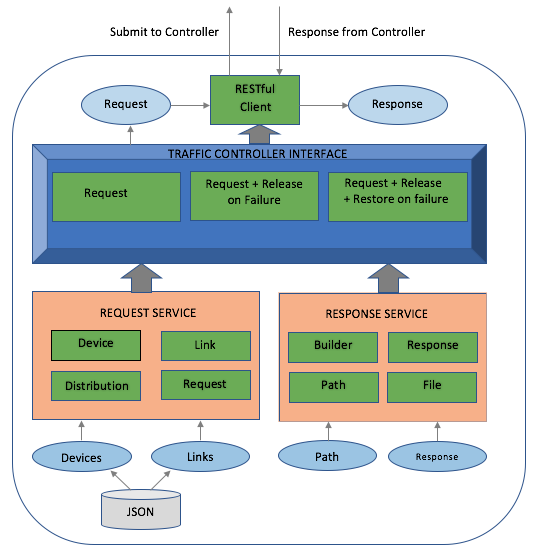
TRAFFIC GENERATOR



Traffic Generator Architecture

The traffic generator is merely a client to the controller (OSCARS SDN) which uses the controller’s API to submit the request. Since controller does most of the tasks from running a simple algorithm to installing flows in the networking devices in the topology and other business related operations, the traffic generator only has to submit the request to controller which then reserves circuits end-to-end as per the request parameters. Upon receiving the response from the controller, the traffic generator parses, processes them locally and carry out operations related to the research requirements.

Traffic Generator mainly comprises of two major components, traffic controller interface and service components that deals with all the business-related functionalities.

THE CONTROLLER INTERFACE

Each controller maintains its own service flows in service classes and based on which future actions are decided. The future actions include,

1. New connection request
2. Aborting previously held connection request
3. Aborting previously held but failed request (RELEASE)
4. Converting the connection request from one type to another (Example: Multiple Unicasts to Unicast Survivability)
5. Restore previously failed connection (RESTORE)

All these future actions are controlled based on what type of traffic one would like to generate. You can invoke the controller by its name (see the script file in git). One can also include/write more controllers if the actions of the current controllers cannot satisfy the demands of traffic generation. Read the complete document at [30] on adding more traffic controllers and how to configure/register them with the spring container.

This simple controller actions take one of the below three actions. They are,

1. Submitting the next request if the previous request is completely settled.
2. Releasing the resources of failed connection if necessary. (optional)
3. Restoring the connections based on the requirements. (optional)

These actions are decided upon receiving the response from the controller OSCARS. Although controller (OSCSARS) hosts most of the reservation capabilities and routing algorithms, the research scope is somewhat limited. Hence it is important for traffic generator to consider the possibilities to explore the controller OSCARS for new algorithms that may be future aspirants. This workaround can be possible until most of the end-to-end path computation algorithms runs on top of Djikstra’s. That said, traffic generator relies entirely on the algorithms of controller OSCARS except that it could only force the input parameters to OSCARS to pursue and obtain the status of the network it wants to have. Current workarounds in OSCARS controller are,

1. Unicast/ Anycast/ Manycast (directly uses OSCARS API)

2. All release and restoration actions for protection

3. Iterative path

4. Parallel paths

5. Single/Multi-link disjoint Djikstra

Service Components

Service classes helps traffic controller decide the type of request it has to submit to the controller OSCARS. It mainly has two components, request and response. Each of these components maintain its own models.

Request Components

Request components have the total control over input parameters. These includes devices and links for requests (also makes sure one each of them are unique), start time, end time and bandwidths of connection based on respective distributions and finally blacklisting of links if required. They also maintain the models for devices and links.

Response Components

The response components vary based on the type of traffic controller chosen. The traffic controller interface commits to a service for a given set of input parameters to the traffic generator. They can be,

1.General response service (Unicast, Anycast, Manycast)

2.Protection service (Release and Restore)

3.Iterative path service (Release and Restore)

Similarly, more services can be added. The idea behind this way of implementation is to make the tool maintainable and easily modifiable. The detailed report on this can be found at [30]

Other Service Components and Utilities

Besides Request and response there are other service components such as file and builder services and utility classes.

Rest API

Once the request is ready to submit to OSCARS, the traffic controller pushes it to rest controller. Upon receiving the request, the rest controller converts the Json to response objects.

Note: Current implementation of rest controller handles few HTTP exceptions and status messages. However, there may be some unexplored exceptions which needs to be handled in a proper way to get the accurate results. Visit the github link provided above to know how the exceptions are handled and what specific actions need to be taken.

Configuration files

These files contain devices, links and the length of links. Hence, if the properties of a link needs to be changed, it is enough to change the respective configuration file. This also allows reshaping of topology and easy replacement of entire network topology.

Note: The topology being modified or replaced should follow the json file format with the same set of keys.