Exploiting Path Diversity in Datacenters Using MPTCP-aware SDN

MARTIN ORAVSKY, TOMAS URBAN

SLOVAK UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

ARCHITECTURE OF COMMUNICATION SYSTEMS

1. Analysis

1.1. Chapter introduction

This chapter is a brief analysis of Software defined networking and how it can benefit from multipath transmission control protocols, especially in datacenter-like topologies by exploiting path diversity.

In chapter 1.2, the basic concept of software defined networking will be introduced. Chapter 1.3 will briefly talk about Multipath TCP protocol which will be used later in this document. Chapter 1.4 will consider several problems regarding poor performance of SDNs due to random path selection.

1.2. Software defined networking

Today's networks mainly consist of routers, switches, hosts and many other network devices. In order for hosts to communicate, routers and switches performs as a transfer point between source and destination. These devices must provide very fast and reliable transfer of information, which, since the need for fast Internet connection is bigger and bigger every year, can be challenging.

The basic network device consists of data plane and control plane. Data plane is responsible for receiving the incoming packet and forwarding it to device's control plane, where the packet is processed. Control plane then forwards the packet again to the data plane, which then sends the packet via correct outgoing interface. This causes network device to store a lot of information and the CPU load can become quite high.

"Software defined networking is an architectural approach that optimizes and simplifies network operation by more closely binding the interaction (i.e., provosioning, messaging and alarming) among applications and network services and devices, whether they be real or virtualized. " [1]

SDNs use centralized object known as controller, which does all the computing and routing of the communication. The controller installs various forwarding rules based on global view of network topology on forwarders. Forwarders (also known as switches) are simple devices which forward the communication based on the rules installed by controller. If packet arrives on forwarder interface and no rule is matched, forwarder sends the packet to controller using OpenFlow protocol. The controller then processes the packet (e.g. flooding ARP request) and sends the packet back to forwarders along with corresponding rule.

This enables the network to better handle the communication while using the hardware resources better.

1.3. Multipath TCP

There are several options when sending communication on multiple path simultaneously while providing better redundancy and throughput. One can use Stream control transmission protocol (SCTP) which uses the concept of streams. When there is a need for L2 redundancy, bonding has proven itself as a good alternative.

In this document we propose Multipath TCP as a great alternative for improving path diversity in multipath communication in datacenter-like topologies.

Multipath TCP utilizes one TCP connection on multiple paths, on which separate TCP subflows are created, thus providing improved throughput and redundancy. Multipath TCP eliminates single point of failure by ability to switch communication to a working path when another path goes down. Multipath TCP enabled hosts use Multipath TCP options to establish a connection or to add a new subflow to an existing connection.

1.4. Problems with Multipath TCP and SDNs

Software defined networking can greatly benefit from Multipath TCP capabilities. However, software defined networks do not handle MPTCP traffic differently than the classic single-path TCP traffic. In order to exploit path diversity better and to provide true redundancy to the network, further implementation on the controller is necessary.

“A key aspect that affects MPTCP performance is the routing mechanism of the subflows. Currently, the most prominent and widely deployed routing mechanism in datacenters is a flow-based variant of Equal-Cost Multi-Path Routing (ECMP)” [2]. However, ECMP uses random hashing to split the subflows over different paths. This can cause a variety of problems, from which the most important is the high probability that multiple subflows end up on the same path while other paths remain not utilized properly. This also destroys the idea of redundancy.

The number of subflows is also a crucial aspect to performance of the network. Although the idea of more subflows can mean faster networks, the more subflows MPTCP use, the bigger the overhead is. More subflows mean more rules installed in forwarders and higher CPU utilization for the controller.

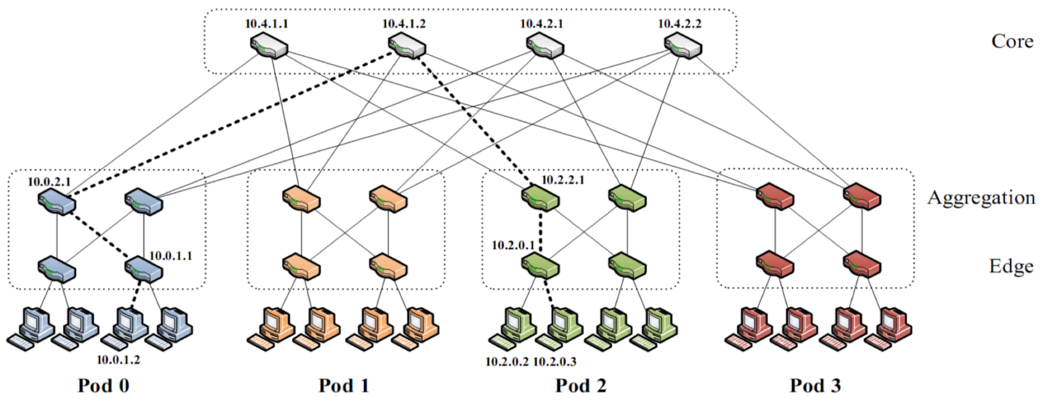
As shown in previous papers, MPTCP throughput also correlates with number of used interfaces on hosts. We will also deal with this issue since one-homed devices can quickly become a bottleneck in this type of network.

The purpose of this assessment is to implement an intelligent routing protocol that makes software defined networks MPTCP-aware by exploiting path diversity.

2. Design

2.1. Topology

The right choice of topology is crucial for datacenter network performance. One of the most used datacenter network topology is FatTree topology, which is shown below on picture 1. The topology consists of pods, which are connected on multiple core switches. The higher the level, the faster the bandwidth.



Picture 1 - FatTree topology

2.2. Controller

To make SDN MPTCP-aware, it is necessary to let controller know that multi-path communication occurs in the network. This can be achieved by packet inspection on controller. When a packet arrives on a forwarder and no rule is matched, the packet is sent to SDN controller which calculates the path for the subflow by using its global view of the topology. SDN controller then installs the rules onto every single forwarder that lies on the chosen path. The implementation will use Floodlight controller written in Java.

The shortest paths available will be computed using a graph traversal algorithm. Sets of paths will be stored on controller and these will be deterministically assigned to subflows. While using multiple redundant paths, MPTCP's performance should be maximized using a small number of subflows. Each path should be assigned to only one subflow, this can be achieved by storing IP addresses to paths.

When a new packet arrives on the controller, it will extract the needed information from the packet (IPs, ports, MPTCP options). If the option says we are dealing with a new subflow (MP_CAPABLE), it will find the shortest path between IPs and store the path into a hashtable. This path is then assigned to a subflow. If the options says we are dealing with additional subflow (MP_JOIN), we extract the token from the packet and find another shortest path for the subflow. The rules are installed on all switches belonging to the chosen path.

2.3. Testbed

To properly evaluate the algorithm, a virtual testbed will be needed. The tests will be performed on a Linux virtual machine running Floodlight controller and Mininet emulator. We create two more Linux virtual machines with multiple interfaces that will be connected to ports on the switches emulated in Mininet. After running FatTree topology in Mininet, we will evaluate how many subflows are needed in order to maximize MPTCP performance. We will then compare this approach to random approach used by ECMP protocol.

Literature

[1] T. D. Nadeau a K. Gray, SDN: Software Defined Networks, Sebastopol: O'Reilly Media, 2013.

[2] Savvas Zannettou, Michael Sirivianos, Fragkiskos Papadopoulos: Exploiting path diversity in datacenters using MPTCP-aware SDN, *2016 IEEE Symposium on Computers and Communication (ISCC)*