1 Software defined networking

1.1 Networking devices

A typical network device (e.g routers, firewalls...) is usually composed by:

1. A data plane
2. A control plane

The data plane is used to handle individual data packets locally. For example the data plane in a router is used to determine to which interface a packet should be forwarded to. The data plane usually applies a large amount of small, fast operations.
The control plane is used to handle control messages from other devices in the network, and allows the network device to work from a network wide perspective.
The messages from other devices may be used to configure policies which are then applied by the data plane (e.g the control plane receives a policy that determines which packets should be filtered, while the data place does the actual filtering).
Compared to the data plane, the control plane is used for more complex operations that usually require coordination along network devices.

1.2 Definition

The idea behind software defined networking (SDN) is to de-couple the control plane from the data plane, not only on a logical level but also physically.

The control plane is no longer part of the network devices and is instead moved to a centralized software component called a *controller*.

The basic architecture of a SDN is composed by the controller and by a series of network devices which in this context are called *switches*.

A SDN controller

1. Is able to communicate with the switches
2. Can be programmed: it is now possible to write software that defines how the whole network behaves.
   The switches keep executing their data plane locally, according to the policies received by the controller.
3. Is able to acquire information about the network it is managing, e.g it can discover the topology of the network to know where the switches are places and how they are connected.

The goal of the controller is to communicate with all the switches to inject into them the desired network behavior

The controller is logically placed between the network infrastructure and the application layer.

Using a series of high level programmable interfaces (also called *APIs* A*pplication P*rogrammable *I*interfaces) the controller is able to communicate with its neighboring layers.

Using the APIs connected with to the application layer (the *northbound*)interfaces various network devices can be implemented, e.g a network balancer, a firewall.

The communication between the controller and the switches is made possibile by a series of *southbound* interfaces.

Using these interfaces the controller can effectively apply the policies to the network, e.g it can send traffic rules to the switches connected to the controller.

Inside each switch there is a set of *generic tables* (also called *flow tables*) that are used to store the configuration sent by the southbound interfaces.

The southbound interfaces can also be used in "reverse": the switches are able to communicate with the controller using a series of predefined messages.

1.3 Openflow

Despite being relatively new, software defined networking became a popular approach for organizing new networks (e.g in 5G networks) Compared to other widespread network technologies such as the TCP/IP stack, software defined networking is a relatively new approach: the first paper defining protocol for SDN (the *Openflow Protocol*) was published in 2008.

The OpenFlow protocol defines a sets of API for remote communication with network plane devices, i.e. a southbound interface.

SDN switches inside the context of OpenFlow are also called *generic* switches; This is to avoid confusion with the usual meaning of the term switch, which usually refers to a device used to manage level 2 functionalities (i.e. the data link level, used to handle the physical addressing with MAC addresses). Openflow switches are able to interact up to the fourth layer (i.e. the transport layer, which may include protocols such as TCP or UDP).

A large portion of an OpenFlow Switch is composed by functionalities related to the data plane, the only portion dedicated to the control plane is delegated to the *control channel*.

The control channel includes one or more OpenFlow channel that allows the switches to communicate with one or more controllers (multiple controllers may be used to avoid a single point of failure in the infrastructure). dd OpenFlow switches include three kind of tables, used for handling data plane functionalities:

- *Flow tables*: the most important kind of table, they can be chained in a pipeline.
  They contain the rules that are applied to the incoming and outgoing traffic.
- *Group tables*: Users of a SDN can be assigned to different group; group tables allow traffic to be sent in multicast to users of a specific group (e.g. group tables can be used to send outgoing traffic to multiple ports if necessary)
- *Meter table*: table used to collects statistic about the current network.
  The gathered data can then be used to handle QoS (e.g rate limiting for traffic directed to a specific port)

1.3.1 Configuring the tables

To work properly, the flow tables inside the switches have to be configured by the controllers in the network. The solution is usually an hybrid of two types of configuration:

1. *Proactive configuration*: When the switch is first inserted into the network, the controller proactively fills the tables.
   The entries in the tables are static and do not change across the lifespan of the switch in the network.
   This kind of approach is obviously unfit for rapid changing scenarios such as those encountered in managing networks.
2. *Reactive configuration*: Initially there no entries in the flow tables; the tables are fully configured at runtime according to the network state.
   If a switch does not known where a package should be send to, it forwards the packet to the controller.
   The controller is aware of the full topology of the network, and is then able to edit the received package to include the routing information, before sending the package back to the switch..
   This leads to increased round trip time for the first packets sent across the network.
   However, communication between the switches and controller usually represent a bottleneck for the network's performance; a large amount of packets sent to and from the controller may cause congestion in the network and limit the available bandwidth.

Usually the preferred solution is based on an hybrid approach.
A starting set of rules is used at the start up of the network, to cover the expected data flows in the network.
For any unknown data flow that may arise during the lifespan of the network new routing rules will be issued by the controller.

1.4 SD-WAN

1.4.1 WANs

WANs (*Wide Area Networks*) are typically used by businesses who need to interconnect different sites of an organization (e.g branches of a bank or point of sales of a retail store).

Businesses may decide to invest in a WAN to reach for a higher level of quality than what may be offered to a typical consumer by an ISP.

WAN connectivity, also known as *dedicated connectivity* is by far more expensive than generalized connectivity (i.e the "normal" consumer internet).

The extra cost may be justifies by the guarantees a WAN can provide in terms of quality of service (e.g availability, end-to end latency...).

In a WAN the local networks of the branches are connected to the core of the network via border routers; there are multiple ways of achieving this design:

1. *Dedicated physical WAN* : the business buys the full physical infrastructure that the networks is built on (the cables, the switches etc.).
   While the company is the owner of the whole WAN, the cost is extremely high and network management (e.g enforcing security) is the the company's responsibility.

2. *Leased Lines* : the business relies on an ISP to establish private circuits between the branches. (e.g branch 1 must be connected to branch 2, which must be connected to branches 3 and 4)
   The private circuit may be created by assigning a dedicated wavelength to a particular customer.
   This solution offers only partial control, but is cheaper and offsets some of the network's management issues to the ISP.

3. *MPLS networks*: The business still relies on an ISP to provide connectivity, but unlike leased lines there are no established circuits between the branches. Instead, a mesh connectivity is provided , with a certain QoS guarantee.
   For each branch a *customer edge router* is installed, allowing the branch to access the MPLS network via a *provider edge router*.
   The *provider edge router* (PE) is in between the MPLS network and the companies, one PE may manage connections for multiple customers.

4. *SD-WAN*: this kind of WANs will be discussed in detail in the next section.

1.4.2 SD-WAN

The principles of software defined networking can be applied when building a WAN; While SD-WAN were introduced recently (in 2014) their popularity increased due to the possible decrease in costs when compared to traditional WAN solutions.

SD-WAN works by stipulating multiple contracts of generalized internet connections (e.g 4G, POM . . .) and then combining the different networks to guarantee a certain quality of service level.

Like the MPLS approach, a CPE (*Customer Premises Equipment*) is installed in each branch of the business.

The CPEs is connected to the branch's LAN and to multiple generalized connectivity networks.

The traffic generated by the LAN may then travel in different networks, potentially even from different ISPs.

An SDN controller, in this specific context called an *SD-WAN Controller* is responsible to communicate to each CPE how the outgoing traffic should be managed (e.g two branches may be connected using a fiber cable, if the cable connection fails the controller will instruct the branches to communicate using a 4G network).

MPLS may still be used in a SD-WAN context, a company could decide to use other (usually cheaper) kind of connections for most use cases and reserve a MPLS connection only exceptionally.

1.4.3 Load balancing and SLA in SD-WANs

As we previously discussed, in SD-WANs a controller's task is to instruct CPEs on how they should distribute outgoing traffic among the available links.

The traffic handled in a SD-WAN context is often heterogeneous and generated by different applications (i.e VoIP, emails, video streaming . . .).

Different applications have usually varying requirements in regards to the acceptable delay and the packet loss that can happen during the transmission of data.

Two important terms have to be introduced to describe the needs of each network application:

1. *QoS (Quality of Service)*: a quality indicator used to measure the service level of a given network communication. It is defined with a series of guarantees that must be respected during a network communication.
   In this paper QoS will always be defined in absolute terms (e.g. end to end latency is less than 5 milliseconds, 2% of packets have been dropped), however it can also be defined in relative tems (i.e how a given traffic flow is treated with respect to the others)

2. *SLA (Service Level Agreement)*: the SLA specifies the QoS that has to be guaranteed for a given application. It provides a series of bound on various measurable parameters (i.e the latency *must* be less than 5 milliseconds)

Usually the challenge is in minimizing the overall cost of transmitting a given amount of packets while still guaranteeing the SLA requirements.

1. Formalizing the problem: how can we define the load balancing problem in a structured away, to then be able to solve it?
2. Evaluating a strategy: what kind of strategy can be used to minimize the economic cost? As we previously discussed the most reliable solutions are usually the most expensive and viceversa: if possibile we would live to adopt the more expensive options only when strictly necessary.

1.4.4 Problem definition

As a first step in formalizing the problem, we can picture a scenario in which we have a CPE sending an arbitrary amount of packets to a sink.

The CPE is connected to the sink through a set of distinct links.

The goal of the CPE is to allow an *application* to transmit an arbitrary amount of packets to the sink. The transmission of data has some bounds in regards to:

- Average speed
- Average delay
- Packet loss

Each link has different characteristics in regard to:

- Available bandwidth *(i.e how many bytes can be sent per second through the link)*
- Delay *(i.e how long it takes for a packet to reach the sink)*
- Cost *(i.e a metric used to track how convenient is to use a link instead of another, the section 1.4.5 goes into further details*
- QoS *(i.e how many packets are lost and/or need to be sent multiple times?)*

With *application* we mean any kind of source of internet traffic with a defined set of requirements.
An application has a *life cycle*: it starts, it generates data according to arbitrary rules and it terminates when no more data has to be sent.

1.4.5 Measuring the cost of links

Each link as a cost associated with sending a packet on the link itself.
The cost of sending a packet is not a monetary cost, rather it is a measure of how convenient it is to use a given link.
Monetary costs for different kinds of connections are measured in multiple ways: i.e MPLS connections charge increasing amount of money depending on the amount of traffic, while typical consumer ISPs charge a fixed amount regardless of the amount of traffic.
This cost for each packet is fixed for each link, and it does not change through the lifetime of any application.
This means that when deciding the packet cost for a given link we assign lower costs to the links that have a fixed monetary cost, and an higher cost to the link the have a pay-as-you-go model.

1.4.6 The problem as a linear optimization problem

We can formalize the problem as a linear optimization problem; As a starting point we consider an application sending an amount N of packets of fixed size using three different links.

The amount of packets is fixed and generated by a single application: our goal is to minimize the overall cost of the transmission while respecting the different requirements.

To simplify the problem we set the bandwidth of the links in a way that it is always a multiple of the size of a single packet (i.e the bandwith of a link can only be an integer, 1 packet per second, 5 packet per second . . .)

We consider a set of links l_1, l_2, l_3.

For each one we define

- The link delay: d_1, d_2, d_3, measured in *ms*
- The link bandwidth: b_1, b_2, b_3, measured in *packets per second (pps)*
- The cost for sending a single packet c_1, c_2, c_3
- The probability of losing a packet, expressed as a number in the range $[0, 1)$

The total cost time required to send the packets can be expressed as:

$$N_1 \cdot c_1 + N_2 \cdot c_2 + N_3 \cdot c_3$$

Where N_i is the amount of packets sent on the the *ith* interface.

We also have to consider the bound on average delay:

$$\frac{N_1 * d_1 + N_2 * d_2 + N_3 * d_3}{N} \leq \text{delay}$$

The bound on average bandwidth:

$$\frac{N_1 * b_1 + N_2 * b_2 + N_3 * b_3}{N} \leq \text{bandwidth}$$

The bound on packet loss, expressed as a percentage:

$$\frac{N_1 \cdot p_1 + N_2 \cdot p_2 + N_3 \cdot p_3}{N} \leq \text{packet loss}$$

Finally, we need to express a bound to make sure every packet is actually sent.

This simplified version of the problem can be expressed as:

$$\begin{cases} \text{min} & N_1 \cdot c_1 + N_2 \cdot c_2 + N_3 \cdot c_3 \\ \text{subject to:} & \dfrac{N_1 \cdot d_1 + N_2 \cdot d_2 + N_3 \cdot d_3}{N} \leq \text{delay} \\ & \dfrac{N_1 \cdot b_1 + N_2 \cdot b_2 + N_3 \cdot b_3}{N} \leq \text{bandwidth} \\ & \dfrac{N_1 \cdot p_1 + N_2 \cdot p_2 + N_3 \cdot p_3}{N} \leq \text{packet loss} \\ & N_1 + N_2 + N_3 \geq N \end{cases} \tag{1}$$

1.5 Extending the problem

While the linear optimization problem allows for an easy and clear solution, it can only work if some assumptions are made:

- Only a single application is sending data at any given time
- The amount of traffic generated by the application must be known beforehand (this is usually unrealistic, unless for specific situations, i.e file transfer)

The linear optimization strategy is still useful to set a lower bound on the lowest possible overall transmission cost.

To allow for modeling of more complex scenarios, we extend the problem definition to allow the presence of multiple applications.

In this new scenario, each application can now generate traffic at any point of its lifetime: the total amount of traffic can not be determined before the .

Examples of different applications may include some that generates:

- A fixed amount of traffic every second.
- A random amount of traffic every hundred milliseconds
- A burst of traffic when starting the application, then no more.
- Any kind of commmonly studied traffic model (i.e Poisson distribution traffic model, Pareto distribution model)

1.6 Existing strategies

In the paper *Intent-Based Routing Policy Optimization in SD-WAN* a strategy is discussed to optimize different metrics in a scenario with multiple applications with different QOS requirements.

The goal of the strategy is to provide a split ratio for the traffic generated by the different applications among the available links

The original paper provides a formulation with a model that may include up to three different weighted objectives, including:

1. An objective that measures if the SLA for a given application (in the original paper called a *flow group*)can be violated.
2. An objective function that minimizes the MLU (Maximum link utilization)
3. An objective function that can eventually be used to decrease the delay beyond SLA requirements.

For our purposes, we can slightly modify the proposed model.

We can ignore the third objective function that may eventually improve the obtained result, as we are only interested in respecting the boundaries.