1
Introduction

Computation resource has been a human requirement for a long time. Nowadays, computation is one of our essential requirements of daily life, like water and electricity. The demand for computation resource has increased drastically in the last few decades as the world is changing to a digital era. This demand for computation resource will increase in future also. The current computation facilities are not able enough to hold the future requirement. To adopt the future modifications, people always go for over provisioning of some computation resources to minimize the risk. These over provisioned resources are also utilized by people, as human beings tend to consume whatever they have. Due to physical limitations, computation capacity of a resource be under an obligation with some upper bound. Very Large Scale Integration (VLSI) technology is mainly responsible for speeding up the computation. Although VLSI helps to execute millions of instructions per second, at the hardware level infinite speedup is not possible.

Designing a large computation resource by means of many small computers is possible with the help of distributed computing. In distributed computing, a group of servers can execute different parts of a program concurrently. MapReduce, Hadoop and Dryad are well known distributed computing frameworks. This parallel execution of distributed computing is a way to speed up the execution and meet high computation requirements. Speeding up of execution in distributed computing may hinge on network performance. With the benefits provided by distributed computing "how to make a giant computer from many small computers?" is a question. This giant computer is known as data center in the modern world. Data center is a cluster of computers that is owned and operated by a single organization. It can facilitate heavy computational resource for industries. Data center is designed for fast computing through distributed computing in mind.

In today's world, web services are an essential requirement of our daily life. Billions of users are dependent on these services for business and entertainment purposes. With the help of smartphones and laptops, online social networking services like Facebook and Twitter change the style of liaisons among people. Applications like social networking, web search, online gaming are part of our routine life. Such applications employ service oriented architecture. In this architecture, retrieval of a single web page requires coordination and communication with hundreds of individual sub services running on remote nodes. The web services are accessed by millions of users simultaneously. The services are rated based on the satisfaction level of the user. Users expect rich interface, high accuracy, high availability and low latency from web services. Satisfying millions of users simultaneously with high expectation level, is a tedious job. The requirement of computation existed before the web services were available. However the web services impose heavy computation requirement, which cannot be fulfilled by a stand-alone system.

1.1 DATA CENTER

Data center is a bulky computing resource that operates in a controlled environment under centralized management. The data center is administrated by a single organization. With the help of data center, enterprises operate round the clock, apropos to their business need. Physically data center refers to a large cluster of thousands of computers (servers) like a big fish is designed

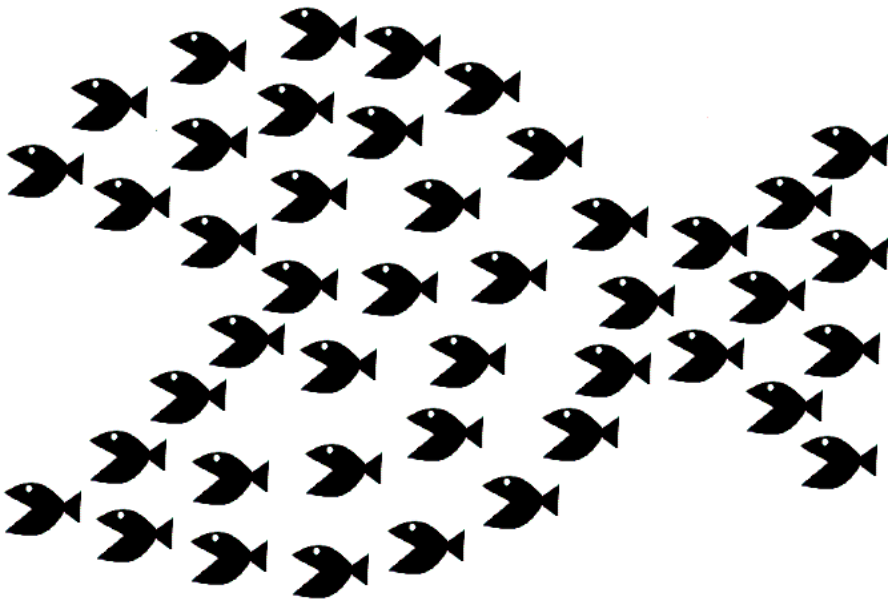


Figure 1.1. : Philosophy of Data center

with many small fish as shown in Figure 1.1. These servers are hosted in the form of racks in data centers. Every rack contains many servers, and all servers in the rack are connected to Top of Rack (ToR) switch. In each row, there are multiple racks hosting the servers. ToR switch of every rack is connected through an aggregation switch. In a data center, there are multiple rows, and these are connected with each other by core switches. The number of racks and rows can also be increased depending on the requirement. Core switches act like a gateway between the data center and the Internet. All incoming and outgoing data must be forwarded through core switch. The main problems in data center are scalability, bisectional bandwidth [1], compatibility and cost. Along with above problems dealing with cabling, power management system, and heat in a data center is a hectic job.

The demand for always-on and fast-responsive online services has led to humongous growth in the data center technology. A data center may own thousands of servers, for these servers providing high bisectional bandwidth at low cost (with commodity switch), enforces designers to go for multiple paths. There are many multipath topologies proposed in literature like Dcell [Guo et al., 2008], Bcube [Guo et al., 2009], Fat-Tree [Al-Fares et al., 2008], leafspine etc. With the help of multiple equal cost paths between host pairs, designers try to even up performance limitations of commercial switches.

Data centers are very important for business purposes in the IT industry. Amazon, Google, Microsoft, Yahoo and Facebook all have their own data centers for different applications such as online gaming, web search, social networking, data mining etc [Vamanan et al., 2012][Wilson et al., 2011][Hong et al., 2012]. These industries increase the size and number of data centers rapidly. Data centers are the source of revenue of these industries. For these industries, data center down time, service degradation or inability to enroll new services lead to loss in business. The topology and protocols used in data center have an effect on the performance of service. The TCP/IP protocol stack used in the Internet is not suitable in data center, as it is optimized for throughput and

[1] The minimum achievable bandwidth between two equal parts of network, for further details see Chapter 2.1

Figure 1.2. : Industries relying on the Data center

fairness. Although, users of data center applications just want to finish flows as fast as possible, they are unconcerned about the throughput or efficient utilization of network. Similarly topological structure of data center decides the bandwidth on which communication among the servers is possible. This bandwidth among servers is very important for performance of any distributed computing application.

Virtualization is used in data center to achieve better server utilization, more flexible resource allocation, reducing hot-spot and limiting power consumption. With the help of hypervisor, multiple Virtual Machine (VM) can operate on one physical server. Although virtualization increases server utilization, due to additional processing at hypervisor layer, it increases the latency in communication. As a result ping between two virtual machines that are running on different servers takes more time as compared to that of ping between these servers.

Nowadays cloud data center offers on demand computational resources for business enterprises in pay-as-you-go manner. Amazon EC2 (Elastic Compute Cloud) charge $0.85/Hr per VM. By utilizing the cloud facility, customers are free from planning, purchasing, operating and maintaining physical hardware and software. Customers dub up only for VMs they hired. In such cloud services, applications of multiple customers are multiplexed and share computing resources with others. Cloud providers do not offer guaranteed network resources to customers. Most of the applications hosted by these customers are distributed which require bandwidth, and hence they are dependent on network. These VMs (hired by customers) communicate over the shared network among VMs of other customers. Therefore the performance of customer's application is not only dependent on the number of VMs hired but also depends on the network bandwidth. Bandwidth achieved by customer's application depends on traffic load and VM placement. The cloud service provider does not provide any control over VM placement to customer. In order to provide a good quality of service to customers, without sharing application and data storage locations, a service provider should achieve a predictable network latency.

Generally there are two high level choices for building the network for large scale computers. One option uses specialized hardware and protocols like InfiniBand and Myrinet. These solutions are scalable and provide sufficient bandwidth. However, these solutions are not compatible with traditional network standards and are also more expensive [Kant, 2009]. For example InfiniBand has its unique cabling /connectors, Physical layer, Link layer, Network layer and Transport layer. This architecture does not support Ethernet.

1.2 CHALLENGES FOR LATENCY CRITICAL APPLICATIONS

For the success of any business, the satisfaction of customers is very important. In cloud computing, the satisfaction of users is centered around availability, data security, cost, performance etc. Applications, such as online gaming require continuous interactions with the users. Millions of users are accessing such type of same application simultaneously. All users expect the response of keystroke or mouse click on screen as soon as possible. Such type of applications impose greater pressure on the service provider for meeting Service Level Agreement (SLA). Applications that require extensive database consultation before responding to user's request are called data intensive applications. Web search is an online data intensive application.

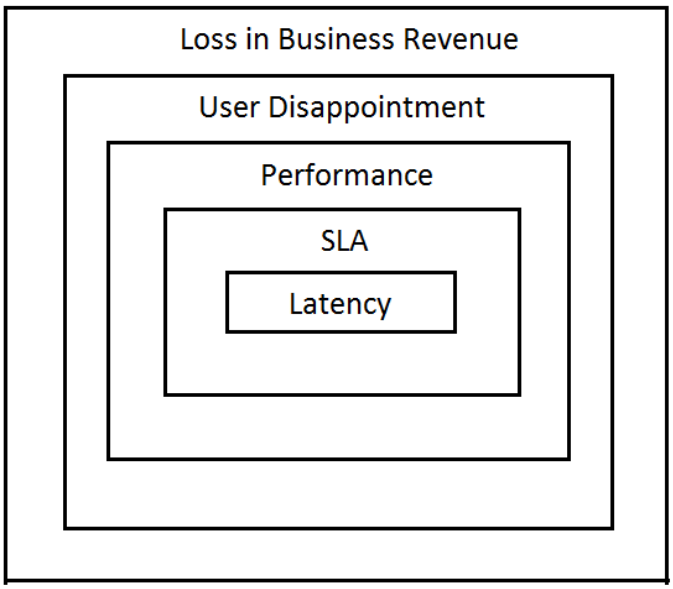


Figure 1.3. : Importance of Latency

Service providers of online data intensive applications are very concerned about low latencies as the low latency communication is essential for overall user experience. A good user experience relies on predictable latency and bandwidth across varying traffic pattern. The small amount of latency at micro level can be seen as liable for losing service level agreement Figure 1.3 which in turn compounded as unacceptable performance at user level. Google observed a 20% traffic reduction from an extra 500ms of latency, and Amazon found that every additional 100ms of latency cost them a 1% loss in business revenue [Alizadeh et al., 2010] [Hoff, 2009]. Interactive applications such as online search are significantly impacted by network activity [Mai et al., 2014]. In Microsoft Bing search engine network contributes 12% of latency [Jalaparti et al., 2013]. Network transfers in Facebook for MapReduce jobs, responsible for on average 33% of the execution time of the jobs [Chowdhury et al., 2011]. This latency imposed by the network can be presumed as snag for service performance.

1.3 MEETING USER INTERACTIVITY DEADLINES

Data centers are designed mainly for heavy computation purposes. Heavy computation task with strict SLA, enforces the requirement of breaking down a large computation into small pieces. In order to meet SLAs in data intensive applications like web search, the whole task is divided among thousands of servers [Alizadeh et al., 2010][Zats et al., 2012]. Aggregation/Partition (horizontal scalability) technique is used to divide a large computation task into many small computations [Wilson et al., 2011][Alizadeh et al., 2010]. In this technique, master server distributes each task

of small chunks to thousand of workers at multiple layers. Each worker after completing its job, sends back a response to the respective master where the master server prepares final response as shown in Figure 1.4. To meet a deadline these workers have to send back responses within a time limit. Any task that does not complete by its deadline is not included in response, thus hurting the quality of response and wasting network bandwidth [Joy and Nayak, 2015]. The maximum response time of any sub task dictates the overall response time. Completion time of the request (demand) is defined as the time taken for the last flow to complete. Even in the presence of thread level parallelism, the communication response overhead imposed by network and protocol stack can ultimately limit the performance of the application.

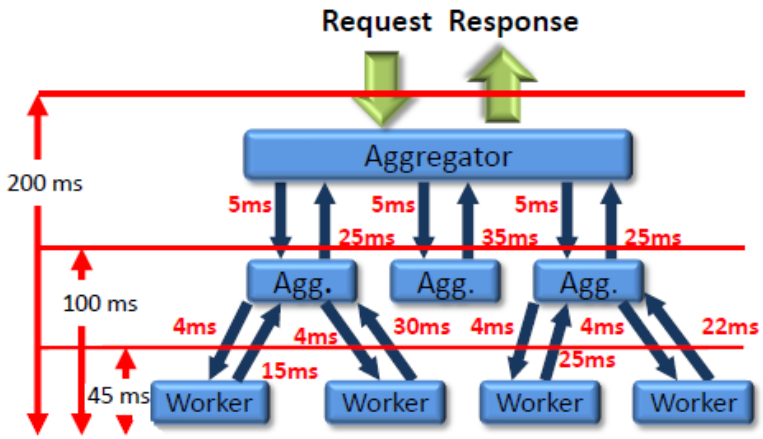


Figure 1.4. : Example of Aggregation/Partition technique [Wilson et al., 2011]

Generally switches are designed to face temporary oversubscription and they require buffering of packets to absorb rate variations and avoid throughput losses. When thousands of workers respond simultaneously to a master, the buffer of the switch connecting workers and master will face overflow problem and several packets are lost as shown in Figure 1.5. In Aggregation/Partition technique, workers are unaware of this incident, and wait for acknowledgment which finally leads to timeout event [Vasudevan et al., 2009]. Waiting till the timeout increases the response time at the master. This problem is known as Incast which arises due to communication pattern, loss recovery mechanism of Transmission Control Protocol (TCP), and buffer space at the switch. Due to this problem in Aggregation/Partition technique a master server excludes those responses that are leading to missing the SLA due to high latency, which leads to further degradation of the quality of the final response.

1.4 TRAFFIC IN DATA CENTER NETWORK

For fully understanding the problem and its importance, it is necessary to know about the differences of traffic in the data center networks and the Internet. As described in literature, most of the Internet traffic either starts or ends in a data center. The amount of traffic on the Internet is less compared to that in data centers. According to the report [Cisco, 2015], the amount of total global traffic over the Internet is projected to reach 2.3 zettabytes per year by 2020. On the other hand the amount of annual global data center traffic was already estimated as 4.7 zettabytes per year in 2015. This is expected that by 2020 annual data center traffic will triple to reach 15.3 zettabytes per year. Further the traffic of data center can be classified as

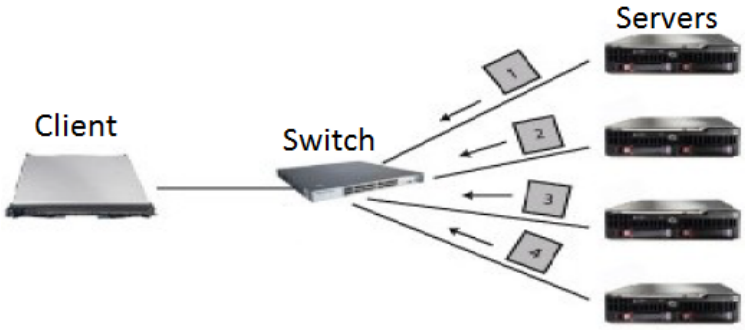


Figure 1.5. : The problem of Incast in Aggregation/Partition

- Traffic that remains within the data center
- Traffic from one data center to another data center
- Traffic between the data center to the end users via the Internet

Generally in literature, researchers use the nomenclature as East-West traffic and North-South traffic to describe data center traffic. All traffic between the data center to the end users is treated as North-South traffic. Whereas the rest of the traffic is treated as East-West traffic. In other words, all traffic within the data center or traffic between one data center to another data center is called East-West traffic. As one can observe in Figure 1.6 that most of the traffic (approx 77%) never leaves the data center. The East-West traffic represents 86% of the total traffic and North-South traffic is only 14% of traffic associated with data centers.

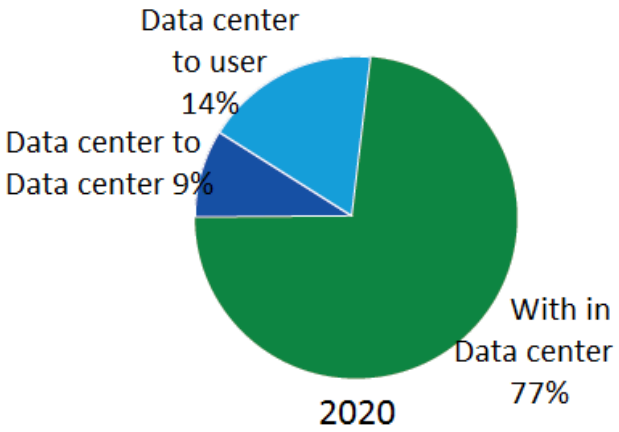


Figure 1.6. : Global Data center traffic in 2020 [Cisco, 2015]

Web services hosted by data center have two tiers namely front-end and back-end. The front-end tier is responsible for the user interface, which is accessed by the users through the Internet. On the other hand, the back-end is responsible for storage and structure of workflow. In data center storage system, data is stored across many servers to improve both reliability and

performance. Most of the traffic that remains within the data center is generated by these back-end tier of a web application. These applications need to dig out many servers and terabytes of data for a single user request.

Traffic within a data center can be classified as deadline sensitive and deadline insensitive. Flows that are used for data replication and database update do not have the requirement of completion in time limit. These flows are bursty and very large in size (elephant flows[2]) compared to the deadline sensitive flows (mice flows). Flows sensitive to the deadline (mice flows) are more in number compared to the deadline insensitive flows (elephant flows). These deadline sensitive flows are responsible for achieving SLA, whereas deadline insensitive flows are responsible for consistency and freshness of database. Traffic within the flow is generally ordered, hence elephant flows can create a set of "hot-spot". Elephant flows will build up queue at the switch and the mice flows will patiently wait behind these elephant flows [Abts and Felderman, 2012] [Alizadeh et al., 2010]. The large buffer size is good for the higher throughput of elephant flows, but it also increases queuing time for mice flows as shown in Figure 1.7. Due to this queuing time, mice flows struggle to achieve SLA. The amalgam of delay sensitive and deadline insensitive traffic on the same network may necessitate QoS mechanisms for performance isolation.

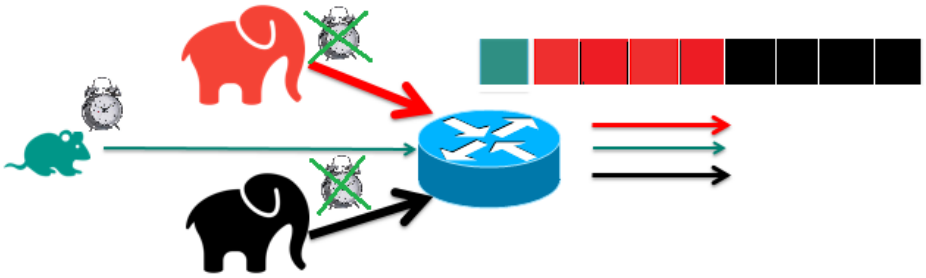


Figure 1.7. : Queuing of mice flows

1.5 ISSUES WITH TRADITIONAL TCP IN DATA CENTER NETWORKING

Traditional TCP used in the Internet is not suitable in a data center. TCP was designed for transferring data from one host to another host in a reliable manner. The congestion control algorithm in TCP is designed for black box network topology. The limitations pertaining to the usage of original TCP in data center are described as follows.

1.5.1 Data Copy

In traditional socket interface, the data is mimed from application buffer to kernel buffer and back from kernel buffer to application buffer at the sender and receiver respectively. This copying of data from one buffer to another can incur significant delay for application. The techniques such as OS bypass or zero copy aim at removing this copying for every message. In data center, latency is very critical and this copying back and forth can be problematic.

1.5.2 Time Scale of RTT and RTO

In Data center, the Round Trip Time (RTT) is in the order of microsecond, whereas the RTT in the Internet is in the order of millisecond. In Traditional TCP, the timeout interval is one or two order of magnitude higher than the RTT. In data center, where the round trip time

[2] A flow is called as a large flow (elephant) or a small flow (mice) depending on the number of packets that it would send.

is in order of microseconds, TCP timeout interval imposes the delay of few milliseconds. Due to this additional latency, performance of delay sensitive applications may suffer. Researchers have suggested that in data center at transport layer, Re-transmission Timeout (RTO) should have at the same scale as network latency [Vasudevan et al., 2009].

1.5.3 Small BDP

In data center due to low RTT, Bandwidth Delay Product (BDP) is small. When multiple flows are contending for its own share in this small BDP, each flow gets a small congestion window. Due to this small congestion window, when a packet is dropped, flow can not recover via fast re-transmission and is stuck until a TCP timeout. For delay sensitive flows this single timeout may be the reason for missing the deadline of the flow. For early detection of packet drop, scale of RTO may be reduced to microsecond level. However reducing RTO at microsecond scale leads to chance of spurious re-transmission.

1.5.4 Designing Goal

Original TCP was designed for optimizing throughput and fairness. If multiple competing flows share a link, they all utilize the bandwidth equally. Fair sharing of original TCP protocol is far from optimal, for satisfying the latency requirements (in turns of minimizing latency) in data center. Protocols designed for high throughput increase the average latency of flow, which impacts on the quality of service.

1.5.5 Delayed Convergence

TCP uses Additive Increase and Multiplicative Decrease (AIMD) to find fair share bandwidth. Finding a fair share of a flow, with the help of AIMD, requires multiple RTTs. These multiple RTTs, increase the duration of the flow [Dukkipati and McKeown, 2006]. It is also possible that a small flow in data center will complete before finding its fair share.

1.5.6 Heavy Congestion Control

As data centers have fewer hops, low packet drop probability and administrated by a single authority, the congestion control mechanism of older TCP is heavy and puts extra processing burden on the transport layer protocol in data center.

1.6 MULTIPATH ROUTING IN DATA CENTER

In today's network, resources like bandwidth and processing power increase continuously. However often these resources are not being fully utilized by the users due to protocol constraints. When the original Internet was designed, hosts had a single interface and only intermediate devices were equipped with several physical interfaces. Nowadays most of the equipment have more than single network interface. For instance, laptops have usually at least both a wired (Ethernet) and a wireless (Wi-Fi) network adapters. Similarly smart phones and tablets have more than one interface such as Wi-Fi and 3G/4GLTE. In data centers each pair of nodes have more than one path. In order to improve the robustness and reduce the latency, a device must utilize all the paths concurrently. If a flow could make use of all the available paths between two end points, there would be performance improvement as each of the paths would carry some amount of data in parallel.

Data centers typically own thousands of servers with multiple paths as shown in Figure 1.8. The data center network designers go for multiple paths in order to provide high bisection bandwidth for the servers at a low cost (with commodity switches). With the help of multiple equal cost paths between host pairs, designers try to overcome performance limitations of commercial switches. Balancing traffic between these multiple paths is very critical to the performance of data center

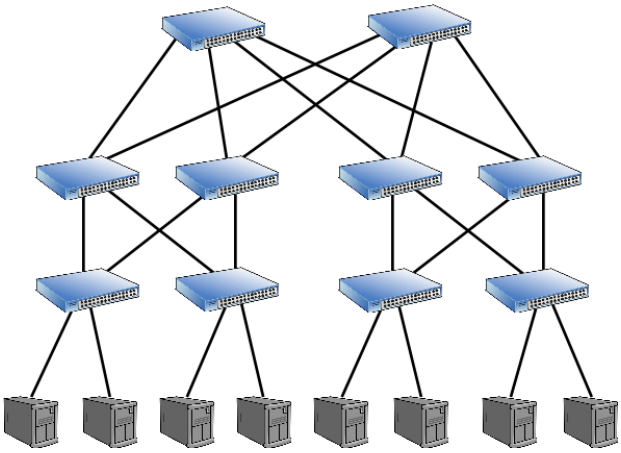
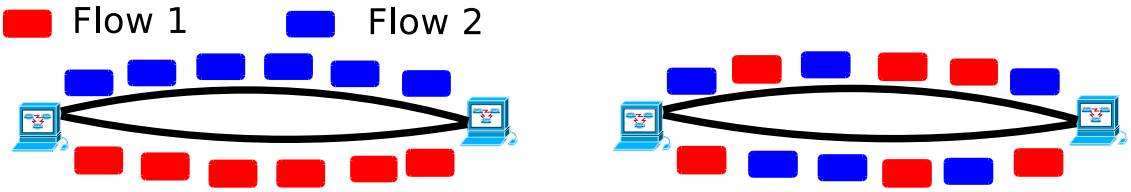


Figure 1.8. : Multipath topology in Data center

[Alizadeh et al., 2014]. If a flow has multiple paths, it is more likely to meet its deadline by following the least congested path. Load-balancing across multiple paths is important for both latency and throughput of application. A naive approach (load oblivious) may lead to creation of congestion on some paths unnecessarily, while other paths will go underutilized. With such a naive approach benefit of multiple paths is scaled down. Even static load oblivious flow-to-path assignment performs worse than a single path due to long lasting congestion [Kabbani et al., 2014]. To overcome this limitation, researchers come up with load sensitive traffic balancing. By means of load sensitive balancing adaptive routing is possible. Through adaptive routing, the path of a flow can be changed whenever it faces congestion. The adaptive load-balancing approach is supplementary to the other congestion control approach.

In the literature, there are several ways of classifying load-balancing solutions. One way to classify the load-balancing schemes, is based on centralized or distributed. In centralized schemes only single host or controller is responsible for finding appropriate path for flows. The controller calculates a path arrangement and periodically updates the routing table of switches. In the networks, with dynamic traffic pattern (small inter-arrival time), centralized solutions face scalability problem. Due to single point of all processing, centralized schemes are less fault-tolerant. In distributed schemes, there is nothing like controller, all intermediate nodes are self sufficient to handle traffic. Hence, in distributed approaches, since all hosts calculate path independently, the solutions are more reliable (more fault-tolerant). Distributed solutions are also scalable and not affected by dynamic pattern of traffic. Coordination between the hosts is a problem in distributed solution, which is not present in centralized solution. However, once coordination among hosts is completed, the distributed schemes perform better compared to the centralized one.

On the other hand load-balancing can be classified based on the path that was taken by the packets. In flow-level (per flow) solutions, all the packets of a flow take the same path [Cui and Qian, 2014]. While in per-packet load-balancing, packets of a flow go through different paths as shown in Figure 1.9. Per-packet solutions use multiple paths simultaneously for each flow as shown in Figure 1.9(b). Such techniques, work well only with a symmetric topology, but they incur high degree of packet re-ordering [Zats et al., 2012]. In case of link failures or asymmetry in the network, these approaches are inefficient. Efficient load-balancing does not necessarily mean that a flow should spread its packets across multiple paths simultaneously and then stitch back together at the



(a) Flow-level Solution (b) Packet-level Solution

Figure 1.9. : An example of flow and packet-level multipath routing

receiver even before starting the congestion. On the other side, in per flow solutions, the flows are evenly distributed over all the available paths. That is all flows are balanced over multiple paths. These schemes balance load less efficiently as compared to the packet-level schemes. However these schemes follow strict packet ordering shown in Figure 1.9(a) and have very little to do with re-ordering problem [Hopps, 2000]. Due to this strict packet ordering, flow-level approaches perform well compared to a single path, and able to cope up with asymmetry in the networks.

In addition to re-routing, deciding next appropriate path for a congested flow is also an essential component of the dynamic multipath routing scheme. Due to re-routing, a congested flow can be re-routed to any other path (highly utilized path) naively, may lead to the multiple re-routing events. An intelligent scheme is required at switches for assigning a new path to the flow. This decision of new path assignment is based on either local information or global information. As described in literature, decision making based on local information is not optimal compared to decision making based on global information. However, collecting global information for all links and switches in the network itself is a complex task; whereas, any switch can collect and maintain local information easily.

1.7 OVERVIEW OF THESIS

As requirements of industry and users for computing facility are growing day by day, the data center technology is an attempt to fulfill the needs of computational demand, presently. In order to provide best services to the customers, the performance of a data center must be predictable. Otherwise, any kind of dissatisfaction at customers end, may lead to a toppling the repute and market of business. The gratification of customer mainly depends on UI (user interface) and UX (user experience). UI of any application refers to its presentation, look and feel. For providing high level presentation (look and feel) many tools are available. UX of any application highly depends on the speed of interactive nature of an application. The small amount of protraction at user end, after submitting the request to the web based application may hamper the UX.

In order to offer good UX to the user, the performance of the data center must be predictable. With predictable performance of data center, a web application designer may infer and reserve required resources. As UX rely on the interactive capability of the service, a predictable network latency is exigency of a web based distributed application. Reducing the network latency not only provides good UX to the user but it is also crucial to increasing the resources utilization of data center, which can further reduce the over all cost of service.

Presently, web based distributed applications adhere to service oriented architecture, in which a single page is a collection of multiple services. In order to respond to a user request in timely fashion, computation related to all the services must be finished on time. These services are distributed over multiple servers and depend on the data transfer by the flows to complete the jobs.

The flows responsible for data transfer of such services are delay sensitive, and must go through networking stack. A networking stack optimized for latency, can assist such flows in transferring the data in a timely manner. Further, an optimized design of network may help to serve a better UX to the customers and business management operations.

This thesis tries to utilize different structures of data center network to reduce the latency of the flows. The topological structures used in the data centers are discussed in Chapter 2. The topologies in data center are well defined, symmetric and under the control of a single authority. This motivates us to take the initiative for utilization of topological structures for reducing the latency. In order to reduce the latency for the delay sensitive flows, a prioritization scheme with topology information is proposed in Chapter 3 of the thesis. An algorithm to utilize the oversubscribed topology structure in data center, with the help of jumbo frames, is also proposed. A flow-level adaptive routing proposal, for well defined topological structure like Fat-Tree, is proposed. This adaptive routing helps the flows to finish quickly even in the presence of faulty links in data center. Further, the same proposal tries to improve the path selection with the help of Analytic Hierarchy Process (AHP) on local information.

1.8 THESIS ROADMAP

The rest of the thesis is organized as follows.

Chapter 2 discusses the related work done in the area of data center. This chapter is divided in three main parts, namely Topologies for Data center Network, Data Transport Protocols and Multipath load-balancing Protocols in data center. Each of the three parts is further divided in two sub parts. The first sub part discusses the classification of approaches and second sub part presents the prominent existing solutions.

Chapter 3 improves the priority scheme of shortest remaining size first scheduling. In this proposal, namely Topology Aware pFabric (TAP), the priority of a flow is computed as a function of both flow size (the amount of data being transferred by flow) and flow distance (in number of hops). In addition to the TAP, Topology aware Preemptive-Shortest Remaining Size First (TP-SRSF) gives higher priority to the newer flows compared to the older ones.

Chapter 4 utilizes the oversubscribed topology structure of data centers. This proposal uses jumbo size frames to minimize the overhead at higher layers of data center topology. The proposed approach requires modification to queue management policy at ToR switches.

Chapter 5 proposes an adaptive routing scheme namely FlowFurl, in which a flow with multiple paths to the destination can be re-routed in case of congestion. The level of comfort for re-routing of a flow is reduced as the number of paths to the destination goes down, and finally re-routing of a flow is not possible when there exists only one path to the destination. This proposal uses connectivity information of topology with congestion information and provides a better way to take re-routing decision.

Chapter 6 finally concludes the thesis by giving a summary of the research work, and proposes future work that can be taken to reduce the latency in data center network.

In addition to these works, in Annexure A, a scheme to minimize the packet re-ordering problem in packet-level load-balancing is presented. In the Annexure MPTCP is used as a multipath packet-level scheme.

...