**CS 6250: Computer Networks**

**Application-Aware Routing in SDNs**

Abhishek Jain

Anish Achuthan

Srikar Durbha

Introduction

Traditional networks were designed to forward packets from source to destination using the shortest route possible. Routers and switches were mostly agnostic to the applications being served by the network.

Today’s networks are forced to be application-aware, to improve user experience, provide service differentiation, and reduce operational costs.

Software-defined network (SDN) architecture allows service providers to build networks with increased application awareness, which can be built into the network by developing SDN controller applications that keep track of application-level characteristics and use that intelligence to provision flow into the network switches.

Application-Aware Routing

Application awareness is the result of gained intelligence about Layer 4 to Layer 7 protocol attributes and delivery requirements. The following are some use cases where the network has to be application-aware:

* A user watching videos requires more network bandwidth compared to a user browsing the Internet. The network has to dynamically adjust the bandwidth allocation to users based on the application, instead of statically allocating capacity.
* Gaming applications require lower network latency compared to applications transferring log files over the network. The network has to adapt to the application’s latency needs.
* A router forwarding request to a content server (cache or streaming server) doesn’t know whether the content resides in the content server. This increases the number of hops a request has to traverse before reaching the content server that can serve the request.

There are many such use cases that require the networks to be application-aware.

Traditional routing techniques, the ones in use today, are not application-aware. The application-deliver infrastructure in today’s internet is loosely-coupled with the packet forwarding infrastructure, which leads to duplication of functions in the network. This, in turn, impacts user experience, and increases operational costs. The request routing techniques in use today are –

* Policy-Based Routing (PBR)
* BGP Route Advertisement
* Use of a Centralized Request Router

These techniques have some limitations, which are listed below –

* The absence of mechanisms for the centralized request router to communicate with the L3 router to dynamically increase the bandwidth availability for a given user session. The network is always provisioned for peak bandwidth usage, resulting in unused bandwidth.
* Latency requirements of the application are not taken into consideration by L3 routers when routing the requests.
* CAPEX is increased as a result of maintaining a dedicated infrastructure for L7 request routing in addition to the existing routing infrastructure.

Software-defined network (SDN) architecture allows service providers to build networks with increased application awareness, which can be built into the network by developing SDN controller applications that keep track of application-level characteristics and use that intelligence to provision flow into the network switches.

SDN technology opens up an array of opportunities for network service providers. It helps them roll out monetizable services in their networks faster without having to depend on network equipment manufacturers. Application-aware routing architecture using SDN allows service providers to lower CAPEX/OPEX, and improve the overall end-user experience.

Methodology

Below, we discuss the methodology we have implemented.

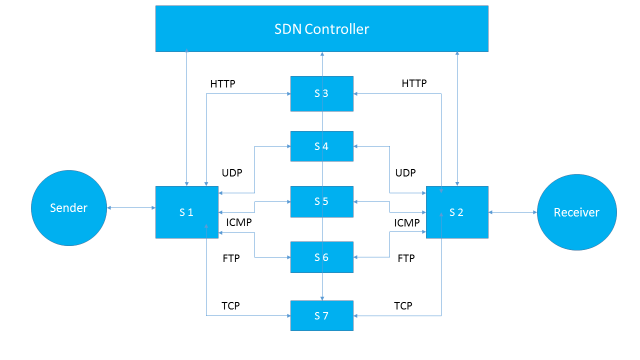
When the client transmits an IP packet, the switch inspects the packet and depending on the policy/rule installed in it, forwards the packet on a particular route.

If the switch doesn’t have any policy/rule installed, it sends the packet to the controller. The controller inspects the packet header and/or the payload, determines the type of packet (TCP, UDP, HTTP, etc.), and installs a policy/rule on the switch instructing it to forward packets along a particular route.

Our Implementation

We have created different paths for TCP, UDP and ICMP traffic. We have further classified TCP traffic in to HTTP and FTP traffic. For each different type of traffic, we have assigned a dedicated path.

Our implementation could easily be extended to be adaptive, so that the features of an application-aware network mentioned above could be realized.



POX Implementation

When the switch forwards the packet it receives to the controller, the controller checks the Ethernet packet for IP content.

* If it’s indeed an IP packet, the controller then checks the protocol header to determine if it’s a TCP, UDP or an ICMP packet.
* If the packet is a TCP packet, then it further determines if the packet is HTTP or FTP by inspecting the port number.
* After the controller determines the type of packet, it instructs the switch to send the packet along a specific path/route.
* According to our topology -
  + - HTTP packets are routed along S3.
    - FTP packets are routed along S6.
    - Other TCP packets are routed along S7.
    - UDP packets are routed along S4.
    - ICMP packets are routed along S5.

We identify different packets as being TCP/UDP using the identifier in the IP header for the transport protocol, after IP packets themselves are identified using an identifier in the Ethernet header.

The applications are identified using the server port being contacted by the client. We class packets as belonging to an FTP connection if the server port number is 21, and as HTTP if the server port being contacted is 8000.

C++ Implementation

We have also performed Deep Packet Inspection (DPI) using C++, where a similar topology was used. The controller installs a rule on the switch, instructing it to forward the packets along the different paths depending on the type of the packet.

The controller inspects the Ethernet packet for IP content and further for TCP, UDP and ICMP content based on the protocol header.

TCP packets are further inspected for HTTP and FTP traffic by performing a string search in the payload for keywords (HTTP 1.1 for HTTP traffic and FTP/SFTP for FTP traffic).

Only the first packet from a flow is sent to the controller, as in Pyretic/POX. Thereafter, a rule is installed on the switch for the flow. This allows the flow to be dealt with by the switch itself, saving the controller from having to deal with too many packets.