Network Load Balancing Using SDN

Akhila Athresh, Naveen Revanna, Niket Kandya

Introduction

We present SDN based load balancing and discuss its architecture, design and implementation in this report. Traditional load balancing employs a single server or a group of servers in a network, functioning as load balancers. All requests to the application servers are routed through the load balancers, with chances of the load balancers becoming a bottleneck in a high traffic network. With SDN, we delegate load balancing to OpenFlow enabled switches, working in conjunction with the Floodlight controller to build a robust and scalable load balancing system.

Architecture & Implementation:

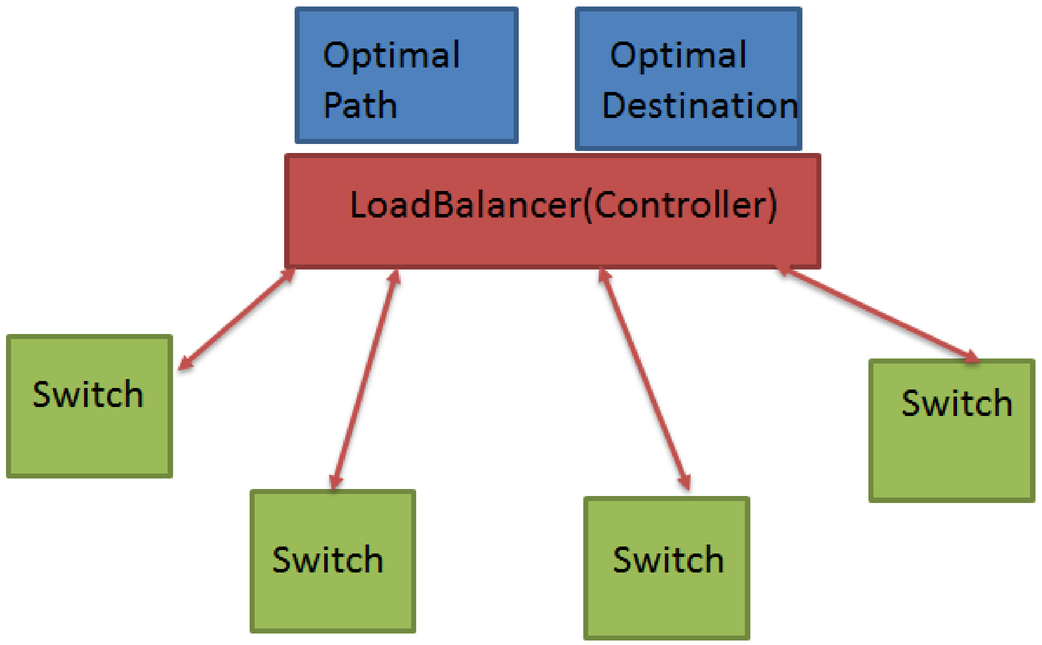

Figure 1. Architecture of SDN based Load Balancer

Our SDN based load balancer is a typical SDN network with Hosts connected to a switch which are controlled through OpenFlow protocol by a Floodlight controller. Load Balancing is implemented as a Floodlight module. This module determines the optimal destination host and also the optimal routes. Optimal routes are needed when there are multiple routes to a destination host.

Network services are made known to the hosts in the form of virtual IP addresses and virtual port numbers. There will be no host in the network which will be associated with the virtual IP address that will be chosen. The Load Balancer (LB) module is assigned with the responsibility of responding to network packets addressed to virtual IPs. Thus, it will respond to arp requests for the virtual ip addresses that will be requested by the hosts. When a host needs a service, it makes a service request addressed to the virtual IP and a virtual port number. When the packet reaches the switch for the first time, the switch sends a PACKET_IN message to the Floodlight controller. This packet is handled by the LB module, which chooses a server from its pool of service providers using a Load Balancing Algorithm. The controller sends a FLOW_MOD message to install a flow in the switch. This flow will rewrite the destination IP and port number of the current and all future packets addressed to virtual IP:virtual port. The flow also contains the port of the switch on which the packet should be forwarded after rewriting the destination IP. The choice of the switch port is made by LB module based on a similar Load Balancing algorithm. An appropriate hard timeout is also chosen by the FB module. Before the flow expires, all packets that arrive at the switch are automatically modified and forwarded by the switch based on the flow rules installed. Choice of the hard timeout value is also done by the algorithm that is used.

The various components discussed in the architecture are explained in detail in the following sections. Figure 2 depicts the complete workflow of our Load Balancer.

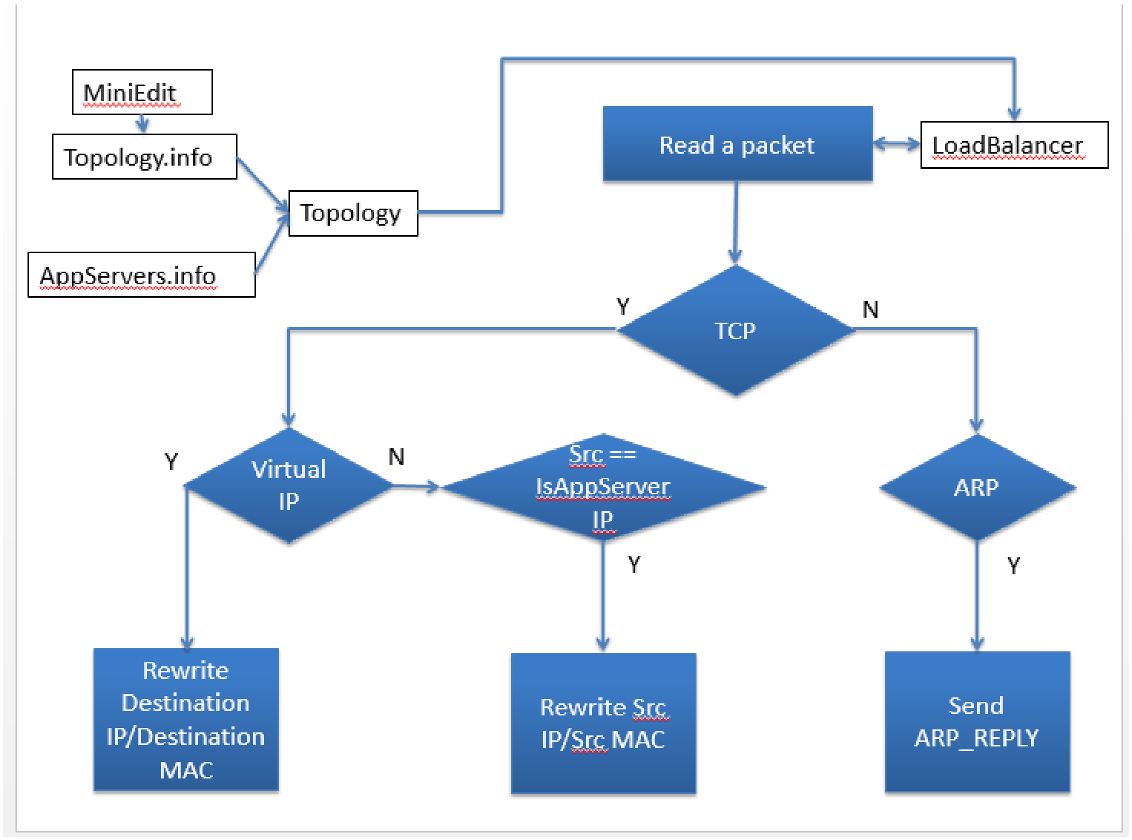

Figure 2: Load Balancer workflow

MiniEdit

MiniEdit is a python frontend for Mininet. It provides a GUI interface to draw a network consisting of Hosts and Open Flow switches. It leverages on the python api available for Mininet. We hacked this script to produce the topology information in the form of a CSV file. This file, named Topology.info, is read by the Topology class which will be explained in the further sections. MiniEdit can also be used to launch an xterm connected to the hosts. In our demo, we chose some of these hosts as service providers and used a shell script to launch a http server on the specified host. These scripts generate the file Appservers.info which contain information in the form of IP address and port number pair where the service is running. This information is also used by the Load Balancer Algorithm to choose the actual server to serve a service request.

LoadBalancer Module

The load balancer class is the main floodlight module that we implemented in this project. This class does all the decision making in the network based on the information available to it. For the prototype implementation, we used the Round Robin strategy for network based application level load balancing and network based network load balancing. The implementation however is generic and can be extended to use network statistics to make more network aware decisions.

Load Balancing Algorithm

We used Round Robin algorithm in our load balancer and we have the infrastructure in place for the least loaded variant of Round Robin algorithm to work. We employed Round Robin in both selection of destination server as well as in the selection of route to the destination server, if there are multiple paths to the destination server.

Round Robin mode passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced. Round Robin is simple in that it alternates the order of the address each time a request is sent. Simple Round Robin does not take transaction time, server load, network congestion, etc. into account while deciding the destination server or route. This works best for services with a large number of uniformly distributed connections to servers of equivalent capacity especially if the equipment that we are load balancing is roughly equal in processing speed and memory. Otherwise, it simply does load distribution.

A variant of the Round-Robin is to select the least loaded path and least loaded route while making load balancing decisions. After a flow times out, the controller collects statistics on the nature of traffic to each of the destination server. The controller uses this information to determine the least loaded server and installs corresponding rules in the switches, next time a request is received by the host.

This traffic information is also sent to the route manager, which uses it to determine the least loaded path to the chosen destination server, using one of the available shortest path routing algorithms.

ARP Handler

As mentioned before, ARP request packets from the end hosts are handled by the controller. The controller sends the ARP reply packets with a virtual mac address. If this is not done, the hosts will assume that no node with that IP address exists in the network and won't proceed to make a service request.

Topology Module:

For network load balancing, we used services from the topology class. The topology class keeps track of the network topology at all times. This information is read from a custom configuration file which is generated from our tweaked MiniEdit script. Having read this information, the network is modeled as an undirected graph.

As part of this prototype we implemented only Round Robin algorithm to choose available paths. This gives us the convenience of not calculating the routes multiple times. As such preprocessing involves calculating routes from all switches to all hosts using a modified version of Breadth First Search algorithm. After this, as a network packet is encountered, we query the topology class to provide us the next best route. The design in this class is flexible to plugin different algorithms for choosing routes. Network statistics as generated by the load balancer module can also be used in conjunction with a Djikstra-like algorithm to give the best network aware path.

SDN based Load Balancing in comparison to its traditional counterparts

1. Requests don't get queued up at a single server. Only a fraction of the service request packets queue up at the controller. The rest are handled by the Open Flow switches in the network.
2. Load Balancing becomes a property of the network. This makes the solution more scalable ,flexible and cost effective. Less work needs to be done when there is a new kind of network that is created. No need to purchase additional load balancing servers when a new service needs to be started.
3. When multiple services need the same algorithm, they can leverage on a single instance of the algorithm without the need to modify them to suit each service type.
4. New Load Balancer algorithm can be plugged in easily without much hassle.
5. Handling downtimes and changing locations of App servers (servers which serve a service) becomes easier and transparent to the hosts.

Conclusion

With SDN based Load Balancing , switches take up the role of load balancing and it becomes a property of the network. It can adapt and scale to network changes. SDN load balancers give the flexibility in choosing and experimenting with different load balancer algorithms, without the need to modify network state.

Github location:

https://github.com/AAthresh/LoadBalancer.git