An Optimized Server Load Balancing for Unstructured Networks Using SDN

1Shilpa Shaji, 2Arvind Shivaprakash and 3Nima S Nair

1Department of Computer Science & IT, School of Arts and Sciences, Amrita University, Kochi. kichushilpa@gmail.com

2Department of Computer Science & IT, School of Arts and Sciences, Amrita University, Kochi. asp.rondaime@gmail.com

3Department of Computer Science & IT, School of Arts and Sciences, Amrita University, Kochi. nimasnair@gmail.com

Abstract

In today’s contemporary world, digitization has become a fundamental necessity in our lives. It has led to high network traffic, thereby making the overall network management extremely complex. Software Defined Networks is a new standard that satisfies the onerous needs of the network. Load balancing is used to reduce the complexity between networks, by partitioning oodles of task into multiple systems. This paper suggests a Dynamic Control for Load Balancing in Servers using threshold concept, by introducing three new components called Load Conciliator, Course Conciliator and Maxima (Threshold value). The proposed system introduces threshold value which is integrated in the server which will continuously monitor the server load and when the load exceeds the threshold value an alert message is generated which contains the server status of the overloaded server and this message is sent to the controller. This approach gives better response time and throughput compared to other types of load balancing commonly used in SDN.

Key Words: SDN, Load Conciliator, Course Conciliator, Load balancing, Threshold value.


1. Introduction

Network services can be managed through lower level functionality using software defined networks. It’s commonly associated with OpenFlow Protocol. SDN focuses on the fact that traditional network architecture doesn't support the needs of more modern computing environments. The Architecture of SDN is dynamic, manageable, cost-effective, adaptable, directly programmable, agile, programmatically configured. SDN decouples network control and forwarding functions. SDN possesses a centralized controller with a network-wide view that controls multiple packet forwarding switches that can be configured via an interface [1]. Interfaces that support SDN are OpenFlow or ForCES. In OpenFlow Protocol packet sequences are routed using TCAM tables. TCAM or Ternary CAM table is an advanced form of binary CAM table i.e. Binary CAM uses data words composed of 1s and 0s. TCAM allows a third state for one or more bits in the stored data word. So TCAM has more flexibility [2]. The additional state in TCAM is implemented by adding a Mask bit to every memory cell. When a packet arrives at a switch, the flow table is referred. In the case when no reference is found in the flow table, a request is forwarded to the controller for further processing.

The below figure illustrates the basic SDN components. The initial view of the architecture comprised of infrastructure, control and application layers [3]. The infrastructure layer constitutes network elements, which exhibits their capabilities towards control layer or controller plane via interfaces in the D-CPI (Data Controller Plane Interface). SDN applications exist in the application plane and communicate their network requirements through the controller plane via the A-CPI (Application-controller plane interface). In the central part of the architecture the SDN controller translates the application level requirements and exerts low level control over the network elements [4]. An SDN controller coordinates competing application demands for limited network resources.

![Figure 1.1](image-url)

Figure 1.1
Load balancing refers to partitioning the oodles of tasks that a computer performs while interacting with other systems, so that more tasks get executed in the same amount of time so that all users get served faster. Load balancing focuses on optimizing use of resources, maximize throughput, minimize response time and avoid overloading of any single resource. Actualization of load balancing can be achieved using hardware, software or a combination of both. Load balancing differs from channel bonding as it divides traffic between network interfaces on a network socket, while channel bonding is an arrangement in which multiple network interfaces are combined for increased throughput and transmission speed [5]. Load balancing has two schemes, static load balancing and dynamic load balancing. In static load balancing the node performance is determined during the execution time, where as in Dynamic load balancing it makes changes to the distribution of work among workstations at run-time.

In our proposed work we introduced the concept of threshold value, which is calculated dynamically by the servers and when it reaches the limit, it automatically invokes the course conciliator. It will then collects the load information and later used for calculating the load status table by the controller. The proposed system is proved to be efficient for unstructured networks like campus networks or small/medium scale enterprise networks, which do not have a welldefined custom build networks and servers.

The rest of the paper is arranged as follows. The section 2 describes different ways and related work on the area of load balancing in normal IP networks and in SDN. Section 3 conferred the problems in the existing load balancing methods in SDN and proposed a solution to the problem by introducing the concept of threshold value. Section 4 deals with the implementation and result analysis. Section 5 concludes the paper.

2. Related Work

Several research works are progressed in the area of server load balancing in SDN. Extensive usage of internet leads to immense network traffic which is very difficult to handle without proper load balancing techniques. OpenFlow is commonly used protocol in SDN networks that is used to interact with the controller and all its network nodes. Round Robin can be easily implemented in topographically scattered web servers [6]. Load balancing can be obtained by installing specialized software, gateway or load balancers. Due to continuous monitoring of servers the processes are frequently changing in dynamic load balancing. Different aspects of real time server performance analysis can be used to accurately distribute the connections [7].

In Mediator based Dynamic Server Load Balancing approach [8] introduces two mediators called Load Mediator and Channel Mediator. The two mediators are programmed as two modules. One module is installed inside the controller and the other module is coded in the servers of the server farm. The load
mediator maintains server status in Load Status Table inside the controller. The main operation of the mediator is to calculate each server status using the information provided by the channel mediator. The channel mediator determines each server status. When the controller invokes a request to the server farm, the channel mediator code in the server will posts the load status to the controller. The fundamental function of the load balancer is to distribute traffic among a cluster of servers such that a single server does not become over-utilized and ensure that critical services executes correctly. The new approach reduces cost, deployment time and increases flexibility in configuration.

“Dynamic load balancing using SDN” shows network topology, were load balancing can be implemented using POX controller [9]. Here they use Mininet to create a virtual network and it has six host nodes, were two are Http server nodes and the other four are Http client nodes. Depending on the arrival of traffic from client to server, the server is rescheduled. The round robin policy employs a circular queue to determine where to send a request [10]. In another case load balancing is implemented according to the PyResonance controller. PyResonance is Resonance implemented with Pyretic. Resonance is a SDN control platform that advocates event driven control. In an experimenting environment Mininet is used to animate the infrastructure of the network. It presents a method for load balancing based in SDN it achieves every part of load balancing module.

In round robin load balancing method is used incoming packets (requests) are allocated across available switches in the network [11]. We can select this load balancer if all the switches present in the cluster have the same potentiality and handle equal amount of load. By considering this scenario round robin load balancing method is simple and most effective method for load distribution. Here dynamic load balancer is a module integrated with the floodlight controller to appoint different custom cost on each link dynamically [12]. By using this module in the floodlight controller new custom costs are set on links on both directions. The dynamic load balancer is an algorithm used for optimum load management through the best calculated path to reduce the collision and information loss, when the load on the link is greater than the bandwidth of the link. This means that it can handle more packets and has greater efficiency.

3. Proposed System

In the existing approach which uses Mediator Based Dynamic Load Balancing in SDN introduced new components that will monitor the server load regularly. These components are basically programmed modules. One of the components namely Load Mediator’s main role is to update the load status table periodically according to the current server status. The Channel Mediator is a two part module, one of which is installed in the controller and other one in each server in the server farm. Load mediator which resides in the controller updates the server load status in the load status table. The Channel Mediator
will gather information from the servers in every 120 seconds. When the channel mediator functions, code in the controller invokes the internal channel mediator code in the server, it will gather information from each server and sends them back to the controller, load mediator residing in the controller then will receives the information and updates the Load status table and also the Flow table entries. This cycle is repeated continuously. One of the major demerits of this approach is that after a single load status update there is a time interval of 120 seconds for the next status update and little more time to update the flow table. This creates a major flaw in the system, since there is a minuscule chance that the server can overload in that time interval. In this scenario the channel mediator as well as the controller will be unaware that the server is getting overloaded until the next status update. This causes major issues like the server overload or system crash which will decelerate the entire network.

In this paper, we proposed an enhanced and optimized Dynamic Control for Load Balancing in Servers using SDN in an unstructured network. We introduced three new components called Load Conciliator, Course Conciliator and Maxima (threshold value set in the servers). Since load balancing in the servers requires searching of under loaded servers, the two new components fulfil this purpose accurately without putting excessive burden on the network. The two new components are programmed as two functional modules. The Load Conciliator is installed inside the controller whereas Course Conciliator is installed in the servers. “Maxima” is a threshold value used to monitor and accurately gauge the server load.

Figure 3.1
A. Load Conciliator (LC)

It maintains a table that contains the load information of servers in the network. The table is updated according to the information provided by the course conciliator. The Load Conciliator calculates the least loaded server and invokes the flow table pusher to update the flow table so that the request to the server is routed to the least loaded server.

B. Course Conciliator (CC)

Course conciliator’s functional code is a two part code where one is installed in the controller and other in servers in the network. This will collect server load details like memory usage, CPU utilization, storage details etc. This information is then passed to the load conciliator for updating the Load status table.

C. Maxima (threshold value)

Maxima - a threshold value that is set in the servers to monitor all the data flow and load in each server. When the server gets overloaded an alert message is generated. That contains current status of the overloaded server with a caution message and this message is sent to the controller through course conciliator.

The Threshold value acts as a Brimming point to monitor the load overflow of the servers in the SDN networks. The threshold value (maxima) is set in the servers to constantly monitor the server load and when it exceeds the threshold value, an alert message is generated which contains the overall details, load information and the current status of the server. This alert message is sent to the controller via the course conciliator instantaneously. When the message is received in the controller, the load conciliator installed in the controller will grab the server information from the message and updates the load status table instantly. On the basis of load status table the controller will reroute incoming packets until the overloaded server returns to normal state. This approach can prove to be a better alternate since there is no time interval involved, therefore the system can be monitored constantly and reduces the risk of server or network crash.

4. Implementation

We used Mininet a tool used for creating the virtual networks and supports the SDN and OpenFlow. The load balancer in our system is implemented as three functional modules and is integrated with the POX controller. The servers in our unstructured network are installed with a part of course conciliator, which consider RAM Utilization (Lr), CPU Utilization (Lc), and Throughput (Ts) of the servers for calculating the load and the threshold value.
Algorithm for Course Conciliator
1. Calculates \( L_r = \frac{\text{RAM used by the Server}}{\text{Total Memory available}} \)
2. \( L_c = \frac{\text{Total instructions executed per unit time}}{\text{The actual capacity of the Server CPU}} \)
3. Throughput (\( T_s \)) = No of request handled per unit time
4. Find \( L_{\text{avg}} = \frac{L_r + L_c + T_s}{3} \)
5. Calculates the threshold value \( \text{Maxima} = L_{\text{avg}} + \Delta T \) (\( \Delta T \) - load margin)
6. If \( \text{Maxima} >= 65\% \) Construct a Load status packet \( \text{Lpkt}\{\text{ServerID, Lr, Lc, Ts}\} \)
7. \( \text{Lpkt} \rightarrow \text{LC module in the controller} \)

Thus the overall Load balancing process is invoked by the CC module. On receiving the Lpkt the Load Conciliator, calculates the load and fitness value. It also maintains a Load Status Table. Refer Table 1.

Algorithm for Load Conciliator
1. Determines Resources Available (Memory, CPU, Storage and Throughput)
2. Populates the Load Status Table.
3. Calculates the Fitness Value \( F = \frac{\text{Available}}{\text{total}} * 100 \)
4. If \( F >= 30\% \) Set Status= Normal else Set Status="Overloaded"
5. Call FlowPusher()

FlowPusher()
1. Fetch Sid from Load Status Table with status “Normal”
2. Push a Flow Rule to the Flow Table to route any request for the server to Sid Table 1 – Load Status Table

<table>
<thead>
<tr>
<th>Sid</th>
<th>RAM Utilization</th>
<th>CPU Utilization</th>
<th>Fitness</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>R1</td>
<td>C1</td>
<td>F1</td>
<td>Normal</td>
</tr>
<tr>
<td>S2</td>
<td>R2</td>
<td>C2</td>
<td>F2</td>
<td>OverLoaded</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>Normal</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Sn</td>
<td>Rn</td>
<td>Cn</td>
<td>Fn</td>
<td>Overloaded</td>
</tr>
</tbody>
</table>

All these functions are implemented as python modules and integrated to POX controller and simulated in Mininet Network Simulator. Whenever a request to the server comes on a forwarding device it will forward the request to the least loaded server and all the load balancing tasks are done in background. The Load status table and thus the Flow table entries are updated when the threshold hits the limit. An additional margin is also set considering the time taken for updating the flow table entries.

This method is proved efficient than the Mediator based method because the entire process begins from the server side itself. It also reduces the time taken for updating the flow table. This method is then compared with Random and mediator based load balancing methods. The result is shown in the figure below.
The above graphical representation shows the response time of three methods (Random, Mediator and Threshold). The x-axis depicts the request arrival rate/sec and y-axis depicts response time per sec. In this case random and mediator method shows 25 seconds and 18 seconds of response time. Random and mediator methods can serve requests faster at low rate, while our proposed method provides response time of 12 seconds when rate of requests increases.

5. Conclusion

In this paper, we proposed dynamic load balancing of servers using the threshold concept. In this method we used threshold value which is integrated in the server which will continuously monitor the server load and when the load exceeds the threshold value an alert message is generated which contains the server status of the overloaded server and this message is sent to the controller. This approach gives better response time and throughput compared to mediator based load balancing.

References


