Introduction to Cloud Computing (CS 524)

(Homework 6)

Prof. Igor Faynberg

Student Name: **Paras Garg**

Course Section: **CS 524-A**

Homework 6.1 –
Explain the motivation for developing COPS. What were the goals of COPS framework?

Solution 6.1 –
The Common Open Policy Service (COPS) Protocol is part of the internet protocol suite as defined by the IETF's RFC 2748. COPS specifies a simple client/server model for supporting policy control over Quality of Service (QoS) signaling protocols (e.g. RSVP). Policies are stored on servers, and acted upon by Policy Decision Points (PDP), and are enforced on clients, also known as Policy Enforcement Points (PEP).

Motivation for developing COPS
The IETF started to address the problem gradually, strictly on a specific need basis. The first such need was the policy configuration in support of QoS. The propose model had introduced new challenges - the need to maintain a synchronized state between the network manager and the device. Another challenge came from the potential interference among two or more network managers administering the same device. And then there is a need for policy-based management.

Network providers wanted to have a mechanism that would enable granting a resource based on a set of policy rules. The decision on whether to grant the resource takes into account information about the user, the requested service, and the network itself. Employing SNMP for this purpose was not straightforward, and so the IETF developed a new protocol, for communications between the network element and the *Policy Decision Point* (*PDP*)—where the policy-based decisions were made. The protocol is called *Common Open Policy Service* (*COPS*).

As an important aside, COPS has greatly influenced the Next-Generation telecommunications Network (NGN) standards, characterized by (1) the prevalent use of IP for end-to-end packet transfer and (2) the drive to convergence between wireline and wireless technologies.

Goals of COPS framework
A chief objective of this policy control protocol is to begin with a simple but extensible design. The main characteristics of the COPS includes:

- The execution of policy-based control over the QoS admission control decisions, with the primary focus on the RSVP protocol. It also support for pre-emption, various policy styles, monitoring, and accounting. Pre-emption here means the ability to remove a previously granted resource so as to accommodate a new request.

- The protocol employs a client/server model where the PEP sends requests, updates, and deletes to the remote PDP and the PDP returns decisions back to the PEP.

- The protocol uses TCP as its transport protocol for reliable exchange of messages between policy clients and a server. Therefore, no additional mechanisms are necessary for reliable communication between a server and its clients.

- The protocol is extensible in that it is designed to leverage off self-identifying objects and can support diverse client specific information without requiring modifications to the COPS protocol itself. The protocol was created for the general administration, configuration, and enforcement of policies.

- COPS provides message level security for authentication, replay protection, and message integrity. COPS can also reuse existing protocols for security such as IPSec (Internet Protocol Security) or TLS (Transport Layer Security) to authenticate and secure the channel between the PEP and the PDP.

(Reference: https://en.wikipedia.org/wiki/Common_Open_Policy_Service, Cloud Computing: Business Trends and Technologies,
http://wwwen.zte.com.cn/endata/magazine/ztetechnologies/2002year/no2/articles/200312/t20031212_161146.html,)

Homework 6.2 –
What feature is intrinsically new to COPs (compared to the SNMP) .

Solution 6.2 –
Something intrinsically new about COPS as compared with SNMP or CMIP is that COPS employs a *stateful* client–server model, which is different from that of the remote procedure call. As in any client, server model, the PEP (client) sends *requests* to the remote PDP (server), and the PDP responds with the *decisions*. But all the requests from the client PEP are *installed* and remembered by the remote PDP until they are explicitly deleted by the PEP. The decisions can come in the form of a series of notifications to a single request. This, in fact, introduces a new behavior: two identical requests may result in different responses because the states of the system when the first and second of these requests arrive may be different, depending on which states had been installed. Another stateful feature of COPS is that PDP may "push" the configuration information to the client and later remove it.

Unlike SNMP, COPS was designed to leverage self-identifying objects and therefore it is extensible. COPS also runs on TCP, which ensures reliable transport. Although COPS may rely on TLS, it also has its own mechanisms for authentication, protection against replays, and message integrity. The COPS model was found very useful in telecommunications, where it was both applied and further extended for QoS support. As far as Cloud Computing is concerned, the primary application of COPS is SDN.

(Reference: https://en.wikipedia.org/wiki/Data_center_bridging, Cloud Computing: Business Trends and Technologies)

Homework 6.3 –
Motivation for developing NETCONF (Read RFC 3535, and answer the following questions you can cite the relevant text of RFC 3535 verbatim)

a. Why the SNMP transactional model and the protocol constraints make it more complex to implement MIBs, as compared to the implementation of commands of a command-line interface interpreter?

b. What is the problem with the SNMP lack of support for easy retrieval and playback of Configurations?

c. List the operators' requirements for network management.

Solution 6.3 –

a. The SNMP transactional model and the protocol constraints make it more complex to implement MIBs, as compared to the implementation of commands of a command line interface interpreter. A logical operation on a MIB can turn into a sequence of SNMP interactions where the implementation has to maintain state until the operation is complete, or until a failure has been determined. In case of a failure, a robust implementation must be smart enough to roll the device back into a consistent state.

b. SNMP does not support easy retrieval and playback of configurations. One part of the problem is that it is not easy to identify configuration objects. Another part of the problem is that the naming system is very specific and physical device reconfigurations can thus break the capability to play back a previous configuration.

c. List of the operators' required for newtwork management are:

   During the breakout session, the operators were asked to identify needs that have not been sufficiently addressed. The results produced during the breakout session were later discussed and resulted in the following list of operator requirements.

   - Ease of use is a key requirement for any network management technology from the operators point of view.

   - It is necessary to make a clear distinction between configuration data, data thatdescribes operational state and statistics. Some devices make it very hard to determine which parameters were administratively configured and which were obtained via other mechanisms such as routing protocols.

   - It is required to be able to fetch separately configuration data, operational state data, and statistics from devices, and to be able to compare these between devices.

   - It is necessary to enable operators to concentrate on the configuration of the network as a whole rather than individual devices.

- Support for configuration transactions across a number of devices would significantly simplify network configuration management.

- Given configuration A and configuration B, it should be possible to generate the operations necessary to get from A to B with minimal state changes and effects on network and systems. It is important to minimize the impact caused by configuration changes.

- A mechanism to dump and restore configurations is a primitive operation needed by operators. Standards for pulling and pushing configurations from/to devices are desirable.

- It must be easy to do consistency checks of configurations over time and between the ends of a link in order to determine the changes between two configurations and whether those configurations are consistent.

- Network wide configurations are typically stored in central master databases and transformed into formats that can be pushed to devices, either by generating sequences of CLI commands or complete configuration files that are pushed to devices. There is no common database schema for network configuration, although the models used by various operators are probably very similar. It is desirable to extract, document, and standardize the common parts of these network wide configuration database schemas.

- It is highly desirable that text processing tools such as diff, and version management tools such as RCS or CVS, can be used to process configurations, which implies that devices should not arbitrarily reorder data such as access control lists.

- The granularity of access control needed on management interface needs to match operational needs. Typical requirements are a role-based access control model and the principle of least privilege, where a user can be given only the minimum access necessary to perform a required task.

- It must be possible to do consistency checks of access control lists across devices.

- It is important to distinguish between the distribution of configurations and the activation of a certain configuration. Devices should be able to hold multiple configurations.

- SNMP access control is data-oriented, while CLI access control is usually command (task) oriented. Depending on the management function, sometimes data-oriented or task-oriented access control makes more sense. As such, it is a requirement to support both data-oriented and task-oriented access control.

(Reference: Cloud Computing: Business Trends and Technologies, https://mentor.ieee.org/802.11/dcn/16/11-16-1436-01-0arc-yang-modelling-and-netconf-protocol-discussion.pptx)

Homework 6.4 –
Does NETCONF use REST API in its Messages layer?

Solution 6.4 –
No, A RESTful protocol that provides a programmatic interface over HTTP for accessing data defined in YANG using the datastores defined in NETCONF.

	SNMP	NETCONF	SOAP	REST
Standard	IETF	IETF	W3C	-
Resources	OIDs	Paths		URLs
Data models	Defined in MIBs	YANG Core Models		
Data Modeling Language	SMI	YANG	(WSDL, not data)	Undefined, (WSDL), WADL, text...
Management Operations	SNMP	NETCONF	In the XML Schema, not standardized	HTTP operations
Encoding	BER	XML	XML	XML, JSON,...
Transport Stack	UDP	SSH TCP	SSL HTTP TCP	SSL HTTP TCP

The Messages layer provides a mechanism for encoding remote procedure calls (RPCs) and notifications. The <rpc> element has a mandatory attribute "message-id", which is a string chosen by the sender of the RPC that will commonly encode a monotonically increasing integer. The receiver of the RPC does not decode or interpret this string but simply saves it to be used as a "message-id" attribute in any resulting <rpc-reply> message. The sender MUST ensure that the "message-id" value is normalized according to the XML attribute value normalization rules defined in [W3C.REC-xml-20001006] if the sender wants the string to be returned unmodified.

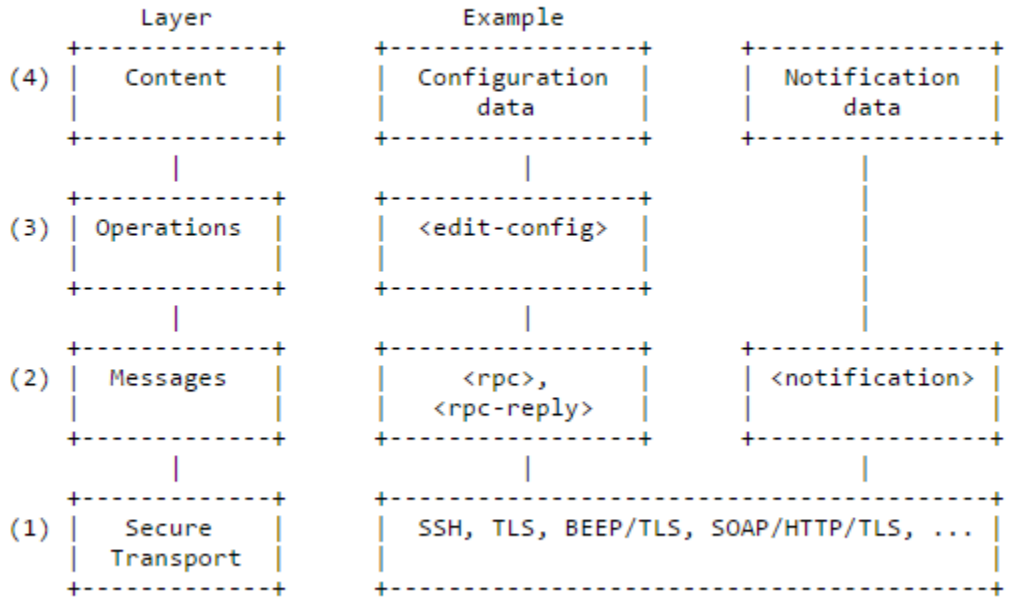


Figure 1: NETCONF Protocol Layers

(Reference: Cloud Computing: Business Trends and Technologies, https://www.ietf.org/slides/slides-edu-netconf-yang-00.pdf, https://tools.ietf.org/html/rfc6241)

Homework 6.5 –

Name a de-facto modeling language for NETCONF. What document is it specified in? What is the name of this language's XML-based representation?

Solution 6.5 –

A de-facto landuage for NETCONF is **YANG**, and it is specified in **IETF 6020** under category Standard Track having ISSN 2070-1721 (https://tools.ietf.org/html/rfc6020).

The module is the base unit of definition in YANG. A module defines a single data model. A module can define a complete, cohesive model, or augment an existing data model with additional nodes. The names of all standard modules and submodules MUST be unique. Developers of enterprise modules are RECOMMENDED to choose names for their modules that will have a low probability of colliding with standard or other enterprise modules.

But the name of XML-based representation of YANG is **YIN** Module.

(Reference: Cloud Computing: Business Trends and Technologies, https://tools.ietf.org/html/rfc6020, https://www.ietf.org/proceedings/72/slides/netmod-5.pdf)

Homework 6.6 –

List the steps involved in *onboarding* of an application?

Solution 6.6 –

The actual process of onboarding ('forklifting' workloads) has seven relatively straightforward steps:

1. **Define the workload –** The number and type of virtual machines required for migration will depend on the nature and scale of the workload, and the way it interacts with software and services not being migrated.

2. **Provision cloud resources –** Service providers will have a self-service interface for the creation of accounts and purchase of the services that you need (eg, servers, storage, network).

3. **Establish a connectivity bridge** – Secure and transparent bi-directional connectivity, usually through an internet VPN, is required between your data centre and the cloud, both for the migration itself and for cross-platform application interactions after migration.

4. **Deploy the workload** – With connectivity in place, virtual machines can be configured and connected to services remaining behind (such as Active Directory), followed by the transfer of the application and any associated databases, software and services being migrated.

5. **Ensure seamless two-way access –** Smooth integration is required between the cloud workload and services not migrated, and you need to be able to monitor and manage the application as well as the cloud infrastructure.

6. **Test and validate** – However well you've prepared and tested prior to deployment, there may be surprises. Has everything been transferred correctly.

7. **Discontinue the old service –** When you're certain that everything is working well, you can give access to users and decommission the enterprise service.

(Reference: http://www.interxion.com/globalassets/_documents/whitepapers-and-pdfs/cloud/WP_CLOUDONBOARDING_en_0715.pdf)

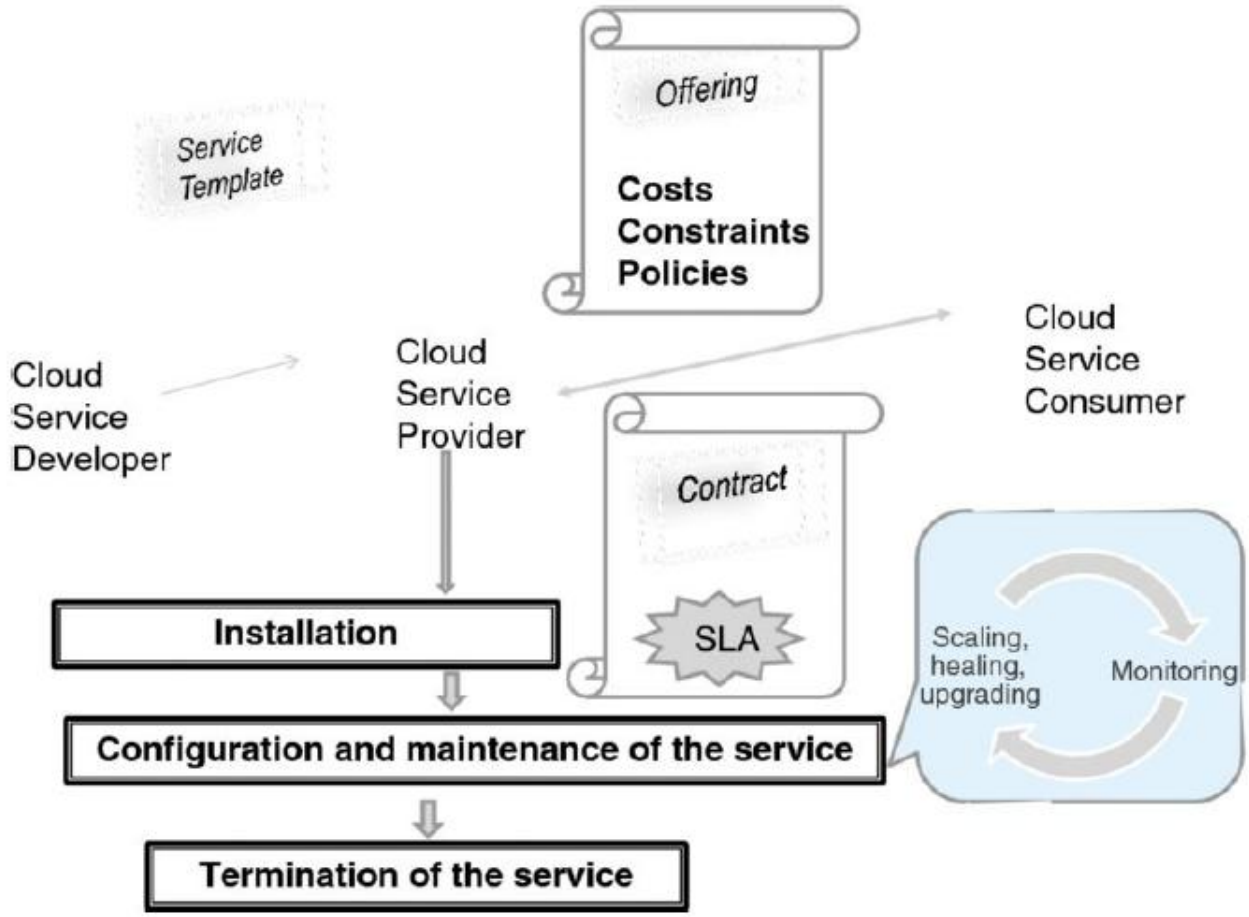
Homework 6.7 –
List the actors involved in the service life cycle and its stages. What constitutes an *offering?*

Solution 6.7 –
The three entities involved here are the Cloud service provider, the Cloud service developer, and the Cloud service consumer.

First, suppose the instances for a load balancer and two servers have been created successfully, but creating the virtual machine for the third server has failed. What should the user program do? Deleting all other instances and restarting again is hardly an efficient course of action for the following reasons. From the service developer's point of view, this would greatly complicate the program (which is supposed to be fairly simple). From the service provider's point of view, this would result in wasting the resources which were first allocated and then released but never used.

Second, assuming that all instances have been created, a service provider needs to support elasticity. The question is: How can this be (a) specified and (b) effected? Suppose each of the three servers has reached its threshold CPU utilization. Then a straightforward solution is to create yet another instance (which can be deleted once the burst of activity is over), but how can all this be done automatically? To this end, perhaps, maybe not three but only two instances should have been created in the first place.

The solution adopted by the industry is to define a service in more general terms (we will clarify this with examples), so that the creation of a service is an atomic operation performed by the service provider— this is where orchestration first comes into the picture. And once the service is deployed, the orchestrator itself will then add and delete instances (or other resources) as specified in the service definition.



The service provider creates an *offering* for a service consumer by augmenting this template with the constraints, costs, policies, and SLA. On accepting the offering, the consumer and provider enter into a *contract*, which contains, among other items, the SLA and a set of specific, measurable aspects of the SLA called *Service-Level Objectives* (*SLOs*).

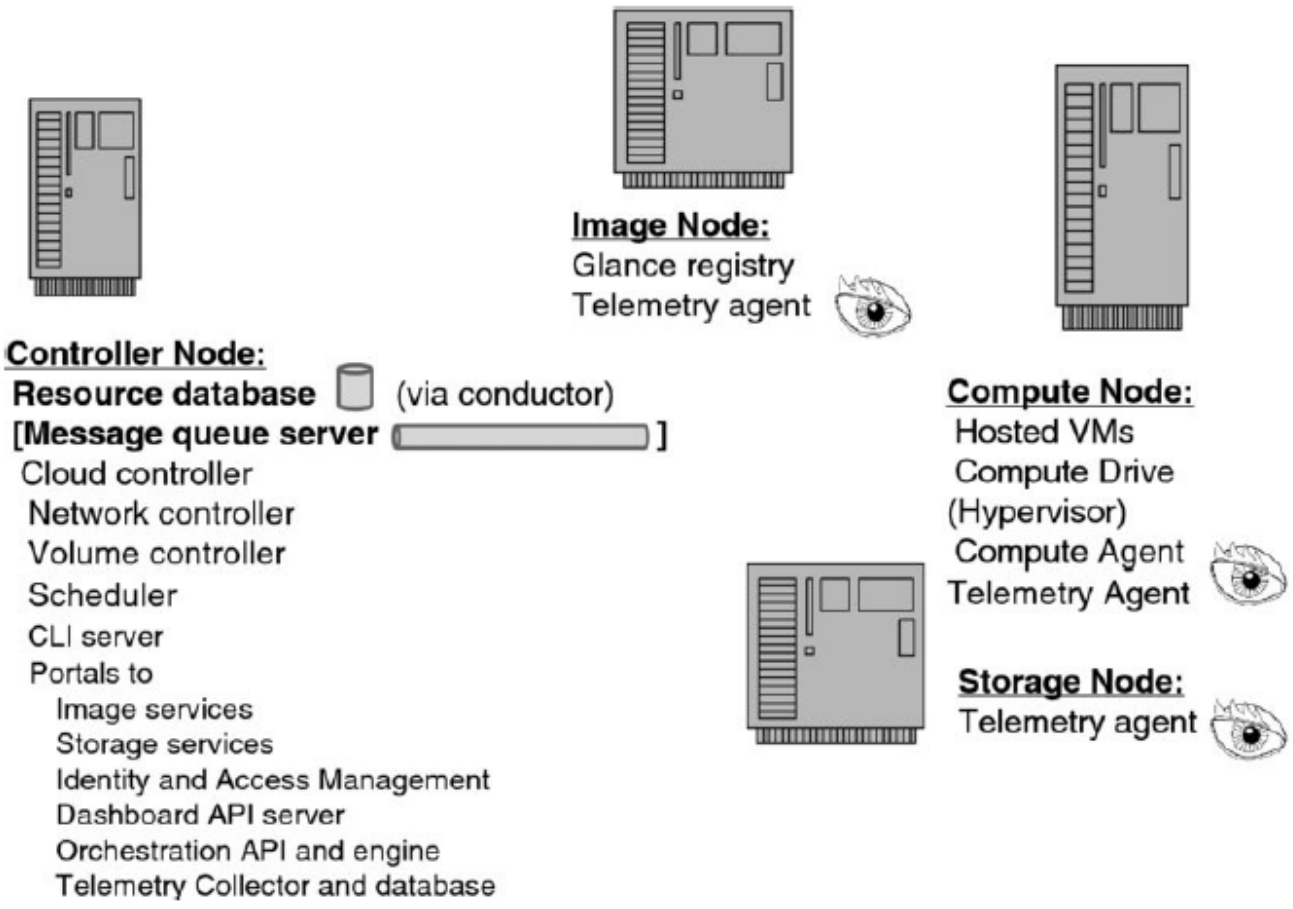
(Reference: Cloud Computing: Business Trends and Technologies)

Homework 6.8 –

What is the name of the protocol used for communications among the *OpenStack* daemons?

Solution 6.8 –

All OpenStack modules that have "API" in their names (i.e., nova-api) are daemons providing REST services (discussed in the Appendix). Communications among daemons are carried out via the Advanced Message Queuing Protocol (AMQP). AMQP can be initiated from either end of the pipe. On the contrary, an HTTP transaction can be initiated only by the client because HTTP is a pure client/server protocol.



(Reference: Cloud Computing: Business Trends and Technologies)