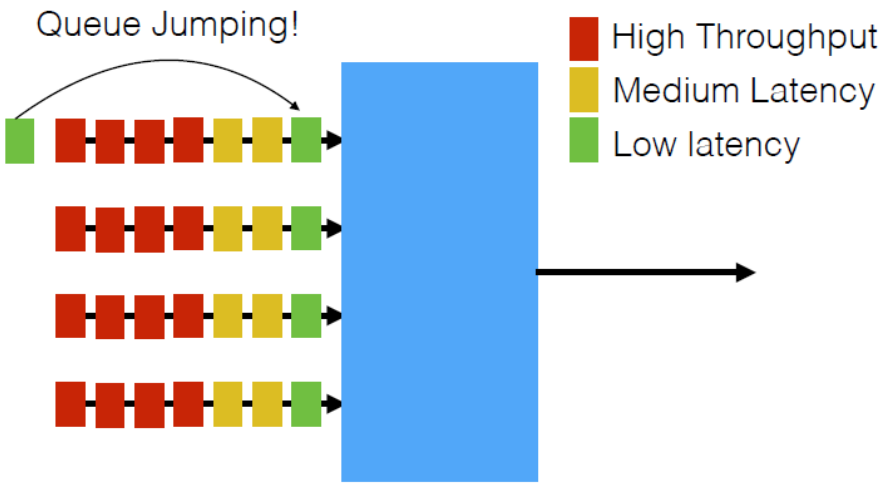


Figure 2.21. : QJUMP design

ICTCP

ICTCP [Wu et al., 2013] is a proactive approach to solve the incast problem by avoiding packet drops and timeouts. The authors suggested that avoiding incast is more appealing than recovering from losses. It changes TCP advertised window (at the receiver side) before a packet loss event. In this scheme, receiver adjusts its window for controlling the congestion. ICTCP aims to maintain a small window size which is good enough to avoid congestion while being large enough for achieving high throughput. This window adjustment at receiver depends on the available bandwidth at the last hop to the receiver. When the ratio of the difference between the measured and expected throughputs, and the expected throughput, is small, ICTCP increases the receiver's window otherwise decreases the window size. ICTCP works well at last hop of connection to mitigate the incast problem. However, at the switches other than the last hop, ICTCP can not solve the problem.

NDP

NDP [Handley et al., 2017] takes a different approach to simultaneously achieving both low delay and high throughput. NDP has no connection setup handshake, and allows flows to start sending instantly at full rate. It uses per-packet multipath load balancing, which avoids core network congestion at the expense of re-ordering, and switches uses an approach to trim the payloads of packets when a switch queue fills. With this approach a network become lossless for metadata (data about data), but payloads of packets are dropped. Due to this packet trimming approach, receiver gets a complete picture regarding incoming traffic, since headers of all the packets are received correctly.

NDP effects the whole networking stack, which include switch behavior, routing, and transport protocol as shown in Figure 2.22. To achieve minimal latency for short-flows, senders must send the packets in first RTT at line rate, NDP cannot probe network before starting the transmission. This idea works well most of the time. At the switch when the queue fills beyond a fixed threshold, rather than dropping a packet, the switch trims off the packet payload, queuing just the header. The rationale is that packets are not lost silently, allowing rapid re-transmission without waiting for a timeout. In NDP a switch maintains two queues: a lower priority queue for data packets and a higher priority queue for trimmed headers, ACKs and NACKs. NDP uses a receiver-

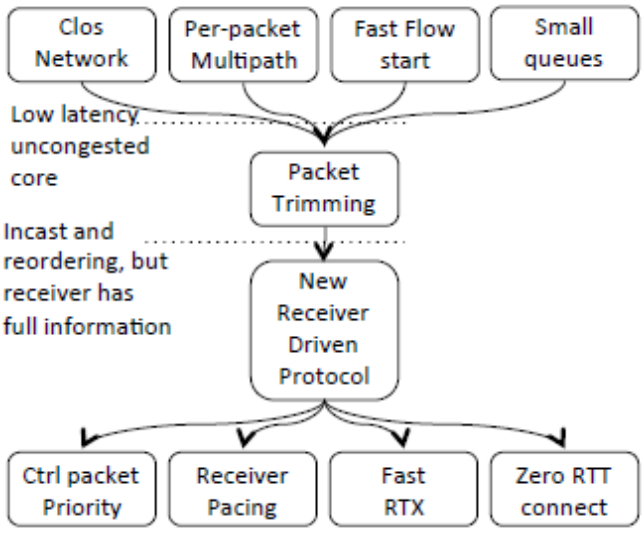


Figure 2.22. : Key components of NDP [Handley et al., 2017]

driven transport protocol designed specifically to take the advantages of multipath forwarding, packet trimming, and short switch queues.

NDP is very close to optimal in fully-provisioned folded Clos topologies. However in heavily oversubscribed networks where the core is persistently congested, NDP's aggressive design will lead to continuous packet trimming even after the first RTT. In asymmetric topologies, NDP will behave poorly because it will spray packets on the paths of different length that are costly to use when the network is heavily loaded.

At the end of discussion on data transport protocols, Table 2.2 summarizes the proposals of data transport protocol in data center network.

2.3 MULTIPATH LOAD-BALANCING PROTOCOLS

Today's networks tend to have multiple paths between the source and destination nodes. As discussed in the previous section, many data center topologies provide multiple equal cost paths between the end hosts. In order to utilize network resources efficiently, there should be a mechanism to distribute the user's data over the available paths concurrently. Data distribution over multiple paths is not only the problem of data centers, but for any topology structure where multiple paths between end hosts exist. Another issue with TCP/IP protocol stack is the implementation of the multipath capability. Transport layer by design is more concerned about network congestion level, flow rate, and RTT, while being unaware of its multipath capability. Network layer is designed for forwarding the packets; but utilizing multipath at the network layer with simple TCP will confuse the sender, since RTT of different paths are different[Ramaboli et al., 2012].

2.3.1 Classification of multipath protocols

Centralized and distributed solutions for utilizing the multiple paths have their own pros and cons, which have been already discussed in Chapter 1. In a data center, for small delay sensitive flows, centralized solutions impose high latency as they require to compute the path and installing entries in routing tables of the switches along the path. These schemes are too slow to

Proposal	Queue length	Rate Assignment	Priority based	Multipath
DCTCP	✓	×	×	×
HULL	✓	×	×	×
D^2TCP	✓	×	×	×
L^2TCP	✓	×	×	×
PDQ	×	✓	×	×
D^3	×	✓	×	×
PASE	×	✓	✓	×
pFabric	×	×	✓	×
RepFlow	×	×	×	✓
DX	✓	×	×	×
QJUMP	×	×	✓	×
NDP	×	×	×	✓

Table 2.2. : Summary of proposed data transport protocols for data center network

deal with the latencies for deadline sensitive small flows. The capability of centralized solutions also depends on the traffic characteristics. Traffic with low inter-arrival times between the flows, restricts the amount of processing required for each new flow at the central controller [Xia et al., 2017]. The performance of a centralized scheme also depends on the accuracy of information. In addition to centralized and distributed approaches, multipath solutions used in data centers can be classified into four prominent ways as based on solution's adaptiveness with time, simultaneously use of multipaths at flow-level or user level. Further a scheme can be classify as balancing the traffic based on the current load of network. A scheme can also be classified based on decision maker of balancing technique. Most types of solution are under dynamic category, since dynamic schemes are performing better, so most researchers focus on dynamic and distributed approaches.

Static and dynamic

In static schemes of multipath load-balancing, a flow utilizes the same path for the entire span of the data transfer, which gets decided at its beginning. In such schemes, at the event of congestion in the network later in time, the flow can not change its path. This static nature of flow can create a situation of congestion collapse resulting in further degradation in the performance. Load-balancing does not mean that only initially balance the traffic on some basis, and the path of the flow can not be changed in case of congestion. On the other hand, in dynamic multipath load-balancing schemes, the flows can alter their paths according to the traffic condition in the network. Dynamic schemes are more robust in handling different kinds of traffic. In such approaches, it is possible that a flow shifts its path multiple times, between its beginning and termination as shown in Figure 2.23.

Flow and packet-level

If the flows at application level utilize multiple paths by associating different flows with different paths, it is termed as flow-level load-balancing. In such approaches, all the transmitted packets follow the current path of the flow. On the other hand, a scheme is called packet-level load-balancing, if a flow itself utilizes multiple paths by sending different packets over different paths like NDP [Handley et al., 2017]. In packet-level load-balancing schemes, an output port is

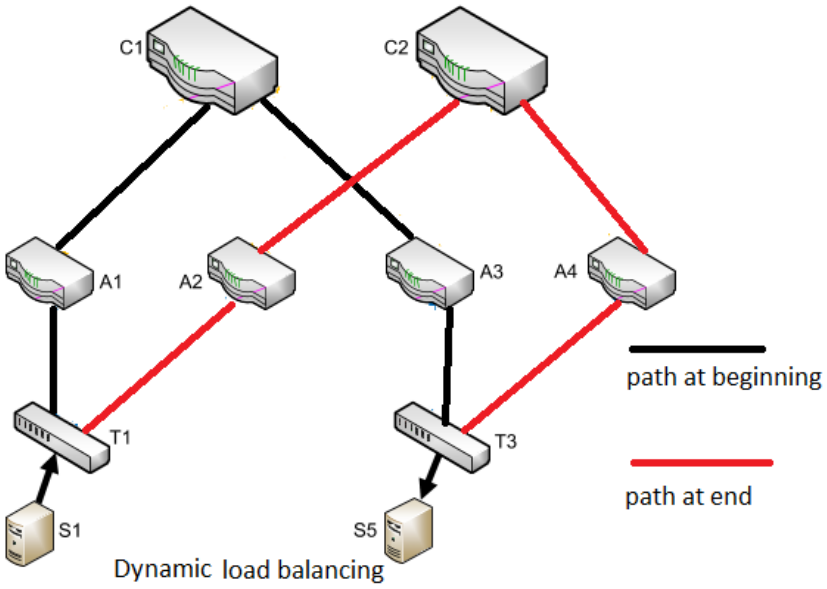


Figure 2.23. : Path change in dynamic load-balancing scheme

independently selected among the set of eligible ports for every packet [Dixit et al., 2013]. A flow faces packet re-ordering problem and degrades the performance in an asymmetric topology in a packet-level load-balancing. While packet re-ordering with flow-level schemes is negligible, all the packets of a flow follow the same path while not allowing the dispersion of packets of a single flow on different paths.

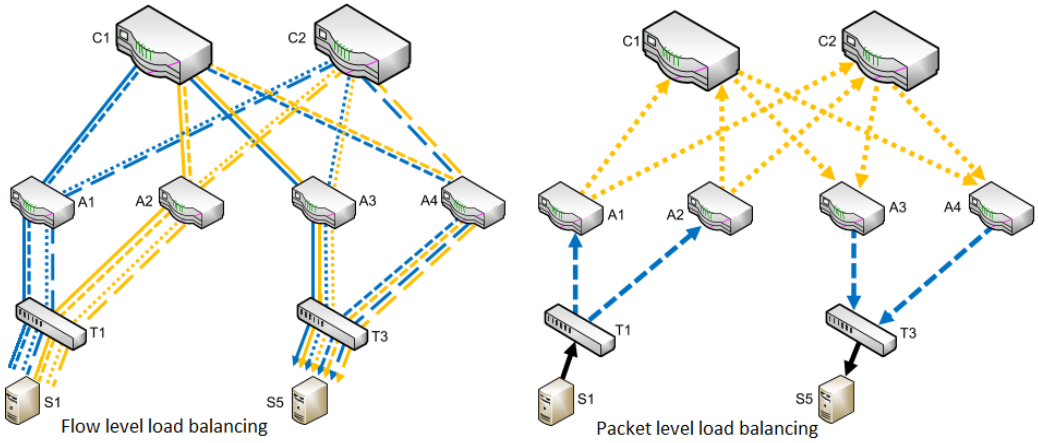


Figure 2.24. : Flow and Packet-level load-balancing

Source based and switch based

A load-balancing scheme is called a source based load-balancing if a source is able to decide or alter the path. On the contrary, if a switch is able to decide or alter the current path without informing the source it is called a switch based load-balancing as shown in Figure 2.25. In a switch-centric topology (as discussed in the previous section), a switch is responsible for the forwarding of a packet. A host will be connected with a switch by a single link which in turn will be connected

with other switches by multiple links. In other words, switches are generally multi-homed with multiple NICs while source being single-homed with only one NIC. For such network topologies, a source should rely on switches for altering the path of the flow. Whenever a source wishes to change the path of the flow, it sends some predefined signal to the switch, which executes the path shift operation. In case of source based load-balancing, the congestion persists for longer duration as compared to switch based load-balancing; since the congestion feedback requires to travel longer distance (hops). However, implementation of switch based load-balancing is more complex compared to source based load-balancing. In source based schemes, on top of Equal Cost MultiPath (ECMP), a source only needs to change few fields of the packet header.

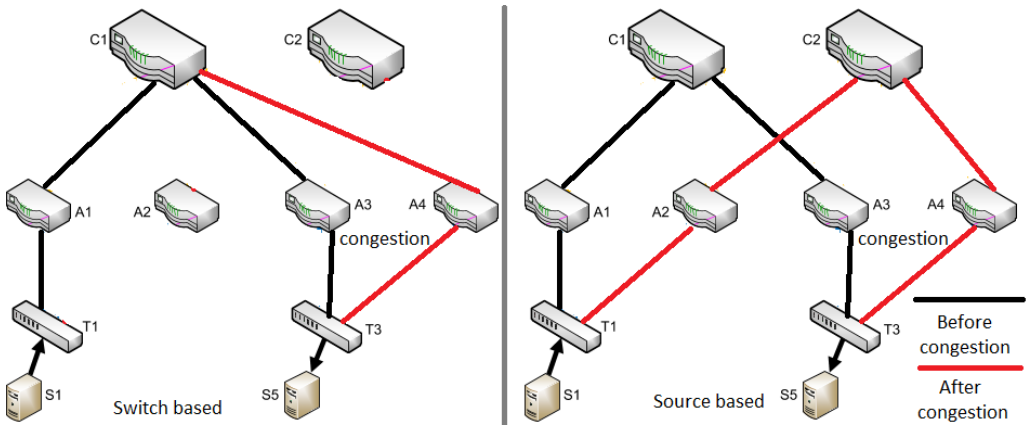


Figure 2.25. : Source and Switch based load-balancing

As shown in Figure 2.25, after the occurrence of congestion at switch A3, the path of the flow is changed by switch C1 in case of switch based load-balancing. While in source based load-balancing, source S1 sends signal along the path triggering the switch T1 (first switch along the path) to change the path of the flow.

Load oblivious and load aware

A multipath load-balancing scheme in which load of the network is not considered at the time of balancing the traffic, is called a load oblivious scheme. On the contrary, a load aware scheme must take the load into account, while balancing the traffic. A load oblivious scheme is inefficient compared to load aware schemes, since it allows the possibility that a new flow gets assigned to a congested path. In a synchronized network, load aware approaches may lead to oscillation and instability, by shifting more than one flow from a highly utilized path to under utilized path simultaneously. Further, a load aware scheme can be classified based on the local and global information. A global load aware traffic balancing scheme like Hermes [Zhang et al., 2017] performs better, compared to local schemes. However overhead associated with global load aware schemes is much higher compared to local schemes.

2.3.2 Proposals in literature

The importance of multipath load-balancing in data center has been discussed earlier. For efficient utilization of network resources of data center, researcher have proposed several proposals for fully exploring the multipath capabilities of the network. In this section, some of the multipath load-balancing proposals for data center network are described.

Hedera

Hedera [Al-fares et al., 2010] implemented on Openflow switches, used a centralized controller for dynamic flow-to-path assignment of elephant flows. This assignment is based on the load at the switch. Hedera assigns a single core switch to deal with all the flows of a destination. The controller calculates a path arrangement and periodically updates the routing table of the switches on the path to alleviate hot-spots. The operation of Hedera is divided into three steps : 1)Detection of elephant flows at the ToR switches. A flow is called an elephant flow if it occupies at least 10% of the link capacity. This threshold of 10% link capacity is necessary to take out small flows from the bother of centralized controller. 2)For demand estimation of flow, a max-min fairness algorithm runs iteratively. Central controller calculates the bandwidth demand of elephant flows. In order to find appropriate path for flows, Herdera uses two placement algorithms namely the Global First Fit (GFF) and Simulated Annealing (SA). 3) With the help of Openflow, paths of flows are installed in the forwarding table of the switches.

Hedera faces scalability problem because of the dynamic traffic pattern, and is less fault-tolerant due to a single point of failure.

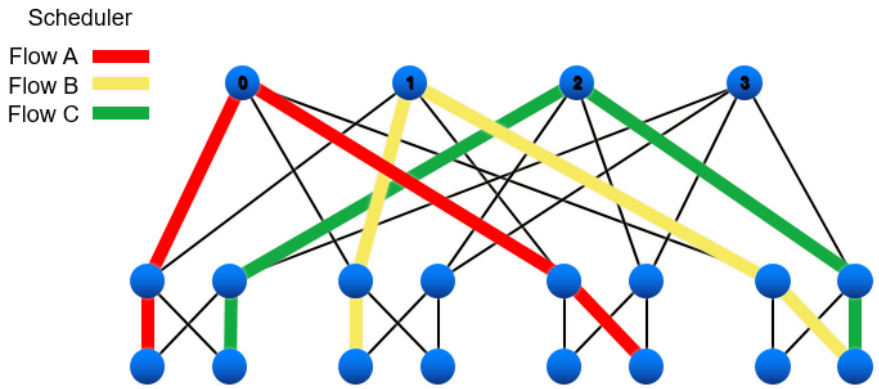
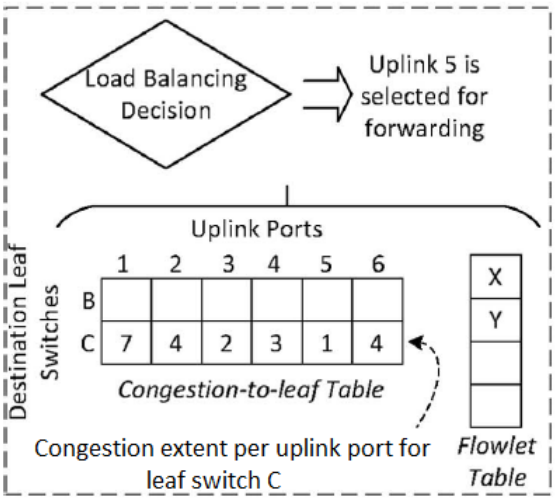


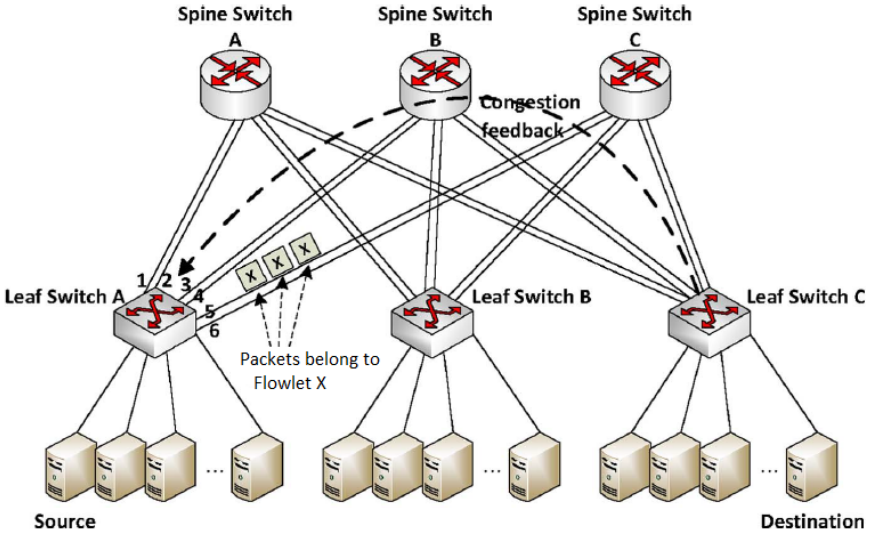
Figure 2.26. : Flow placement in Hedera with GFF

CONGA

CONGA [Alizadeh et al., 2014] is a congestion aware, in-network and distributed load-balancing solution for data center network. It is based on explicit in-band congestion signaling between the source and the destination leaf (ToR) switches. The network switches maintain a table of path level congestion. Using this information, the source leaf (ToR) makes the load-balancing decision with global congestion awareness. CONGA is an in between solution of packet and flow-level load-balancing. The basic unit for traffic balancing in CONGA is flowlet. Flowlet of a flow is defined as a burst of packets separated by a major gap, more recent work related to flowlet can be found in Flowtune[Perry et al., 2017] and LetFlow[Vanini et al., 2017]. CONGA transmits a flowlet to the least congested path. With the use of flowlets, possibility of out of the order arrival of the packets at the receiver is less, compared to any other packet scattering scheme. It also increases the granularity of load-balancing compared to flow-level schemes. CONGA is designed for the simple leafspine topology. This approach cannot be generalized to larger topologies like Fat-Tree, and it also requires specialized networking hardware. The overhead associated with CONGA is too high as leaf switches need to exchange the path level congestion information.



(a) Table of leaf switch A



(b) Forwarding of packets belongs to flowlet X

Figure 2.27. : CONGA load-balancing

DeTail

DeTail [Zats et al., 2012] is a packet-level, cross-layer based, adaptive load-balancing solution. In DeTail stack, multiple layers (from data link to application layer) will cooperate with each other for exchanging information. With the use of flow control at the data link layer, a loss less fabric is built up in DeTail. This flow control at data link layer prevents packet drops and helps in creating loss less fabric. It uses port buffer occupancy of a link to measure the congestion and then applies back-pressure scheme for forwarding packets at the network layer. DeTail applies per-packet adaptive load-balancing between all shortest paths. In DeTail every packet chooses the less congested port among the eligible ports. At the transport layer, to nullify the effect of re-ordering, it disables the fast re-transmission algorithm of TCP. DeTail also utilizes ECN like scheme, which is based on queue occupancy, for conveying information about congestion to the sender. Upon receiving the congestion information the sender decreases the rate of the flow. At the application layer, flows are prioritized based on their latency requirement and assigning high priority to delay sensitive flows over elephant flows. Since DeTail prioritizes flows, the egress and ingress queues at switches have multiple priority queues for every port. Prioritization of ingress and egress queues

in DeTail, helps in the measurement of congestion and reducing the waiting time for high priority packets respectively. DeTail requires modification at the switch hardware and it is also not compatible with original TCP. It also suffers from the tree saturation problem due to its reliance on loss less fabric.

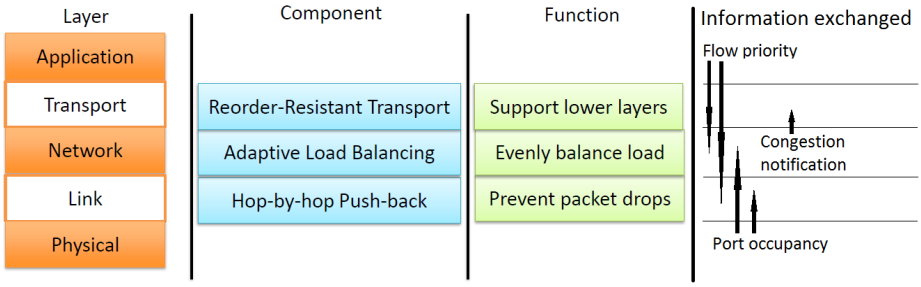


Figure 2.28. : Cross-layer protocol stack of DeTail

ECMP

ECMP (Equal Cost MultiPath) [Hopps, 2000] is a switch based, distributed, de facto solution in the data center. ECMP hashes five tuples of the packet header to assign a path to a flow, allowing the flows between the same hosts to take different equal cost paths. This scheme is load oblivious and does not take care of flow size or load in the network, which may lead to an uneven traffic assignment. Due to its load oblivious nature, it is also possible that multiple elephant flows collide at a switch. Its static nature, allows the possibility that some elephant flows collide and suffer long lasting congestion, resulting in a major performance degradation. ECMP is fully compatible with the current systems and works well with short-lived flows. Majority of proposals for multipath load-balancing use ECMP.

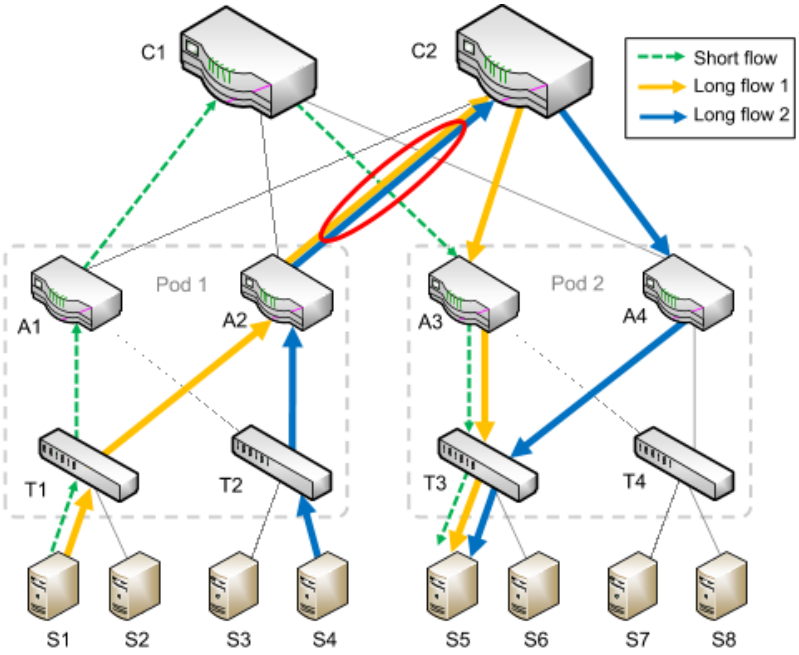


Figure 2.29. : Problem of load oblivious load-balancing in ECMP

MPTCP

MPTCP (MultiPath TCP) [Raiciu et al., 2011] is an extended version of TCP, host (source) based solution to balance traffic by splitting the flow into multiple subflows. In MPTCP, hosts open multiple subflows to the same destination as shown in Figure 2.30. With the help of different port numbers, a subflow is hashed to a different path in an ECMP enabled switch. Each subflow, in MPTCP, behaves like a traditional TCP flow as shown in Figure 2.31. At the time of congestion, MPTCP balances traffic among subflows. In other words the congestion window of the only congested subflow is reduced to eliminate the congestion along its path. Due to flow splitting, MPTCP suffers from packet re-ordering problem. This solution is not appropriate for small size flows due to time required for connection establishment and signaling. Also, it is highly complex at both sender and receiver sides due to the employed congestion control algorithm and packet reassembling logic respectively.

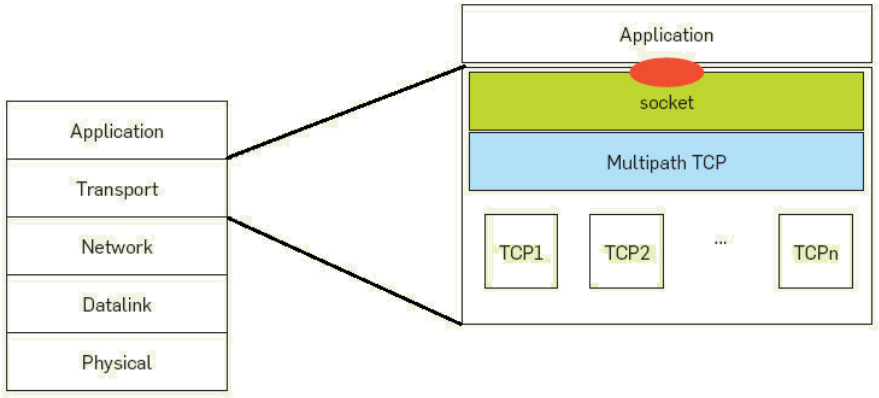


Figure 2.30. : MPTCP architecture

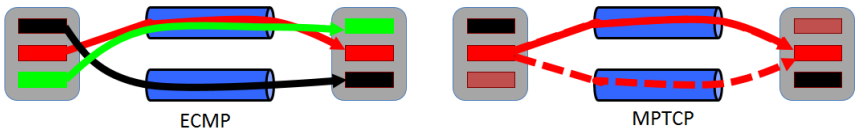


Figure 2.31. : load-balancing in MPTCP

Presto

Presto [He et al., 2015] assumes that the small size and size uniformity of data units is enough for achieving sufficient load-balance in symmetric topologies. So it divides flows into equal size chunks of 64KB data called flowcells by using vSwitches at the host as shown in Figure 2.32. These flowcells are routed in a round-robin fashion among available paths. The hosts are responsible to route the flowcells using source routing. Most of the delay sensitive flows are less than 64KB in size, hence with idea of flowcells, these flows do not face packet re-ordering. For reducing the re-ordering problem at the receiver, Presto uses Generic Receive Offload (GRO), which is responsible for merging the Maximum Transfer Unit (MTU) size packets. The GRO not allow to be merged the re-ordered packet with present segment and create a new segment for re-ordered packet. GRO sends large TCP segment up at the time, a gap in sequence numbers is detected, or Maximum Segment Size (MSS) of TCP is reached, or timeout fired Figure 2.33 shows the aggregation of packets. The GRO reduces the Central Processing Unit (CPU) processing overhead and helps in distinguishing the loss from re-ordering. Presto also utilizes centralized controller to react failures

in topology.

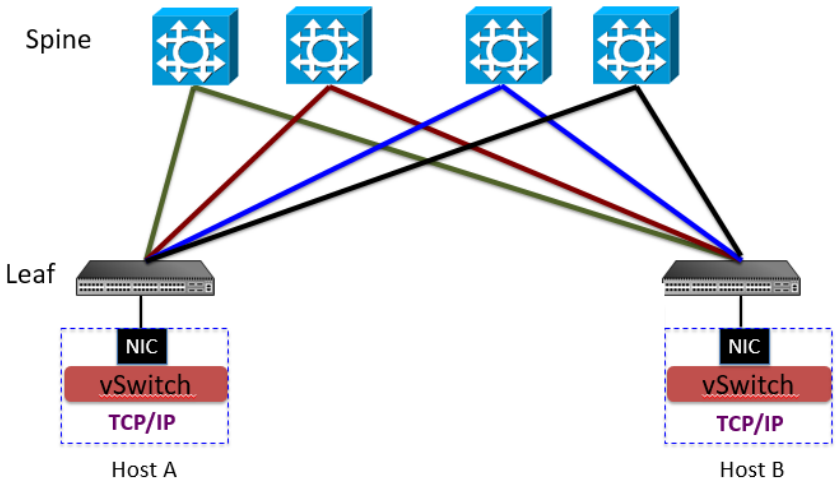


Figure 2.32. : Presto architecture

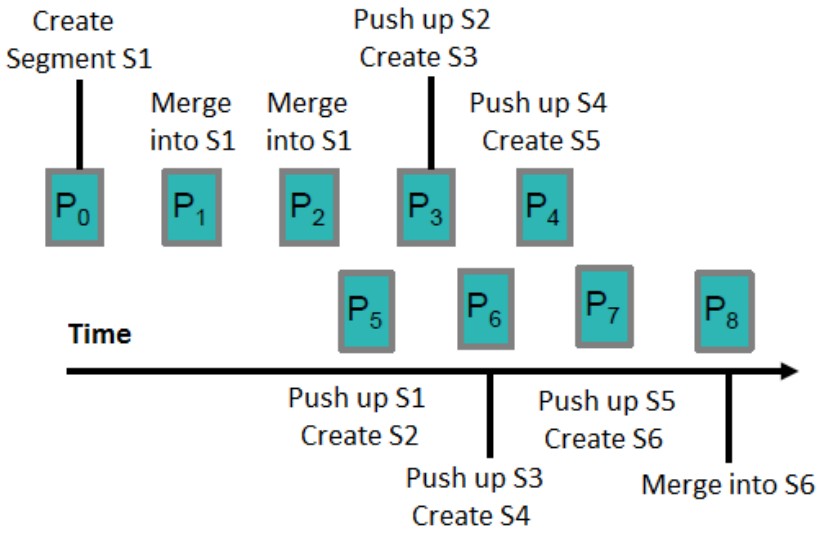


Figure 2.33. : Packet aggregation using GRO [He et al., 2015]

DARD

DARD [Wu and Yang, 2012] is an host (source) based solution in which every host monitors the network bandwidth utilization and then selects a desired path for a flow based on the network condition. It uses selfish path selection, which will converge to the equilibrium in few steps. DARD is a source routing based approach and uses hierarchical addressing to represent an end-to-end path. Hierarchical addressing with source routing is used for adaptive routing in DARD. With the help of tunneling, a source can decide the path of a flow. Due to monitoring of entire network by every host, the amount of overhead is high, which also limits its scalability.

FlowBender

FlowBender [Kabbani et al., 2014] is a dynamic, end host driven, reactive and flow-level solution for balancing the load. FlowBender sends packets of a flow as a single stream until the flow gets congested. This scheme re-routes the flows when congestion is detected in the network. It uses the ECN as a notification of congestion similar to the DCTCP. Whenever queue length of any switch reaches a threshold, it sets the CE bit in the packet header. Based on the ECN-bit, end host initiates re-routing of the flow, by changing a few unused fields in the header. In order to re-route, hash function of Equal Cost MultiPath (ECMP) is modified to use the fields of header for flow-to-path assignment. However, FlowBender re-routes the flows based on only local congestion knowledge of the current path, while being unaware of the knowledge of global congestion. Thus, it is possible that FlowBender shift a flow to a more congested path.

DiFS

DiFS [Cui and Qian, 2014] is a flow-level, dynamic forwarding mechanism which requires modification in the functionality of the switches. DiFS aims to balance the number of elephant flows among different paths. For resolving remote collisions, whenever a switch detects flow-to-link imbalance, it sends a notification message along the reverse path of the flow. Intermediate switches check of moving a flow to another path is possible or not. In case of failure, they would forward this notification in the reverse path of the flow. Since, DiFS maintains a counter for the number of transmitted bytes in each flow, it will increase the overhead of solution.

RPS

RPS (Random Packet Scatter) [Dixit et al., 2013] is a switch based, static and packet-level solution. In this scheme packets are forwarded to their destinations through different shortest paths. For each packet, a path is selected randomly with uniform probability. In other words, a switch randomly selects an output port from a set of eligible ports (alternate equal cost paths) independently for every packet. It does not take the load into account while forwarding a packet. RPS assumes that, all alternative equal cost paths are symmetric and hence have the similar latencies. RPS also suffers from out of order packet delivery at the receiver due to different latencies in alternate paths. Further, these out of order packets may be misinterpreted as a sign of congestion by TCP leading to performance degradation of the scheme. This approach works well with symmetric topology. However when asymmetries arise in the network due to faults and dynamic traffic pattern, this scheme turns less efficient.

Hermes

Hermes [Zhang et al., 2017] is an edge-based solution that balances the load by sensing the path uncertainties and timely reacting to solve the problem. For sensing the path uncertainties, such as congestion, asymmetry, and failures, it makes use of transport-level signals such as ECN and RTT to measure path congestion, while leveraging events such as re-transmissions and timeouts to detect packet blackholes and random packet drops. Hermes employs active probing that can effectively increase the scope of sensing at minimal cost to improve the visibility of network. Hermes assesses flow status and path conditions, and makes deliberate re-routing decisions only if they bring performance gains. This enables Hermes to prune unnecessary re-routing and to reduce packet re-ordering and congestion mismatch, and making it transport-friendly.

Hermes highlights few problems of previous proposals. 1)Flowlets are decided by many factors such as applications and transport protocols. Solutions relying on flowlet switching are inherently passive and cannot always timely react to congestion by splitting traffic when needed. 2)Congestion control algorithms adjust the rate (window) of a flow based on the congestion state of the current path. Hence, each re-routing event can cause a mismatch between the sending rate

and the state of the new path. Therefore within a flow, the congestion states of different paths are mixed together, and congestion on one path may be mistakenly used to adjust the rate on another path.

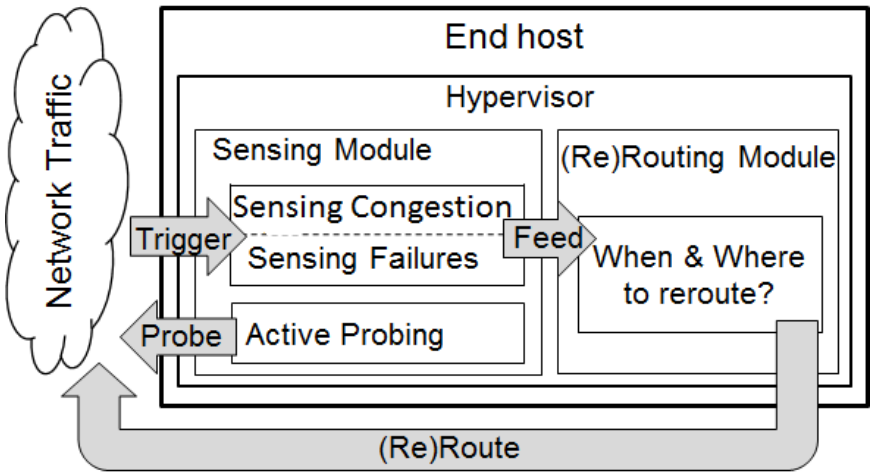


Figure 2.34. : Hermes overview [Zhang et al., 2017]

Figure 2.34 overviews the two main modules of Hermes: (1) The sensing module that is responsible for sensing path conditions; and (2) The re-routing module that is responsible for determining when and where to re-route the traffic.

For path sensing, Hermes leverages both RTT and ECN to detect path conditions. It characterizes a path to be good if both RTT measurement and ECN fraction are low. In contrast, if both RTT measurement and ECN fraction are high, then path is identify as congested. Otherwise, in all the other cases, a path is classified as gray path. Hermes leverages packet re-transmission and timeout events to infer switch failures such as packet blackholes and random drops. Hermes uses packet as the minimum switchable granularity so that it is capable of timely reacting to uncertainties. Hermes tries to be transport-friendly by reducing the frequency of re-routing and it makes re-routing decisions based on both path conditions and flow status.

In case of routing for new flows Hermes first tries to select a good path with the least sending rate in order to prevent local hotspots. If it fails, Hermes further checks gray paths. If it fails again, Hermes randomly chooses an available path with no failure. For congested flows, Hermes first tries to evaluate the benefit of re-routing. Hermes checks the sending rate and size sent, and decides to re-route only when both conditions for cautious re-routing are met. The re-routing procedure in congested case is similar to that in the former case, except that the selected path should be notably better than the current path. The flow stays on its original path if no better path is found.

LetFlow

LetFlow [Vanini et al., 2017] is a recent work, the simple mechanism of LetFlow splits flows into flowlets in the network. This work shows that flowlet switching, an idea first proposed more than a decade ago, is a powerful technique for resilient load balancing with asymmetry. Flowlets have a remarkable elasticity property i.e. their size changes automatically based on traffic conditions on their path. LetFlow simply picks paths at random for flowlets and lets their elasticity naturally balance the traffic on different paths. On slow paths, with low per-flow bandwidth and high latency, flowlets tend to be smaller because there is a greater chance of a flowlet timeout. On fast paths,

on the other hand, flowlets grow larger since flowlet timeouts are less likely to occur than on slow paths. Flowlets can compensate for inaccurate load balancing decisions. Flowlets shifts traffic away from slow (congested) paths to fast (uncongested) paths.

In LetFlow all functionality resides at the switches. The switches pick an outgoing port at random among available choices for each flowlet. The decisions are made on the first packet of each flowlet; subsequent packets follow the same uplink as long as the flowlet remains active. In this proposal switch uses a Flowlet Table to detect flowlets. When a packet arrives, its 5-tuple header is hashed into an index for accessing an entry in the flowlet table. If the entry is active the packet is sent to the port stored in the entry. If the entry is not valid, the load balancer randomly picks a port from the available choices to forward the packet. It then records the chosen port in the table. For examining the validity of each entry a separate checker process is used at switch. This process regularly checks the each entry of table after a fixed amount of time for validity purpose.

Flowtune

Flowtune [Perry et al., 2017] uses a centralized controller, but congestion control decisions are made at the granularity of a flowlet (which refers to a batch of packets backlogged at a sender), with the goal of achieving rapid convergence to a desired rate allocation. Flowtune aims to achieve fast convergence to optimal rates by avoiding packet-level rate fluctuations. In this proposal a centralized allocator computes the optimal rates for a set of active flowlets, and those rates are updated dynamically when flowlets enter or leave the network. In particular, the allocated rates maximize the specified network utility.

Flowtune applies Fastpass-style ideas to the centralized management of flowlets, instead of packets. This allows the centralized arbiter in Flowtune to handle larger networks than Fastpass. Due to flowlets Flowtune is more scalable compare to Fastpass. However Flowtune is applicable to networks of a modest size, until the centralized arbiter becomes a bottleneck. With flowlet control, allocations have to change only when flowlets arrive or leave. The centralized allocator receives flowlet start and end notifications from endpoints. The allocator computes optimal rates and updates endpoint congestion-control parameters as shown in Figure 2.35.

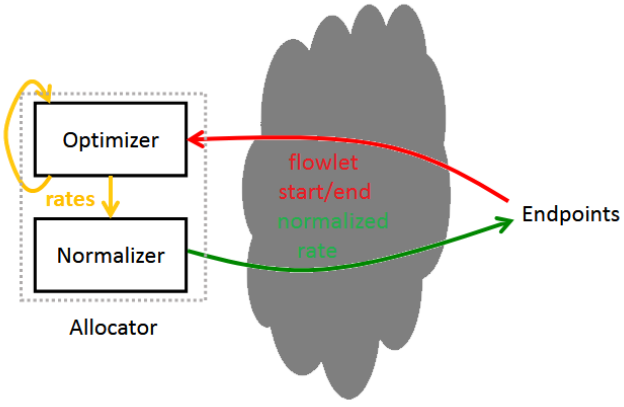


Figure 2.35. : Working of Flowtune

Computing the optimal rates is difficult because even one flowlet arriving or leaving could, in general, cause updates to the rates of many existing flows. Flows that share a bottleneck with the new or ending flow would change their rates. The Flowtune rate allocator chooses flow rates

by introducing link prices. The allocator adjusts link prices based on demand, increasing the price when the demand is high and decreasing it when demand is low.

This concludes the discussion on different multipath load-balancing methods. Table 2.3 summarizes the proposed schemes for load-balancing in a data center.

Proposal	Approach	Granularity	Knowledge	Device
Hedera	Centralize	Flow-level	Global	Controller
DeTail	Distributed	Packet-level	Local	Switch
CONGA	Distributed	Flowlet	Global	Switch
ECMP	Distributed	Flow-level	Oblivious	Switch
MPTCP	Distributed	Subflow	Local	Source
Presto	Centralize	Flowcell	Global	Source
DARD	Distributed	Flow-level	Global	Source
FlowBender	Distributed	Flow-level	Local	Source
DiFS	Distributed	Flow-level	Local	Switch
RPS	Distributed	Packet-level	Oblivious	Switch
Hermes	Distributed	Packet-level	Global	Source
LetFlow	Distributed	Flowlet	Local	Switch
Flowtune	Centralize	Flowlet	Global	Arbiter

Table 2.3. : Summary of multipath load-balancing schemes proposed for data center network

...