International Conference on Latest Advancements & Future Trends In Engineering, Science & Management
Sambhram Institute of Technology, Bangalore (India)
03rd - 04th May 2019 www.conferenceworld.in
SAMBHRAM INSTITUTE OF TECHNOLOGY
ISBN : 978-93-87793-86-6

Efficient load balancing over asymmetric data centre topologies

Mrs. Manjula H Nebagiri
Asst professor
Atria Institute of technology

Dr.Latha P H
Head of Department
Sambram Institute of Technology

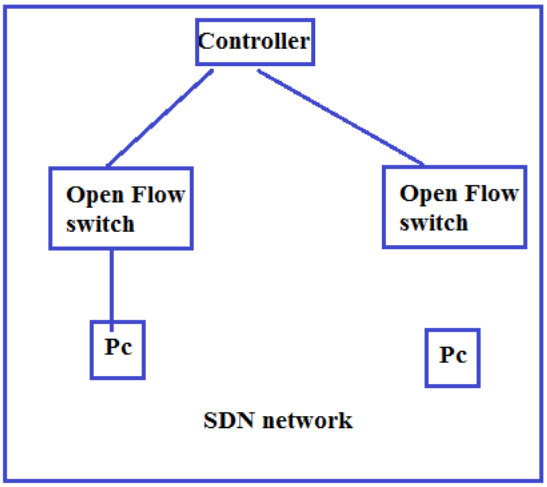
Abstract: *Various approaches have been proposed for data centers to balance the data traffic load recently. Both network topology and the routing approaches can affect the smoothness/quality of the network broadcast. Recently a new system/network architecture called fat-tree becomes one of the trend approach/architecture which is widely used topologies for data centre networks. The routing algorithms such as global load balancing(GLB) and dynamic load balancing* ***(DLB)*** *approaches, are the proposals and both use fat-tree(which are triangular interconnected topologies) topology for the data centers using SDN(Software defined network) approach. Basically GLB and* ***DLB*** *will be having limited of link information storing and path finding between node edges. So this work proposes an efficient framework dynamic cluster-topology with triangular model and also* ***dynamic sub topology load balancing(DCLB)*** *for fat free to schedule the network flows by taking some IP header path information addition dynamically in data centres networks. The DCLB approach will overcome the existing limitations of both GLB and DLB approaches. This approach not only uses less storage information about the link attributes in the controller, but also choose the path with lowest and neutralized parameters dynamically to have the best routed path and will be updated in the current node in the data center's host IP dynamically. Some sub approaches are considered to overcome the load balancing traffic load.*

Keywords: Load balance,Fat-free triangular network,SDN

Introduction:

In the current and recent works with the maximum improvement of the network, the data/information of the internet is more and more complex issue, and the traffic load become more and more higher. The online data centres may give all varieties of services for people, so they play vital roles in the networks nowadays. Some works proposes very important architectures for the internet data centre, and this current architecture might the normal binary tree issue. The issue is that the closer to the root of the binary triangular tree, with the higher the network traffic. used a new approach to solve the problem, which would use more than 2 or even more to broadcast the internet data flows. The way to achieve the load balancing in the data center is main issues. To minimize the network latency and maximize the throughput is a issue n the data centre networks. This work proposes 2 models and objectives the hardware and software approaches. The hardware(simulation) model proposes more network triangular topologies but basically this is cost effective. So the software approach apply load balancing methods in the triangular interconnected topologies for higher bandwidth utilization in the current networks. This approach needs to upgrade the current approaches of network traffic flows scheduling. The load balancing approaches can be divided into two classes , static and dynamic load balancing. In data centre the network flows may change all the time , so the link costs of the network have to be changed and machine should learn and cost should change automatically to learn.. The static load balancing approaches have no ability to get the information on real-time loads and to decrease time of dispatching data flows among nodes. So dynamic load balancing can overcome the issue, they will bring more work for monitoring network statistics and scheduling data flows.

ISBN : 978-93-87793-86-6



Achieving the load balancing at the datacenter level is the key issue now a days. To reduce the network latency and to increase the thoughput is a big problem over the datacenter topology. There were two models[2] proposed , they are hardware and software have been proposed. The model of hardware and and some models of the network topologies, but generally the cost of hardware over the datacenter is over cost than the regular hardware. The software model will apply load balancing models for peak bandwidth usage in current topologies. This infrastrucute requied to modelize the existing models with network data traffic scheduling with respect to flows. The load balancing approaches can be classified into 2 classes[15] ,dynamic and static load balancing. Over the datacenter , the network topologies models of flows might change all the time, so the link costs of the topology changes with accordingly. The static model of load balancing have not much ability to fetch the information over th real-time loads and to decrease time of delivering data broadcast between nodes. But dynamic load balancing models can overcome the issue, they will fetch more working models to monitor the topology statistics with proper scheduling the model of data flows.

This proposed DCLB(Dynamic sub-topology load balancing algorithm, which use the fat-free topology which figure shows a fat-free topology that fig(2) shows a fat-free topology that is a typical k=4 fat-tree network, which has three layers and consists of$(k/2)^2$ core layer switches and kpods, each pod has k number of k-port switches, In one pod, each ToR(To-of-Rack) switch is connected to every aggregation switch and (k/2) hosts. Each aggregation switch connects $(k/2)^2$ to switches on the core layer. This work present a new DCLB approach for that triangular fat-tree. The algorithm can the current link cost and switch status to modify IP headers with shortest routing header with neutralization of connected fat-tree topology. This approach can utilize current connection(link) cost and exchange status to change the Dijkstra approach[10]. This work finally use lossy packets , bandwidth, jitter for evolution the emulation.

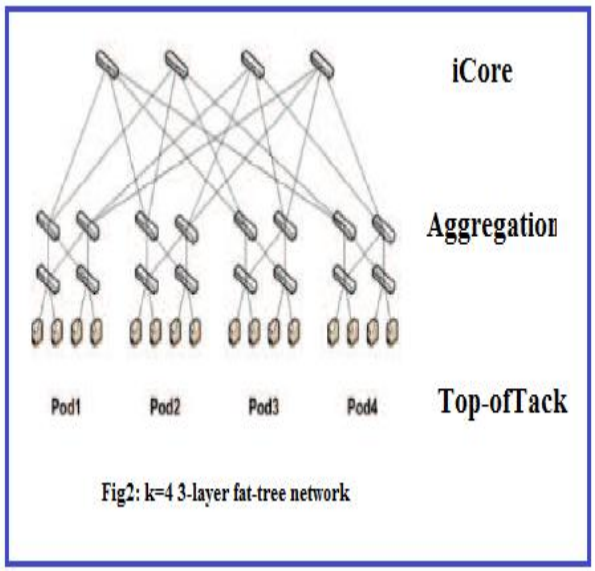


Fig2: k=4 3-layer fat-tree network

SDN(Software defined network) is new network methodology. The key technology of the SDN is open flow. The open flow is a protocol that is used in SDN, and the SDN makes the control plane and data plane separated, Unlike traditional switches in the network, the roles that the open flow switches use to forward data packets re determined by the controller, which is the center component in the open flow network. The controller can easily control the switches by using any programming language to generated the forwarding the roles.

Related Work:

- POX: POX[5] is very ligh weight openflow controller which was implemented in python. It's a frame work for ferocious implementation and prototyp of topology control software using python. And at the bottom level this is widely used infrastrcuture to implement the open flow controller. The infrastrcture for interacting with open flow migrations, it's on the basis for with part of this outgoing work, which gets to make the emerging deciplinary of SDN in this work. This can be utilized for exploring and prototype with SDN debug, topology virtualization, design of controller and type of programming. POX needs java. in this work we use triagulary POX controller to fulfill the algorithms.
- Mininet[6] is a topology for SDN which is having the capability of topology of virtual peers , switches , open flow controllers and links. Mininet hosts run standard operating system sofware and the belonging switches support open flow for huge feasible created routing SDN. Mininet supports R&D , testing , debugging, testing and other any tasks that could give benefit from creating the complete experimental topology over the system. The mininet provides a small and less expensive topology testing bed for implementation of the open flow application, which can leads multiple concurren development environments independantly over the same network(topology). This supports even all levels of regression tests, which can be reused and flexible for packaged. Mininet enables more complex topology testing without the need of the phisical network. It includes the CLI which is topology aware and openflow aware, for application debugging or to run the topology wide tests. It can even supports the custom orbitraty topologies, that includes a basic of bunch of topologies which are parameterized. It provides a straightforward and powerful java API for topology API for network framing and also for experimental purpose. Mininet provides an easy way to achieve the corrent triangular system behaviour and capacity, and to do experiment with networks. Mininet networks run real software code which can include standard OS network applications and rela kernal level topology stack with kernal extensions which might required for compatible topologies. Just because of this, entire mininet will be used to build the fat – free network topology for this work. All the development codes developed with this work and tested with mininet for an open flow controller, a changed switch or a peer, can be migrated to a original system with minimum changes. And these can be used for real – world testing, deployement and evaluation.
- Open vSwithch[7] is a quality wise production and open source implementation of proper virtual distributed multiplayer switch. The main use of open vSwitch is to give a proper switching stack for hardware virtualization frameworks, while supporting more protocols and standards used in a pc network. In this work we use open vSwith for swicthing in the fat-tree network. in this work[8], the authors suggested a solution(naming as DLB) in which only the existing current link was totally ignored. This could lead to a result that the link load may be less, but the end link load might be high. So we suggest and propose a new DCLB approach to improve the situation. The DCLB utilizez the modified Dijkstra approach to produce the shortest paths. The new approach can be better solve the issues which exist in DLB approach. In work[9], the authors proposed a new approach that is GLB(Global Load Balancing). This approach has a huge time complexity, the approach will find the out all connected links between two hops, and calculate all the costs of these network links to get the low load path, and then route path will be generated to the switches by the controller. But when algorithm considered all the link loads, the controller must store more links information. A partial path with in a main path will stored repeatedly. This will cast much more resources and attributes for the controller

to preserve the link information. The DCLB approach can ignore the issue. It will only preserve the topology information rather than the duplicate link information in a route.

System and algorithm design:

DCLB, is the new solution for load balancing for fat free of this work, which contains 3 objectives: the cost of the link module, shortest path module and sub topology discovery module. The approach can update dynamically the link cost of the entire topology. During the search of new routing path, the sub topology discovery module will extract the data/information from the full topology to build the sub topology,and then use the modified repaired Dikstr approch with the sub topology to discover the shortest path with lowes link cost.

1. **The cost of the link module**: In the DCLB approach,the cost of the link module can update dynamicall update the cost of the link of the full topology. The entire topology can be achieved during the open flow topology intiative time. The cost of the link module with the timer for monitoring purpose the entire topology change, and the module rechecks and recalculates the cost of the link for every threshold based time to check and update the loads of the link.

The cost of the link module

```
1: IF  full topology is ready THEN
2:     DO updatecostlink()
3: END IF
4: ELSE
5:     pause
6: updatecostlink():
7: IF  flowstatsreceived THEN
8:     cost = (receivebytes - bytelasttime)/4
9:     update();
10:END IF
```

1 and 5 th lines of this algorithm 1 describes that when the network topology is ready, the sub function ***updatecostlink()*** is called, else function ***updatecostlink()*** which will lead to execute the how to work. when the trigger or event ***flowstartrecieved*** happens, the approach calculates the cost of the link function ***update()*** for updating the costs of the link of the entire network topology.

2. **Shortest path module:** This module utilizes the enhanced Dijkstra approach with the sub topology to produce the shortest path, which will be constructed by the sub topology discovery module.

Shortest path algorithm

```
Input:   sub-topology (G=(V, E)) , switch status w[v]
Output: ShortestPath[switches]
1:while (G)
2:u←NextLink in G
3:for v in G
4:IF d[v] > d[u]+w[v] THEN
        …
5:      v joins ShortestPath
        …
6:      d[v]←d[u]
7:END IF
```

Line 1 to 7 piece of code is partial code in the enhanced Dijkstra approach. The w[v] in lin2 4 indicates the bytes received by all the ways of the switch in threshold time. when searching for the preceeding switch in the path, this approach not only consider the cost of the link but also considers the status of all the switches. So this enhancement will be more appropriately suitable for the DCLB algorithm.

3. **The Sub topology discovery module**: The sub topology discovery module is most innovative and important component/objective in the DCLB algorithm[9] solution only finds the all the paths between 2 peers, but the methodology having a demerit when the scale of the network topology was growing larger and data storage structures for preserving all the feasible paths were growing bigger at the same give time. In this work , we propose a new approach to ignore this issue. The sub topology discovery module is used to preserve sub

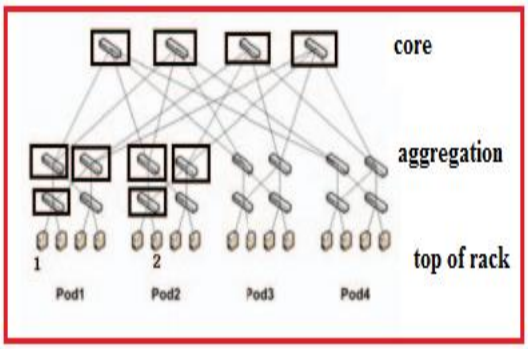
ISBN : 978-93-87793-86-6

topology of all the swithes and closely related links, which are established paths between two peers. By the time of network start up, the module stores the full topology at the beginning and then it discovers the which layer each switch and where it belongs to. After the discovery has been done, the sub topology will work when discovering the shortest path between two peers. The approach will be a working model from 2 ends, source and destination, to discover for the upper layer switch to the bottom switch. The sub topology module preserve the switches discovers during thediscovery process until 2 ends reaches to the same layer.

The sub topology discovery module

```
Input : FullTopologyInfo, src , dst
Output : SubTopologyInfo
1:SourceMACSwitch joins src
2:DestinationMACSwitch joins dst
3:SRC ←[]
4:DST ←[]
5:for switchx in src
6:    for switchy in topo
7:        IF switchxLayer > switchyLayer THEN
8:            switchy joins SubTopology
9:            switchy joins SRC
10:       END IF
11:for switchx in dst
12:    for switchy in topo
13:       IF switchxLayer > switchyLayer THEN
14:           switchy joins SubTopology
15:           switchy joins DST
16:       END IF
```

Line 1 to 16 of this algorithm describes the whole work and execution flow. If the layer of the existing current switch from the source and layer of the current switch from the destination is same, the approach will end up. Else both source and destination will discover the same direction according the fat tree topology until they discover the current same layer where the switches which both sides are the same and approach ends up. The following figure shows the sub



From the above figure the***host1*** will broadcast the information to***host2***, and routing paths that the flow probably network traffic are the frames with switches. At the end these switches will frame the sub topology from the full topology.

Open Flow Network Implemenation:

To implement the network(load balancing) there some important modules(important) should be taken to considered. How to design the system and other is how to discover the shortest path with less link cost to dispatch the flow over the system.

System Design: In our execution flow, when system starts up, the approach will first discovers the full topology of the network and then preserves all the topology information from the switches. When the network topology discovered the framework can do the emulation of this fat free network topology. Frame work can use ***host1 ping host2*** instruction to discover the path with low cost. The entire process of path discovery is that when discovering it first sends the ARP packet to discover the MAC of the destination host and then approach will utilize both ends(source and destination) in the fat tree network. When both ends reaches to the same layer and the switches at the both ends gets the same discovery over the sub topology constructed. Then algogirthm 2 will utilize the sub topology to discover the path from source and destination with low cast. And the framework distributes the path to all the relevant switches.

The central repository is used a framework based on the timer to monitor all the ports of the switches

over the network. The timer will use the system time cycle that is T, and to set the time to threshold time(4 seconds). There are three important structures for data preserve name as full topology data structure. This will preserve the entire information of the network topology. And switch layer structure preserves the full topology of the entire network. This one preserves which layer each switch is belonging to in the topology. When mininet initiates the full topology network also will be initiated. The layer of switch belongs to in the topology. System design just consider the switches which direcly interacts and connects host nodes to be as first layer switches, and the switches connects first layer or second and so on. The switch layer structure will be utilized during sub topology discovery process. The sub topology module will be started while discovering the short path between hosts. with enhanced Dijkstra approach. So the discovered result will be preserved in the sub topology structure.

OpenFlow flow dispatch:

We use the POX controller to be our open flow topology controller. The open flow switches utilizes the flow tables to compare the every flow. The comparing strategies or attributes can be IP of source and destination strategies, any combination of these attributes and strategies. In this framework , we use source MAC address and destination MAC address to compare the flow. When this flow reaches and OpenFlow port of switch , the matching and relevant function will functions. If the switch discovers a match in this process, it will send this flow to relevant port. Else the OpenFlow switch will packs this network packet in a Packet_in message and broadcast to POX controller. After that controller will unpack this data to get the source MAC ip address and destination Mac address. The controller will use all the algorithms to disover the shortest path with low cost of link. Now the controller sends the OFP_FLOW_MOD messages to relevant switches, which based on discovered shortest path.

Emulation Framework in load balancing:

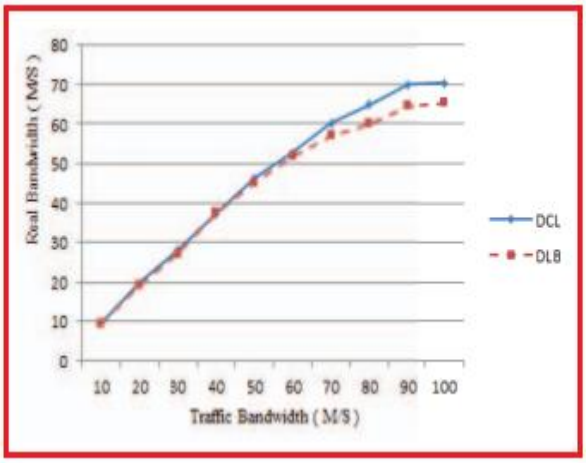
Environment of Emulation: The mininet is running on a PC server with 4-core CPU and it is emulation platfor which can be used to construct the self created topology. The POX is running on a PC with 2-core CPU, and it is on java.

Design of datacenter: This emulation network of data center made of 60 switches and 120 hosts. This network topology is defined by java in mininet. The most difficult issue we face is to generate the real network flow of traffic in the virtual network. works[11][12] proposed new algorithms which could be used to solve the issue in the virtual network.

Comparision of algorithms: in the emulation , this work compares the DCLB approach with DLB approach, the reason is they have similarities in may aspects. The main difference is to choose path strategy. The DLB approach will consider the cost of existing current link and ignores the cost of the link near to destination. but DCLB will consider full link cost with lowest cost it uses the enhanced Dijkstra approach to discover the link with low cost. In emulation this work will evaluate and compare the two approaches by real bandwidths and loss of packet with jitter.

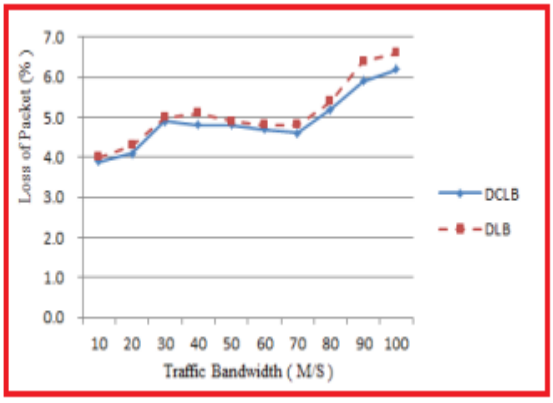
Result of analsys:

Result of bandwidth: The bandwidth can be achieved by the mininet server. We utilize the traffic load from 120M to 170M to achieve the real broadcast bandwidth of the DLB and DCLB approach.



Loss of Packet: Loss of packet of UDP flow from client and server . The drastic loss of packet after 70M traffic load although the rates of the packet loss of the DCLB and DLB increases due to increased rates of the packets of broadcasting and

changes of congetion.The loss rates of both becomes much higher. But DCLB's is always lower than the DLB's.



Conclusion and Future work:

In this work we propose a dynamic sub topology load balancing approach called DCLB for schedule OpenFlow flow for the fat free. The DCLB approach can discover for a shortest path by using the enhanced Dijkshtra approach. We implement all three algorithms on the POX controller. We utilize mininet network emulator to evaluate the dynamically created subtopology. By comparing the both , we get the results, which will exhibit the DCLB approach has the higher realistic transmission rate when the traffic load increases, and the DCLB approach has better jitter performance than the DLB approach.

But the solution is not realistic so far. The most importange and immediate enhancemnet of it wll be that should be considered the other alternative load balancing paths with low cost of link while discovering for the shortest path, because we just considered discovering in the bottom up way(direction) in the sub topology which is constructed from the full fat tree network topology

REFERENCES

[1] Leiserson C E. Fat-trees: universal supercomputing[J]. Computers, IEEE 892-901.
[2] Al-Fares M, Loukissas A, Vahdat A. A network architecture[C]//ACM SIGCO Review. ACM, 2008, 38(4): 63-74.
[3] Xia W, Wen Y, Foh C H, et al. Networking[J]. 2014.
[4] OpenFlow http://archive.OpenFlow.org
[5] POX OpenFlow controller http://www.noxrepo.org/pox/about-po
[6] Mininet http://mininet.org/
[7] Open vSwitch http://openvswitch.org
[8] Li Y, Pan D. OpenFlow based load balancing for Fat-Tree networks with multipath support [C]. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary. 2013: 1-5.
[9] OpenFlow-Based Global Load Balancing In Fat-Tree[J].Advanced Materials Research, 2014, Vol.2784 (827), 4794-4798.
[10] Jehn-Ruey Jiang, Hsin-Wen Huan. Extending Dijkstra's shortest path algorithm for software defined networking[C]. Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific.
[11] Benson T, Anand A, Akella A, et al. Understanding data center traffic characteristics[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(1): 92-99.
[12] Kandula S, Sengupta S, Greenberg A, et al. The nature of data center traffic: measurements & analysis[C]//Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009: 202-208.
[13] He C, Yeung K L, Jamin S. Packet-based load-balancing in fat-tree based data center networks[C]//Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014: 4011-40
[14] Chueh H S, Lien C M, Chang C S, et al. Load-balancingd Birkhoff-von Neumann switches and fat-tree networks[C]//High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on. IEEE, 2013: 142-147.
[15] Tong R, Zhu X. A load balancing strategy based on the combination of static and dynamic[C]//Database Technology and Applications (DBTA), 2010 2nd International Workshop on. IEEE, 2010: 1-4.