(10)

REpresentational State Transfer (REST)

- It is an architectural style, that is used to define APIs on the northbound.
- It is a standard way of constructing northbound APIs for SDN controller.
- A REST API is not a protocol or a language, and there is no well known established standard available yet.
- REST apis contains different constraints
- Some of the constraints are mentioned here,

  Client - Server

  Stateless

  Cache

  Uniform interface

  Layered System

  Code on demand

- Any API for the northbound of the SDN architecture, they have to follow these given restrictions.
- REST architectural style is actually used to define other APIs.

And this is now becoming a standard way of constructing northbound API for the SDN controller.

- This all given constraints are important to be restful.

- the objective of this constraints is to maximize the scalability and interoperability or independence of these software interactions with the other component.

- this REST take care of concept of a web-based access.

- If you are using the REST, you are using the web-based access for interaction blw the application and the services.

- ① Client - Server.

  ↳ the interaction blw the application and the server is in the client-server, request-response style.

  - Client make request, server respond to those request. It means that the interaction blw the applications and the server must be in form of request-response style.
  - It will allow client and server

Components to evolved independently means — one component either the server or the client would be able to change without effecting the other component.

- It also supports the portability of the server side functions to multiple platforms.

(2) Stateless

↳ this is related to the web → Statefull, Stateless.

- Stateless means that, each request from the client to a server must contain all necessary info. to understand the request.
- It cannot take advantage of any stored context on the server.
- Same in the case from responses from the server side.
- Any response from the server side will contain all the desire info. for the request.
- In simple, server does not retain any record of the client state.

- If the client and server resides on different machine and they communicate with the help of a protocol, that protocol need to be connection oriented.
- So REST run over http protocol. and that protocol is a stateless protocol.
- And stateless protocol means, you have to retain no info. neither on the server side nor on the client side,
- But, we do have support for cache.

- (3) Cache.
  - But to use cache we need permission.
  - Cache constraints means, the data within a response to a request.
  - If a response is cacheable, then a client cache is given the right to reuse that response for later use. And same case for request.
  - Here client is given permission to remember this data because the data is not likely to change on the server side.
  - If it is changed then, we will be in

inconsistent state.

- The adv. of this approach is, we are reducing the comm^n overhead blw the client and the server.
- And also reducing the server processing burden.
- If you have requested for information, you are a client then you have to store those info. in your local cache
- and it will minimize the load on the server. as well as it will reduce the comm^n overhead blw the client and the server.

- (4) Uniform Interface.
- This REST give emphasis on the uniform interface blw the components.
- It will enable the Controller services to evolve independently.
- It will provide the ability for the SDN controller to use SW components from various vendors in order to implement Controller.
- If you are using a Uniform Interface, this will enable the controller services to evolve independently.

(5) Layered system and code on demand. (6)

(6)

- The layered system means, that a given function is organized in layer.
- And each layer only have direct interaction with the layers immediatly above it or immediately below it.

→ Code on demand.

- The REST allow client functionality to be extended by downloading and by executing code in form of applets or in the form of script.

(11) Cooperation and Coordination Among Controllers.

- We have northbound and southbound APIs but what about the east and west APIs?
- The SDN controller will have an east and west interfaces.
- That will enable communication within the SDN controllers or maybe with other networks.
- To understand the cooperation and coordination, we need to understand the concept of distributed environment.
- A key architectural design decision is whether you are using a single controller or more than one controller.
- A single centralized controller or distributed set of controllers.
- We have 2 options, either we are using a single controller or we are using a set of controllers.
- If we are using a set of controllers then those controllers are distributed.

The job of controller is to control the Data plane switches to instruct the Data Plane switches.

- If we are using a single centralized controller then a single server will manage all the data plane switches in the network.
- In larger network setups, in larger organizations, in large enterprises, the deployment of a single controller to manage all network devices would be undesirable. would not be a good practice.
- A more likely scenario is that, the operator of a large organization, they will divide the whole network into a number of domains, and those domains are non-overlapping and those domains are commonly reffered by SDN Iceland, and they are going to be managed by the distributed controllers.

Reasons for using SDN domains include those in the list

—
—
—
—

↑ Advantages of using more than one controller, or for using distributed controller.

① → Scalability
- The number of devices and SDN controller can manage is limited.
- A large network setup may need to deploy multiple SDN controllers

② → Reliability
- The use of multiple controllers would avoid the risk of a single point of failure.

③ → Privacy
- A carrier may choose to implement different policies for privacy in different domains.

→ for eg:- In SDN Domain, may be dedicated to a set of customers who implement their own higly customized policies requiring that some networking information in this domain should not be disclosed to an external entity.

- So inside a specific domain, you can specify diff^n types of privacy policies that would be implemented within that domain.
- And it will increase the deployment.
- As we know, dividing a network into multiple individual domains allow for more flexible incremental deployment.
- You have 4 domains and after some time the requirements have been changed and you want to add another domain, that is going to be a flexible solution and easy solution.

- Now, we have more than one controllers, so we need them to coordinate and cooperate with one another.
- Distributed controllers may be allocated in a small area or maybe they are deployed worldwidely.
- In either case, they are distributed but horizontally.
  - ↳ Each controller control a non-overlapping subset of the data plane switches.
- A vertical distributed architecture is also possible and in that case, the controller tasks are distributed to different controllers depending on the criteria, such as,
  - locality,
  - Network topology - - - -
- Each one is responsible for controlling non-overlapping subset and for that, the coordination is require for communication.

— Here we can see the various routes, using which user can send the data. (1)

(1) Chap - 5 (last Part)

SDN Application Plane.

→ We have already covered the SDN data plane and control plane. And also a different types of APIs. and functionalities.

- Now today we will explore Application plane in details.
- We will also see different types of application which are related to security network, services and also some business application related to data center, cloud computing and information centric networks.

* SDN Application plane Architecture.

- Application plane contains different applications and services.
- These Applications and services will define or it will explore how we can actually monitor and control network resources and network behaviour.
- These Application will interact with the SDN control plane with the help of application control interfaces
- Here, we can see different types of

interfaces like, the Northbound Interfaces, Remote Interface, User Interface (local) and User Interface (Remote).

- And also different types of applications and functions and services like, the data centre, Mobility, Information centric Networking, Security, Measurement and Monitoring and Traffic Engineering.

- We will see this from the bottom.

(1) North Bound Interface:

- Here we will see the North Bound Interface with the Application layer point of view.
- This interface enable applications to have access to the control plane functionalities and their services.
- But the good point is, it will not need to know the details of the underlying network SDN switches because abstraction is there. (Details should be hidden from the layers above it).
- This interface provides an abstract level view of the network resources, that are controlled by the S/W in the SDN control Plane.
- This interface can be either local or it can be remote.
- local interface means, the SDN applications are running on the same machine as the control plane and

while on the other side, the remote northbound interface mean that, the application can be running on a remote machine.

- One of the well-known example is the REST APIs.
- ABove these interfaces we have the Network Services Abstraction layer.
- this is define in one of the RFC - 7426.
- this RFC defines this network abstraction layer and this layer is b/w the control and the application plane.
- It describe it as a layer, that will provide the service abstractions.
- this layer will provide an abstract view of the nework resources. and it will hide the details of the underlying data plane switches or devices.
- Why we need this layer? so that our applications on top of it could be written, that would operate across a large number of network OS.
- This concept is same as hypervisor like virtual machine monitor that also decouple applications from the underlying

and underlying hlw, OS.

The Network services Abstraction Layer could be considered to be part of this North bound Interface, Not as a seperate layer.

- The funationality is incorporated in the control plane rather than the application plane.

~~- It is possible to~~

- then we have different types of network applications on top of that and that are the actual things that happens at the application plane.
- many different applications could be implemented or could be deployed for an SDN and here we can see, six categories of them and they are related to data center, traffic engineering, mobility, monitoring, security and information centric.

→ And the other component is User interface.

- Interfaces enable a user to configure different types of parameter in the SDN applications and it enables the user to interact with the application itself.
- There are 2 possible types of interfaces, Local and Remote.

If the user is in the same machine, there is an example of local interface. But if user login to the ~~diff~~ server from remote area ~~to the application server~~ by using a communication network facility then that is going to be the example of a remote user interface.

Network Services Abstraction layer.

- Abstraction is the concept that refers to the amount details about the lower levels of any mode, that is visible to the higher level.
- In Reality, more abstraction means less details of the underlying hardware and less abstraction means more details about the underlying hardware resources.
- More abstraction is an important design principle for SDN.
- On the other side, abstraction layer is a technique, that will translate high level request like the application layer request or the User Request into low level commands that hw and switches can understand.
- And these commands are issued to perform certain task on request generated by the user.
- And APIs, that is one such mechanism to achieve this objective. Why? Because it will hide the implementation details of a lower level of abstraction from the cut a heigher level.
- This represents the basic properties of

characteristic of network entities, like, switches, links, & interfaces, ports, flow of data in such a way that the network programs can focus on the functionalities without having to program the detail action for the lower level.

- Developer can focus to develope the higher level of the development.