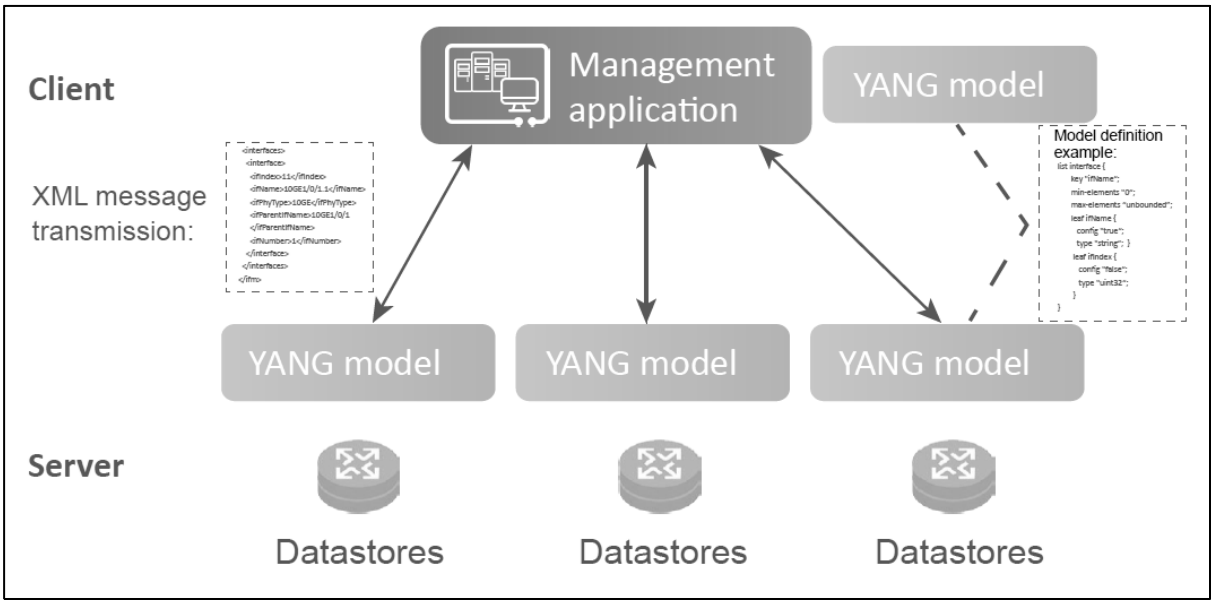
Name: Kazi Jawwad A Rahim *Roll No: 28*

Case Study – IoT System Management with NETCONF YANG

What are NETCONF and YANG:

Network Configuration Protocol (NETCONF) is a network device management protocol that is similar to SNMP. NETCONF provides a framework for users to add, modify, or delete network device configurations, or query configurations, status, and statistics. Similar to SNMP that uses MIB files to model data, NETCONF uses Yet Another Next Generation (YANG) as a data modeling language to describe the interaction models between the NETCONF client and server.



```
<interfaces>
  <interface>
    <ifIndex>11</ifIndex>
    <ifName>10GE1/0/1.1</ifName>
    <ifPhyType>10GE</ifPhyType>
    <ifParentIfName>10GE1/0/1
    </ifParentIfName>
    <ifNumber>1</ifNumber>
  </interface>
 </interfaces>
</ifm>
```

```
list interface {
    key "ifName";
    min-elements "0";
    max-elements "unbounded";
    leaf ifName {
      config "true";
      type "string"; }
    leaf ifIndex {
      config "false";
      type "uint32";
    }
}
```

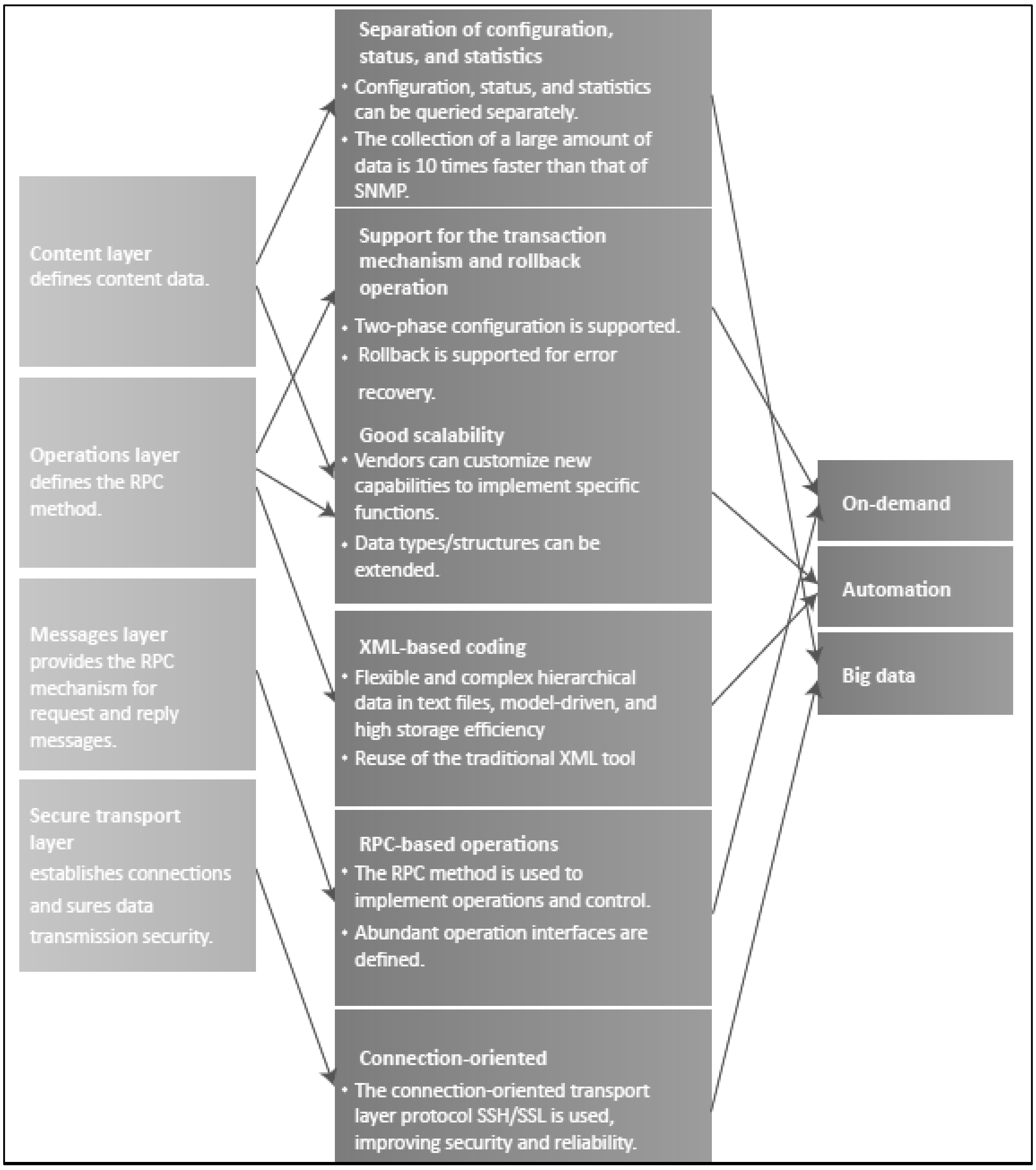
Why are NETCONF and YANG required:

One of the key network requirements of the cloud era is network automation, which includes quick, automatic, and on-demand service provisioning and automatic O&M. The traditional CLI mode and SNMP do not meet the requirements of cloud-based networks due to the following disadvantage:

- The traditional CLI mode is based on man-machine interfaces. The configuration is complex and varies with vendors. Therefore, the manual learning cost is high.
- The SNMP configuration efficiency is low, and the transaction mechanism is not supported. Therefore, SNMP is often used for monitoring.

Name: Kazi Jawwad A Rahim *Roll No: 28*

1. **NETCONF uses a hierarchical protocol framework, better meeting the on-demand, automatic, and big data requirements of cloud-based networks.**

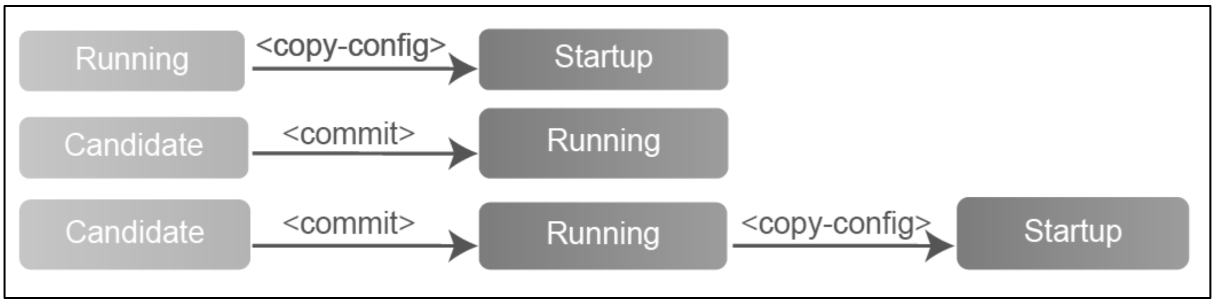

Content layer defines content data.
Operations layer defines the RPC method.
Messages layer provides the RPC mechanism for request and reply messages.
Secure transport layer establishes connections and sures data transmission security.
Separation of configuration, status, and statistics
• Configuration, status, and statistics can be queried separately.
• The collection of a large amount of data is 10 times faster than that of SNMP.
Support for the transaction mechanism and rollback operation
• Two-phase configuration is supported.
• Rollback is supported for error recovery.
Good scalability
• Vendors can customize new capabilities to implement specific functions.
• Data types/structures can be extended.
XML-based coding
• Flexible and complex hierarchical data in text files, model-driven, and high storage efficiency
• Reuse of the traditional XML tool
RPC-based operations
• The RPC method is used to implement operations and control.
• Abundant operation interfaces are defined.
Connection-oriented
• The connection-oriented transport layer protocol SSH/SSL is used, improving security and reliability.
On-demand
Automation
Big data

2. **NETCONF supports classified data storage and migration, phase-based committing, and configuration isolation.**

<running/>: <running/> configuration datastore that stores the complete set of active configurations of a network device.
<candidate/>: <candidate/> configuration datastore that stores various configuration data to be committed to the <running/> configuration datastore. Changes in the <candidate/> configuration datastore do not directly affect the involved device.

<startup/>: <startup/> configuration datastore that stores configuration data loaded (similar to a saved configuration file) during device startup.

Data can be migrated between configuration database.



3. **NETCONF defines abundant operation interfaces and supports extension based on capability sets.**

Basic Operations Supported by NETCONF (RFC 6241)

<get>: obtains part or all of the running configuration data and status data from the <running/> configuration datastore.

<get-config>: obtains configuration data.

<edit-config>: creates, modifies, or deletes configuration data.

<copy-config>: replaces a configuration datastore with the contents of another complete configuration datastore.

<delete-config>: deletes all data in a non-running configuration datastore.

<lock>: locks the configuration datastore of a device. A locked configuration datastore cannot be modified by other NETCONF users.

<unlock>: unlocks the configuration datastore of a device.

<close-session>: terminates a NETCONF session gracefully.

<kill-session>: forcibly terminates another NETCONF session.

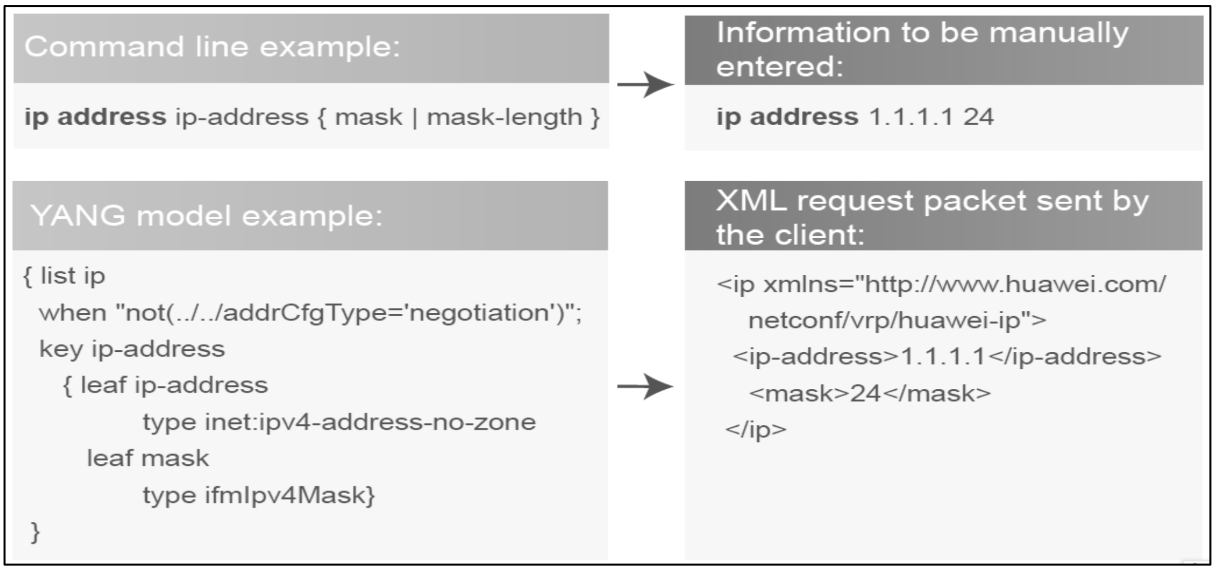
Capabilities that can be extended:

RFC 6241:	Writable-Running, Candidate Configuration, Confirmed Commit, Rollback-on-Error, Validate, Startup, URL and XPath.
RFC 5277:	Notification and Interleave.
RFC 6243:	with-defaults.
RFC 6022:	IIetf-netconf-monitoring

4. NETCONF operations are performed based on the YANG model.

The YANG model defines device function configuration templates. Compared with the CLI mode, the YANG model has the following advantages:

- **Rich definition:** Various basic data types and data attributes can be defined.
- **Machine language:** Structured definition is adopted, allowing users to define constraints that can be directly identified by the computer without manual intervention.
- **Good scalability:** The grouping, refine, augment, and typedef statements are supported for users to extend YANG models and data types.
- **Easy integration:** The IETF defines multiple YANG models and data types for reference.



Command line example:

```
ip address ip-address { mask | mask-length }
```

Information to be manually entered:

```
ip address 1.1.1.1 24
```

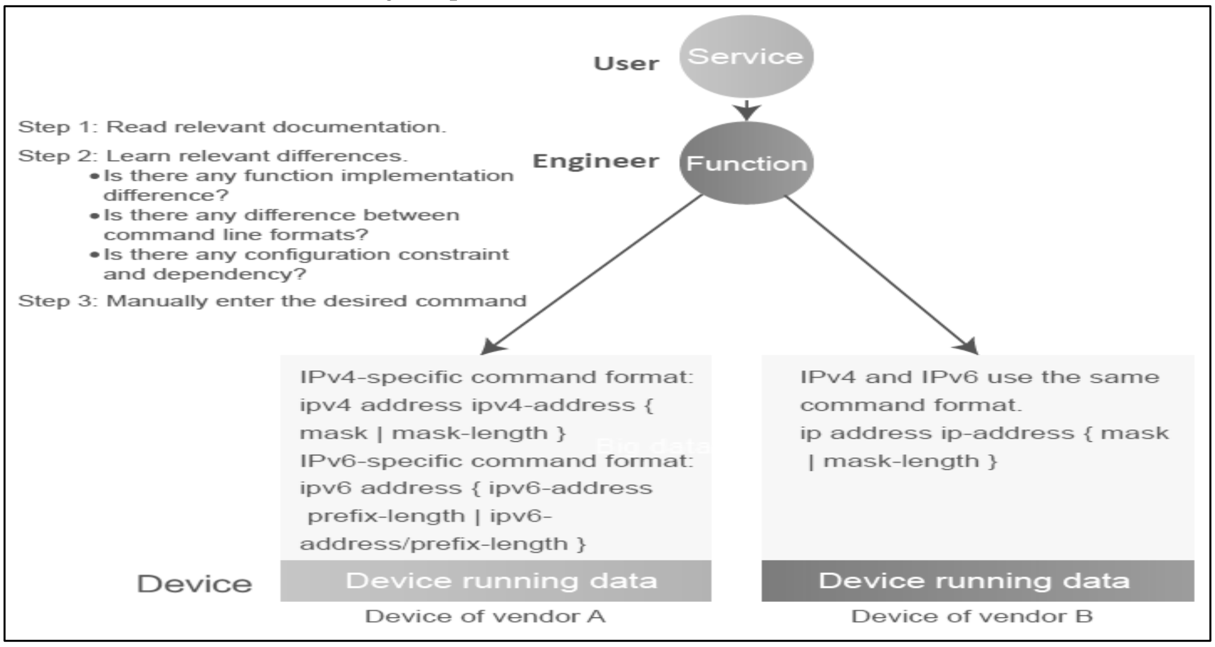
YANG model example:

```
{ list ip
  when "not(../../addrCfgType='negotiation')";
  key ip-address
    { leaf ip-address
            type inet:ipv4-address-no-zone
      leaf mask
            type ifmIpv4Mask}
}
```

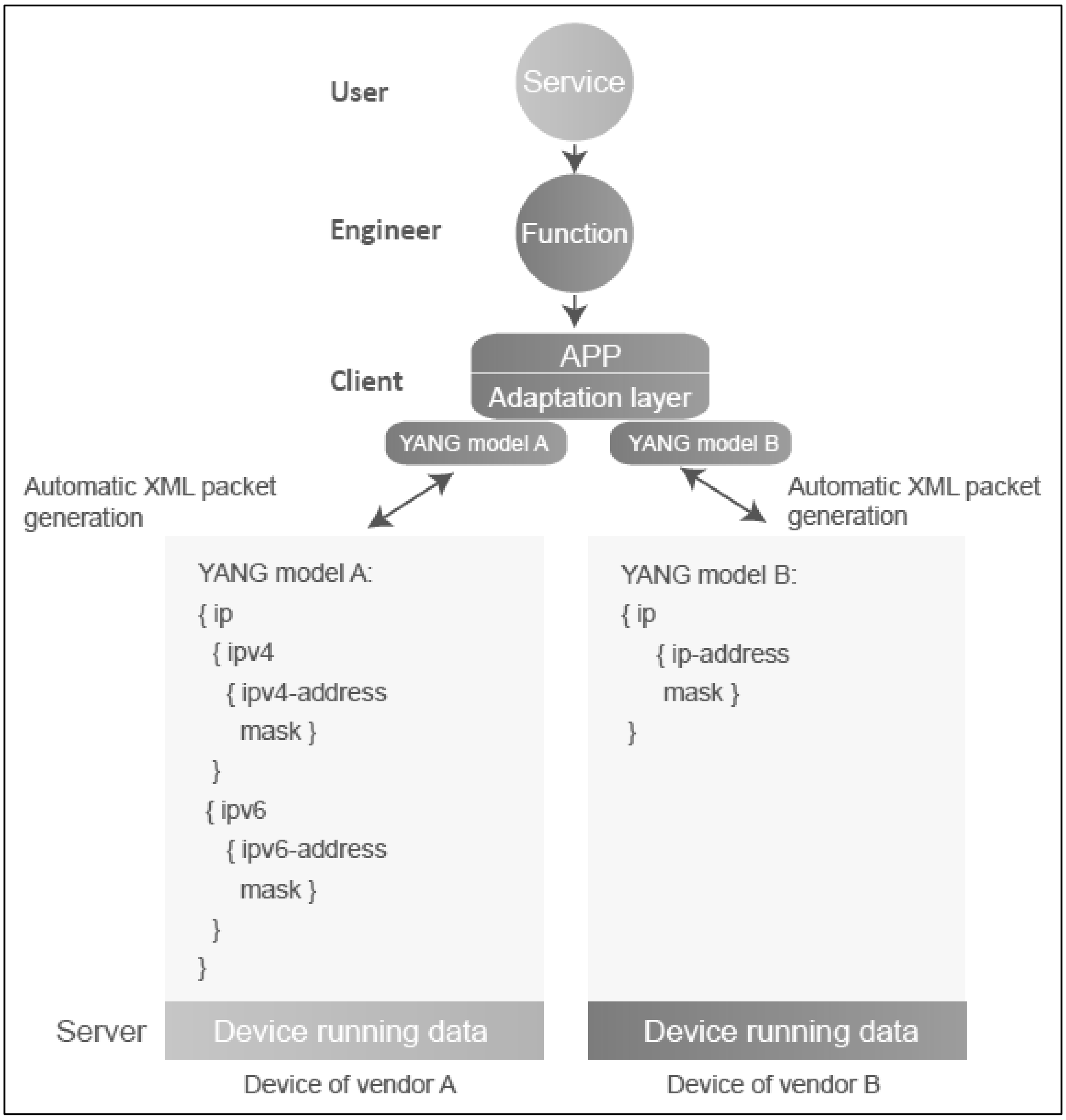
XML request packet sent by the client:

```
<ip xmlns="http://www.huawei.com/
    netconf/vrp/huawei-ip">
  <ip-address>1.1.1.1</ip-address>
    <mask>24</mask>
 </ip>
```

If functions are implemented in different ways on devices from different vendors:
In traditional CLI mode, engineers need to learn the differences between command lines of different vendors and manually map the command lines.


User
Service
Engineer
Function
Step 1: Read relevant documentation.
Step 2: Learn relevant differences.
•Is there any function implementation difference?
•Is there any difference between command line formats?
•Is there any configuration constraint and dependency?
Step 3: Manually enter the desired command
IPv4-specific command format: ipv4 address ipv4-address { mask | mask-length }
IPv6-specific command format: ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length }
IPv4 and IPv6 use the same command format. ip address ip-address { mask | mask-length }
Device
Device running data
Device of vendor A
Device running data
Device of vendor B

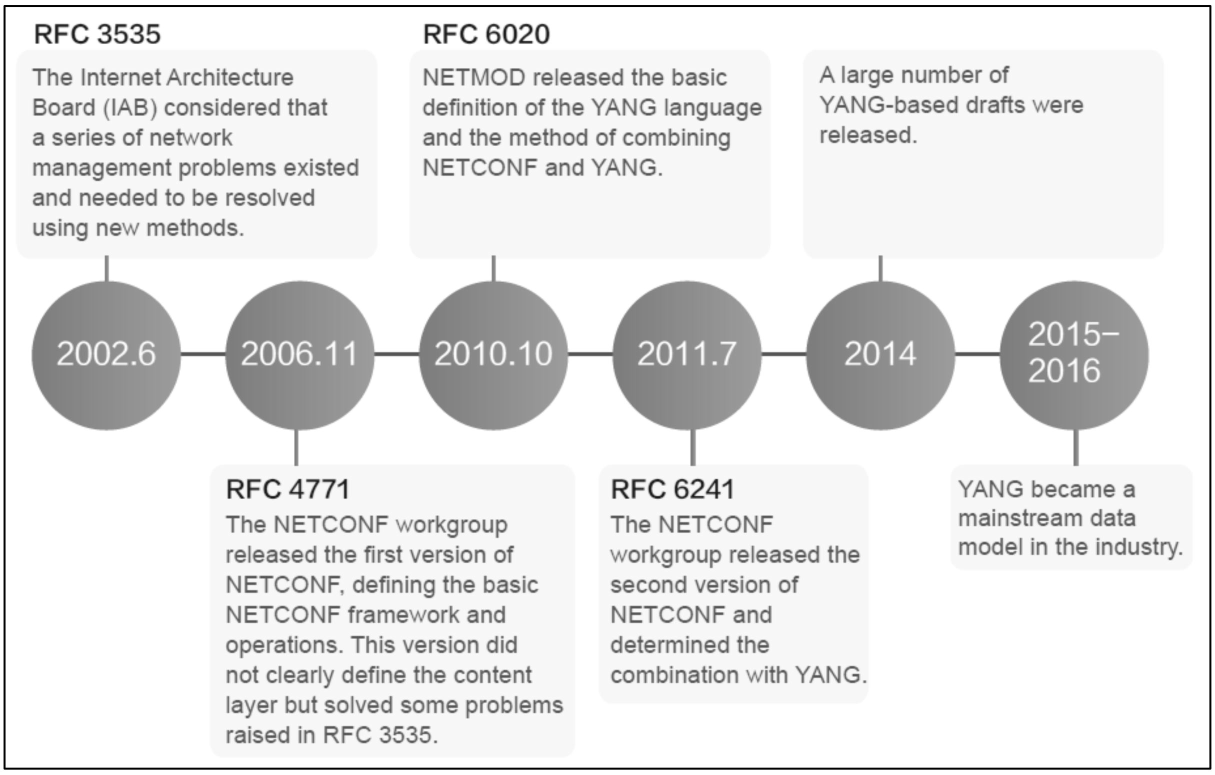
If functions are implemented based on YANG models, configuration engineers do not need to pay attention to the definitions of YANG models and the differences between the YANG models. The corresponding application automatically parses the YANG model data, shifting the focus of engineers from device and function differences to user requirements. The engineers can achieve automatic configuration only by operating the graphical application.



NETCONF and YANG Development:

YANG drives the development of NETCONF.

- Universal models are defined based on YANG, making a breakthrough in NETCONF implementation and laying the foundation for model unification between vendors.
- YANG-based network models are implemented on devices of different vendors, promoting NETCONF development.


RFC 3535
The Internet Architecture Board (IAB) considered that a series of network management problems existed and needed to be resolved using new methods.
RFC 6020
NETMOD released the basic definition of the YANG language and the method of combining NETCONF and YANG.
A large number of YANG-based drafts were released.
2002.6
2006.11
2010.10
2011.7
2014
2015–2016
RFC 4771
The NETCONF workgroup released the first version of NETCONF, defining the basic NETCONF framework and operations. This version did not clearly define the content layer but solved some problems raised in RFC 3535.
RFC 6241
The NETCONF workgroup released the second version of NETCONF and determined the combination with YANG.
YANG became a mainstream data model in the industry.

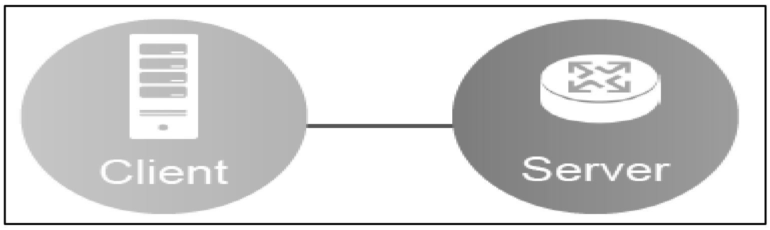
Example for the basic NETCONF session process:

Scenario:

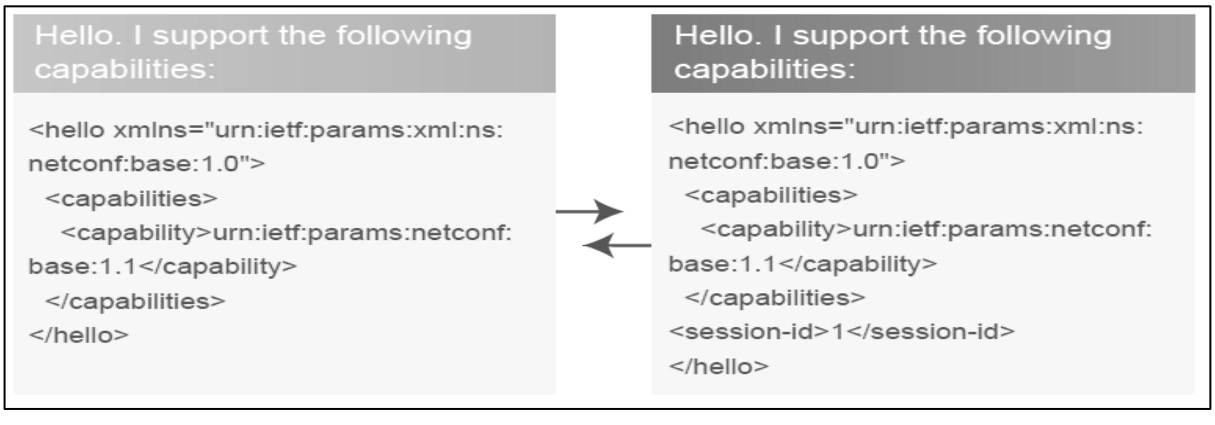
Use a NETCONF client to change the IP address of a device in two-phase validation mode.

Prerequisites:

The SSH connection, authentication, and authorization are complete.



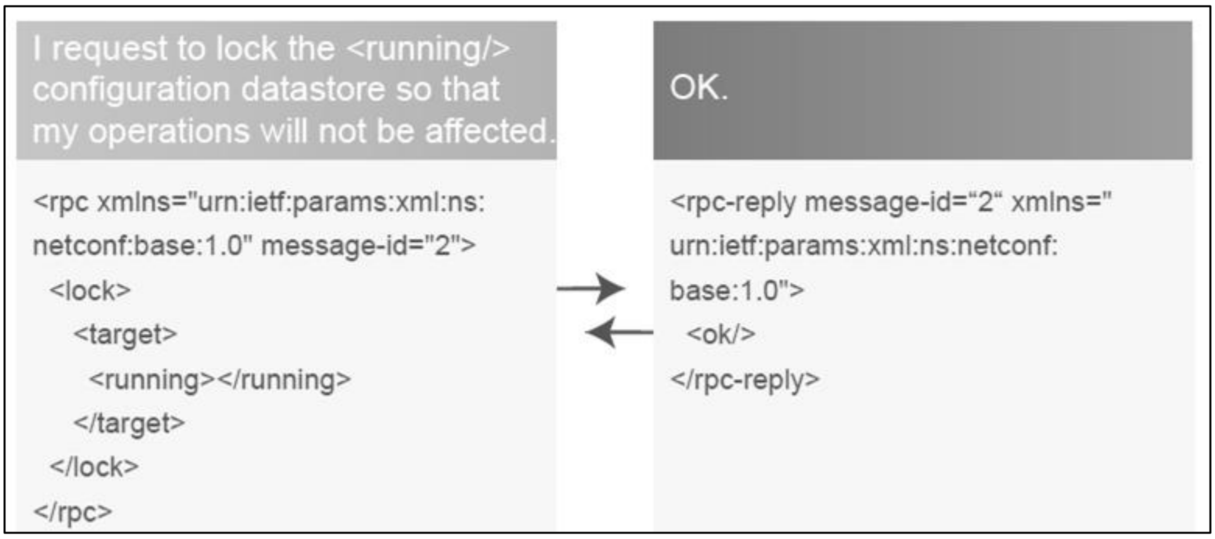
Step 1: Establish a NETCONF session and advertise capabilities through <hello> messages.


Hello. I support the following capabilities:
Hello. I support the following capabilities:

```
<hello xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:
base:1.1</capability>
  </capabilities>
</hello>
```

```
<hello xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:
base:1.1</capability>
  </capabilities>
<session-id>1</session-id>
</hello>
```

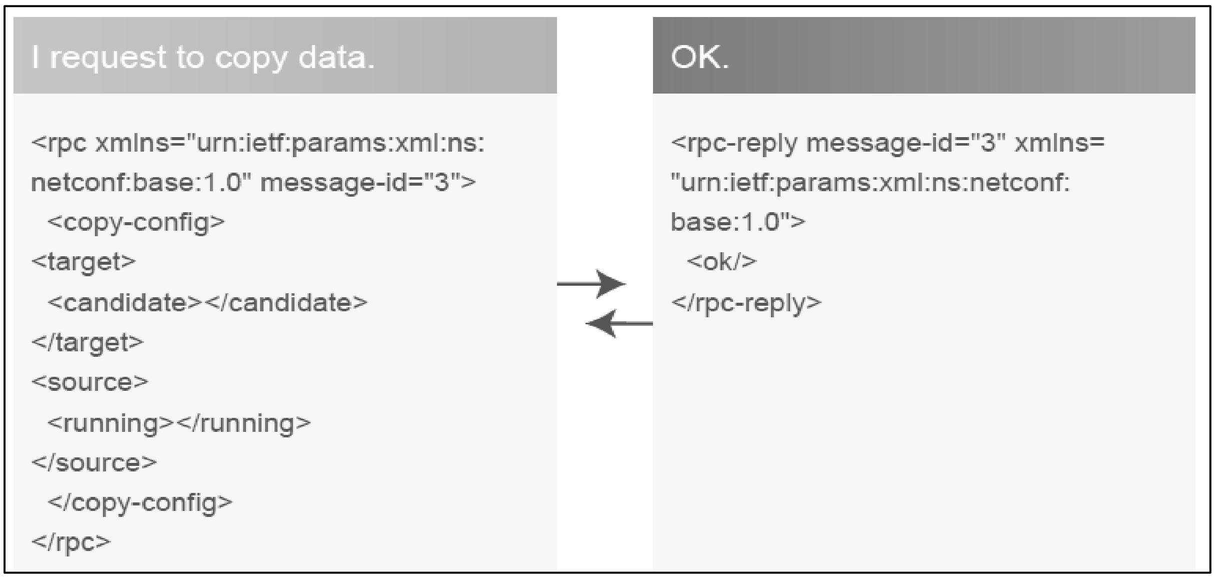
Step 2: Lock the <running/> configuration datastore to avoid conflicts with other clients.


I request to lock the <running/> configuration datastore so that my operations will not be affected.
OK.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="2">
  <lock>
    <target>
      <running></running>
    </target>
  </lock>
</rpc>
```

```
<rpc-reply message-id="2" xmlns="
urn:ietf:params:xml:ns:netconf:
base:1.0">
  <ok/>
</rpc-reply>
```

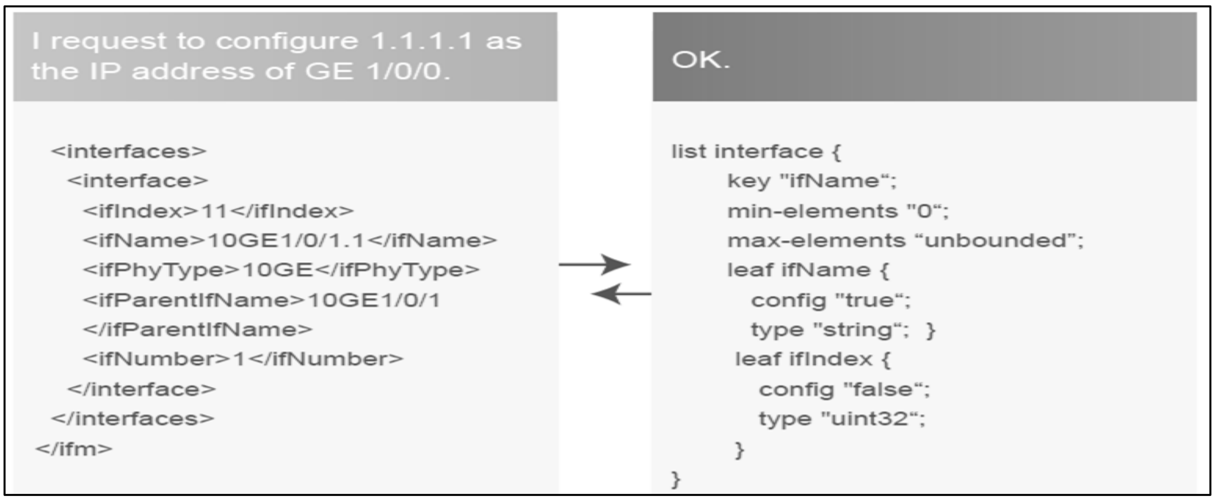
Step 3: Copy the data in the <running/> configuration datastore to the <candidate/> configuration datastore to ensure that the configurations are the latest.



```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="3">
  <copy-config>
<target>
  <candidate></candidate>
</target>
<source>
  <running></running>
</source>
  </copy-config>
</rpc>
```

```
<rpc-reply message-id="3" xmlns=
"urn:ietf:params:xml:ns:netconf:
base:1.0">
  <ok/>
</rpc-reply>
```

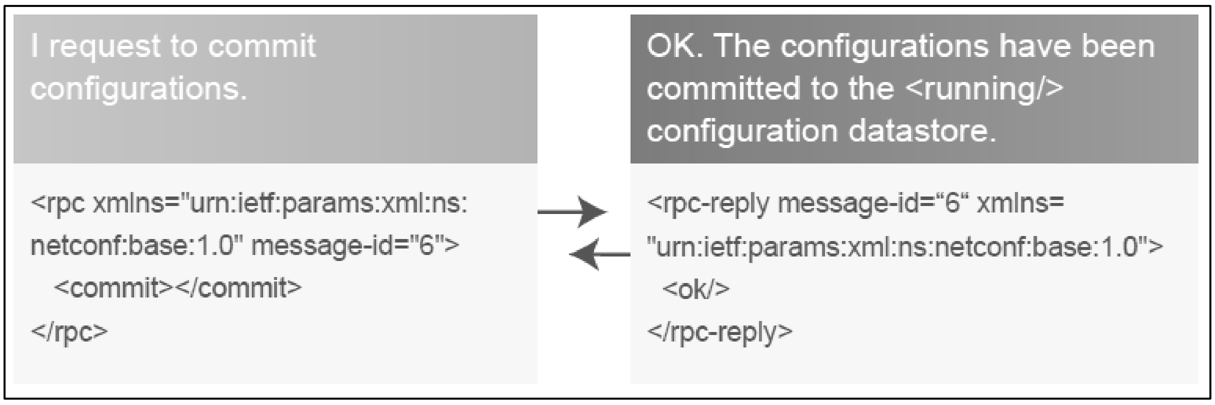
Step 4: Edit configurations in the <candidate/> configuration datastore.


I request to configure 1.1.1.1 as the IP address of GE 1/0/0.
OK.

```
  <interfaces>
    <interface>
      <ifIndex>11</ifIndex>
      <ifName>10GE1/0/1.1</ifName>
      <ifPhyType>10GE</ifPhyType>
      <ifParentIfName>10GE1/0/1
      </ifParentIfName>
      <ifNumber>1</ifNumber>
    </interface>
  </interfaces>
</ifm>
```

```
list interface {
      key "ifName";
      min-elements "0";
      max-elements "unbounded";
      leaf ifName {
        config "true";
        type "string";  }
       leaf ifIndex {
         config "false";
         type "uint32";
       }
}
```

Step 5: Commit configurations in the <candidate/> configuration datastore to the <running/> configuration datastore.

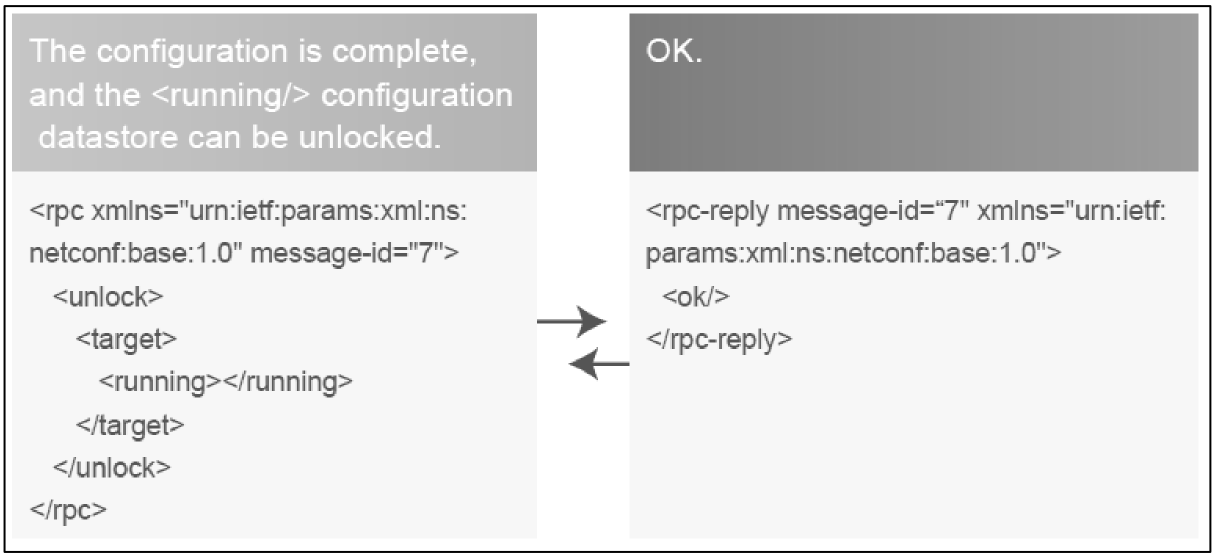

I request to commit configurations.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="6">
  <commit></commit>
</rpc>
```

OK. The configurations have been committed to the <running/> configuration datastore.

```
<rpc-reply message-id="6" xmlns=
"urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Step 6: Unlock the <running/> configuration datastore.


The configuration is complete, and the <running/> configuration datastore can be unlocked.

```
<rpc xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" message-id="7">
  <unlock>
    <target>
      <running></running>
    </target>
  </unlock>
</rpc>
```

OK.

```
<rpc-reply message-id="7" xmlns="urn:ietf:
params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Follow-up procedure: Terminate the NETCONF session and tear down the SSH connection.