Homework 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 specify 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:

   i. While a device may have to change in response to users' requests, it is hardly acceptable to allocate network resources based only on user requests—that is, always give whatever one asks.
   ii. Network providers wanted to have a mechanism that would enable granting a resource based on a set of policy rules.
   iii. The decision on whether to grant the resource considers information about the user, the requested service, and the network itself. There was a need for policy management.
   iv. Employing SNMP for this purpose was not straightforward, and so the IETF developed a new protocol, Common Open Policy Service (COPS) for communications between the network element and the Policy Decision Point (PDP)— where the policy-based decisions were made.

   Goals of COPS framework:

   i. The objective of the framework is to describe the execution of policy-based control over the QoS admission control decisions, with the primary focus on the RSVP protocol as an example.
   ii. Among the goals of the framework are support for pre-emption, various policy styles, monitoring, and accounting. Pre-emption here means the ability to remove a previously granted resource to accommodate a new request.
   iii. As far as the policy styles go, those to be supported include bi-lateral and multi-lateral service agreements and policies based on the notion of relative priority, as defined by the provider.
   iv. Support for monitoring and accounting is affected by gathering the resource use and access data. The framework also states the requirement for fault tolerance and recovery in cases when a PDP fails or cannot be reached.

   Source: [Cloud Computing: Business Trends and Technologies]

2. Some of the features which are intrinsically new to COPS as compared to SNMP are:

   • 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.

- The COPS stateful model supports two mechanisms of policy control, called respectively the outsourcing model and the configuration model. With the outsourcing mechanism, PEP queries PDP every time it needs a decision; when the configuration mechanism is employed, PDP provisions the policy decision within the PEP.

- 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.

Source: [Cloud Computing: Business Trends and Technologies]

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 because:

   - Standardized MIB modules often lack writable MIB objects which can be used for configuration, and this leads to a situation where the interesting writable objects exist in proprietary MIB modules.

   - There are scaling problems with regard to the number of objects in a device. While SNMP provides reasonable performance for the retrieval of a small amount of data from many devices, it becomes rather slow when retrieving large amounts of data (such as routing tables) from a few devices.

   - There is too little deployment of writable MIB modules. While there are some notable exceptions in areas, such as cable modems where writable MIB modules are essential, it appears that router equipment is usually not fully configurable via SNMP.

• 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.

• There is often a semantic mismatch between the task-oriented view of the world usually preferred by operators and the data-centric view of the world provided by SNMP. Mapping from a task-oriented view to the data-centric view often requires some non-trivial code on the management application side.

• Several standardized MIB modules lack a description of high-level procedures. It is often not obvious from reading the MIB modules how certain high-level tasks are accomplished, which leads to several different ways to achieve the same goal, which increases costs and hinders interoperability.

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 operator requirements for network management are:
• 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 that describes 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.
• 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 to determine the changes between two configurations and whether those configurations are consistent.
• Support for configuration transactions across several devices would significantly simplify network configuration management.
• 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

rather than individual devices.
- 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 interfaces 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.

Source: [https://tools.ietf.org/html/rfc3535#section-2.1]

4. The Messages layer is merely a transport-independent framing mechanism for encoding both RPC-related and notifications-related structures. NETCONF does not use REST API in its Messages layer. The model is RPC.
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.

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

5. YANG is the de-facto NETCONF modeling language. It is well structured, so following a module one can find both its high-level view and the ultimate encoding in NETCONF

operations. By design, YANG has also been made extensible, thus allowing other SDOs to develop its extensions and individual programmers to produce plug-and-play modules. "YANG modules can be translated into an equivalent XML syntax called YANG Independent Notation (YIN), allowing applications using XML parsers and Extensible Stylesheet Language Transformations (XSLT) scripts to operate on the models. The conversion from YANG to YIN is lossless, so content in YIN can be round-tripped back into YANG." It is specified in www.ietf.org/rfc/rfc6020.txt.

Source: [Cloud Computing: Business Trends and Technologies]

6. In the context of migration to a cloud environment, 'onboarding' refers to the deployment of applications, data or both to the chosen cloud infrastructure (public, private or hybrid). It's essentially the final stage of the migration process: the equivalent of the transition from one network to another in a network migration project.
   The actual process of onboarding ('forklifting' workloads) has seven relatively straightforward steps:

i. Defining 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.
ii. Provisioning cloud resources: Service providers will have a self-service interface for the creation of accounts and purchase of the services that you need (e.g., servers, storage, network).
iii. Establishing 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.
iv. Deploying 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.
v. Ensuring 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.
vi. Testing and validating: However well you've prepared and tested prior to deployment, there may be surprises. Has everything been transferred correctly? Do network, storage, compute and database configurations remain intact? Can you see and manage the cloud environment properly? Does your cloud backup process work?
vii. Discontinuing the old service: When you're certain that everything is working well, you can give access to users and decommission the enterprise service.

Source: [https://www.interxion.com/sites/default/files/2020-01/A%20Practical%20Guide%20to%20Cloud%20Onboarding.pdf]

7. The three entities involved in the service life cycle 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 a contract, which contains, among other items, the SLA and a set of specific, measurable aspects of the SLA called Service-Level Objectives (SLOs).

   Source: [Cloud Computing Business Trends and Technologies]

8. All OpenStack modules that have “API” in their names (i.e., nova-api) are daemons providing REST services. Communications among daemons are carried out via the Advanced Message Queuing Protocol (AMQP). The message queue of the figure below is the structure that enables this. AMQP can be initiated from either end of the pipe.

   Source: [Cloud Computing Business Trends and Technologies]